DataPort Programmer's Guide

PLUM A+™

PLUM A+3™

For use with Lists 11973-04, 12393-04, & 12348-04

Back Prime	A	B	Options/ Vol Inf

Hospira

Hospira, Inc.
Lake Forest, IL 60045, USA

430-95008-002 (Rev 6/04)

Change History

Title	Description of Change	Pages Affected
430-95008-001 (Rev. 8/00)	Original Issue	All
430-95008-A01 (Rev. 7/02)	Updated Cover Updated Copyright & History Updated Programmer Qualification Updated Back Cover	Cover Front Matter 1-1 Back
430-95008-002 (Rev. 6/04)	Second Release	All

This page intentionally left blank.

CONTENTS

Section 1

Section 2

Section 3

Section 4

Section 5

Section 6

Appendix A

Appendix B

Appendix C

Appendix D

Appendix E

Appendix F

FIGURES

TABLES

Section 1
INTRODUCTION

1.1
PROGRAMMER QUALIFICATION

The Plum A+ Infusion System With DataPort Programmer's Guide is intended for programmers in the healthcare profession and other qualified professionals who wish to develop software for remote monitoring of the Plum A+ Infusion System With DataPort, herein referred to as the pump. The manual describes the procedures for monitoring up to 15 pumps using a host computer with a serial port. This manual can also be used for the Plum A+3 (List 12348-04), with the exception that only four (4) Plum A+3 units are monitored from the same host computer.

This manual is written with the assumption the programmer is familiar with serial communications principles, computer data handling, and infusion pumps.

Complete instructions for connecting the pump to a host computer and monitoring the device from a remote location are provided. However, this manual is not a technical service manual nor an operating manual for the pump.

1.2
DATAPORT SYSTEM OVERVIEW

DataPort is the communications system used by the pump. This communications system can be used to monitor the pump status.

The DataPort system consists of commands issued by the host computer and responses returned from the pump. All commands must have responses.

The pump does not spontaneously communicate with the DataPort, but must be specifically interrogated.

The DataPort system is used to remotely monitor pumps from a host computer. It cannot be used to remotely control pumps other than setting their soft IDs. The pump is controlled locally.

1.3 HOW TO USE THIS MANUAL

Users should read this manual thoroughly prior to testing or connecting any pump to the host computer.

CAUTION: The pump must be initially started on AC power, otherwise the DataPort may not operate properly.

Pay special attention to Appendix A, *NOTATIONAL CONVENTIONS*. Appendix A explains Extended Backus Notation (EBN), the set of symbols used in this document to describe the syntax of packets. Note that a glossary also appears in Appendix B.

At this point, begin familiarization with remote monitoring of the pump. Keep this manual and the Plum A+ Infuser With DataPort Operating Manual available during program development and testing.

In the next three sections, the manual describes hardware and electrical specifications, low-level software, and high-level software. Programming guidelines and troubleshooting assistance are provided in *Section 5, PROGRAMMING GUIDELINES* and *Section 6, TROUBLESHOOTING*. The appendices provide an outline of notational conventions, definitions of terms, suggested readings, and a list of hardware parts for the DataPort system.

Section 2

HARDWARE DESCRIPTION

This section describes electrical specification standards and hardware requirements for configuring DataPort connections.

2.1 THE EIA-232-D ELECTRICAL SPECIFICATION STANDARD

The EIA-232-D is the electrical specification standard used by the pump. This standard was selected for its ease of use with personal computers, which are frequently equipped with a compatible RS-232 serial port.

The DataPort system conforms to the EIA-232-D standard, with the following exceptions:

- ❏ DataPort uses a DB-15 and 6-pin modular connectors in addition to the standard DB-25 and DB-9 connectors.
- ❏ With DataPort, more than one pump is allowed on a line.
- ❏ The DataPort minimum line impedance is 2K Ohms (EIA-232-D standard: 3K Ohms minimum).
- ❏ The DataPort maximum line impedance is 30K Ohms (EIA-232-D standard: 7K Ohms maximum).
- ❏ The maximum line capacitance is 13,000 pf (EIA-232-D standard: 2,500 pf maximum).

2.2 SYSTEM CONFIGURATION OPTIONS

The DataPort system may be configured in either of two ways as detailed in the following sections:

2.2.1 DIRECT CONFIGURATION

The host computer is connected to one pump via a Plum to PC cable having a DB-15 connector for the pump and a DB-9 or DB-25 connector for the host computer.

2.2.2 BUS CONFIGURATION

In a bus configuration, each pump has a Hospira infusion pump Junction-Box attached, and each Junction-box on the channel is configured to a different hard ID between 1 and 15. The host computer is attached to a pump Junction-box via the Junction-Box to PC cable (6-pin modular connector to DB-9 or DB-25 connector). The other Junction-box port may be connected to another junction-box via a Junction-Box to Junction-Box cable (6-pin modular connectors on each end). In this manner, up to 15 devices may be attached to each other, with one attached to the host. The order in which pumps are connected to the channel is of no consequence.

Note: To remove a pump from the channel, detach the device from the junction-box. Do not detach the junction-box from the cable; it disrupts the continuity of the bus.

2.3 CABLE REQUIREMENTS

The DataPort system uses cables supplied by Hospira to fit the configurations diagrammed in .

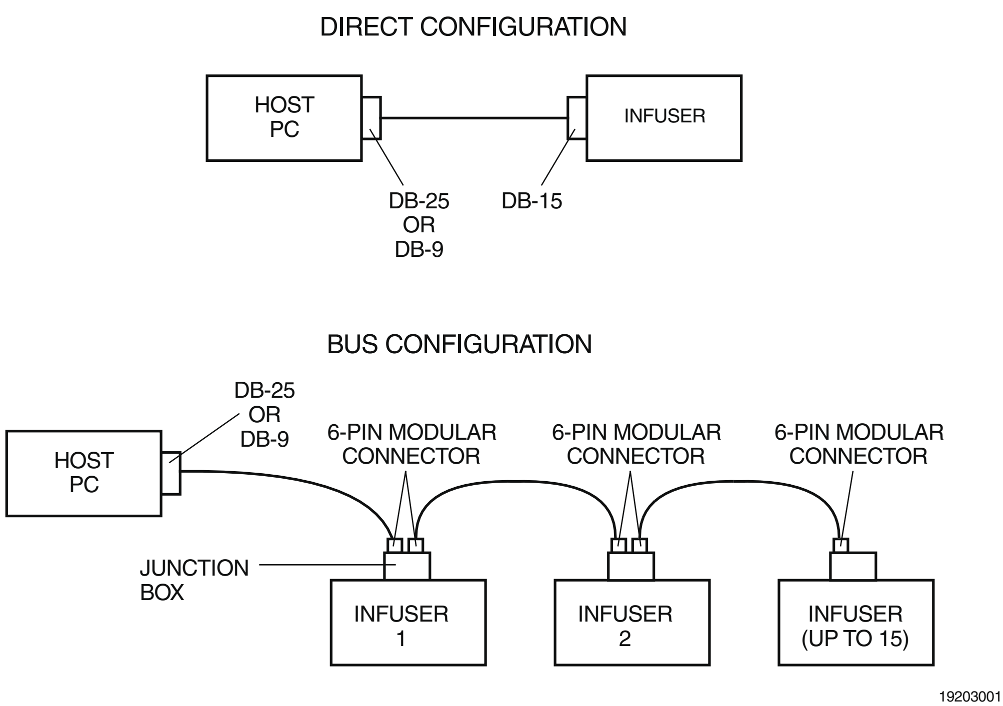


Figure 2-1. Cable Configurations for the DataPort System

See *Appendix E, DATAPORT ACCESSORIES LIST*, for descriptions and list numbers for available cables and junction-boxes.

2.4 6-PIN MODULAR CONNECTOR PIN ASSIGNMENTS

Two 6-pin modular connectors can be attached to each junction-box. *Figure 2-2, Pin Locations for 6-Pin Modular Connector*, illustrates connector pin use.

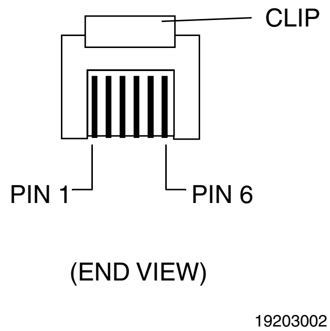


Figure 2-2. Pin Locations for 6-Pin Modular Connector

Table 2-1, Pin Assignments for the 6-Pin Modular Connector, lists pin assignments and signal descriptions.

Table 2-1. Pin Assignments for the 6-Pin Modular Connector

Pin Number	Name	Description
1	TX	Host Transmit
2	COMM	Common Ground
3	NC1	Not Connected
4	NC2	Not Connected
5	RX	Host Receive
6	CTS	Clear To Send, set high by a pump to signify at least one device on a channel.

2.5
DB-15 CONNECTOR PIN ASSIGNMENTS

Each pump is equipped with a female DB-15 port on the rear of the device for the RS-232 DataPort connection. Pin locations are illustrated in *Figure 2-3, Pin Locations for the DB-15 Connector*.

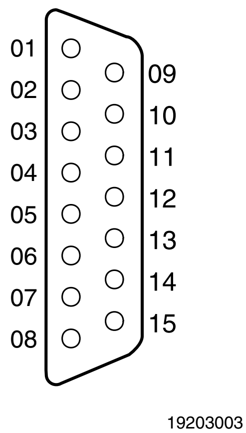


Figure 2-3. Pin Locations for the DB-15 Connector

Pin assignments and signal descriptions are shown in *Table 2-2, Pin Assignments for the DB-15 Connector*.

Table 2-2. Pin Assignments for the DB-15 Connector

PIN NUMBER	SIGNAL NAME	SIGNAL DESCRIPTION
1	TX	Host Transmit
2	RX	Host Receive
3	AUXOUT	Spare Output
4	NC	Not Connected
5	NC	Not Connected
6	COM_A1	Hard ID Bit 1
7	COM_A3	Hard ID Bit 3
8	DGND	Ground for Address Lines
9	COMM	Common Ground
10	CTS	Clear To Send
11	AUXIN	Spare Input
12	NC	Not Connected
13	COM_A0	Hard ID Bit 0

Table 2-2. Pin Assignments for the DB-15 Connector		
PIN NUMBER	**SIGNAL NAME**	**SIGNAL DESCRIPTION**
14	COM_A2	Hard ID Bit 2
15	COM_A4	Hard ID Bit 4 (used as parity bit on the address lines — even parity)
Note: Pins COM_A0, COM_A1, COM_A2, COM_A3, and COM_A4 are high-impedance lines connected to +5V and shorted to DGND to create the hard ID. AUXOUT and AUXIN are unused output- and input-lines, respectively.		

2.5.1 DB-15 CONNECTOR LOGIC GROUND AND PUMP COMMON

Pin 8 is Logic Ground for pins 6, 7, 13, 14, and 15, which determine the hard ID for each individual pump. Pins 6 to 8 and 13 to 15 have no application in the host computer.

2.5.2 SIGNAL LINES

The RX and TX lines are used to transmit serial data using EIA-232-D rise time and voltage levels. The CTS line is held high by the pump throughout the time the pump is on.

2.6 JUNCTION-BOX ADDRESS ASSIGNMENT

The junction-box supplies a hard ID number between 1 and 15 through five pins in the DB-15 connector (hard ID is set using the DIP switches in the junction-box). Four pins generate a binary number; the fifth pin is used to generate even parity.

When a bus configuration is used, each junction-box must be set to a hard ID unique on that channel, so that the attached pump can be uniquely identified by its hard ID. When direct configuration is used, the pump can be addressed by either a soft ID or hard ID of 0.

Note: A label is provided on the outside of the junction-box for use in writing its hard ID. Writing the hard ID on the junction-box label facilitates maintaining unique hard IDs on a channel.

The following signals are required on lines COM_A0 through COM_A4 to achieve hard IDs 0 through 15 (See *Table 2-3, Required Signals for Hard IDs 0 Through 15*. “H” is for high, “L” for low):

Table 2-3. Required Signals for Hard IDs 0 Through 15

HARD ID	COM_A4 (PARITY)	COM_A3	COM_A2	COM_A1	COM_A0
0*	H	H	H	H	H
1	L	H	H	H	L
2	L	H	H	L	H
3	H	H	H	L	L
4	L	H	L	H	H
5	H	H	L	H	L
6	H	H	L	L	H
7	L	H	L	L	L
8	L	L	H	H	H
9	H	L	H	H	L
10	H	L	H	L	H
11	L	L	H	L	L
12	H	L	L	H	H
13	L	L	L	H	L
14	L	L	L	L	H
15	H	L	L	L	L

* A hard ID of 0 is used in direct configuration only

2.6.1 JUNCTION-BOX DIP SWITCH SETTINGS

Table 2-4, DIP Switch Settings, lists switches and their corresponding address lines.

Table 2-4. DIP Switch Settings

LINE	SWITCH
COM_A4	5
COM_A3	4
COM_A2	3
COM_A1	2
COM_A0	1
Note: Switch position ON generates a low signal on the corresponding address line ("L" on the table of hard IDs).	

Hard ID numbers and corresponding DIP switch settings are illustrated in *Figure 2-4, Junction-Box Switch Positions*.

ID NO	SW POSITION 1	2	3	4	5
1	■	○	○	○	■
2	○	■	○	○	■
3	■	■	○	○	○
4	○	○	■	○	■
5	■	○	■	○	○
6	○	■	■	○	○
7	■	■	■	○	■
8	○	○	○	■	■
9	■	○	○	■	○
10	○	■	○	■	○
11	■	■	○	■	■
12	○	○	■	■	○
13	■	○	■	■	■
14	○	■	■	■	■
15	■	■	■	■	○

■ = ON ○ = OFF

0368101

Figure 2-4. Junction-Box Switch Positions

This page intentionally left blank.

Section 3

LOW-LEVEL SOFTWARE

Low-level communications software has two components: receiving and transmitting. On the receiving side, low-level software receives characters and assembles them into packets. Low-Level software then verifies that the packets have been received correctly.

On the transmitting side, the low-level software takes messages from high-level software, formats them as packets, and causes each character of the packet to be transmitted.

This section describes the communication format, timing, packet format, soft ID, verification of ID, low-level DataPort errors, and error recovery.

3.1 COMMUNICATION FORMAT

The baud rate for the Plum A+ is selectable and can be set to 1200, 2400, 4800, or 9600 baud. Parity is none, with 8 data bits and 1 stop bit. (Operational note: the user should make sure the selected baud rate is compatible for all devices connected to the DataPort System.)

The data format on the serial port is a 10-bit frame with asynchronous start and stop. Characters contain 8 bits (one byte) of information, 1 start-bit, and 1 stop-bit as illustrated in *Figure 3-1., Communication Format.*

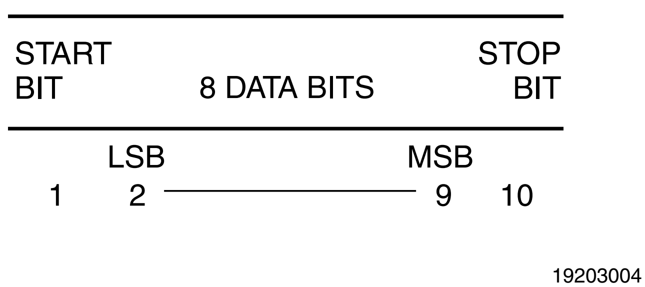


Figure 3-1. Communication Format

3.2 TIMING

The host computer transmits a packet, one character at a time, using any inter-character delay determined effective. After transmitting the last character of the packet, the host computer receives the pump response.

For optimal speed in receiving packets, the host computer should buffer all incoming characters until it receives a carriage return or the three-frame inter-character time limit is exceeded. When either of these events occurs, the host computer may begin processing the characters as a packet.

The host computer can expect to receive the first character of a pump's response within 40 ms of transmitting the last bit of the last character of a packet to the device. If the device does not respond within this time, the host computer should send the flush-character and retransmit the previous packet.

Upon receiving a complete packet, the host computer should check for character errors and Cyclic Redundancy Check (CRC) errors. The CRC and the carriage return may then be stripped from the packet.

3.3 PACKET FORMAT

Information is communicated by formatting a message, adding information to build a packet, and sending the packet. A packet consists of addressing information, a message, a four-character ASCII CRC, and a carriage return. A packet is subject to the following restrictions:

- ❏ The characters which may be included are printable ASCII and a carriage return.
- ❏ A packet may not contain a carriage return other than the one to terminate the packet.
- ❏ Maximum length of a packet that can be received by a pump is 28 characters; maximum length of a packet sent by a pump is 256 characters.

The format for each packet is as follows. (See *Appendix A, NOTATIONAL CONVENTIONS*, for an explanation of symbols.)

```
packet              ::= command packet|response packet
command-packet      ::= msg-to {soft-id|@hard-id}};
                        command-message crc cr
response-packet     ::= msg-from {hard-id|?}; soft-id;
                        response-message crc cr
msg-to              ::= T
msg-from            ::= F
hard-ID             ::= digit...(integer in range 0 - 15)
soft-id             ::= digit...(integer in range 0 - 99999999)
crc                 ::= HHHH (where "H" is a digit or uppercase
                        letter in the range A to F)
cr                  ::= ASCII carriage return
```

Note: *Section 4.5, COMMAND RESPONSE PAIRS*, provides command-message and response-message description.

3.3.1 CRC

Each packet contains a 2-byte CRC formatted as four ASCII characters.

The CRC is formed according to the CCITT-REV method, in which the polynomial is x**16 + x**11 + x**4 + 1 (** represents exponentiation). The polynomial is also characterized by its 16-bit generator value of 8408h. See *Section 5.3, PROGRAMMING EXAMPLES* for sample code, and *Appendix C, SUGGESTED READINGS*.

3.3.2 SAMPLE PACKETS

The following examples illustrate the structure of packets:

Example 1

Packet:

```
T500;IALR;DV1;DV2;622A<CR>
```

Meaning:

From the host to the pump with soft ID 500A (`T500`).
Inquire (`I`) about alarm conditions (`ALR`),
Primary delivery rate (`DV1`),
Secondary delivery rate (`DV2`),
The value of the CRC is `622A`.
<CR> is the terminating carriage return.

Example 2

Packet:

```
F11;500;ROK;125;200;875B<CR>
```

Meaning:

From pump with soft ID 500 and hard ID 11 (`F11;500`).
Respond normally (`R`), indicate no alarms (`OK`),
Indicate primary delivery rate (`125`) mL/hr,
Indicate secondary delivery rate (`200`) mL/hr,
The value of the CRC is `875B`.
<CR> is the terminating carriage return.

3.4 SOFT ID

The pump retains its soft ID in non-volital memory. The soft ID number is an integer in the range 0 through 99999999. Each pump must have a unique soft ID other than 0 when used in a bus configuration *(see Section 3.5.2, Pump Verification)*. The soft ID of a pump can be set through its DataPort.

In order to ensure that the Soft ID of each pump is unique, Soft ID's must be recorded and tracked to confirm that no duplicate ID's exist. One method to record and track Soft ID's is to set up a computer database which associates the Soft ID of each pump with its serial number. This database can be designed to prevent duplicate ID's from being entered.

Note: It is the responsibility of the hospital to enforce this requirement.

3.5 ID VERIFICATION

The following paragraphs detail packet identification by the host computer and the pump.

3.5.1 HOST VERIFICATION

The host computer is responsible for confirming that each response packet received is from the correct pump. Specifically, if the outgoing packet was addressed by hard ID, the hard ID on the returned packet must match. If the outgoing packet was addressed by soft ID, the soft ID on the returned packet must match. The host computer should perform at least one retry before reporting the error.

3.5.2 PUMP VERIFICATION

When the pump receives a packet, it verifies that the ID on the packet matches its own ID, whether the packet ID is hard or soft. If the ID matches, the pump must respond. If the ID does not match, the pump will not respond.

A hard ID or a soft ID of 0 is always considered a match by the pump. A pump will respond to a message addressed to either a hard or a soft ID of 0 even when the parity setting on the junction-box is incorrect.

Note: Note: Do not use a hard or soft ID of 0 when using a bus configuration. This will prevent every device on the bus from responding to the host computer simultaneously.

3.6 DATAPORT ERRORS

DataPort errors are communication errors, such as improper checksum.

The pump will not respond to an erroneous transmission, nor will it attempt to inform the host computer that DataPort errors have occurred.

Low-level software DataPort errors are listed in *Table 3-1, Low-Level Software Errors*.

Table 3-1. Low-Level Software Errors

ERROR	DESCRIPTION
Framing error	The received packet is missing a valid start or stop bit.
Overrun error	The data in the receive register of the UART was overwritten by an incoming character before the CPU was able to accept it.
Break interrupt	The received data is held in a logic “0" state (+5V to +15V) longer than a normal byte transmission time.
CRC error	When a packet is received, the CRC of the packet contents (exclusive of the CRC and the carriage return) is calculated and compared to the CRC transmitted in the packet. If there is a difference, a transmission error has occurred.

3.6.1
RECOVERY FROM DATAPORT ERRORS

If a host computer receives an incomplete or corrupted packet from a pump, the host transmits a flush-character (03 hexadecimal) and repeats its previous message. If the host continues to receive bad packets in response, the host computer should indicate a communication failure. A pump detecting a transmission error will not respond.

This page intentionally left blank.

Section 4
HIGH-LEVEL SOFTWARE

High-level software performs several major tasks: communicating with connected devices; finding and reporting new, disconnected, and moved devices; and detecting equipment failures.

4.1
MESSAGE PASSING

The host computer initiates all conversations; the pump does not transmit unless responding to a message from the host computer.

4.2
PUMP STATUS DETECTION

The host computer routinely transmits queries on each channel to detect the following:

NEW PUMP The host computer periodically queries each location on each channel where CTS is high to find pumps that were previously unconnected.

OFF OR DISCONNECTED PUMP If a pump fails to respond within 40 ms, it is considered to have been turned off or disconnected.

EQUIPMENT FAILURE The host computer should attempt to detect and report the conditions listed below. Detecting and reporting other equipment failures is not required by the communication specification, except the failure of a pump to respond.

- CTS Stuck Low The host computer may test the CTS lines by periodically scanning each location on each channel where CTS is low. Each location should be queried for the existence of a pump. If there is any response, the user should be notified that a CTS line is defective.
- CTS Stuck High The host computer should recognize when CTS is high on a channel and no pumps respond, and should report it to the user.

Note: When a pump is turned on, the CTS line will go high immediately. The pump will not respond to inquires until its self test is complete.

4.3
DEVICE ADDRESSING

A device may be addressed by its hard ID or soft ID.

4.4 MESSAGE STRUCTURE

The structure of messages is specified using Extended Backus Notation, described in *Appendix A, Notational Conventions*. Messages are structured as follows:

command-message	::=command-type [command ;]...
response-message	::=response-type [response ;]...
command-type	::=I | S (Respectively, Interrogate and Set)
response-type	::=E | A | R | e | a | r (Respectively, Error, Acknowledge, normal Response. For each response type, lowercase is used to signify that an alarm condition exists. Otherwise, uppercase is used.)
command	::=param-code sp field
sp	::=x (where "x" is an ASCII space)
param-code	::=letter [letter|digit]... (See *Section 4.6.1, PARAMETER CODES (PARAM-CODES)*)
field	::=boolean-value | number | qstring | name | hex-number | time | out-of-range
boolean-value	::=T | F
number	::=[-] integer [.integer]
qstring	::="[x...]" (where x is a printable ASCII character, and the string is interpreted as defined in the C programming language)
name	::=letter [letter|digit]...
hex-number	::=hex-digit...
hex-digit	::= 0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F
integer	::=digit...
digit	::=0|1|2|3|4|5|6|7|8|9
time	::=hh:mm:ss (where "hh", "mm", and "ss" are hours, minutes, and seconds, respectively, and represent time of day in 24-hour format)

```
response                      ::=field [sp field]...

out-of-range                  ::=?     (ASCII question-mark)
```

4.5 COMMAND RESPONSE PAIRS

This section lists possible command messages and the correct response messages (see *Appendix A, NOTATIONAL CONVENTIONS*, for an explanation of Extended Backus Notation).

4.5.1 COMMAND: INTERROGATE

The format of the command message is as follows:

```
command-message               ::=I [command ;] ...

command                       ::=param-code
```

An interrogate may be issued with no commands, if desired.

The format of the normal response is as follows:

```
response-message              ::={{R | r} [response ;]...}

response                      ::=field [sp field] ...
```

A response follows each command, issued in the same order as the commands.

The format of the error-response is as follows:

```
response-message              ::={E|e} error ;

error                         ::={BAD | [SYN location] | TBO}
```

See *Section 4.6.4, ERROR-RESPONSE*, for a description of responses.

4.5.2 COMMAND: SET

The format of the command message is as follows:

```
command-message              ::=S command ;

command                      ::=param-code sp field
```

The format of the normal response is as follows:

```
response-message             ::=A | a
```

The format of the error-response is as follows:

```
response-message             ::={E|e} error ;

error                        ::=BAD | {SYN location} | RDO | MAN | TBO
```

See *Section 4.6.4,* for a description of responses.

4.6 CODES

This section describes codes and responses for the DataPort system.

4.6.1 PARAMETER CODES (PARAM-CODES)

Table 4-1, Parameter Codes, specifies information used in the normal operation of the pump.

Table 4-1. Parameter Codes

Command Format	Description	LifeCare 5000 w/DataPort	Plum XLD	Plum A+
AHF	**A**larm **H**istory **F**irst: Moves the **A**larm **H**istory Pointer to the **F**irst set-of-alarm-histories (i.e. positions the pointer to the beginning of the ALARM HISTORY BUFFER, such that the pointer points to the most recent event placed in the buffer.) (Note: the AHF command is intended to be used with the AHR command. See the AHR command. AHF is a Set type command.)	**No**	**No**	**Yes**

Table 4-1. Parameter Codes				
Command Format	**Description**	**LifeCare 5000 w/DataPort**	**Plum XLD**	**Plum A+**
AHN	**A**larm **H**istory **N**ext Moves the **A**larm **H**istory Pointer to the **N**ext set-of-alarm- histories. (Note: the AHN command is intended to be used with the AHR command. See the AHR command. AHN is a Set type command.)	**No**	**No**	**Yes**
AHR	**A**larm **H**istory **R**ead a) Starting at the pointer position read the corresponding **set-of-alarm-histories** and format the information into a packet for transmission to the host. b) The transmit packet size **shall not** exceed 256 ASCII characters. c) By use of the AHF, and AHN commands in conjunction with the AHR command, the device **shall** allow the host to read at least the 30 most recent Alarm and Malfunction Condition code events. d) If an attempt is made to read beyond the end of the Alarm History Buffer, an End-Of-History condition **shall** result. (Note: ❑ The Alarm History Pointer position is selected by using the **AHF** and **AHN** commands. ❑ See the **EOH** error response code. ❑ This is an Inquiry type command. ❑ AHR does not change the Alarm History Pointer position.)	**No**	**No**	**Yes**

Table 4-1. Parameter Codes				
Command Format	**Description**	**LifeCare 5000 w/DataPort**	**Plum XLD**	**Plum A+**
ALR	**Whether the device is in an alarm state and, if so, what the reason is.** The response is zero or more alarm-codes, each preceded by a blank, signifying the current alarm conditions. The codes returned by this response appear in this document, and may consist of Alarm Type Codes, Warning Type Codes (such as a Low Battery Warning), and Malfunction Codes. (See *Section 4.6.3, ALARM CODES* for details.)	**Yes**	**Yes**	**Yes**
CEn	**Callback Enable**. Boolean (‘T’ or ‘F’). If enabled, device alarms when delivery on line “n” changes.	**No**	**No**	**Yes**
CFG	**How the device is configured.** Possible responses: ASC (Single Channel) ADC (Dual Channel) AMD (Multidose) MSC (Micro Single Channel) MDC (Micro Dual Channel) MMD (Micro Multidose) Possible Plum A+ responses: ADC; MDC (ADC means Macro Dual Channel for Plum A+)	**Yes**	**No**	**Yes**
CNF	**Concurrent Flow.** Boolean ('T' or 'F')	**Yes**	**No**	**Yes**
CV2	Container volume in line B. Amount of fluid in container before any was administered, in mL.	**Yes**	**No**	**No**
DC2	Dose Count in line B. Number of complete doses delivered of a multidose regimen.	**Yes**	**No**	**No**
DCn	**Drug Calculation Units**, Line n.	**No**	**No**	**Yes**

Table 4-1. Parameter Codes				
Command Format	**Description**	**LifeCare 5000 w/DataPort**	**Plum XLD**	**Plum A+**
DDn	Dose delivered in line n. Total volume delivered in the current dose, in m1. This number is reset at the beginning of each dose under a multidose regimen.	**Yes**	**No**	**No**
DI2	Dose Interval in line B. Time between start of one dose and start of next dose, in time format.	**Yes**	**No**	**No**
DKn	**Drug Concentration**, (mg / mL), (units/mL) on line n. (The response = BAD when the parameters are not setup.)	**No**	**No**	**Yes**
DLn	Dose Limit on line n. Maximum volume of fluid to deliver from line n, in mL.	**Yes**	**No**	**No**
DNn	**Drug Name**, on line n. (The response = BAD when the parameter is not setup.)	**No**	**No**	**Yes**
DO2	Dose Overfill on line B. Amount delivered in excess of Dose Limit when Enable Overfill is selected, in ml.	**Yes**	**No**	**No**
DP1	**Distal Pressure**, in psi.	**Yes**	**Yes**	**Yes**
DPL	**Distal Pressure Limit**, in psi.	**Yes**	**Yes**	**Yes**
DS1	**Delayed Start Time** on line A	**No**	**No**	**Yes**
DS2	**Delayed Start Time** on line B	**No**	**No**	**Yes**

Table 4-1. Parameter Codes				
Command Format	**Description**	**LifeCare 5000 w/DataPort**	**Plum XLD**	**Plum A+**
DSB	**D**ataPort **S**tatus while on **B**attery Returns one of two status codes: a) **“DEB”** - **D**ataPort **E**nabled on **B**attery. DataPort operation has been enabled for when the **I**nfuser is on battery. b) **“DDB”**- **D**ataPort **D**isabled on **B**attery. DataPort operation has been disabled for when the **I**nfuser is on battery. (Note: When the DataPort is Disabled for operation on Battery, the DataPort isolation circuitry will be shut down by the **I**nfuser.)	**No**	**No**	**Yes**
DTY	**Device-type.** (Formatted as a "qstring".) a) For LifeCare 5000: "LifeCare 5000". b) For LifeCare XLD: "LifeCare XLD". c) For Plum A+: **“Plum A+”**.	**Yes**	**Yes**	**Yes**
DVn	**Delivery rate (by volume) on line n, in mL/hr**	**Yes**	**Yes**	**Yes**
EC2	Enable Continued Delivery in line B. Boolean ('T' or 'F'). Continue secondary delivery even though the dose has finished. (Allowed only when "Nurse Callback" is enabled, but this is not true for Plum A+.)	**Yes**	**No**	**No**
EHF	**E**vent **H**istory **F**irst Moves the **E**vent **H**istory pointer to the **F**irst set-of-event-histories (i.e. positions the pointer to the beginning of the EVENT HISTORY BUFFER, such that the pointer points to the most recent event placed in the buffer.) (Note: the EHF command is intended to be used with the EHR command. See the EHR command. EHF is a Set type command.)	**No**	**No**	**Yes**

Table 4-1. Parameter Codes				
Command Format	**Description**	**LifeCare 5000 w/DataPort**	**Plum XLD**	**Plum A+**
EHN	**E**vent **H**istory **N**ext Moves the **E**vent **H**istory pointer to the **N**ext set-of- event-histories. (Note: the EHN command is intended to be used with the EHR command. See the EHR command. EHN is a Set type command.)	**No**	**No**	**Yes**
EHR	**E**vent **H**istory **R**ead a) Starting at the pointer position, read the corresponding **set-of-event-histories** and format the information into a packet for transmission to the host. b) The transmit packet size **shall not** exceed 256 ASCII characters. c) By use of the EHC, EHF, and EHN commands in conjunction with the EHR command, the device **shall** allow the host to read at least the 200 most recent events. d) If an attempt is made to read beyond the end of the Event History Buffer, an End-Of-History condition **shall** result. (Note: ❑ The Event History Pointer position is selected by using the EHF and EHN commands. ❑ See the EOH error response code. ❑ This is an Inquiry type command. ❑ EHR does not change the Event History Pointer.)	**No**	**No**	**Yes**

Table 4-1. Parameter Codes

Command Format	Description	LifeCare 5000 w/DataPort	Plum XLD	Plum A+
ENC	Enable Nurse Callback. Boolean ('T' or 'F'). If enabled, device alarms when secondary dose ends.	**Yes**	**No**	**No**
EO2	Enable Overfill Delivery in line B. Boolean ('T' or 'F'). Deliver until 110% of secondary dose is reached or container is empty, whichever comes first. If less than 90% is delivered when the container runs dry, an alarm is sounded. Otherwise, the "Nurse-Callback" and other "dose end" actions are taken.	**Yes**	**No**	**No**
EOH	**E**nd-**O**f-**H**istory (An Error Return) (See EOH, *Section 4.6.4, ERROR-RESPONSE*.)	**No**	**No**	**Yes**
FPL	**Front panel lockout. Boolean ("T" of "F")**	**No**	**Yes**	**Yes**
HBT	**Hours on Battery**. Number of hours operated on battery.	**Yes**	**Yes**	**Yes**
HTT	**Hours, Total**. Number of total hours operated.	**Yes**	**Yes**	**Yes**
MEM[addr, size]	The response to the MEM command is N values read from system RAM beginning at the specified address. The [addr, size] variables associated are address (1 to 4 digits of hex notation) size (N = 1 to 2 digits of hex notation).	**No**	**Yes**	**Yes**
ND2	Number of Doses on line B. Number of doses programmed for multi-dose delivery.	**Yes**	**No**	**No**
NSn	Returns the number of steps programmed for Multistep on line n.	**No**	**No**	**Yes**
PTW	**Patient Weight** (kg)	**No**	**No**	**Yes**

Table 4-1. Parameter Codes				
Command Format	**Description**	**LifeCare 5000 w/DataPort**	**Plum XLD**	**Plum A+**
SID	**Soft ID** of device. This is the only parameter [p1] which may be set. Parameter is supplied when command-type is "set", and omitted otherwise.	**Yes**	**Yes**	**Yes**
STA	Status. Codes signifying stopped, running, backpriming, etc. (See *Section 4.6.2, STATUS CODES*)	**Yes**	**Yes**	**Yes**
SWV	Current **Software Version**. Reply is of type "qstring".	**Yes**	**Yes**	**Yes**
THn	Returns the therapy on line “n”.	**No**	**No**	**Yes**
TN2	Time to Next dose in line B. Time between now and the beginning of the next dose in a multi-dose regimen, in time-format.	**Yes**	**No**	**Yes**
TPn [s]	**T**herapy **P**arameters line **n**: Returns the therapy parameters on line ìnî. The index parameter ìsî is required. The parameter ìsî refers to the therapy step or dose number.	**No**	**No**	**Yes**
TVn	**T**otal **V**olume delivered from line **n**. Total volume delivered from line n since device was turned on or total volume was cleared manually, in mL. Accumulates the value of all doses under a multidose regimen. (For Plum A+: returns the total volume delivered on line n since the device was started or since the total volume on line n was last cleared.)	**Yes**	**No**	**Yes**
VOL	**Total volume**. Sum of TV1 and TV2, in mL (For Plum A+: total volume delivered on both lines since the device was turned on or since the total volume was cleared.)	**Yes**	**Yes**	**Yes**

Table 4-1. Parameter Codes

Command Format	Description	LifeCare 5000 w/DataPort	Plum XLD	Plum A+
VTn	**VTBI, line n**, (Volume To Be Infused: The current amount of fluid remaining to be delivered to the patient on line n. If a therapy is in progress, the VTBI of the current therapy and current step is given. If a therapy is not in progress, the current non-therapy VTBI is given.)	**No**	**Yes**	**Yes**

Note:

1. A lowercase 'n' in a param-code indicates a channel specific or cassette specific datum, and may have values '1' or '2', unless otherwise noted. Typically channel '1' is the primary channel and channel '2' is the secondary or piggyback channel.
2. Channel '1' and channel '2' equate to line A and line B, respectively, for Plum A+.

4.6.2 STATUS CODES

Codes used to respond to the STA (status inquiry) command are listed in *Table 4-2, Status Codes*.

Table 4-2. Status Codes

Type	Description	LifeCare 5000 w/DataPort	Plum XLD	Plum A+
ALM	Alarm	**No**	**No**	**Yes**
APR	Autopriming	**No**	**No**	**Yes**
BPR	BackPriming	**No**	**Yes**	**Yes**
DLY	Pumping Delayed	**No**	**No**	**Yes**
MFN	Malfunction	**Yes**	**Yes**	**Yes**
PCN	Pumping Concurrent.	**Yes**	**No**	**No**
PKV	Pumping KVO Mode	**Yes**	**Yes**	**No**
PMP	Pumping	**No**	**No**	**Yes**
SDO	Stopped, Door Open.	**Yes**	**Yes**	**Yes**
STP	Stopped	**Yes**	**Yes**	**Yes**

4.6.3 ALARM CODES

Alarm codes describe an alarm condition within the pump and are of lexical type “name”.

When the host computer asks for the alarm status, the pump responds with one of the following alarm messages:

Response:

`alarm-type`

Where the value of alarm-type is listed in *Table 4-3, Alarm Messages*:

Table 4-3. Alarm Messages

AlarmType	Description	LifeCare 5000 w/DataPort	Plum XLD	Plum A+
AD1	**A**ir in line, **D**istal	**Yes**	**No**	**No**
ADB	**A**ir in line, **D**istal (**B**olus)	**No**	**Yes**	**Yes**
ADC	Air in line, Distal (Cumulative)	**No**	**Yes**	**Yes**
AP1	Air in line, Proximal	**Yes**	**Yes**	**No**
APA	**A**ir in line, **P**roximal line **A**	**No**	**No**	**Yes**
APB	**A**ir in line, **P**roximal line **B**	**No**	**No**	**Yes**
APT	**A**ir in line, **P**roximal **T**otal	**No**	**No**	**Yes**
BDP	**B**attery **D**e**p**leted	**No**	**No**	**Yes**
BPR	BackPriming	**Yes**	**No**	**No**
CHK	Check settings, rate or VTBI not set.	**No**	**Yes**	**No**
CS1	Cassette Failure, Valve Failure	**Yes**	**Yes**	**Yes**
DCO1	**D**oor, **C**assette, **O**pen, **Door-Open**	**Yes**	**Yes**	**Yes**
DEn	Dose End, line n	**Yes**	**No**	**No**
DTL	Control Dial turned when lockout switch is on.	**No**	**Yes**	**No**
FDF	Flow Detector, False (unplugged while pumping)	**Yes**	**No**	**No**
FDT	Flow Detector, True (plugged in while pumping)	**Yes**	**No**	**No**
FLF	Flow, False	**Yes**	**No**	**No**

Table 4-3. Alarm Messages

AlarmType	Description	LifeCare 5000 w/DataPort	Plum XLD	Plum A+
LCK	LockOut Enabled	**No**	**No**	**Yes**
LOV	**L**ock**O**ut **V**iolation	**No**	**No**	**Yes**
MALxxx	**Malfunction** of type xxx where "xxx" is the code reported on the screen.	**Yes**	**Yes**	**Yes**
NAA	**N**o **A**ction **A**larm (No operator action and no delivery for 2 minutes when delivery parameters are being entered or modified.)	**No**	**No**	**Yes**
NCn	**N**urse **C**all **B**ack, line n	**No**	**No**	**Yes**
OBP	Proximal Occlusion on secondary during backpriming	**No**	**Yes**	**No**
OCP	Proximal Occlusion	**No**	**Yes**	**No**
OD1	**O**cclusion, **D**istal	**Yes**	**Yes**	**Yes**
OK	**No alarms**	**Yes**	**Yes**	**Yes**
OPn	**Occlusion, Proximal**, line n	**Yes**	**No**	**Yes**
PR1	Pressure out of range	**Yes**	**No**	**No**
RL	in **Reset too Long** or **system Idle too Long** (over 5 minutes) (2 minutes for Plum A+)	**Yes**	**Yes**	**Yes**
URC	Unrecognizable Cassette	**No**	**No**	**Yes**
VTBn	Programmed **VTBI completed** in line n.	**No**	**Yes**	**Yes**

4.6.4 ERROR-RESPONSE

Error responses and descriptions are listed in *Table 4-4, Error Response*.

Format:

```
error-response              ::={E|e} error-description ;

error-description           ::=error-opcode [sp error- parameter]
```

Table 4-4. Error Response	
Format	**Meaning**
BAD	Unspecified parameter error **Note:** A general-purpose error-response when parameters are invalid. To be used by "dumb" devices and when no other description can be found.
EOH	**E**nd-**O**f-**H**istory Sent to the host when the end of the EVENT HISTORY BUFFER or the ALARM HISTORY BUFFER is encountered.
INV	Invalid parameter error **Note:** Response when a particular parameter is invalid.
RDO	**Read Only:** Parameter is read only; host cannot set it.
MAN	"set" command ignored because value is being edited at the front panel.
SYN space location	**Syntax error:** Unspecified syntax error in parameter. (Response format: **SYN** space **location**.) Parameters: location: Error position relative to the beginning of the packet. (position zero is the first character of the packet) **Note:** A crude diagnostic tool for host programmers. Also provides a simple means of identifying "wrong character" errors.
TBO	**T**ransmit-**B**uffer **O**verflow. The device's transmit buffer has overflowed in forming the response to the last message. This occurs if the response packet contains **more than 256 characters**.

Note: BAD, INV, RDO, MAN, SYN, and TBO may appear with a prepended “E” in the response packet.

4.7 ALARM HANDLING

If there are any alarm conditions present at the time a pump is preparing its reply to a message, the pump changes its response code to lowercase.

This page intentionally left blank.

Section 5
PROGRAMMING GUIDELINES

5.1
INSTALLATIONS WITH MULTIPLE PORTS

In some installations, it may be desirable to communicate with pumps connected to more than one serial port. If the host computer is equipped with enough serial ports, the host software may be programmed to use the multiple ports. If, however, there are not enough ports on the host computer to connect all the pumps to be monitored, a switched serial port (computer activated switch, or CAS) is necessary. Under certain circumstances, however, unwanted characters may be sent to the pumps on one of the output ports. A solution is to send a flush-character to the pumps before sending command packets whenever it is suspected that unwanted characters may have been sent previously.

5.2
ALGORITHM FOR CHECKING ALL PUMPS

This section describes a method of communicating with a number of pumps to:

A. Obtain the status and location of pumps known to be in use.

B. Identify formerly active pumps that are turned off, are unplugged from their communication lines, or otherwise cease to communicate.

C. Identify new pumps which are put into service.

D. Compensate for the actions of some computer activated switches (CAS) by using the flush character.

Priority is given to obtaining the status of known pumps (tasks A and B) over identifying new pumps. Prioritization reduces the inherent delay between an abnormal condition of a pump and the monitoring system recognizing that condition. This delay can be further reduced by distributing task C over a longer period of time.

The following “sample data” and “pseudocode” are not intended to dictate design or to be implemented in the monitoring system.

SAMPLE DATA:

pump_table :

soft-id	room-#	hard-id	status	total vol. line1	total vol. line2	alarm
794	4	5	PMP	51.0	15.2	-
315	4	1	STP	348.5	0.0	-
790	3	2	PMP	7.53	17.1	-
5004	3	1	PMP	39.2	20.2	DE2 (dose-end, line 2)

room_table : (Ordered by room number)

room	patient I.D.	patient-name	CTS stuck-low (T/F)
3301	0	(empty)	F
3302	0	(empty)	F
3303	456-78-9753	smith	F
3304	003-19-1234	jones	F
-			
-			

drug_pat : (Optional, no particular ordering). Links prescriptions with patients. This data could be used in a patient-display or a room-display to show the patient medication)

Rx-ID	patient-id	drug-id
3	456-78-9753	LYTC
10	003-19-1234	NACL
9	003-19-1234	M05

drug_table: (Optional. Used with drug_pat table)

drug-id	drug-name
LYTC	lidocaine
M05	morphine, 0.5 mg/ml
NACL	saline solution, 0.9%

5.2 ALGORITHM FOR CHECKING ALL PUMPS

```
            PSEUDOCODE:
Initialize device/location table to empty
       (one row for each hard ID possibly present on line)

while true do
              /*————————————————————————————----------------------*/
             /* CHECK OLD PUMPS, FIND NEW ONES                              */
              /*————————————————————————————-----------------------*/
for each line with CTS high (—> active_line) do
       send flush-character       /* reset pumps' receive-buffers         */
                                  /* check old pumps                        */
       for each known device on the line (—> device_id) do
                                  /* get desired info from device  */
             send_message (in: hard-id-out, msg_out;
                    out: error_code, msg_in, hard_id_in, soft_id_in)
                                         /* handle response               */
             case error_code:
             ok:
                    decode message, update status of device


             garbage:
                    report “garbage: line x, device-type y, location z”
             timeout:
                    report “device removed”
             wrong_device:
                    report “wrong device: asked for x, received y”


             end /* case error_code */
       end /* for each known device on the line */
             /*————————————————————————————---------------------*/
             /*—  FIND NEW PUMPS                   ——----————————-*/
             /*——————————————————————————----------------------——*/
       for each unused location on the line (—> hard_id_out) do
                                  /* see if device is there         */
             msg_out = “I”           /* (“interrogate”, w/ no args)    */
             send_message_to_loc (in: hard_id_out, msg_out;
                    out: error_code, msg_in, soft_id_in, hard_id_in)
                                       /* handle response               */
             case error_code:
             ok:
                    if hard_id_in is not equal to hard_id_out
                           report “hard ID failure, line x, loc. y
                                  reports loc. z"
                    else
                           report “new device, id x on line y, location z”
```

```
                garbage:
                      report “garbage: line x, device-type y,
                            location z"
                timeout:
                      report “device removed”
                wrong_device:
                      report “wrong device: asked for x, received y”
                end /* case error_code */
          end /* for each unused location on the line */

    end /* for each live line */
          /*——————————————————————————*/
           /*  Check “dead” line  ——————————————-*/
          /*——————————————————————————*/
    for one “dead” line (—> inactive_line_no) do

          for each location on the line (—> hard_id_out) do
                            /* see if device is there         */
                msg_out = “I”        /* (“interrogate”, w/ no args)    */
                send_message_to_loc (in: hard_id_out, msg_out;
                      out: error_code, msg_in, soft_id_in, hard_id_in)
                            /* handle response
                                    */
                case error_code:
                ok:
                      if hard_id_in is not equal to hard_id_out
                      report “hard ID failure, line x, loc. y
                            reports loc. z"
                      else
                            report “CTS failure, line (inactive_line_no)”
                            in room_table, mark CTS as bad

                garbage:
                      report “garbage: line x, device-type y,
                            location z"

                timeout: (OK—no action needed)

                wrong_device:
                       report “wrong device: asked for x, received y”
                      end /* case error_code */

                end /* for each location on the line (—> hard_id_out) */
          end /* for one “dead” line (—> inactive_line_no) */
    end /* while true */
```

5.3 PROGRAMMING EXAMPLES

The following sample source code for CRC calculation was compiled and run using Microsoft® C, version 6.0. The result of running the sample program is:

The CRC of “T500;IALAR;DV1;DV2;<CR>” is 622A

```
/*file: crc_ex.c: C-language unit for performing CRC calculations,    */
/*            using a shortcut method.                                 */
/* Author: V. Le, adapted by Chris Lynch.                              */
/* Date  : NOV 91                                                      */
/* Reference:                                                          */
/*    Campbell, Joe, “C Programmer's Guide to Serial Communications”, */
/*             Howard Sams and Company, a division of MacMillan.      */
/* Modifications:                                                      */
/*                                                                     */
/*********************************************************************/
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#define CRC_LEN 4                       /* # of chars in CRC           */
#define BLANK 0x20
#define MAX_ASCII 256                   /* # of 8-bit values           */
#define CCCITT_REV 0x8408               /* CRC polynomial—CRCCITT,    */
                                        /*   reversed                  */
extern char *crc_get (char *packet, int len) ;
extern int crchware (unsigned short data, unsigned short genpoly,
    unsigned short accum) ;
extern void i_crc (void) ;
static int crc_tab[MAX_ASCII];          /* CRC table                   */

/*——————————————————————————————————*/
/*                        PROTOTYPES                                   */
main()
{
    char *packet = “T500;IALR;DV1;DV2;”;
    i_crc();
    printf (“The CRC of \”%s\" is %s.\n", packet,
    crc_get (packet, strlen(packet)));
}
/*——————————————————————————————————*/
```

```
/*******************************************************************/
/*FD crc_get: calc. CRC for a packet                               */
/* Author: Vu Le                                                   */
/* Date  : Oct 91                                                  */
/* Description: This function will calculate a crc for the packet,  */
/*              using table “crc_tab” to get the “combining value” of */
/*              each character.                                    */
/* Returns:  pointer to static crc-buffer.                         */
/* Parameter: packet (char *), Global variable - crc_table.        */
/* Constraint: None                                                */
/* Interrupt: None                                                 */
/*******************************************************************/

char *crc_get(packet, len)           /*FH                          */
char *packet;                         /*P chars in packet           */
int len;                              /*P # of chars to “CRC”       */
{
    int i;                            /* loop-var                   */
    int data;                         /* temp. var                  */
    unsigned short crccitt;           /* numeric version of CRC     */
    static char crc_buf[CRC_LEN+1];   /* returned value             */
    /* begin func crc_get */
    crccitt = 0;                      /* preload with zeroes        */
                                      /* combine data with precalc- */
                                      /*   ulated crc “combining    */
                                      /*   values"                  */
    for (i = 0; i < len; i++)
    {
        data= crccitt ^ (int) packet[i];
        crccitt=(crccitt >> 8) ^ crc_tab[data & 0x00ff];
    }
    sprintf(crc_buf,"%4x",crccitt);   /* result ==> ASCII hex       */
    for (i=0; crc_buf[i] != '\0';i++) /* replaces blanks with zeroes,*/
    {                                 /*   lower- with upper-case   */
        crc_buf[i] = (char) toupper ((int) crc_buf[i]) ;
        if (crc_buf[i] == BLANK)
            crc_buf[i]='0';
    }
    return(crc_buf);
}                                     /* end func crc_get */
```

```
/*********************************************************************/
/*FD crchware : calculate the CRC value of a data-byte,              */
/*F      given the data, a 16-bit polynomial, & the old CRC value from*/
/*F      the previous call.  Simulates the method used to generate    */
/*F      CRC's in hardware.                                           */
/*F  Returns: the next value of the CRC                               */
/*********************************************************************/
int crchware (data, genpoly, accum)    /*FH                          */
unsigned short data;                   /*P data-byte in low byte      */
unsigned short genpoly;                /*P coefficients of poly.      */
unsigned short accum;                  /*P old CRC value              */
{
    int i;                             /* loop-counter                */
    int lsb_diff;                      /* boolean: lsb's of data &    */
                                       /*   accum are different       */
    /* begin func crchware */
    for (i = 0; i < 8; i++)            /* for each data bit...        */
    {                                  /* are lsb's of data and       */
                                       /*   accum different?          */
        lsb_diff = ((data ^ accum) & 0x0001) ;
        if (lsb_diff)
                                        /* YES: shift and subtract    */
                                        /*      polynomial            */
            accum = (accum >> 1) ^ genpoly;
        else
            accum >>= 1;               /* NO: just shift              */
        data >>= 1;                    /* look at next data-bit       */
    }
    return (accum);
}   /* end func crchware */
/*********************************************************************/
/*FD i_crc : initialize crc-unit. Generates table of CRC components   */
/*F  This function must be called once before crc_get() is called.    */
/*********************************************************************/
void i_crc ()                          /*FH */
{
    int i;                             /* loop-var                    */
    /* begin func i_crc */
        /* for each 8-bit value, simulate shifting it through a */
        /*    16-bit CRC shift-register, preloaded with zeroes, */
        /*    and store the result.                              */
    for (i = 0 ; i < MAX_ASCII ; i++)
        crc_tab[i]= crchware (i,CCCITT_REV,0);
}   /* end func i_crc */
```

5.4
DIALOG BETWEEN HOST COMPUTER AND PUMPS

Figure 5-1, Sample Query of Pump by Host Computer, illustrates a typical transmission by a host computer and a pump's response.

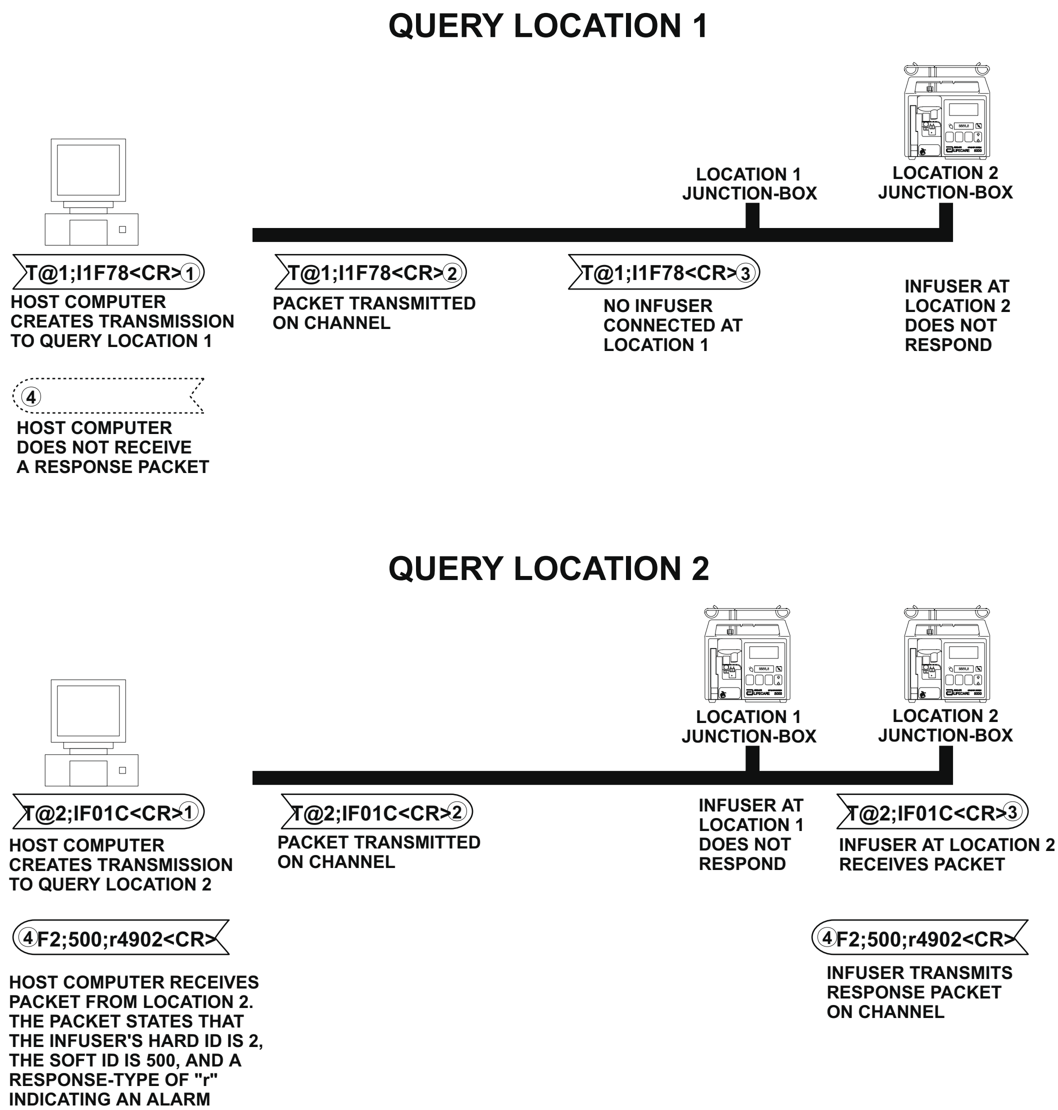


Figure 5-1. Sample Query of Pump by Host Computer

5.5
DATAPORT COMMUNICATION TEST

The following program, written in BASIC, tests the DataPort communication hardware of the pump. This program may be run if a DataPort communications hardware malfunction is suspected. Running this program successfully (e.g., TEST PASSED) confirms that the pump can receive and transmit packets.

To perform the DataPort communication test, use a direct connection between the host computer and the pump, then run the following program:

```
 10 REM ************************************************************
 20 REM ***
 30 REM * Program:     LCTEST.BAS        REV:1.01
 40 REM * Description:
 50 REM     This program will test the hardware of the LC5000
 60 REM     DATAPORT system.  A single packet will be sent to the
 70 REM     pump and one will be expected in reply. The CRC is
 80 REM     pre-calculated.  This program will communicate with only
 90 REM     one pump—communication with multiple pumps on a single
100 REM     bus line will not function with this program.
110 REM * Interpreter : IBM BASIC Version 2.0
120 REM ***
130 REM ************************************************************
140 REM *** Beginning of program.
150 REM *** Clear computer screen.
160 CLS
170 REM *** Indicate “no packets received”.
180 LCSTR$ = “”
190 LCLEN = 0
200 REM *** If error then report failure of computer port.
210 ON ERROR GOTO 450
220 REM *** Activate communication port on the computer:
230 REM ***   port = 1, baud rate = 1200, parity = none,
240 REM ***   data bits = 8, stop bits = 1.
250 COM(1) ON
260 ON COM(1) GOSUB 530
270 OPEN “COM1:1200,N,8,1" AS #1
280 REM *** Send packet to pump:
290 REM ***   Flush and ask for status from Hard-ID 0.
300 PRINT #1,CHR$(3);
310 PRINT #1,"T@0;ISTA;2FAD"
320 REM *** Wait for a reply packet from pump.
330 FOR I=1 TO 25000
340 NEXT
350 REM *** Test for a received packet. If received packet is empty
360 REM *** then test FAILS. Otherwise, test PASSes and the received
370 REM *** packet is printed.
380 REM ***
390 IF LCLEN = 1 THEN GOTO 400  ELSE GOTO 420
400 PRINT “** TEST PASSED, received packet:”;LCSTR$
410 GOTO 500
420 PRINT “** TEST FAILED, no communication from pump.”
430 GOTO 500
440 REM *** Communication port error.
450 PRINT CHR$(13);CHR$(13);CHR$(13)
460 PRINT “Communication ERROR on COM1 port—check cable connections.”
470 GOTO 510
480 REM *** Close communication port.
```

```
490 COM(1) OFF
500 CLOSE
510 END
520 REM **** Receive the packet.
530 INPUT #1,LCSTR$
540 COM(1) OFF
550 LCLEN = 1
560 RETURN
570 REM *** End of program.
```

Section 6
TROUBLESHOOTING

6.1
TROUBLESHOOTING GUIDE

Table 6-1, Troubleshooting Guide, lists DataPort malfunctions and corrective action.

Note: Before troubleshooting the pump, verify that your software performs at least one retry which is preceded by a flush character.

Table 6-1. Troubleshooting Guide

Symptom	Probable Causes	Corrective Action
Pump does not reply to packet sent by host computer	❏ Pump not connected to cable or not connected to DataPort bus	Confirm all cable and junction-box connections
	❏ Malfunction in host computer	Check host computer. Confirm validity of transmitted packet by monitoring the host computer TX line (see *Section 4.4, MESSAGE STRUCTURE*)
	❏ Pump not functional either because it is off or there is a malfunction	Confirm pump is active or not malfunctioning
	❏ Pump with incorrect software revision	Check pump software revision
	❏ Host computer transmitting with wrong Checksum, Device Address, baud rate, framing bits, or parity	Check host computer communication parameters
	❏ Corrupted soft ID Note: A detected change in soft ID resets device soft ID to zero	Re-enter soft ID
	❏ RX circuit stuck in mark state due to hardware failure	Replace pump

Table 6-1. Troubleshooting Guide

Symptom	Probable Causes	Corrective Action
Pump does not reply to packet sent by host computer (cont.)	❑ Damaged or lost end-of-packet character or corrupted data within packet (should be detected as an incorrect CRC)	Confirm valid transmission by source. If valid, check for electrostatic or electromagnetic interference
	❑ Incorrect hard ID due to damaged Junction-Boxes or multiple broken or stuck lines	Replace Junction-Boxes or pumps
	❑ TX circuit is stuck on (High or Low) due to hardware failure	Replace host computer communications port
Pump will not respond to packets directed to any non-zero hard ID, but will respond to global packets and packets directed to soft ID	❑ Invalid hard ID parity due to damaged Junction-Box or single broken or stuck line	Replace Junction-Box or replace pump
Packets are received incorrectly by pump or host computer	❑ Cable disconnected while transmission in progress	Check condition of connector. Re-attach connector
	❑ Bus contention due to more than one pump at a Device Address	Confirm hard IDs on bus are unique
	❑ Bus wire length or electrical signals do not meet EIA-232-D standards. Leads can be open or shorted	Use port that conforms to EIA-232-D standard. Replace cables
	❑ Electromagnetic interference from adjacent equipment	Remove or repair source of interference
	❑ Electrostatic discharge causing noise interference at host computer, pump, or cabling	Remove source of interference or change cabling route
	❑ Bus traffic resulting from a connection to a non-LifeCare 5000 Plum1.6 Pump With DataPort	Disconnect non-conforming equipment

Table 6-1. Troubleshooting Guide

Symptom	Probable Causes	Corrective Action
	❏ Damaged or lost end-of-packet character or corrupted data within packet (should be detected as an incorrect CRC)	Confirm valid transmission by source. If valid, check for electrostatic or electromagnetic interference
Host receives garbled responses to packets sent to pump	❏ Duplicate pump hard or soft IDs	Re-enter and confirm that hard and soft IDs on a bus are unique
Pump responds to packet that was not addressed to it	❏ Corrupted soft ID **Note:** A detected change in soft ID causes device soft ID to reset to zero	Re-enter soft ID
Host computer detects pumps that are not present	❏ CTS stuck high due to hardware failure in Junction-Box	Replace Junction-Box
Addressing a pump by hard ID works sporadically	❏ Erratic hard ID **Note:** A pump does not consider a hard ID valid until it has remained stable for one second	Replace Junction-Box

This page intentionally left blank.

Appendix A
NOTATIONAL CONVENTIONS

Throughout this manual, the syntax of packets is described with Extended Backus Notation (EBN), a set of symbols commonly used to describe computer languages. EBN symbols are explained below. A glossary also appears in *Appendix B, GLOSSARY*.

NOTATION VARIABLE: Names a constituent of the protocol. It is a sequence of lowercase letters, digits, and hyphens. (A variable may not start or end with a hyphen and may not start with a digit.) Notation variable represents information that must be supplied by the user. It is defined formally or informally in preceding adjacent text.

NOTATION CONSTANT: Stands for itself and is represented by capital letters or by special characters. A notation constant must be written as shown in the definition (see "is defined to be").

SYNTACTICAL UNIT: A notation variable, a notation constant, or a collection of notation variables, notation constants, and notation symbols enclosed in braces or brackets.

VERTICAL BAR (|): Read "or." A choice must be made between the item to the left of the bar and the item to the right of the bar.

BRACES ({ AND }): Used for grouping or to indicate that a choice is to be made among the syntactical units contained within the braces.

BRACKETS ([AND]): Represents an option. The syntactical units enclosed within brackets can be omitted.

IS DEFINED TO BE (::=): Denoted by the composite symbol ::= used in the following manner: defined type ::= definition.

ELLIPSIS (...): Indicates that the preceding syntactical unit may be repeated one or more times.

This page intentionally left blank.

Appendix B
GLOSSARY

BAUD RATE: Baud rate is the number of bits transmitted per second within a character frame.

BROADCAST: Message intended for all pumps on the channel. Also, a message intended for one pump on the channel, but received by all pumps on the channel.

BUS: Topology in which stations (pumps) are attached to a linear (i.e., non-branching, non-looping) transmission medium. Transmissions propagate the length of the medium and are available to all pumps. Used interchangeably with “channel”.

CHANNEL: Transmission path through which a message travels between communicating pumps. Used interchangeably with “bus.”

CHARACTER FRAME: Data which is sent to communicate one character of information. Also, the amount of time required to send one character (including start- and stop-bits) at the baud rate in use.

CHECKSUM: Method used to ensure the integrity of a received message. The transmitting program calculates a number and attaches it to the message. The receiving program performs the same calculation. If the results differ, an error is known to have occurred.

COMPUTER ACTIVATED SWITCH: Electronic device which allows one input/output port on a host computer to be connected to a number of different input/output ports, known as switched ports. See also modem-port and switched port. An example of such a device is the model number CAS-161A, from Western Telematics, Inc.

COMMAND: Portion of a message which tells a device to do something. Multiple commands may be present in a message.

CRC: Cyclic Redundancy Check. A sophisticated type of checksum capable of detecting error-bursts.

CRC-CCITT-REV: A particular form of CRC used by DataPort software. The details of this CRC are specified by Comite Consultatif Internationale de Telegraphique et Telephonique (CCITT).

DEVICE: For purposes of this manual, a medical instrument attached to a host so it can be monitored remotely.

FIELD: Sequence of characters which indicates a data value in a message. There may be several fields (or none) within a particular command or response.

FLUSH-CHARACTER (CONTROL C): Character which is sent by the host to the pump as a means of recovering from errors.

FRAME: Character frame.

HARD ID (ALSO CALLED LOCATION ID): For a Plum A+ pump, its hard ID is an integer between 0 and 15 that the pump senses through the DataPort connector on the back of the pump. Hard ID values between 1 and 15 are supplied by an attached junction-box when using a bus configuration. No hard ID may be used more than once among a set of junction-boxes connected on one bus.

All Plum A+ pumps connected on a bus will reply to a message directed to a hard ID of 0. Hard ID of 0 is thus restricted for use in the direct configuration only. The Hospira supplied cables listed in *Appendix E, DataPort Accessories List,* present a hard ID of 0 to the pump.

HOST: Computer used to monitor pump.

INTER-CHARACTER TIMEOUT: Maximum amount of time which a pump may delay between each character it transmits.

JUNCTION-BOX: The device attached to a bus to enable communications with a host. The junction-box also supplies a hard ID to the attached pump.

LEXICAL TYPE: Characteristic of a field in a message. Examples are name and number.

LOCATION: The hard ID read by the pump.

MESSAGE: Sequence of bytes which conveys information. Only one message is transmitted inside a packet.

MODEM-PORT: Input/output port on a computer-activated switch which is connected to the host. See also computer activated switch and switched port.

OPERATOR: Person who interacts with the host computer.

PACKET: Sequence of characters consisting of a message and other information required to transport it.

PROTOCOL, COMMUNICATION: Set of rules which allows two or more computers to transfer data

RESPONSE: Portion of a message sent by a pump in response to a command from the host. Multiple responses may be present in a message. This information may be only as little as an acknowledgment that the previous message was correctly received.

RESPONSE TIMEOUT: Maximum amount of time a pump may delay before transmitting the first character of its response.

SOFT ID: For a Plum A+ pump, its integer between 0 and 99999999. This ID is used to identify the pump among the pumps that may be connected with the DataPort. The soft ID is stored in the non-volatile memory of the pump. No soft ID may appear more than once among the pumps to be connected at a particular customer site. The soft ID of a pump may be set by a DataPort command or by using the touchswitches on the front panel of the pump.

All Plum A+ pumps connected on a bus will reply to a message directed to a soft ID of 0. A soft ID of 0 is thus restricted for use in the direct configuration only.

SWITCHED PORT: Input/output port on a computer-activated switch which is connected to the junction box or device. See also modem-port and computer activated switch.

TERMINATOR CHARACTER (CONTROL M): Single ASCII character which may only appear at the end of a packet. Also called the end-of-packet character.

TREE CONFIGURATION: Configuration where a host is connected to multiple bus-structures.

WALL JACK: Receptacle mounted in the wall.

This page intentionally left blank.

Appendix C

SUGGESTED READINGS

"Plum A+ with DataPort System Operating Manual." Hospira.

Tanenbaum, Andrew. "Computer Networks." Englewood Cliffs, NJ: Prentice-Hall, 1981.

McNamara, John. "Technical Aspects of Data Communication." Digital (DEC) Press, 1982.

Campbell, Joe. "C Programmer's Guide to Serial Communications." Howard SAMS and Company, 1987.

Note: The "C Programmer's Guide to Serial Communications" also contains information about the method used to compute CRCs rapidly (*see Section 5.3*).

EIA standard # EIA-232-D (revision of EIA-232-C). "Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange." Electronic Industries Association (EIA), January, 1987.

This page intentionally left blank.

Appendix D
SUGGESTED SOFTWARE LIBRARIES

Hospira suggests that programmers use available software libraries where feasible in order to provide more robust code. Some functions which can be supplied in software libraries include:

- ❑ Communications
- ❑ User interface
- ❑ Task switching (for constructing an asynchronous system architecture)
- ❑ Data management (for crash-resistant data logging)
- ❑ Memory management (for use of extended and expanded memory)
- ❑ Printing (for printing graphics)

This page intentionally left blank.

Appendix E
DATAPORT ACCESSORIES LIST

Plum A+ with DataPort accessories are listed below:

LIST	DESCRIPTION
12074	Junction-Box
11431-01	Plum to PC cable. 8-foot. Male DB-15 to female DB-9 connector
11431-02	Plum to PC cable. 8-foot. Male DB-15 to female DB-25 connector
11431-03	Junction-Box to PC cable. 8-foot. 6-wire, 6-pin modular connector to female DB-9 connector
11431-04	Junction-Box to PC cable. 8-foot. 6-wire, 6-pin modular connector to female DB-25 connector
11431-06	Junction-Box to Junction-Box cable. 2-foot. 6-wire, 6-pin modular connector to 6-wire, 6-pin modular connector
11431-07	Junction-Box to Junction-Box cable. 4-foot. 6-wire, 6-pin modular connector to 6-wire, 6-pin modular connector
11431-08	Junction-Box to Junction-Box cable. 8-foot. 6-wire, 6-pin modular connector to 6-wire, 6-pin modular connector

This page intentionally left blank.

Appendix F

DRAWINGS

Complete schematics, interconnect diagrams, and functional block diagrams are available in the Plum A+ with DataPort Technical Service Manual. *Figure F-1* through *Figure F-3* illustrate Junction-Box, cables, and DataPort circuitry.

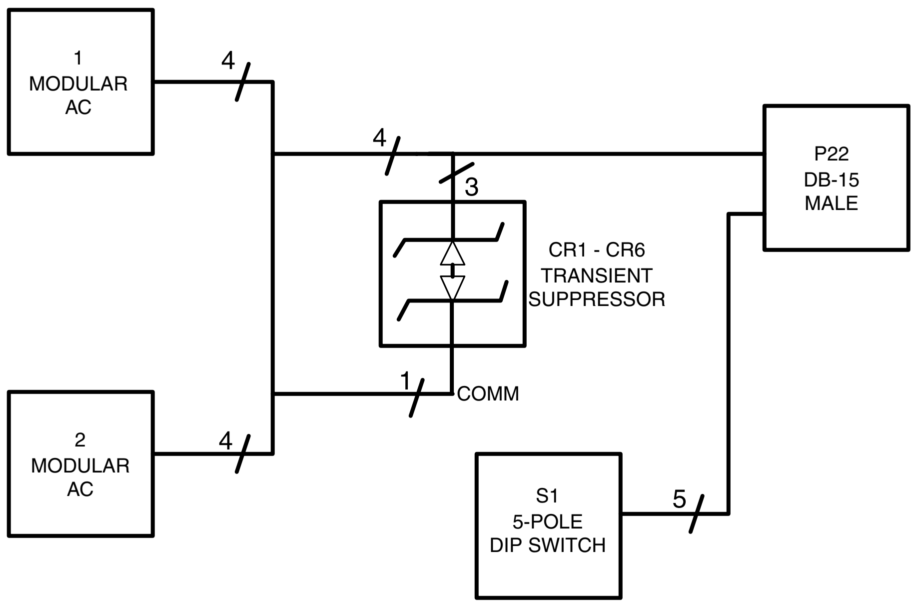
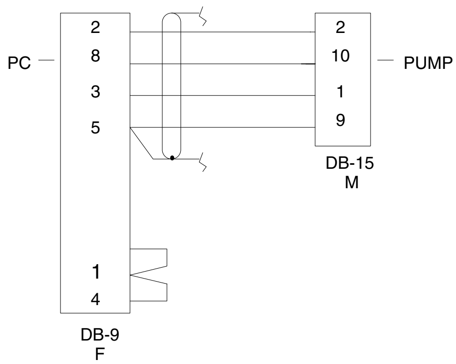
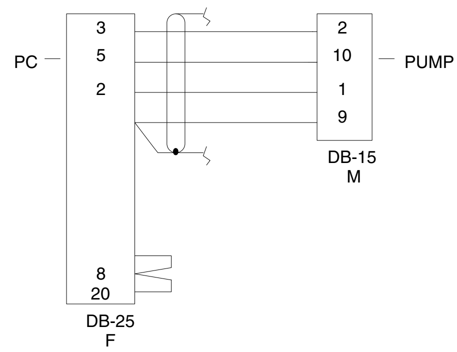


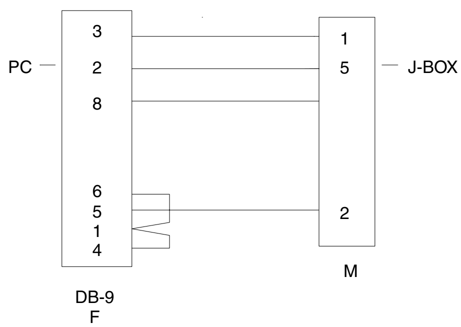
Figure F-1. Junction-Box



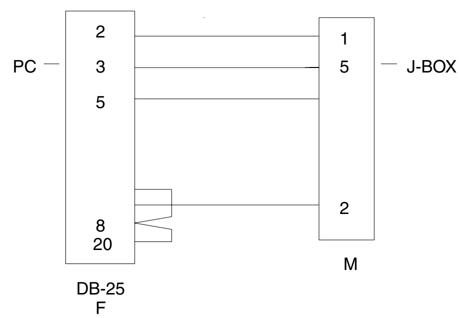
(a) DB-9 FEMALE TO DB-15 MALE CONNECTOR



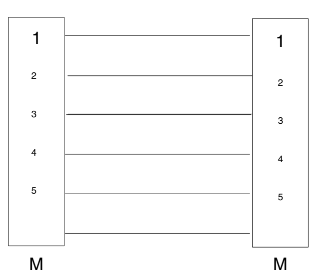
(b) DB-25 FEMALE TO DB-15 MALE CONNECTOR



(c) DB-9 FEMALE CONNECTOR TO MODULAR CONNECTOR



(d) DB-25 FEMALE CONNECTOR TO MODULAR CONNECTOR



(e) 6-PIN MODULAR CONNECTOR TO 6-PIN MODULAR CONNECTOR

Figure F-2. DataPort Cable Schematic

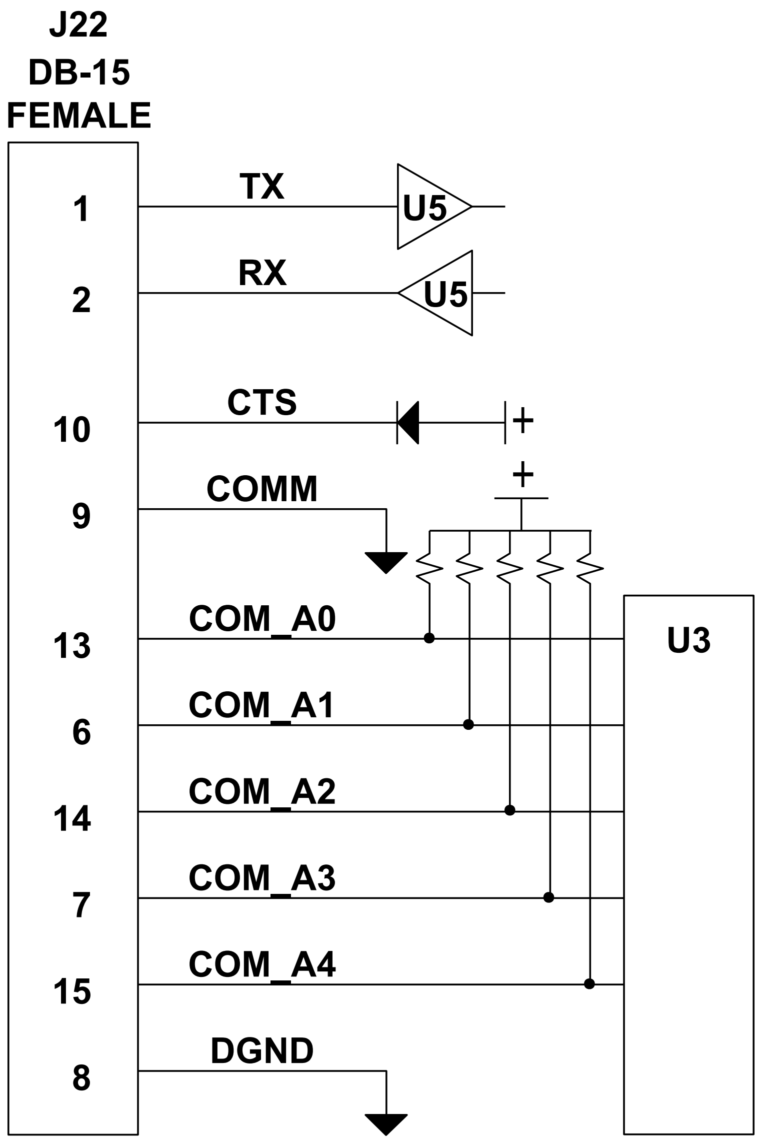


Figure F-3. DataPort Circuitry

This page intentionally left blank.

For customer service within the United States, contact:

1-877-9HOSPIRA or 1-877-946-7747

For technical assistance, product return authorization, and to order parts, accessories, or manuals within the United States, contact Hospira Technical Support Operations.

1-800-241-4002

To order parts using the online eCatalog, download technical publications, technical training courses, and additional services, visit the website at:

www.hospiraparts.com

After authorization, ship prepaid product returns to the following address:

Hospira, Inc.
Technical Support Operations
755 Jarvis Drive
Morgan Hill, CA 95037

Note: Outside the U.S., contact your local Hospira sales office.

CAUTION: Federal (USA) law restricts this pump to sale by or on the order of a physician or other licensed practitioner.

WARNING

POSSIBLE EXPLOSION HAZARD EXISTS IF THE PUMP IS USED IN THE PRESENCE OF FLAMMABLE ANESTHETICS.

Covered by the following U.S. Patents: 4,552,336, 4,639,245, and 4,703,775.

®
NRTL/C
CSA 22.2
No. 125
UL 544

CSA is a registered trademark of the Canadian Standards Association. The use of NRTL/C adjacent to the CSA mark indicates that the product has been certified by CSA to U.S. and Canadian standards. CSA has been accredited by the U.S. Occupational Safety and Health Administration (OSHA), as a Nationally Recognized Test Laboratory (NRTL).

UL listing does not cover Plum A+ sets.

Printed in USA

Hospira, Inc., Lake Forest, IL 60045, USA