Ovation Link Controller (LC) User's Guide

Section **Title** **Page**

Section 1. Introduction

Section 2. Link Controller Hardware

Section 3. Link Controller Initialization

Table of Contents, Cont'd

Section 1. Introduction

1-1. Overview

This manual describes the configuration and operation of the Ovation Link Controller module in an Ovation Process Control System.

1-1.1. System Overview

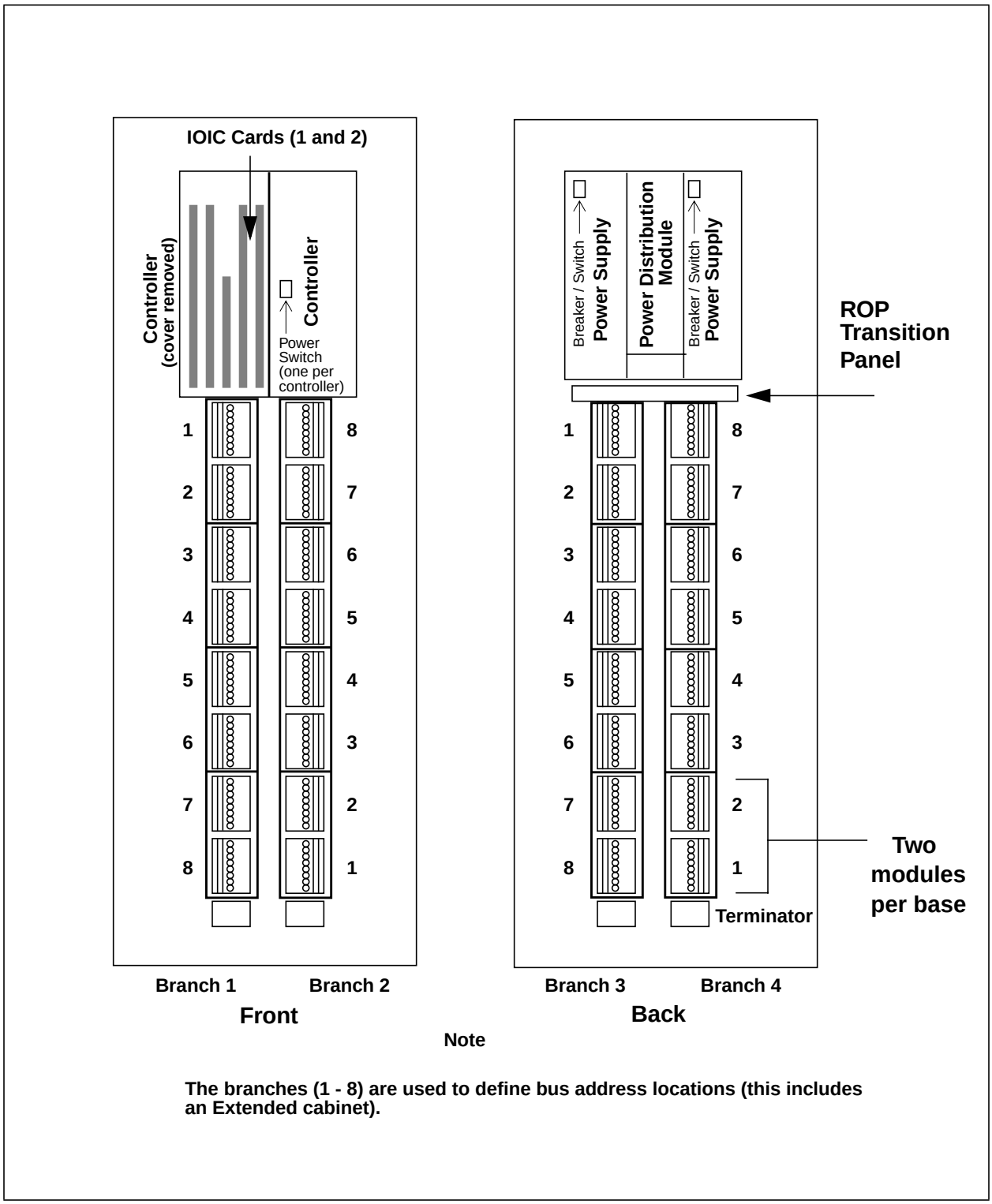
The Ovation Distributed Control System provides modulating control, sequential control, and data acquisition for a variety of system applications. This system consists of a configurable mix of functional Input/Output (I/O) modules (described in "Ovation I/O Reference Manual" (R3-1150)) that communicate on the I/O bus to the Ovation Controller.

An Ovation I/O module consists of an electronics module and a personality module. These modules are "plug-in" components with built-in fault tolerance and diagnostics. They are able, by combining different personality modules with electronics modules, to operate at a wide range of signals, and perform a multitude of functions.

The Ovation I/O modules are locked into Base Units (two modules per Base Unit). These Base Units are housed in the Controller cabinets where they are mounted on DIN rails and wired to the appropriate field devices (see example of Controller cabinet in Figure 1-1 and detailed cabinet descriptions in "Ovation Planning and Installation"(U3-1000).

The modular components consist of the following:

- Electronics module
- Personality module
- Base Units (containing the field terminations)



Note

The branches (1 - 8) are used to define bus address locations (this includes an Extended cabinet).

Figure 1-1. Ovation Controller Cabinet Example (illustrating modules)

1-1.2. Link Controller Overview

The Link Controller (LC) module provides the Ovation Controller with a serial data communications link to a third-party device or system. This communication is done via a serial RS-232, RS-422, or RS-485 data link.

Two serial ports are provided:

- The Applications port provides an interface to a third-party device or system.
- The Programming port provides an interface to an IBM-compatible personal computer's COM1 or COM2 serial port (see Section 2).

The LC processor is similar to an IBM-compatible personal computer and uses a modified IBM-PC Basic Input/Output System (BIOS) and the DOS operating system. Because the LC's architecture and operating system are compatible with an IBM-compatible personal computer, the LC can execute programs written using commercially available compilers.

Due to its flexible and open design, the LC can be used for various applications:

- The LC can provide online plant performance calculations.
- Programs designed to execute in an IBM-PC environment can communicate with the Controller and the Ovation system using the LC.
- The LC is applicable for use in CE Mark-certified systems.

1-2. Contents of this Document

This document is organized into the following sections:

- **Section 1. Introduction** describes the Ovation Link Controller module and its uses. This section also describes the contents of this manual and lists additional manuals that might be helpful to the user.
- **Section 2. Link Controller Hardware** provides information about the hardware and the wiring used to provide communication between an Ovation Controller and a third-party device via the Link Controller module.
- **Section 3. Link Controller Initialization** describes the procedure used to initialize the Ovation Link Controller module.
- **Section 4. Link Controller Programming and Operation** describes different programming approaches, and provides application notes that are helpful to the Link Controller user.
- **Appendix A. Link Controller Programming Examples** provides examples of application programs written for the Link Controller in C language.

1-3. Additional Reference Documentation

Table 1-1 lists additional reference documentation that may be helpful while using this document.

Table 1-1. Reference Documents

Document Number	Title	Description
R3-1100	Ovation Algorithms Reference Manual	Provides detailed descriptions of LC (SLC) algorithms.
R3-1140	Ovation Record Types Reference Manual	Discusses Ovation process points and record fields.
R3-1150	Ovation I/O Reference Manual	Provides information about all the Ovation I/O modules.
U3-1000	Planning and Installing Your Ovation System	Provides guidelines and procedures for planning and installing an Ovation system.
U3-1032	Ovation Controller User's Guide	Describes the configuration and use of the Ovation Controller.
U3-1040	Ovation Control Builder User's Guide	Provides general information on Control application programming, including procedures to add text algorithms to an application.

Section 2. Link Controller Hardware

2-1. Section Overview

This section describes the Ovation Link Controller module, the wiring, the optional jumpers, and the diagnostic LEDs used to provide communication between an Ovation Controller and a third-party device or system. This communication is done via a serial RS-232, RS-422, or RS-485 data link.

Prior to initializing and using the Link Controller module, verify that the hardware has been properly configured.

2-2. Module Description

This section describes the Ovation Link Controller module which consists of three parts (refer to Figure 2-1):

- Base Unit (containing the field terminations)
- Electronics Module
- Personality Module

2-2.1. Base Unit

The base unit (containing two terminal blocks) mounts onto a DIN rail. Each base unit accommodates two I/O modules. If you use only one I/O module, you still must use a base unit that contains two terminal blocks.

The footprint of the base unit (containing Electronic and Personality module) is:
27.9 cm Long (11 in)
12.7 cm Wide (5 in)
16.5 cm High (6.5 in)

2-2.2. Electronics Module

The Electronics module (configured by adding the appropriate Personality module) fits into the base unit. There is one Electronics module group for the Link Controller Module:

- **1C31166G01** provides for communication to a third-party device or system.

2-2.3. Personality Module

The Link Controller Personality module provides two (J1 and J2) male DB9 connectors for the desired serial links (see Figure 2-1).

There are two Personality module groups for the Link Controller Module. Select the appropriate group for the desired serial link.

Group One Personality Module

Group One (**1C31169G01**) provides for an RS-232 serial link.

- J1 Port is an RS-232 Programming Port which is used to interface to a Personal Computer (used for keyboard input, CRT display, and file transfer functions).
- J2 Port is an RS-232 Applications Port (in **CE Mark certified systems**, the application port cable must be less than 10 meters (32.8 ft)).

Group Two Personality Module

Group Two (**1C31169G02**) provides for an RS-485/RS-422 four-wire full duplex serial link.

- J1 Port is an RS-232 Programming Port which is used to interface to a Personal Computer.
- J2 Port is an RS-485 Applications Port (also may be used as an RS-422 serial link).
  - — The J2 Port transmitter has a software controlled tri-state enable input (TX-ENA). TX-ENA must be asserted (high or logic one) if the J2 Applications port is to transmit RS-485/RS-422 serial data. It is the responsibility of the Link Controller's application program to ensure that TX-ENA is asserted whenever serial data is transmitted from the J2 port.
  - — The application program must use Bit 1 of the Applications port UART Modem Control register (MCR) to control the state of the UART's RTS/ output. MCR is an eight-bit register mapped to the LC processor I/O address 03FCH.

— RTS/ directly controls the state of the TX-ENA signal as shown below:

MCR Bit 1	UART RTS/ Output State	TX-ENA State	RS-485 Transmitter Output State
0	1	0	Tri-stated (disabled)
1	0	1	Enabled

If two-wire half duplex communication is desired, connect the Link Controller's base unit terminal block input terminals to the output terminals (that is, connect RX- to TX-, and connect RX+ to TX+). See Figure 2-5.

Caution

If the LC's RS-485 output terminals are connected to the RS-485 input terminals, the RS-485 receiver is enabled whenever the applications program enables the RS-485 transmitter. This means that the RS-485 receiver can receive the data being transmitted by the RS-485 transmitter. The applications program must be able to handle this situation.

Installing Personality Module into Base Unit

The Personality module (used to configure the Electronics module) fits into the base unit beside the appropriate Electronics module. Note that the Personality module is installed in the base unit **first**. Then, the Electronics module is installed and interlocks with the Personality module. The blue corner latches on the Electronics module secure both modules into the base unit.

The wires from the customer field devices are connected to the terminal block in the base unit (see Figure 2-2).

The wiring connections to the terminal block for each combination of Electronics module and Personality module are printed on each Personality module and are illustrated in each module description in the following sections.

Note

Be sure that each wire opening in the terminal block is **fully open** before inserting the wire. This will ensure that the wire is clamped securely when the screw is tightened.

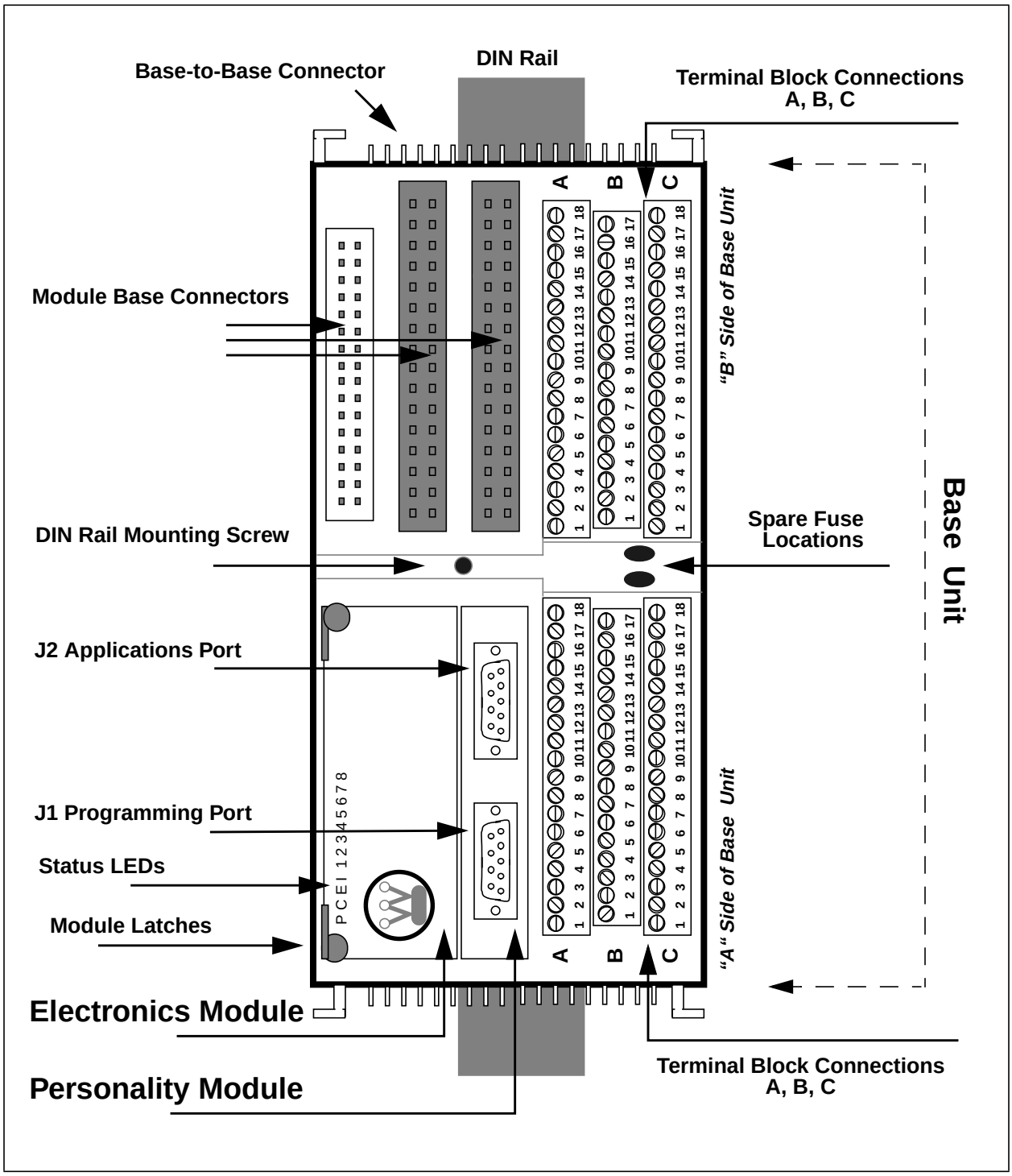


Figure 2-1. Link Controller Modules (Top View)

2-3. Module Specifications

Table 2-1. Link Controller Module Specifications

Description	Value
Number of serial ports	2
Applications serial port (In **CE Mark certified systems**, the Link Controller must be used only in communication fault tolerant systems. That is, communication retries must be tolerated.)	Signals are galvanically isolated from logic common. This port may be interfaced via the Personality module's J2 male DB9 connector (refer to Table 2-2 or Table 2-3) OR via the Base unit terminal block (refer to Figure 2-2). (1C31169G01) RS-232 signal levels: Note that standard IBM PC/AT COM1 port pin assignments are present at the J2 DB9 connector OR (1C31169G02) RS-485/RS-422 signal levels: four wire link, RS-485/RS-422 transmitter/receiver. Resistor terminations are user selectable.
Applications serial port baud rate	300, 600, 1200, 2400, 4800, 9600, or 19200
Applications serial port dielectric isolation: (Port to logic)	1000 V AC peak/DC
Programming serial port	Signals are not galvanically isolated from logic common. RS-232 signal levels (TX/ and RX/) This port must be interfaced via the Personality module's J1 male DB9 connector (refer to Table 2-7)
Programming serial port baud rate	19200 baud (default) OR 9600 baud (user selectable option)
Processor	80C186EC 16-bit
Program RAM	640 Kbytes
Dual port RAM	4 Kwords (data interface between the I/O bus and the Link Controller module)
Module power	3 W typical 4 W maximum
Operating temperature range	0 to 60°C (32°F to 140°F)
Storage temperature range	-40°C to 85°C (-40°F to 185 °F)
Humidity (non-condensing)	0 to 95%

2-4. Interface Connections

2-4.1. Terminal Block Wiring Information

The field connection to the LC module can be made either at the nine pin J2 connector of the personality module or at the terminal block of the base unit. It is common for RS-422 and RS-485 communication, because the cable length is typically longer, for the connection to be made as field wiring rather than as a pre-manufactured cable. The terminal block connection is convenient for these cases.

Each Personality module has a simplified wiring diagram label on its side, which appears above the terminal block. This diagram (see Figure 2-2) indicates how the wiring from the field is to be connected to the terminal block in the base unit. The following table lists and defines the abbreviations used in the diagram.

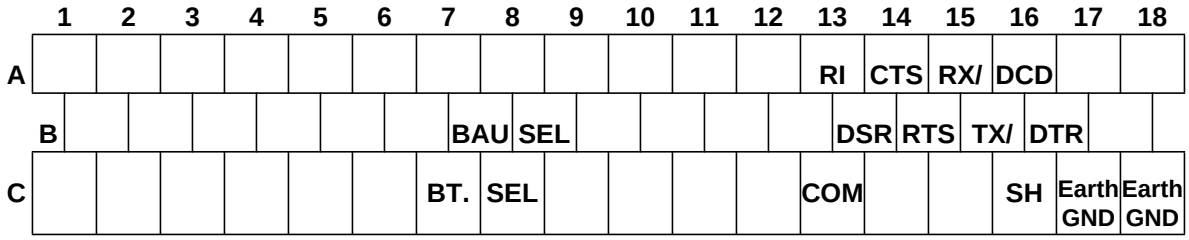
Abbreviation	Definition
BAU	Programming port baud rate select jumper
BT.	Link Controller boot-up option select jumper
COM *	Applications port signal common reference
CTS *	RS-232 Clear To Send (Input)
DCD *	RS-232 Data Carrier Detect (Input)
DSR *	RS-232 Data Set Ready (Input)
DTR *	RS-232 Data Terminal Ready (Output)
Earth GND *	Application port cable assembly shield connection (for locally grounded shields). Connects to the Base Unit DIN rail.
RES	A14 = RS-485/RS-422 receive data termination resistor (tie to RX+ (A15) if used) B14 = RS-485/RS-422 transmit data termination resistor (tie to TX+ (B15) if used)
RI *	RS-232 Ring Indicator (Input)
RTS *	RS-232 Request To Send (Output)
RX/ *	RS-232 Receive Data (Input)
RX+ *	RS-485/RS-422 Receive Data (Input)
RX- *	RS-485/RS-422 Receive Data (Input)
SEL	B8 = Programming port baud rate select jumper C8 = Link Controller boot-up option select jumper

Abbreviation	Definition
SH *	Applications port cable assembly shield connection (for remotely grounded shields) (for non-CE Mark certified systems)
TX/ *	RS-232 Transmit Data (Output)
TX+ *	RS-485/RS-422 Transmit Data (Output)
TX- *	RS-485/RS-422 Transmit Data (Output)
* Applications port	

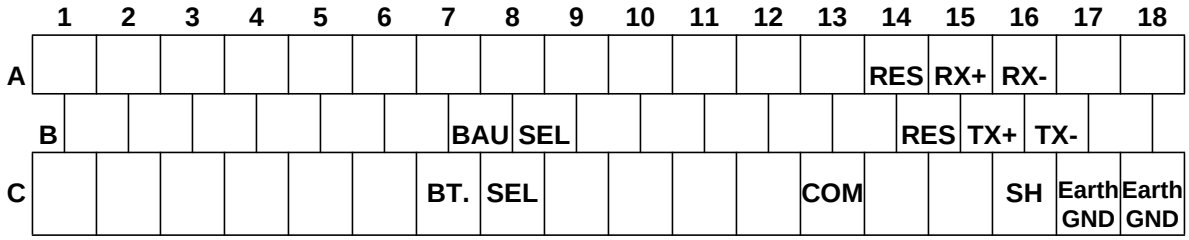
Notes

1. Do not use unmarked terminal block locations.
2. Shield terminals (SH) are not connected in **CE Mark certified systems**.

RS-232 Applications: (1C31169G01)



RS-485/RS-422 Four Wire Applications: (1C31169G02)



Notes

1. COM is the galvanically isolated common terminal of the Applications Port.
2. Use an overall shield conductor for the Applications Port cable:
   A. To ground the cable shield locally, connect the cable shield to terminal C17 (Earth Ground).
   B. To ground the cable shield remotely, connect the cable shield to terminal C16 (SH).
3. Connect a wire jumper for the following:
   A. To boot-up from an external IBM-compatible PC, connect a jumper between SEL and BT.
   B. To set Programming Port baud rate to 9600 (default is 19200), connect a jumper between SEL and BAU.
4. For RS-485/RS-422 four wire applications, connect a wire jumper for the following:
   A. To use a transmitter termination resistor, connect a jumper between RES (B14) and TX+ (B15).
   B. To use a receiver termination resistor, connect a jumper between RES (A14) and RX+ (A15).

   For a four wire RS-485/RS-422 communications network, a receiver termination resistor is recommended if the receiver is located at the end of the communications link. A transmitter termination resistor is recommended if the transmitter is at the end of the communications link and if the transmitter is tri-stated (disabled) when data is not being transmitted. Otherwise, a transmitter termination resistor is not necessary.

Figure 2-2. Terminal Block Connections for the Link Controller Personality Module

2-4.2. Applications Port Terminal Block Field Connections

Non-CE Mark Certified Systems

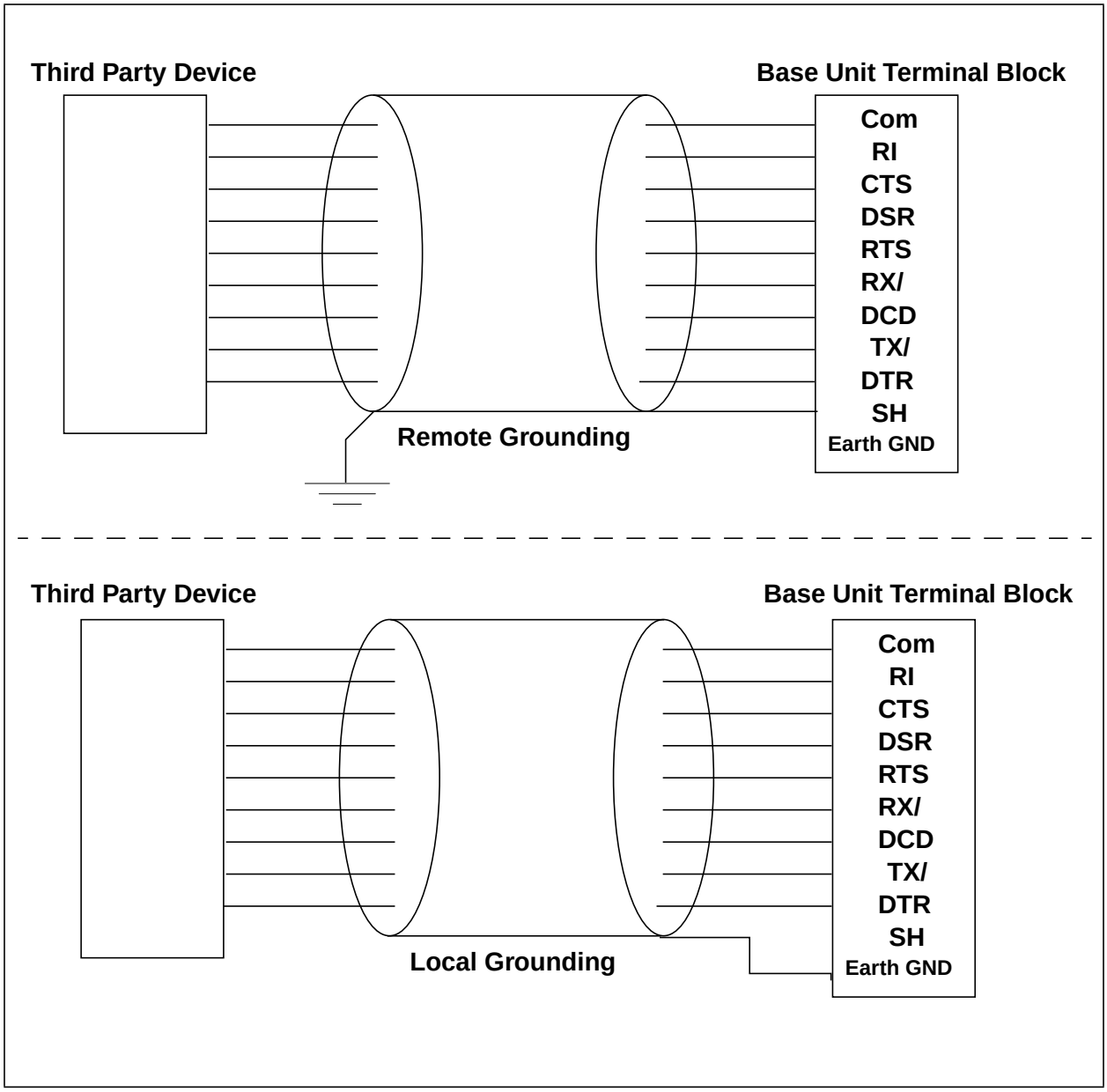


Figure 2-3. Terminal Block RS-232 Interface (1C31169G01) (Non-CE Mark)

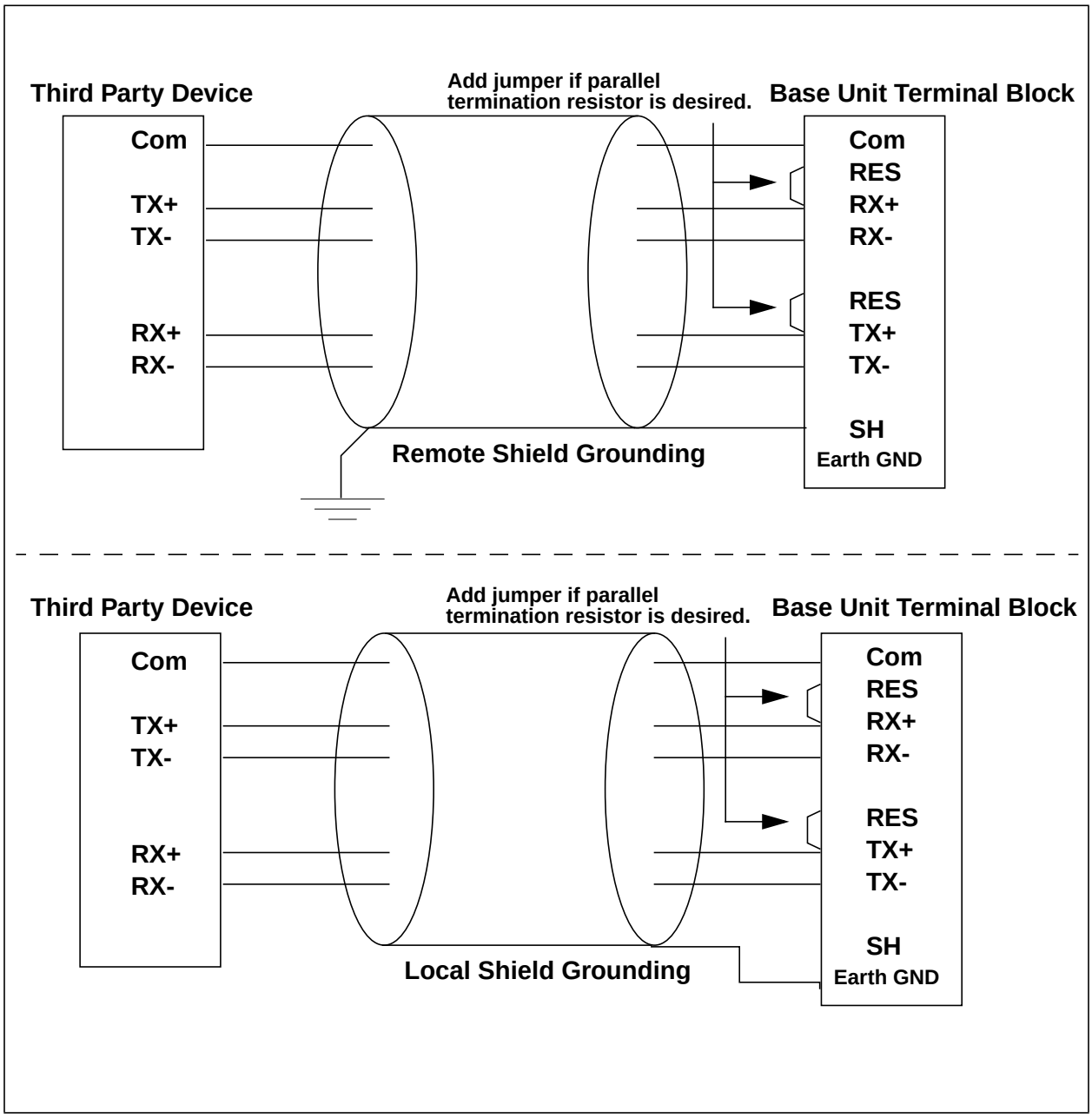
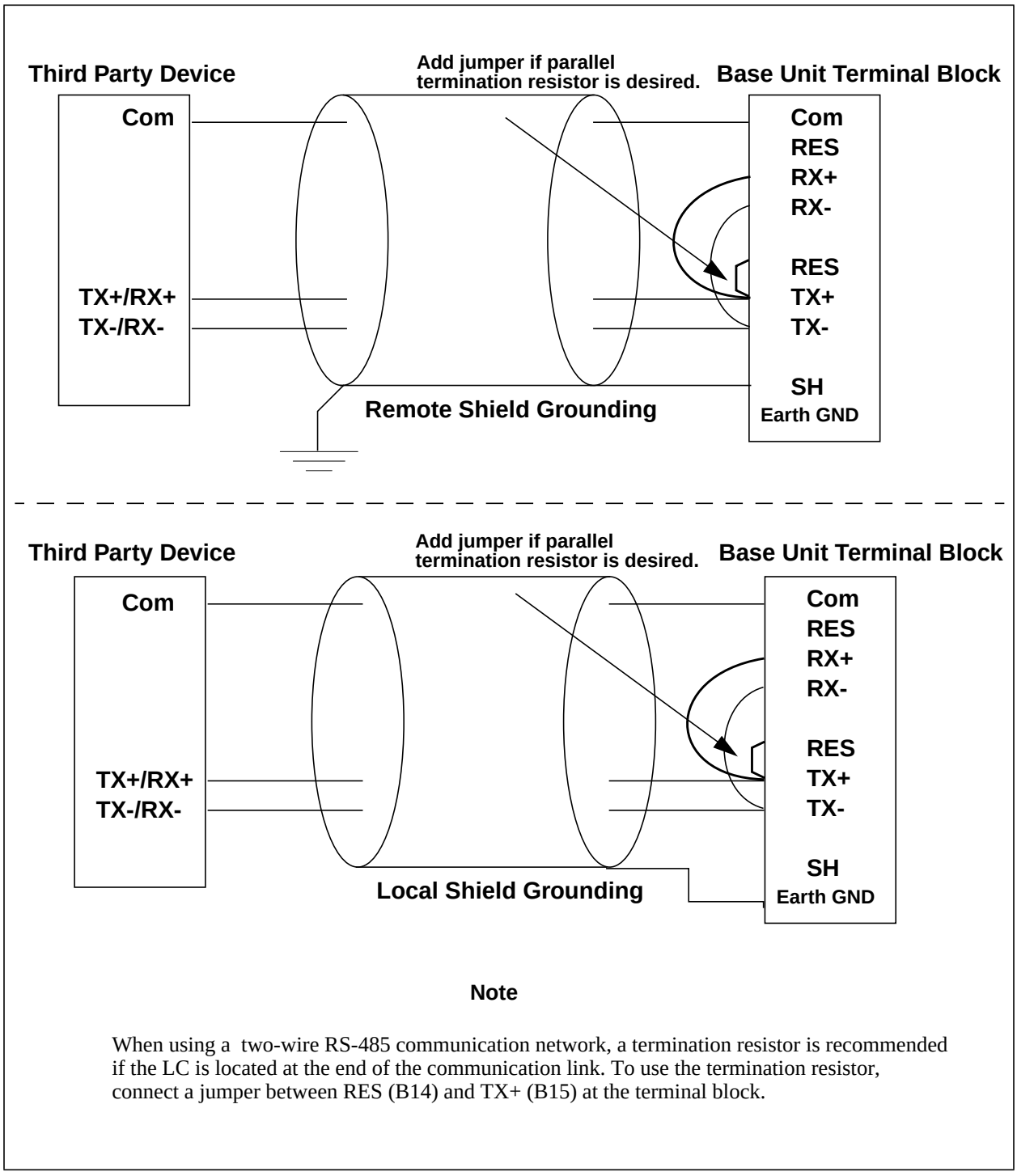


Figure 2-4. Terminal Block RS-485/RS-422 Four-Wire Serial Interface (1C31169G02) (Non-CE Mark)

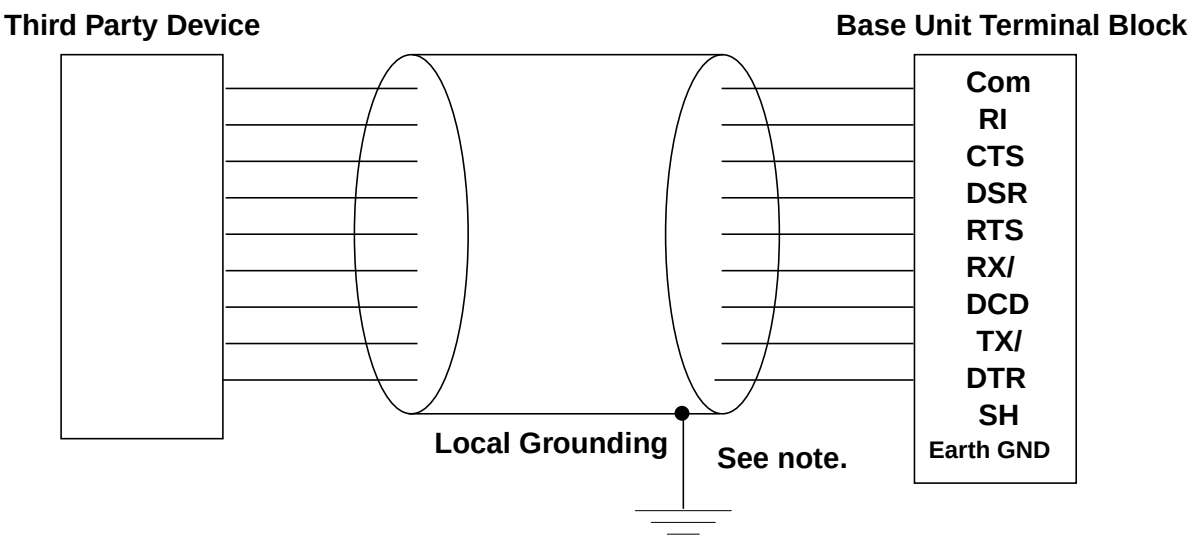


Note

When using a two-wire RS-485 communication network, a termination resistor is recommended if the LC is located at the end of the communication link. To use the termination resistor, connect a jumper between RES (B14) and TX+ (B15) at the terminal block.

Figure 2-5. Terminal Block RS-485 Two-Wire Serial Interface (1C31169G02) (Non-CE Mark)

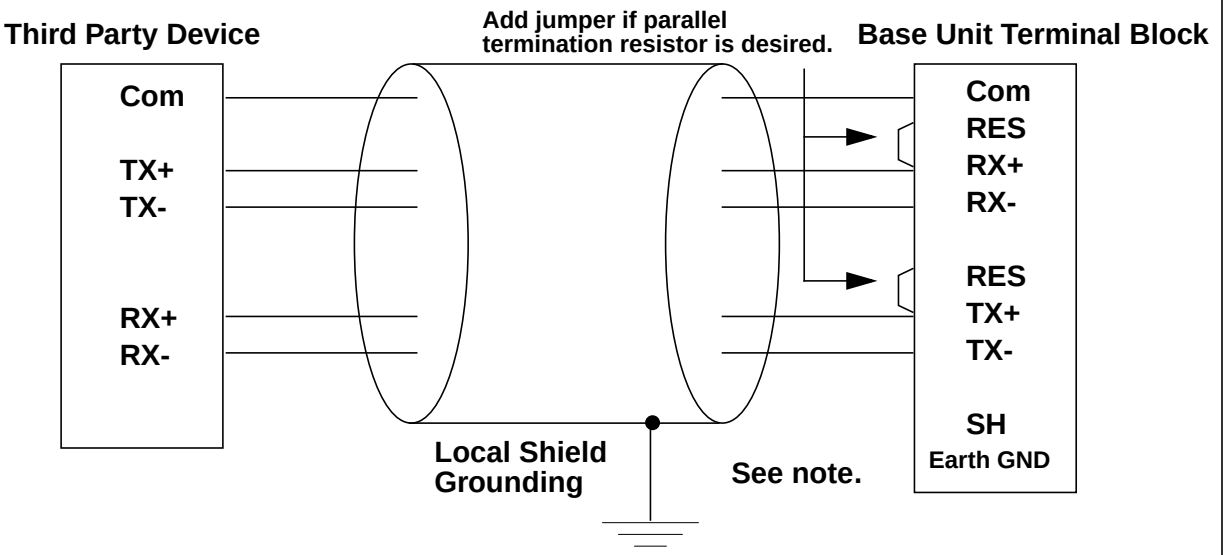
CE Mark Certified Systems



Note

All field wiring must be braid-shielded and grounded at the entry point of the cabinet using the recommended hardware (refer to "Cable Guidelines" in U3-1000).

Figure 2-6. Terminal Block RS-232 Interface (1C31169G01) (CE Mark)



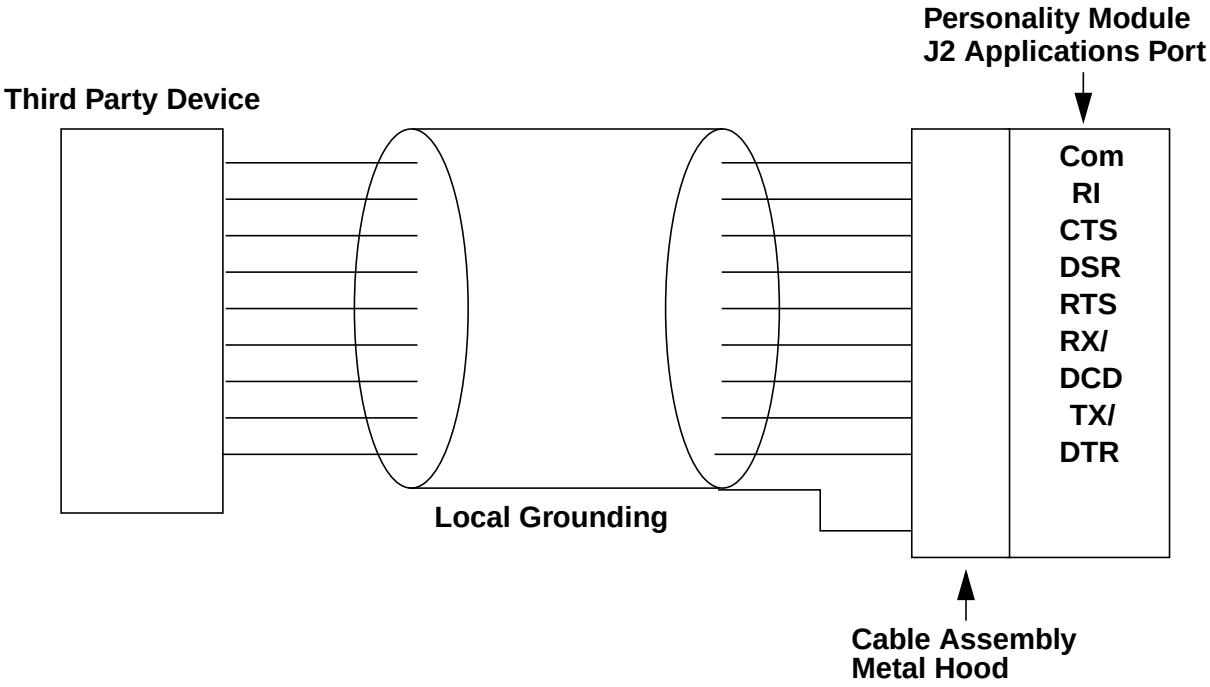
Note

All field wiring must be braid-shielded and grounded at the entry point of the cabinet using the recommended hardware (refer to "Cable Guidelines" in U3-1000).

Figure 2-7. Terminal Block RS-485/RS-422 Four-Wire Serial Interface (1C31169G02) (CE Mark)

2-4.3. Applications Port (J2) Field Connections

Non-CE Mark Certified Systems



Note

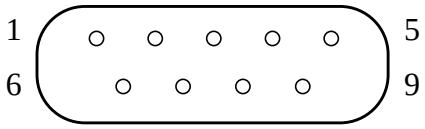
If the Personality module's J2 RS-232 Applications port DB9 connector is used in place of the base unit terminal block, only local shield grounding may be implemented. The cable assembly metal hood must be in electrical contact with the cable's shield. Refer to Table 2-2 for pin assignments.

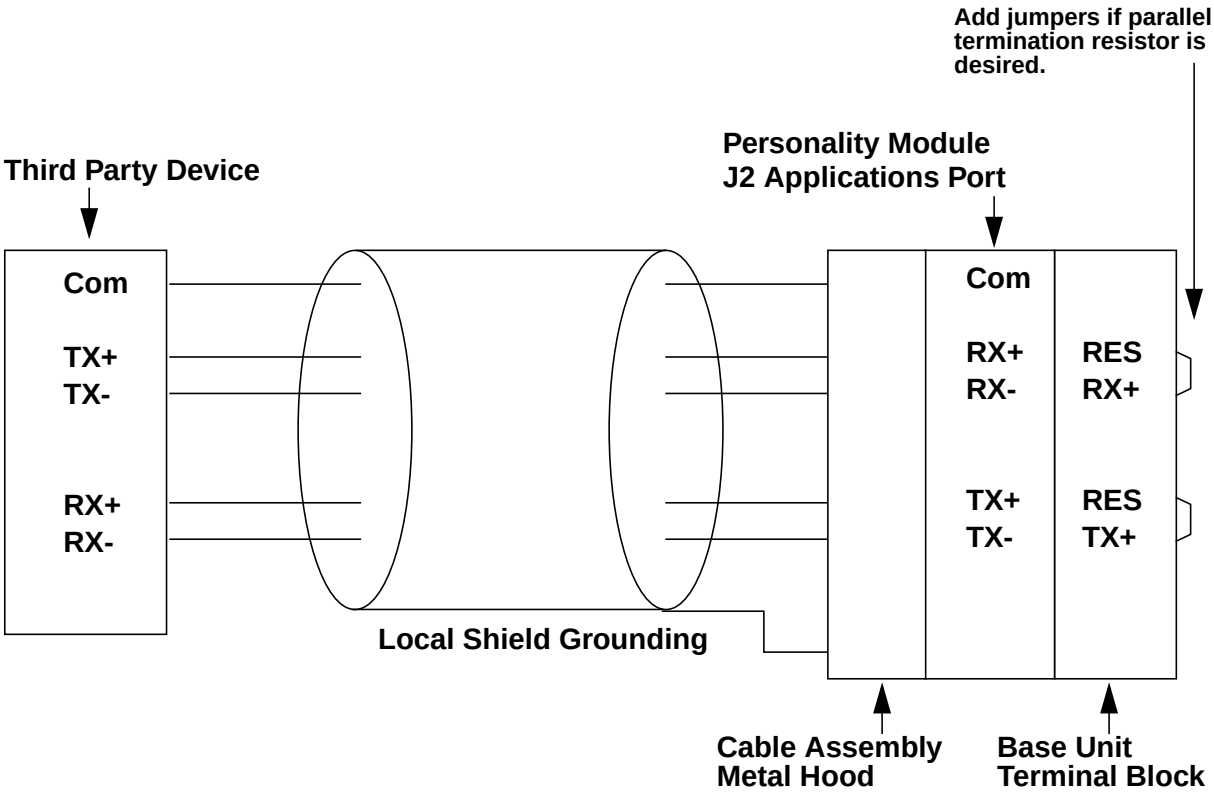
Figure 2-8. J2 RS-232 Interface (1C31169G01) (Non-CE Mark)

Table 2-2. Pin Assignments for J2 Applications Port RS-232 Interface

Pin Number	Signal Name (Function)	Signal Direction
1	DCD (Data Carrier Detect)	Input
2	RX/ (Receive Data)	Input
3	TX/ (Transmit Data)	Output
4	DTR (Data Terminal Ready)	Output
5	Com (Isolated Common)	
6	DSR (Data Set Ready)	Input
7	RTS (Request to Send)	Output
8	CTS (Clear to Send)	Input
9	RI (Ring Indicator)	Input

Top View of J2 Connector





Note

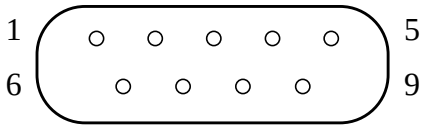
If the Personality module's J2 RS-485/RS-422 Applications port DB9 connector is used in place of the base unit terminal block, only local shield grounding may be implemented. The cable assembly metal hood must be in electrical contact with the cable's shield.
If parallel termination resistors are desired, jumpers must be added to the terminal block (as shown above). Refer to Table 2-3 for pin assignments.

Figure 2-9. J2 RS-485/RS-422 Four-Wire Serial Interface (1C31169G02) (Non-CE Mark)

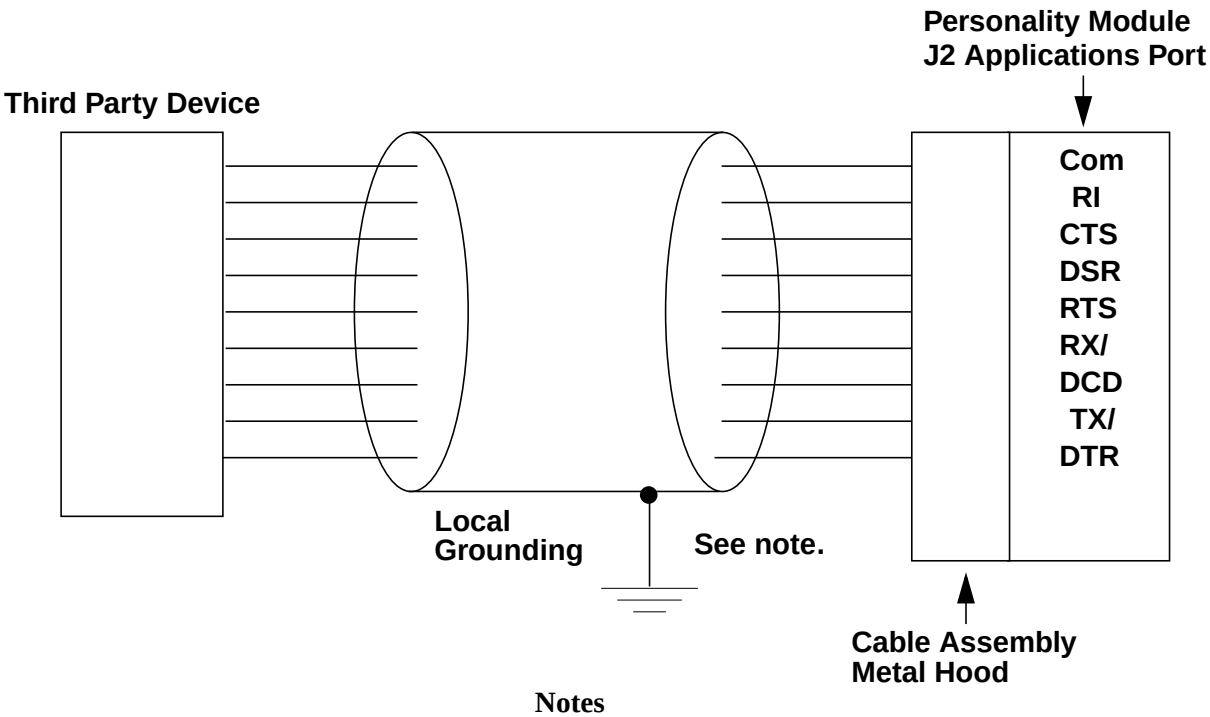
Table 2-3. Pin Assignments for J2 Applications Port RS-485/RS-422 Four-Wire Interface

Pin Number	Signal Name (Function)	Signal Direction
1	RX-	Input
2	RX+	Input
3	TX+	Output
4	TX-	Output
5	Com (Isolated Common)	
6		
7		
8		
9		

Top View of J2 Connector



CE Mark Certified Systems



Notes

If the Personality module's J2 RS-232 Applications port DB9 connector is used in place of the base unit terminal block, only local shield grounding may be implemented. The cable assembly metal hood must NOT be in electrical contact with the cable's shield. Refer to Table 2-4 for pin assignments.

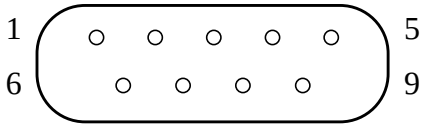
The cable MUST be braid-shielded and grounded at the entry point of the cabinet using the recommended hardware (refer to "Cable Guidelines" in U3-1000).

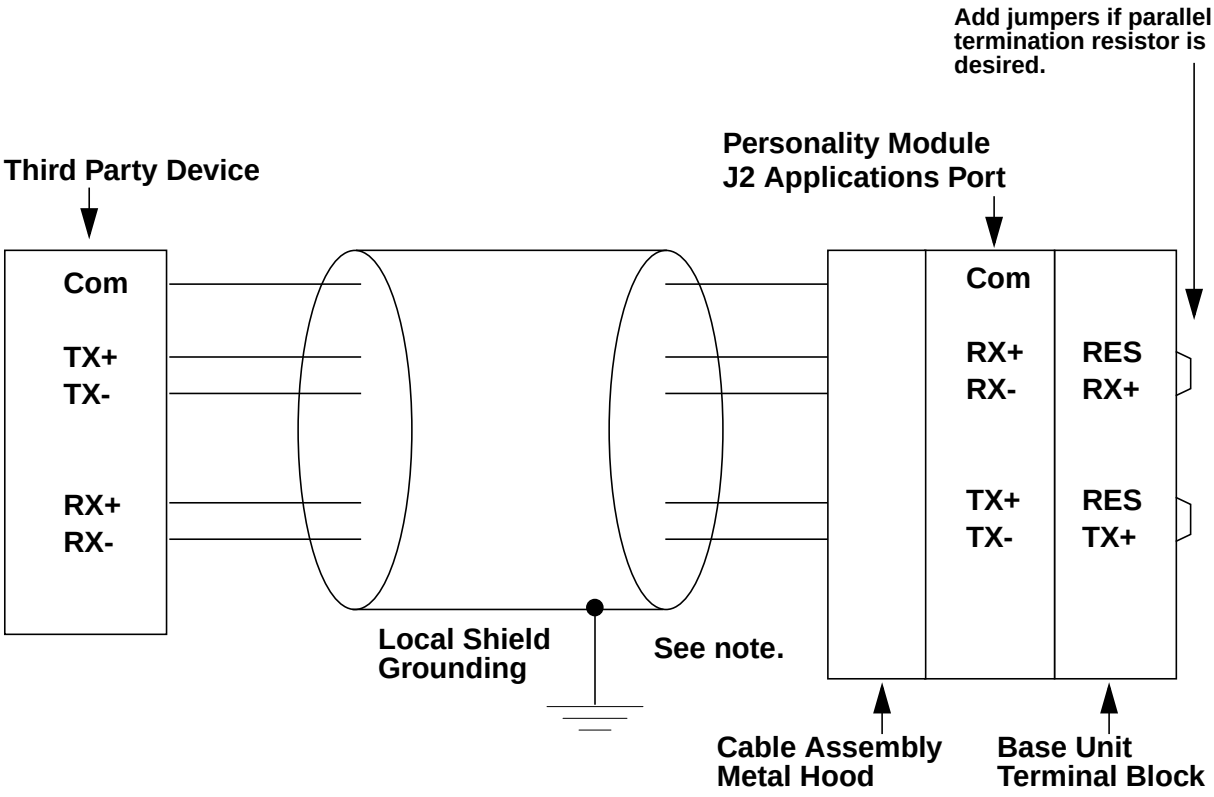
Figure 2-10. J2 RS-232 Interface (1C31169G01) (CE Mark)

Table 2-4. Pin Assignments for J2 Applications Port RS-232 Interface

Pin Number	Signal Name (Function)	Signal Direction
1	DCD (Data Carrier Detect)	Input
2	RX/ (Receive Data)	Input
3	TX/ (Transmit Data)	Output
4	DTR (Data Terminal Ready)	Output
5	Com (Isolated Common)	
6	DSR (Data Set Ready)	Input
7	RTS (Request to Send)	Output
8	CTS (Clear to Send)	Input
9	RI (Ring Indicator)	Input

Top View of J2 Connector





Notes

If the Personality module's J2 RS-485/RS-422 Applications port DB9 connector is used in place of the base unit terminal block, only local shield grounding may be implemented. The cable assembly metal hood must NOT be in electrical contact with the cable's shield. If parallel termination resistors are desired, jumpers must be added to the terminal block (as shown above). Refer to Table 2-5 for pin assignments.

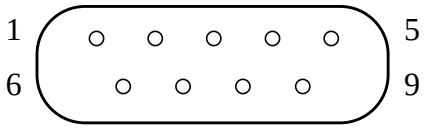
The cable MUST be braid-shielded and grounded at the entry point of the cabinet using the recommended hardware (refer to "Cable Guidelines" in U3-1000).

Figure 2-11. J2 RS-485/RS-422 Four-Wire Serial Interface (1C31169G02) (CE Mark)

Table 2-5. Pin Assignments for J2 Applications Port RS-485/RS-422 Four-Wire Interface

Pin Number	Signal Name (Function)	Signal Direction
1	RX-	Input
2	RX+	Input
3	TX+	Output
4	TX-	Output
5	Com (Isolated Common)	
6		
7		
8		
9		

Top View of J2 Connector



2-4.4. Generic Cabling Schemes

Once the pin-out of the other device's serial port connector has been determined, a cable can be made to connect the device to the J2 Port of the LC module. The module's transmit signal (RS-232) or signal pair (RS-485/RS-422) must be connected to the receive signal (RS-232) or signal pair (RS-485/RS-422) of the other device. Likewise, the LC receive signal or signal pair is connected to the transmit of the other device.

Generic RS-232 and RS-485/RS-422 cables are shown in Figure 2-12 and Figure 2-13. Many manufacturers use DB-9 or DB-25 connectors for the serial connection. Terminal blocks instead of connectors are also common for RS-485/RS-422.

These figures show no pin numbers for the other device since there is much variation among manufacturers.

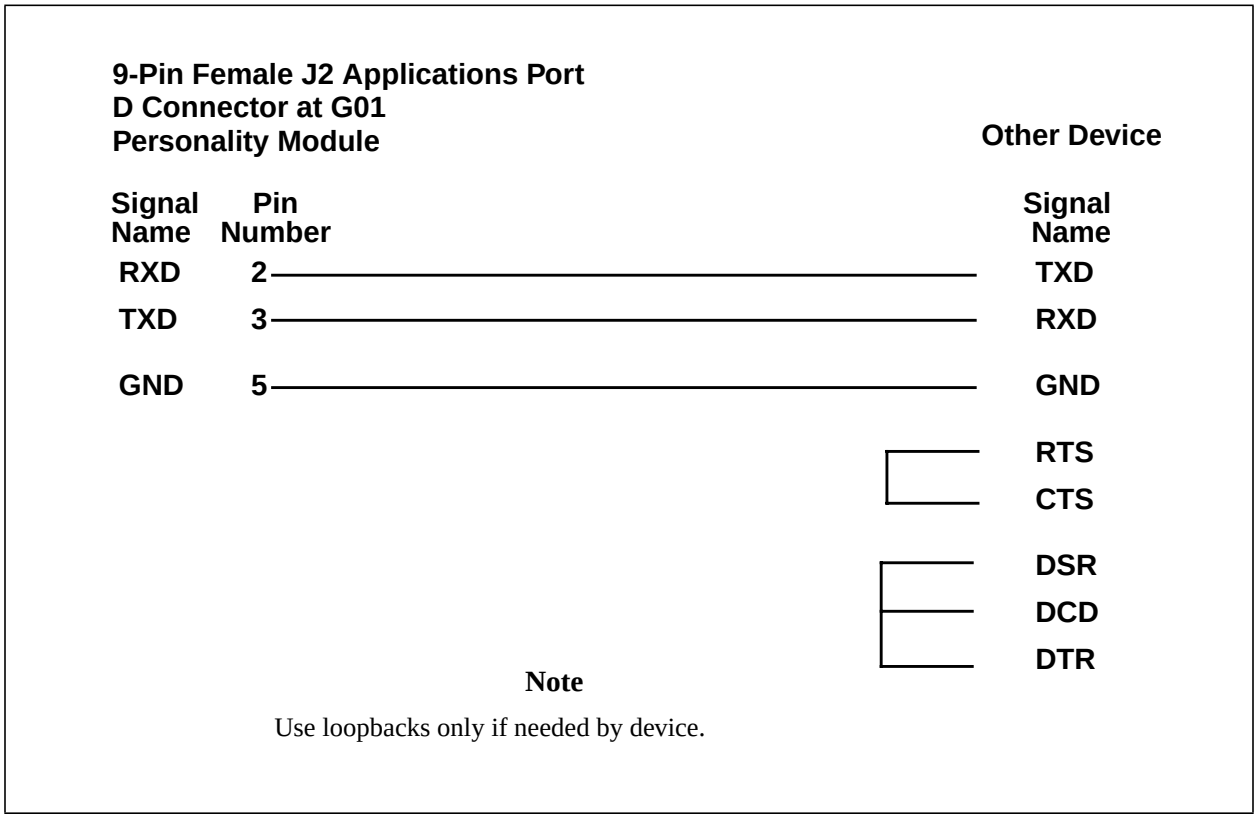


Figure 2-12. Generic RS-232 Cabling Scheme

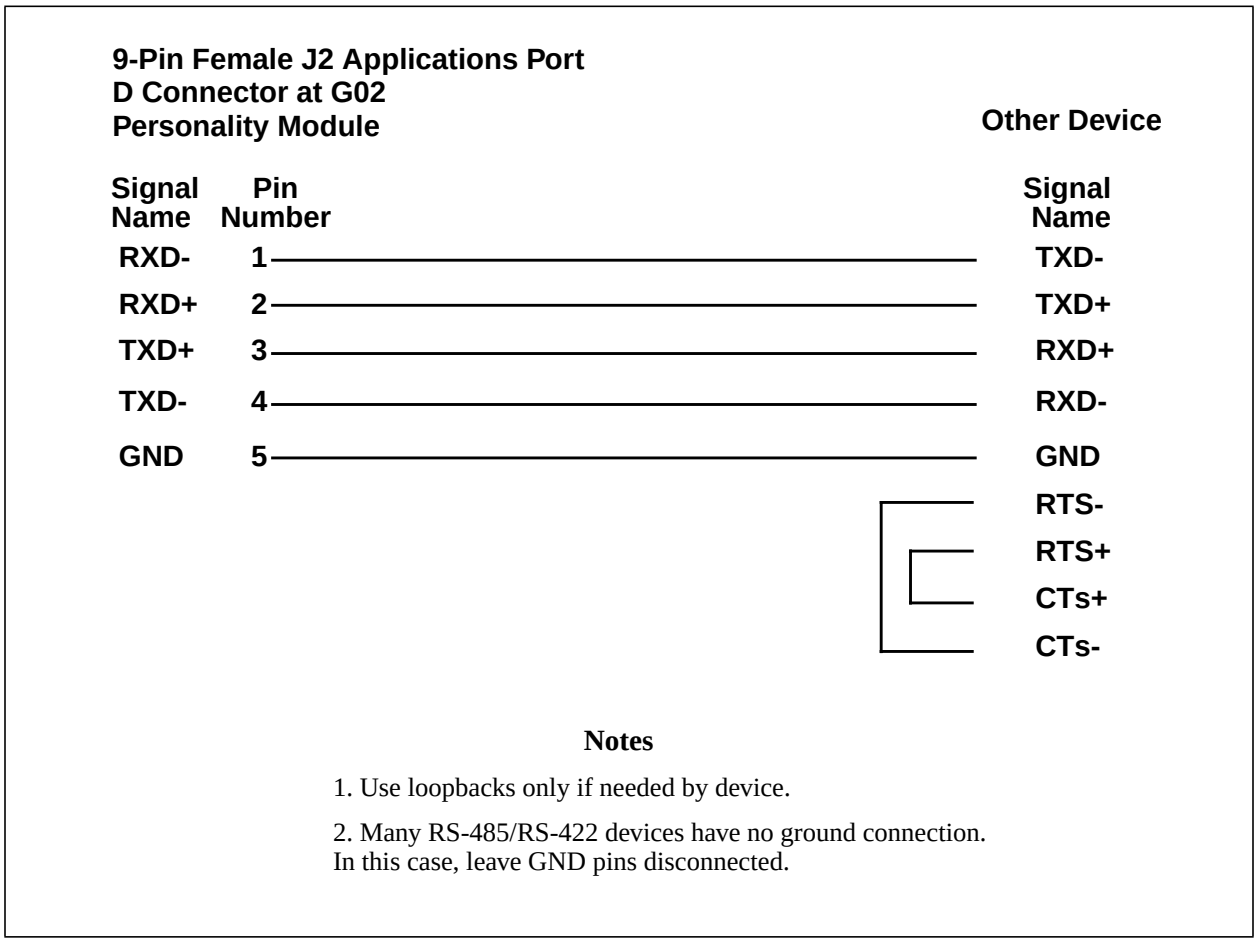


Figure 2-13. Generic RS-485/RS-422 LC Cabling Scheme

2-4.5. Cable Selection

Cable for RS-232 serial communication should be multiple conductor with shield. The shield drain wire should be used for the ground connection.

Cable for RS-485/RS-422 serial communication should be multiple twisted pairs, overall or individually shielded. Each signal's two differential connections should be made within a twisted pair; signals should not cross pairs.

The RS-485/RS-422 Group 2 personality module includes terminating resistors which can be placed across each signal pair by jumpering the base unit terminal block between B14 and B15 for transmit and between A14 and A15 for receive. Refer to Figure 2-2.

Some field devices will have resistors which can be switched in. For other devices, terminating resistors may need to be added. A typical value is 120 ohms across the + and - pins. In particularly noisy environments, the resistor can be split into 60 ohms from + to ground and 60 ohms from - to ground.

Refer to the "Cable Guidelines" section in U3-1000 for detailed information about cable usage and installation.

2-5. Jumpers

Wire jumpers may be installed in the Base Unit terminal block to select different Link Controller options. Table 2-6 describes the options and the jumpers used to select those options.

Table 2-6. LC Jumper Options

Option	Jumper Positions	Description
Baud Rate (for J1 Programming Port)	B7 and B8 connected	Baud rate = 9600
	B7 and B8 not connected	Baud rate = 19200 (default)
Boot LC module	C7 and C8 connected	Can boot from an external computer and load DOS (through the Programming Port). Typically, these jumpers are not used since DOS is installed on the LC module at the factory. However, if the module's RAM disk becomes corrupted, the jumpers must be installed so that the module can be reloaded with DOS and rebooted.
	C7 and C8 not connected	Can boot from LC Flash memory (by removing module from Base Unit and then replacing it) (default)
Enable Pmod (1C31169G02) J2 Applications Port RS-485/RS-422 **transmitter** termination resistor	B14 and B15 connected	Termination resistor enabled
	B14 and B15 not connected	Termination resistor isolated (default)
Enable Pmod (1C31169G02) J2 Applications Port RS-485/RS-422 **receiver** termination resistor	A14 and A15 connected	Termination resistor enabled
	A14 and A15 not connected	Termination resistor isolated (default)

2-6. External Personal Computer Connections

A Personal Computer is connected to the Link Controller module (as shown in Figure 2-14) in order to perform the following:

- Initialize the LC module.
- Provide keyboard input.
- Provide CRT display.
- File transfer functions.

2-6.1. Computer Requirements

The Personal Computer used to interface to the LC module must meet the following requirements:

- Must be an IBM-compatible computer.
- Must have a DOS 5.0 Operating System, or have a DOS 5.0 bootable floppy available.
- Must be equipped with a 3-1/2 inch floppy drive.
- Must have a COM1 serial port.

2-6.2. Cable Requirements

The cable used to connect the LC module to a Personal Computer must meet the following requirements:

- The end that plugs into the LC module must have a female DB9 connector with 4-40 jack screws.
- The end that plugs into the PC must have an appropriate connector (DB9 or DB25) that is compatible with the computer’s serial port connector.

2-6.3. Jumper Requirements

Typically, the LC module has DOS installed at the factory on its RAM disk and does not need to have it installed from an external computer. Therefore, the LC module can be rebooted from its internal Flash memory by either removing the module from the Base Unit and then replacing it, or by using the key sequence **Control-Shift-Delete** at an external Personal Computer which has already established command.

However, if the LC module needs to have DOS installed from an external computer (for example, if the RAM disk is corrupted), a jumper must be inserted between **C7** and **C8** in the Base Unit terminal block in order to be able to install DOS and to boot the LC module (see Section 2-5 for additional jumper information).

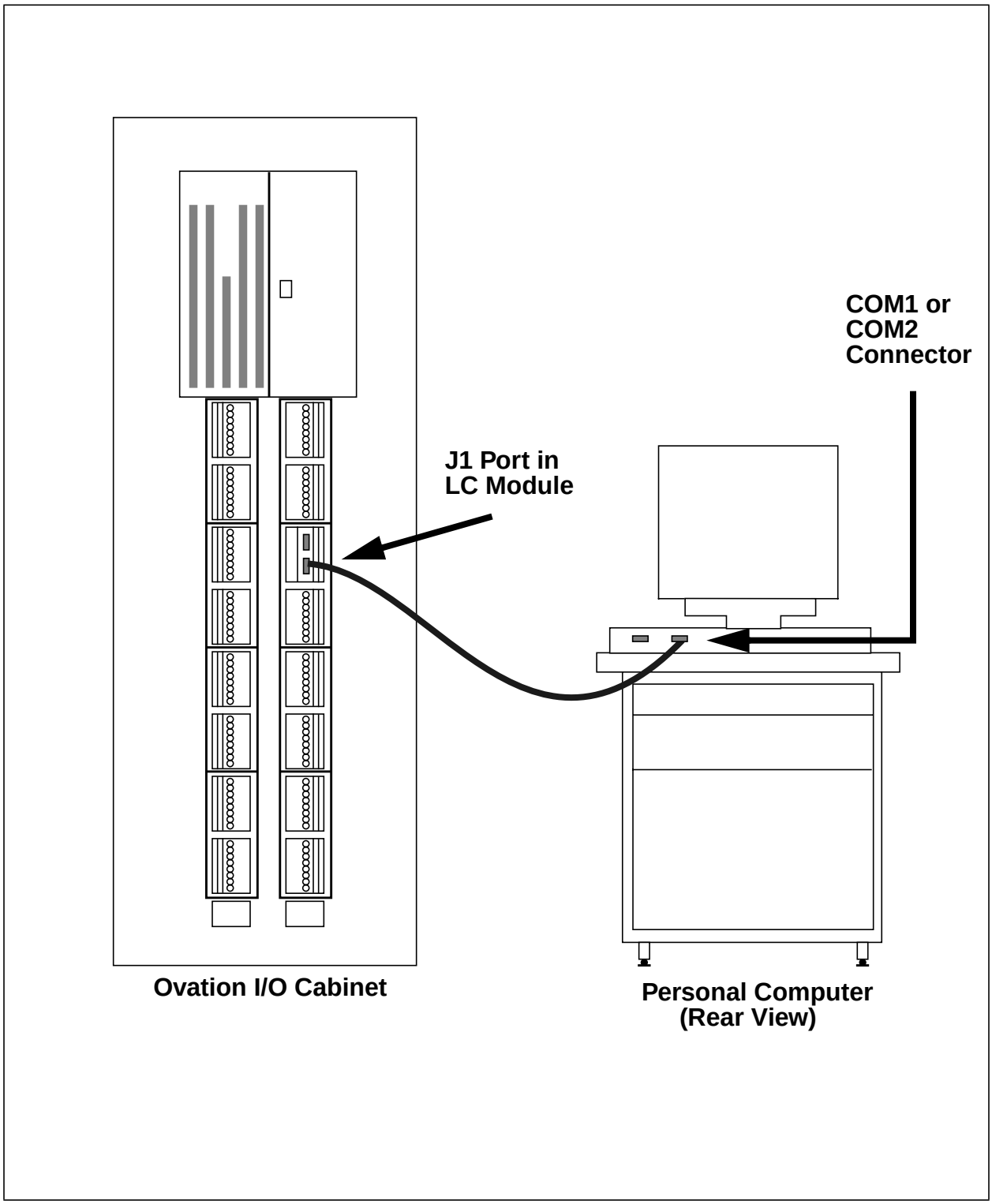
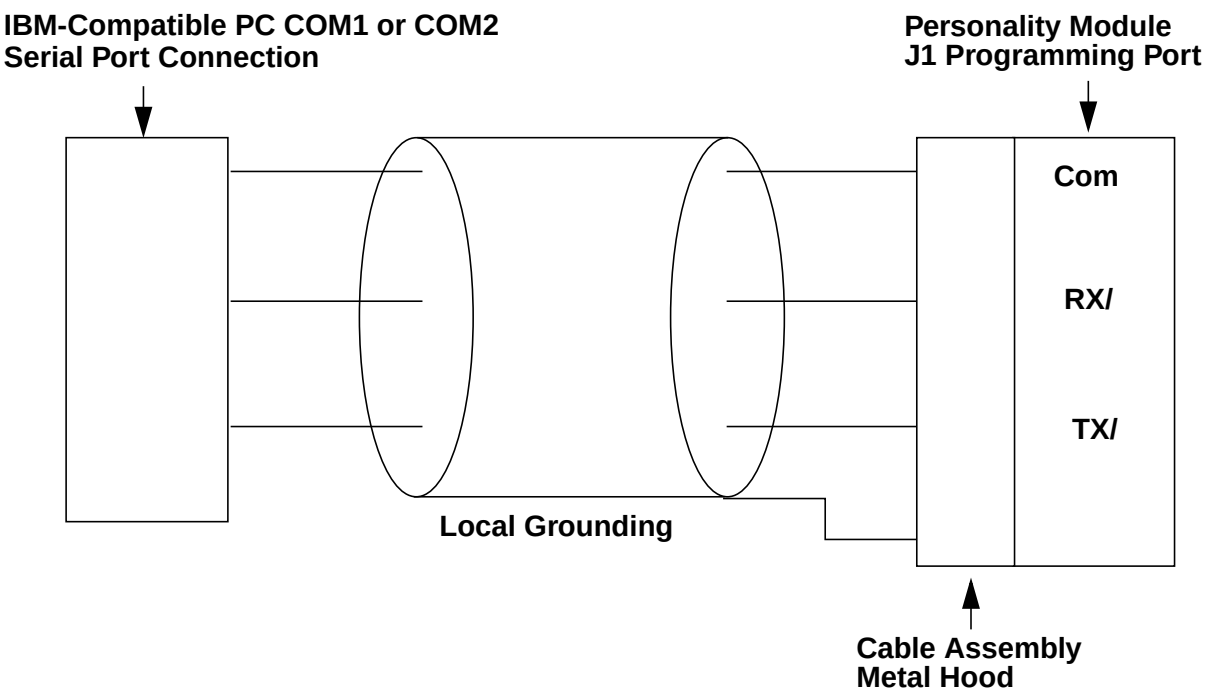


Figure 2-14. Connection Between Link Controller Module and Personal Computer

2-6.4. Programming Port (J1) Signal Connections



Notes

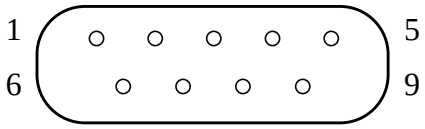
1. For the Personality module's J1 Programming port, only local shield grounding may be implemented. The cable assembly metal hood must be in electrical contact with the cable's shield. Refer to Table 2-7 for pin assignments.
2. This port cannot be a permanent connection in a CE Mark certified system.

Figure 2-15. J1 Programming Port Interface (1C31169G01/1C31169G02) (Non-CE Mark)

Table 2-7. Pin Assignments for J1 Interface

Pin Number	Signal Name (Function)	Signal Direction
1		
2	RX/ (Receive Data)	Input
3	TX/ (Transmit Data)	Output
4		
5	Com (Logic Common)	
6		
7		
8		
9		

Top View of J1 Connector



2-7. Link Controller Address Locations

An Ovation I/O module has 16 address locations, but typically does not use all 16 addresses.

There are four possible address locations in each I/O module that are reserved for special use. Three of these addresses provide configuration (Write) and status (Read) information:

- Address word 13 (D in Hex) is present for every module and is used for configuration and status (see Table 2-8). The module status provides diagnostic information that is read by the Controller when it is on-line. The status register can be read by using the Point Information window at an Ovation Operator Station (see the Bit Pattern Field on the Hardware Tab).
- Address word 14 (E in Hex) is used as a secondary or expansion configuration register and is only used when needed.
- Address word 12 (C in Hex) is used for reporting point specific fault information and optionally as an expansion configuration register.
- Address word 15 (F in Hex), is used for the module Electronic ID information. This location and its use is identical for all modules. Refer to "Ovation I/O Reference Manual" (R3-1150) for information about the Electronic ID.

Table 2-8 lists the configuration information for the Link Controller module.

Table 2-8. Link Controller Configuration Register (Address 13 or D in Hex)

Bit	Configuration Register (Write)	Status Register (Read)
0	Configure Module (1 = configure; 0 = unconfigure, causing an attention status)	Configured Module (1 = configured; 0 = unconfigured, causing an attention status)
1	Force Error (1 = force an attention status to be read by Controller; 0 = no forced error)	Forced Error (1 = forced error to be read by Controller; 0 = no forced error)
2	Not defined	Applications time-out/
3	Not defined	Not defined
4	Not defined	Not defined
5	Not defined	Not defined
6	Not defined	Internal Error
7	Not defined	External Error
8	Not defined	Not defined
9	Reserved	Reserved
10	Not defined	Not defined
11	Not defined	Not defined

Table 2-8. Link Controller Configuration Register (Address 13 or D in Hex)

12	Not defined	Not defined
13	Not defined	Not defined
14	Not defined	Not defined
15	Not defined	Not defined

Bit 0: After module reset, the Controller must write to the module's configuration register and set bit 0. Then the Controller can access other Serial Link module registers with offsets less than 12. If the Controller has not set bit 0, any Controller attempts to access address offsets less than 12 will yield an Attention status.

Bit 1: If the Controller sets this bit, any Controller attempts to access address offsets less than 12 will yield an Attention status.

Bits 2 in the status register is reset if the internal application program is not present or is not operating properly. It is set if an internal application program is present and active.

Bits 3 through **15** are not defined in the Configuration register.

Bit 6 in the Status register is asserted by the FLC board's processor when it wants to illuminate the LLC board's Internal Error LED.

Bit 7 in the Status register is asserted by the FLC board's processor when it wants to illuminate the LLC board's External Error LED.

2-8. Diagnostic LEDs

The Link Controller has 12 LEDs on the Personality module. These LEDs are used to show the status of the module and how it is communicating to the Controller.

Table 2-9. Link Controller Module Diagnostic LEDs

LED	Description
P (Green)	Power OK LED. Lit when the +5V power is OK
C (Green)	Communications OK LED. Lit when the Controller is communicating with the Serial Link Controller module.
E (Red)	External Fault LED. Lit when the Electronic module's 80C186EC CPU sets bit 1 of the internal Electronics module Board Control register.
I (Red)	Internal Fault LED. Lit whenever the Force Error bit (Bit 1) of the Configuration Register is active or when a timeout of the I/O bus watchdog timer occurs when the Controller stops communicating with the module. The Electronic module's 80C186EC processor may also set bit 0 of the internal Electronic module's Board Control register that forces the Internal Fault LED to illuminate. Also lit following module power-up until the Electronic module's 80C186EC processor resets internal Electronic module's Board Control register bit 0.
1 through 8 (Green)	Status LED. Lit when the Electronic module's 80C186EC processor sets the corresponding bit in the internal Electronic module's LED status.

Section 3. Link Controller Initialization

3-1. Section Overview

This section describes the initialization of the Ovation Link Controller card.

3-2. Software Needed

The following programs are provided on floppy disks for the initialization of the LC module through an external personal computer:

- RLCFLASH.EXE (provided on disk RLC10A)

  Backs up the Link Controller's RAM disk to flash (non-volatile) memory.

- RLCEXTPC.EXE (provided on disk RLC20A)

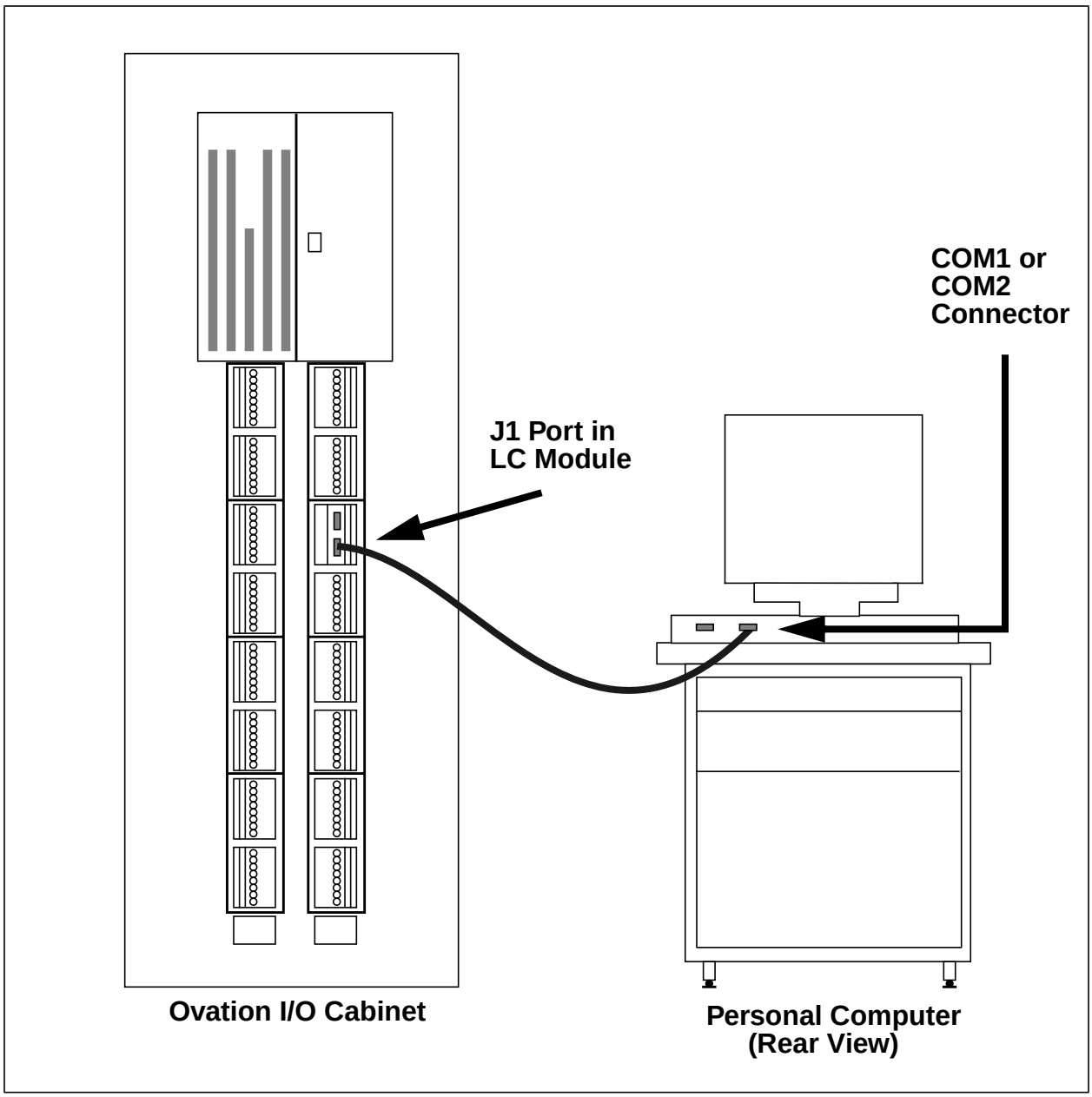
  The external Personal Computer host program formats the LC's RAM disk (if needed), loads DOS (if needed) and copies any desired files to the Link Controller RAM disk. Allows the LC to receive commands from the computer and to write information to the PC's CRT.

Note

It is assumed that the user is proficient in DOS.
(DOS 5.0 is the required version.)

3-3. Link Controller Initialization

To perform the initial programming (or any later action requiring keyboard/CRT I/O), a serial port (J1) on the LC is linked to the external personal computer's COM1 or COM2 port. In this configuration, code generated on the external personal computer can be loaded into the LC.



Once the LC is initialized, the external personal computer can be removed, and the LC operates as a stand-alone IBM-compatible microcomputer.

Note

After initialization, the RAM must be backed up to non-volatile memory.

Two programs are provided for use in LC initialization: RLCEXTPC.EXE and RLCFLASH.EXE (described in Section 3-2).

3-3.1. Procedure 1

LC Modules Containing DOS 5.0

Typically, LC modules have DOS 5.0 installed and tested at the factory, and are configured to boot from the local RAM disk before they are shipped to the field. These modules can be initialized by Procedure 1.

However, if the LC module does not have DOS 5.0 installed on it, or its RAM memory has become corrupted, use Procedure 2 described in Section 3-3.2 to initialize the LC module.

It is recommended that all LC initialization and file operations be performed from a floppy disk using the following procedure (which assumes that "C" is the PC hard drive and "A" is the PC floppy disk drive):

Note

While communicating with the LC, only one disk drive on the personal computer will be accessible. All desired files must be on that drive.

1. Copy the following files to **Drive C** on the PC:
   - — RLCFLASH.EXE program (on RLC10A disk)
   - — RLCEXTPC.EXE program (on RLC 20A disk)
   - — Any applicable LC executable programs
   - — Any applicable configuration files
2. Place the LC module (Personality and Electronics) in an appropriate Base Unit in an Ovation I/O cabinet (if necessary, refer to "Planning and Installing Your Ovation System" (U3-1000)).

3. Connect an applicable cable from the LC module (J1 Programming Port) to the personal computer (COM1 or COM2). (See Section 2 for additional information on the cable to be used.)

4. Copy the following programs and files from **Drive C** on the PC to the floppy disk in **Drive A**.

   — RLCFLASH.EXE program (on RLC10A disk)

   — RLCEXTPC.EXE program (on RLC20A disk)

   — Any applicable LC executable programs

   — Any applicable configuration files

5. Run RLCEXTPC.EXE from Drive A, using the command line syntax shown below:

   ```
   A:\>RLCEXTPC.EXE [port]  [baud]
   ```

   where:

   port = COM1 or COM2 (default = COM1)

   baud = 9600 or 19200 (default = 19200 with no jumper installed)

   Note

   If a baud rate of 9600 is desired, install a wire jumper in the Base Unit terminal block of the LC module between B7 and B8.

   For example, the following command line specifies that the LC is linked to COM1 and that the baud rate is 19200:

   ```
   A:\>RLCEXTPC.EXE COM1 19200
   ```

   If the port and baud rate are not specified, the default values will apply (port = COM1, baud = 19200).

6. Reset the LC module by removing it from its Base Unit, waiting five seconds, and then replacing it.

7. The LC module will perform a set of self-test diagnostics. Figure 3-1 illustrates the diagnostic LEDs on the LC Electronics module (refer to Table 2-9 for LED descriptions).

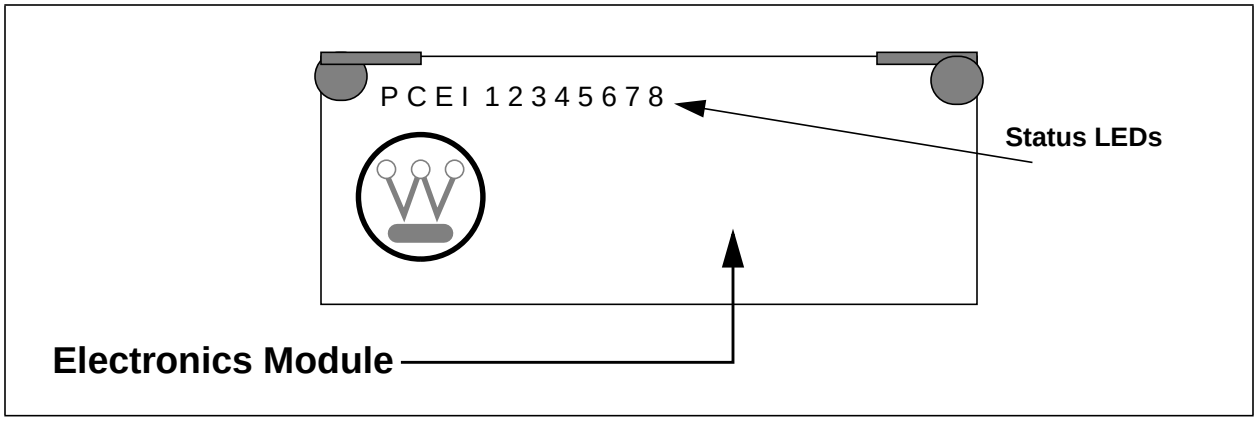


Figure 3-1. Link Controller Module LED Positions (Top View)

8. Observe the following indications:

   — Each of the set of eight LEDs (1 - 8) will be individually lit and then turned off in sequence starting with LED 1. When this test is completed, all LEDs (except P and C) should be off.

   — The 640 Kbytes of user RAM will be tested. The amount tested will be displayed on the external personal computer. If the test encounters an error, LED 1 will be lit and an error message will be written to the external personal computer.

   — If the LC boots successfully, the following will occur:

      — Only LED P (Power OK) will be lit.

      — The screen on the PC will display the **A:** \ > prompt.

      — The internal disk on the LC card is known as **Drive A**.

      — The external floppy is known as **Drive B**.

9. Copy any programs and files necessary for the operation of the LC application to **Drive A** from **Drive B**. These may include the following:

   — Any applicable LC executable programs.

   — RLCFLASH.EXE program (if desired, and if there is sufficient space on the LC's internal Drive A).

   — Any applicable configuration files.

   — Custom AUTOEXEC.BAT file (typically required to auto-start an LC application upon reset or power up of the LC). Do not place this file on the LC card until the application has been tested.

10. Enter the following command to save the current configuration of the LC's internal disk:

    ```
    B:\>RLCFLASH
    ```

    OR

    ```
    B:\>A:\RLCFLASH
    ```

    Caution

    It is extremely important to run the RLCFLASH.EXE program at this time. If this is not done, then the data on the LC internal disk will be lost.

11. To auto-start the application, reboot the LC by removing it from the Base Unit and then replacing it, or by pressing **Control-Shift-Delete**.

    After initial configuration of the LC card, it is still recommended that the LC card be operated from a floppy disk. This will avoid any potential problems with the hard disk drive.

12. The LC operates as an IBM-compatible personal computer. Executable files which are copied to (or created on) the RAM disk can be executed by entering the program name.

13. To exit the RLCEXTPC.EXE program at the external personal computer, press **Control-Break**.

For general information on the recommended LC programming approach, as well as application notes which may be helpful in programming the LC, refer to Section 4. For additional information on IBM-PC programming, refer to the applicable IBM and DOS documentation.

3-3.2. Procedure 2

LC Module Not Containing DOS 5.0, or LC Module with a Corrupted RAM Disk

It is recommended that all LC initialization and file operations be performed from a floppy disk using the following procedure (which assumes that "C" is the PC hard drive and "A" is the PC floppy disk drive):

Note

While communicating with the LC, only one disk drive on the personal computer will be accessible. All desired files must be on that drive.

1. Copy the following files to **Drive C** on the PC:

   — RLCFLASH.EXE program (on RLC10A disk)

   — RLCEXTPC.EXE program (on RLC 20A disk)

   — Any applicable LC executable programs

   — Any applicable configuration files

2. Place the LC module (Personality and Electronics) in an appropriate Base Unit in an Ovation I/O cabinet (if necessary, refer to "Planning and Installing Your Ovation System" (U3-1000)).

3. Connect an applicable cable from the LC module (J1 Programming Port) to the personal computer (COM1 or COM2). (See Section 2 for additional information on the cable to be used.)

4. Format a floppy disk as a DOS (5.0) bootable floppy by doing the following:

   Place the disk into a floppy drive at the PC and type the following command (assuming the disk is in Drive A):

   ```
   C:\>FORMAT   A: /S
   ```

5. Copy the following programs and files from **Drive C** on the PC to the floppy disk in **Drive A**.

   — FORMAT.COM program (DOS program)

   — RLCFLASH.EXE program (on RLC10A disk)

   — RLCEXTPC.EXE program (on RLC20A disk)

   — Any applicable LC executable programs

   — Custom AUTOEXEC.BAT file (required for automatic start-up of the LC application)

   — Any applicable configuration files

6. Install the following jumper in the Base Unit terminal block of the LC module in order to communicate with the external PC and to boot from the external PC disk:

   - Wire jumper between terminal block positions C7 and C8

7. The baud rate for the Programming Port defaults to 19200 (no jumper installed).

   If a baud rate of 9600 is desired, install the following jumper in the Base Unit terminal block of the LC module:

   - Wire jumper between terminal block positions B7 and B8

8. Set the floppy drive as the default disk by typing the following command (assuming the disk is in Drive A):

   ```
   C:\>A:
   ```

9. Run RLCEXTPC.EXE using the command line syntax shown below:

   ```
   RLCEXTPC.EXE [port]  [baud]
   ```

   where:

   port = COM1 or COM2 (default = COM1)

   baud = 9600 or 19200 (default = 19200 with no jumper installed)

   Note

   If a baud rate of 9600 is desired, install a wire jumper in the Base Unit terminal block of the LC module between B7 and B8.

   For example, the following command line specifies that the LC is linked to COM1 and that the baud rate is 19200:

   ```
   A:\>RLCEXTPC.EXE COM1 19200
   ```

   If the port and baud rate are not specified, the default values will apply (port = COM1, baud = 19200).

10. Reset the LC module by removing it from its Base Unit, waiting five seconds, and then replacing it. This will cause the LC to initialize itself and then to load DOS from the floppy disk.

    After DOS is loaded, the following will occur:

    - The screen on the PC will display the **`A:\>`** prompt.
    - DOS is executing on the LC card.
    - The external floppy is known as **Drive A**.
    - The internal disk on the LC card is known as **Drive B**.

11. When power is applied, the LC board will perform a set of self-test diagnostics. Figure 3-2 illustrates the diagnostic LEDs on the LC Electronics module. Refer to Table 2-9 for LED descriptions.

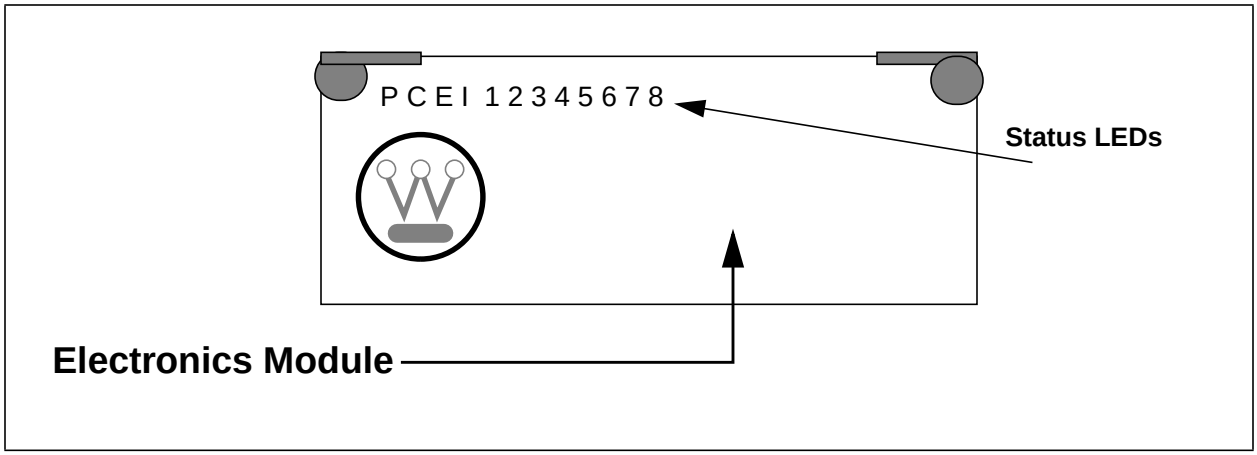


Figure 3-2. Link Controller Module LED Positions (Top View)

12. Observe the following LED indications:

    — Each of the set of eight LEDs (1 - 8) will be individually lit and then turned off in sequence starting with LED 1. When this test is completed, all LEDs (except P and C) should be off.

    — The 640 Kbytes of user RAM will be tested. The amount tested will be displayed on the external personal computer. If the test encounters an error, LED 1 will be lit and an error message will be written to the external personal computer.

    — The LC will now enter its bootstrap routine. If the LC is configured to boot from the external disk, and it cannot communicate with the external personal computer, LED2 will be lit. If this occurs, check the cabling.

    If no errors occur, when the LC has completed its start-up routine, only LEDs P (Power OK) and C (Communication OK) will be lit.

13. After the LC card has been booted from the external PC, format **Drive B** as a system disk by entering the following command:

    ```
    A:\>FORMAT B: /S
    ```

14. Copy any programs and files necessary for the operation of the LC application to the LC disk (**Drive B**). These may include the following:

    — Any applicable LC executable programs.

    — RLCFLASH.EXE program (if desired, and if there is sufficient space on the LC's internal disk, Drive B).

    — Any applicable configuration files.

    — AUTOEXEC.BAT file (typically required in a LC application). Do not place this file on the LC card until the application has been tested.

15. Enter the following command to save the current configuration of the LC's internal disk:

    ```
    B:\>RLCFLASH
    ```

    OR

    ```
    B:\>A:\RLCFLASH
    ```

    Caution

    It is extremely important to run the RLCFLASH.EXE program at this time. If this is not done, then the data on the LC internal disk will be lost.

16. At this time, remove the wire jumper between C7 and C8 on the LC Base Unit terminal block. This tells the LC to boot from the internal disk.

    Reboot the LC by removing it from the Base Unit and then replacing it.

    After initial configuration of the LC card, it is still recommended that the LC card be operated from a floppy disk. This will avoid any potential problems with the hard disk drive.

17. Once DOS is loaded to the RAM disk, the LC operates as an IBM-compatible personal computer. Executable files which are copied to (or created on) the RAM disk can be executed by entering the program name.

18. To exit the RLCEXTPC.EXE program at the external personal computer, press **Control-Break**.

For general information on the recommended LC programming approach, as well as application notes which may be helpful in programming the LC, refer to Section 4. For additional information on IBM-PC programming, refer to the applicable IBM and DOS documentation.

Section 4. Link Controller Programming and Operation

4-1. Section Overview

Due to the LC module's open design and the variety of possible applications, programming of the LC is highly application dependent. However, in order to design an effective LC application, the user must be aware of how the LC communicates through the Ovation I/O Bus (OIOB).

This section provides the following information:

- Programming approaches.
- General application notes.

Appendix A provides examples of LC application programs written in C language, as well as sample Controller application entries.

4-2. LC Programming Approach

In a typical LC application, information is sent from a field device to the LC's J2 Application serial port to be transferred to the Controller. Data or commands from the Controller may also be obtained by the LC and transmitted to the field device.

To implement an LC application, the following general steps are required:

- Define the format of the data to be received (or transmitted) through the J2 serial port.
- Create an LC application program to convert the data to an appropriate format and place it in OIOB-accessible dual port RAM (or read data placed in dual port RAM by the OIOB).
- Create the Controller application program to place the OIOB information into Ovation process points (or write point data to the OIOB).

Figure 4-1 illustrates the information flow between a field device and a Controller, using the LC interface. In this example, data is transmitted to the LC from a field device in RS-232 format. This data is made available to the Controller through the OIOB.

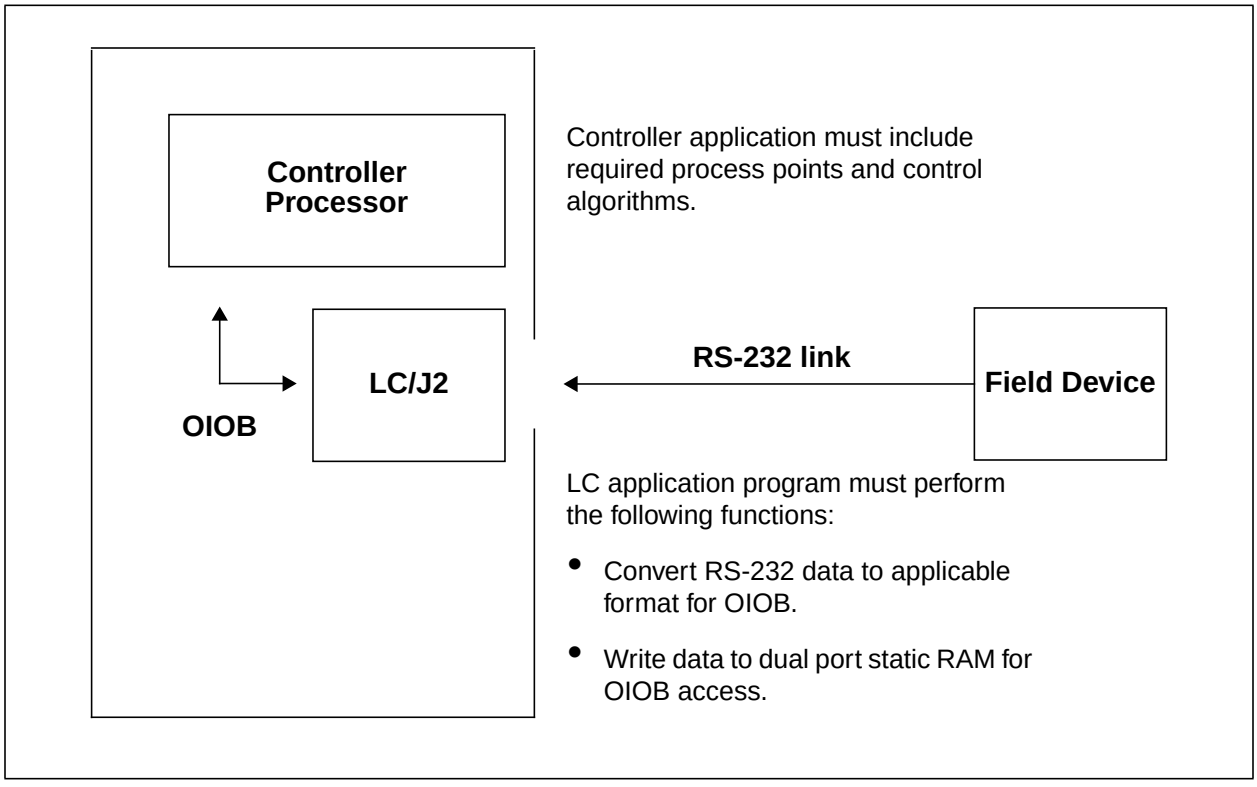


Figure 4-1. Example LC Application

4-2.1. Application Programming for the LC Module

When using the LC, the entire array of dual port LC RAM is available to the OIOB. Special Controller algorithms reference this memory as holding registers or buffers.

When planning and implementing an LC application, the following reference documents should be available:

- For detailed descriptions of the Ovation SLC algorithms, see "Ovation Algorithms Reference Manual" (R3-1100).
- For information on Ovation point record types and fields, see "Ovation Record Types Reference Manual" (R3-1140).
- For general information on Controller application programming, including procedures to add text algorithms to an application, see "Ovation Control Builder User's Guide" (U3-1040).

Figure 4-2 illustrates the interface between the OIOB and the LC dual port static RAM.

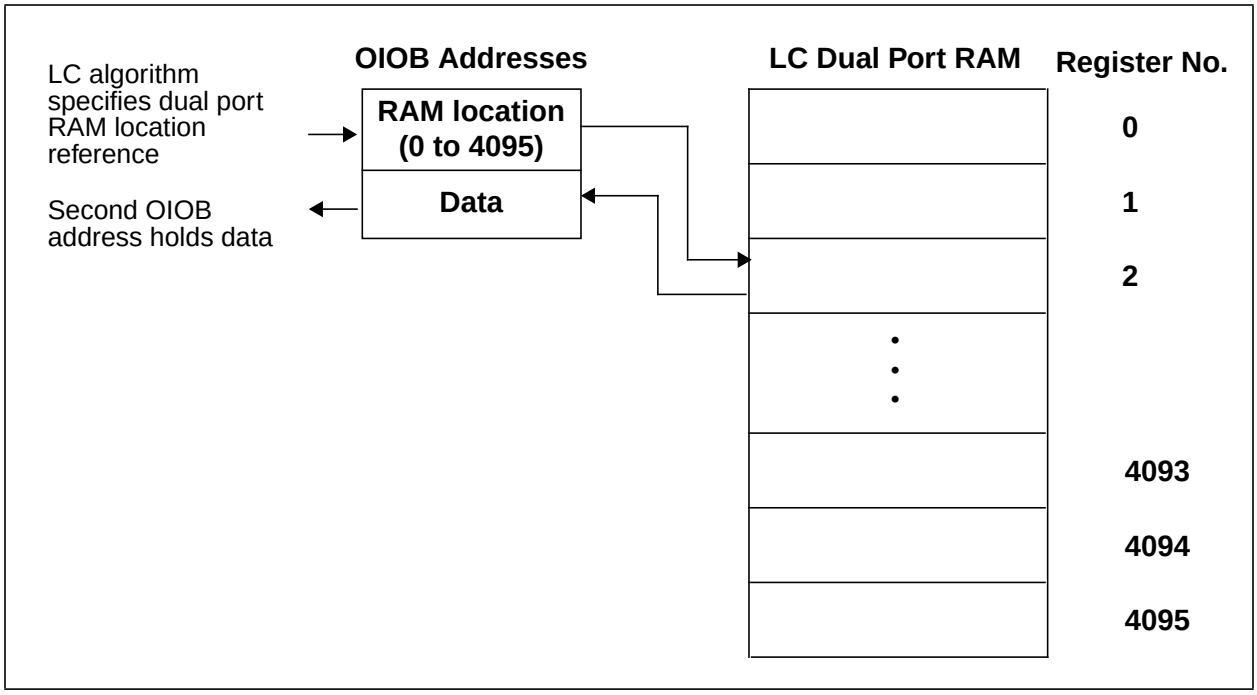


Figure 4-2. LC Ovation IO Bus Interface

The SLC algorithms reference locations in LC memory in one of two ways:

- As a series of 16-bit registers.
- As a series of 256-word circular buffers (for ASCII data only).

Figure 4-3 illustrates the correlation between the register numbers and memory addresses. Figure 4-4 illustrates the correlation between the buffers and memory addresses.

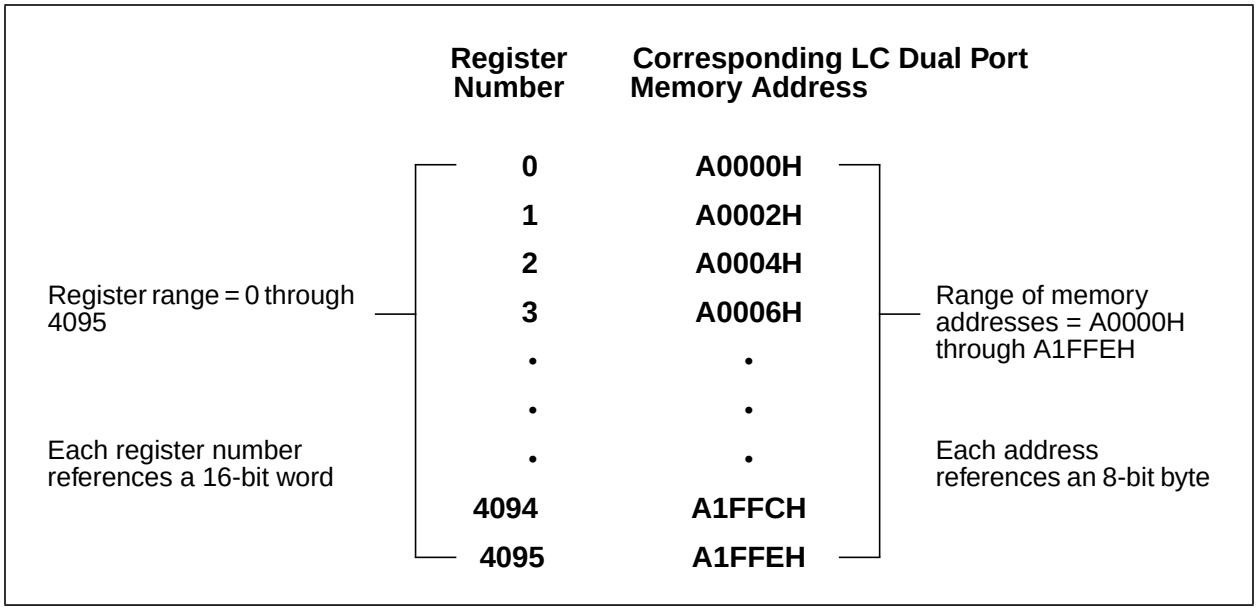


Figure 4-3. Registers Numbers and Memory Addresses (LC Algorithms)

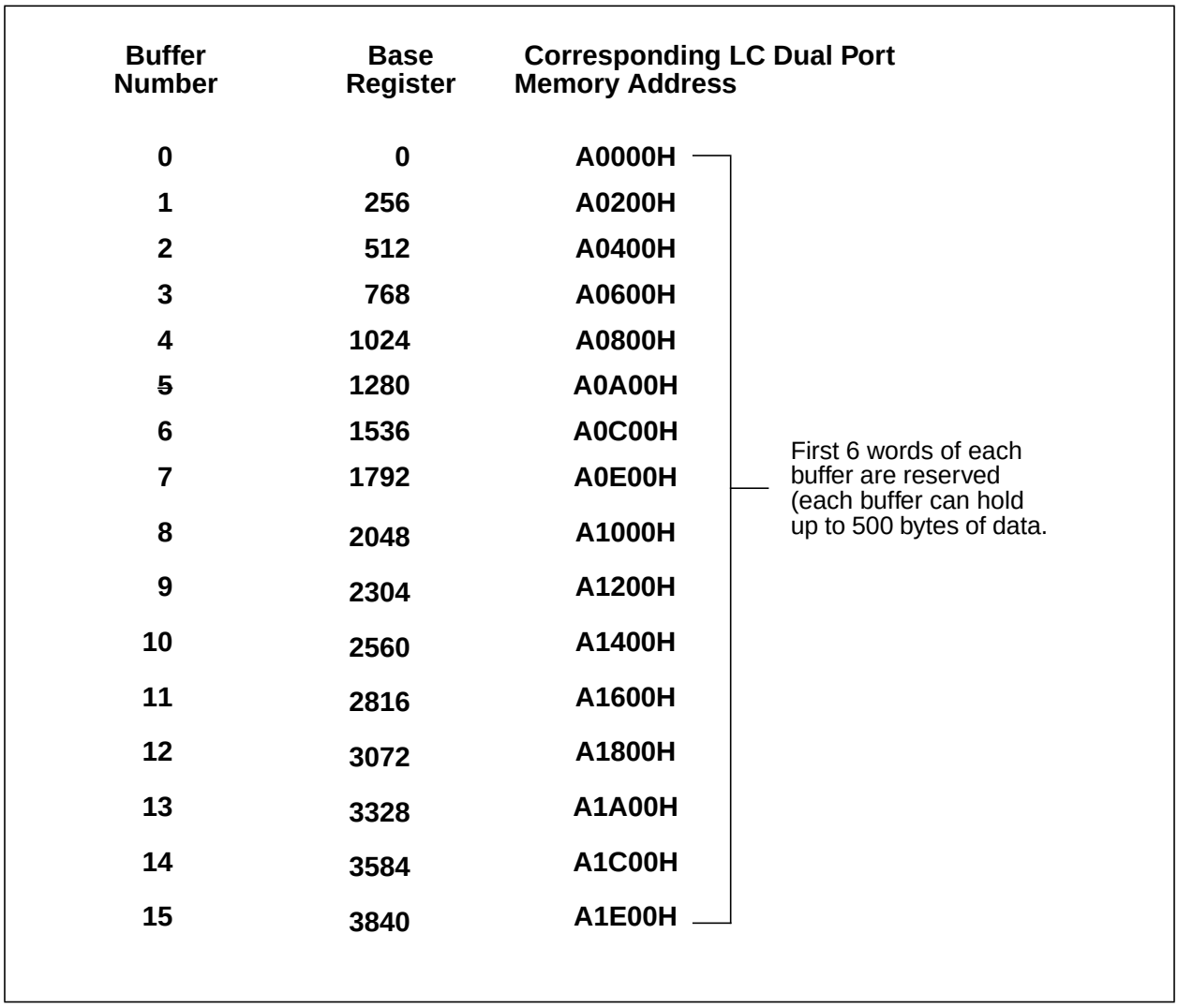


Figure 4-4. Buffer Numbers and Memory Addresses (LC Algorithms)

Provided Algorithms

The following Ovation text algorithms are provided for LC applications:

- **SLCAIN** reads up to 16 analog point values from an LC (or redundant pair of LCs). The values are read from consecutive registers in the LC static RAM, starting at a user-specified register. The data format may be specified as integer, real (Intel floating point format), or real with status word (1W record field).
- **SLCAOUT** writes up to 16 analog point values to an LC (or redundant pair of LCs). The values are written to consecutive registers in the LC static RAM, starting at a user-specified register. The data format may be specified as integer, real (Intel floating point format), or real with status word (1W record field).
- **SLCDIN** reads up to 16 digital point values from an LC (or redundant pair of LCs). The values are read from consecutive registers in the LC static RAM, starting at a user-specified register. Each digital point occupies one data register (equivalent to the 1W record field).
- **SLCDOUT** writes up to 16 digital point values to an LC (or redundant pair of LCs). The values are written to consecutive registers in the LC static RAM, starting at a user-specified register. Each digital point occupies one data register (equivalent to the 1W record field).
- **SLCSTATUS** reads hardware and user application status values from an LC (or redundant pair of LCs). The hardware status information returned includes the watchdog timer status and configuration switch settings. The application status values can be used to specify a drop fault code and parameters, as well as auxiliary information required by the specific application. The status information is placed in packed points.

The LC application program must be designed to convert data to and from the formats used by the SLC algorithms, as necessary. In addition, the program must read and write data to the appropriate locations within the LC static RAM (corresponding to the registers/buffers specified by the SLC algorithms).

4-3. Application Notes

The following application notes may be helpful in planning the LC application and in programming the LC module.

1. Typically, the programs to be run in the LC will be specified in the AUTOEXEC.BAT file. As in any other IBM-compatible microcomputer, AUTOEXEC.BAT will be executed automatically when the LC is reset or powered-up.

2. When defining an AUTOEXEC.BAT file (to run an application in an LC without a connected external personal computer), it may be useful to include the command BREAK=ON. This will allow the user to connect an external personal computer, start the RLCEXTPC.EXE program, and use **Control-C** to interrupt the program running in the LC.

3. Set the environment variable **NO87** by using the command:

   ```
   Set NO87=NO_87
   ```

   Also include this command in the module's AUTOEXEC.BAT file.

   This variable informs the application program that no floating point chip is installed.

4. When the LC is linked to the external personal computer, an LC reboot can be performed by pressing **Ctrl-Shift-Delete** (at the personal computer keyboard). If the left-hand **Shift** key is used, a warm reboot will be performed; a cold reboot will be performed if the right-hand **Shift** key is used.

5. DOS DEBUG can be used (as on an IBM-PC); however, it will not correctly display 80C186EC extended instructions. Microsoft CODEVIEW may be used in sequential mode only and must be invoked with the /D option. For additional information, refer to the applicable manufacturer's documentation.

6. Video display capability is only supported through the BIOS. Do not use any software that reads or writes directly to the video memory.

7. Due to differences between the LC and IBM-PC hardware, do not use software that attempts to program timers, the interrupt controller, or any other processor peripherals.

8. Figure 4-5 provides a memory map for the 80C186EC processor. Note the following points:

   — The 640 Kbytes DRAM (addresses 00000H - 9FFFFH) are available for use by the DOS operating system and DOS application programs.

   — For purposes of communication with the Controller, store data in the dual port RAM.

   — The remaining memory addresses are reserved for Westinghouse use.

9. Specific I/O addresses are available to the user application to write to the eight status LEDs (1 - 8), and read or write to the J2 Applications serial port. Table 4-1 describes the I/O addresses which may be accessed by the user application. All other I/O addresses are reserved for Westinghouse use.

10. The LC incorporates a watchdog timer which can be used by the application, if desired. The timer will time-out after 160 msec, unless the application program writes any byte value to I/O address 0900H.

11. When using the LC module, the watchdog status is available to the Controller through the SLCSTATUS algorithm. If the watchdog timer feature is not implemented in the user's application, the applicable status bit will always equal 0.

    If an LC watchdog time-out is detected, the LC can be reset by removing the LC module from its Base Unit, waiting five seconds, and then replacing the module.

12. After the LC is reset, the Ovation IO Bus registers (dual port RAM) are uninitialized. The application program in the LC should initialize these memory locations, as necessary.

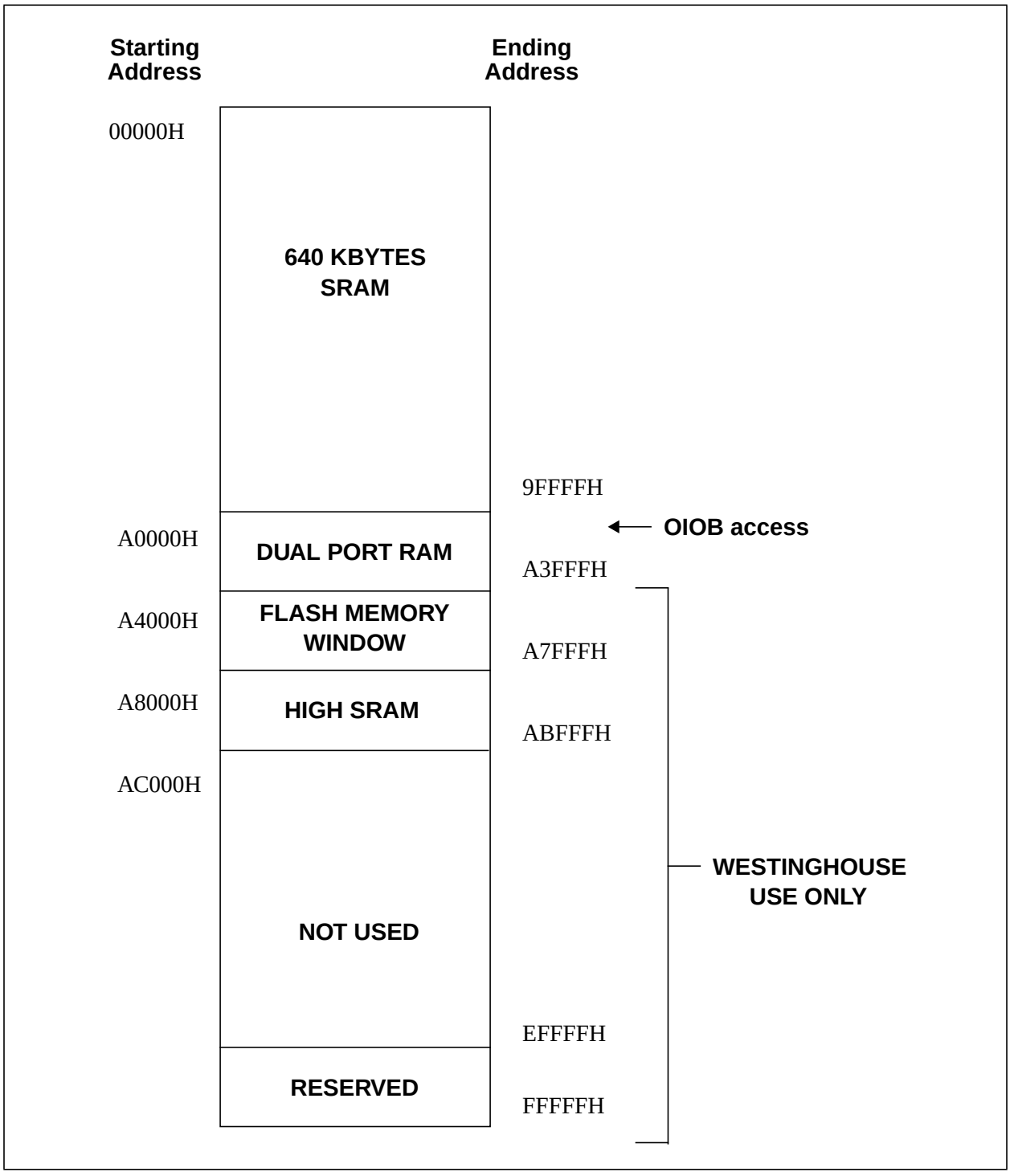


Figure 4-5. LC Memory Map

Table 4-1. User-Accessible I/O

Description	I/O Address	Width	Bit Definitions
LED Status Register (Write only)	0980H	8 bits	Bit 0 = LED 1 (1 = lit, 0 = not lit) Bit 1 = LED 2 (1 = lit, 0 = not lit) **Note** LED 1 and LED 2 are used by the BIOS during LC start-up, but may also be defined by the user application. Bit 2 = LED 3 (1 = lit, 0 = not lit) Bit 3 = LED 4 (1 = lit, 0 = not lit) Bit 4 = LED 5 (1 = lit, 0 = not lit) Bit 5 = LED 6 (1 = lit, 0 = not lit) Bit 6 = LED 7 (1 = lit, 0 = not lit) Bit 7 = LED 8 (1 = lit, 0 = not lit)
Board Option Register (Read only)	0A00H	8 bits	Bit 0 = (BOOT_SELECT) 1 = Boot up the LC from internal Flash memory. 0 = Boot up the LC via the Programming port if the Programming port is connected to the serial port of an external PC that is executing the RLCEXTPC.EXE code. Bit 1 = 1 (No function) Bit 2 = (LOC_BAUD_SEL) 1 = Use a 19200 baud rate for the Pmod's J1 Programming port. 0 = Use a 9600 baud rate for the Pmod's J1 Programming port. Bit 3 = (Mod_IDA/) Bit 4 = (Mod_IDB/) **Bit 4 3 Personality Module Style** 0 0 RS-232 (1C31169G01) 0 1 RS-485/RS-422 Four wire (1C31169G02) 1 0 Not defined 1 1 Not defined

Table 4-1. User-Accessible I/O (Cont'd)

Description	I/O Address	Width	Bit Definitions
Board Option Register (Read only) (Cont'd)	0A00H	8 bits	Bit 5 = 0 (No function) Bit 6 = 0 (No function) Bit 7 = 0 (No function)
Board Control Register (Write only)	0A80H	8 bits	Bit 0 = (CPU_INT-ERR) 0 = Internal error has not been detected by application program. 1 = Internal error has been detected by application program. Bit 1 = (CPU_EXT-ERR) 0 = External error has not been detected by application program. 1 = External error has been detected by application program. Bit 2 = Not used Bit 3 = Not used **Note** Bit 4 (PWRUPRDY/) is cleared upon module power-up and should remain in that state during normal module operation. Bit 4= (PWRUPRDY/) 0 = Provide the 80C186 processor with a READY signal every bus cycle. 1 = Do not provide the 80C186 processor with a READY signal every bus cycle. Bit 5 = Not used Bit 6 = Not used Bit 7 = (VPP-CTL) 0 = Turn off flash memory VPP programming voltage 1 = Turn on flash memory VPP programming voltage

Table 4-1. User-Accessible I/O (Cont'd)

Description	I/O Address	Width	Bit Definitions
COM1 Serial Port	03F8H - 03FFH	8 bits	Compatible with IBM-PC COM1 adapter
Watchdog Timer Reset (Write only)	0900H	8 bits	Data does not matter (see Application Note 10).

Notes

1. When using the LC module, status values equivalent to the board status register bits (with the exception of bit 5) are available to the Controller through the SLCSTATUS algorithm. For details, refer to "Ovation Algorithm Reference Manual" (R3-1100).

Appendix A. Link Controller Programming Examples

A-1. Section Overview

This appendix contains example application programs for the LC. These examples (written in C language) illustrate the following:

- Functions used to transfer values to and from the Ovation IO Bus (OIOB) registers.
- Test of LC functions (access OIOB registers and LC status register, and set LC LEDs) from the external personal computer.

A-2. Example Functions

The following listing contains functions used to obtain and store OIOB data, to read the LC status register, to write to the LC LEDs, and to write ASCII data to the J2 Applications serial port.

```
# include “lc.h”
# include <bios.h>

/* get integer value (format 0) from OIOB register */
int get_fm0(int reg)
  {
  int far *base = (int far *) 0xa0000000;
  return (*(reg + base));
  }

/* store integer value (format 0)) in OIOB register */
void str_fm0(int reg, int value)
  {
  int far *base = (int far *) 0xa0000000;
  *(reg + base) = value;
  }

/* get real value (format 1) from OIOB register */
float get_fm1(int reg)
  {
  int far *base = (int far *) 0xa0000000;
  return (* (float *)(reg + base));
  }

/* store real value (format 1) in OIOB register */
void str_fm1(int reg, float value)
  {
  int far *base = (int far *) 0xa0000000;
*((float*)(reg + base)) = value;
  }

/* get real with status (format 2) from OIOB register */
format2 get_fm2(int reg)
  {
  int far *base = (int far *) 0xa0000000;
  return ((format2) *(reg + base));
  }
```

```
/* store real with status (format 2) in OIOB register */
void str_fm2(int reg, format2 value)
  {
  int far *base = (int far *) 0xa0000000;
  ((format2) *(reg + base)) = value;
  }

/* set LED state */
void lc_led(char status)
  {
  outp(0x980,status);
  }

/* read LC status */
int lc_stat()
  {
  return(inp(0x0a00));
  }

/* write string to dumb terminal */
void string_out ( char *str )
{
  int i;

  for (i =0; i <strlen(str); i++)
    _bios_serialcom(_COM_SEND,0,str[i]);
}
```

A-3. Function Declarations (LC.H)

The following header file contains data structure and function declarations for the example functions in Section A-2.

```
typedef struct

  {
  int status;
  float ireal;
  } format2; /* real with status */

int get_fm0(int reg);                  /* get integer value          */
void str_fm0(int reg, int value);      /* store integer value        */

float get_fm1(int reg);                /* get real value             */
void str_fm1(int reg, float value);    /* store real value           */

format2 get_fm2(int reg);              /* get real w/ status         */
void str_fm2(int reg, format2 value);  /* store real w/ status       */

int lc_stat();                         /* read LC status */

int lc_led(char status);               /* set LEDs */

void string_out ( char *str );         /* write to dumb terminal     */
```

A-4. Example LC Test

This example program can be used to test the following LC functions:

- Store a value (integer, real, or real with status) from a specified OIOB access register.
- Display a value (integer, real, or real with status) from a specified OIOB access register.
- Display the LC status register value.
- Set the LC status LEDs (to a specified hexadecimal value).

To use this program, the external personal computer must be connected and RLCEXTPC.EXE must be running, as described in Section 3.

```
#include<dos.h>
#include<stdio.h>
#include<lc.h>
#include<conio.h>
main()
{
int data0,count,cmd,reg,status;

float data1
format2 data2;
char *cls[3];
*cls = "cls";
cmd = 10;
data1 = 0;
while(cmd != 0)
{
```

```
printf("LC TEST FUNCTIONS \n");
printf("1: Store Hex integer in Controller Buffer Area \n");
printf("2: Retrieve Hex integer from Controller Buffer Area \n");
printf("3: Store Real in Controller Buffer Area \n");
printf"4: Retrieve Real from Controller Buffer Area \n")
printf("5: Store Real with status in Controller Buffer Area \n");
printf("6: Retrieve Real with status from Controller Buffer Area \n");
printf("7: Read LC Status Register \n");
printf("8: Write to LC LED Register \n");
printf("0: EXIT \n");
printf("ENTER NUMBER OF FUNCTION TO TEST:     ");
scanf("%d" ,&cmd);
printf("\n\n");
switch(cmd){
  case 1:
           {
           printf("ENTER REGISTER NUMBER AND DATA reg=data(0xhex)  ");
           scanf("%d=%x" ,&reg,&data0);
           printf("\n");
           str_fm0(reg,data0);
           data0 = 0x0;
           break;
           }
case 2:{
           printf("ENTER REGISTER NUMBER TO RETRIEVE DATA          ");
           scanf("%d" ,&reg);
           data0 = get_fm0(reg);
           printf("\n HEX in reg# %d is %04xH \n\n" ,reg,data0);
           break;
           }
```

```
case 3:{
           printf("ENTER REGISTER NUMBER TO STORE REAL    reg=data  ");
           scanf("%d=%f" ,&reg,&data1);
           printf("\n");
           str_fm1(reg,data1);
           break;
           }
case 4:{
           printf("ENTER REGISTER NUMBER TO RETRIEVE REAL          ");
           scanf("%d" ,&reg);
           data1 = get_fm1(reg);                                    ‘
           printf("\n Real in reg# &d is %f \n\n" ,reg,data1);
           break;
           }
case 5:{
           printf("ENTER REGISTER NUMBER STATUS & REAL VALUE
           reg=0xhex:value                                          ");
           scanf("%d=%x:%f" ,&reg,&data2.status,&data2.ireal);
           printf("\n");
           str_fm2(reg,data2);
           break;
           }
case 6:{
           printf("ENTER REGISTER TO RETRIEVE REAL PLUS STATUS     ");
           scanf("%d" ,&reg);
           data2 = get_fm2(reg);
           printf("\nStatus and Real in reg# ");
           printf("\%d are %04x:%f \n\n" ,reg,data2.status,data2.ireal);
           break;
           }
```

```
case 7:
           {
           status = lc_stat();
           printf("\nLC Status register value is %02x \n\n" ,status);
           break;
           }
case 8:
           {
           printf("\nENTER STATUS TO DISPLAY ON LEDs    (hex):     ");
           scanf("%x" ,&status);
           lc_led(status);
           printf("\n");
           break;
           }
           }
           printf("\n\n\n Strike any key to continue");
           while(kbhit() == 0);
           system(*cls);
}
}
```

M

O

P

R

S

T

V

W