

TABLE OF CONTENTS

LONWORKS INTERFACE MODULE

OpenNet Interface Modules	1
LONWORKS Interface Module Features	1
About LON	1
LONWORKS Network Components	2
LONWORKS Network System Setup	3
LONWORKS Interface Module Parts Description	4
LONWORKS Interface Module Specifications	5
Wiring LONWORKS Interface Module	6
Terminator	7
Link Registers for LONWORKS Network Communication	8
Transmission Time	9
Function Area Setting for LONWORKS Node	10
Programming Transmit/Receive Data Using WindLDR	11
Starting Operation	12
Network Management	12
Precautions for Modifying Application Program	13
LONWORKS Interface Module Internal Structure	14
Data Exchange between LONWORKS Interface Module and CPU Module	16
Application Program Examples	18
Initialization	18
Writing Receive Data to Data Registers in the LONWORKS Interface Module	21
Reading Transmit Data from Data Registers in the LONWORKS Interface Module	22
Defined Network Variables	23
LONWORKS Network Troubleshooting	25
Troubleshooting Diagram 1	25
Troubleshooting Diagram 2	26
Troubleshooting Diagram 3	27
Troubleshooting Diagram 4	28
Troubleshooting Diagram 5	28
Troubleshooting Diagram 6	28
Index	29

LONWORKS INTERFACE MODULE

Introduction

This manual describes LONWORKS interface module FC3A-SX5LS1 used with the OpenNet Controller™ to interface with the LONWORKS network, and provides details on the LONWORKS system setup and the LONWORKS interface module specifications.

For general information about safety precautions, installation, wiring, and dimensions, see the OpenNet Controller user's manual EM333.

OpenNet Interface Modules

The OpenNet Controller can be linked to three major open networks; INTERBUS, DeviceNet™, and LONWORKS®. For communication through these networks, OpenNet interface modules are available. Mounting the LONWORKS interface module beside the OpenNet Controller CPU module makes up a node on a LONWORKS network. The node can communicate I/O data with other nodes in a distributed network.

LONWORKS Interface Module Features

The LONWORKS interface module conforms to the specifications of LONWORKS that is recognized worldwide as a de facto industry standard open network, so the OpenNet Controller can be linked to the LONWORKS networks consisting of LONWORKS compliant products manufactured by many different vendors, such as I/O terminals, sensors, drives, operator interfaces, and barcode readers. The flexible, configurable, and interoperable features of the LONWORKS network make it possible to build, expand, or modify production lines with reduced cost.

The transmit/receive data quantity can be selected from 0 through 8 bytes (64 bits) in 1-byte increments. One LONWORKS interface module enables the OpenNet Controller CPU module to transmit 64 bits and receive 64 bits at the maximum to and from the LONWORKS network.

The network can be configured either in bus or free topology. The total transmission distance can be 1,400m in bus topology and 500m in free topology. The free topology makes it possible to configure a flexible network.

About LON

The LON® (Local Operating Network) technology is a network control system developed by Echelon, USA. The LON technology is an intelligent, distributed network for communication with various sensors and actuators at a maximum of 32,385 nodes.

LONWORKS is the open control standard for buildings, factories, houses, and transportation systems. Now, LONWORKS networks are widely used in major building automation (BA), process automation (PA), and many other industries in the world.

Communication between application programs installed in LonWorks compliant nodes is performed using the LonTalk protocol based on the reference model of the Open System Interconnection (OSI) issued by the International Standard Organization (ISO).

OpenNet Controller and WindLDR are trademarks of IDEC CORPORATION.

LON, LONWORKS, LonBuilder, Echelon, Neuron, LonTalk, and 3150 are registered trademarks of Echelon Corporation registered in the United States and other countries. LonMaker is a trademark of Echelon Corporation.

DeviceNet is a trademark of Open DeviceNet Vendor Association, Inc. (ODVA).

LONWORKS Network Components

Physical Layer — Transceiver

The LONWORKS interface module incorporates an FTT-10A (Free Topology Twisted Pair Transceiver) for the physical layer. The FTT-10A transceiver is a transformer-isolated type and has the following specifications:

Name	Communication Media	Transmission Rate	Transmission Distance	Topology
FTT-10A Transceiver	Twisted pair cable	78 kbps	500m (maximum total wire length) 400m (maximum node-to-node distance)	Free
			1,150m	Bus

Note: The transmission distance is the value when Level 4 AWG22 cables and proper terminators are used.

LonTalk Protocol

The LonTalk protocol has all seven layers in compliance with the reference model of the Open System Interconnection (OSI) issued by the International Standard Organization (ISO).

Neuron Chip

Some special LSI Neuron Chips that support the LonTalk protocol have firmware embedded in the built-in memory. The Neuron Chip used in the LONWORKS interface module is Toshiba TMP3150B1AF, with firmware embedded in the external memory (flash memory). This Neuron Chip uses a 10MHz quartz clock oscillator. The Neuron Chip and peripheral circuit are powered through the CPU bus.

Application Program

The application program for the LONWORKS interface module is in compliance with the application layer of the OSI reference model, and is described in Neuron C that is derived from ANSI C.

Communication data is transferred through the registers located between the OpenNet Controller CPU bus and the Neuron Chip external memory expansion bus. An application program including access to the registers is created and embedded in the external memory (flash memory) along with firmware by IDEC before shipment. Users do not have to create and install application programs, although programmers familiar with Neuron C can also create or modify the application program using a special tool, such as LonBuilder Developer's Kit. When a user creates or modifies the application program, the user must keep a backup file. For application program examples, see pages 18 through 22.

Network Variables

The LonTalk protocol allocates communication data to network variables (NV) specifically designed to simplify the procedures for packet transmission. The variables are available in input network variables and output network variables. The values of output network variables are transmitted to input network variables of the target node on the network. Details are described on pages 9 and 23.

Network Management

When setting up a LONWORKS network system, the user has to install network configuration information shown below.

- Addressing: Determines each node address
- Binding: Determines target nodes to communicate with
- Configuration: Determines the type of message service, retry cycles, timeout period, etc.

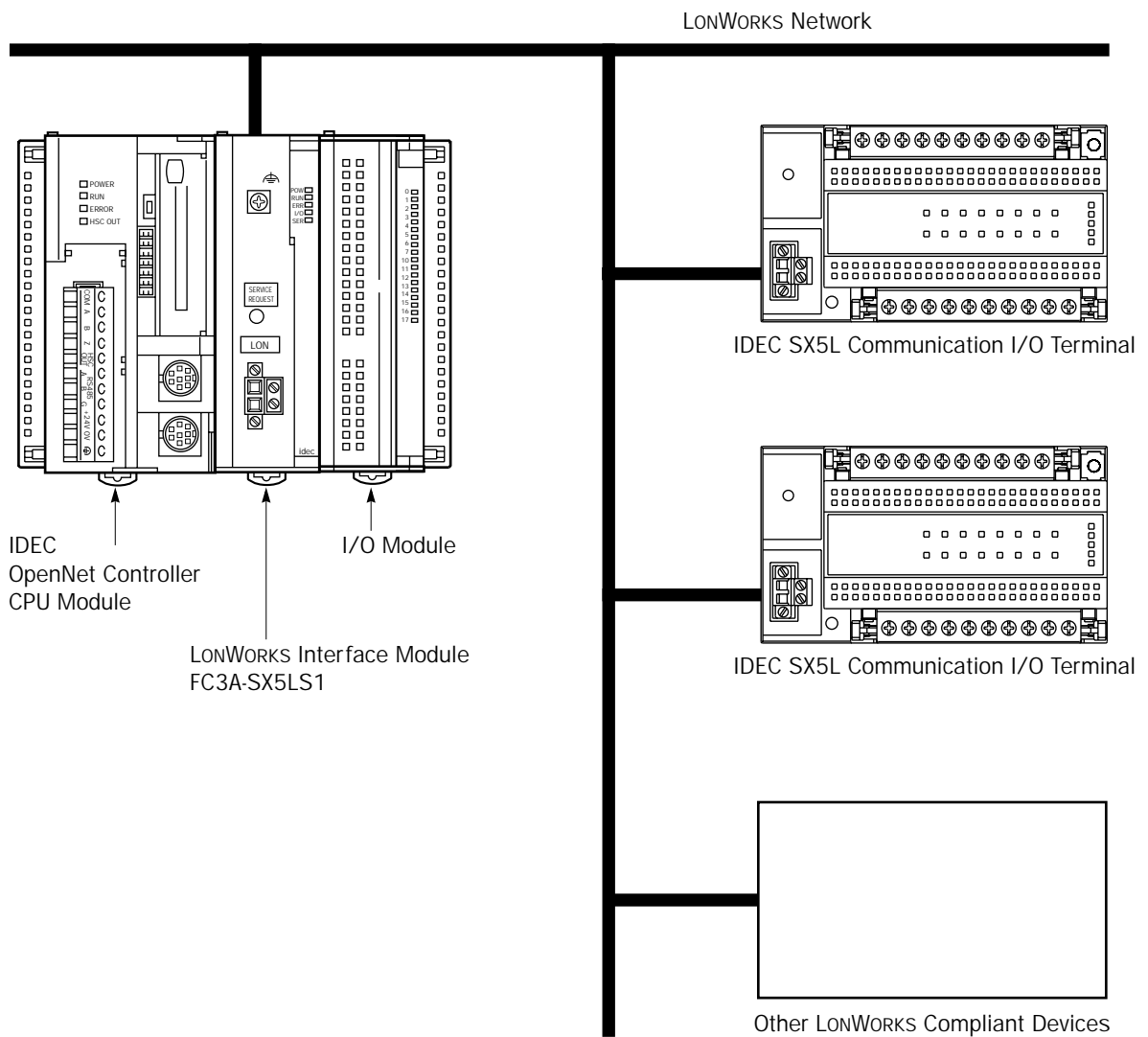
Use a network management tool from other manufacturers (such as LonMaker for Windows Integration Tool) to install network configuration information. An external interface file (XIF extension) unique to each product series is needed to install the network configuration information. The external interface file for the LONWORKS interface module is available from IDEC. The user must keep a backup file of the information used for network management.

LONWORKS Network System Setup

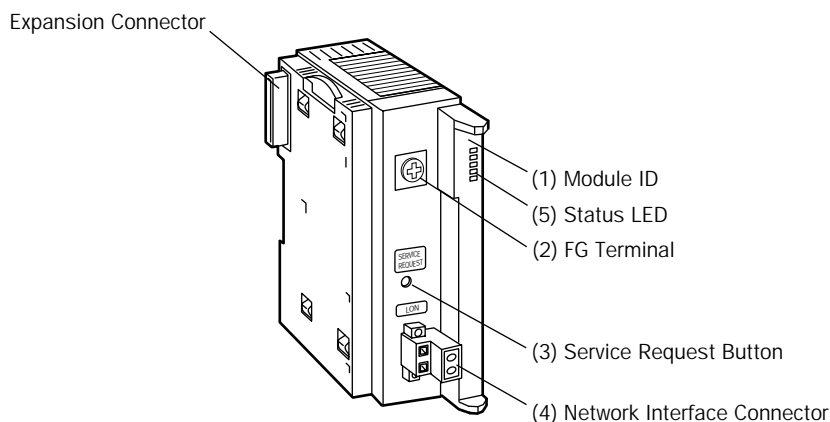
Various LONWORKS compliant devices, such as the LONWORKS interface module and IDEC SX5L communication I/O terminals, can be connected to the LONWORKS network.

The OpenNet Controller can be used as a node by adding the LONWORKS interface module to the right of the OpenNet Controller CPU module.

A maximum of seven OpenNet interface modules and analog I/O modules can be mounted with one OpenNet Controller CPU module.



LONWORKS Interface Module Parts Description



OpenNet Interface Module for LONWORKS Network

Module Name	LONWORKS Interface Module
Type No.	FC3A-SX5LS1

- (1) **Module ID** FC3A-SX5LS1 indicates the LONWORKS interface module ID.
- (2) **FG Terminal** Frame ground terminal
- (3) **Service Request Button** Pushbutton used for network management
- (4) **Network Interface Connector** For connecting the LONWORKS communication cable
- (5) **Status LED** Indicates operating status

Indicator	Status		Description
POW (POWER)	—	OFF	Module power OFF
	Green	ON	Module power ON
RUN	Green	ON	Normal operation
	—	OFF	Normal operation
ERR (COM_ERROR)	Red	ON	Communication error
	—	OFF	Normal operation
I/O (I/O_ERROR)	Red	ON	Access error to the CPU through I/O bus
	—	OFF	Normal operation
SER (SERVICE)	Yellow	ON	Application program not configured
		Flash	Network management not configured

LONWORKS Interface Module Specifications

Normal Operating Conditions

Operating Ambient Temperature	0 to +55°C (no freezing)
Storage Temperature	-25 to +70°C (no freezing)
Operating Humidity	Level RH1 30 to 90% (no condensation)
Pollution Degree	2 (IEC 60664)
Corrosion Immunity	Free from corrosive gases
Altitude	Operation: 0 to 2000m Transportation: 0 to 3000m
Vibration Resistance	10 to 57 Hz, amplitude 0.075 mm; 57 to 150 Hz, acceleration 9.8 m/sec ² (1G); 10 sweep cycles each in 3 axes (total 80 minutes) (IEC1131)
Shock Resistance	147 m/sec ² (15G), 11 msec, 3 shocks each in 3 axes (IEC1131)

Power Supply (supplied from the OpenNet Controller CPU module)

Dielectric Strength	Between power terminal on CPU module and FG: 500V AC, 1 minute
Insulation Resistance	Between power terminal on CPU module and FG: 10 MΩ (500V DC megger)
Current Draw	Approx. 30 mA

Grounding

Ground Terminal	M3 sems
Grounding Resistance	100Ω maximum
Grounding Wire	UL1015 AWG22, UL1007 AWG18

Weight

Weight	Approx. 180g
--------	--------------

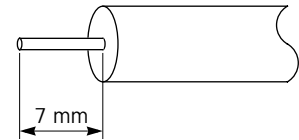
Communication Specifications

Communication System	LON [®] system
Transceiver	FTT-10A (Free Topology Twisted Pair Transceiver made by Echelon)
Transmission Rate	78 kbps
Transmission Distance (when using Level 4 AWG22 cables)	Free topology: Total 500m (400m maximum between nodes) Bus topology: 1,150m (when using FTT-10A transceivers only)
Maximum Nodes	32,385 nodes in a network
Network Interface Connector	In the module: MSTB2.5/2-GF-5.08 (made by Phoenix Contact) To the cable: FRONT-MSTB2.5/2-STF-5.08 (made by Phoenix Contact)
Network Cable	1-wire connection: 0.2 to 2.5 mm ² , AWG24 to 14 2-wire connection: 0.2 to 1.5 mm ² , AWG24 to 16

Wiring LONWORKS Interface Module

Precautions for Wiring

- Use a twisted-pair cable to connect the LONWORKS interface module to the network. Do not run the network cable in parallel with or near power lines, output lines, and motor lines. Keep the network cable away from noise sources.
- Power down the LONWORKS interface module before you start wiring. Make sure wiring is correct before powering up the LONWORKS interface module.
- One or two cables can be connected to one terminal of the network interface connector. When connecting one cable, use AWG24 to AWG14 cables (core cross-section 0.2 to 2.5 mm²). When connecting two cables to one terminal, use the same cables of AWG24 to AWG16 (0.2 to 1.5 mm²). Do not use cables of different diameters. Strip the cable insulation as shown at right.
- Tighten the mounting screws of the network interface connector to a recommended torque of 0.3 to 0.5 N·m.
- Tighten the terminal screws of the network interface connector to a recommended torque of 0.5 to 0.6 N·m.
- To prevent electrical shocks or communication error due to noises, connect the FG terminal to a proper ground using a grounding wire of UL1015 AWG22 or UL1007 AWG18 (grounding resistance 100Ω maximum). Do not connect the grounding wire in common with the grounding wire of motor equipment.



Ferrules, Crimping Tool, and Screwdriver for Phoenix Terminal Blocks

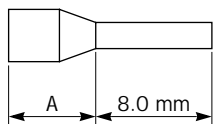
The screw terminal block of the network interface connector can be wired with or without using ferrules on the end of the cable. Applicable ferrules for the terminal block and crimping tool for the ferrules are listed below. Use a screwdriver to tighten the screw terminals on the LONWORKS interface module. Ferrules, crimping tool, and screwdriver are made by and available from Phoenix Contact.

Type numbers of Phoenix Contact ferrules, crimping tool, and screwdriver are listed below. When ordering these products from Phoenix Contact, specify the Order No. and quantity listed below.

• Ferrule Order No.

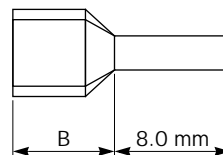
Applicable Wire Size		For 1-wire connection		For 2-wire connection		Pcs./Pkt.
mm ²	AWG	Phoenix Type	Order No.	Phoenix Type	Order No.	
0.25	24	AI 0,25-8 YE	32 00 85 2	—	—	100
0.5	20	AI 0,5-8 WH	32 00 01 4	AI-TWIN 2 x 0,5-8 WH	32 00 93 3	100
0.75	18	AI 0,75-8 GY	32 00 51 9	AI-TWIN 2 x 0,75-8 GY	32 00 80 7	100
1.0	18	AI 1-8 RD	32 00 03 0	AI-TWIN 2 x 1-8 RD	32 00 81 0	100
1.5	16	AI 1,5-8 BK	32 00 04 3	AI-TWIN 2 x 1,5-8 BK	32 00 82 3	100
2.5	14	AI 2,5-8 BU	32 00 52 2	—	—	100

For 1-wire Connection



Ferrule	Dimension A
AI 0,25-8 YE	4.5 mm
AI 0,5-8 WH	6.0 mm
AI 0,75-8 GY	
AI 1-8 RD	
AI 1,5-8 BK	
AI 2,5-8 BU	

For 2-wire connection



Ferrule	Dimension B
AI-TWIN 2 x 0,5-8 WH	7.0 mm
AI-TWIN 2 x 0,75-8 GY	
AI-TWIN 2 x 1-8 RD	
AI-TWIN 2 x 1,5-8 BK	8.0 mm

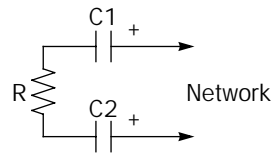
• Crimping Tool and Screwdriver Order No.

Tool Name	Phoenix Type	Order No.	Pcs./Pkt.
Crimping Tool	CRIMPFOX UD 6	12 04 43 6	1
Screwdriver	SZS 0,6 x 2,5	12 05 04 0	10

Terminator

Terminators must be connected to the LONWORKS network. When setting up a network, connect one or two terminators depending on the topology. The terminator consists of one resistor and two capacitors as illustrated below:

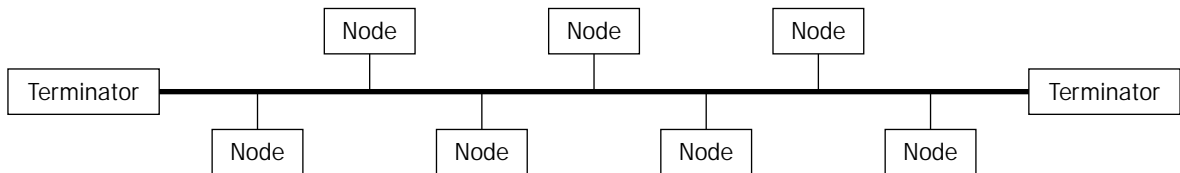
Terminator Configuration



Bus Topology

Connect terminators to the both ends of the bus topology network.

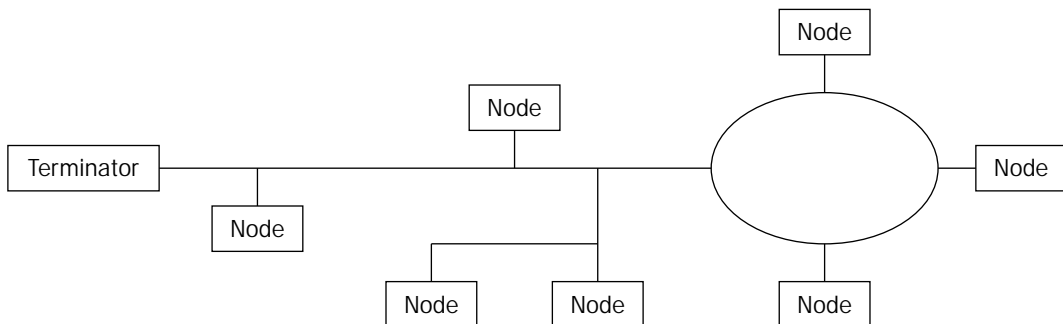
R	105Ω, 1%, 1/8W
C1 and C2	100 μF, ≥50V (note the polarity)



Free Topology

Connect a terminator to any position on the free topology network.

R	52.3Ω, 1%, 1/8W
C1 and C2	100 μF, ≥50V (note the polarity)



Link Registers for LONWORKS Network Communication

LONWORKS network communication data is stored to link registers in the OpenNet Controller CPU module and the data is communicated through the LONWORKS interface module.

Since seven functional modules, including a LONWORKS interface module, can be mounted with one OpenNet Controller CPU module, link registers are allocated depending on the position where the LONWORKS interface module is mounted.

Link Register Allocation Numbers

Allocation Number	Area	Function	Description	R/W
L*00	Data area	Receive data	Stores received data from the network	Read
L*01	Data area	Receive data	Stores received data from the network	Read
L*02	Data area	Receive data	Stores received data from the network	Read
L*03	Data area	Receive data	Stores received data from the network	Read
L*04	Data area	Transmit data	Stores transmit data for the network	Write
L*05	Data area	Transmit data	Stores transmit data for the network	Write
L*06	Data area	Transmit data	Stores transmit data for the network	Write
L*07	Data area	Transmit data	Stores transmit data for the network	Write
L*12	Status area	Error data	Stores various error codes	Read
L*13	Status area	I/O counts	Stores the byte counts of transmit/receive data	Read
L*24	ID area	Software version	Stores the user application software version	Read
L*25	ID area	Expansion module ID	Stores the user program module ID	Read

Note: A number 1 through 7 comes in place of * depending on the position where the functional module is mounted, such as OpenNet interface module or analog I/O module. Consequently, operand numbers are automatically allocated to each functional module in the order of increasing distance from the CPU module, starting with L100, L200, L300, through L700.

Error Data (Status Area) L*12

L*12	b15	b14: unused	b13	b12	b11	b10-b0: unused
------	-----	-------------	-----	-----	-----	----------------

When an error occurs, the I/O or ERR LED on the LONWORKS interface module goes on, according to the error, and a corresponding bit in the link register goes on. The status LED goes off when the cause of the error is removed. The error data bit remains on until the CPU is powered up again or reset.

b15 (initialization error)

This bit goes on when the CPU module fails to acknowledge the completion of initialization for communication with the LONWORKS interface module. When this bit goes on, the I/O LED also goes on.

b13 (I/O error)

This bit goes on when an error occurs during communication with the LONWORKS interface module through the CPU bus. When this bit goes on, the I/O LED also goes on.

b12 (transaction timeout)

This bit goes on when the CPU module fails to receive an acknowledge reply during communication through the LONWORKS network, with the acknowledge (ACKD) service enabled. When this bit goes on, the ERR LED also goes on. The transaction timeout is enabled only when the ACKD service is selected.

b11 (transmission error)

This bit goes on when a CRC error is detected while receiving incoming data from the LONWORKS network. When this bit goes on, the ERR LED also goes on.

I/O Counts (Status Area) L*13

L*13	b15-b12: transmit bytes	b11-b8: receive bytes	b7-b0: unused
------	-------------------------	-----------------------	---------------

This link register stores the transmit and receive byte counts selected in the Function Area Setting > Open Bus in WindLDR™.

Link Registers and Network Variables

Network variables are allocated to data areas of the link registers as shown below.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
L*00	nv_i8[1]							nv_i8[0]								
L*01	nv_i8[3]							nv_i8[2]								
L*02	nv_i8[5]							nv_i8[4]								
L*03	nv_i8[7]							nv_i8[6]								
L*04	nv_o8[1]							nv_o8[0]								
L*05	nv_o8[3]							nv_o8[2]								
L*06	nv_o8[5]							nv_o8[4]								
L*07	nv_o8[7]							nv_o8[6]								

• **Example**

Network variables nv_i8[0] and nv_i8[1] are allocated to link register data areas L100.00 through L100.15 as listed below.

L100	nv_i8[1]							nv_i8[0]								
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
	MSB							LSB	MSB							LSB
	1	0	0	0	1	1	1	1	0	1	0	0	0	1	1	1

Transmission Time

The transmission time depends on the network configuration, application program, and user program. It is recommended that you confirm the transmission time on the actual network system.

Processing transmit and receive data to and from the LONWORKS network is described below:

• **Processing Transmit Data**

The data in link registers are updated each time the CPU module scans the user program. The LONWORKS interface module reads data from the link registers allocated to transmit data in the OpenNet Controller CPU module. When any changes are found in the comparison between the new and old read data, the interface module updates the transmit network variables of which the data has been changed, and the new data is transmitted to the network.

The refresh cycle of reading from the link register to the interface module is approximately 15 msec. When the data in the link register is changed within 15 msec, the preceding data is not transmitted to the interface module. Data communication between the CPU module and the interface module through link registers is not in synchronism with the user program scanning.

When the CPU is powered up, the transmit data in the link registers are cleared to 0. Consequently, 0 cannot be transmitted in the first cycle immediately after the CPU is powered up because the transmit network variables are not updated.

• **Processing Receive Data**

When the interface module receives data from the network, corresponding receive network variables are updated, and the updated data is stored to the receive data area of link registers in the CPU module.

The refresh cycle of reading from the interface module to the link register is also approximately 15 msec, and is not in synchronism with the user program scanning. When the interface module receives subsequent data within 15 msec, the incoming data is stored in the buffer and is transmitted to link registers every 15 msec. The data in the link register is read each time the CPU module scans the user program.

Function Area Setting for LONWORKS Node

The quantity of transmit/receive data for LONWORKS network communication is specified using the Function Area Setting in WindLDR. The OpenNet Controller CPU module recognizes all functional modules, such as OpenNet interface modules and analog I/O modules, automatically at power-up and exchanges data with LONWORKS nodes through the link registers allocated to each node.

Since these settings relate to the user program, the user program must be downloaded to the OpenNet Controller after changing any of these settings.

Programming WindLDR

1. From the WindLDR menu bar, select **Configure > Function Area Settings**. The Function Area Setting dialog box appears.
2. Select the **Open Bus** tab.

Configure Communication Master Module Check Box
Check this box only when the remote I/O master module is used.

Slave Station Transmit/Receive Data Quantity (Bytes)
When using OpenNet interface modules for DeviceNet, INTERBUS, or LONWORKS, specify the data bytes to communicate through each OpenNet interface module.

Module	Transmit	Receive
1:	8	4
2:	8	8
3:	8	8
4:	8	8
5:	8	8
6:	8	8
7:	8	8

Quantity of Nodes Connected
When using the remote I/O master module, specify the quantity of nodes from 1 through 32.

Transmit/Receive Bytes 0 to 8 (default: 8 bytes)
This value determines the data quantity 0 through 8 bytes (64 bits) to communicate with the network.
For the example on the next page, select 8 transmit bytes and 4 receive bytes for Module 1.

3. Select transmit and receive data bytes for module position 1 through 7 where the LONWORKS interface module is mounted.
4. Click the **OK** button and download the user program to the OpenNet Controller.

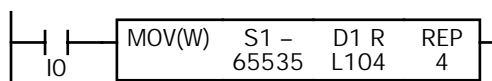
Programming Transmit/Receive Data Using WindLDR

The OpenNet interface module exchanges data between the open network and the link registers in the CPU module allocated to the OpenNet interface module, depending on the slot where the OpenNet interface module is mounted.

To create a communication program for an OpenNet interface module, first determine the slot number where the OpenNet interface module is mounted, and make a program to write data to link registers allocated to transmit data and to read data from link registers allocated to receive data.

Example: When a LONWORKS interface module is mounted in the first slot of all functional modules

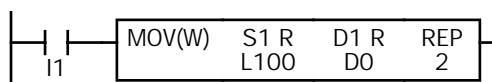
• Transmit Data



65535 → L104 through L107

When input I0 is on, constant 65535 (FFFFh) designated by source operand S1 is moved to four link registers L104 through L107 designated by destination operand D1. All 64 bits (8 bytes) in link registers L104 through L107 are turned on. Since link registers L104 through L107 transmit data, the data is transmitted to the network.

• Receive Data



L100-L101 → D0-D1

When input I1 is on, 32-bit (4-byte) data in two link registers L100 and L101 designated by source operand S1 is moved to data registers D0 and D1 designated by destination operand D1. Since link registers L100 and L101 receive data, communication data read to L100 and L101 is moved to data registers D0 and D1.

Starting Operation

The LONWORKS network requires installation of network configuration information into each node. When setting up the LONWORKS network for the first time, follow the procedures described below:

1. Set up the OpenNet Controller CPU and LONWORKS interface modules, connect the LONWORKS interface module to the LONWORKS network using LONWORKS cables, and power up the CPU module.
2. Connect a network management tool to the network and install network configuration information to the LONWORKS interface module. See Network Management described below.
3. Download the user program to the CPU module.
4. Start the CPU module to run, then the CPU module starts to communicate with other nodes on the LONWORKS network as specified in the network configuration information and user program.

The delay until the communication starts after power-up depends on the size of the user program and the system setup.

While the CPU is stopped, data exchange between the CPU and LONWORKS interface modules is halted, but communication with the LONWORKS network continues.

Data exchange between the CPU and LONWORKS interface modules is asynchronous with the user program scanning in the CPU module.

Network Management

When setting up a LONWORKS network system, the user has to install network configuration information into each node. Use a network management tool available from other manufacturers (such as LonMaker for Windows Integration Tool) to install network configuration information. An external interface file (XIF extension) unique to each product series is needed to install the network configuration information. The external interface file for the LONWORKS interface module is available from IDEC. Find an XIF No. printed on the side of the LONWORKS interface module or on the shipping package. When requesting an external interface file, inform IDEC of the XIF No. that represents the external interface file version number. Without a correct external interface file of the matching XIF No., network configuration information cannot be installed successfully.

The network configuration information includes addressing, binding, and configuration.

Addressing:	Determines each node address
Binding:	Determines target nodes to communicate with
Configuration:	Determines the type of message service, retry cycles, timeout period, etc.



Caution

- When using the LONWORKS interface module, select the acknowledge (ACKD) service to enable the message service for network variables and set the retry cycles to a value of 1 or more. If communication is performed using other than the ACKD service, the ERR LED on the interface module does not function properly.
- When installing the network configuration information without modifying the application program, an external interface file (XIF extension) containing information, such as the network variables of the LONWORKS interface module, is needed. Consult IDEC for the external interface file.
- The user must keep a backup file of the network configuration information used for network management.

Precautions for Modifying Application Program

The LONWORKS interface module is shipped with a standard application program installed. Users with expertise in programming can also modify or create application programs using a special programming tool, such as LonBuilder Developer's Kit. The application program is written in Neuron C. Read this section before starting modifications.

Define Neuron Chip I/O pins

As shown in the sample program on page 19, define I/O pins IO.0 through IO.4 and IO.6 of the Neuron Chip. If these pins are not defined correctly, the LONWORKS interface module may be damaged. For the description of I/O pins, see page 15.

Include necessary codes in the application program

When you modify or create an application program, make sure that the codes shown in italics in the application program examples on pages 18 through 22 are included in the application program.

Defined network variables

The application program installed in the LONWORKS interface module defines network variables for transmit and receive data listed on page 23. When you modify or create an application program, do not use these variable names, otherwise verification of the application program will be difficult.

Precautions for writing and reading registers

Make a program to write and read data to and from registers in the LONWORKS interface module as shown in the sample programs on pages 21 and 22.

While data write or read is in progress, do not execute any other command.

Precautions for downloading an application program to the flash memory through the network

A special tool is required to download an application program. Before starting download, stop the OpenNet Controller CPU operation. While downloading is in progress, make sure the power voltage is within the rated operating voltage range.

Precautions for flash memory used for the application program

Do not store variables to the flash memory. To hold variables and other data while power is off, use the RAM backup function of the CPU module.

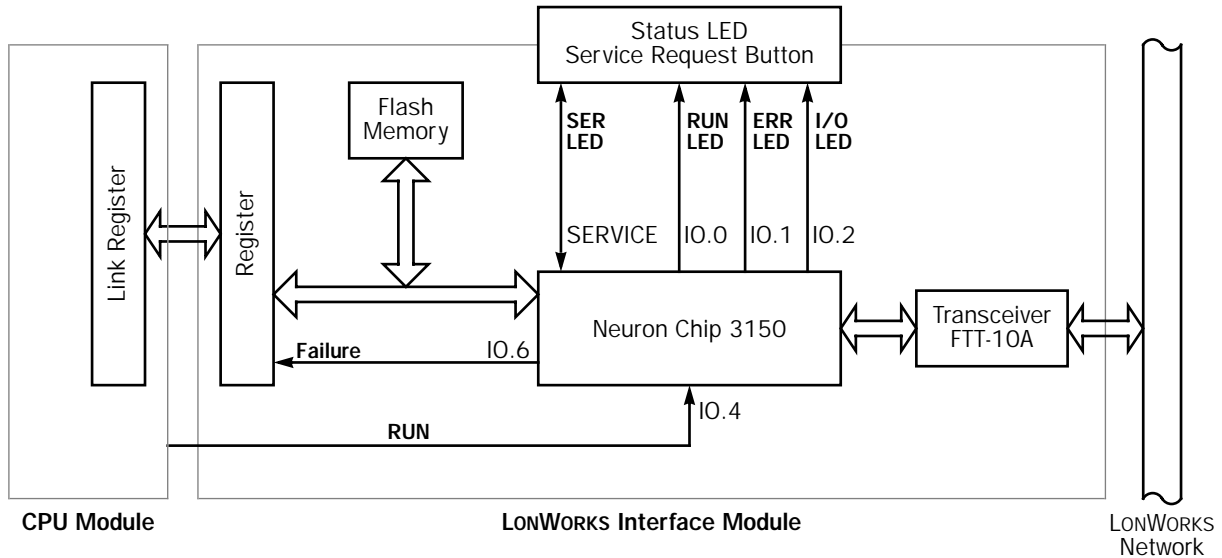
The flash memory can be rewritten a maximum of 10,000 times.

Precautions for system setup

Set the retry cycles of the message service to a value of 1 or more.

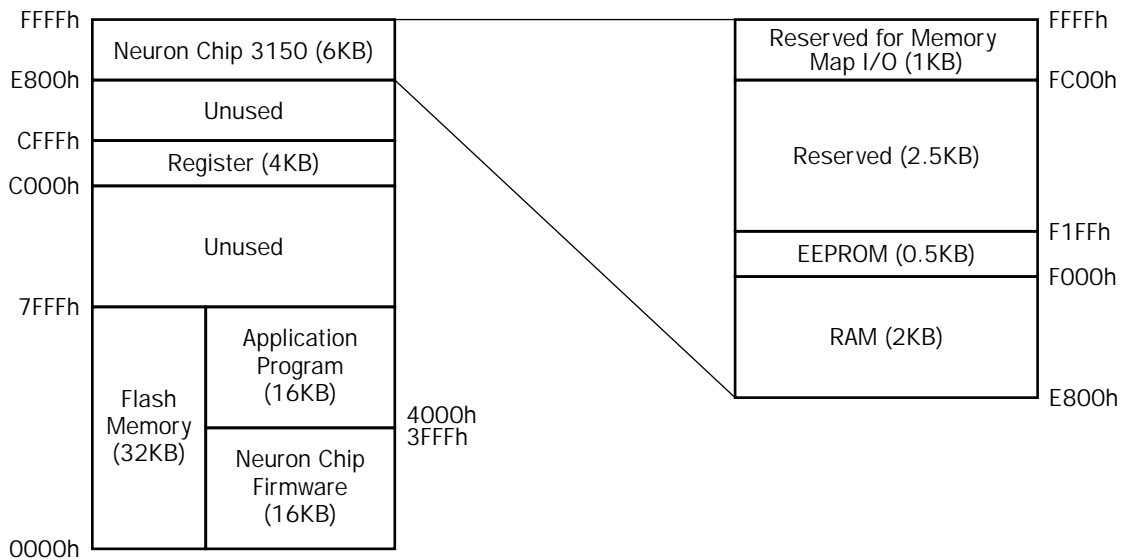
LONWORKS Interface Module Internal Structure

The LONWORKS interface module block diagram is illustrated in the figure below:



Memory Map

The LONWORKS interface module memory map is illustrated in the figure below:



Flash Memory

The LONWORKS interface module contains a 32KB nonvolatile rewritable memory. Of the 32KB memory area, a 16KB area of 0000h through 3FFFh is allocated to the Neuron Chip firmware, and the remaining 16KB area of 4000h through 7FFFh is allocated to the application program.

Neuron Chip I/O Pins and Status LEDs

Neuron Chip I/O pins and status LEDs are assigned as listed below:

I/O Pin No.	I/O	Signal Name	Description
0	Output	RUN LED	Controls the RUN LED (green). 0: ON, 1: OFF
1	Output	ERR LED	Controls the ERR LED (red). 0: ON, 1: OFF
2	Output	I/O LED	Controls the I/O LED (red). 0: ON, 1: OFF
3	Input	—	The IO.3 pin must be defied as an input when the application program is modified by the user. See page 19.
4	Input	RUN	Monitors the CPU module operating status. 0: CPU stopped, 1: CPU in operation
5	—	unused	
6	Output	Failure	Error signal to the CPU 0: The Neuron Chip cannot write data to registers. When modifying the application program, make sure to turn this pin to 0 when an unrecoverable critical error occurs. 1: Normal operation
7-10	—	unused	

Registers

The OpenNet Controller CPU module exchanges communication data through the registers in the LONWORKS interface module. The register addresses are listed in the table below:

Address	Name	Data Flow Direction		Description
		CPU Module	Interface Module	
C000h - C007h	Data register (8 bytes)	←	→	Allocate network variables to these addresses to exchange data between the CPU and interface modules.
C008h - C00Fh	Data register (8 bytes)	→	←	
C010h - C011h	reserved	—	—	Do not write data into this area.
C012h	Error data	←	→	Use this address to read error data from the interface module.
C013h	I/O counts	→	←	Use this address to store the byte counts of transmit/receive data selected in WindLDR Function Area Settings.
C014h - C017h	reserved	—	—	Do not write data into this area.
C018h	Software version	←	→	Use this address to write the user application software version number (use any number other than 00h).
C019h	Expansion module ID	←	→	Use this address to write the user program module ID (use a number 40h through 7Fh).
C01Ah - CFFFh	reserved	—	—	Do not write data into this area.

Data Exchange between LONWORKS Interface Module and CPU Module

Communication data, status data, and ID data are exchanged through registers in the LONWORKS interface module and link registers in the CPU module. The registers correspond to link registers as listed below:

Register Address in LONWORKS Interface Module	Link Register in CPU Module	Function	Area
C000h - C001h	L*00	Receive Data	Communication Data Area
C002h - C003h	L*01		
C004h - C005h	L*02		
C006h - C007h	L*03		
C008h - C009h	L*04	Transmit Data	
C00Ah - C00Bh	L*05		
C00Ch - C00Dh	L*06		
C00Eh - C00Fh	L*07		
C012h	L*12	Error Data	Status Area
C013h	L*13	I/O Counts	
C018h	L*24	Software Version	ID Area
C019h	L*25	Expansion Module ID	

Note: A number 1 through 7 comes in place of * depending on the position where the functional module, such as OpenNet interface module or analog I/O module, is mounted. Consequently, operand numbers are automatically allocated to each functional module in the order of increasing distance from the CPU module, starting with L100, L200, L300, through L700.

Example 1: Receive Data in Registers C000h and C001h

When receive data enters registers C000h and C001h in the LONWORKS interface module, the data is transferred to a link register in the CPU module as illustrated below:

Registers in the LONWORKS Interface Module	C001h (8 bits)								C000h (8 bits)							
	b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
	MSB							LSB	MSB							LSB
	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
	↓								↓							
Link Register L*00 in the CPU Module	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
	MSB															LSB
	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

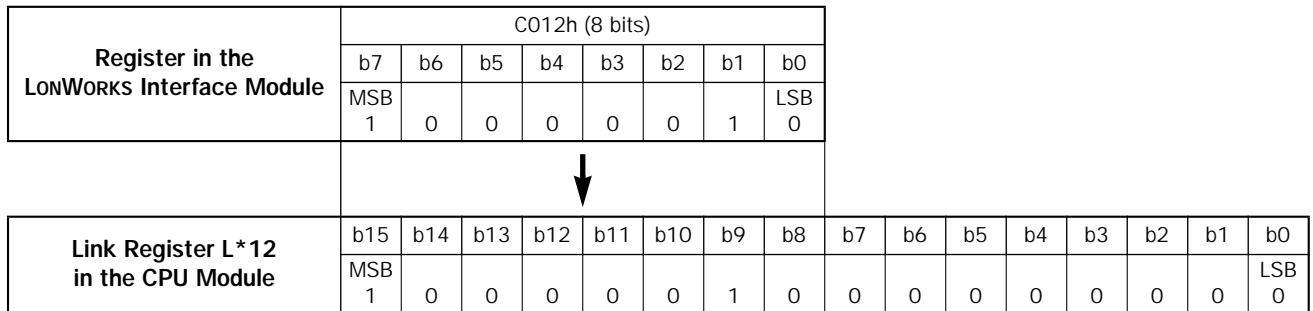
Example 2: Transmit Data in Link Register L*04

When transmit data is stored to link register L*04 in the CPU module, the data is transferred to registers in the LONWORKS interface module as illustrated below:

Link Register L*04 in the CPU Module	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
	MSB															LSB
	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0
	↓								↓							
Registers in the LONWORKS Interface Module	C009h (8 bits)								C008h (8 bits)							
	b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
	MSB							LSB	MSB							LSB
	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0

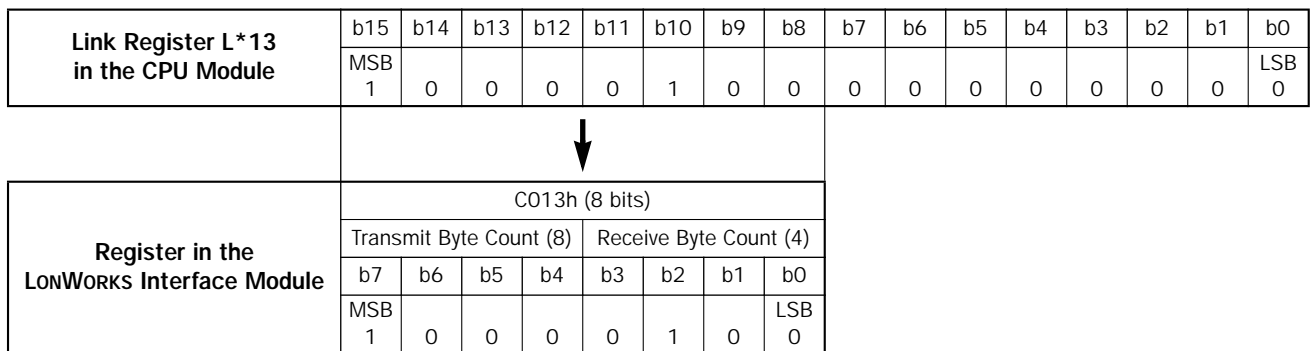
Example 3: Error Data in Register C012h

When error data enters register C012h in the LONWORKS interface module, the data is transferred to a link register in the CPU module as illustrated below:



Example 4: I/O Counts in Link Register L*13

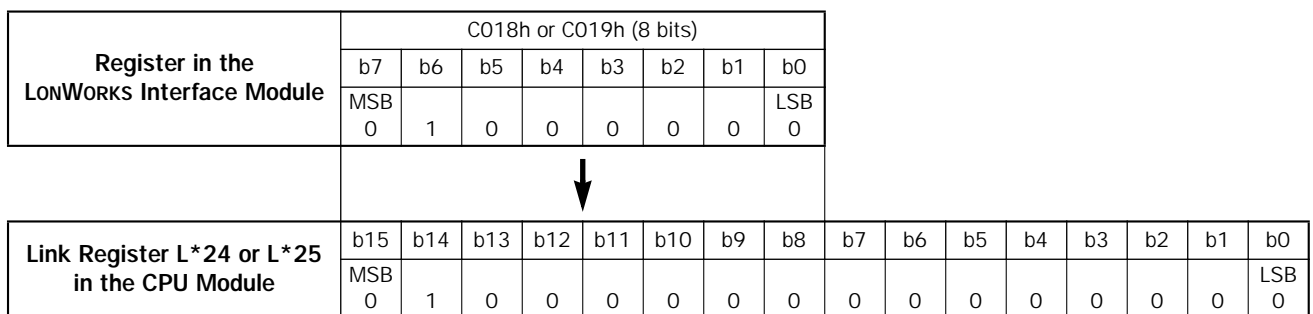
When 8 bytes (output) and 4 bytes (input) are selected as the transmit and receive data quantities in WindLDR Function Area Settings, respectively, these values are stored to link register L*13 in the CPU module, and the data is transferred to register C013h in the LONWORKS interface module as illustrated below:



Note: Link register L*13 is for read only. Do not write data into L*13.

Example 5: Software Version in Register C018h and Expansion Module ID in Register C019h

When a software version number is stored to register C018h in the LONWORKS interface module, or when an expansion module ID is stored to register C019h in the LONWORKS interface module, the data is transferred to a link register in the CPU module as illustrated below:

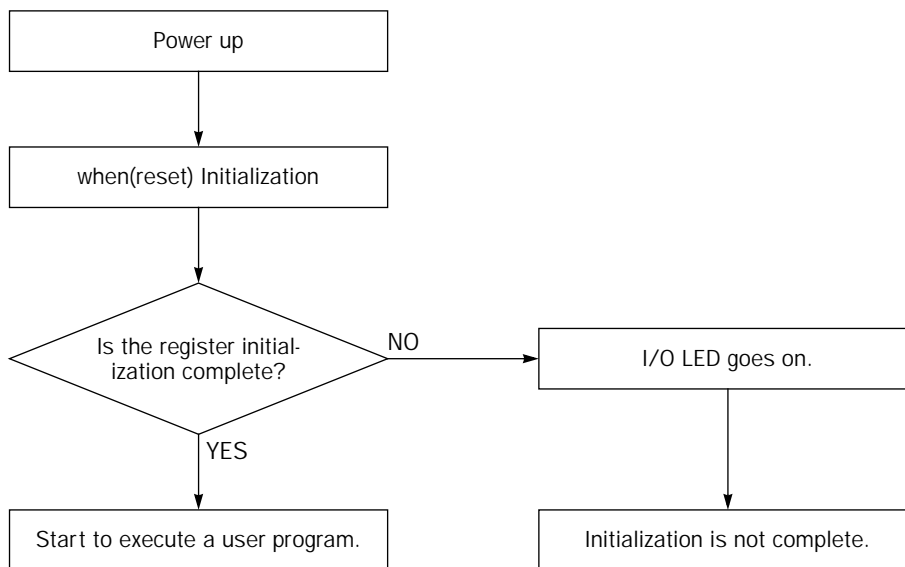


Application Program Examples

This section describes application program examples for initializing the registers in the LONWORKS interface module, writing receive data to data registers, and reading transmit data from data registers.

Initialization

Before starting LONWORKS communication through the network, the data registers in the LONWORKS interface module have to be initialized. The initialization sequence is illustrated in the chart below:



The following program is an example of an application program in Neuron C to initialize the LONWORKS interface module, consisting of initialization codes and a header file. *When you modify or create an application program, make sure that the application program includes the following codes in italics.*

Initialization Codes

```

1.  //////////////////////////////////////
2.  ///      PRAGMA          ///
3.  //////////////////////////////////////
4.  #pragma scheduler_reset
5.  /******
6.  Network Variable
7.  /******
8.  /* Define network variables          */
9.  /******
10. Write the software version number to C018h
11. /******
12. #define FC3ASX5L_VERSION    0x10
13. /******
14. Write the expansion module ID to C019h
15. /******
16. #define EMID_CODE          0x50
17. /******
18. include file
19. /******
20. #include <access.h>
21. #include <msg_addr.h>
22. #include <control.h>
23. #include <status.h>
24. #include <snvt_lev.h>
25. #include "fc3asx5l.h"          /* Refer to the header file shown below */
26. /******
  
```

```

27. Main Program
28. *****/
29. when(reset){
30.     initialize();
31. /* Insert other commands here to execute within when(reset), if required. */
32. }

```

Header File (fc3asx51.h)

```

1. //Header File: fc3asx51.h
2. *****/
3. /* Common Definition */
4. *****/
5. #define LED_OFF 1
6. #define LED_ON 0
7. #define OK 1
8. #define NG 0
9. #define HIGH 1
10. #define LOW 0
11. /* Timer Value */
12. #define DTm_5sec 5000
13. *****/
14. /* Memory Mapped I/O Definition */
15. *****/
16. #define IO_GA_BASE 0xc000 // I/O Base Address
17. *****/
18. /* Digital I/O Register Address */
19. *****/
20. #define GA_FCDR (IO_GA_BASE + 0x00) // Data Register
21. #define GA_CSR_ERR (IO_GA_BASE + 0x12) // Error Register
22. #define GA_FVER (IO_GA_BASE + 0x18) // I/O Version Register
23. #define GA_EMID (IO_GA_BASE + 0x19) // Expansion Module ID Register
24. #define GA_BCTL (IO_GA_BASE + 0x1a)
25. *****/
26. /* I/O Register Bit Definition */
27. *****/
28. #define BCTL_CENABLE 0x10
29. #define BCTL_NWR_REQ 0x04
30. #define BCTL_NENABLE 0x01
31. #define MAX_FCDR_DATA_LEN 16
32. /* Define Neuron Chip IO pins as follows. */
33. IO_0 output bit PO_RUN_LED = HIGH;
34. IO_1 output bit PO_ERR_LED = HIGH;
35. IO_2 output bit PO_IO_LED = HIGH;
36. IO_3 input bit PI_ODE;
37. IO_4 input bit PI_RUN;
38. IO_6 output bit PO_F_ERR = LOW;
39. *****/
40. /* Prototype */
41. *****/
42. void initialize(void);
43. void init_internal_io(void);
44. void init_external_io(void);
45. void init_gate_array(void);
46.
47. *****/
48. /* Global Variable */
49. *****/
50. mtimer io_check_timer;
51. unsigned char csr_error_data; // CSR_ERROR Reg. data save area
52.
53. void initialize(void){

```

```

54.     init_internal_io();
55.     init_external_io();
56. }
57. void init_internal_io(void){
58.     io_change_init(PI_ODE);
59.     io_change_init(PI_RUN);
60. }
61. void init_external_io(void){
62.     init_gate_array();
63. }
64. void init_gate_array(void){
65.     int st, n;
66.     unsigned char *pGA;
67.     unsigned char dat;
68.
69.     io_check_timer = DTm_5sec;
70.     while(TRUE){
71.         post_events();
72.         pGA = (unsigned char *)GA_BCTL;
73.         *pGA |= BCTL_NWR_REQ;
74.         dat = *pGA;
75.         if (dat & BCTL_NWR_REQ){
76.             pGA = (unsigned char *)GA_FCDR;
77.             for (n = 0; n < MAX_FCDR_DATA_LEN; n++){
78.                 *pGA++ = 0x00;
79.             }
80.             pGA = (unsigned char *)GA_CSR_ERR;
81.             csr_error_data = 0;
82.             *pGA = csr_error_data;
83.             pGA = (unsigned char *)GA_FVER;
84.             *pGA = FC3ASX5L_VERSION;
85.             pGA = (unsigned char *)GA_EMID;
86.             *pGA = EMID_CODE;
87.             pGA = (unsigned char *)GA_BCTL;
88.             *pGA |= BCTL_NENABLE;
89.             dat = *pGA;
90.             if (dat & BCTL_NENABLE){
91.                 *pGA &= ~BCTL_NWR_REQ;
92.                 break;
93.             }else{
94.                 *pGA &= ~BCTL_NWR_REQ;
95.             }
96.         }
97.         /* The following program turns on the I/O LED when initialization fails within 5 seconds, and
98.         can be modified by the user. */
99.         if (timer_expires(io_check_timer)){
100.            io_out(PO_IO_LED, LOW);           /* I/O LED goes on when timeout */
101.            break;
102.        }
103.    }

```

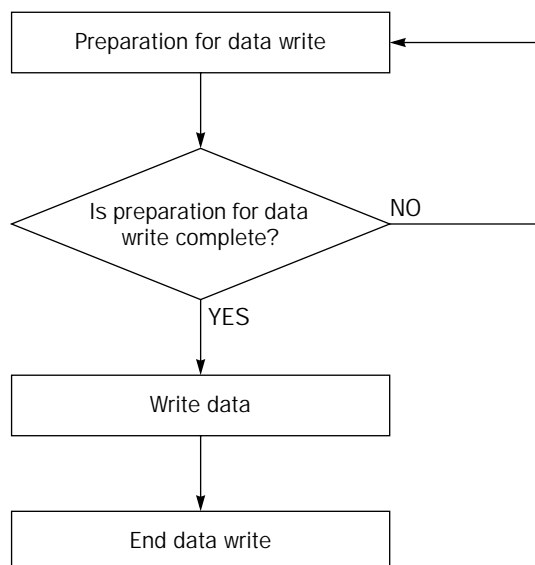
Note: ~ is an exclusive OR of every bit.

Brief description of functions used for the initialization program

- `init_internal_io()` function
This function initializes the Neuron Chip internal IO pins.
- `init_external_io()` function
This function substitutes the number of register IO points for `max_out_number` or `max_in_number`.
- `init_gate_array()` function
This function turns on the I/O LED when initialization of registers fails within 5 seconds.

Writing Receive Data to Data Registers in the LONWORKS Interface Module

The following diagram shows a typical example of writing receive data to the data registers in the LONWORKS interface module.



Application Program Example for Data Write

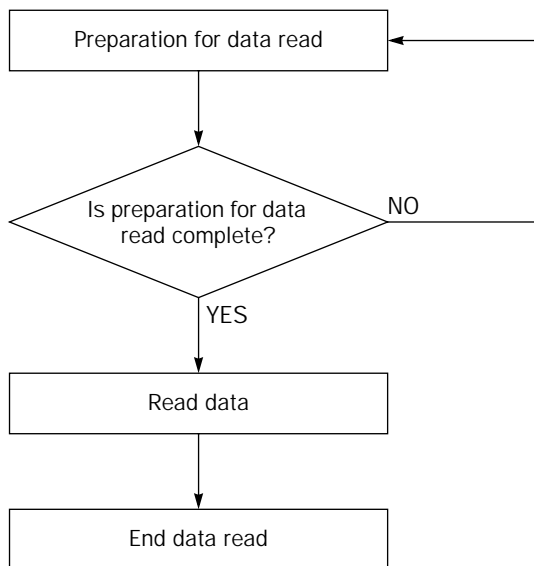
The following program is an example to write receive data to data register C000h of the LONWORKS interface module when an 8-bit input network variable (nv_i8) is updated. *When you modify or create an application program, make sure that the application program includes the following codes in italics.*

```

1.  /* Input Network Variables */
2.  network input unsigned char nv_i8;
3.  /* define */
4.  #define GA_BCTL          0xC01A
5.  #define BCTL_NWR_REQ    0x04
6.  #define GA_FCDR_RX     0xC000
7.
8.  when(nv_update_occurs(nv_i8)){          /* Acknowledge input network variable update */
9.      unsigned char *pGA;
10.     unsigned char dat;
11.     while(TRUE){
12.         pGA = (unsigned char *)GA_BCTL;  /* Preparation for data write */
13.         *pGA |= BCTL_NWR_REQ;
14.         dat = *pGA;
15.         if (dat & BCTL_NWR_REQ){        /* Preparation for data write complete */
16.             pGA = (unsigned char *)GA_FCDR_RX;
17.             pGA = nv_i8;                /* Write input NV data to data register C000h */
18.             pGA = (unsigned char *)GA_BCTL;
19.             pGA &= ~BCTL_NWR_REQ;      /* End data write */
20.             break;
21.         }
22.     }
23. }
  
```

Reading Transmit Data from Data Registers in the LONWORKS Interface Module

The following diagram is a typical example of reading transmit data from the data registers in the LONWORKS interface module.



Application Program Example for Data Read

The following program is an example to substitute transmit data of data register C008h for an 8-bit output network variable (nv_o8). *When you modify or create an application program, make sure that the application program includes the following codes in italics.*

```

1.  /* Output Network Variables */
2.  network output unsigned char nv_o8;
3.  /* define */
4.  #define GA_BCTL          0xC01A
5.  #define GA_FCDR_TX      0xC008
6.  #define BCTL_NWR_REQ    0x04
7.  #define HIGH           1
8.  /* Define IO_4 RUN */
9.  IO_4 input bit PI_RUN;
10.
11. when(TRUE){
12.     unsigned char *pGA;
13.     unsigned char dat;
14.     unsigned char tx_dat;
15.     while(TRUE){
16.         if (io_in(PI_RUN) == HIGH){
17.             pGA = (unsigned char *)GA_BCTL;          /* Preparation for data read */
18.             *pGA |= BCTL_NWR_REQ;
19.             dat = *pGA;
20.             if (dat & BCTL_NWR_REQ){                  /* Preparation for data read complete */
21.                 pGA = (unsigned char *)GA_FCDR_TX;
22.                 tx_dat = *pGA;                       /* Read data from register C008h */
23.                 pGA = (unsigned char *)GA_BCTL;
24.                 pGA &= ~BCTL_NWR_REQ;               /* End data read */
25.                 nv_o8 = tx_dat; /* Substitute the value for output network variable (nv_o8) */
26.                 break;
27.             }
28.         }
29.     }
30. }
    
```

Defined Network Variables

The application program installed in the LONWORKS interface module defines network variables for transmit and receive data listed below. When you modify or create an application program, do not use these variables, otherwise verification of the application program will be difficult. The network variables, their data type and structure are listed in the following tables.

Input Network Variables

Input Network Variable	Data Type and Structure	Used For
nv_i8[0]	unsigned char	8-point inputs, 8 bits
nv_i8[1]	unsigned char	8-point inputs, 8 bits
nv_i8[2]	unsigned char	8-point inputs, 8 bits
nv_i8[3]	unsigned char	8-point inputs, 8 bits
nv_i8[4]	unsigned char	8-point inputs, 8 bits
nv_i8[5]	unsigned char	8-point inputs, 8 bits
nv_i8[6]	unsigned char	8-point inputs, 8 bits
nv_i8[7]	unsigned char	8-point inputs, 8 bits
nv_i16	BIT16_DAT	16-point inputs, 8 bits × 2
nv_i24	BIT24_DAT	24-point inputs, 8 bits × 3
nv_i32	BIT32_DAT	32-point inputs, 8 bits × 4
nv_i40	BIT40_DAT	40-point inputs, 8 bits × 5
nv_i48	BIT48_DAT	48-point inputs, 8 bits × 6
nv_i56	BIT56_DAT	56-point inputs, 8 bits × 7
nv_i64	BIT64_DAT	64-point inputs, 8 bits × 8

Output Network Variables

Output Network Variable	Data Type and Structure	Used For
nv_o8[0]	unsigned char	8-point outputs, 8 bits
nv_o8[1]	unsigned char	8-point outputs, 8 bits
nv_o8[2]	unsigned char	8-point outputs, 8 bits
nv_o8[3]	unsigned char	8-point outputs, 8 bits
nv_o8[4]	unsigned char	8-point outputs, 8 bits
nv_o8[5]	unsigned char	8-point outputs, 8 bits
nv_o8[6]	unsigned char	8-point outputs, 8 bits
nv_o8[7]	unsigned char	8-point outputs, 8 bits
nv_o16	BIT16_DAT	16-point outputs, 8 bits × 2
nv_o24	BIT24_DAT	24-point outputs, 8 bits × 3
nv_o32	BIT32_DAT	32-point outputs, 8 bits × 4
nv_o40	BIT40_DAT	40-point outputs, 8 bits × 5
nv_o48	BIT48_DAT	48-point outputs, 8 bits × 6
nv_o56	BIT56_DAT	56-point outputs, 8 bits × 7
nv_o64	BIT64_DAT	64-point outputs, 8 bits × 8

Structure Name	Structure	Used For
BIT16_DAT	typedef struct { unsigned char dat[2]; }BIT16_DAT	16-point outputs, 8 bits × 2
BIT24_DAT	typedef struct { unsigned char dat[3]; }BIT24_DAT	24-point outputs, 8 bits × 3
BIT32_DAT	typedef struct { unsigned char dat[4]; }BIT32_DAT	32-point outputs, 8 bits × 4
BIT40_DAT	typedef struct { unsigned char dat[5]; }BIT40_DAT	40-point outputs, 8 bits × 5
BIT48_DAT	typedef struct { unsigned char dat[6]; }BIT48_DAT	48-point outputs, 8 bits × 6
BIT56_DAT	typedef struct { unsigned char dat[7]; }BIT56_DAT	56-point outputs, 8 bits × 7
BIT64_DAT	typedef struct { unsigned char dat[8]; }BIT64_DAT	64-point outputs, 8 bits × 8

Example:

When the transmit and receive bytes are set to 3 using WindLDR (on the Open Bus page selected from Configure > Function Area Settings), only 24-point type declared network variables (nv_i24 and nv_o24) and the network variables shown in the table below can be used. Then, link registers listed below can be used for transmission and receiving.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
L*00	nv_i8[1]								nv_i8[0]							
L*01	cannot be used								nv_i8[2]							
L*04	nv_o8[1]								nv_o8[0]							
L*05	cannot be used								nv_o8[2]							

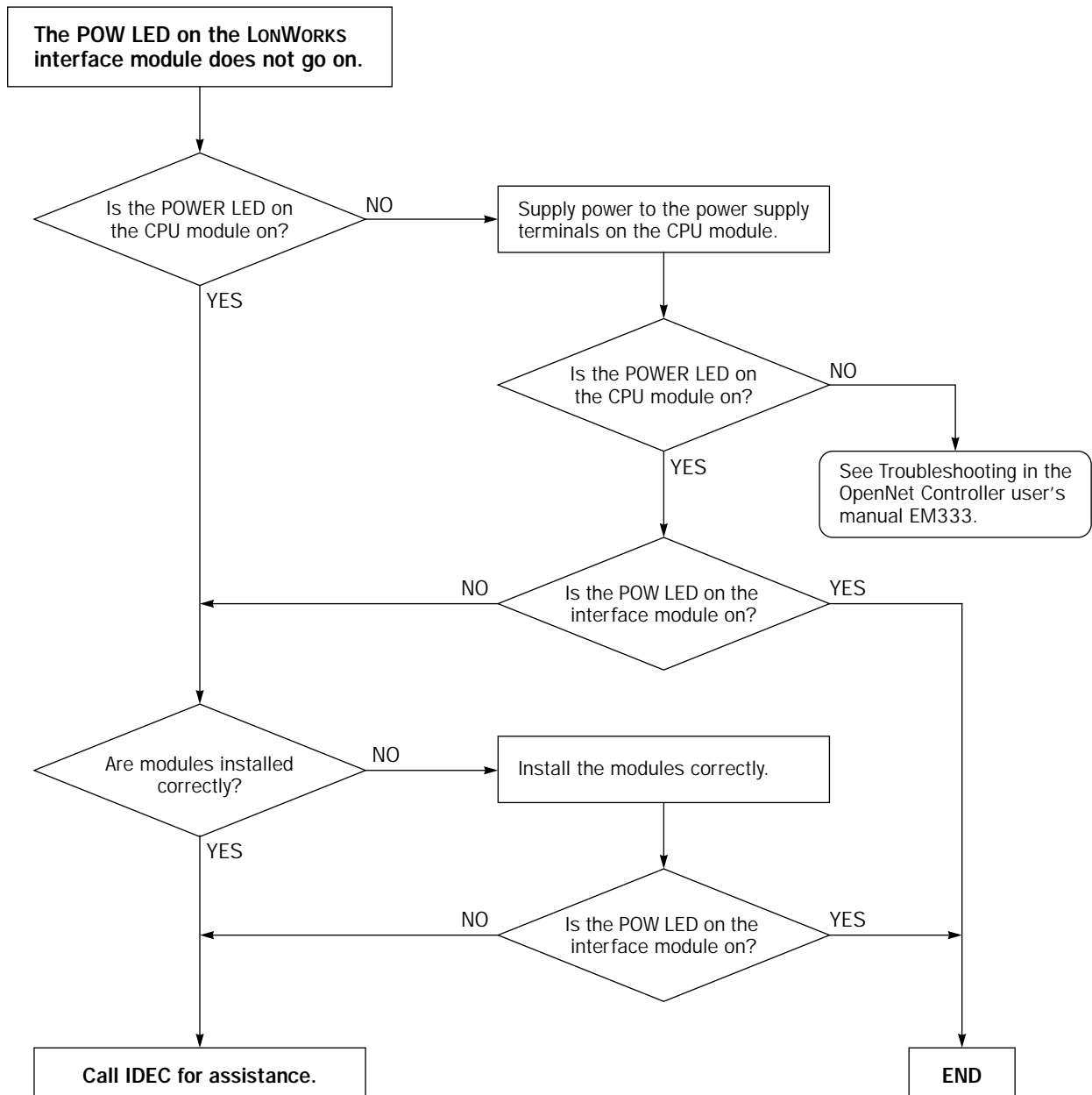
LONWORKS Network Troubleshooting

This section describes the procedures to determine the cause of trouble and actions to be taken when any trouble occurs while operating the LONWORKS interface module.

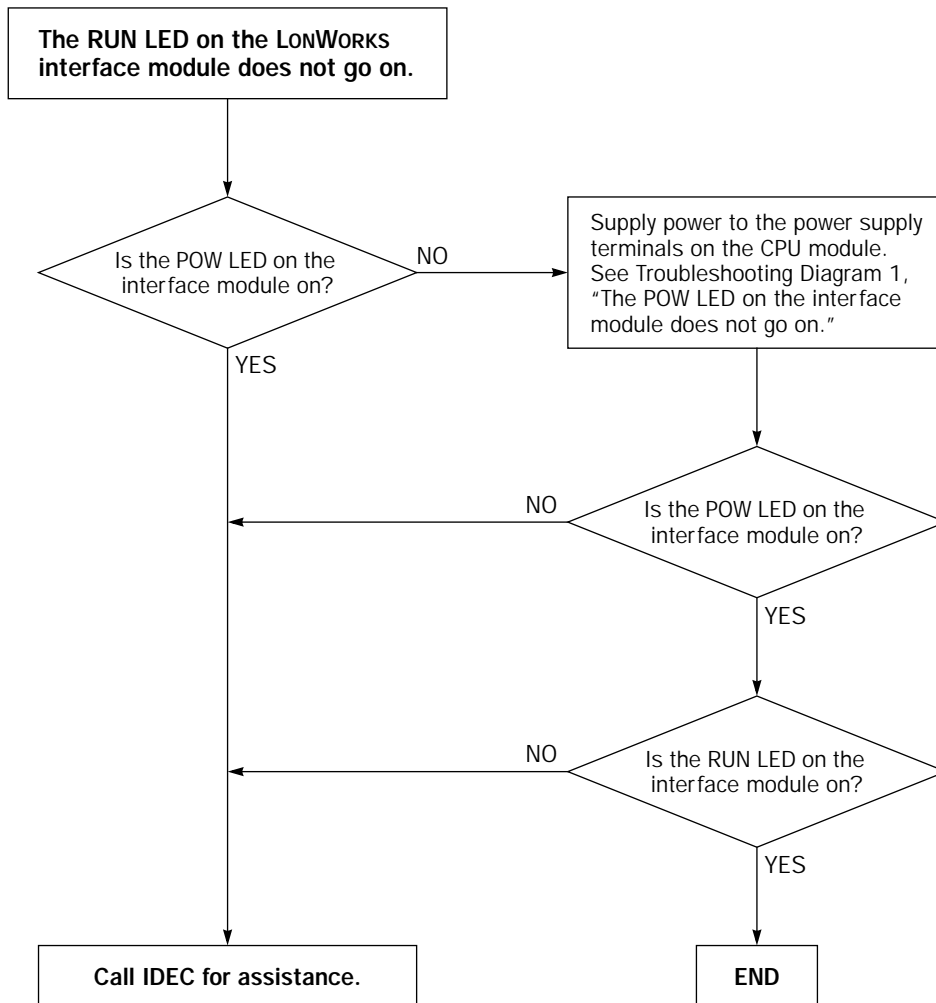
Probable Causes for Network Errors

- A network cable is disconnected or shorted.
- Strong external noise
- The power voltage to the module has dropped below the minimum operating voltage at least momentarily.
- Use of a faulty communication line, cable other than twisted-pair cables, or transmission beyond the rated distance.
- Improper terminator

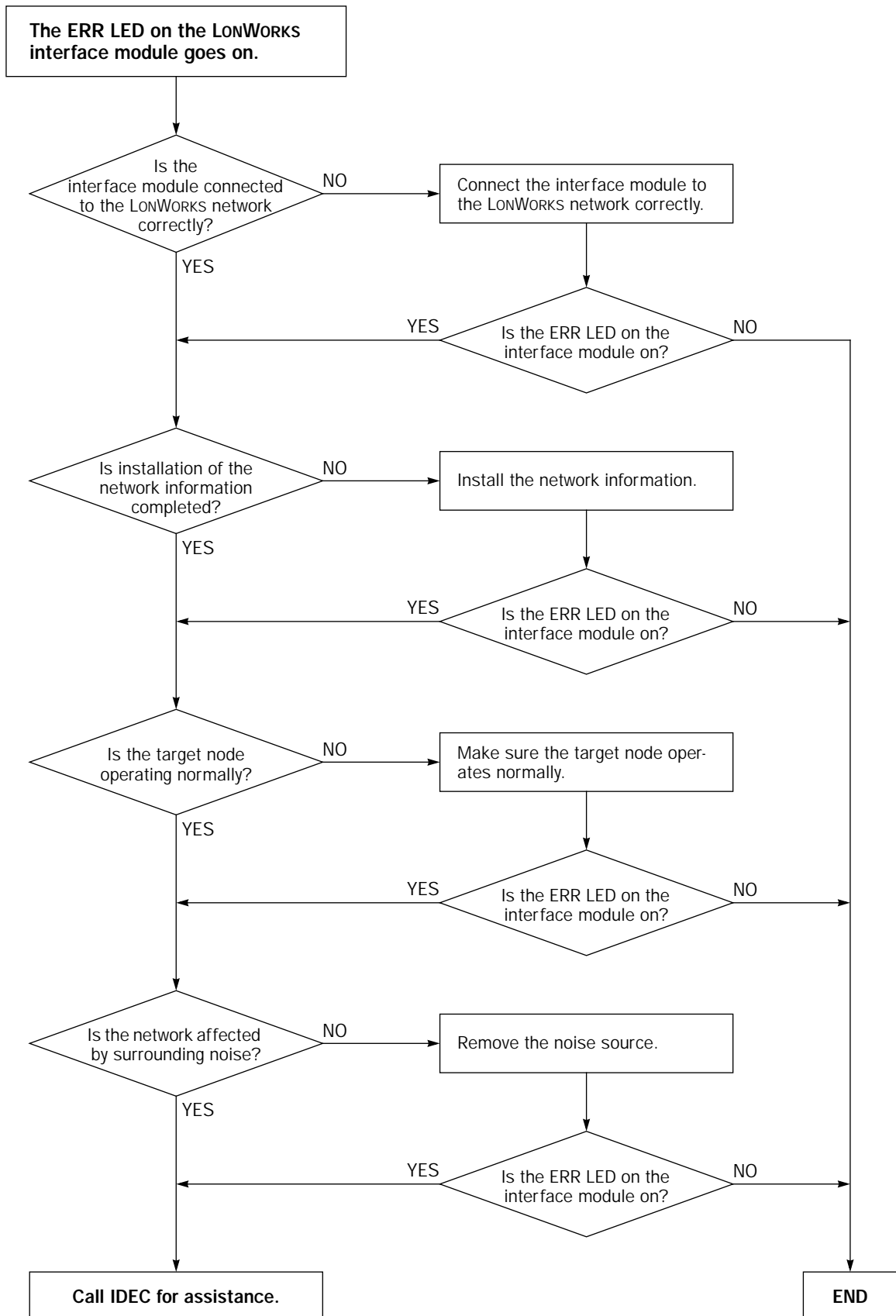
Troubleshooting Diagram 1



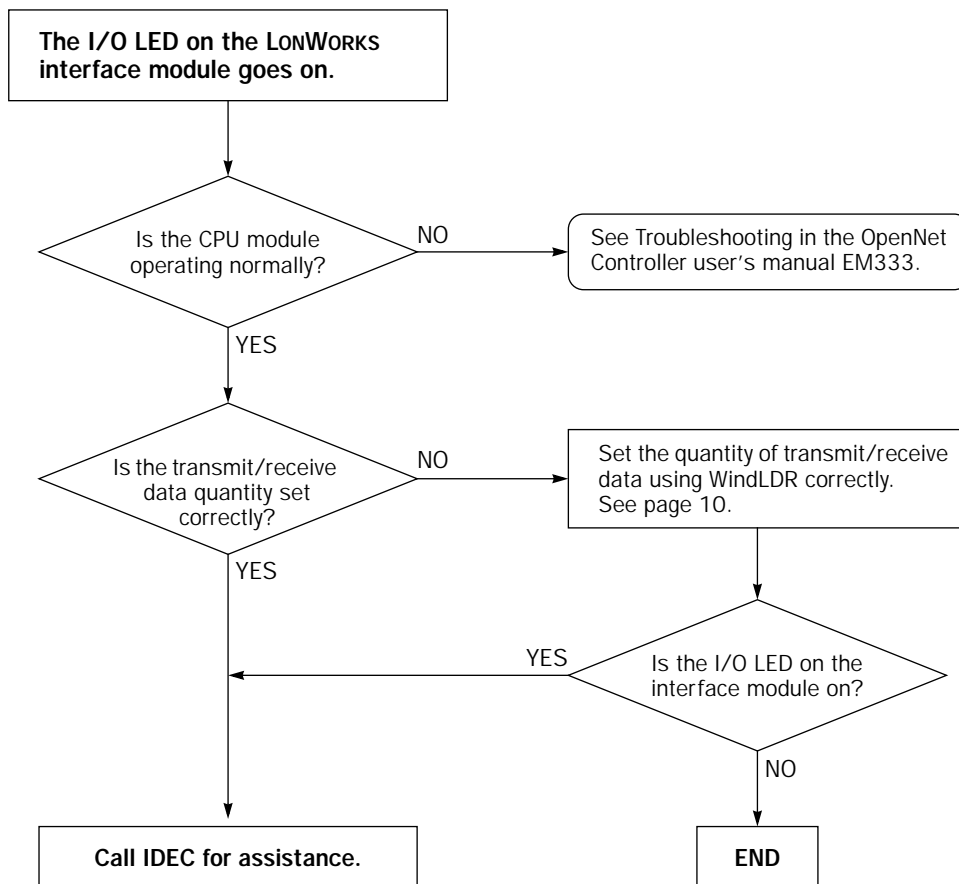
Troubleshooting Diagram 2



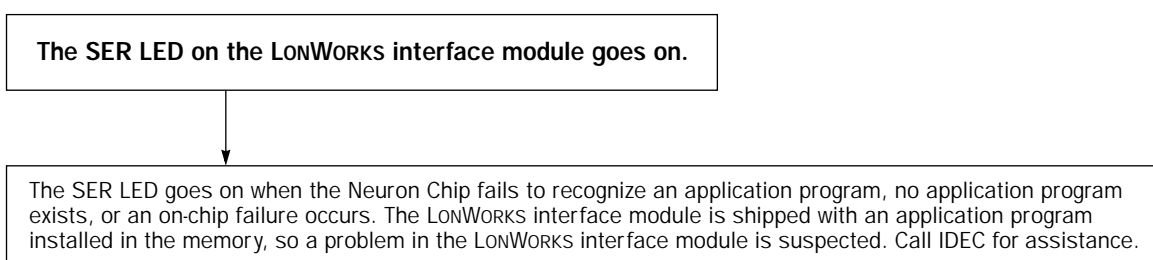
Troubleshooting Diagram 3



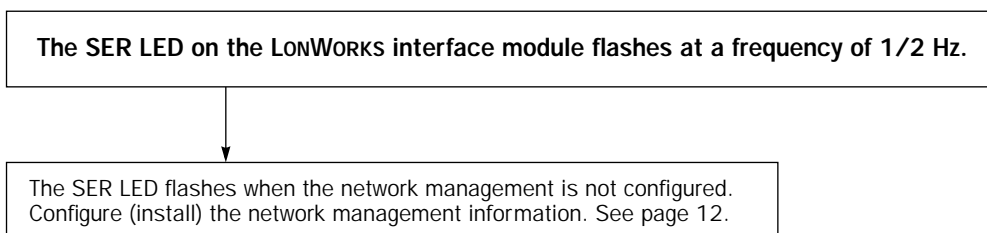
Troubleshooting Diagram 4



Troubleshooting Diagram 5



Troubleshooting Diagram 6



INDEX

- A**
 - ACKD 12
 - acknowledge service 12
 - application program 2
 - examples 18
 - modifying 13
- B**
 - bus topology 7
- C**
 - communication terminals SX5L 3
 - crimping tool 6
- D**
 - data
 - type 23
 - defined network variables 23
- E**
 - error data 17
 - expansion module ID 17
 - external interface file 2, 12
- F**
 - ferrule 6
 - flash memory 14
 - free topology 7
 - function area setting LonWorks node 10
- H**
 - header file 19
- I**
 - I/O counts 17
 - I/O pins 13, 15
 - initialization 18
 - codes 18
 - input network variables 23
 - internal structure LonWorks interface module 14
 - International Standard Organization 2
 - ISO 2
- L**
 - link registers for LonWorks network communication 8
 - LON 1
 - LonMaker 2, 12
 - LonTalk protocol 2
 - LonWorks 1
 - network system setup 3
- M**
 - memory map 14
 - message service 13
 - modifying application program 13
- N**
 - network
 - configuration information 2
 - management 2, 12
 - variables 2, 9, 23
 - Neuron chip 2
 - I/O pins 13, 15
 - NV 2
- O**
 - open system interconnection 2
 - opennet interface module 1, 4
 - OSI 2
 - output network variables 23
- P**
 - programming transmit/receive data using WindLDR 11
- R**
 - reading transmit data 22
 - receive data 11, 16
 - writing 21
 - registers 15
- S**
 - screwdriver 6
 - software version 17
 - specifications LonWorks interface module 5
 - starting operation 12
 - status LEDs 15
 - structure 23
 - SX5L communication terminals 3
 - system setup LonWorks network 3
- T**
 - terminator 7
 - transceiver 2
 - transmission time 9
 - transmit data 11, 16
 - reading 22
 - troubleshooting LonWorks network 25
- W**
 - WindLDR programming transmit/receive data 11
 - wiring LonWorks interface module 6
 - writing receive data 21
- X**
 - XIF 2, 12
 - No. 12