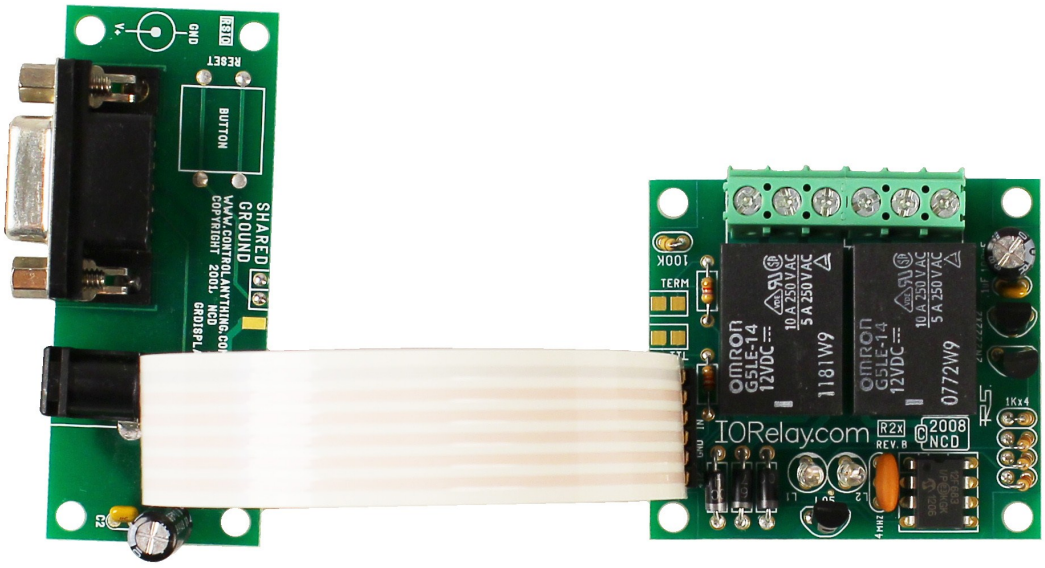
Quick Start Guide

R1x/R2x

RS-232 Serial

Single / Dual Relay Controllers



NOTICE: Portions of this Manual REQUIRE Internet Access

Table of Contents

Introduction

NCD Single and Dual Channel Relay Controllers were designed to service the needs of computer controlled switching applications using RS-232 direct wired communications or ZigBee Wireless Communications. Available in SPST, SPDT, and DPDT Relay Configurations, NCD offers the most versatile line of low-cost relay controllers in the world.

RS-232 Versions:
The RS-232 version requires no-more than 25ma to power the microprocessor. Each relay that is activated will impose no more than 80ma additional current draw. RS-232 versions can draw up to 185ma MAX when both relays are activated. Actual current may very depending on model, but will not exceed these current requirements. Standard operating voltage is 12VDC. Voltage should not exceed automotive application voltages (13.8VDC). Low voltages (below 10VDC) may allow CPU to function, but the on-board relays may not activate. RS-232 versions are compatible with IBM/PC systems with a compatible RS-232 Serial Port, USB to Serial Adapters, and Embedded microcontrollers with a TTL/CMOS communication lines. This device should not be used with Macintosh computers and should not be uses with UART/USART lines of a microprocessor without inversion of output signals (use a MAX202 or equivalent RS-232 line driver or inverter circuit to make this device compatible with UART/USART data lines). RS-232 versions are available in a low cost direct wired configuration or with an integrated RS-232 port. The integrated RS-232 port version requires a serial extension cable, Male to Female, and a 12VDC power supply with a 2.1mm Center Positive Barrel Connector. Integrated LEDs will light when a relay is activated. The RS-232 series controllers should not be powered without a connection to a data source (relays may randomly activate).

RS-232 versions operate exclusively at 9600 Baud (8 Data bits, 1 Stop Bit, No Parity, No Flow Control), and include a E3C command set, which allows a single computer to talk to multiple relay controllers.

SignalSwitch Compatibility
All RS-232 Versions of this device are compatible with www.signalswitch.com, which allows you to control these devices from anywhere in the world using a free account with a user-customized interface. A dedicated computer is required to process commands from www.signalswitch.com. The computer is used to communicate data from www.signalswitch.com to the relay controller using a standard COM port. Listener software (provided by www.signalswitch.com) serves as the software link between the device and www.signalswitch.com. Listener software MUST BE CONFIGURED for 9600 Baud for proper operation.

Getting Started

Important Power Supply Requirements:

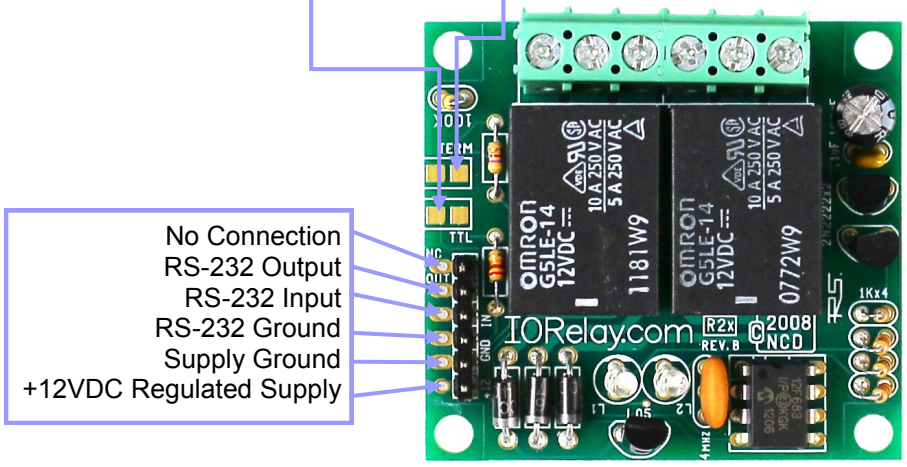
1. Do NOT use a wall wart type unregulated power supply.
2. Use only a computer grade regulated power supply rated at 12 Volts DC, 200ma or greater.
3. Use a supply rated for more amperage when powering multiple boards.
4. This device is compatible with automotive electrical systems.
5. DC power should never travel greater than 20 foot distances. A separate power supply should be used for each controller if controllers are not located with 20 feet of each other.

Note: We highly recommend the use of our QS12-F6 Quick Start Kit.

BAUD RATE IS FIXED AT 9600 BAUD.

TERM Solder Pad:
This solder pad terminates the RS-232 Data Output. These solder pads MUST be bridged with solder when using R2 connected to your serial port. If using multiple R2 boards attached to the same serial port, bridge these pad with solder on the R2 that is wired closest to your computer. Leave these pad unbridged on all other boards.

TTL Solder Pad:
When these solder pads are bridged, the R2 is configured to accept RS-232 data a TTL logic levels. These solder pads should be bridged when using Basic Stamp. Leave these solder pads unbridged when using a computer.



Relay Switching:

When switching voltages using the R2, a user-supplied voltage is typically connected to the COMMON post of the relay. When the Relay is OFF, the voltage on the COMMON is passed to the NORMALLY CLOSED position. When the relay is switched ON, the voltage on the COMMON is disconnected from the NORMALLY CLOSED and reconnected on the NORMALLY OPEN terminal.

Connecting Relays

Connector Pin outs for 1-Amp DPDT Series Controllers

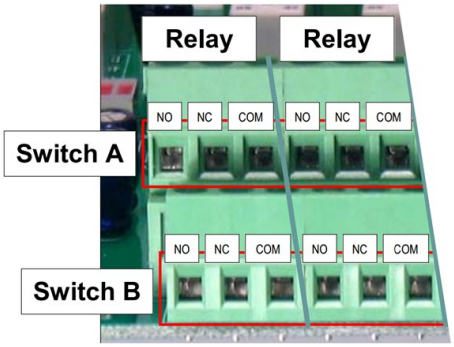
Any time you see one of our relay controllers with a green 12-Position terminal block and small white relays labeled AX-ICOM on the board, you can assume the wiring connections shown to the right will always apply. Always view the connector from the angle shown in the picture to the right. This will ensure connections are consistently made to 1A DPDT series controllers.

A DPDT relay has two separate switches inside each relay. These switches are labeled Switch A and Switch B in the diagram at right. Switches A and B are completely separated from each other, there is no electrical connection between these switches. When a DPDT relay is activated, both switches A and B "activate" or more appropriately change position at the same time. In the diagram at right, each connection is labeled NO, NC and COM.

COM: Common
This terminal is connected to NC when relay is off.
This terminal swings over to NO when the relay is on.

NO: Normally Open
This terminal has no connection with the relay is off.
This terminal is connected to COM when the relay is on.

NC: Normally Close
This terminal is connected to COM when the relay is off.
This terminal has no connection with the relay is on.



Connector Pin outs for 3-Amp and 5-Amp DPDT Series Controllers

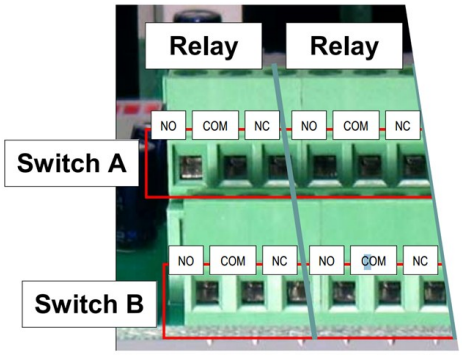
Any time you see one of our relay controllers with a green 12-Position terminal block and black relays on the board, you can assume the wiring connections shown to the right will always apply. Always view the connector from the angle shown in the picture to the right. This will ensure connections are consistently made to 3A and 5A DPDT series controllers.

A DPDT relay has two separate switches inside each relay. These switches are labeled Switch A and Switch B in the diagram at right. Switches A and B are completely separated from each other, there is no electrical connection between these switches. When a DPDT relay is activated, both switches A and B "activate" or more appropriately change position at the same time. In the diagram at right, each connection is labeled NO, NC and COM.

COM: Common
This terminal is connected to NC when relay is off.
This terminal swings over to NO when the relay is on.

NO: Normally Open
This terminal has no connection with the relay is off.
This terminal is connected to COM when the relay is on.

NC: Normally Close
This terminal is connected to COM when the relay is off.
This terminal has no connection with the relay is on.



Base Station Software

You will now need to download and install NCD Base Station Software. Base Station Software allows you to communicate with and test your device. When Base Station Software runs on your Windows computer, the software will identify the type of controller and will build a list of command sets that are compatible with your controller. Below are the basic parts to the main Base software application:

1. **Identification and Documentation**: This button displays Read-Only information stored in the controller. It can help you identify the type of device that is connected and what its capabilities are. This portion of the software also builds a library of documents that will be helpful in using the controller it has identified. This library of documents will change depending on the type of device which has been identified.

2. **COMM Operator**: COMM Operator is a tool for testing and learning how to communicate with all Network and COM based devices. This manual may include command codes you can send to the device using COMM Operator. COMM Operator should be thought of as a terminal to send and receive bytes of data. COMM Operator was used extensively in the development of this device and should be referenced throughout the learning process. COMM Operator is a commercial product and is NOT Free. The 30-Day Evaluation version is provided and users may purchase a license for this software if they would like to continue use beyond 30 days.

3. **Device Command Sets**: Each Device contains a set of commands that are identified by the Base Station software. Choose the command set you would like to explore in this box. Click one time on the command set you would like to explore.

Note: There may be more basic features depending on your device. Other features you may notice:

- **Device Configuration**: This button allows you to modify important device settings to help improve communication speed, functionality, and timing parameters of the device. Device Configuration is rarely used by the user

- **Display Command Set**: When working with Base software, many windows will display the actual command codes used to trigger a particular function. This option allows you to choose Decimal and Hex formats. This manual is shown in both Hex and Decimal format. Decimal is typically used in COMM Operator and is our preferred format, but Hex works great too.

- **Run Mode**: Run mode is used for daily operations and is the default mode of operation. To prevent accidental writes to non-volatile memory, the device must be placed in Configuration Mode to change EEPROM memory. Click this button anytime you need to change modes. Note that a jumper on the controller will force this device into Configuration mode. If a device is powered up in Configuration Mode and you are using a Web-i interface, the Web-i will boot in DHCP mode as a safeguard in case the device becomes inaccessible with a static IP address.

Networking Multiple R2 Relay Controllers

The R2 supports our E3C command set, allowing you to control up 256 NCD devices using a single RS-232 serial port. To control multiple R2 controllers from a single serial port, follow these simple steps:

Programming

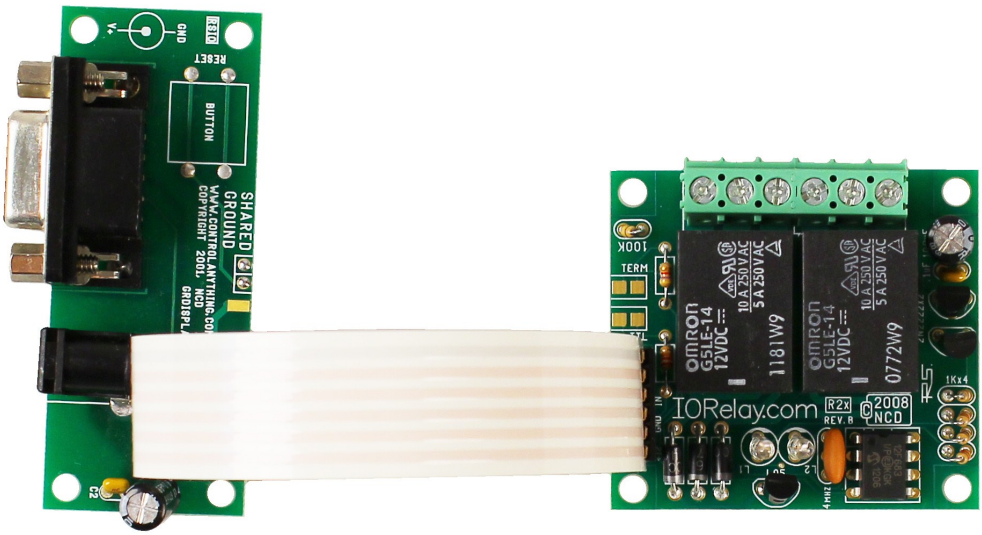
1. Connect a single R2 to the serial port of your computer.
2. Program the R2 with a unique device number.
3. Label the R2 controller with its device number.
4. Repeat these steps until ALL R2 controllers are programmed with a unique device number and are clearly labeled.

Wiring

Once all devices have been individually programmed with a unique device number, you are ready to connect multiple R2 controllers on a single serial port. Connect power and serial wires to EVERY R2 controller on the network. The quickest way to do this is use our Quick Start 12 kit (QS 12-F6) to provide serial and power connections to the first board. Next, additional flat-ribbon cable can be used to connect from the first R2 to the next R2 in the chain. Keep running flex cable in parallel with the first board until all R2 controllers are connected. Once connected, bridge the TERM solder pad with the solder on the R2 board that is physically located closest to your computer. This will terminate RS-232 communications coming from the R2 back to your computer..

Controlling

Once all boards are connected as described above and powered up, all R2 Controllers will respond to your commands. So if you were to issue a command for switching, all R2 controllers will respond to your switch command. Simply use the E3C command set to control which devices you would like to speak to using the E3C device number you have programmed into the controllers. The E3C command set supports commands for controlling all boards at once, each board individually (command 252, the most popular choice), or you can program some devices to listen and other devices to ignore your commands.



Above: R2x connected to the RSIO (Included with the Quick Start 12 Kit-QS12-F6).

Sending Commands to the R2

The R2 is capable of receiving data via RS-232 serial communications and is compatible with just about any computer or microcontroller ever produced, including the Macintosh, Amiga, Basic Stamp, and of course, Windows & DOS based machines. Regardless of the system you are using, you will need access to a programming language that supports program control of the serial port on your system.

A terminal program is not suitable for controlling this device. Commands should be sent using ASCII character codes 0-255 rather than ASCII characters (A, B, C, 1, 2, 3, 4, etc.). See "ASCII Codes vs. Characters" on this page. Note that ASCII character codes are used for commanding the CLCD while actual ASCII characters are used to send text to the display.

Most systems require you to open the appropriate serial port (COM port) prior to sending or receive data.

Because there are so many different ways to send and receive data from various languages on various platforms, we will provide generic instructions that can be easily converted to your favorite language.

For example, if this manual says "Send ASCII 254", the user will need to translate this instruction into a command that is capable of sending ASCII character code 254.

To Send ASCII 254 from Visual Basic, you will use the following line:

MSComm1.Output = Chr$(254)

In Qbasic, you can send ASCII 254 using the following line of code:

Print #1, Chr$(254);

Note that sending ASCII character code 254 is NOT the same as sending ASCII characters 2, 5, and 4 from a terminal program. Typing 2, 5, and 4 on the keyboard will transmit three ASCII character codes.

For your convenience, we have provided programming examples for Visual Basic 6. The provided examples should greatly speed development time. You may want to visit www.controlanything.com for the latest software and programming examples.

Programming examples for this device are much more extensive for Visual Basic 6 users than for any other programming language. If you are not a VB programmer, you may consider looking at the VB6 source code, as it is easily translated into other popular languages.

Regardless of your programming background, the provided Visual Basic 6 source code is very easy to understand and will likely resolve any communication questions you may have. VB6 programming examples may be viewed in any text editor such as Notepad or WordPad.

ASCII Codes vs. Characters

The differences between ASCII codes and ASCII characters tend to generate a lot of confusion among first-time RS-232 programmers. It is important to understand that a computer only works with numbers. With regard to RS-232 data, the computer is only capable of sending and receiving numbers from 0 to 255.

What confuses people is the simple idea that the numbers 0 to 255 are assigned letters. For instance, the number 65 represents the letter A. The number 66 represents the letter B. Every character (including numbers and punctuation) is assigned a numeric value. This standard of assignments is called ASCII, and is a universal standard adopted by all computers with an RS-232 serial port. ASCII characters codes can be clearly defined as numbers from 0 to 255.

ASCII characters however are best defined as letters, A, B, C, D, as well as punctuation, !@#$%, and even the numbers 0-9. Virtually all programming languages permit you to send ASCII in the form of letters or numbers. If you wanted to send the word “Hello” out the serial port, it is much easier to send the letters H, e, l, l, and o than it is to send the ASCII character codes that represent each letter.

For the purposes of controlling NCD devices however, it is much easier to build a numeric command set. Especially when communicating to devices that have numeric parameters, such as the E3C command set. Put simply, it is easier to control NCD devices using ASCII character codes 0 to 255 than it is to use ASCII characters A, B, C, D, etc.

Because terminal programs are ASCII character based, it may be difficult to generate the proper series of keystrokes that would be necessary to activate a particular function. Therefore, terminal programs are not suitable for controlling NCD devices. In a real world control application, a terminal program would not likely be used to control NCD devices anyway. Therefore, a programming language that supports the transmission and reception of ASCII character codes 0 to 255 is highly recommended.

E3C Command Set

The E3C command set allows you to control up to 256 NCD devices from a single serial port. It is OK to mix different types of devices, as long as the devices are E3C compliant. This device supports the full set of E3C commands.

How Does E3C Work?
First of all, each device must be assigned a device number from 0 to 255. The device number may be programmed using command 255, described later in the document, using our Visual Basic 6 example code, or using our Runtime .EXE programs. E3C stands for Enabled 3-Wire Communication. Put Simply, when you first power up your computer and all the devices attached to the serial port, all devices will respond to your commands. Using the E3C command set, you can specify which devices will listen and which devices ignore your commands. Note that E3C commands are never ignored by any device, regardless of the commands you sent to the controller. The number to the left of each command indicates the ASCII character code that must be sent to issue the command. All commands must be preceded with ASCII character code 254 to place the device in command mode. See examples:

Read E3C Device Number

Reports back the pre-programmed E3C device number of the R2. This command will return a value of 0-255.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	247
Hex Values	0xFE	0xF7

Receive Byte:		
	Decimal:	0-255
	Hex:	0x00-0xFF

Sample Code: Get E3C Device Number

```
MSComm1 . Output = Chr$(254)    'Enter Command Mode
MSComm1 . Output = Chr$(247)    'Read Device Number Command
Do                              'Wait for Device to  Rely
        DoEvents                'Allow Windows to Multitask
Until MSComm1. InBufferCount  >  0 'If the Device Replies
GetDeviceNumber = Asc(MSComm1. Input) 'Get Device Number from Buffer
```

Enable All Devices

Tells all devices to respond to your commands.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	248
Hex Values	0xFE	0xF8

Receive Byte:		
	Decimal:	85
	Hex:	0x55

Sample Code: Enable all E3C Devices

```
MSComm1 . Output = Chr$(254)    'Enter Command Mode
MSComm1 . Output = Chr$(248)    'E3C Enable All Device Command
```

E3C Command Set

Disable All Devices
Tells all devices to ignore your commands.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	249
Hex Values	0xFE	0xF9

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Disable all E3C Devices

```
MSComm1 . Output = Chr$(254)     'Enter Command Mode
MSComm1 . Output = Chr$(249)     'E3C Disable All Device Command
```

Enable A Selected Device
Tells a specific device to listen to your commands.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	250
Hex Values	0xFE	0xFA

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Enable a specific E3C device, other devices will be unchanged

```
MSComm1 . Output = Chr$(254)     'Enter Command Mode
MSComm1 . Output = Chr$(250)     'E3C Enable Specific Device Command
MSComm1 . Output = Chr$(Device)  'Device Number that will be enabled
```

Disable Selected Device
Tells a specific device to ignore your commands.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	251
Hex Values	0xFE	0xFB

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Disable a specific E3C device, other devices will be unchanged

```
MSComm1 . Output = Chr$(254)     'Enter Command Mode
MSComm1 . Output = Chr$(251)     'E3C Disable Specific Device Command
MSComm1 . Output = Chr$(Device)  'Device Number that will be inactive
```

E3C Command Set

Enable Selected Device Only

Tells a specific device to listen to your commands, all other devices will ignore your commands.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	252
Hex Values	0xFE	0xFC

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Disable all E3C devices except (Device)

```
MSComm1 . Output = Chr$(254)     'Enter Command Mode
MSComm1 . Output = Chr$(252)     'E3C Disable All Device Except Command
MSComm1 . Output = Chr$(Device)  'Device Number that will be Active
```

Disable a Selected Device Only

Tells a specific device to ignore your commands, all others will listen.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	253
Hex Values	0xFE	0xFD

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Enable all E3C devices except (Device)

```
MSComm1 . Output = Chr$(254)     'Enter Command Mode
MSComm1 . Output = Chr$(253)     'E3C Enable All Devices Except Command
MSComm1 . Output = Chr$(Device)  'Device Number that will be Inactive
```

E3C Command Set

Extended E3C Commands

The R4x/R8x Pro supports two additional E3C commands which should only be used when a single device is attached to your serial port. Extended commands will report back to the computer.

Store Device Number

Stores the device number into the controller. The device number takes effect immediately. The enabled/disabled status of the device is unchanged.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	255
Hex Values	0xFE	0xFF

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Store an E3c device number into the controller

```
MSComm1 . Output = Chr$(254)            'Enter Command Mode
MSComm1 . Output = Chr$(255)            'E3C Store Device Number Command
MSComm1 . Output = Chr$(Device)         'Device Number that will be stored
WaitForReply
```

Recall Device Number

Allows you to read the stored device number from the controller.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	247
Hex Values	0xFE	0xF7

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Read the E3C device number from the controller

```
MSComm1 . Output = Chr$(254)            'Enter Command Mode
MSComm1 . Output = Chr$(253)            'E3C Get Device Number Command
Do                                      'Wait for device to Reply
        DoEvents                        'Allow Windows to MultiTask
Until MSComm1 . InBufferCount > 0       'If the device Replies
GetDeviceNumber=Asc(MSComm1 . Input)    'Get Device Number from Buffer
```

Using RS-232 Single/Dual Relay Controllers

Note: Send ALL commands to RS-232 Single/Dual Relay Controllers at 9600 Baud, 8 Data Bits, No Parity, No Flow Control.

254, 0: Turn Off Relay 1
Turns off Relay 1

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	0
Hex Values	0xFE	0x00

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Relay One Off

```
MSComm1 . Output = Chr$(254)    'Enter Command Mode
MSComm1 . Output = Chr$(0)      'Turn Off Relay 1
```

254, 1: Turn On Relay 1
Turns on Relay 1

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	1
Hex Values	0xFE	0x01

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Relay One ON

```
MSComm1 . Output = Chr$(254)    'Enter Command Mode
MSComm1 . Output = Chr$(1)      'Turn On Relay 1
```

254, 2: Turn Off Relay 2
Turns off Relay 2

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	2
Hex Values	0xFE	0x02

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Relay Two OFF

```
MSComm1 . Output = Chr$(254)    'Enter Command Mode
MSComm1 . Output = Chr$(2)      'Turn Off Relay 2
```

Using RS-232 Single/Dual Relay Controllers

254, 3: Turn On Relay 2
Turns on Relay 2

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	3
Hex Values	0xFE	0x03

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Relay Two ON

```
MSComm1 . Output = Chr$(254)    'Enter Command Mode
MSComm1 . Output = Chr$(3)      'Turn On Relay 2
```

254, 4: Get Relay 1 Status
This command reports back 0 or 1 indicating the status of Relay 1. 0=Off, 1=On.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	4
Hex Values	0xFE	0x04

Receive Byte: Decimal: 0 or 1
Hex: 0x00 or 0x01

Sample Code: Relay One Status

```
MSComm1 . Output = Chr$(254)    'Enter Command Mode
MSComm1 . Output = Chr$(4)      'Ask for Status of Relay 1
Do                              'Wait for Device to Reply
        DoEvents                'Allow Windows to Multitask
Until MSComm1. InBufferCount>0  'If the Device Replies
RelayOneStatus=ASC(MSComm1.Input)  'Get Status of Relay 1
```

254, 5: Get Relay 2 Status
This command reports back 0 or 1 indicating the status of Relay 1. 0=Off, 1=On.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	5
Hex Values	0xFE	0x05

Receive Byte: Decimal: 0 or 1
Hex: 0x00 or 0x01

Sample Code: Relay Two Status

```
MSComm1 . Output = Chr$(254)    'Enter Command Mode
MSComm1 . Output = Chr$(5)      'Ask for Status of Relay 2
Do                              'Wait for Device to Reply
        DoEvents                'Allow Windows to Multitask
Until MSComm1. InBufferCount>0  'If the Device Replies
RelayOneStatus=ASC(MSComm1.Input)  'Get Status of Relay 2
```

Using RS-232 Single/Dual Relay Controllers

254, 6: Set Status of Both Relays

This command sets the status of BOTH relays at one time. This command requires a parameter of 0-3:
Parameter 0: Turn Off Both Relays
Parameter 1: Turn On Relay 1, Turn Off Relay 2
Parameter 2: Turn On Relay 2, Turn Off Relay 1
Parameter 3: Turn On Both Relays

Send Bytes:	**Byte 1:**	**Byte 2:**	**Byte 3:**
Function:	Command	Command	Parameter
Decimal Values:	254	6	0 - 3
Hex Values	0xFE	0x06	0x00 - 0x03

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Set Both Relays

```
MSComm1 . Output = Chr$(254)     'Enter Command Mode
MSComm1 . Output = Chr$(6)       'Set Both Relays Command
MSComm1 . Output = Chr$(Status) 'Set Status of Both Relays
```

254, 7: Get Status of Both Relays

This command reports the On/Off status of both relays. This command will send a byte from 0 to 3 back to the user indicating the status of both relays:
Return Byte 0: Both Relays are Off
Return Byte 1: Relay 1 is On, Relay 2 is Off
Return Byte 2: Relay 2 is On, Relay 1 is Off
Return Byte 3: Both Relays are On

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	7
Hex Values	0xFE	0x07

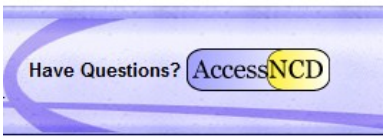
Receive Byte: Decimal: 0 - 3
Hex: 0x00 - 0x03

Sample Code: Get Both Relay Status

```
MSComm1 . Output = Chr$(254)     'Enter Command Mode
MSComm1 . Output = Chr$(7)       'Ask for Status of Both Relays
Do                               'Wait for Device to Reply
        DoEvents                 'Allow Windows to Multitask
Until MSComm1. InBufferCount>0   'If the Device Replies
GetBothRelayStatus=Asc(MSComm.1Input)  'Get Status of Both Relays
```

Technical Support

Technical support is available through our website, controlanything.com.
AccessNCD is the way we connect NCD engineers to our customers.
Click on the **AccessNCD** button located on the top right of the header of each page of our website.



For technical support and application information, contact Travis Elliott, our technical engineer. If you feel that you have discovered a bug in the firmware of our controllers, contact Ryan Sheldon, our hardware developer. If you have programming-related questions or have discovered a bug in our software, please contact Shirui Xu, our software engineer.

Click the '*Tech Support Staff*' tab and click on the appropriate engineer link for assistance. Click on our '*Forum*' tab if you would like to post publicly or review problems that other customers have had and our recommended solutions.

Our engineers monitor questions and respond continually throughout the day. Before requesting telephone technical support, we ask that customers please try to resolve their problems through **AccessNCD** first. However, for persistent problems, NCD technical support engineers will schedule a phone consultation.

Contact Information

National Control Devices, LLC
PO Box 455
Osceola, MO 64776
417-646-5644 phone
866-562-0406 fax
Open 9 a.m. - 4 p.m. CST

All orders *must* be placed online at our website, www.controlanything.com

Notice:
The only authorized resellers of NCD products are

- www.controlanything.com
- www.relaycontrollers.com
- www.relaypros.com
- www.amazon.com

All other websites are not authorized dealers; we have noticed some retailers offering our products fraudulently.