

RS-232 Digital Relay I/O

Model 232DRIO

Documentation Number 232DRIO1005
pn#4520-r2

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) 433-5104

Internet:

<http://www.bb-elec.com>
sales@bb-elec.com
support@bb.elec.com

© B&B Electronics – January 2008

232DRIO-0308 Manual

Cover Page

B&B Electronics -- PO Box 1040 -- Ottawa, IL 61350
PH (815) 433-5100 -- FAX (815) 433-5104

Table of Contents

Chapter 1- <i>Introduction</i>	3
232DRIO Features	3
Packing List	2
232DRIO Specifications.....	3
Opto-Isolated Input.....	3
Relay Outputs	3
Power Supply	3
Communications	3
Environment	3
Size	3
Chapter 2 - <i>Connections</i>	5
I/O Connections	5
Opto-isolated Input.....	5
Relay Outputs	5
Ground	5
Serial Port Connections	6
Power Supply Connections.....	7
Chapter 3 - <i>Commands</i>	9
Syntax	10
I/O Data Bytes.....	11
Read I/O Lines Command	13
Set Output Lines Command	13
Chapter 4 - <i>I/O Interfacing</i>	15
Opto-isolated Input	15
Relay Outputs.....	17
Chapter 5 - <i>Software</i>	19
Programming Techniques.....	19
Opening a COM Port.....	19
Closing a COM Port.....	19
QuickBasic	20
Read I/O States Command.....	20
Set Output States Command	20
C/C++	21
Read I/O States Command.....	21

Chapter 1- *Introduction*

232DRIO Features

The 232DRIO is a general purpose data acquisition controller that is connected to your computer's RS-232 serial port. The 232DRIO offers 1 opto-isolated input and 2 relay outputs. With these features, the controller can be used to sense external ON/OFF conditions and to control a variety of devices.

The relay outputs are CMOS/TTL compatible. The optically isolated AC/DC input are also CMOS/TTL compatible. For maximum flexibility, both input and outputs are connected with internal screw type terminal blocks. Three LEDs are provided to monitor the status of each of these.

The 232DRIO connects to your computer's RS-232 serial port through a DB-9F connector. The unit communicates at a baud rate of 9600, 8 data bits, no parity, and 1 stop bit. This unit can **not** be powered through the handshake lines because of the higher current requirement to drive the relays. However, the unit may be powered by an external +12Vdc source brought in through the power pin, pin #9 of the serial port. If the 232DRIO cannot be powered using the previous method, it may be powered with +12Vdc @ 100mA through the supplied terminal blocks. Power supply is available.



Figure 1.1 – 232DRIO Module

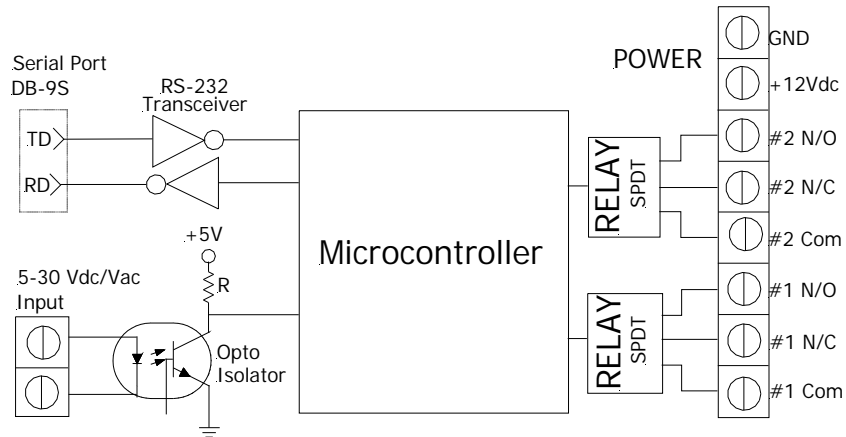


Figure 1.2 – Simplified Block Diagram

Packing List

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

1. 232DRIO unit
2. Software
3. This instruction manual

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

232DRIO Specifications

Non-polarized, Optically Isolated Input

Number of Channels:	1
Indication Mode	
Logic "0":	LED on, digital voltage input high
Logic "1":	LED off, digital voltage input low
Input Electrical Characteristic	
Voltage input low:	<1.5 VAC/VDC
Voltage input high:	5-30 VAC/VDC @ 1mA to 30mA
Isolation Voltage:	2500 V
Leakage Current:	10 μ A max.

Relay Outputs

Number of Channels:	2 electromechanical relays
Factory Default:	Both relays de-energized
Indication Mode	
Logic "0":	LED off, Relays de-energized
Logic "1":	LED on, Relays energized
Relay Ratings	
Contact (standard):	10A @ 120VAC 8A @ 30VDC (resistive)
Max. switching capacity:	1200VA/240W
Max. operating voltage:	250VAC/125VDC
Max. carrying current:	10A (AC), 8A (DC) – standard
Min. permissible load:	100 mA @ 5 VDC
Relay Form:	Form C, single-pole double-throw (SPDT)
Output Terminals:	Normally open (N/O) or Normally close (N/C)
Relay Life (mech.):	10 million operations min.
Relay Life (load dependent):	100 thousand operations min.
Operating Time:	10 msec. max. (mean: ~5.1ms)

Power Supply

Input Voltage:	9-16 VDC
Input Current:	100 mA
Connections:	Blue terminal block field wiring Pin #9 (Power pin) of DB9F

Communications

Standard:	RS-232 (unit is DCE)
Baud Rate:	9600
Format:	8 data bits, 1 stop bit, no parity

Environment

Operating Temperature:	0°-70°C
Operating Humidity:	0-95%, non-condensing
Storage Temperature:	-20°-70°C

Size 4.60"L x 2.40"W x 1.32"H

Chapter 2 - Connections

Do not make any connections to the 232DRIO until you have read this chapter. Remember to power down the unit and any other connected devices before making any kinds of connections. Also be sure that the wires connected to the terminal blocks will support the voltage and current requirements of your external devices.

This chapter will cover the connections for the 232DRIO. There are four sets of connections: opto-isolated input (optional), relay outputs, serial port, and power supply.

I/O Connections

Connections to the I/O lines are made through the internal block terminals. These should clearly be labeled on the board. Refer to Table 2.1. See Chapter 5 for I/O interfacing examples.

Opto-isolated Input (black terminal blocks)

The non-polarized, opto-isolated input line is CMOS/TTL compatible and can handle voltages from 0Vdc/Vac to +30Vdc/Vac. Refer to the Specification for a complete detail.

Relay Outputs (blue terminal blocks)

The relay output lines have a maximum operating voltage of +250Vac/+125Vdc and are CMOS/TTL compatible. Refer to the Specification for a complete detail. **Both relays are de-energized at factory default settings.**

Ground

Connect the ground correspondingly as marked. Do not interchange the grounds for input and power. Otherwise, the input will not be optically isolated. The input is non-polarized so the input ground could be connected to either one of the black terminal while the power ground must be connected as marked on the blue terminal.

Table 2.1 - 232DRIO Terminal Blocks

Blue Pin #	Function	Black Pin #	Function
1	- Ground Input Power	1	Opto-Isolated Input
2	+12Vdc Input Power	2	Opto-Isolated Input
3	#2 N/C-Normally Closed		Non-polarized
4	#2 N/O-Normally Opened		0-30 Vdc/Vac
5	#2 C-Common		
6	#1 C-Common		
7	#1 N/C-Normally Closed		
8	#1 N/O-Normally Opened		

Serial Port Connections

In order to communicate to the 232DRIO module it must be connected to an RS-232 serial port. The unit is designed for a baud rate of 9600, so the serial port must be set for this rate. A data format of 8 data bits, 1 stop bit and no parity is used. The 232DRIO is configured as a DCE device (See Table 2.2). If your communications equipment is configured as a DTE device, such as a standard IBM PC serial port, the 232DRIO should be connected using a "straight through" DB-25 cable or a standard DB-9 to DB-25 cable adapter as shown in Table 2.3. If your communications equipment is configured as a DCE device, such as a modem, the 232DRIO should be connected using a "null modem" cable (see Table 2.4).

Table 2.2 – RS232 Connector Pinout

DB-9S Pin #	Signal	Signal Direction at 232DRIO	Notes
2	Receive Data (RD)	Output	Required
3	Transmit Data (TD)	Input	Required
5	Signal Ground (SG)	-----	Required
9	Power pin	Input	Can be wired to power unit

Table 2.3 – 232DRIO To DTE Connections

232DRIO Pin #	Signal	DTE DB-25 Connection	DTE DB-9 Connection
2	Receive Data (RD)	3	2
3	Transmit Data (TD)	2	3
5	Signal Ground (SG)	7	5
9	Ring Indicator	22	9

Table 2.4 – 232DRIO To DCE Connections

232DRIO Pin #	Signal	DCE DB-25 Connection	DCE DB-9 Connection
2	Receive Data (RD)	2	3
3	Transmit Data (TD)	3	2
5	Signal Ground (SG)	7	5
9	Ring Indicator	22	9

Power Supply Connections

The power requirement is 12VDC at 100mA. Power to the 232DRIO can be wired through the RS-232 serial port pin #9 (power pin) or by an external power supply through the blue terminal blocks. Serial ports can **not** provide enough power to supply the 232DRIO's 100 milliamp requirement.

Chapter 3 - *Commands*

There are only two basic commands required to control the 232DRIO: set output lines and read I/O lines. For normal environments, command strings are from three to four bytes in length; the “!” character, the “0” (zero) character, one command character, and one data byte (if required).

With serial communications in a laboratory environment, the possibility of a communication error occurring is minimal. However, in a harsh or an industrial environment the possibility increases. A communication error occurs when a bit transmitted as a “1” is received as a “0” or vice versa. If the 232DRIO receives an error in one or more of the first three command characters (“!0X”), the unit will not execute the command. However, if the 232DRIO receives a communication error on a data byte (I/O byte for Read Digital command or state byte for Set Output State command), the command will be executed since the unit has no way of knowing that there was an error.

To provide the 232DRIO with a way of detecting errors in the data fields, an additional set of commands can be used. This set of commands begins with the “#” (23h) character, instead of the “!” (21h) character. Refer to Table 3.1. With these commands every data byte that is transmitted or received is followed by its complement. For example, to read I/O lines:

Command syntax:

#0R

Response syntax:

{DATA}{~DATA}

Where “~” is used to indicate the “complement of.” If DATA has a reading of 1, the following would be received:

{01}{FE}

Where FEh is the complement of 1. The complement of number “x” can be calculated in QuickBasic as follows:

comp = (NOT x) AND &HFF

Table 3.1 – 232DRIO Commands

Function	Command	Response
Set Output Lines	!0S{...}	No response
Read I/O Lines	!0R	{...}
Set Output Lines	#0S{...}{~...}	No response
Read I/O Lines	#0R	{...}{~...}

Symbol: {...} represents one byte
 <...> represents a numeric value
 ~ complement of the specified data byte

Before going into the specifics of each command, it is important to understand that a byte has a numeric value from 0 to 255. The byte's value can be represented in decimal (0 -255) format, hexadecimal (00 - FF) format, binary (00000000 - 11111111) format or as an ASCII character. The fixed bytes of each command will be represented as ASCII characters, for example: "!0R". Refer to Table 3.1. However, it is important to remember that an ASCII character has a numeric value. Example: the ASCII "0" (zero) does not have a value of zero but has a value of 48. The decimal and hexadecimal equivalents of some ASCII characters are shown in Table 3.2. Some commands require an additional data byte to complete the command. These data bytes may be represented in any of the formats listed above. Refer to Appendix A for more ASCII and decimal equivalents.

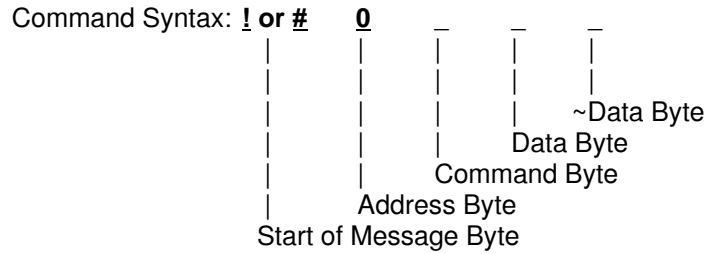
Table 3.2 – Equivalent Values

ASCII	Decimal	Hexadecimal
!	33	21h
#	35	23h
0	48	30h
R	82	52h
S	83	53h

Syntax

Command strings consists of three to five bytes depending on commands used for normal or harsh environments. The first byte is always the start of message byte. The start of message byte is either the ASCII "!" character (normal) or the ASCII "#" character (harsh). The second byte is the address byte. This byte allows each unit to have a unique address (useful in RS-485 networks). Since the 232DRIO uses RS-232 communications, this byte is always the ASCII "0" character and can not be changed. The next

byte is the command character. This byte is ASCII character and used to specify which command will be executed by the controller, either the "R" or "S" character. The set output command requires one argument field (for normal environments) or two argument fields (data and its complement for harsh environments). This field contains the fourth and/or fifth data byte.



I/O Data Bytes

When constructing commands to manipulate output lines or when reading the state of the I/O lines it is necessary to know how to select and interpret the I/O data byte. The three I/O lines are represented by one data byte.

A byte represents an eight-bit binary number (11111111), therefore each byte can represent eight I/O lines. Each bit is assigned a bit position and a weight (value). Refer to Table 3.3.

Table 3.3 – Bit Assignments for I/O Lines

Input → I #						I	R	R
Relay → R #	X	X	X	X	X	#1	#2	#1
Bit Position	7	6	5	4	3	2	1	0
Hex Weight	80	40	20	10	8	4	2	1
Decimal Weight	128	64	32	16	8	4	2	1

To set an output to a HIGH state the corresponding bit position must be set to a "1". Conversely to set an output LOW the corresponding bit position must be set to a "0". When reading I/O lines, any bit set to a "0" indicates the corresponding I/O line is in the LOW state and any bit set to a "1" indicates the corresponding I/O line is in the HIGH state.

Example 3.1 Set Output Lines

<u>Binary</u>	<u>Data Byte</u>		<u>Relays</u>	
	<u>Decimal</u>	<u>Hex</u>	<u>Relay #2</u>	<u>Relay #1</u>
XXXXXX00	0	0	de-energized	de-energized
XXXXXX01	1	1	de-energized	energized
XXXXXX10	2	2	energized	de-energized
XXXXXX11	3	3	energized	energized

Symbol: X don't cares (default setting should all be "0")

To energize the relays, the corresponding bit of the data byte must be set high. Conversely, to de-energize the relays the corresponding bit of the data byte must be set low.
(00000011, 3 in decimal, 3 in hex: energized both relay #1 and #2)

Example 3.2 Read I/O Lines

<u>Binary</u>	<u>Data Byte Returned</u>			<u>Input #1</u>	<u>Status</u>	
	<u>Dec</u>	<u>Hex</u>	<u>ASCII</u>		<u>Relay #2</u>	<u>Relay #1</u>
00000000	0	00	NUL	low	de-energ.	de-energ.
00000001	1	01	SOH	low	de-energ.	energ.
00000010	2	02	STX	low	energ.	de-energ.
00000011	3	03	ETX	low	energ.	energ.
00000100	4	04	EOT	high	de-energ.	de-energ.
00000101	5	05	ENQ	high	de-energ.	energ.
00000110	6	06	ACK	high	energ.	de-energ.
00000111	7	07	BEL	high	energ.	energ.

Notice that the returned data byte starts at address 00000000 in binary, 0 in dec., 0 in hex., or NUL in ASCII. From left to right (msb to lsb), the first five bits are really XXXXX "don't cares" but are defaulted as "00000". The remaining three bits represent input #1, relay #2, and relay#1, respectively. A "1" or a HIGH represents relay energized/input present, and a "0" or a LOW represents relay de-energized/input absent.

Read I/O Lines Command

The Read I/O Lines command returns one data byte (with the "!") and two data bytes (data and its complement with the "#") that reflects the state of the I/O lines. Bit 2 is Input #1, Bit 1 is Relay #2, and finally Bit 0 corresponds to Relay #1. If a bit is a "0" then the state of that I/O line is LOW. If a bit is a "1" then the state of that I/O line is HIGH.

Command: !0R

Argument: none

Response: the state of input and 2 relays (**shown in bold face**)

ASCII Example: !0R<NUL>

Dec. Example: !0R<**0**>

Hex. Example: !0R<**0**>

Bin. Example: !0R<**00000000**>

Description: This indicates that the input and both relays are "0"s or LOW.

Command: #0R

Argument: none

Response: data and its complement, state of the input and 2 relays (**shown in bold face**)

ASCII Example: #0R<NUL><blank 'FF'>

Dec. Example: #0R<**0**><**255**>

Hex. Example: #0R<**00**><**FF**>

Bin. Example: #0R<**00000000**><**11111111**>

Description: This indicates that the input and both relays are "0"s or LOW. Note that the complement is returned immediately following the data byte.

Set Output Lines Command

The Set Output Lines command is used to set the states of the relay outputs. This command requires one data byte (in normal environment with "!") or two data bytes (in harsh environment with "#"), data and its complement. This data byte specifies the output state of each relay output. Bit 1 is Relay #2, and Bit 0 is Relay #1. If a bit position is set to a "0" then the state of that output line will be

set LOW. If a bit position is set to a "1" then the state of that output line will be set HIGH.

Command: !0S

Argument: {...} (**shown in bold face**)

Response: none

ASCII Example: !0S<ETX>

Dec. Example: !0S<3>

Hex. Example: !0S<03>

Bin. Example: !0S<**00000011**>

Description: Relay #2 is energized ("1"), and Relay #1 is energized ("1").

Command: #0S

Argument: {...}{~...} (**shown in bold face**)

Response: none

ASCII Example: #0S<ETX><n>

Dec. Example: #0S<3><**252**>

Hex. Example: #0S<03><**FC**>

Bin. Example: #0S<**00000011**><**11111100**>

Description: Relay #2 is energized ("1"), and Relay #1 is energized ("1"). Note that the complement is returned immediately following the data byte.

NOTE: If any of these lines are defined as inputs the bit settings are ignored. Since Bit 2 is defined as Input #1 the bit setting is ignored.

Two important ideas to keep in mind when sending commands back to back or in sequence are the following:

- 1) If the "**!0R**" is sent, **one** character delay must follow immediately before any other commands are sent.
- 2) If the "**#0R**" is sent, **two** character delay must follow immediately before any other commands are sent.

For example: 1) "**!0R_!0S1**"
2) "**#0R_ _ #0S1**"

Otherwise, the microcontroller will miss the second "!" or "#" and not interpret the second command at all since it is still trying to transmit the returned data byte(s) to the user.

Chapter 4 - I/O Interfacing

This chapter will explain "HIGH" and "LOW" states and show some general examples of how to interface to the I/O lines. Caution must be taken not to exceed 232DRIO specifications listed in Chapter 1 when interfacing to external devices. Failure to stay within these specifications could result in damage to the unit and will void warranty.

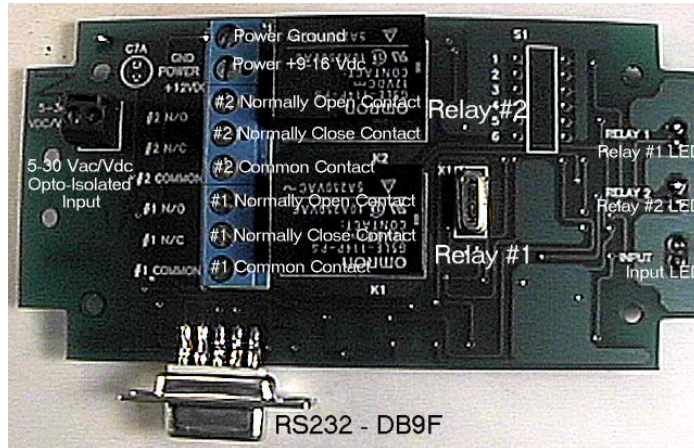


Figure 4.1 – Board Layout

Opto-isolated Input

As stated earlier, the opto-isolated input line is CMOS/TTL compatible and can also handle voltages from 0 to +30 Vdc or Vac.

The isolated input is used to sense a HIGH or a LOW state. This can be accomplished via switch closures, contact closures, or a solid state digital signal. When this input senses a voltage level +5Vdc/Vac or above, it will be considered "LOW" and its input state will be read as a "0" to the PIC® microcontroller. Conversely, when an input senses a voltage level below +1.50Vdc/Vac, it will be considered "HIGH" and its input state will be read as a "1" to the PIC microcontroller. Correspondingly, the PIC will then evaluate the incoming input and declare it as a "HIGH" if it is a "0" (at least +5V) and as a "LOW" if it is a "1" (less than +1.5V).

Figures 4.2 - 4.4 show examples of some typical input interfaces.

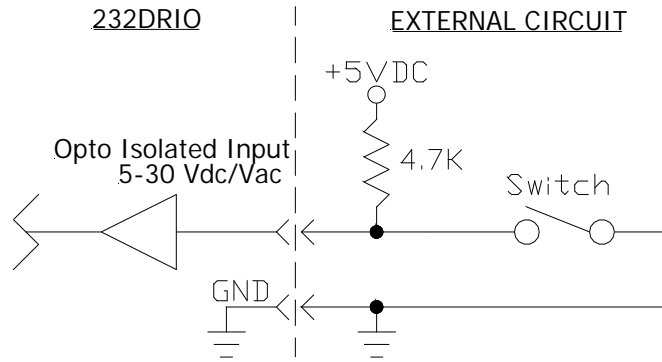


Figure 4.2 - Switch Input

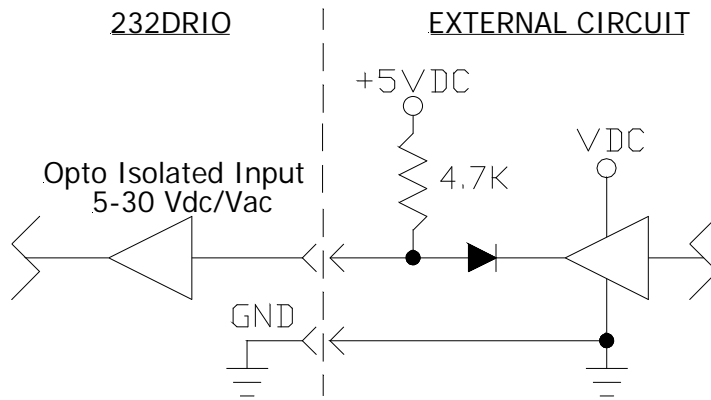


Figure 4.3 - Solid State Input

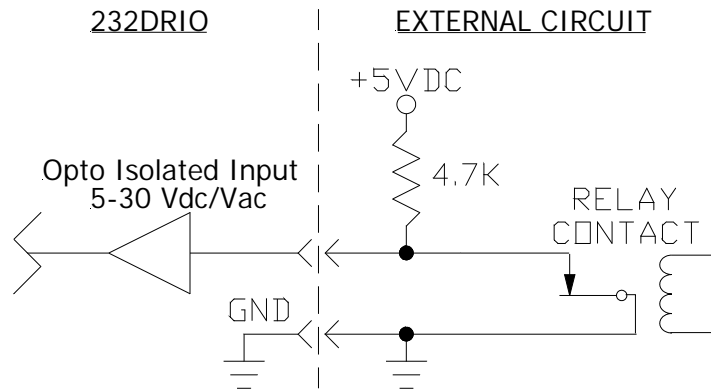


Figure 4.4 - Isolated Mechanical Input

Relay Outputs

Relay outputs are used to turn external devices on or off. Relay outputs are CMOS/TTL compatible and can also operate between 0-24Vdc and 0-115Vac. Outputs can be used to control solid state output modules, CMOS and TTL logic circuits. Caution must be taken not to exceed the power capability of the outputs. Refer to the output specifications in Chapter 1.

Setting an output line to a "1" forces the output HIGH, and setting an output line to a "0" forces the output LOW. **Both relays are de-energized at factory default settings.**

NOTE: The relay circuits are **not** UL® approved for 120 Vac @ 10A or 30 Vdc @ 8A (resistive) maximum line service in regards to contact ratings. However, the printed circuit traces were followed using UL standards (UL #1950) for minimum width and separation. For the previous reason and safety concerns, we recommend a maximum continuous rating of 8 amperes. In conclusion, the voltage should still comply with the rated relay contact voltage of 120 Vac/30 Vdc max. but recommended @ 8A or lower.

It is the customer's responsibility to determine the ultimate safety and acceptability of our products. Failure to stay within these specifications could result in damage to the unit and will void warranty.

Figures 4.5 - 4.6 show examples of some typical output interfaces.

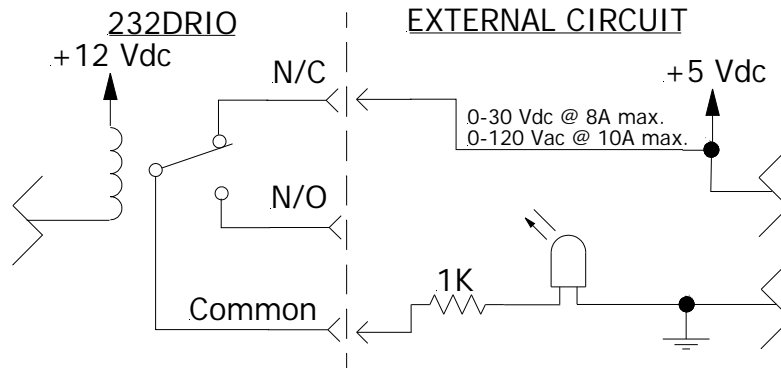


Figure 4.5 – Isolated Connection to External Circuit

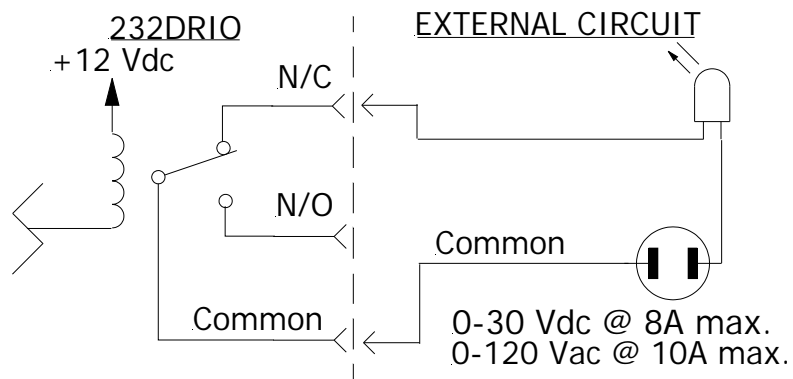


Figure 4.6 – Isolated Connection to External Devices

Chapter 5 - Software

This chapter will be divided into two sections. The first section covers programming techniques for opening/closing a com port, receiving data, and manipulating data in QuickBasic and C/C++. The second section discusses how to install and run the demonstration program on an IBM PC or compatible.

Table 5.1 - Digital I/O Mask Values

I/O Line #	Mask Values	
	Hexadecimal	Decimal
Relay #1	1H (0x01)	1
Relay #2	2H (0x02)	2
Input #1	4H (0x04)	4
X	8H (0x08)	8
X	10H (0x10)	16
X	20H (0x20)	32
X	40H (0x40)	64
X	80H (0x80)	128

Symbol: X Don't cares (default setting should all be "0")

Programming Techniques

232DRIO.LIB

Opening a COM Port

HComDev = initComPort(unsigned short addr, unsigned char irq, unsigned long baud, unsigned short PortIndex)

The last parameter, ***PortIndex*** will always be 0 for programming the 232DRIO module:

initComPort() → Returns 0 on error

For example, the code to open COM port 1 and setting the baud rate to 9600 would be:

HComDev = initComPort(0x03F8,4,9600,0)

Closing a COM Port

DeinitComPort(hComDev) → No return value

QuickBasic (DEMO232.EXE and DEMO232.BAS)

This section shows steps and examples of programming the 232DRIO in QuickBasic. If you are programming in another language, this section can be helpful as a guideline for programming the 232DRIO.

Using the 232DRIO.LIB with QuickBasic:

Using this library will make it easier to program the 232DRIO module and allows greater flexibility in choosing serial ports. At the DOS prompt, start QuickBasic with the 232DRIO Quick Library by typing:

QB program.bas /L 232DRIO.QLB

This allows you to call the 232DRIO functions from the QuickBasic editor.

'\$INCLUDE: '232DRIO.BI' → Include the function definitions

mem = SETMEM(-2000) → Tell QB to set aside memory for the 232DRIO library to use

Read I/O States Command

states = bbDRIOReadIOLines(hComDev, mode)

When the line is executed, **states** will contain a byte that can be read by "ANDing" **states** with the appropriate mask.

Example 5.1 - Determining the status of Relay #1

If (states AND 0x01) → True if relay 1 is on

If (states & 0x01)

Set Output States Command

status = bbDRIOSetOutputLines(unsigned short hComDev, unsigned char states, int mode) → Returns a negative number on error

Example 5.2 – Various ways to energized/de-energized Relay #1

1) A relay can be energized by "ORing" states with the appropriate mask.

states = states OR &H01 → Will energized Relay #1

2) A relay can be de-energized by "ANDing" with the complement of the mask. (~mask)

states = states AND &HFE → Will de-energized Relay #1

3) A relay can be switched by "Exclusive ORing" states with the appropriate mask.

states = states XOR &H01 → Will energized Relay #1 if it is

C/C++ (DEMO232.EXE and DEMO232.CPP)

DEMO232.CPP is a demonstration program for use with B&B Electronics model 232DRIO module. Much of the work associated with using COM ports in **C/C++** has been made easier by including the 232DRIO.LIB with this program.

Including “**232DRIO.H**” in the pre-processor directive and compiling your .cpp file as a project along with the 232DRIO.LIB file will enable you to use the following functions in your program.

Important Note: For Borland C++, the compiler **must** be set for **large** memory model for your program to compile properly. Access this by the pulldown menu **Options/Compiler/Code Generation/** to change model size and click on **ok**.

```
#include"232DRIO.H"
void main ()
{
  /* put your program here */
}
```

Read I/O States Command

states = bbDRIO_ReadIOLines(hComDev, int mode)

Mode is 1 if harsh environment is enabled or 0 if disabled. Returns -1 if an error occurred.

Example 5.3 – Determining the status of Relay #1

If(states & 0x01) → True if Relay #1 is on

When the line is executed, **states** will contain a byte that can be read by “ANDing” states with the appropriate mask.

Set Output States Command

bbDRIO_SetOutputLines(unsigned short hComDev, unsigned char states, int mode)

Example 5.4 – Various ways to energized/de-energized Relay #1

- 1) A relay can be energized by “ORing” states with the appropriate mask.
states |= 0x01 → Will energized Relay #1
- 2) A relay can be de-energized by “ANDing” with the complement of the mask. (~mask)
states &= 0xFE → Will de-energized Relay #1
- 3) A relay can be switched by “Exclusive ORing” states with the appropriate mask.

states ^= 0x01 → Will energized Relay #1 if it is de-energized or vice versa

Demonstration Program

The Demonstration Programs (IBM PC or Compatible) provide the user with examples of how to receive and transmit commands to the 232DRIO. The DEMO232.EXE is the executable program and the DEMO232.BAS file is the source code in QuickBasic. The DEMO232.EXE is the executable program and the DEMO232.CPP file is the source code in C/C++. The source codes provide an illustration of how to send and receive commands from the 232DRIO.

NOTE: This is a demonstration program only and not intended for system applications.

Running Demonstration Program

Before you can run the demonstration program you must run the install program in the Hard Drive Installation section. If you are running Windows, exit Windows to DOS.

To run the program follow these steps from the DOS prompt:

QuickBasic

1. Type **CD \BASIC** and press the **<ENTER>** key.
2. Type **DEMO232** and press the **<ENTER>** key.

C/C++

1. Type **CD \BC** and press the **<ENTER>** key.
2. Type **DEMO232** and press the **<ENTER>** key.

APPENDIX A

ASCII Character Codes

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

Hexadecimal/Decimal Conversions

The decimal (base 10) numbering system represents each position in successive powers of 10, with each decimal symbol having a value from 0 to 9. The hexadecimal (base 16) numbering system represents each position in successive powers of 16 with each hex symbol having a value of 0 to 15. Since each hex position must have a single symbol, the symbols "A" through "F" are assigned to values 10 through 15 respectively. Refer to Table 1. The information and examples to follow will explain how to convert from a decimal number to a hexadecimal number and vice versa.

Table 1.

Decimal Value	Hexadecimal Symbol
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Hexadecimal to Decimal Conversion:

$$\begin{aligned} \text{Decimal} = & \text{(1st Hex digit} \times 4096) + \\ & \text{(2nd Hex digit} \times 256) + \\ & \text{(3rd Hex digit} \times 16) + \\ & \text{(4th Hex digit)} \end{aligned}$$

Each "Hex digit" is the decimal equivalent value of the hexadecimal symbol.

Example: Convert 10FC hexadecimal to decimal.

$$\begin{array}{r} 1 \times 4096 = 4096 \\ 0 \times 256 = 0 \\ 15 \times 16 = 240 \\ 12 \times 1 = 12 \\ \hline 4348 \end{array}$$

10FC hex equals 4348 decimal.

Decimal to Hexadecimal Conversion:

Example: Convert 4348 decimal to hexadecimal.

$$\begin{array}{l} 4096 \overline{) 4348} = 1 = 1 \text{ (1st Hex digit)} \\ \quad 4096 \\ \quad \hline \quad 252 \\ 256 \overline{) 252} = 0 = 0 \text{ (2nd Hex digit)} \\ \quad 0 \\ 16 \overline{) 252} = 15 = F \text{ (3rd Hex digit)} \\ \quad 240 \\ \quad \hline \quad 12 \\ 1 \overline{) 12} = 12 = C \text{ (4th Hex digit)} \\ \quad 12 \\ \quad \hline \quad 0 \end{array}$$

4348 decimal equals 10FC hexadecimal.