

***Not Recommended for New Installations.***  
Please contact Technical Support for more information.

***Addressable RS-485 Data Acquisition  
and Digital to Analog Conversion Module***  
**485SPDA and 485SPDACL**  
Documentation Number 485SPDA3798

This product  
**Designed and Manufactured  
In Ottawa, Illinois  
USA**

of domestic and imported parts by

**B&B Electronics Mfg. Co. Inc.**  
707 Dayton Road -- P.O. Box 1040 -- Ottawa, IL 61350  
PH (815) 433-5100 -- FAX (815) 433-5105

**Internet:**  
<http://www.bb-elec.com>  
[orders@bb-elec.com](mailto:orders@bb-elec.com)  
[support@bb.elec.com](mailto:support@bb.elec.com)

Copyright ©1996-1997 by B&B Electronics Mfg. Co.  
All rights reserved.  
Revised September 1998

# TABLE OF CONTENTS

<b>CHAPTER 1- INTRODUCTION.....</b>	<b>1</b>
485SPDA FEATURES.....	1
PACKING LIST.....	3
SOFTWARE INSTALLATION.....	3
GETTING STARTED.....	3
485SPDA / 485SPDA <sub>CL</sub> SPECIFICATIONS.....	4
<i>Analog to Digital Converter</i> .....	4
<i>5 Volt Reference</i> .....	4
<i>D/A Converter</i> .....	5
<i>4-20mA Current Loop Output (485SPDA<sub>CL</sub> only)</i> .....	5
<i>Digital Inputs</i> .....	5
<i>Digital Output</i> .....	5
<i>Power Supply</i> .....	5
<i>Communications</i> .....	6
FACTORY DEFAULT SETTINGS.....	6
<b>CHAPTER 2 - CONNECTIONS.....</b>	<b>7</b>
A/D CONNECTIONS.....	7
<i>A/D Inputs #0-6</i> .....	7
<i>A/D Ref Input +</i> .....	7
<i>A/D Ref Input -</i> .....	7
<i>Analog Ground</i> .....	8
<i>Typical Connections</i> .....	8
485SPDA <sub>CL</sub> MODULE.....	9
D/A CONNECTIONS.....	9
<i>D/A 0-3</i> .....	9
<i>D/A Ref 0-3</i> .....	9
<i>Typical Connections</i> .....	10
4-20MA CURRENT LOOP OUTPUT CONNECTIONS.....	10
<i>Current Loop Output</i> .....	10
<i>4-20mA Current Loop Reference</i> .....	11
<i>Typical Connections</i> .....	11
<i>Converting the Current in the Loop to a Voltage</i> .....	11
DIGITAL I/O CONNECTIONS.....	12
<i>Digital Inputs #0-1</i> .....	12
<i>Digital Output #0</i> .....	12
<i>Digital Ground</i> .....	12
<i>Typical Connections</i> .....	13
SERIAL PORT CONNECTIONS.....	13
POWER SUPPLY CONNECTIONS.....	14
<b>CHAPTER 3 - COMMANDS.....</b>	<b>15</b>

SYNTAX .....	17
READING A/D CHANNELS COMMAND .....	18
OUTPUT ANALOG VOLTAGE COMMAND.....	18
OUTPUT 4-20MA CURRENT (485SPDACL ONLY!) .....	19
READING DIGITAL I/O COMMAND.....	19
SET DIGITAL OUTPUT COMMAND.....	20
SET MODULE ADDRESS COMMAND .....	20
SET POWER-UP STATES COMMAND .....	21
SET TURN-AROUND DELAY COMMAND.....	21
READ MODULE CONFIGURATION COMMAND .....	22
<b>CHAPTER 4 - A/D.....</b>	<b>23</b>
SAMPLING RATE .....	23
A/D INPUT RANGE.....	23
REFERENCE INPUTS.....	23
DATA RANGE.....	23
CONVERTING DATA .....	24
<b>CHAPTER 5 - D/A .....</b>	<b>27</b>
D/A REFERENCE INPUTS.....	27
DIGITAL INFORMATION FORMAT .....	28
MULTIPLIER BIT .....	28
CALIBRATING D/A REF 0-3 .....	29
CONVERTING DATA .....	29
<b>CHAPTER 6 - 4-20MA CURRENT LOOP OUTPUT.....</b>	<b>31</b>
<b>CHAPTER 7- SOFTWARE.....</b>	<b>33</b>
READ A/D COMMAND .....	33
READ DIGITAL I/O COMMAND .....	34
SET DIGITAL OUTPUT STATE.....	36
SET MODULE ADDRESS .....	36
SET POWER-UP STATES.....	37
SET TURN-AROUND DELAY.....	38
READ MODULE CONFIGURATION COMMAND .....	39

<b>CHAPTER 8: API FUNCTIONS .....</b>	<b>41</b>
COMPILING AND LINKING .....	41
<i>Borland / Turbo C++</i> .....	41
<i>Borland / Turbo Pascal</i> .....	41
<i>Microsoft® QuickBASIC</i> .....	41
FUNCTION REFERENCE .....	42
<i>deinitComPort</i> .....	42
<i>initComPort</i> .....	42
<i>spda_get_module_type</i> .....	43
<i>spda_deinit</i> .....	44
<i>spda_init</i> .....	44
<i>spda_read_analog_inputs</i> .....	45
<i>spda_read_config</i> .....	45
<i>spda_read_digital_io</i> .....	46
<i>spda_set_analog_outputs</i> .....	47
<i>spda_set_digital_outputs</i> .....	47
<i>spda_set_error_detection</i> .....	48
<i>spda_set_module_address</i> .....	48
<i>spda_set_power_up_states</i> .....	49
<i>spda_set_turn_around_delay</i> .....	49
<i>spda_startup</i> .....	50
<b>APPENDIX A: DECIMAL TO HEX TO ASCII TABLE .....</b>	<b>A-1</b>
<b>APPENDIX B: 485SPDAL/CL CIRCUIT SCHEMATIC .....</b>	<b>B-1</b>

# Chapter 1- Introduction

## 485SPDA Features

This manual covers both the 485SPDA and the 485SPDACL modules. The 485SPDA is a general purpose control module which operates through an RS-485 interface. The 485SPDA offers 7 channels of 12-bit A/D, 4 channels of 8-bit D/A, 2 digital inputs and 1 digital output. The only difference between the two modules is a two-wire, 4-20mA current loop output which is denoted by the CL suffix. With these features, the module can be used to sense a wide range of external conditions and to control a variety of devices.

The 7 A/D channels allow you to measure voltages from 0 to 5 Volts. The 4 D/A channels allow you to output a voltage between 0V and 4.3V. The digital output is CMOS compatible. The 2 digital inputs are CMOS/TTL compatible. The 485SPDACL uses D/A 0 to control a 4-20mA current loop output. The A/D, D/A and digital I/O lines are available through a DB-25S (female) connector.

The 485SPDA connects to the host computer's RS-485 or RS-422 serial port using terminal blocks. The address and turn-around delay are software programmable to allow for use of multiple devices or connection to existing systems. The unit automatically detects baud rates from 1200 to 9600. A data format of 8 data bits, 1 stop bit and no parity is used.

Configuration parameters are stored in non-volatile memory. The configuration parameters consist of the module address, communication turn-around delay and digital output power-up states.



Figure 1.1 - 485SPDACL Unit

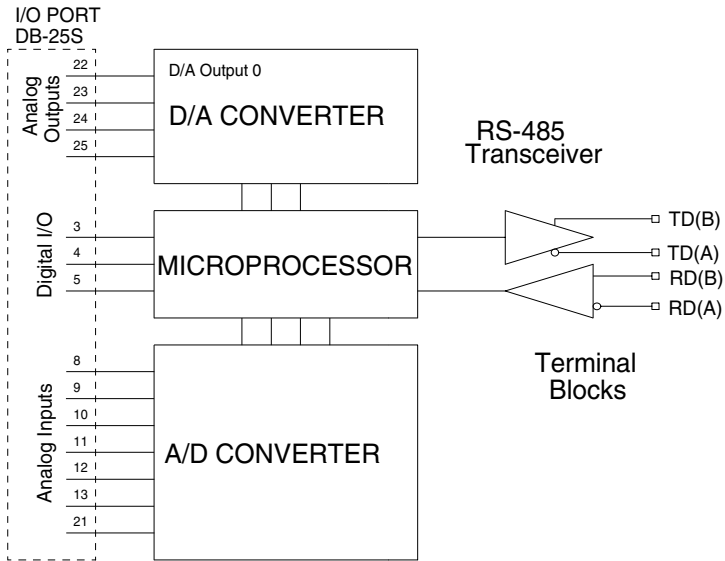


Figure 1.2 - Block Diagram of 485SPDA

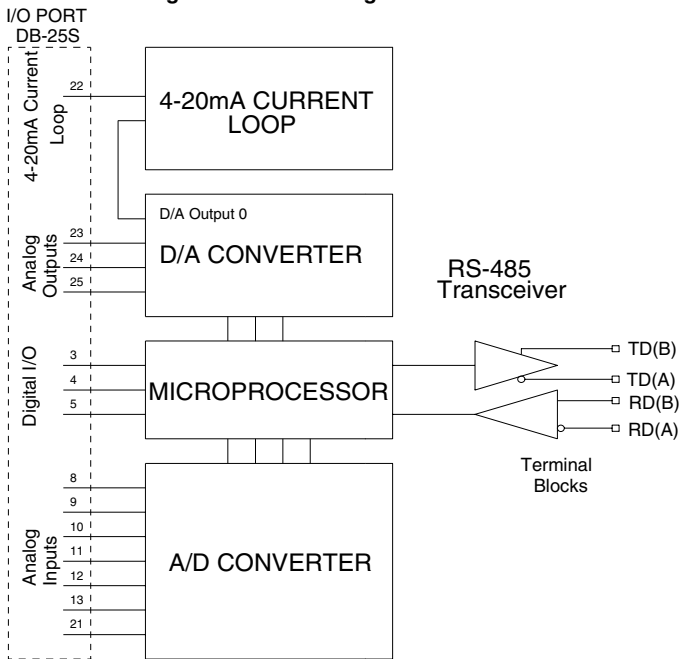


Figure 1.3 - Block Diagram of 485SPDACL

## Packing List

Examine the shipping carton and contents for physical damage. The following items should be in the shipping carton:

1. 485SPDA / 485SPDA CL unit
2. One 485SPDA / 485SPDA CL 3.5" disk
3. This instruction manual

If any of these items are damaged or missing contact B&B Electronics immediately.

## Software Installation

The 485SPDA comes with several demonstration programs. Your disk contains a file named "FILES.LST", which is a listing of the programs with descriptions. To install these files on your hard drive:

- Place the disk in drive A.
- Type **A:** and press the <ENTER> key.
- Type **INSTALL** and press the <ENTER> key.
- Follow the instructions given by the program.

## Getting Started

This section will provide a quick example using the 485SPDA and the demonstration program. If you experience any problems, refer to Chapter 2 for more precise information on connections. The demo program continually reads the A/D inputs and the digital I/O, and the D/A outputs and digital output can be continuously changed. The serial port is configured for 9600 baud, 8 data bits, No parity, 1 stop bits. The program supports standard addresses and IRQ's for COM1 and COM2.

Connect a 0 to 5VDC analog device to A/D Input #0 or you can connect a variable resistor as shown in Figure 1.4. The variable resistor must be greater than 1k Ohms to limit the output current to 5 milliamps. Connect A/D Ref. Input + to +5VDC. Connect A/D Ref. Input - to analog ground (See Figure 1.4). Then connect the 485SPDA to a RS-422 or 4-wire RS-485 serial port. Once your connections have been made, run the demo program. Any change in A/D or digital lines on the 485SPDA will automatically be displayed on the screen.

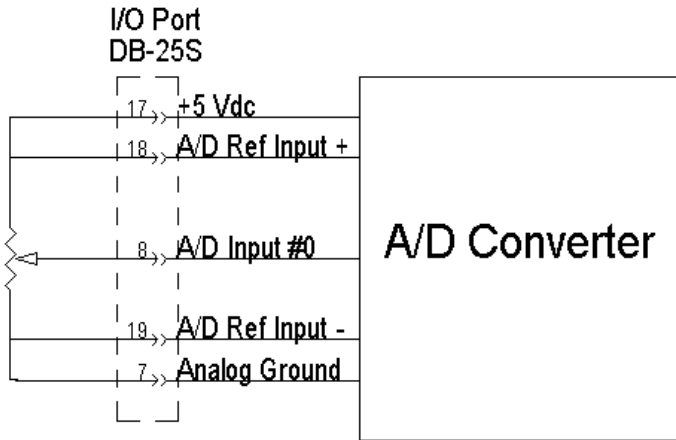


Figure 1.4 - A/D with Variable Resistor

## 485SPDA / 485SPDACL Specifications

### Analog to Digital Converter

Resolution:	12 bit
Channels:	7
Reference Range:	5.0 VDC max. (1.221 mV per bit) 2.5 VDC min. (0.610 mV per bit)
A/D Ref. Input -	0 VDC to 2.5 VDC
A/D Ref. Input +	2.5 VDC to 5.0 VDC
Input Voltage Range:	-0.3 VDC to 5.3 VDC
Total Unadjusted Error:	+/- 1.75 LSB max.

A/D input channels must be driven from a source impedance less than 1 k $\Omega$ .

### 5 Volt Reference

Output Voltage:	4.975 to 5.025 VDC (5.0 VDC typ.)
Accuracy:	+/- 0.5 %
Output Current:	5 milliamps max.

## D/A Converter

Resolution:	8 bit
Channels:	4 / (3 channels in the 485SPDACL)
Voltage Range:	0VDC to 4.3VDC
Reference Range	-0.3VDC to 5.3VDC
Output Voltage Error:	+/- 0.07V when multiplier bit (A5)=0 +/- 0.14V when multiplier bit (A5)=1

D/A output channels must have a load resistance greater than or equal to 10 k $\Omega$  and a capacitive load of less than or equal to 100 pF.

## 4-20mA Current Loop Output (485SPDACL only)

Resolution:	8 bit
Channels:	1
Output Range;	4mA to 20mA
Reference:	internally set to 2VDC
Maximum Load:	500 $\Omega$ (includes resistance of wire)
Total error:	0.20mA (absolute worst case)

## Digital Inputs

Channels:	2
Voltage Range:	-30 VDC to 30 VDC
Low Voltage:	-30 VDC to 1.0 VDC
High Voltage:	2.0 VDC to 30 VDC
Leakage Current:	1 microamp max.

## Digital Output

Channels:	1
Low Voltage:	0.6 VDC @ 8.7 milliamps
High Voltage:	4.3 VDC @ -5.4 milliamps

## Power Supply

Input Voltage:	12VDC to 16VDC @ 50 milliamps (Specification for the 485SPDACL. Does not include the power consumption of external devices.)
----------------	---

## **Communications**

Standard: RS-422/485  
Addresses: 256  
Turn-around Delay: Software programmable from 0 to 255 character transmission times. (1 char. trans. time @ 9600 baud = 1 millisecond)  
Baud Rate: 1200 to 9600 (automatic detection)  
Format: 8 data bits, 1 stop bit, no parity  
Connector: Terminal blocks

## **Factory Default Settings**

Address: ASCII "0" (48 decimal or 30h)  
Turn-around Delay: 1  
Power-up States: 0

## Chapter 2 - Connections

This chapter will cover the connections required for the 485SPDA. There are six sets of connections: A/D converter, D/A converter, 4-20mA Current Loop Output, digital I/O, serial port and power supply. Do not make any connections to the 485SPDA until you have read this chapter. If you do not intend to use a section, it is still important to read each section.

### A/D Connections

The A/D connections are made on the I/O port, which is a DB-25S (female) connector. Table 2.1 shows the pinout of the I/O port. The next sections explain the functions and connections for the various analog signals.

#### A/D Inputs #0-6

These are the analog input channels. The analog data that is read from the 485SPDA is related to the voltage on these pins. Connect your devices to the analog input channels. Voltages on these inputs must remain between 0 and 5.0 VDC. Unused A/D Inputs should be connected to analog ground.

#### A/D Ref Input +

The voltage connected to this pin determines the upper end of the input voltage range. For proper operation, this pin must be connected to a DC voltage between +2.5 and +5.0 Volts. The 485SPDA provides a 5.0V +/-0.5% reference. The 5V reference can be used if you require a 0 to 5VDC input range. If your application requires a better reference voltage or a different input range, you must supply the appropriate reference to this pin. This voltage **must be at least 2.5V greater than A/D Ref Input-**. Bypassing this pin with 0.01 $\mu$ F ceramic and 10 $\mu$ F tantalum capacitors to analog ground will help minimize noise.

#### A/D Ref Input -

The voltage connected to this pin determines the low end of the input voltage range. For proper operation, this pin must be connected to a DC voltage between 0 and +2.5 Volts. Typically, this is connected to your device's ground and analog ground, which is 0 Volts.

## Analog Ground

The pin should be connected to your analog devices ground. If ground (0V) is the low end of your input voltage range, A/D Ref Input - should be connected to this pin. To keep noise at a minimum **do not connect** analog ground and digital ground together. Unused A/D Inputs should be connected to analog ground.

## Typical Connections

Figure 2.1 shows the typical connections of the 485SPDA for a 0 to 5VDC input range.

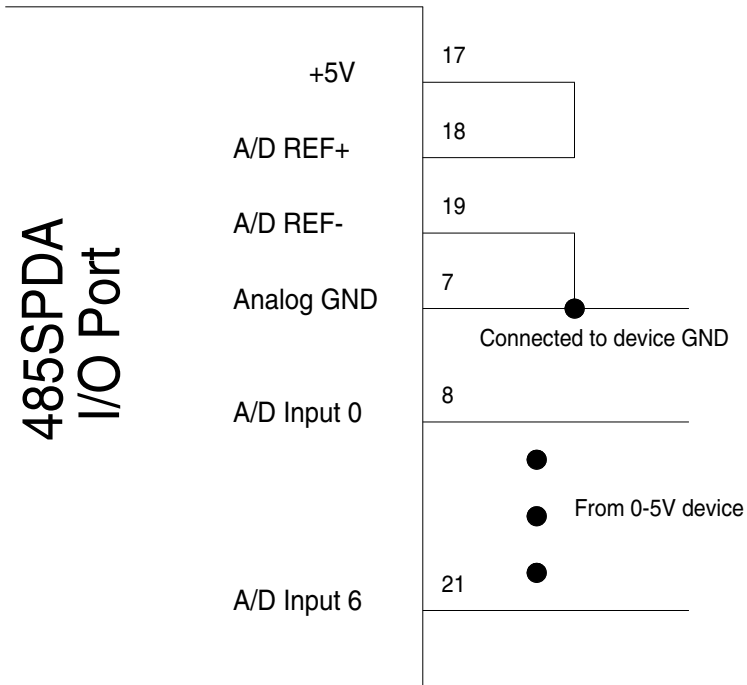


Figure 2.1 - Typical 0-5V A/D Connection

**Table 2.1 - 485SPDA I/O Port Pinout**

DB-25S Pin #	Function	DB-25S Pin #	Function
1	GND	14	D/A Ref. 1
2	+12VDC Output*	15	D/A Ref. 2
3	Digital Output #0	16	D/A Ref. 3
4	Digital Input #0	17	+5VDC Output
5	Digital Input #1	18	A/D Ref. Input +
6	Digital GND	19	A/D Ref. Input -
7	Analog GND	20	No connection
8	A/D Input #0	21	A/D Input #6
9	A/D Input #1	22	D/A 0**
10	A/D Input #2	23	D/A 1
11	A/D Input #3	24	D/A 2
12	A/D Input #4	25	D/A 3
13	A/D Input #5		

\*Actual output is equal to power supply input minus 0.7VDC

\*\*D/A Output 0 is used for the 4-20mA Current Loop in the 485SPDACL Module.

## D/A Connections

The D/A connections are made on the I/O port, which is a DB-25S (female) connector. Table 2.1 shows the pinout of the I/O port. The following sections explain the functions and connections needed to output an analog voltage.

### D/A 0-3

These are the analog output channels. An 8-bit digital value is converted into an analog voltage and is placed on these pins. The analog voltages can range from 0V to 4.3V.

### D/A Ref 0-3

Each D/A channel outputs a voltage with respect to its reference. D/A Ref 0 is internally fixed at 5.0V. D/A Ref. 1-3 are determined by the voltage applied to pin #14-16. These reference voltages must be between 0V and 5V. Although 5V is the maximum voltage that can be applied to the D/A references, 3.75V is the maximum voltage that will be used in the conversion equation. Therefore, any voltage applied to D/A Ref 1-3 greater than 3.75V will produce the same result as a 3.75V reference. The value 3.75 will vary slightly depending on each module. A calibration procedure will be covered in Chapter 5.

## Typical Connections

Figure 2.2 shows the typical connections of the unit for a 0V to 4.3V analog voltage output range.

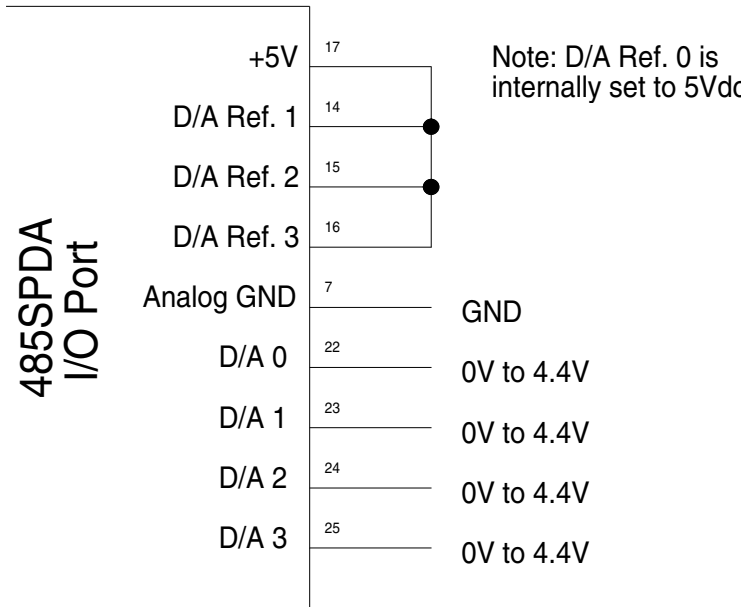


Figure 2.2 - Typical D/A connections

## 4-20mA Current Loop Output Connections (485SPDACL only!)

The Current Loop Output connections are made on the I/O port which is a DB-25S (female) connector. Table 2.1 shows the pinout of the I/O port. Note I/O pin 22 is the current loop output. The following sections explain the functions and connections needed to operate the 4-20mA Current Loop.

### Current Loop Output

The 485SPDACL has one two-wire, 4-20mA current loop output channel. I/O pin 22 is used as the current loop output. The second wire in the current loop is the GND wire which is connected to I/O pin 7. In the 485SPDACL module, D/A 0 is used internally to control the current loop output. The voltage produced by D/A 0 has a linear relationship with the current in the loop. Since, D/A 0 is used internally, only D/A 1-3 are available on I/O pins in the 485SPDACL module.

## 4-20mA Current Loop Reference

The Current Loop reference is internally set to 2VDC in the 485SPDAACL.

## Typical Connections

Figure 2.3 shows the typical connections for the 485SPDAACL to produce a 4-20mA current loop. **The maximum resistance of the current loop and receiver must be less than 500Ω!**

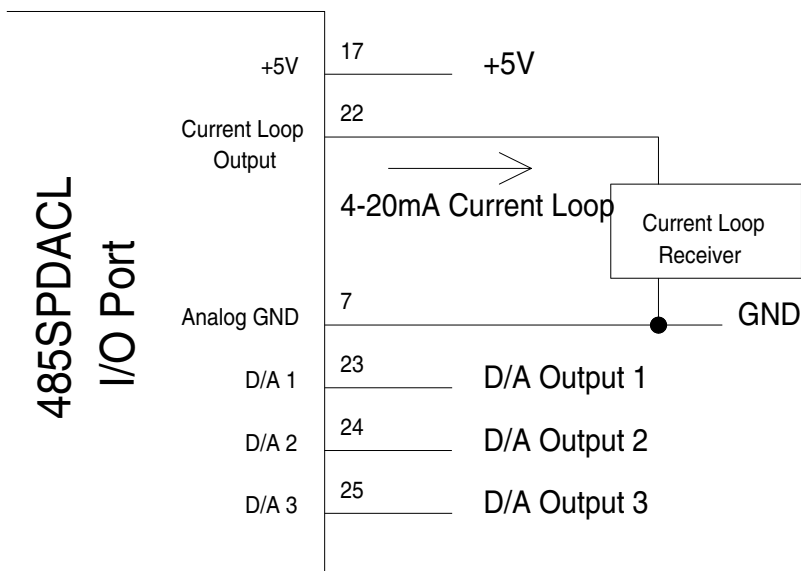


Figure 2.3 - Typical Current Loop Connections

## Converting the Current in the Loop to a Voltage

To convert the current in the loop to a voltage, you can place a resistor from I/O pin 22 to I/O pin 7. The voltage drop across this resistor has the linear relationship of  $V = I \times R$ . Figure 2.4 shows the setup needed to convert the 4-20mA current in the loop to a voltage. The resistor value plus the resistance of the wire in the loop must be less than 500Ω.

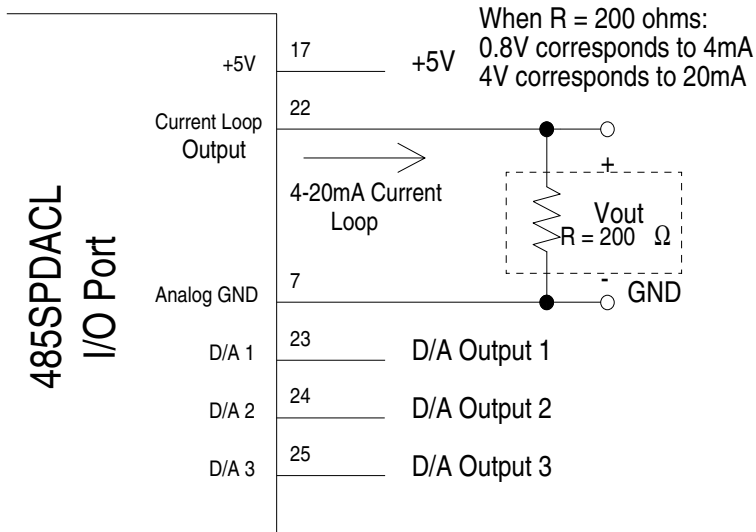


Figure 2.4 - Current to Voltage Conversion Setup

## Digital I/O Connections

The digital I/O connections are made on the I/O port, which is a DB-25S (female) connector. Table 2.1 shows the pinout of the I/O port. The next sections explain the functions and connections for the various digital signals.

### Digital Inputs #0-1

The digital input lines are CMOS/TTL compatible and can handle voltages from -30VDC to +30VDC. If a digital input is from -30 VDC to 1.0VDC, the state will be read as a “0” (LOW). If a digital input is from 2.0VDC to 30VDC, the state will be read as a “1” (HIGH). Unused digital inputs should be connected to digital ground.

### Digital Output #0

The digital output line is CMOS compatible. A digital output that is set to a “0” (LOW) will output a voltage from 0 to 0.6VDC. A digital output that is set to a “1” will output a voltage from 4.3VDC to 5.0VDC. Refer to Chapter 1 Specifications for more information.

### Digital Ground

This pin should be connected to your digital devices ground. To keep noise at a minimum **do not connect** analog ground and digital ground together. Unused digital inputs should be connected to digital ground.

## Typical Connections

Figure 2.5 shows the typical connections of the 485SPDA for the Digital I/O lines.

485SPDA I/O Port	Digital Output	3	To 0-5V device
	Digital Input 0	4	From -30 to 30V device
	Digital Input 1	5	From -30 to 30V device
	Digital GND	6	Connect to device GND

Figure 2.5 - Typical Digital I/O Connections

## Serial Port Connections

In order to communicate to the 485SPDA module it must be connected to an RS-422/RS-485 serial port. The 485SPDA will work with 2-wire or 4-wire RS-485. The unit automatically detects baud rates from 1200 to 9600. A data format of 8 data bits, 1 stop bit and no parity is used. Connections are made using terminal blocks. Table 2.2 shows the terminal blocks and their functions.

Table 2.2 - RS-485 Terminal Block Connections

TB Label	Signal	485SPDA Function	Notes
TD(A)	Transmit Data (A)	Output	Connection is required. [Loop to RD(A) for 2-wire hookup]
TD(B)	Transmit Data (B)	Output	Connection is required. [Loop to RD(B) for 2-wire hookup]
RD(A)	Receive Data (A)	Input	Connection is required. [Loop to RD(A) for 2-wire hookup]
RD(B)	Receive Data (B)	Input	Connection is required. [Loop to RD(B) for 2-wire hookup]
GND	Ground	-	Connection for Signal GND and Power Supply GND.
+12V	+12 VDC Power	Input	Connection is required.

## **Power Supply Connections**

The 485SPDACL requires 12 to 16 VDC at 50 milliamps. Remember that the 50 milliamp requirement doesn't include the power consumption of any external devices. Therefore, any current that you source with the digital outputs must be added to this value.

## Chapter 3 - Commands

There are only four commands required to control the 485SPDA: the read A/D command, output analog voltage command, read digital I/O command and the set output state command. There are also four commands that are used to configure the module: set power-up states command, set turn-around delay command, set module address command, and the read module configuration command. The command string consists of four bytes. Some commands require one or two additional data bytes. See Table 3.1.

**Table 3.1 - 485SPDA Commands**

Function	Command	Response
Read A/D Channels	!{addr}RA{#}	{ch#msb}{ch#lsb} {ch(#-1)msb}... {ch0msb}{ch0lsb}
Read Digital I/O	!{addr}RD	{I/O states}
Set Output States	!{addr}SO{#}	no response
Output Analog Voltage	!{addr}SV{#}{#}	no response
Set Module Address	!{addr}SA{new addr}	no response
Set Power-up States	!{addr}SS{#}	no response
Set Turn-around Delay	!{addr}SC{#}	no response
Read Configuration	!{addr}RC	{addr}{powerup states}{turn-around delay}

Note: Each {...} represents one byte.

An extended set of commands that provide bit error detection is available with the 485SPDA and 485SPDACL. With these extended commands, the “!” character is replaced with the “#” character. Also, the compliment of each data byte follows each data byte. See Table 3.2.

**Table 3.2 - 485SPDA Extended Commands**

Function	Command	Response
Read A/D Channels	# $\{addr\}$ RA $\{\#\}$ ~ $\{\#\}$	{ch#msb}~{ch#msb} {ch#lsb}~{ch#lsb} {ch(#-1)msb} ~{ch(#-1)msb} ...{ch0msb}~{ch0msb} {ch0lsb}~{ch0lsb}
Read Digital I/O	# $\{addr\}$ RD	{I/O states} ~{I/O states}
Set Output States	# $\{addr\}$ SO $\{\#\}$ ~ $\{\#\}$	no response
Output Analog Voltage	# $\{addr\}$ SV $\{\#\}$ ~ $\{\#\}$ $\{\#\}$ ~ $\{\#\}$	no response
Set Module Address	# $\{addr\}$ SA{new addr}~{new addr}	no response
Set Power-up States	# $\{addr\}$ SS $\{\#\}$ ~ $\{\#\}$	no response
Set Turn-around Delay	# $\{addr\}$ SC $\{\#\}$ ~ $\{\#\}$	no response
Read Configuration	# $\{addr\}$ RC	{addr}~{addr}{powerup states}~{powerup states}{turn-around delay}~{turn-around delay}

Note: Each ~{...} represents the compliment of one byte.

Before going into the specifics of each command, it is important to understand that a byte has a value from 0 to 255 and can be represented in decimal (0 to 255), hexadecimal (00 to FF), or by an ASCII character. The decimal and hexadecimal equivalents of some ASCII characters are shown in Table 3.2. Notice that the ASCII representation of the character "0" does not have a value of 0. Refer to Appendix B for more ASCII, decimal and hexadecimal equivalents.

**Table 3.2 - Equivalent Values**

ASCII	Decimal	Hexadecimal
!	33	21h
#	35	23h
0	48	30h
A	65	41h
D	68	44h
O	79	4Fh
R	82	52h
S	83	53h
V	86	56h
NUL	0	0h
SOH	1	1h
STX	2	2h
ETX	3	3h
EOT	4	4h
ENQ	5	5h
ACK	6	6h
BEL	7	7h

## Syntax

As mentioned earlier, the command string consists of four bytes. The first byte is the start of message byte. The start of message byte is always the “!” character. The second byte is the address byte. This byte allows each unit to have a unique address. The factory default address is the ASCII “0” (zero) character. The next two bytes are the command characters. These bytes are used to specify which command will be executed by the module. Some commands require a fifth and a sixth byte, data bytes.

Command Syntax: !

						Data Byte
						Data Byte
						2nd Command Byte
						1st Command Byte
						Address Byte
						Start of Message Byte

## Reading A/D Channels Command

The Read A/D channels command returns two bytes for each channel read. The two bytes represent the most significant byte (MSB) and least significant byte (LSB) of the reading. The MSB is received first, followed by the LSB. This command requires a data byte. The data byte is used to specify the number of the highest channel to be read. All channels less than this channel will be read as well. For example: if the data byte has a value of 6, then channels 0 to 6 will be read. The highest channel is read first.

Command Syntax

**!{addr}RA{#}**

Response Syntax

**{ch(#)MSB}{ch(#)LSB}{ch(#-1)MSB}...{ch0MSB}{ch0LSB}**

Where “{#}” is a byte that specifies the number of the highest channel to be read. The most significant byte of the channel specified is received first. The least significant byte and the lower channels will follow in descending order. “{chxMSB}” and “{chxLSB}” represent the most and least significant bytes of the A/D conversion result. See Table 3.3.

**Table 3.3 - Read A/D Response**

Channel # Specified			Response	
decimal	Hex	ASCII	Channels Returned (order of response)	Bytes Returned
0	0	NUL	Channel 0	2
1	1	SOH	Channels 1,0	4
2	2	STX	Channels 2,1,0	6
3	3	ETX	Channels 3,2,...,0	8
4	4	EOT	Channels 4,3,...,0	10
5	5	ENQ	Channels 5,4,...,0	12
6	6	ACK	Channels 6,5,...,0	14

## Output Analog Voltage Command

The Output Analog Voltage Command is used to output an analog voltage on one of the four D/A channels. This command requires two data bytes. These data bytes are used to select one of the four D/A channels, a multiplier, and the analog voltage to be placed on the selected D/A output line.

Command Syntax

**!{addr}SV{#}{#}**

Response Syntax

no response

Where {#}{#} are two bytes in which the 11 most significant bits represent the channel selection, multiplier, and analog voltage. Bits 4-0 of the second byte are ignored. More information pertaining to D/A channel selection, multiplier selection, and operation of the D/A is covered in chapter 5.

## **Output 4-20mA Current (485SPDACL only!)**

The output current command is used to control the current in the current loop. This command is similar to the output analog voltage. This command requires 2 data bytes which contain the current to be placed in the loop.

Command Syntax

**!{addr}SV{#}{#}**

Response Syntax

no Response

Where {#}{#} are two bytes which contain the current loop value. The three most significant bits of the first byte must be zeros to control the loop current. The exact bit format is discussed in more detail in chapter 6.

## **Reading Digital I/O Command**

The Read Digital I/O command returns a byte which represents the states of the 2 digital inputs and the digital output. Bit #3 corresponds to the state of the digital output. Bits 4-5 correspond to the states of digital inputs 0-1. If a bit is a 0 then the digital state of that digital I/O is LOW. If a bit is a 1 then the digital state of the I/O is HIGH. Refer to Table 3.4. Note: This command ignore Bits 0-2 and 6-7 of the response byte.

Command Syntax

**!{addr}RD**

Response Syntax

**{states}**

Where **{states}** is a byte in which Bit #3 corresponds to the current state of the Digital Output, and Bits 4-5 correspond to the current states of Digital Inputs 0-1.

**Table 3.4 - Read Digital I/O Response**

Response Byte			Digital I/O		
Bit 5	Bit 4	Bit 3	Input 1	Input 0	Output
0	0	0	LOW	LOW	LOW
0	0	1	LOW	LOW	HIGH
0	1	0	LOW	HIGH	LOW
0	1	1	LOW	HIGH	HIGH
1	0	0	HIGH	LOW	LOW
1	0	1	HIGH	LOW	HIGH
1	1	0	HIGH	HIGH	LOW
1	1	1	HIGH	HIGH	HIGH

### Set Digital Output Command

The Set Digital Output command is used to set the state of the digital output line. This command requires a data byte. The data byte is used to specify the output state. Bit #3 corresponds to the state of digital output 0. If a bit is a 0 then the output will be set LOW. If a bit is a 1 then the output will be set HIGH. Note: This command ignores Bits 0-2 and 4-7 of the data byte.

Command Syntax

**!{addr}SO{states}**

Response Syntax

no response

Where **{states}** is a byte in which Bit #3 corresponds to the state of Digital Output.

### Set Module Address Command

The Set Module Address command is used to change the address of a 485SPDA. This command requires a data byte. The data byte is used to specify the new address of the unit. The address of a module is stored in non-volatile memory.

Command Syntax

**!{addr}SA{new address}**

Response Syntax

no response

Where **{addr}** is the current address of the module and **{new address}** is a byte representing the new address.

## Set Power-up States Command

The Set Power-up States command is used to set the state of the digital output at power-up. This command requires a data byte. The data byte is used to specify the power-up output state. Bits #3 corresponds to the power-up state of the digital output. If the bit is a 0 then the output will be set LOW at power-up. If the bit is a 1 then the output will be set HIGH at power-up. Refer to Table 3.5. Note: This command ignores Bits 0-2 and 4-7 of the data byte.

Command Syntax

**!{addr}SS{states}**

Response Syntax

no response

Where **{states}** is a byte in which Bit #3 corresponds to the power-up state of Digital Output.

## Set Turn-around Delay Command

The Set Turn-around Delay command sets the amount of time the 485SPDA waits to respond after executing a command. To ensure that two transmitters are not enabled at the same time, this delay is necessary when two RS-485 transmitters are sharing a pair of wires. Refer to Appendix A for more information on RS-422/485. The turn-around delay is stored in non-volatile memory. This command requires a data byte. The data byte is used to specify the turn-around delay. One unit of turn-around delay is equal to the transmission time of one character. This transmission time can be computed as follows:

$$\text{time} = (1 / \text{baud rate}) * 10$$

Command Syntax

**!{addr}SC{delay}**

Response Syntax

no response

Where **{delay}** is a byte used to specify the turn-around delay.

## Read Module Configuration Command

The Read Module Configuration Command reads the 485SPDA's address, power-up states, and turn-around delay (in that order).

Command Syntax

**!{addr}RC**

Response Syntax

**{address}{states}{delay}**

Where **{address}** is a byte representing the module's current address, **{states}** is a byte representing the module's power-up states, and **{delay}** is a byte representing the module's turn-around delay.

## Chapter 4 - A/D

This chapter will deal with manipulating an A/D reading and cover some of the aspects that were not explained in the A/D connections chapter.

### Sampling Rate

The A/D converter has a conversion time around 20 microseconds, however the sampling rate is limited by the serial communications. The actual sampling rate for a single channel is around 120 samples per second (9600). This rate drops to 37 samples per seconds when sampling all of the channels. When reading an A/D input, the 485SPDA takes four readings and returns the average (0.5 and greater are rounded up) of these readings. This averaging helps filter out noise.

### A/D Input Range

The A/D input range on the 485SPDA is 0V to +5VDC. If it is possible for your device to output a voltage that doesn't fall in this range, steps must to taken to insure that the voltage remains between 0 and +5VDC.

### Reference Inputs

The A/D reference inputs set the top and bottom of the data range. A/D Ref. Input - sets the bottom of the data range. A/D Ref. Input + sets the top of the data range. Since these inputs are directly related to the data range, it is important that a precision reference is used. The 485SPDA has a 5VDC +/- 0.5% reference available. The voltage on A/D Ref. Input + must be at least 2.5VDC greater than A/D Ref. Input -. The voltage difference between A/D Ref. Input + and A/D Ref. Input - is referred to as the Reference Range.

Reference Range = (A/D Ref. Input +) - (A/D Ref. Input -)  
Typically A/D Ref. Input - is connected to Analog ground and A/D Ref. Input + is connected to +5VDC. Figure 2.1 in Chapter 2 shows the typical connections for a reference range of 0 to 5VDC.

### Data Range

The data range of the A/D converter is determined by A/D Ref. Input + and A/D Ref. Input -. A/D Ref. Input - sets the bottom of the data range. Any input voltage that is less than or equal the A/D Ref. Input - will be read as a zero. A/D Ref. Input + sets the top of the data range. Any input voltage that is greater than or equal the A/D Ref. Input + will be read as a 4095 (0FFFH). The data range is as follows:

Data Range = (A/D Ref. Input -) to (A/D Ref. Input +)  
 Data Range = 0 to 4095  
 Data Range = 0 to 0FFFH

Figure 4.1 shows the Data Range and A/D Ref Inputs relationship.

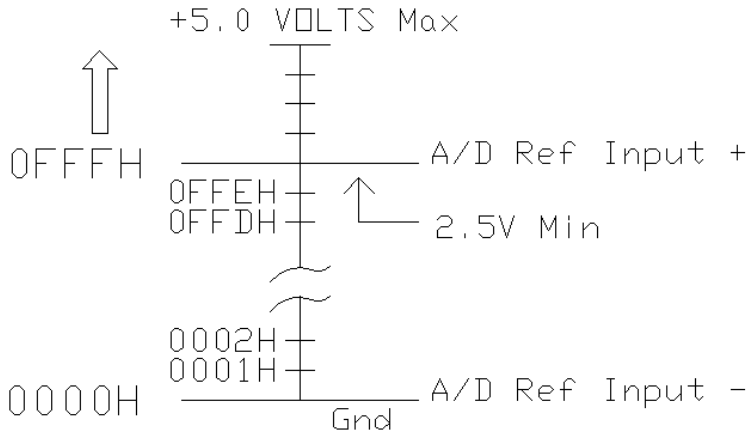


Figure 4.1 - A/D Converter Data Range

## Converting Data

The data read from the 485SPDA A/D converter is directly related to the A/D input channel and the reference range (discussed in previous sections). The 485SPDA has a 12-bit A/D converter. A 12-bit A/D has 4096 possible output values, 0 to 4095 (0 to 0FFFH). These 4096 output values are divided into equal steps over the reference range. The size of each step can be computed as follows:

$$\text{Step size} = (\text{Reference Range}) / 4095$$

The step size is also referred to as the resolution. Once the step size is known, all that is needed to determine the voltage of an A/D input is the number of steps. The data returned from the 485SPDA is the number of steps. The voltage at the A/D inputs can be calculated as follows:

$$\text{Voltage} = (\# \text{ of steps}) * (\text{Step size})$$

**Example 4.1** - Assume: A/D Ref. Input + = 5.0 VDC and A/D Ref.Input - = 0 VDC.

Therefore:

$$\text{Reference Range} = (\text{A/D Ref. Input } +) - (\text{A/D Ref. Input } -)$$

$$\text{Reference Range} = (5.0 \text{ VDC}) - (0 \text{ VDC})$$

$$\text{Reference Range} = 5.0 \text{ VDC}$$

$$\text{Step size} = (\text{Reference Range}) / 4095$$

$$\text{Step size} = (5.0 \text{ VDC}) / 4095$$

$$\text{Step size} = 1.221 \text{ millivolts}$$

Condition #1: A/D reading = 4095 (0FFFH)

$$\text{A/D voltage} = \text{reading} * \text{step size}$$

$$\text{A/D voltage} = 4095 * 1.221 \text{ millivolts}$$

$$\text{A/D voltage} = 5.0 \text{ Volts}$$

Condition #2: A/D reading = 0

$$\text{A/D voltage} = 0 * 1.221 \text{ millivolts}$$

$$\text{A/D voltage} = 0 \text{ Volts}$$

Condition #3: A/D reading = 675 (2A3H)

$$\text{A/D voltage} = 675 * 1.221 \text{ millivolts}$$

$$\text{A/D voltage} = 0.8242 \text{ Volts}$$

## Chapter 5 - D/A

This chapter will deal with converting a digital value into an analog voltage and some related aspects that were not explained in the D/A connections chapter and the Commands chapter.

### D/A Output Range

The four D/A channels have two different ranges. A multiplier bit is used to select between the two ranges. The first range includes voltages from 0V to 3.75V. The second voltage range includes voltages from 0V to 4.3V, but the resolution of the second range is not as good as the first range. The multiplier bit is discussed in more detail in the **Multiplier Bit** section. The value 3.75V mentioned above can vary slightly depending on the D/A converter. A calibration procedure is covered later in this chapter.

### D/A Reference Inputs

The D/A References set the upper limit of the voltage ranges for each D/A channel. D/A Ref 0 is internally fixed at 5V. A voltage between 0V and 5V can be applied to D/A Ref. 1-3 (I/O pins 14-16). Although up to 5V can be applied to pins 14-16, the maximum value used for D/A Ref. 0-3 in the conversion equation is 3.75V. Table 5.1 compares values of D/A Ref. to actual voltage on I/O pins 14-16, and shows the maximum analog output voltage available in both voltage ranges.

Table 5.1 - D/A Ref., Actual Reference Voltage, and Maximum Output Voltage

Voltage Applied to pins 14-16	Value of D/A Ref. used in Conversion Equation	Maximum D/A Output Bit A5 = 0 x1	Maximum D/A Output Bit A5 = 1 x2
0V	0V	0V (FFh)	0V (FFh)
1V	1V	1V (FFh)	1.99V (FFh)
2V	2V	1.99V (FFh)	3.98V (FFh)
2.5V	2.5V	2.49V (FFh)	4.3V
3V	3V	2.99V (FFh)	4.3V
3.75V	3.75V	3.75V (FFh)	4.3V
4V	3.75V	3.75V (FFh)	4.3V
5V	3.75V	3.75V (FFh)	4.3V

Note: Bit A5 is the multiplier bit which selects one of two ranges.

Note: When Bit A5 is 1, the output is limited to 4.3V because of device limitations.

Note: The values in the last column were calculated using the conversion equation, and corrected for device limitations.

## Digital Information Format

The information needed to select a D/A channel, the multiplier, and the output analog voltage is included in 2 bytes. Table 5.2 shows the bit assignments for these two bytes. Bits A7-A6 of byte1 select the D/A channel. Table 5.3 shows the channel selection based on the bit combinations. Bit A5 of byte1 multiplies the analog voltage by 1 or 2. Bits A4-A0 of the first byte and bits B7-B5 of byte2 represent the 8-bit analog voltage to be placed on the selected D/A channel. Bit A4 is the most significant bit and bit B5 is the least significant bit of the analog voltage. Bits B4-B0 of byte2 are ignored.

**Table 5.2 - Bit Assignments**

Byte 1								Byte 2							
Ch. select		8-bit voltage data								Ignored bits					
A	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Note: A5 is the Multiplier Bit.

**Table 5.3 - Channel Selection**

A7	A6	D/A Channel
0	0	D/A Output 0
0	1	D/A Output 1
1	0	D/A Output 2
1	1	D/A Output 3

## Multiplier Bit

Now, the multiplier bit will be covered in more detail. Bit A5 is used as a multiplier. When A5 is 0, the output analog voltage is multiplied by 1. When A5 is 1, the output analog voltage is multiplied by 2. This can be seen more clearly in the conversion equation. This multiplier is useful when analog output voltages must be greater than 3.75V. Setting bit A5 to 1 increases the maximum analog output voltage to 4.3V, but the resolution decreases, and the total error increases. The demo program is set up so that bit A5 is 0 when the output voltage is less than 3.75V, and bit A5 is set to 1 when the output voltage is greater than 3.75V. See Table 5.1 for the maximum analog output voltage for each range.

## Calibrating D/A Ref 0-3

As stated earlier, the value for D/A Ref 0-3 can vary slightly depending on the D/A converter. To calibrate for this error the following steps can be used:

1. Apply 5VDC to D/A Ref 1-3 (I/O pins 14-16).
2. Using the demo software, output FFh (255 dec.) to D/A 0-3. The value 255 is the Digital Code.
3. Measure the voltages on I/O pins 22-25 with a digital multimeter or the A/D channels. These values are the analog voltages for each D/A channel.
4. Using these values and A5=0, solve the Voltage Conversion Equation below for D/A Ref 0-3. These calculated values are the maximum values for D/A Ref 0-3.
5. The calculated value for D/A Ref 0 will always be used in the conversion equation for D/A 0.
6. If the voltage on pins 14-16 is greater then the calculated values of D/A Ref 1-3, then the calculated values need to be used in the conversion equation. If the voltages on pins 14-16 is less than the calculated D/A Ref 1-3, then the voltages on pins 14-16 are used in the conversion equation for each D/A channel.

Usually, the maximum value for D/A Ref 0-3 is between 3.75V and 3.84V. If you have a 485SPDA module, you do not need to calibrate D/A 0, since it is used internally to control the 4-20mA current loop

## Converting Data

The four D/A channels take an 8-bit digital voltage and produce an analog voltage that is placed on I/O pins 22-25.

Data Range = 0 to 255 decimal (00H to FFH)

In this data range, 00H will produce an output analog voltage of 0V, and FFH will produce an output voltage equal to D/A Ref.(max = 3.75V) or 2 x D/A Ref.(max = 4.3V) of the selected channel.

The following equation is used to calculate the analog output voltage.

**Voltage Conversion Equation:**

$\text{Analog Voltage} = (\text{D/A Ref 0-3}) \times (\text{Digital Code}) \times (1 + \text{A5}) / 256$
--

1. Analog Voltage is the voltage on D/A 0-3 (I/O pins 22-25)
2. Digital Code is the digital representation of the analog voltage converted to a decimal value (0 to 255).
3. The value 256 is the total number of steps available in an 8-bit D/A converter.
4. A5 is the multiplier bit.
5. D/A Ref 0-3 is the reference voltage for each of the four D/A channels. The maximum value for D/A Ref 0-3 is usually between 3.77V and 3.84V

**Note: If D/A Ref. 0-3 is 3.75V and bit A5 is 1, the conversion equation implies that voltages greater than 4.3V can be obtained. This is not physically possible. The 485SPDA is limited to a maximum analog output of 4.3V because of the electrical characteristics of the D/A converter.**

Table 5.4 shows several conversion equation examples.

**Table 5.4 - Voltage Conversion Examples**

Digital Code								Analog Voltage
A 4	A 3	A 2	A 1	A 0	B 7	B 6	B 5	
0	0	0	0	0	0	0	0	GND
0	0	0	0	0	0	0	1	$(1/256) \times (\text{D/A Ref}) \times (1 + \text{A5})$
-	-	-	-	-	-	-	-	-
0	1	1	1	1	1	1	1	$(127/256) \times (\text{D/A Ref}) \times (1 + \text{A5})$
1	0	0	0	0	0	0	0	$(128/256) \times (\text{D/A Ref}) \times (1 + \text{A5})$
-	-	-	-	-	-	-	-	-
1	1	1	1	1	1	1	1	$(255/256) \times (\text{D/A Ref}) \times (1 + \text{A5})$

## Chapter 6 - 4-20mA Current Loop Output (485SPDACL only!)

This chapter will cover the operation of the 4-20mA current loop output in the 485SPDACL. If you have a 485SPDA module, you can ignore this chapter.

### Current Loop Output Range

Controlling the 4-20mA current loop output is actually an extension of outputting an analog voltage. D/A 0 is used to control the current in the loop. A direct relationship exists between D/A 0 and the current in the loop. When D/A 0 is 0V (00h), the current in the loop is 4mA. When D/A 0 is 2V (FFh), the current in the loop is 20mA. **D/A 0 is used internally to control the 4-20mA current loop output. D/A 0 is not connected to an I/O pin!**

### Current Loop Reference

As stated above, D/A 0 is used to control the current loop output. The reference for D/A 0 is internally set to 2V. This means that D/A 0 can produce a voltage between 0V (00h) and 2V (FFh), but you do not have access to this D/A channel!

### Digital Information Format

The control of the 4-20mA current loop output is very similar to the process of outputting an analog voltage. Two data bytes are needed to control the current in the 4-20mA current loop. Figure 6.1 shows the two data bytes. Bits A7, A6, and A5 must be **zeros** when controlling the current in the current loop. Bits A4-A0 and bits B7-B5 correspond to the current in the loop, and bits B4-B0 are ignored.

Figure 6.1 - Current Control Data Bytes

Ch. select		Byte 1								Byte 2					
		8-bit current data								Ignored bits					
A	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0									x	x	x	x	x

1. x denotes don't cares
2. A4-A0 and B7-B5 can be 0 or 1.

## Converting Data

The 485SPDACL takes an 8-bit digital value and produces an analog voltage using D/A 0 which is used to control the current in the loop.

Data Range = 0 to 255 decimal (00h to FFh)

In this data range 00h will produce an output current of 4mA, and FFh will produce an output current of 20mA.

The following equation is used to calculate the current in the loop.

### Current Conversion Equation:

$$\text{Loop Current} = 4\text{mA} + (\text{Digital Code} / 256) \times 16\text{mA}$$

1. Digital Code is the digital representation of the current converted to a decimal value.
2. The number 256 represents the total number of steps in the data range.

Table 6.2 shows some current conversion equation examples.

**Table 6.2 - Current Conversion Examples**

Digital Code								Equation	Current (mA)
A 4	A 3	A 2	A 1	A 0	B 7	B 6	B 5		
0	0	0	0	0	0	0	0	GND	4.00
0	0	0	0	0	0	0	1	$((1/256) \times 16) + 4$	4.06
-	-	-	-	-	-	-	-	-	-
0	1	1	1	1	1	1	1	$((127/256) \times 16) + 4$	11.94
1	0	0	0	0	0	0	0	$((128/256) \times 16) + 4$	12.00
-	-	-	-	-	-	-	-	-	-
1	1	1	1	1	1	1	1	$((255/256) \times 16) + 4$	19.94

## Chapter 7- Software

This section covers programming techniques such as constructing a command string, receiving data and manipulating data. The various steps and examples are shown in QuickBASIC. If you are programming in another language, this section can be used as a guideline for programming the 485SDPA and 485SPDACL.

### Read A/D Command

The Read A/D channels command returns two bytes for each channel read. The two bytes represent the most significant byte (MSB) and least significant byte (LSB) of the reading. The MSB is received first, followed by the LSB. This command requires a data byte. The data byte is used to specify the number of the highest channel to be read. All channels less than this channel will be read as well.

Step 1 - Constructing the command string:

```
Command$ = "!" + CHR$(addr) + "RA" +  
CHR$(channel)
```

Where *addr* is the module's address and the value of *channel* is equal to the highest channel to be read.

Step 2 - Transmitting the command string:

```
Print #1, Command$
```

Step 3 - Receiving the data:

```
MSB$ = INPUT$(1, #1)  
LSB$ = INPUT$(1, #1)
```

Step 4 - Manipulating the data:

```
reading = (ASC(MSB$) * 256) + ASC(LSB$)
```

The value of *reading* is the result of the A/D conversion.

Step 5 - Repeat Steps 3 & 4 until each channel has been completed.

**Example 6.1: Read A/D channels 0 and 1 of a module with an address of 5.**

```
channel = 1
addr = 5
Command$ = "!" + CHR$(addr) + "RA" +
CHR$(channel)
Print #1, Command$

'Get the value of channel 1
MSB$ = INPUT$(1, #1)
LSB$ = INPUT$(1, #1)
reading1 = (ASC(MSB$) * 256) + ASC(LSB$)

'Get the value of channel 0
MSB$ = INPUT$(1, #1)
LSB$ = INPUT$(1, #1)
reading0 = (ASC(MSB$) * 256) + ASC(LSB$)
```

The value of *reading0* is the result of the A/D conversion on channel 0 and the value of *reading1* is the result of the A/D conversion on channel 1.

## Read Digital I/O Command

The Read Digital I/O command returns a byte which represents the states of the 2 digital inputs and the digital output. Bit 3 corresponds to the state of digital output. Bits 4-5 correspond to the states of digital inputs 0-1. If a bit is a 0 then the digital state of that digital I/O is LOW. If a bit is a 1 then the digital state of the I/O is HIGH.

Step 1 - Constructing the command string:

```
Command$ = "!" + CHR$(addr) + "RD"
```

Where *addr* is the module's address.

Step 2 - Transmitting the command string:

```
Print #1, Command$
```

Step 3 - Receiving the data:

```
Reply$ = INPUT$(1, #1)
```

Step 4 - Manipulating the data:

```
states = ASC(Reply$)
```

Step 5 - Determining an I/O's status

```
status = states AND mask
```

By "ANDing" the value of *states* with the appropriate *mask* of an I/O line, the *status* of can be determined. If *status* is equal to zero, then the I/O line is LOW. If *status* is not equal to zero, then the I/O line is HIGH. Table 5.1 shows the *mask* values for each I/O.

Step 6 - Repeat Step 5 until the status of each I/O has been determined.

**Table 5.1 - Digital I/O Mask Values**

I/O Line	Mask Values	
	Hexadecimal	Decimal
Digital Output #0	8H	8
Digital Input #0	10H	16
Digital Input #1	20H	32

**Example 5.2 - Determining the status of Digital Input #0 of the module with an address of 10.**

```
mask = &H10  
addr = 10  
Command$ = "!" + CHR$(addr) + "RD"  
Print #1, Command$  
Reply$ = INPUT$(1, #1)  
states = ASC (Reply$)  
status = states AND mask
```

If *status* is equal to zero than Digital Input #1 is LOW. If *status* is not equal to zero than Digital Input #1 is HIGH.

## Set Digital Output State

The Set Digital Output command is used to set the state of the digital output line. This command requires a data byte. The data byte is used to specify the output state. Bit 3 corresponds to the state of digital output. If a bit is a 0 then the output will be set LOW. If a bit is a 1 then the output will be set HIGH. Note: This command ignores Bits 0-2 and 4-7 of the data byte.

Step 1a - Constructing the command string:

```
` Set Appropriate Outputs HIGH
states = states OR mask
```

By "ORing" the current *states* with the appropriate *mask* of a digital output (given in Table 5.1), the output's data bit will be set to a "1" (which will be set HIGH).

Step 1b - Set Appropriate Outputs LOW

```
states = states AND (NOT(mask))
```

By "ANDing" the current *states* with the complement of the appropriate *mask* of a digital output (given in Table 5.1), the output's data bit will be set to a "0" (which will be set LOW).

Step 1c - Construct the string

```
Command$ = "!" + CHR$(addr) + "SO" +
CHR$(states)
```

Where *addr* is the module's address.

Step 2 - Transmitting the command string:

```
Print #1, Command$
```

## Set Module Address

The Set Module Address command is used to change the address of a 485SPDA. This command requires a data byte. The data byte is used to specify the new address of the unit.

Step 1a - Constructing the command string:

```
Command$ = "!" + CHR$(addr) + "SA" +  
CHR$(newaddr)
```

Where *addr* is the module's current address and *newaddr* is the module's new address.

Step 2 - Transmitting the command string:

```
Print #1, Command$
```

**Example 5.4 - Change the address of module (at current address 5) to 10.**

```
addr = 5  
newaddr = 10  
Command$ = "!" + CHR$(addr) + "SA" +  
CHR$(newaddr)  
Print #1, Command$
```

The module whose previous address was 5 will now have an address of 10.

## Set Power-up States

The Set Power-up States command is used to set the states of the digital outputs at power-up. This command requires a data byte. The data byte is used to specify the power-up output states. Bit 3 corresponds to the power-up state of the digital outputs. If a bit is a 0 then the output will be set LOW at power-up. If a bit is a 1 then the output will be set HIGH at power-up. Note: This command ignores Bits 0-2 and 4-7 of the data byte.

Step 1a - Constructing the command string:

```
`Set Appropriate Outputs HIGH  
states = states OR mask
```

By "ORing" the current *states* with the appropriate *mask* of a digital output (given in Table 5.1), the output's data bit will be set to a "1" (which will be set HIGH).

Step 1b - Set Appropriate Outputs LOW

```
states = states AND (NOT(mask))
```

By “ANDing” the current *states* with the complement of the appropriate *mask* of a digital output (given in Table 5.1), the output’s data bit will be set to a “0” (which will be set LOW).

### Step1c - Construct the string

```
Command$ = "!" + CHR$(addr) + "SS" +  
CHR$(states)
```

Where *addr* is the module’s address.

### Step 2 - Transmitting the command string:

```
Print #1, Command$
```

### Example 5.5 - Set Digital Output #0 HIGH of the module with an address of 5.

```
addr = 5  
states = 0  
mask = &H08
```

```
` Set bit 0 of states to make  
` Digital Output #0 HIGH  
states = states OR mask
```

```
Command$ = "!" + CHR$(addr) + "SS" +  
CHR$(states)  
Print #1, Command$
```

At power-up: Digital Output #0 will be set HIGH.

## Set Turn-Around Delay

The Set Turn-Around Delay command is used to set the amount of time the 485SPDA / 485SPDACL waits to respond after executing a command. To ensure that two drivers are not enabled at the same time, this delay is necessary when two drivers are sharing a pair of wires. The turn-around delay is stored in non-volatile memory. This command requires a data byte. The data byte is used to specify the turn-around delay. One unit of turn-around is equal to one character transmission time.

Step 1a - Constructing the command string:

```
Command$ = "!" + CHR$(addr) + "SC" +  
CHR$(delay)
```

Where *addr* is the module's current address and *delay* is the turn-around delay. (Refer to this command in Chapter 3 for more information).

Step 2 - Transmitting the command string:

```
Print #1, Command$
```

**Example 5.6 - Set the turn-around delay to 100 character transmission time of the module with an address of 5.**

```
addr = 5  
delay = 100  
Command$ = "!" + CHR$(addr) + "SC" +  
CHR$(delay)  
Print #1, Command$
```

The module at address 5 will now have a turn-around delay of 100 character transmission times.

## Read Module Configuration Command

The Read Module Configuration Command reads the 485SPDA's address, power-up states, and turn-around delay (in that order).

Step 1 - Constructing the command string:

```
Command$ = "!" + CHR$(addr) + "RC"
```

Where *addr* is the module's address.

Step 2 - Transmitting the command string:

```
Print #1, Command$
```

Step 3 - Receiving the address data:

```
Reply$ = INPUT$ (1, #1)
```

Step 4 - Determine the address:

```
address = ASC(Reply$)
```

Step 5 - Receiving the power-up states data:

```
Reply$ = INPUT$ (1, #1)
```

Step 6 - Determine the power-up states:

```
states = ASC(Reply$)
```

Refer to Read Digital I/O earlier in this chapter for the steps to determine the status of individual output states.

Step 7 - Receiving the turn-around delay data:

```
Reply$ = INPUT$ (1, #1)
```

Step 8 - Determine the turn-around delay:

```
delay = ASC(Reply$)
```

**Example 5.7 - Determining the configuration of the module with an address of 10.**

```
addr = 10
Command$ = "!" + CHR$(addr) + "RC"
Print #1, Command$

` Determine the module's address
Reply$ = INPUT$ (1, #1)
address = ASC (Reply$)

` Determine the module's power-up states
Reply$ = INPUT$ (1, #1)
states = ASC (Reply$)

` Determine the module's turn-around delay
Reply$ = INPUT$ (1, #1)
delay = ASC (Reply$)
```

## Chapter 8: API Functions

The application program interface (API) is a set of functions that make it easy for a DOS-based program to interface with the 485SPDA and 485SPDACL modules without knowing the low level details involved in communicating with each module. These functions can be linked to programs written in Borland / Turbo C++, Borland / Turbo Pascal and Microsoft QuickBASIC v4.5.

### Compiling and Linking

#### Borland / Turbo C++

In order to use the API routines with Borland / Turbo C++, you must include the following line at the beginning of your program:

```
#include "b485spda.h"
```

To compile and link a program with Borland / Turbo C++, type the following at the DOS prompt:

```
bcc -ml yourprog.c b485spda.lib ↵
```

#### Borland / Turbo Pascal

In order to use the API routines with Borland / Turbo Pascal, you must include the following lines at the beginning of your program:

```
{$M 4000, 0, 64000}  
uses b485spda;
```

The first line tells the compiler to leave some memory free for the serial communication routines that the API uses to communicate with the module. The second line includes the API routines in your program. To compile and link your program with Borland Pascal or Turbo Pascal, type the following at the DOS prompt:

```
tpc /B+ yourprog.pas ↵
```

#### Microsoft® QuickBASIC

In order to use the API routines with QuickBASIC, you must include the following lines at the beginning of your program:

```
`$include: `b485spda.bi'  
mem = SETMEM(-2000)
```

The first line includes the API routines in your program. The second line tells the compiler to leave some memory free for the serial communication routines that the API uses to communicate with the module. To compile and link a program with Microsoft® QuickBASIC, type the following at the DOS prompt:

```
bc yourprog.bas /D /O /T /C:512; ↵
```

```
link yourprog.obj, yourprog.exe, nul.map,  
bcom45.lib+b485spda.lib /EX /NOE /NOD:brun45.lib;  
↵
```

## Function Reference

The API functions that must be called before any other API functions are used are `initComPort` and `spda_init`. When using Pascal, you must call `spda_startup` before calling `spda_init`. Before exiting the application, the functions, `spda_deinit` and `deinitComPort` must be called. The serial communication routines that the API functions use installs an interrupt service routine and exiting a program without calling `deinitComPort` may cause unpredictable results.

### deinitComPort

---

**Purpose:** Initialize a serial communications port.

**Syntax:**

C: `void pascal far deinitComPort(WORD hComDev);`

Pascal: `procedure deinitComPort(hComDev : word);`

BASIC: `sub deinitComPort(byval hComDev%)`

**Remarks:** *hComDev* is the handle to the serial communications port which is returned from the function, `initComPort`.

**Returns:** Nothing.

**See Also:** `initComPort`

### initComPort

---

**Purpose:** Initialize a serial communications port.

**Syntax:**

C: `WORD pascal far initComPort(WORD addr, BYTE irq, long baud);`

Pascal: `function initComPort(addr : word; irq : byte; baud: longint) : word;`

BASIC: `function initComPort%(byval addr%, byval`

*irq%*, *byval baud%*)

**Remarks:** *addr* is that serial port address. *irq* is the interrupt request number that the serial port uses. The table below shows the common port addresses and *irq*'s.

Name	Address	IRQ
COM1	03F8h	4
COM2	02F8h	3
COM3	03E8h	4
COM4	02E8h	3

*baud* is the baud rate that will be used to communicate with the module. The valid values for *baud* are from 2 to 115200; however, the 485SPDA and 485SPDACL only allow baud rates between 1200 to 9600.

**Returns:** This function returns a handle to a serial communications port or -1 if the port could not be opened.

**See Also:** `deinitComPort`

## spda\_get\_module\_type

**Purpose:** Get the type of module.

**Syntax:** C: `int pascal far spda_get_module_type(int handle);`

Pascal: `function spda_get_module_type(handle : integer) : integer;`

BASIC: `function spdaGetModuleType%(byval handle%)`

**Remarks:** *handle* is the handle to the module which is returned from the function, `spda_init`.

**Returns:** This function returns the module type for the specified handle. 83h (131) is returned for the 485SPDA and 485SPDACL.



## spda\_read\_analog\_inputs

**Purpose:** Read the analog input channels of the module.

**Syntax:** C: `int pascal far spda_read_analog_inputs(WORD handle, WORD channels, WORD far *reply);`

Pascal: `function spda_read_analog_inputs(handle : word; channels : word; var reply : WordArray) : integer;`

BASIC: `function spdaReadAnalogInputs%(byval handle%, byval channels%, byval replyOfs%, byval replySeg%)`

**Remarks:** *handle* is the handle to the module which is returned from the function, *spda\_init*. *channels* is a bit mask for the channels to be read. Setting bit 0 of *channels* causes A/D channel 0 to be read, setting bit 1 causes A/D channel 1 to be read, etc.

Bit	7	6	5	4	3	2	1	0
Value	80h (128)	40h (64)	20h (32)	10h (16)	08h (8)	04h (4)	02h (2)	01h (1)
A/D	-	AD6	AD5	AD4	AD3	AD2	AD1	AD0

*reply* is a pointer to an array of seven 16-bit values where the analog values will be stored.

## spda\_read\_config

**Purpose:** Set the module's configuration. Note that this command is for the 485SPDA12 and will not work on the 232SPDA12.

**Syntax:** C: `int pascal far spda_read_config(int handle, WORD far *reply);`

Pascal: `function spda_read_config(handle : integer; var reply : WordArray) : integer;`

BASIC: `function spdaReadConfig%(byval handle%, byval replyOfs%, byval replySeg%)`

**Remarks:** *handle* is the handle to the module which is returned from the function, *spda\_init*. *reply* is a pointer to an array of three 16-bit values where the configuration values will be stored. The first element of *reply* is the module address, the second is the power-up states and the third is the turn-around delay.

**Returns:** This function returns zero if an error occurred, otherwise it

returns a non-zero value.

**See Also:** `spda_set_module_address`,  
`spda_set_power_up_states`,  
`spda_set_turn_around_delay`

## spda\_read\_digital\_io

---

**Purpose:** Read the state of the digital I/O lines.

**Syntax:** C: `WORD pascal far spda_read_digital_io(int handle);`

Pascal: `function spda_read_digital_io(handle : integer) : word;`

BASIC: `function spdaReadDigitalIO%(byval handle%)`

**Remarks:** *handle* is the handle to the module which is returned from the function, `spda_init`.

**Returns:** This function returns the digital I/O states. If a bit is set, then the corresponding I/O line is on. The table below describes the mapping of the bits that are returned.

Bit	7	6	5	4	3	2	1	0
Value	80h (128)	40h (64)	20h (32)	10h (16)	08h (8)	04h (4)	02h (2)	01h (1)
I/O	-	-	IN1	IN0	OUT0	-	-	-

**See Also:** `spda_set_digital_outputs`

## spda\_set\_analog\_outputs

---

**Purpose:** Set the analog output levels.

**Syntax:** C: `void pascal far spda_set_analog_outputs(int handle, WORD channel, WORD value);`

Pascal: `procedure spda_set_analog_outputs(handle : integer; channel : word; value : word);`

BASIC: `sub spdaSetAnalogOutputs(byval handle%, byval channel%, byval value%)`

**Remarks:** *handle* is the handle to the module which is returned from the function, `spda_init`. *channel* is the analog output channel to set. *value* is the digital value used to set the analog output. The output voltage for values less than 256 is given by:

$$\text{volts} = \frac{\text{value} \times \text{vref}_{\text{channel}}}{255}$$

and for values greater than 255, it is given by:

$$\text{volts} = \frac{2 \times \text{int}(\text{value} / 2) \times \text{vref}_{\text{channel}}}{255}$$

**Returns:** Nothing.

## spda\_set\_digital\_outputs

---

**Purpose:** Turn the digital output off or on.

**Syntax:** C: `void pascal far spda_set_digital_outputs(int handle, BYTE state);`

Pascal: `procedure spda_set_digital_outputs(handle : integer; state : byte);`

BASIC: `sub spdaSetDigitalOutputs(byval handle%, byval state%)`

**Remarks:** *handle* is the handle to the module which is returned from the function, `spda_init`. *state* is the new state of the output. If *state* is zero, then the output is turned off, otherwise it is turned on.

**Returns:** Nothing.

**See Also:** `spda_read_digital_io`

## spda\_set\_error\_detection

---

**Purpose:** This function sets the communication mode that is used with the 232SPDA12 and 485SPDA12 modules. If error detection is on, extra bytes are sent as part of the command message and the reply to help identify communications problems.

**Syntax:** C: `void pascal far spda_set_error_detection(int handle, BYTE state);`

Pascal: `procedure spda_set_error_detection(handle : integer; state : byte);`

BASIC: `sub spdaSetErrorDetection(byval handle%, byval state%)`

**Remarks:** *handle* is the handle to the module which is returned from the function, *spda\_init*. *state* is the state of the error detection. If *state* is zero, then error detection mode is off, otherwise it is on.

**Returns:** Nothing.

## spda\_set\_module\_address

---

**Purpose:** This function changes the module's address. Note that this command is for the 485SPDA12 and will not work on the 232SPDA12.

**Syntax:** C: `void spda_set_module_address(int handle, BYTE newAddr);`

Pascal: `procedure spda_set_module_address(handle : integer; newAddr : byte);`

BASIC: `sub spdaSetModuleAddress(byval handle%, byval newAddr%)`

**Remarks:** *handle* is the handle to the module which is returned from the function, *spda\_init*. *newAddr* is the new module address. The valid values for *newAddr* are 0 through FFh (255).

**Returns:** Nothing.

**See Also:** `spda_read_config`

## spda\_set\_power\_up\_states

---

**Purpose:** This function changes the default state of the digital outputs when the module is powered up. Note that this command is for the 485SPDA12 and will not work on the 232SPDA12.

**Syntax:**

C: `void spda_set_power_up_states(int handle, WORD state);`

Pascal: `procedure spda_set_power_up_states(handle : integer; state : word);`

BASIC: `sub spdaSetPowerUpStates(byval handle%, byval state%)`

**Remarks:** *handle* is the handle to the module which is returned from the function, `spda_init`. *state* is the new power up state. If *state* is zero, then the digital output is initially off after power up, otherwise it is initially on.

**Returns:** Nothing.

**See Also:** `spda_read_config`

## spda\_set\_turn\_around\_delay

---

**Purpose:** Set the turn around delay. Note that this command is for the 485SPDA12 and will not work on the 232SPDA12.

**Syntax:**

C: `void pascal far spda_set_turn_around_delay(int handle, BYTE delay);`

Pascal: `procedure spda_set_turn_around_delay(handle : integer; delay : byte);`

BASIC: `spdaSetTurnAroundDelay(byval handle%, byval delay%)`

**Remarks:** *handle* is the handle to the module which is returned from the function, `spda_init`. *delay* is the value of the turn-around delay. The amount of time for the turn-around delay is calculated by:

$$seconds = \frac{10 \times delay}{baudrate}$$

**Returns:** Nothing.

**See Also:** `spda_read_config`

## **spda\_startup**

---

**Purpose:** Initialize internal variables for the API routines.

**Syntax:** C: n/a  
Pascal: `procedure spda_startup;`  
BASIC: n/a

**Remarks:** For Pascal, this function must be called before any other API functions, because it initializes the variables that the API uses.

# Appendix A: Decimal to HEX to ASCII Table

DECIMAL to HEX to ASCII CONVERSION TABLE												
DEC	HEX	ASCII	KEY	DEC	HEX	ASCII	DEC	HEX	ASCII	DEC	HEX	ASCII
0	0	NUL	ctrl @	32	20	SP	64	40	@	96	60	`
1	1	SOH	ctrl A	33	21	!	65	41	A	97	61	a
2	2	STX	ctrl B	34	22	“	66	42	B	98	62	b
3	3	ETX	ctrl C	35	23	#	67	43	C	99	63	c
4	4	EOT	ctrl D	36	24	\$	68	44	D	100	64	d
5	5	ENQ	ctrl E	37	25	%	69	45	E	101	65	e
6	6	ACK	ctrl F	38	26	&	70	46	F	102	66	f
7	7	BEL	ctrl G	39	27	'	71	47	G	103	67	g
8	8	BS	ctrl H	40	28	(	72	48	H	104	68	h
9	9	HT	ctrl I	41	29	)	73	49	I	105	69	i
10	A	LF	ctrl J	42	2A	*	74	4A	J	106	6A	j
11	B	VT	ctrl K	43	2B	+	75	4B	K	107	6B	k
12	C	FF	ctrl L	44	2C	,	76	4C	L	108	6C	l
13	D	CR	ctrl M	45	2D	-	77	4D	M	109	6D	m
14	E	SO	ctrl N	46	2E	.	78	4E	N	110	6E	n
15	F	SI	ctrl O	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	ctrl P	48	30	0	80	50	P	112	70	p
17	11	DC1	ctrl Q	49	31	1	81	51	Q	113	71	q
18	12	DC2	ctrl R	50	32	2	82	52	R	114	72	r
19	13	DC3	ctrl S	51	33	3	83	53	S	115	73	s
20	14	DC4	ctrl T	52	34	4	84	54	T	116	74	t
21	15	NAK	ctrl U	53	35	5	85	55	U	117	75	u
22	16	SYN	ctrl V	54	36	6	86	56	V	118	76	v
23	17	ETB	ctrl W	55	37	7	87	57	W	119	77	w
24	18	CAN	ctrl X	56	38	8	88	58	X	120	78	x
25	19	EM	ctrl Y	57	39	9	89	59	Y	121	79	y
26	1A	SUB	ctrl Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	ctrl [	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	ctrl \	60	3C	<	92	5C	\	124	7C	
29	1D	GS	ctrl ]	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	ctrl ^	62	3E	>	94	5E	^	126	7E	~
31	1F	US	ctrl _	63	3F	?	95	5F	_	127	7F	DEL

# Appendix B: Circuit Schematic

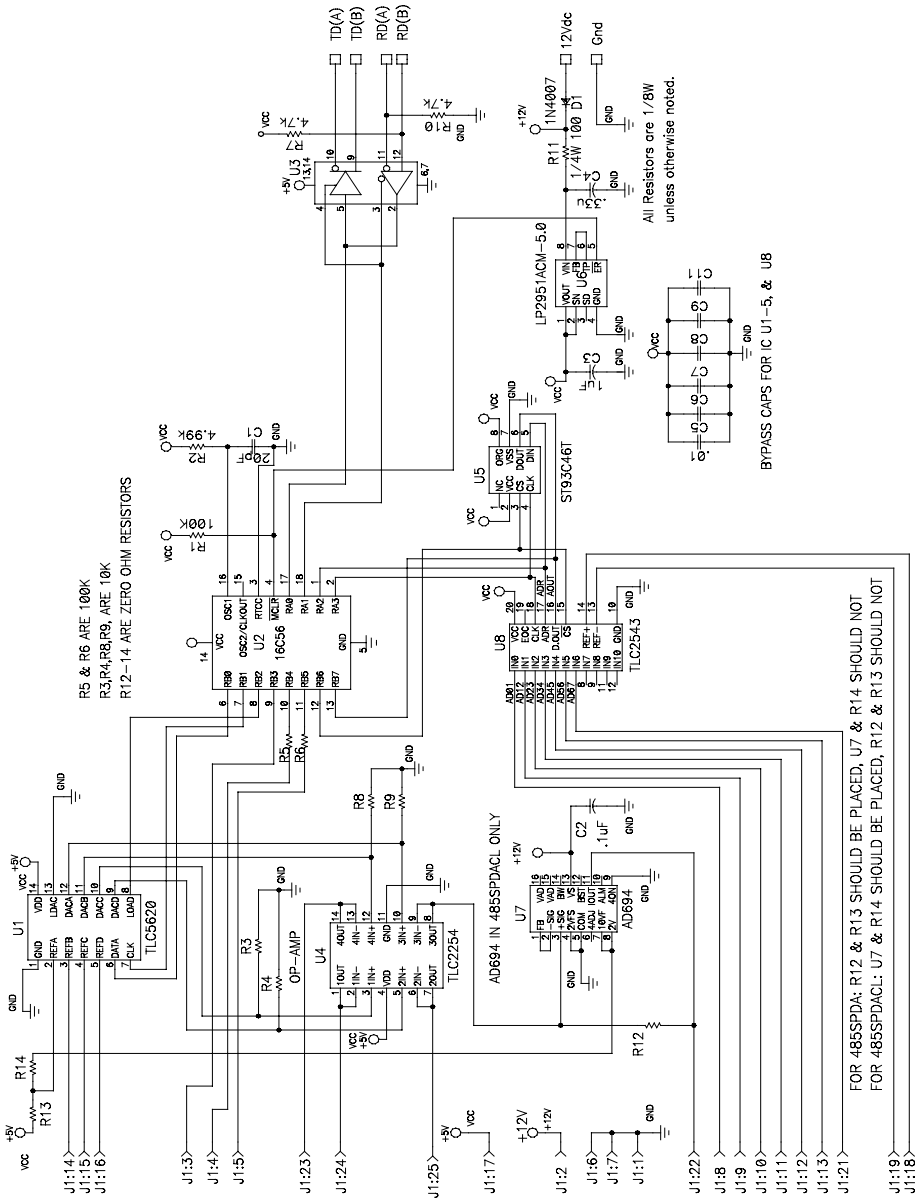


Figure B.1: 485SPDA/485SPDACL Circuit Schematic

## Appendix C: Analog Input Impedance

When interfacing with an A/D converter, it is important that the device you are connecting can drive the A/D input. To determine if your device can drive an A/D input, there are three factors you must consider:

- Output impedance of the device
- Input impedance of A/D
- A/D sampling time

The goal is to have the voltage at the A/D input settle to a voltage close to the output voltage of the device in a time frame that is less than the A/D sampling time (“close to” means a value significantly less than the resolution of the A/D). If the voltage does not settle fast enough, errors will occur in the reading, resulting in a loss of resolution.

The next section, titled “Simplified Analog Input Analysis,” contains information from Texas Instruments data sheet on the TLC1543. The TLC1543 is the A/D converter that is used on the 485SPDA and 485SPDACL. This section provides a simplified calculation which can be used to determine the maximum output impedance the device can have to settle the A/D input to a voltage within ½ LSB.

For the 485SPDA and 485SPDACL:

$$t_c = 100\mu s$$

Using this information:

$$R_s \leq 170k\Omega$$

If the output impedance of your device is 170 kΩ, you should figure an additional error of ½ LSB.

It should be pointed out that **this is a simplified analysis** and there other several other factors that must be considered (pin capacitance, noise immunity, etc.). The data sheet for the TLC1543 states that “The driving source impedance should be less than or equal to 1kΩ.” B&B Electronics recommends placing a voltage follower between the A/D input and any device with output source impedance greater than 1kΩ.

## Simplified Analog Input Analysis

Using the equivalent circuit in Figure C-1, the time required to charge the analog input capacitance from 0 to  $V_s$  within  $\frac{1}{2}$  LSB can be derived as follows:

The capacitance charging voltage is given by

$$V_c = V_s(1 - e^{-t_c/R_t C_i}) \quad (1)$$

where

$$R_t = R_s + r_i$$

The final voltage to  $\frac{1}{2}$  LSB is given by

$$V_c(1/2 \text{ LSB}) = V_s - (V_s/2048) \quad (2)$$

Equating equation 1 to equation 2 and solving for time  $t_c$  gives

$$V_s - (V_s/2048) = V_s(1 - e^{-t_c/R_t C_i}) \quad (3)$$

and

$$t_c(1/2 \text{ LSB}) = R_t \times C_i \times \ln(2048) \quad (4)$$

Therefore, with the values given the time for the analog input signal to settle is

$$t_c(1/2 \text{ LSB}) = (R_s + 1\text{k}\Omega) \times 60\text{pF} \times \ln(2048) \quad (5)$$

This time must be less than the converter sample time shown in the timing diagrams.

$V_i$  = Input Voltage at A0 - A10

$V_s$  = External Driving Source Voltage

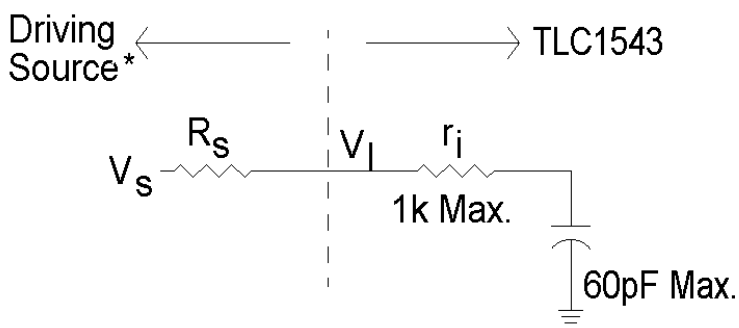
$R_s$  = Source Resistance

$r_i$  = Input Resistance

$C_i$  = Equivalent Input Capacitance

\*Driving source requirements:

- Noise and distortion for the source must be equivalent to the resolution of the converter.
- $R_s$  must be real at the input frequency.



**Figure C-1. Equivalent Input Circuit Including the Driving Source**

## Appendix D: Analog Output Impedance

The load resistance  $R_L$  for the D/A output as shown in Figure D-1 must be greater than or equal to 10 k $\Omega$ . The load capacitance  $C_L$  must be less than or equal to 100 pF. If the device connected to the D/A output does not meet these requirements, B&B Electronics recommends placing a voltage follower similar to the one shown in Figure D-2 between the D/A output and any device with input impedance less than 10 k $\Omega$ . In this circuit  $R_L$  must be greater than or equal to 10 k $\Omega$ .

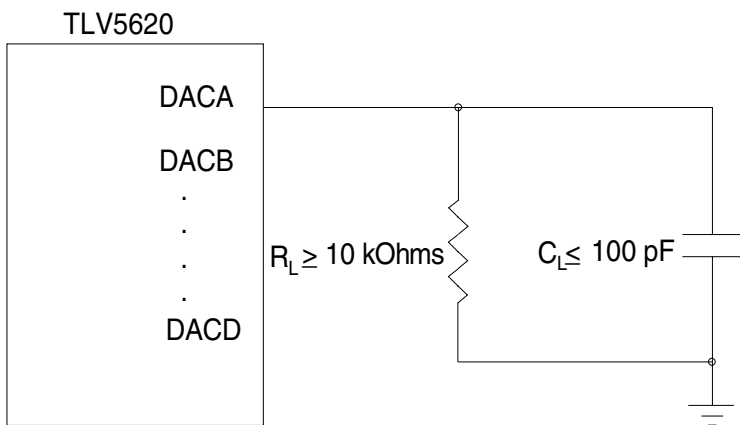


Figure D-1 Maximum D/A Load

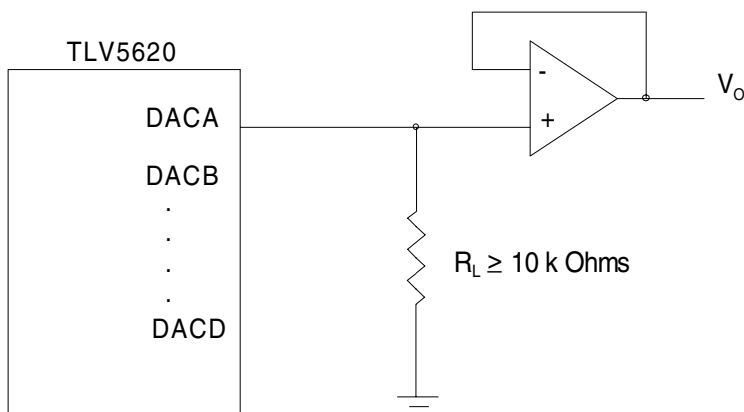


Figure D-2 Voltage Follower Circuit