1782-JDCE
Revision 6
DeviceNet Serial Gateway
User's Manual

Western Reserve Controls, Inc.

Although every effort has been made to insure the accuracy of this document, all information is subject to change without notice. Western Reserve Controls, Inc. assumes no liability for any errors or omissions in this document or for direct, indirect, incidental or consequential damage resulting from the use of this document.

Document 25.0
Rev 6.04
July 2003

Western Reserve Controls, Inc.
1485 Exeter Road
Akron OH 44306
330-733-6662 (Phone)
330-733-6663 (FAX)
sales@wrcakron.com (Email)
http://www.wrcakron.com (Web)

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

1. Overview

The 1782-JDCE is a family of DeviceNet-to-serial link communications gateways that provide a flexible DeviceNet interface to a wide variety of ASCII devices. The JDCE allows the user to easily and conveniently connect and integrate peripheral products with either RS232 or RS485 serial ports into a DeviceNet system.

The JDCE does not interpret the data being transmitted across it, and so the transferred messages may contain data of any nature or definition. This allows you to use the same device for a many different DeviceNet-serial interface applications.

Using the JDCE you may communicate with the connected peripheral devices in the same fashion as the other DeviceNet products in the system. Data may be read/written using either I/O polling or explicit messaging. Typically real-time data is read and written as I/O by the DeviceNet Master via Polled I/O and parameters are read and written with the Explicit Messaging technique. Revision 6 also allows you to read/write serial data via explicit messages.

The 1782-JDCE is defined as a Communications Adapter device on the DeviceNet system. It has a 3-pin plug connector for connection to a RS232 or RS485 interface port on your device and a 5-pin pluggable DeviceNet connector for connections to the DeviceNet network. The device does baud rate selection automatically when it is powered up on a network. The 1782-JDCE has one assigned DeviceNet address, which is set by a 6-position DIPswitch on the unit. Other JDCE parameters are software-configurable. Each 1782-JDCE has 2 standard green/red DeviceNet LED's for module status and network status and two green LED's to indicate RS485/232 transmit and receive activity.

The RS232 version may be used for point-to-point connection to a single serial device. The RS485 version may be connected in a point-to-point fashion to a single device, or to multiple devices in the standard RS485 convention.

Figure 1-1 1782-JDCE

1.1 Features

The 1782-JDCE has the following features:

- Translates messages and data between DeviceNet and a serial peripheral device
- ODVA Group 2 Only Slave
- ODVA Conformance tested to DeviceNet Spec 2.0
- Defined as a DeviceNet Communications Device Profile 12 (C_{hex})
- Autobaud operation
- I/O Messaging of Serial Data
  - Poll
  - COS
  - Cyclic
- Explicit Messaging
  - Serial data
  - Configuration data
- Pad mode option
- Byte swapping option
- Transaction header or data only mode option
- Master-Slave handshake option
- Software Configurable Parameters for serial port operation
- Address selection via DIP switches
- DIN rail mount
- Pluggable 5-pin DeviceNet connection
- Pluggable RS-485 2-pin connection / RS-232 3 Pin Connection
- 2 standard DeviceNet module and network status LED's
- 2 serial transmit and receive LED's
- Powered from DeviceNet 11-25 Vdc network power
- ASCII string length up to 50 bytes
- Serial port baud rate up to 38.4k baud
- Optional isolated RS485 or RS232 interface

1.2 Revision 6 Enhancements

This manual applies to 1782-JDCE version 6.01. The specific enhancements incorporated in Revision 6 of the product include:

- Explicit Message ASCII string send and receive
- Start string delimiter operation
- Transaction header mode or data only mode selection option
- Master-Slave handshake option

1.3 Typical Applications

- Weigh scales
- Bar code readers and scanners
- Display panels
- Robots
- Drives
- Operator stations / HMI
- Magnetic code readers

1.4 Basic Operation

The JDCE operates as the DeviceNet front-end to the serial device(s). The DeviceNet Master can receive and send data to and from the 1782-JDCE via the methods described in this section. It sends the data to the device and likewise accepts responses from the device, which are passed back to the DeviceNet system as required.

The JDCE has one DeviceNet address. All DeviceNet messages to the JDCE itself (to read / write its internal data) are sent to this address. DeviceNet messages to and from the serial device can be sent to the JDCE DeviceNet assembly objects using either I/O or Explicit Message commands.

The JDCE Parameter Object allows you to define the specific operation of each JDCE. These parameters include all the set-up required for the serial communications link.

The following chart and section defines the various messaging methods used for "typical" data types at your serial device and a brief explanation follows.

Table 1-1 I/O Message Types

Typical Data	Polled	Cyclic	Bit-Strobe	Change-of-State	Explicit Message
Commands	√	√		√	√
Status	√	√		√	√
Parameters					√

1.1.1.1 Polled I/O

The DeviceNet Master uses the JDCE's predefined polled IO connection to send serial input and output data to the JDCE. When a poll is received and the record has changed since the last poll was sent, the JDCE sends the associated transmit data out the serial port to the remote ASCII device. When the JDCE receives serial data from a device on the serial link, the poll response data to the Master contains up to 50 bytes of received data.

1.1.1.2 Cyclic Input Message

Cyclic I/O is the function by which a slave device sends its input data to the master at a specific time period without the host explicitly requesting it. When the specified time interval (defined by you) elapses, the most recent input data from the serial port are transmitted to the master. This data is the same format as a poll response.

1.1.1.3 Change-of-State, or C.O.S.

C.O.S. I/O is the function by which a slave device sends its input data to the master when defined input data changes without the host explicitly requesting it. In the case of the JDCE, this occurs when the delimiter character is asynchronously received from the serial device, when the defined number of characters is received or when the internal buffer is filled. This data is the same format as a poll response.

1.1.1.4 Explicit Messages

Explicit messages are typically used to read and write configuration data. This data allows the JDCE to change its internal operating parameters such as baudrate and parity.

In product revision 6 and higher, explicit messages can be used to read and write the serial port data as well.

1.5 Major Option Selections

The 1782-JDCE has several different operating modes. Some of these are available only in certain combinations. These option selections are described here briefly and in more detail later.

1.5.2 Header vs. Data Only Option

In **Data Only Mode**, only the actual ASCII string data is transmitted between the JDCE and the Master. This data may or may not include any selected delimiter characters. The data itself may or may not contain a 1-byte or 2- byte data length value, depending upon which data type (array, short_string, or string) is selected.

In **Header Mode**, there are 1 or 2 bytes of header information that precede the data string, as described above. The Header information includes a record number (transaction ID) and a status byte. (Default.)

1.5.3 Master-Slave Handshake vs. Immediate Option

If DeviceNet Master-Slave **Handshake Mode** is selected, the DeviceNet Master can inhibit the JDCE from sending new ASCII data until the Master is ready to receive and process the new data. This option is only available if the Header option is also selected, and it is used only with a Poll I/O or Explicit Message.

The JDCE will indicate to the Master that new data is available by setting the New Data Flag in the Status byte of the Produce message. When the Master is ready to receive new serial data, it sets a new number in the new record number byte of the next poll command message. Note that this applies only to data being sent from the JDCE to the Master.

In **Immediate Mode** this handshaking is not active and the JDCE sends new data as soon as it is received from the ASCII device. (Default.)

1.5.4 Pad vs. No Pad Option

If the **Pad** option is selected, the JDCE will always send a fixed number of data bytes to the Master. It is typically used when a Terminating Character trigger is used to stop receiving ASCII data. This is useful if the ASCII device(s) connected to the JDCE have varying data lengths and the Master cannot accept varying length I/O messages. The JDCE will fill a short message with the pre-defined Pad Character. The data length is defined by the Max Receive Length parameter.

If the **No Pad** option is selected, the JDCE will return the actual number of characters defined, plus the Header data if selected. (Default.)

1.6 Product version and EDS

This manual applies to 1782-JDCE-x version 6.01 and higher. An EDS (Electronic Data Sheet) for the 1782-JDCE, is shipped with your device or is available on WRC's web site: http://www.wrcakron.com/.

1.7 Ordering Information

Table 1-2 Part Numbers

Serial Interface	Model Number
RS232C	1782-JDCE-1
RS485	1782-JDCE-2
Isolated RS232C	1782-JDCE-3
Isolated RS485	1782-JDCE-4

2. Quick Start

To quickly install your 1782-JDCE in your DeviceNet system, follow the instructions below. For more details, see Section 4.

2.1 How to Install and Establish DeviceNet Communications

1. Connect your DeviceNet network cable to a 5-pin female (Phoenix-type) plug according to DeviceNet cable wiring specifications
2. Make sure that the DeviceNet network is properly terminated.
3. The JDCE Node Address (MacID) is set to 63 at the factory. Make sure no other device on the network is set to 63, or change the JDCE address to one that is not currently used (see below).
4. The JDCE baud rate is set to Autobaud operation at the factory. No baud rate setting is required.
5. Make sure that there is power on the DeviceNet network and plug the cable into the 1782-JDCE.
6. The 1782-JDCE will undergo its initialization sequence, flashing both LED's red and green. After approximately 5 seconds, the Module Status LED (labeled "MS") will flash green. The Network Status LED (labeled "NS") will remain off. This condition occurs while the JDCE is attempting to synchronize to the network baudrate.
7. The Module Status LED ("MS") will go on solid after the Device successfully determines the network baudrate. This requires devices on the network attempting to communicate with each other. The Network Status LED (labeled "NS") will begin to flash green. If it turns solid red, check for a duplicate MacID on the network. It will remain off until the JDCE receives a valid DeviceNet message from which it will set its baud rate.
8. Once the Master recognizes the unit on the link and allocates the connection (initiates communications), The Network Status LED will be solid green. The device is now being actively scanned.
9. The 1782-JDCE is now operating on the network.

2.2 Default Settings

The following list shows the default set-up for all the parameters.

Serial Port	**Default Operation**
Serial Character Framing	7N2
Serial Port Comm Speed	9600 baud
Serial Port Receive from ASCII Device	**Default Operation**
Max Number of Receive Chars	20
Receive Record Start Mode	Not enabled
Receive Start Delimiter	Colon
Receive Record End Mode	Enabled and include with data string
Receive End Delimiter	Carriage return
Gateway Send (Produce) on DeviceNet to Master	**Default Operation**
Receive String Data Type	Short_string: data with preceding 1 byte length
Pad Mode	Off
Pad Character	null character (0)
Receive Swap Mode	Off
DeviceNet Handshake Mode	Off
Gateway Produce Assembly Size	23 bytes
Serial Data	--
Actual Received Data Size	0
Receive Record Number	0
Serial Port Transmit to ASCII Device	**Default Operation**
Max Number of Transmit Chars	20
Transmit End Delimiter Mode	Include delimiter with
Transmit End Delimiter Character	Include
Gateway Receive (Consume) on DeviceNet from Master	Carriage return
Transmit String Data Type	Short_String: data with 1 preceding length byte
Transmit Swap Mode	Off
Record Header Mode	Active; record header bytes precede data string
Gateway Consume Assembly Size	22 bytes
Explicit Messages from EDS Editor	**Default Operation**
Actual Serial Data String to Send to ASCII Device	--
Transmit Serial Data Size	0
Transmit Record Number	0

Status	0; no errors
DeviceNet Set-up	**Default Operation**
DeviceNet Baud Rate	Autobaud

2.3 How to Install a Serial Network

1. The communication between your serial device(s) and the 1782-JDCE is an RS232 3-wire or RS-485 2-wire differential network. Connect an appropriate cable to your device. (In an RS485 network, make sure at least one point on the link is grounded.)
2. Connect the other end of the cable to the JDCE using the 3-point terminal plug provided. Note the terminal markings on the JDCE case. See Figure 4-1.
3. Turn on power to the serial device and the JDCE.
4. Set up the ASCII buffer sizes on the JDCE. (The defaults are 20 and 20). If more than 20 bytes are required for the transmit or receive buffers, set the appropriate parameters in your configuration file to the buffer size you need for your ASCII data.

   NOTE: This will modify the IO message size. You will need to reconfigure the poll / C.O.S. / cyclic transmit and receive data sizes if you modify the ASCII buffer size from the default value. In many configuration tools, this will unmap the data in your scanner's scan table. They must be remapped in order to be able to process the data in your PLC or PC software. These values are displayed in the Parameter Object, Class 15 (F_{hex}).

2.4 How to Read Serial Device Input Data from the JDCE

1. Connect to the JDCE from your configuration tool.
2. Connect the serial side of the JDCE to your computer's serial port or another serial device.
3. Go to the device configuration screen in the configuration tool.
4. Make sure that the JDCE is in the default factory configuration.
5. Set the baudrate and framing format of the serial port to the baudrate and framing format of the serial device that you are using.
6. Put the Configuration tool in to monitor mode.
7. Direct the device that you are communicating with to send data. For example, if you are connected to a computer terminal program, type a message into the terminal. When you hit enter, the module will update the data with the message that you typed, and increment the record number.
8. Poll messages work in the same manner as the parameter object interface. The default assembly format of the poll message is shown below.

Table 2-1 Default Receive Data Assembly Format

Byte 1	Byte 2	Byte 3	Byte 4-X (X ≤ 22)	Byte X+1

				(max = 23)
Transaction ID Byte	Status Byte	Length Byte	ASCII Data (following the length byte)	<CR> (Terminator)

2.5 How to Write Serial Output Data to the JDCE

1. Do steps 1-6 of Section 2.4 above.
2. Enter the serial data that you wish to send in the transmit data parameter.
3. Change the Length of the data in the length byte to reflect the length you wish to send.
4. Change the Transmit record number.
5. The JDCE will generate the characters that you typed in on the computer screen.
6. Poll message works in the same manner as the parameter object interface. The assembly formats of these messages are configurable and are covered in the next chapter.

Table 2-2 Default Transmit Data Assembly Format

Byte 1	Byte 2	Bytes 3-X (X≤21)	Byte X+1 (max ≤ 22)
Transaction ID Byte	Length Byte	ASCII Data	<CR> (Terminator)

3. General Specifications

Product:	1782-JDCE Device-Serial Gateway
Description:	Communications gateway between a serial capable device over an RS232 or RS485 interface and a DeviceNet network.
Device Type:	Communications Adapter, C_{hex} (12)
Device Profile:	Identity Object Message Router Object DeviceNet Object Connection Object Parameter Object Serial I/O Object (vendor-specific) Transmit Serial Object (vendor-specific) Receive Serial Object (vendor-specific)
Product Revision:	6.01
DeviceNet Conformance:	Designed to conform to the ODVA DeviceNet Specification Volume I and II, Version 2.0.
DeviceNet Communications:	Predefined Master/Slave Connection Set, Group 2 Only Server
DeviceNet:	**Baud rate selection:** Autobaud operation (default) Fixed baud (software selectable) – 125k, 250k and 500k baud **Address selection:** Address number 0 to 63, switch selectable (default = 63) **Cable Connection**: JDCE: 5-pin pluggable header (male) Phoenix Contact MSTBA 2.5/5-G-5.08/AU or equivalent DeviceNet Cable: 5-contact plug (female contacts) Phoenix Contact MSTB 2.5/5-ST-5.08/AU or equivalent (included) **Status Indicators:** Module Status: green/red bi-color LED Network Status: green/red bi-color LED
Serial port:	**Baud rate**: 1200, 2400, 4800, 9600, 19.2k, 38.4k baud (software selectable) **Parity**: Odd/even/none (software selectable) **Data bits**: 7 or 8 (software selectable) **Cable Connection:** JDCE: 3-pin pluggable header (male) Phoenix Contact MSTBA 2.5/3-G-5.08/AU or equivalent

	Serial Cable: 3-contact plug (female contacts) Phoenix Contact MSTB 2.5/3-ST-5.08/AU or equivalent (included) **Status Indicators**: Transmit Active: green LED Receive Active: green LED
Network Isolation:	2500V (optional, models JDCE-3 or JDCE-4 only)
Max Power:	1.75 watts: 160 mA @ 11 Vdc – 70 mA @ 25 Vdc unregulated power supply
Mounting:	DIN rail mount, EN 50022
Size:	• Depth: 3.54" (90 mm) • Width: 0.98" (25 mm) • Height: 3.11" (79 mm)
Operating Temp:	0-70 ºC
Humidity:	0-95% RH, non-condensing

4. Hardware Installation and Set-Up

4.1 Overview

The JDCE is mounted on an EN50022 DIN rail.

The JDCE contains two LED's to indicate the status of the device and the status of the network. The device can be connected to the main DeviceNet trunk line or to a drop line via a 5-pin female plug-style connector. It also has two green LED's to indicate the presence of activity on the RS-232/485 transmit and receive lines.

All power for the JDCE is derived from the DeviceNet power.

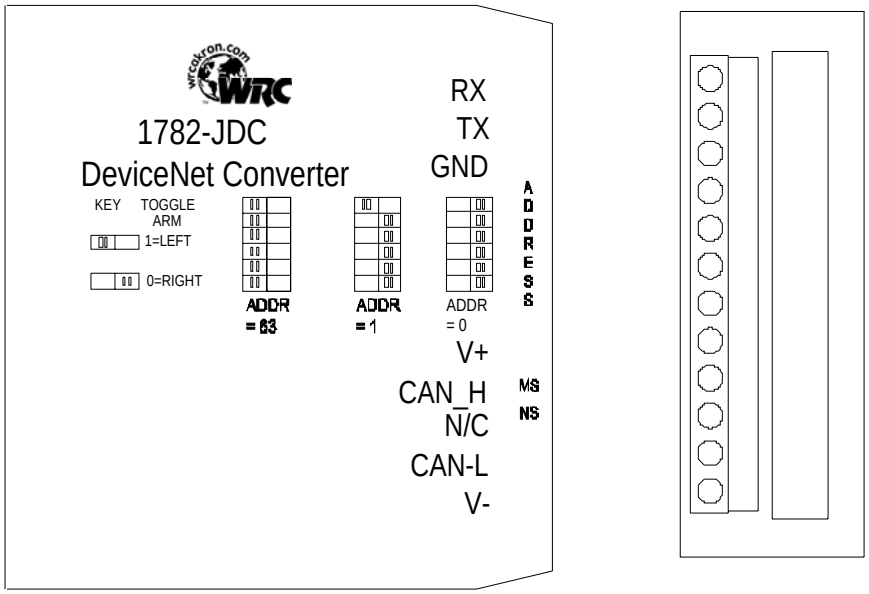

Figure 4-1 1782-JDCE Outline Drawing

4.2 LED Operation

4.2.1 DeviceNet LEDs

The JDCE has two LEDs that provide visual status information to the user about the product and the DeviceNet network. See Tables 5-1 and 5-2 that follow below for how to interpret LED status indications.

Table 4-1 Module Status LED (labeled MS)

LED State	Module Status	Meaning
OFF	No Power	There is no power through DeviceNet.
Green	Device Operational	JDCE is operating normally.
Flashing Green	Device in Standby	JDCE needs commissioning (e.g. attempting autobaud).
Flashing Red	Minor Fault	Recoverable fault.
Red	Unrecoverable Fault	JDCE may need replaced.
Flashing Red/Green	Device Self-Testing	JDCE is in self-test mode.

Table 4-2 Network Status LED (labeled NS)

LED State	Network Status	Meaning
OFF	No Power / Not on-line	JDCE has no power or has not completed the Dup_MAC_ID test.
Flashing Green	On-line, not connected	JDCE is on-line but is not allocated to a Master.
Green	On-line	JDCE is operating normally.
Flashing Red	Connection time-out	One or more I/O connections are timed out.
Red	Critical link failure	JDCE has detected an error that makes it incapable of communicating on the link. (Bus off or Duplicate MAC ID).

4.2.2 Serial Port LEDs

The JDCE also has two (2) RS-232/485 activity LEDs: one for transmit (TX) or Signal + (SG+); and one for receive (RX) or Signal– (SG-). These LEDs are electrically tied to the serial data lines and will illuminate when there is data signals active on the respective data lines and the JDCE has power

4.3 Serial Port Connector

The ASCII devices are connected to the JDCE via a 3-wire communications cable. See your

ASCII device's User Manual for details on the proper connections.

The RX and TX designators are referenced with respect to the JDCE.

Table 4-3 RS232/RS485 Connector Signals

Pin #	Designator (RS232/RS485)	RS232 Signal	RS485 Signal
3	RX/SG-	Receive	Signal -
2	TX/SG+	Transmit	Signal +
1	GND	Ground	Shield

Note: The RS232 max distance spec is 50 feet (15m).

Note: The RS485 max distance sped is 4000 feet (1219m).

4.4 DeviceNet Configuration

DeviceNet specifications provide for a maximum network distances for the main trunk line and drop lines, depending upon the baud rate used on the network. See Table 4-4

Table 4-4 Maximum Network Cable Lengths

	Trunk Line Length		Drop Length			
	Maximum Distance		Maximum		Cumulative	
Baud Rate	Meters	Feet	Meters	Feet	Meters	Feet
125k baud	500 m	1640 ft	6 m	20 ft	156 m	512 ft.
250k baud	250 m	820 ft	6 m	20 ft	78 m	256 ft.
500k baud	100 m	328 ft	6 m	20 ft	39 m	128 ft.

4.4.1 Network Termination

A DeviceNet system **must be terminated at each end of the trunk line**. The host controller and the **last** JDCE or other DeviceNet device on the network must always be terminated to eliminate reflections, even if only two nodes are present. The DeviceNet specifications for the terminating resistor are:

- 121 ohm
- 1% metal film
- 1/4 Watt

An appropriate terminating resistor, WRC part number RM121DN, may be purchased from WRC.

IMPORTANT: **Per the DeviceNet spec -- do not terminate devices on drop lines.**

NOTE: If you feel you are having DeviceNet communications errors, check your network terminations.

With power removed from the network, measure the dc resistance with an ohmmeter. It should measure 60Ω. If it measures 121Ω, add another 121Ω terminating resistor. If it measures ~40Ω, then you have 3 terminators and must remove one.

4.4.2 DeviceNet Connection Wiring

The JDCE uses a 5-pin plug-style DeviceNet connector, which has male pins.

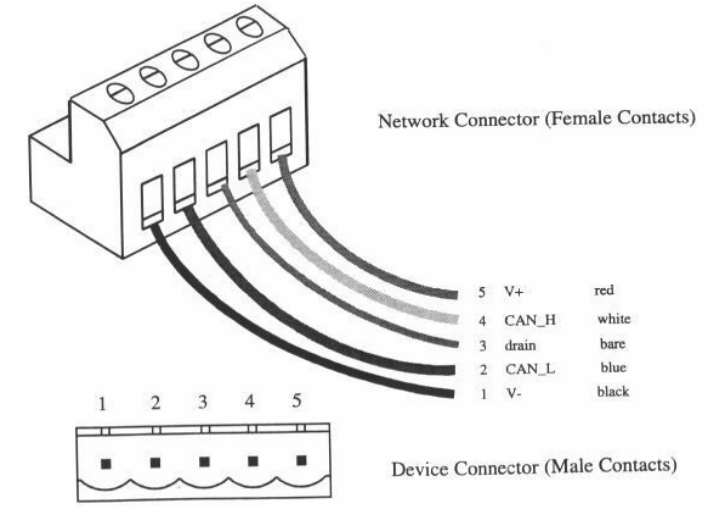

Figure 4-2 DeviceNet cable connector

5. SOFTWARE Configuration and Set-Up

The 1782-JDCE-x is an easy device to set up and configure. Using features like the EDS sheets for configuration can expedite the process if you use a network configuration tool that supports them. They provide a graphical interface to the device's parameters and allow the addition of helpful text descriptions in setting up your device. The current EDS file is available on our website, www.wrcakron.com. If your configuration tool does not support the EDS device profiles, the set up of a DeviceNet device requires a little more understanding of DeviceNet and its operation. This section is designed to fully describe the features of the 1782-JDCE and to help you set them up. If you have problems setting up this or any other WRC device, we are available to help you. We can be reached via phone at (330) 733-6662, or via the Internet at support@wrcakron.com.

The Parameters are defined in this section.

5.1 Device Parameters

The JDCE is configured using the Parameter Object, Class 15 ($0F_{hex}$) as defined in Table 5-1.

Table 5-1 Configuration Parameter List

Parameter	Param. Instance	Access	Description	Parameter Choices		Default Setting	Default Value	Data Type
Serial Port Parameters								
Serial Character Framing	1	Get/Set	Defines the number of data bits, stop bits and parity in data character frames	0 = 7N2 1 = 7E1 2 = 7O1 3 = 8N1 4 = 8N2	5 = 8E1 6 = 8O1 7 = 7E2 8 = 7O2	7N2	0	USINT
Serial Port Comm Speed	2	Get/Set	Defines the baud rate of the serial port	0 =9600 1 = 1200 2 = 2400	3 = 4800 4 = 19.2k 5 = 38.4k	9600 baud	0	USINT
Serial Port Receive from ASCII Device								
Max Number of Receive Chars	3	Get/Set	Maximum number of characters the 1782-JDCE expects to receive into its ASCII port from the serial device	1 – 50		20 chars	20	USINT
Receive Record Start Mode	4	Get/Set	Selects whether or not the start delimiter is included with the received data	0 = No Start Delimiter 1 = Exclude Start Delimiter 2 = Include Start Delimiter		No Start Delimiter	0	USINT
Receive Start Delimiter	5	Get/Set	Character which identifies the beginning of the data string from the ASCII device when the length is specified as 0	Any valid standard ASCII character (0 – 127, 0-255)		Colon	0x3A	USINT
Receive Record End Mode	6	Get/Set	Selects whether or not the End delimiter is included with the received data	0 = No End Delimiter 1 = Exclude End Delimiter 2 = Include End Delimiter		Include End Delimiter	2	USINT
Receive End Delimiter	7	Get/Set	Character which identifies the end of the data string from the ASCII device when the length is specified as 0	Any valid standard ASCII character (0 – 127, 0-255)		Carriage return	D_{hex}	USINT
Gateway Send (Produce) on DeviceNet to Master								
Receive String Data Type	8	Get/Set	Defines the format of the data string sent to the Master	0 = Array 1 = Short_String 2 = String		Short_String	1	USINT
Pad Mode	9	Get/Set	Indicates whether to pad the invalid data region after the delimiter with the pad character, or to use variable length ASCII responses	0 = Pad Mode Disabled 1= Pad Mode Enabled		Disabled	0	USINT
Pad Character	10	Get/Set	The value to use to pad the invalid data portion	Any valid standard ASCII character (0 – 127 with 7-bit data,		NULL	0	USINT

			of the poll response	0-255 with 8-bit data)			
Receive Swap Mode	11	Get/Set	If enabled, the position of the bytes in the serial messages will be swapped every 2, 3 or 4 bytes.	0 = Disabled 1 = 16-bit Swap Enabled 2 = 24-bit Swap Enabled 3 = 32-bit Swap Enabled	Disabled	0	USINT
DeviceNet Handshake Mode	12	Get/Set	If enabled, Master must acknowledge it is ready for next new data before JDCE sends the new data.	0 = Master/Slave Handshake 1 = No handshaking.	No handshaking	1	USINT
Gateway Produce Assembly Size	13	Get	Total number of bytes of I/O data that are sent to the Master from the JDCE. This should be the **RX size** of your **Scanner**.	0-54	20 bytes of array data and 3 header bytes	23	USINT
Serial Data	14	Get	Serial data in the receive buffer	Any data string	Empty		SHORT_STRING
Received Data Size	15	Get	Calculated length of the DeviceNet message to send to the ASCII device from the Master	0-54	0	0	USINT
Receive Record Number	16	Get/Set	The Receive Record Number sent from the master	0-255	0	0	USINT
Serial Port Transmit to ASCII Device							
Max Number of Transmit Chars	17	Get/Set	Maximum number of characters the 1782-JDCE expects to transmit out its serial port to the serial device	1-50	20 chars	14_{hex}	USINT
Transmit End Delimiter Mode	18	Get/Set	Selects whether or not the End delimiter is included with the received data	0 = No End Delimiter 1 = Exclude End Delimiter 2 = Include End Delimiter	Include	2	USINT
Transmit End Delimiter Character	19	Get/Set	Character which identifies the end of the transmit data string from DeviceNet to the ASCII device when the length is specified as 0	Any valid standard ASCII character (0 – 127 with 7-bit data, 0-255 with 8-bit data)	Carriage return	D_{hex}	USINT
Gateway Receive (Consume) on DeviceNet from Master							
Transmit String Data Type	20	Get/Set	Defines the format of the data string received from the Master	0 = Array 1 = Short_String 2 = String	Short_String	1	USINT
Transmit Swap Mode	21	Get/Set	If enabled, the position of the bytes in the serial messages will be	0 = Disabled 1 = 16-bit Swap Enabled 2 = 24-bit Swap Enabled	Disabled	0	USINT

			swapped every 2 or 4 bytes.	3 = 32-bit Swap Enabled			
Record Header Mode	22	Get/Set	Selects whether or not the header information is included in the DeviceNet data string	0 = Include record header 1 = Omit record header; send serial data only	Include	0	USINT
Gateway Consume Assembly Size	23	Get	Total number of byte of I/O data that are received from the Master. This should be the **TX size** of your **Scanner**.	0-54	20 bytes of array data and 2 header bytes	22	USINT
Serial Port Transmit / Explicit Messages from EDS Editor							
Transmit Serial Data String	24	Get/Set	Serial data to be sent to the serial transmit buffer	ASCII Block Data	Empty		SHORT_STRING
Transmitted Serial Data Length	25	Get/Set	Length Of the Transmit Serial Data	0-50	0	0	USINT
Transmit Record Number	26	Get/Set	The record number of the current transmit data buffer	0-255	0	0	USINT
Status	27	Get	The Combined status byte for the Serial Port Object, The Receive Record object and the Transmit Record object.	1 – TX FIFO Overflow 2 – Rx FIFO Overflow 4 – Rx Parity Error 64 – Handshake Error 128 – New Data Flag	No Status	0	USINT
DeviceNet Set-up							
DeviceNet Baud Rate	28	Get/Set	DeviceNet Baud Rate	0- 125 Kilobaud 1- 250 Kilobaud 2- 500 Kilobaud 3- Automatic Detection	Automatic Detection	3	USINT

5.2 Setting Up the Serial Link

Parameters 1 and 2 allow you to define the serial link communications options.

Table 5-2 Serial Framing Formats

Value	Baud Rate
0	7N2
1	7E1
2	7E2
3	7O1
4	7O2
5	8N1
6	8N2
7	8E1
8	8O1

Table 5-3 Serial Baud Rates

Value	Baud Rate
0	9600
1	1200
2	2400
3	4800
4	19200
5	38400

5.3 Receiving Serial Data from the ASCII Device

5.3.1 Overview

The JDCE receives a number of characters and transmits these to the DeviceNet Master via

- I/O – poll, COS, Cyclic
- Explicit Message

The received character string is captured when

- the specific number of bytes defined (Receive Character Buffer Length) is received, or
- the defined End-of-String Terminator character is detected.

When either of these events occur the JDCE stores the received message string into its internal buffer and will then transmit (Produce) it onto DeviceNet at the next appropriate opportunity.

5.3.2 Setting up the Receive Character Buffer Length

The receive character buffer length is the number of characters that the JDCE can receive from your I/O device into its buffer at one time. The length of the data string sent to the DeviceNet Master is less than or equal to this size, plus any header options selected (to be discussed later).

If the JDCE receives more characters that this number, it will internally generate an overflow and

force the data into the JDCE DeviceNet transmit buffer to be sent to the Master. The subsequent received characters will then be received into the buffer and handled as the start of next incoming message string. The overflow bit in the status byte will be set as well.

Caution: Incoming characters could be missed in the process of handling a string longer than the defined max length.

This value can be set and retrieved by using the standard set and get services on class 15 (F_{hex}), instance 3, attribute 1.

5.3.3 Setting up and Using Pad Mode

Pad Mode operation is the method used by the JDCE adds extra characters to the end of its received data string (after the delimiter character) from the external I/O device before sending the string to the DeviceNet scanner (Master) as an I/O Response. The quantity added is such that the data string returned to the scanner is always a constant length, and that length is the number specified in the receive_character_length parameter plus any header options. The quantity of pad characters sent can vary from message to message, depending upon the size of the incoming string.

- **Pad Mode Selection**

Pad mode is included with our device for compatibility with Scanners that cannot **receive** variable length I/O messages. (Notable examples include many Allen-Bradley's scanners at the time of printing). For such scanners, you must turn ON Pad mode (a value of 1). Turning Pad mode ON will not harm Scanners that do support variable length receive messages. The default value for Pad Mode is OFF. If your scanner does support variable I/O messaging lengths, you leave the Pad Mode option OFF (a value of 0) to conserve some network bandwidth.

The selection of Pad Mode is valid only for the DeviceNet message that the JDCE produces. It has no effect on DeviceNet messages sent from the Scanner to the JDCE. This value can be set and retrieved by using the standard set and get services on class 15 (F_{hex}), instance 9, attribute 1.

- **Pad Mode Character**

The JDCE allows you to specify the character that pad mode uses to pad the received serial data. This can be set to any valid I/O value (0-127 in 7 bit modes, 0-255 in 8 bit modes). This value can be set and retrieved by using the standard set and get services on class 15 (F_{hex}), instance 10, attribute 1.

5.3.4 Setting Up and Using the Swap Bytes Mode

This option may be helpful if the JDCE is connected to a DeviceNet scanner that organizes the data string characters into data type elements that are larger than 1 byte each. An example is many Allen Bradley PLCs, such as the SLC500. In such cases the bytes of the data in the Master's memory organization can be reversed from the order in which they are sent or received on the DeviceNet and the serial link to the ASCII device. This may cause problems in some cases.

Thus, the message received or desired "ABCDEFGH" string may appear in memory as "BADCFEHG" for 2-byte word organization, and "DCBAHGFE" for 4-byte word organization.

- **Transmit Byte Swapping**

By setting Parameter 21 (class F_{hex}, instance 21, attribute 1), the bytes from the Master will be swapped by the JDCE before transmitting the string to the ASCII device.

- **Receive Byte Swapping**

By setting Parameter 11 (class F_{hex}, instance 11, attribute 1), the JDCE will re-order the bytes received from your ASCII device before sending the string to the Master.

- **Rules for Usage**

1. This feature is set for both transmit and receive independently.
2. The Byte Swapping works better if the string length is an even multiple of the byte-swap size.
3. If a delimiter is received, then

- All characters up to and including the defined delimiter are sent to the DeviceNet Master. If Pad Mode = 1, then the JDCE will fill the Poll Response data with the Pad Char up to the defined size. If Pad Mode = 0, then the JDCE will send only the data up to and including the delimiter

4. If no delimiter is received, then

- The JDCE will receive up to Max_Number_of_Receive_Chars, and then send this string to DeviceNet with an overflow error.
- If will continue to receive and send strings of size Max_Number_of_Receive_Chars, along with the overflow error, until a delimiter is received. This could continue indefinitely if your I/O device does not transmit the specified delimiter.

5.3.5 Setting Up and Using Delimiter Operation

When receiving data strings from your serial device, the JDCE can take advantage of both Start and Stop (End) delimiters. The **Start Delimiter is the start-of-string indicator** and the **End Delimiter is the end-of-string indicator.** This allows you further control over exactly which characters are sent to the Master.

When you select **Start Delimiter** operation, you define a character that prompts the JDCE to start storing the incoming data string. All characters up to this Start Delimiter (after the previous message was completed) are ignored. Once the Start Delimiter is received, all characters are stored until either the **End Delimiter** is received or the Max Receive Char Length is reached. Once the End Delimiter is reached, the data string is captured and prepared to send to the DeviceNet Master.

If either delimiter is used, you also can elect whether or not to include those characters in the string

sent to the Master.

5.3.6 Receive String Data Type

This is the format of the data – array, short_string or string – you will send to the DeviceNet Master. These are shown below: Which one you pick depends on your application, and will modify the format of the data field.

This is the format of the data you will send to the JDCE – array, short_string or string. These are shown below: Which one you pick depends on your application, and will modify the format of the data field.

The **Array** data type does not have a length associated with it. It is equivalent to specifying a length of zero using a string or short string data type.

The **Short_String** data type is the default data type of the device. This will suffice for most applications. The Short_String data type has only one byte of length, and the rest of the data bytes are appended after the length.

The **String** data type has two bytes of length. The String data type is useful in talking to some PLC's or other devices that have a data file specifically made to handle this data type. The length is little endian (low byte, high byte), and the high order byte should always be set to zero. The JDCE will only receive up to 50 bytes of information, so the extra byte, although required for this data type, is always 0.

Note: If the Short_String or String data type is used, the length is **not** sent to the ASCII device.

Data Byte 1	Data Byte 2	. . . . .	Data Byte N

Figure 5-1 Receive Array Data Format

Length Byte	Data Byte 1	Data Byte 2	. . . . .	Data Byte N

Figure 5-2 Receive Short_String Data Format

Len. Byte 0	Len. Byte 1	Data Byte 1	Data Byte 2	. . . . .	Data Byte N

Figure 5-3 Receive String Data Format

5.3.7 Setting Up the Scanner I/O Receive Size

The JDCE automatically calculates the number of bytes it will send the DeviceNet Master. Its value is determined by a combination of the incoming data and the options you have selected. Parameter 15 defines the size of the DeviceNet message to be sent by the JDCE to the Master.

Important: If you are using a Scanner that must receive a constant message length, you must set its Rx (receive) value to this number of bytes.

5.3.8 Explicit Messages to Receive the Serial Data String

Parameters 14, 15 and 16 contain the status of the most recent incoming serial data string. You can

use the data to read your device's ASCII data via the Explicit Messaging technique.

Parameter 14 holds the most recent received data.

As explained in Section 5.3.6, Parameter 15 defines the size, in bytes, of the DeviceNet message to be sent by the JDCE to the Master.

Parameter 16 holds the record number of the data string in Parameter 14 if the Header option is selected.

5.3.9 Status Byte Description

The Serial Status byte is an OR'd bitfield of a number of status and exceptions. Bits 0-2 are defined.

Table 5-4 Serial Status Byte

Bit	Exception
0	TX I/O Overflow
1	Rx I/O Overflow
2	Rx Parity Error
6	Handshake Error
7	New Data Flag

- TX I/O Overflow

The transmit queue has overflowed resulting in a loss of data. The transmit I/O is full of data waiting to be transmitted. Some of the data added has been lost. When space becomes available in the TX I/O, this bit will be reset.

- Rx I/O Overflow

The receive queue has overflowed resulting in a loss of data. The receive I/O is full of data waiting to be processed. The data has been lost. When space becomes available in the RX I/O, this bit will be reset.

- Handshake Error

This error will occur only in Master-Slave Handshake Mode. It indicates that the Master has requested a new data record from the JDCE, but the JDCE has not indicated new data is available to be sent.

- New Data

This bit is used only when the Master-Slave Handshake option is active. When the JDCE receives a new data string into its serial port, it sets this flag in its DeviceNet response message. The bit will remain set for 2 produce messages after the Master requests the new data. It then will be reset.

5.4 Transmitting Serial Data to the ASCII Device

5.4.1 Overview

The JDCE transmits a number of characters from the DeviceNet Master to your serial device via

- Poll I/O
- Explicit Message

The received character string is transmitted when

- the specific number of bytes defined (Transmit Character Buffer Length) is received, or
- the defined End-of-String Terminator character is detected.

When either of these events occurs the JDCE stores the DeviceNet string data into its internal buffer and will then transmit it out its serial port.

In order to transmit data to your serial device, the data must first be sent to the JDCE and then the JDCE must send the data to the serial device. The options for transmitting from the Master to the JDCE are discussed first.

5.4.2 Setting up the Transmit Character Buffer Length

The Transmit character buffer length is the number of characters that the JDCE can receive in its transmit buffer from the DeviceNet system. This size contributes to the I/O's Consume Size. This size can be found in the Parameter object.

5.4.3 Setting Up and Using the Transmit Delimiter

The transmit delimiter is an **end-of-string** character which is used by the JDCE to determine how many bytes to transmit over the serial link to your JDCE device. This Transmit Delimiter will be used if the Transmit Buffer Length equals 0. If the buffer length is not 0 the JDCE will ignore the transmit delimiter.

The JDCE will transmit up to and including the delimiter when the above condition is met. The transmit delimiter can be set to any valid I/O character that can be received over the link. Be very careful not to set the delimiter to a value outside of the valid range for your data bits (Note: A data bit size setting of 7 will only allow you a delimiter range of 0-127 dec., 00-7Fhex). If you do not have a valid delimiter, or the delimiter is never received, the device will only update the output buffer on detection of an overflow condition. These values can be set and retrieved by using the standard set and get services on class 15 (F_{hex}), instance 4, attribute 1.

5.4.4 Setting up and Using the TX Byte Swap Mode

This option may be helpful if the JDCE is connected to a DeviceNet scanner that organizes the data string characters into data type elements that are larger than 1 byte each. An example is many Allen Bradley PLCs, such as the SLC500. In such cases the bytes of the data in the Master's memory organization can be reversed from the order in which they are sent or received on the DeviceNet and the serial link to the ASCII device. This may cause problems in some cases.

See Section 5.3.4 above, Setting Up and Using the Swap Bytes Mode.

5.4.5 Transmitting from the Master to the JDCE

You can transmit data to the JDCE from the Master using 2 methods – Poll I/O or Explicit Messages. Both require understanding and setting up some parameters.

5.4.6 Transmit String Data Type

This is the format of the data you will send from the JDCE – **array**, **short_string** or **string** – to the DeviceNet Master. These are shown below: Which one you pick depends on your application, and **will modify the format of the data field**.

See Section 5.3.6 above, Receive String Data Type for details.

5.4.7 Record Header vs. Data Only Mode

This defines whether the JDCE and Master exchange "header bytes" in the DeviceNet transactions or whether only data is transferred.

Record Header Mode is used primarily in Poll I/O to prevent the JDCE from repeatedly sending its data buffer to the serial device each time it receives a poll command. In this mode of operation the Master will update the Record Number whenever it wants the JDCE to send another serial string to the ASCII device. The JDCE interrogates the record number sent from the Master and will not transmit the serial data until a new record number is received. The Record Numbers do not have to be in any special order from the JDCE perspective. They can be chosen in any fashion selected by the Master.

In Data Only Mode the JDCE transmits out its serial port whenever it receives data into its internal buffer, whether or not the data has been updated. It does not use the header system to initiate serial transmissions. This technique is useful when the target ASCII device is not affected by receiving the same message multiple times and when the Master wants to eliminate the Message Header overhead. It may be most useful with Explicit Messages transmissions to the JDCE.

The Poll I/O and Explicit Message operate slightly differently, but follow the same rules for the options and can be used to perform the same functions.

A transmission of the serial data from the JDCE to your ASCII device can be initiated in 2 different ways. (Remember that if a poll connection is set up, serial data in the mapped I/O location will be sent to the JDCE on every poll command.)

- In Header Mode, changing the record number will always initiate a transmission on the network. Even if you do not set new data into the data string, the old data will be transmitted. See Table 2-2 .
- In Data Only Mode, the JDCE will transmit data out its serial port every time it receives a poll command or Explicit Message to its transmit buffer.

  CAUTION: THIS IS NOT A GOOD IDEA IF YOU ARE OPERATING OVER A JDCE MESSAGE. AN I/O MESSAGE IS REPETITIVLY SENT OVER AND OVER AGAIN, AND GENERATES INTERNAL WRITES TO ATTRIBUTE 2. THIS WILL CAUSE YOU TO SEND LARGE AMOUNTS OF DATA TO YOUR DEVICE AND CAN CAUSE TX I/O OVERFLOWS. THIS ALMOST GARUNTEES CORRUPTED DATA AND IMPROPER INFORMATION SENT TO YOUR SERIAL DEVICE.

5.4.8 Transmitting Serial Data

The length of the string set determines the use of a delimiter in transmitting data to a serial device from the JDCE.

If the string length is zero, or the data type is type Array:

The JDCE receives data sent from the DeviceNet Master and **uses the delimiter** to determine how much data is sent to the serial device. The JDCE will compute the length and then store this as the new length in the string attribute. (This will not show up if the data type is array, you will just see the string truncated, and the length will be in the background)

If a delimiter is contained within the string, then

- all characters up to and including the defined delimiter are stored.

If no delimiter is contained within the string, then

- the JDCE will store all the data received.

If the string length > 0 or the data type is String or Short_String,

The JDCE receives data sent from the DeviceNet Master ignoring any embedded terminator. It will store the number of characters defined in Max_Number_of_Transmit_Chars, or the total sent by the Master, whichever is less.

Now, the JDCE will send the data immediately if attribute X is set to 1. You can always cause this data to be transmitted by incrementing the record counter.

1.5.5 Consume Assemblies

The consume assemblies follow the formats below, shown for both Immediate Mode and Handshaking Mode.

Table 5-5 Consume Assembly, Array Type, No Handshaking

Byte 1	Bytes 2-X (X ≤ 50)	Byte X+1 (max = 51)
Transaction ID Byte	ASCII Data	<CR> (Terminator)

Table 5-6 Consume Assembly, Short_String Type, No Handshaking

Byte 1	Byte 2	Bytes 3-X (X ≤ 51)	Byte X+1 (max = 52)
Transaction ID Byte	Length	ASCII Data	<CR> (Terminator)

Table 5-7 Consume Assembly, String Type, No Handshaking

Byte 1	Byte 2	Byte 3	Bytes 4-X (X ≤ 52)	Byte X+1 (max = 53)
Transaction ID Byte	Length (MSB)	Length (LSB)	ASCII Data	<CR> (Terminator)

5.4.9 Setting Up the Scanner I/O Transmit Size

The JDCE automatically calculates the number of bytes it will receive from the DeviceNet Master. Its value is determined by a combination of the incoming data and the options you have selected. Parameter 23 defines the size of the DeviceNet message to be sent to the JDCE from the Master and should be set as the Tx size in your Scanner's I/O set-up.

Important: You must set your scanner's Tx (transmit) value to this number of bytes.

5.4.10 Master-Slave Handshake vs. Immediate Mode

If DeviceNet Master-Slave **Handshake Mode** is selected, the DeviceNet Master can inhibit the JDCE from sending new ASCII data until the Master is ready to receive and process the new data. This option is available **only if the Header option is also selected**, and it is used only with a Poll I/O or Explicit Message.

In this mode there two data required for the complete transaction:

- A "New Data Available" Flag is set by the JDCE in the status byte. This informs the Master that the JDCE has received a new data string and is waiting for the OK to send it. See Table 5-4.
- An additional "Ready for New Data" byte is pre-pended to the message the Master sends to the JDCE. (The JDCE's Consume Object.) This New Data byte is used to indicate to the JDCE that the Master is ready to receive the new data. (The JDCE's Produce Object does not change format.)

Table 5-8 Consume Assembly, Array Type, With Handshaking

Byte 1	Byte 2	Bytes 3-X (X ≤ 51)	Byte X+1 (max = 52)
New Data Record Byte	Transaction ID Byte	ASCII Data	<CR> (Terminator)

Table 5-9 Consume Assembly, Short_String Type, With Handshaking

Byte 1	Byte 2	Byte 3	Bytes 4-X (X ≤ 52)	Byte X+1 (max = 53)
New Data Record Byte	Transaction ID Byte	Length Byte	ASCII Data	<CR> (Terminator)

Table 5-10 Consume Assembly, String Type, With Handshaking

Byte 1	Byte 2	Byte 3	Byte 4	Bytes 5-X (X ≤ 53)	Byte X+1 (max = 54)
New Data Record Byte	Transaction ID Byte	Length (MSB)	Length (LSB)	ASCII Data	<CR> (Terminator)

The Master monitors this new data flag and when the Master is ready to receive new serial data, it sets a new number in the new record number byte of the next poll command message. Note that this applies only to **data being sent from the JDCE** to the Master.

The operation proceeds as follows:

- The JDCE receives a new data string
- The JDCE sets the New Data Flag in the Status byte of its next produce message.
- The Master sends out messages to the JDCE in the normal fashion. If the Master is not ready to receive new data, the New Record Data byte remains constant.
- When the Master is ready to receive the new data string, it changes the New Record Data byte to any value different than what it had been sending.
- The JDCE will send the new data upon receipt of a record from the Master in which the New Record Data byte has been changed.
- If the JDCE receives an updated New Record Data byte and has no new, it will set the Handshake Error bit in its Produce Status byte.

In **Immediate Mode** this handshaking is not active and the JDCE sends new data as soon as it is received from the ASCII device. It is the Master's responsibility to be ready to accept and process the new data string when it is presented.

5.4.11 Explicit Messages to Transmit Serial Data String

You can use Parameters 24, 25 and 26 to set up the serial data string and send it to your ASCII device via the Explicit Messaging technique.

Parameter 24 will hold the data you wish to send.

Parameter 25 defines the size, in bytes, of the DeviceNet message to be sent from the JDCE to the device.

Parameter 26 holds the record number of the data string in Parameter 24 if the Header option is selected. Changing the record will cause the data in Parameter 24 to be transmitted immediately from the Master to the JDCE.

5.5 Setting up DeviceNet Communications

The 1782-JDCE supports 4 modes of data transfer of the serial buffer. They are:

- Polled I/O
- Change-of-State I/O
- Cyclic I/O
- Explicit Message

5.5.1 Polled I/O

The polled connection is the only manner in which you can send serial output data to the I/O and, therefore, to your I/O device. The DeviceNet Master initiates the polled connection transfer. The Master sends the JDCE its serial output buffer along with a record number and length byte. The JDCE monitors the record number for a change in the record number. If the record number changes, then the 1782-JDCE transmits the data buffer on its serial link. If the record number does not change, then the device does not transmit the data buffer.

After the device has transmitted its data out to the serial link, the JDCE then takes any information that is stored in its current serial input buffer and sends this data to the DeviceNet Master. It sends all characters up to and including the received delimiter, padding only if specified in the parameter object. When the JDCE receives a new message (either with a delimiter or with an overflow condition without a delimiter) the device then increments the receive record, updates the length byte, and copies the new information from the last receive delimiter into the buffer. If an overflow occurs, the JDCE indicates so in its receive status bit. The receive status byte also reflects parity errors in the device.

5.5.2 Cyclic and Change-of-State I/O

The Cyclic connection initiates a transmission every time the connection timer expires. This is explained below in the Section 5.5.4. The cyclic connection can only send data from the JDCE. If you need to transmit on the I/O link, you will need to use the polled connection to do so. The polled and cyclic connections are not exclusive, so both can exist at the same time. The manner in which cyclic connection reports its data is the same as the polled connection. The cyclic connections transmit buffer is the same as the polled connections transmit buffer, so overflows and received delimiters act the same over any connection.

The Change of State (COS) connection is the same as the cyclic connection except that as well as triggering communications on the expiration of the timer, the COS connection also initiates a transfer on a receive of the delimiter or an overflow. The COS connection is mutually exclusive with the cyclic connection, but can coexist with the polled connection. The COS connection operation is very useful in conserving bandwidth, and provides the Master with the most current data as fast or faster than a poll connection. The COS connection automatically turns on the COS mechanism when the connection is created.

5.5.3 Setting up the DeviceNet I/O Connections

It is useful to first set up your serial link before setting up your connection. To set up the communications with your network configuration tool, it is often necessary to know the connection input and output sizes. Instructions for setting up your serial connection are provided above. See the sections on receive and transmit sizes.

If you are using a network configuration tool with some type of scanner or scanning software, you must direct your scanner to set up the connections for you. This often requires some information about the device, such as input and output sizes. The input and output sizes are computed from the transmit size and the receive sizes. These sizes are defined in the parameter object of your device. The transmit size of the poll connection is computed by adding 2 to the Transmit Buffer Size on the 1782-JDCE. The Transmit size for the change of state and cyclic connections are set to 0, because these connections do not initiate a transmission on the serial link. The receive size of all three connections is computed by adding the number of option bytes to the receive buffer size. This final value is reported in Parameter 13.

Important: Remember to re-map the data (if necessary) after you set the sizes, because many configuration tools will automatically unmap your data when you change the connection sizes. If you are not using such a software package, it is probably not necessary to set up the transmit and receive sizes.

5.5.4 Setting up the Connection Timer (EPR)

EPR stands for Expected Packet Rate. This is the value that the JDCE sets the connection timer to for the cyclic and polled connection. This is also the value it uses in the connections to calculate the time the device should wait before signaling a timeout. If you have a scanner or scanning software, you must configure it with the EPR that you want the JDCE to be scanned with. The **scanner will then configure the EPR** in the JDCE at the beginning of communications. Consult your scanner's manuals on how to configure the EPR (the EPR is sometimes referred to as the "scan rate").

Note: If you need to set up the EPR, it can be done manually by performing a set (Service 10_{hex}) on the connection class (Class 5_{hex}) attribute 9. The polled connection uses instance 2, where as the COS and cyclic connections use instance 4. This must be done after allocating the connection.

5.5.5 Setting up the DeviceNet Baudrate

Autobaud is the mechanism that allows the DeviceNet device, in this case the JDCE, to automatically determine the baudrate that is operational on the network to which the JDCE is connected and to adjust its DeviceNet speed to match. The JDCE is shipped with autobaud as its default baudrate.

If you wish to change the baudrate to a fixed speed, you can set it in two different places. 1) The first is in the DeviceNet object (Class 3). This is where most configuration tools will look to change the baudrate. 2) Because autobaud is not supported by the standard DeviceNet Object, the JDCE provides a parameter to allow the autobaud selection. These values can be set and retrieved by using the standard Set and Get services.

A.Theory of Operation

A.1 The Transmit Record Algorithm

A.1.1 Basic Theory of operation

The Transmit record object addresses several issues of utilizing a DeviceNet network device in the process of communicating via a serial data stream. The device takes a block of data passed by the I/O and transmits this data out over the DeviceNet link. This data can be protected via a record counter for allowing a DeviceNet I/O connection to allow a device to control the transmission of this data from the JDCE by changing it.

The data flow and algorithm are shown below for reference.

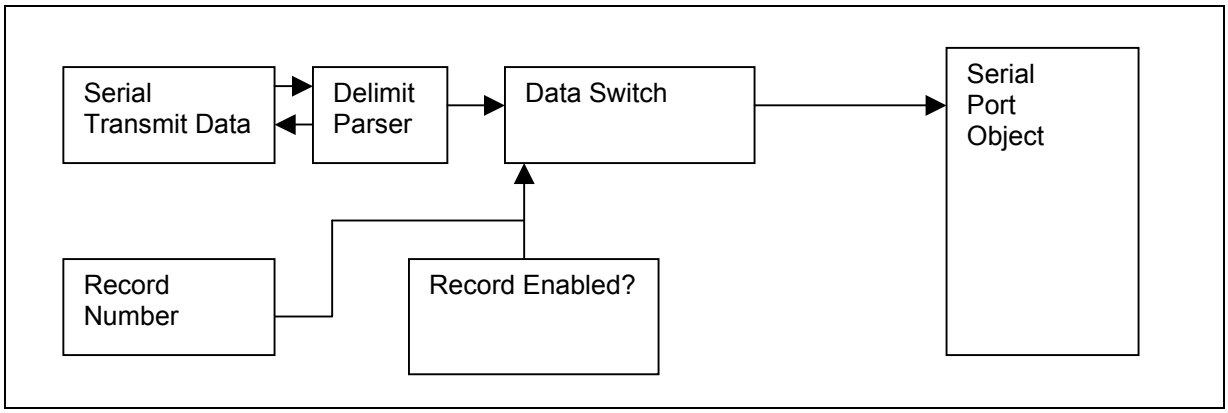

Figure A-1 Transmit Record Algorithm Functional Flowchart

As Diagramed above, the serial data is set and then parsed if the serial data length is set to zero. The serial data parse then sets the length on the serial data. If Record is disabled, the data is immediately added to the serial port's transmit queue. Also, whenever the record number is changed, the serial transmit data is sent to the serial port object.

The delimit parse determines the length of the string by searching the string for a delimiter character. The algorithm can be configured to ignore, include or exclude the character.

A.2 The Receive Record Algorithm

The receive record algorithm was developed in order to enable the receipt of data from a remote serial device and the subsequent transmission of data over a DeviceNet I/O connection. The Receive Record Algorithm Controls when data is acquired and when that acquired data is presented to the connection. This Section will help you understand the algorithm in order to efficiently utilize all of the capabilities of the JDCE to make your job easier.

A.2.1 Basic Theory of operation

The Basic function of the JDCE is to address the problem of receiving and transmitting serial data over DeviceNet. Serial data is a data stream, in which data is presented to a device one byte at a time and does not have a beginning or an end. DeviceNet however, is not stream oriented; data is presented in chunks of defined sizes and bounds. The Receive Record algorithm formats the streaming data into a data block. It uses user-defined events on the receive serial link to determine where this data block starts and ends. It then uses a status byte to notify the controlling JDCE when new data is available. The JDCE then increments the record number and the new data are presented.

The Receive Record Algorithm creates a DeviceNet object and presents the data in an ordered fashion to the DeviceNet I/O connection. The Receive Record object may be reached over DeviceNet at Class 0x72. If you are going to be using explicit messaging to communicate with the device, we recommend that you access the data from this point, and not the parameter object, because the parameter object obeys slightly different rules in order to ease the interface with today's most common configuration tools.

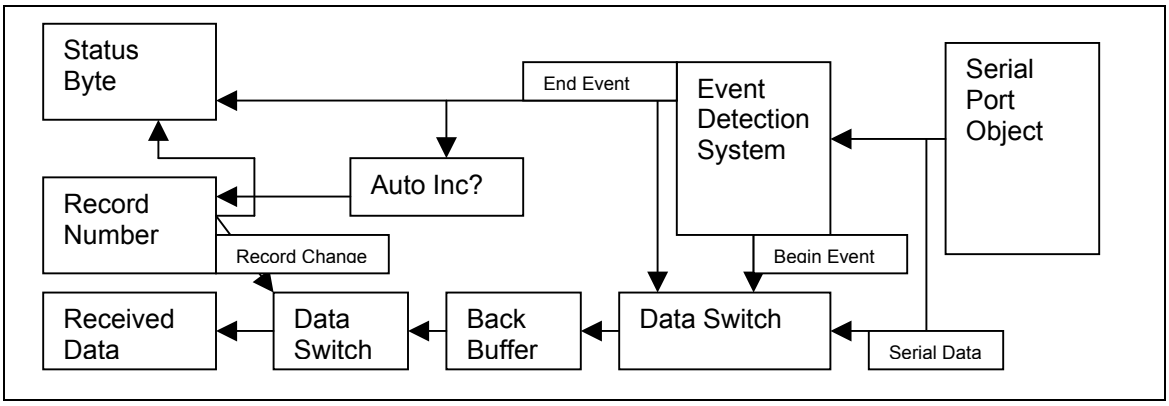

Figure A-2 Receive Record Algorithm Functional Flowchart

The Receive Record Algorithm is diagramed above. The serial stream originates from the serial port object (0x70). It is looked at a byte at a time by the event detection system. If the event detection system detects a beginning event (the beginning delimiter is received or disabled), it opens the first data switch, allowing the serial data to accumulate in the back buffer. Once an end event is received (the End delimiter is received, or the back buffer is full) the event detection system will close the data switch and set a new data status. When the new data status is set, if the device is set up in auto increment mode, the object will automatically increment the record number and clear the new data bit. In this situation, the user will not be allowed to set the Receive Record Number, and it will not be included in the poll request assembly.

The JDCE now includes functionality to allow you to specify the data type of the string. The string may be set to the DeviceNet data types of STRING, SHORT_STRING or ARRAY. The difference between the data types is how the length of the string is reported. The STRING data type has a two-byte indicator for length. This data type may be used to directly map your data into an a-b that supports the string data file. The SHORT_STRING data type is the classic data type from the JDCE revisions 5 and under. This data type has a 1-byte length, allowing for backwards compatibility and data space savings. The ARRAY data type does not have a length field and may be used if you do not need to know the length of your received data.

The JDCE also supports scanners that are not fully DeviceNet compliant and do not support

short poll responses. These short poll responses save bandwidth on the DeviceNet network, but many scanners do not support this functionality and will not allow communication with a device that responds in this manner. The JDCE supports this non-compliant behavior through pad mode. This is a function where a character is appended to the end of the serial data in order to fill out the poll response. The default for pad mode is OFF, and the default for the pad character is 0 (NULL). Turn pad mode ON if you receive errors indicating that the I/O data response is too short.

The JDCE also supports byte-reordering to support non-string data type I/O's. The DeviceNet network data-ordering scheme is little endian, meaning that the low byte of a multi-byte messages transmitted first. This means that if you are using an I/O that is using a 16 bit or larger word size and you map the I/O's data directly into this space without string support, your data bytes will show up swapped. The JDCE implements byte reordering (swapping) in order to ease use on these PLCs. The swapping mechanism re-orders the I/O data so that is appears correctly on I/O's that use Little endian byte order and greater than 8 bit word sizes. To Implement byte swapping, determine the number of bytes in your data size. Subtract 1 from this number, and set the Receive Record object's byte swapping attribute to this value (also available through the parameter object). Now, you must set the length of your message size to a multiple of 1+this value, or the JDCE will not be able to swap the bytes correctly. The data will now be ordered properly in your JDCE.

B. DeviceNet Profile, Objects and Services

B.1 1782-JDCE DeviceNet Profile

This section describes the DeviceNet Objects present in the I/O. The I/O conforms to a Type 12, Communications Adapter Device.

Table B-1 DeviceNet Objects

Object	DeviceNet Object Class	# of Instances
Identity	1	1
Message Router	2	1
DeviceNet	3	1
Connection	5	3 (Explicit Msg., Polled I/O, COS/Cyclic)
Parameter	15 (F_{hex})	28
Serial Port	112 (70_{hex})	1
Serial Transmit	113 (71_{hex})	1
Serial Receive	114 (72_{hex})	1

B.2 Serial I/O Polled Data Formats

Table B-2 Poll Produce Data (ASCII Receive String)

Byte	Character	Description	When Used
0	T/Id #	Transaction ID, Integer value 0 – 255 (0 = Initialized State)	Header Mode
1	Status	Status / Error Value	Header Mode
2	Length	Length of valid data in bytes	With String or Short_String
3	Length	Length of valid data in bytes	With String
4	Delimiter	Start-of-Text Character	Include Start Delimiter
4	1st Char	ASCII Character	
5	2nd Char	ASCII Character	
•			
•			
•			
•	Last Char	ASCII Character	
•	Terminator	End-of-Text Character	Include End Delimiter
•	Any	Pad character (present if characters received is less than Max Receive Chars value)	Pad Mode
•	Any	Pad character (present if characters received is less than Max Receive Chars value)	Pad Mode
Max Rec. Char + 2	Any	Pad character (present if characters received is less than Max Receive Chars value)	Pad Mode

Table B-3 Poll Consume Data (ASCII Transmit String)

Byte	Character	Description	When Used
0	New Data Record Number	Integer value changed to indicate to send new data	Master-Slave Handshake Mode
0	T/ID #	Transaction ID, Integer value 0 – 255 (0 = Initialized State)	Header Mode
1	Length	Number of bytes to transmit. 0 indicates transmit delimited mode, in which the device transmits up to and including the transmit delimiter character defined in the parameter object.	String or Short_String
2	Length	Number of bytes to transmit. 0 indicates transmit delimited mode, in which the device transmits up to and including the transmit delimiter character defined in the parameter object.	String or Short_String
2	1st Char	ASCII Character	
3	2nd Char	ASCII Character	
4	3rd Char	ASCII Character	
•			
•			
•			
•	Last Char	ASCII Character	
•	Terminator	End-of-Text Character	Include End Delimiter
•			
•			
•	Any	Undefined	
•	Any	Undefined (last char in this record transmitted to Master)	

B.3 Identity Object, Class 1

Instances 0 and 1 exist in the 1782-JDCE.

Table B-4 Identity Object Class Attributes (Instance 0)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Value
1	Get	Revision	UINT	Revision of this object	1
2	Get	Max. Object Instance	UINT	Maximum instance number of an object currently	1
6	Get	Max. Class Attribute ID	UINT	Attribute ID number of the last class attribute of the class definition implemented in the device	7
7	Get	Max. Instance Attributes ID	UINT	Attribute ID number of the last instance attribute of the class definition implemented in the device	1

Table B-5 Identity Object Instance Attributes (Instance 1)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Value
1	Get	Vendor	UINT	ODVA Vendor Number for this product	9 = WRC
2	Get	Device Type	UINT	ODVA Communications Device Type	12 = Comm. Adapter
3	Set	Product Code	UINT	WRC Unique Product Code Number	701 = $2bd_{hex}$
4	Get	Revision	STRUCT of:	Revision of this device	
		Major Revision	USINT		>=5
		Minor Revision	USINT		>=1
5	Get	Status	WORD	Summary status of device	
6	Get	Serial Number	UDINT	WRC Unique Device Serial Number	
7	Get	Product Name	SHORT_STRING	ASCII Name of product	1782-JDCE
10	Get/Set	Heartbeat Interval	USINT	The interval in second that the device generates a heartbeat message. A value of 0 disables heartbeat generation.	0

Table B-6 Identity Object Common Services

Service Code	Class	Instance	Service Name	Description of Service
$O5_{hex}$	Yes	Yes	Reset	Invokes the Reset Service for the device.
OE_{hex}	Yes	Yes	Get_Attribute_Single	Returns the contents of the specified attribute.
10_{hex}	No	Yes	Set_Attribute_Single	Modifies an attribute value.

B.4 Parameter Object, Class F_{hex} (15_{dec})

There are many configurable data parameters associated with your JDCE. The JDCE uses a Parameter Object (a collection of these parameters) to assist you in reading and changing configurable data.

Following are the Class Attributes, Instance Attributes and Services that are supported by the JDCE for the Parameter Object.

Table B-7 Parameter Class Attributes (Instance 0)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Value
1	Get	Revision	UINT	Revision of this object.	1
2	Get	Max. Instance	UINT	Maximum instance number of the Parameter object	9
8	Get	Parameter class descriptor	WORD	Bits that describe parameters.	9 (supports parameter instances, params are stored in non-volatile memory)

Table B-8 Parameter Instance Attributes (Instances 1-7)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Value
1	Set	Parameter Value	*data type* specified in Descriptor Data Type and Data Size.	Actual value of parameter. It can be read from or written to. This attribute is read-only if bit 4 of Attribute 4 is TRUE.	
2	Set	Link Path Size	USINT	Size of link path. If this is 0, then no link is specified.	Number of bytes
3	Set	Link Path	ARRAY of DeviceNet path:	DeviceNet path to the object from where this parameter's value is retrieved.	
4	Get	Descriptor	WORD	Description of parameter.	
5	Get	Data Type	USINT	Data type code.	
6	Get	Data Size	USINT	Number of bytes in Parameter Value	

Table B-9 Parameter Common Services

Service Code	Class	Instance	Service Name	Description of Service
$O5_{hex}$	Yes	N/A	Reset	Resets all parameters to "out-of-the-box" values.
OE_{hex}	Yes	Yes	Get_Attribute_Single	Returns the contents of the specified attribute.
10_{hex}	Yes	Yes	Set_Attribute_Single	Modifies an attribute value.

B.5 Serial Port Object (0x70)

Table B-10 Serial Port Object Instance Attributes

Parameter	Param. Instance	Access	Description	Parameter Choices		Default Setting	Default Value	Data Type
Status	1	Get	Serial port status	1 – TX I/O Overflow 2 – Rx I/O Overflow 4 – Rx Parity Error		OK	0	BYTE
Reserved	2	N/a						
Reserved	3	N/a						
Serial Character Framing Format	4	Get/Set	Character framing	0 = 7N2 1 = 7E1 2 = 7O1 3 = 8N1 4 = 8N2	5 = 8E1 6 = 8O1 7 = 7E2 8 = 7O2	7N2	0	USINT
Serial Baud Rate	5	Get/Set	I/O/ I/O communications speed	0 =9600 1 = 1200 2 = 2400	3 = 4800 4 = 19.2k 5 = 38.4k	9600 baud	0	USINT
Notify Rx Path	6	Get	Object to notify when receive events happen	EPATH		NULL		EPATH
Notify Tx Path	7	Get	Object to notify when transmit events happen	EPATH		NULL		EPATH

B.6 Transmit Record Object, Class 113 (0x71)

Table B-11 Transmit Record Object Instance Attributes

Parameter	Param. Instance	Access	Description	Parameter Choices	Default Setting	Default Value	Data Type
Hardware Interface Instance	1	Get	Serial port number	1 serial port only	1	1	USINT
Tx Record Number	2	Get	Record number assigned to the data string to be sent to the serial device	0-255	0	0	USINT
String Data	3	Get	String data to send to the serial device	N/a	None	0	USINT
Data String Type	4	Get /Set	Format of Tx data	0 = Array 1 = Short_String 2 = String	Array	0	USINT
Max Number of Tx Chars	5	Get /Set	Maximum number of characters the JDCE expects to transmit port to the serial device	0 – 50	20 chars	20	USINT
Transmit End Delimiter	6	Get /Set	Character which identifies the end of the data string when the length is specified as 0	Any valid standard I/O character (0 – 127, 0-255)	Carriage return	D_{hex}	USINT
Transmit Record End Mode	7	Get /Set	Selects if the transmit delimiter is used, included in the data string or excluded in the resultant data string	0 =No Delimiter 1 = Exclude the delimiter 2 = Include The delimiter	Include	2	USINT
Receive Swap Mode	8	Get /Set	If enabled, the position of the bytes in the serial messages will be swapped every 2 or 4 bytes.	0 = Disabled 1 = 16-bit Swap Enabled 2 = 24-bit Swap Enabled 3 = 32-bit Swap Enabled	Disabled	0	USINT

Status	9	Get	Status of the Record Object		No Error	0	BYTE

B.7 Receive Record Object, Class 114 (0x72)

Table B-12 Receive Record Object Instance Attributes

Parameter	Param. Instance	Access	Description	Parameter Choices	Default Setting	Default Value	Data Type
Serial Port Instance	1	Get	Instance of the serial port object that we listen to	Only one serial port is available on the 1782-I/O	1	1	USINT
Last Record Number	2	Get/Set	Record number assigned to the last received data string	0-255	0	0	USINT
Received Serial Data	3	Get	Last received serial data	N/a	None	0	USINT
Data Format	4	Get/Set	Format of data	0 = Array 1 = Short_String 2 = String	Short String	1	USINT
Pad Mode	5	Get/Set	Indicates whether to pad the invalid data region after the delimiter with the pad character, or to use variable length I/O responses	0 = Pad Mode Disabled 1= Pad Mode Enabled	Disabled	0	USINT
Pad Character	6	Get/Set	The value to use to pad the invalid data portion of the poll response	Any valid standard I/O character (0 – 127, 0-255)	NULL	0	USINT

Max Receive Characters	7	Get/Set	Maximum number of characters the 1782-I/O expects to receive into its I/O port from the serial device	0 – 50	20 chars	20	USINT
Starting Character Delimiter	8	Get/Set	Character which identifies the beginning of the data string from the I/O device when the length is specified as 0	Any valid standard I/O character (0 – 127, 0-255)	Carriage return	D_{hex}	USINT
Ending Character Delimiter	9	Get/Set	Character which identifies the end of the data string from the I/O device when the length is specified as 0	Any valid standard I/O character (0 – 127, 0-255)	Carriage return	D_{hex}	USINT
Delimiter Mode	10	Get/Set	Selects whether or not the beginning delimiter is included in the data string	0 = No Delimiter 1 = Exclude Delimiter 2 = Include Delimiter	No Delimiter	0	USINT
Include /Exclude Delimiter	11	Get/Set	Selects whether or not the end delimiter is included in the data string	0 =No Delimiter 1 = Exclude Delimiter 2 = Include Delimiter	Include	2	USINT
Byte Swap Mode	12	Get/Set	If enabled, the position of the bytes in the serial messages will be swapped every 2 or 4 bytes.	0 = Disabled 1 = 16-bit Swap Enabled 2 = 24-bit Swap Enabled 3 = 32-bit Swap Enabled	Disabled	0	USINT
Auto Increment	13	Get/Set	Defines if the received string is sent immediately to master (in poll mode) or after a I/O handshake ACK	0 = Set New Data Status Bit and wait for I/O to increment record number 1 = Automatically increment the record number and show the new data when an end event is received	Auto Increment	1	USINT
Record Status	14	Get	Status of the Record object	Bit 7 – New Data Available	No Status	0	BYTE

B.8 Common DeviceNet Services

DeviceNet is divided into logical functional blocks called objects, which provide services that allow for control over the hardware and routines that those objects contain. To allow for multiple similar functions, the objects are built of multiple instances that the services of the objects act upon. A class service acts upon the entire object, allowing one service to be enacted on all of the instances. This saves time, effort and network bandwidth.

The common services are a common set of services that have been provided in most or all of the objects to allow for common functionality in creating, deleting, getting, setting and resetting the variables of the different classes and instances. We will describe two of the services here: get and set.

The get and set services have a common format for specifying what object, instance, attribute and service that the command is specifying. In order of first to last, DeviceNet specifies service, class, instance, attribute and data. The data is always little endian (low byte precedes high order byte), and the others are all one byte in length on the JDCE. Note that the get service has no data.

The get service gets data from an attribute of a class or class instance. The service number of this request is 14 (E_{hex}). The class instance and attribute are all defined by which variable you want to get. The response from a get command takes on the form: service, value. The value will be little-endian and can be of variable length and bounds based on the definition of the attribute. The service will be reported as the get service with the highest bit set to indicate a response.

The set service sets data from an attribute of a class or class instance. The service number of this request is 16 (10_{hex}). The class instance and attribute are all defined by which variable you want to set. The value is little endian, and the size is defined by the attribute that you are setting. The response from a set command only echoes the service. The service will be reported as the set service with the highest bit set to indicate a response.

An error response will have the service set to 94_{hex}. This response will be followed by a two-byte error code, defining the type of fault. For a detailed list of error codes, connect to the ODVA web site at www.odva.org.

A. Accessories and Other WRC Products

The following components can be used with an Ajax for replacements or spare parts, or as complementary devices as a part of your DeviceNet or other CAN-Bus system.

Table A-1 WRC Replacements, Spare Parts and Other Products

Part	WRC Part Number
DIN rail	WRC 50022
Terminating resistor, axial lead	RM121DN
Discrete I/O block – 4 channels	1782-JDB4
Discrete I/O block – 8 channels	1781-JDB8
Analog Input block – 4 channels, 10-bit	1782-JDA4
Analog I/O block – 8 channels, 12-bit	1782-JDA8
DeviceNet to Serial I/O Gateway	1782-JDC
Extended DeviceNet to Serial I/O Gateway	1782-JDCE
DeviceNet to Modbus Gateway	1782-JDM
Discrete I/O block – 24 channels	WRC1-JDB24
Discrete I/O block – 48 channels	WRC1-JDB48
Discrete I/O, Analog Input block – 24 DIO, 32 AI	WRC1-JDA/24
Discrete I/O, Analog Input block – 48 DIO, 32 AI	WRC1-JDA/48
Analog I/O block - 32 channels	WRC1-JDAIO
Discrete and Analog I/O block – 24 DIO, 32 AIO	WRC1-JDAIO/24
Discrete and Analog I/O block – 48IO, 32 AIO	WRC1-JDAIO/48
Discrete I/O block – 8 DIs, 8 DOs, 4 AIs	W5-JDB16x
DeviceNet, CANopen Extender, DIN mount	WRC-CANX-DIN-DN
SDS Extender, DIN mount	WRC-CANX-DIN-SD
DeviceNet, CANopen Extender, DIN mount	WRC-CANX-DIN-C7
DeviceNet, CANopen Extender, NEMA box	WRC-CANX-NEM-AU
DeviceNet, CANopen Extender, NEMA box	WRC-CANX-NEM-DN
SDS Extender, NEMA box	WRC-CANX-NEM-SD
DeviceNet, CANopen Extender, Fiber Optic, NEMA box	WRC-CANR-DF-DN