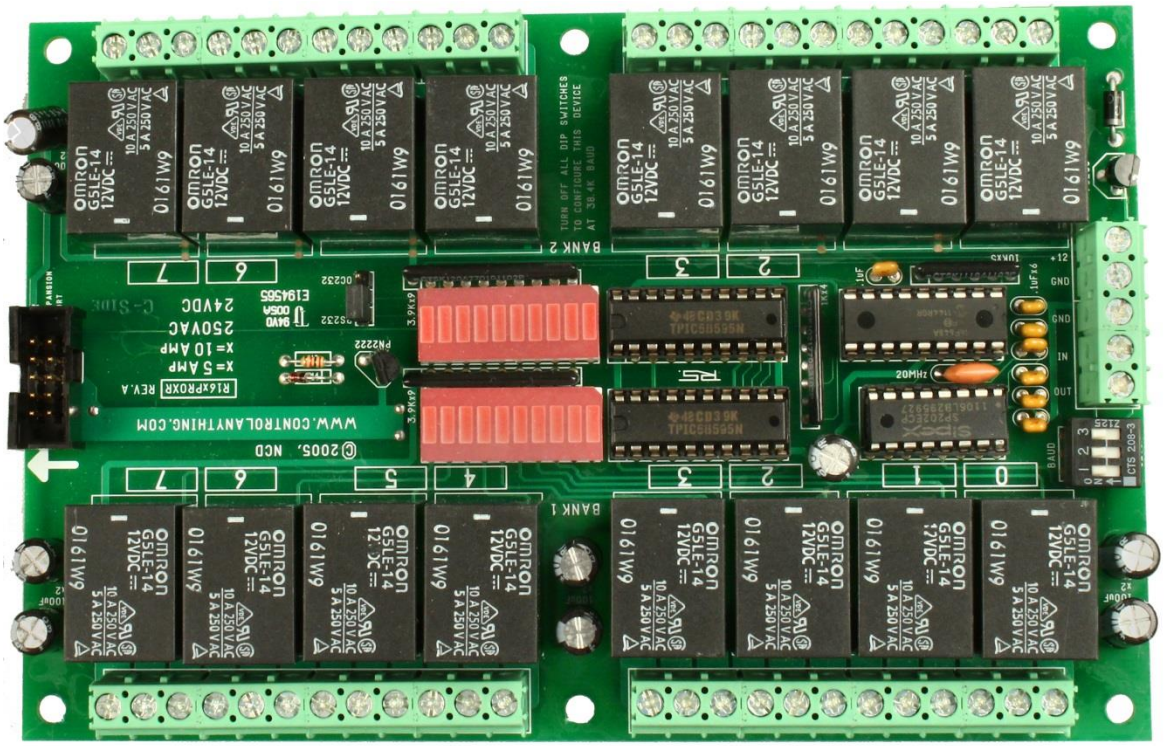
Control Anything...Anywhere.

NATIONAL CONTROL DEVICES

R4x/R8x Pro Quick Start Guide



RS-232 Networkable Relay Controllers

NATIONAL CONTROL DEVICES

R4x/R8x Pro

RS-232 Networkable Relay Controllers

National Control Devices, LLC
PO Box 455
Osceola, MO 64776
Phone 417.646.5644 • Fax (866) 562-0406

Notice: Portions of this manual require internet access.

Table of Contents

Control Anything...Anywhere.

Chapter 1

Getting Started

Introduction

Our R4x or R8x series relay controllers have made their way into thousands of computer control applications all over the world. With the release of more sophisticated microcontrollers, it is now possible to expand the features of our original designs.

The R8x and R4x Professional series relay controllers were designed to take advantage of recent advances in microcontroller technology. Our new Professional series relay controllers offer an enormously powerful new command set. E3C compliance and the ability to emulate the original R4x and R8x commands. In addition, a variety of new products will take advantage of our new microprocessors, adding a large variety of choices to our 4 and 8 relay controller designs. This guide will show you how to take advantage of the extensive new features offered by our firmware. This guide is written to generically cover all R4x Pro and R8x Pro series relay controllers.

Important Power Supply Requirements

1. Do NOT use a wall wart type unregulated power supply.
2. Use only COMPUTER GRADE REGULATED SWITCHER SUPPLY, rated at 12V DC 1.25 AMPS or greater.
3. Use a supply rated for more amperage when powering multiple boards.
4. DC power should never traveler a distance greater than 20 ft.
5. Relay coils are rated at 12V DC. Higher voltages will shorten the coil life. Lower voltages may cause unreliable operation, but will not damage the controller. Minimum supply voltage is 9 VDC.
6. The R32 can be used in 12 volt automated electrical systems.

Chapter 2

Jumper Settings

Jumper	Description	*Gray indicated default settings Jumper Left	*Gray indicated default settings Jumper Right
EMU or Emulate	The EMULATE jumper is used to make the R4x/R8x Pro act as if it were the original R4x or R8x relay controller, responding to all the same commands as the original design. <NEW COMMAND ARE NOT AVAILABLE IN THIS MODE. THIS MODE DOES NOT SUPPORT E3C NETWORKING> Install a jumper closest to the EMU or EMULATE label to activate EMULATION.	Standard E3C Mode with New Command Set.	Emulation Mode is set ACTIVE, New Commands Not Available.
B1 or BAUD1	BAUD select jumper 1.	B1 is OFF when set between the two posts closest to the PIC CPU.	B1 is ON when set between the post furthest from the PIC CPU.
B2 or BAUD2	BAUD select jumper 2.	B2 is OFF when set between the posts closest to the PIC CPU.	B2 is ON when set between the post furthest from the PIC CPU.
PC/MAC	Input data voltage select, PC +/- 12V RS232, MAC +/5V RS232 specification. Use MAC mode for Mac systems, PC laptops, USB to Serial Adapters, and TTL data sources. It is okay to try both settings if you are unsure for your application. No damage will result due to improper setting of this jumper.	Nearest to PC Label: Desktop PC Data Source	Nearest to MAC Label: Laptop PC, MAC, or TTL Data Source
TTL/OC	Output data mode, TTL 0/+5V TTL Data Output compatible with PCs and LAPTOP computers. Use TRUE32 converter (sold separately) for MAC systems. This jumper is ONLY necessary if using 2-way communication. O.C. Output Mode means Open Collector, us with RSB Serial Booster (sold separately) when using this mode. TTL mode is typically used for laptop, Desktop PCs, and Embedded computers. No damage will result to improper setting of this jumper.	Nearest to TTL Label: Data Output compatible with Desktop PCs and Embedded communications.	Nearest to OC Label: Data Output is electrically and Open Collector Signal. Use with NCD RSB Serial Booster ONLY.

Setting Jumpers

1. Jumpers should be set PRIOR to applying power to the board. Jumper setting will have no effect if changed while power is on.
2. Each Jumper option has three pins. For example the TTL/OC jumper has three pins. Installing a jumper between the Right two pins sets the operation to OC mode. A jumper MUST be installed between TWO of the THREE pins.

B1 B2 Baud Select		
B1	**B2**	**Baud**
OFF	OFF	1200
ON	OFF	2400
OFF	ON	9600
ON	ON	119.2K

Status LEDs

Power/E3C Enabled LED:

This first LED is normally lit when power is first supplied to the board. This LED is controlled by the E3C command set. When lit, the R4xR8x Pro is enabled and ready to accept relay control commands. When this LED is off, the only command that will be processed are E3C commands. The R4x/R8x Pro will not process your relay control commands again until this LED is lit. Use the E3C command set to enable the R4x/R8x Pro.

Data In:

The second LED is normally off and flashes as data is received by the relay controller.

Factory Default Settings

Default Settings are Highlighted in gray above.

Communication Parameters

User Selected Baud Rate, 8 Data Bits, 1 Stop Bit, and No Parity.

Control Anything...Anywhere.

Chapter

3

Communication

Two-Way Communication

The R4x/R8x Pro support two-way communication for confirming the receipt of commands and for reporting the status of the relays back to the host computer.

The R4x/R8x Pro should be connected as shown below when using this device for the first time. Even if you plan to connect several R4x/R8x Pro controllers to a single serial port, this wiring diagram must first be used to program the device number into the controller. The R4x/R8x Pro Visual Basic Example Program expects tis wiring configuration.

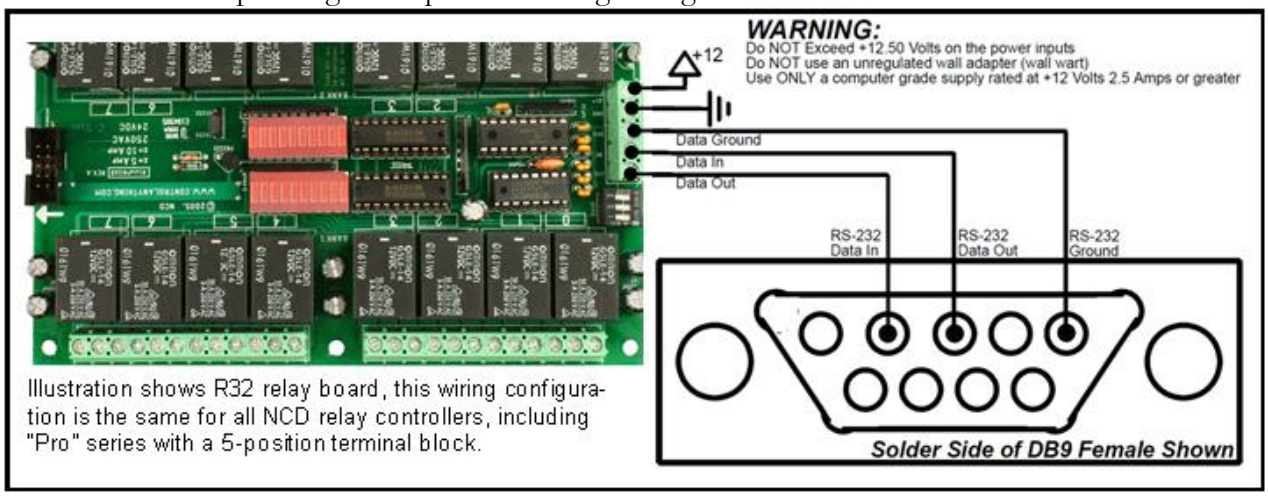


Illustration shows R32 relay board, this wiring configuration is the same for all NCD relay controllers, including "Pro" series with a 5-position terminal block.

R4x/R8x Pro One-Way Communication:

The R4x/R8x Pro can be connected to a computer or microcontroller using as little as two wires. Memory Storage commands may take a little longer to process than others, so it may be necessary to add short delays in your program to allow time for execution of these commands. When used in 1-way mode, reporting should be turned off for highest communication speed. Turning off reporting will allow you to send commands to the R4x/R8x Pro much faster, but it is impossible to ask the controller for the status of relays when wired as shown below.

Reporting Mode is activated by sending ASCII character codes 254, 27.

Reporting Mode is deactivated by sending ASCII character codes 254, 28.

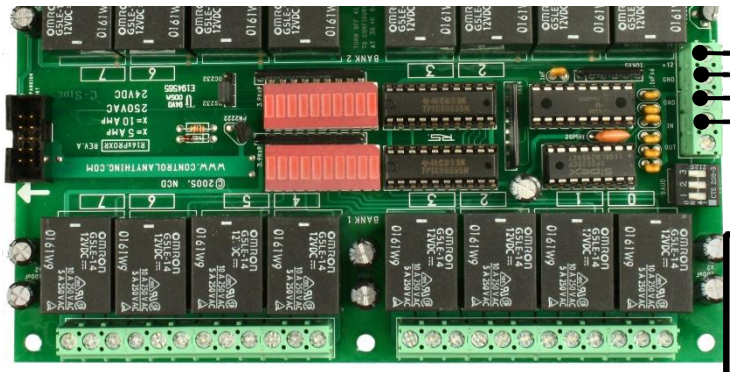
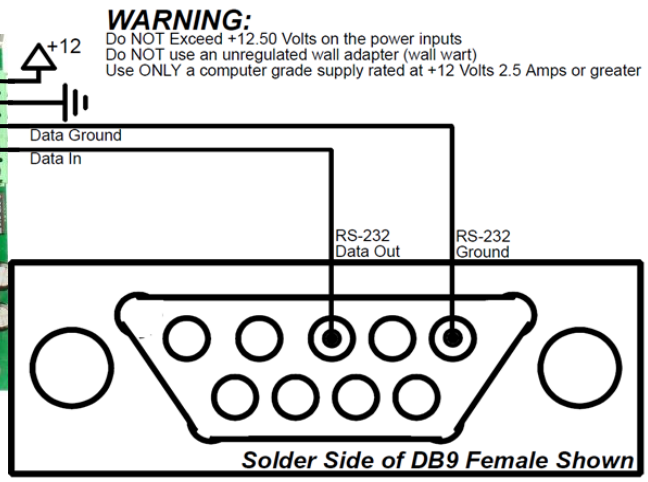


Illustration shows R32 relay board, this wiring configuration is the same for all NCD relay controllers, including “Pro” series with a 5-position terminal block.



Multiple Relay Controller: Two-Way Hybrid Communication

Multiple NCD Devices can be connected to a single serial port and controlled individually. This example shows an R16 and an R32 connected to a single serial port.

Before using this wiring configuration, each device must be programmed with a unique device number. (See E3C Commands in Both Manuals for Details). Once a device number has been stored into each controller this wiring configuration may be used to control up to 256 different relay boards or other NCD devices in any combination. This wiring configuration only allows 2-way communication with the R32. Relay status information cannot be read form the R16.

When all boards are first powered up, all devices will respond to incoming commands. Use E3C Command 252 to speak to one device at a time. Send 252, 0, any subsequent commands should be for the R32, Device 0. Send 252, 1, any subsequent commands should be for the R16, Device 1.

Multiple Relay Controllers: Two-Way & One-Way Hybrid

Step 1: Store a Device Number from 0 to 255 into Each Controller. Example Shows Device 0 and 1.

Step 1: Route Commands to Device 0 Only by Sending the Following Commands:

ASCII 254	'Enter Command Mode
ASCII 252	'Select a Device to Control Command
ASCII O	'Set Device to Control to 0

Step 3: Activate Relay 1 on Device 0 (R32)

ASCII 254	'Enter Command Mode
ASCII 1	'Relay On Command
ASCII 0	'Turn On Relay

Step 4: Route Commands to Device 1 Only by Sending the Following Commands

ASCII 254	'Enter Command Mode
ASCII 252	'Select a Device to Control Command
ASCII 1	'Set Device to Control to 1

Step 5: Activate Relay 1 on Device 1 (R16)

ASCII 254	'Enter Command Mode
ASCII 16	'Turn Relay 0 On

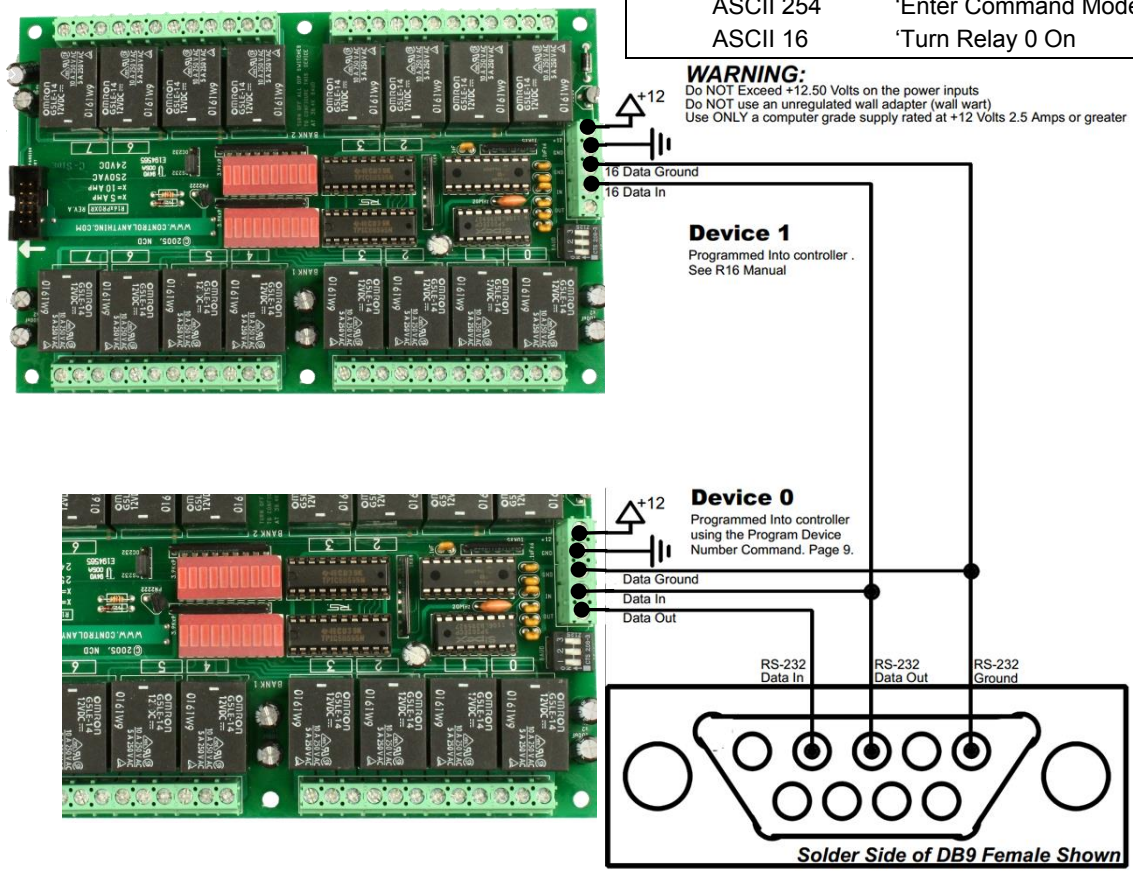


Illustration shows R32 relay board, this wiring configuration is the same for all NCD relay controllers, including "Pro" series with a 5-position terminal block.

Multiple Relay Controllers: Two-Way Communication

Our Relay Controllers support two-way communication to multiple devices using the RSB serial booster. Jumpers must be set for Open Collector data transmission. See the appropriate manual for your relay controller. This is ONLY required when using the RSB serial booster. The RSB serial booster should be used when controlling several devices over long distances (has been tested in excess of 500 feet). Actual reliability over long distances depends greatly on baud rate and type of wire used. Experimentation will be required. Always start at the lowest baud rate and work your way up.

A unique device number should be programmed into each device prior to using this example.

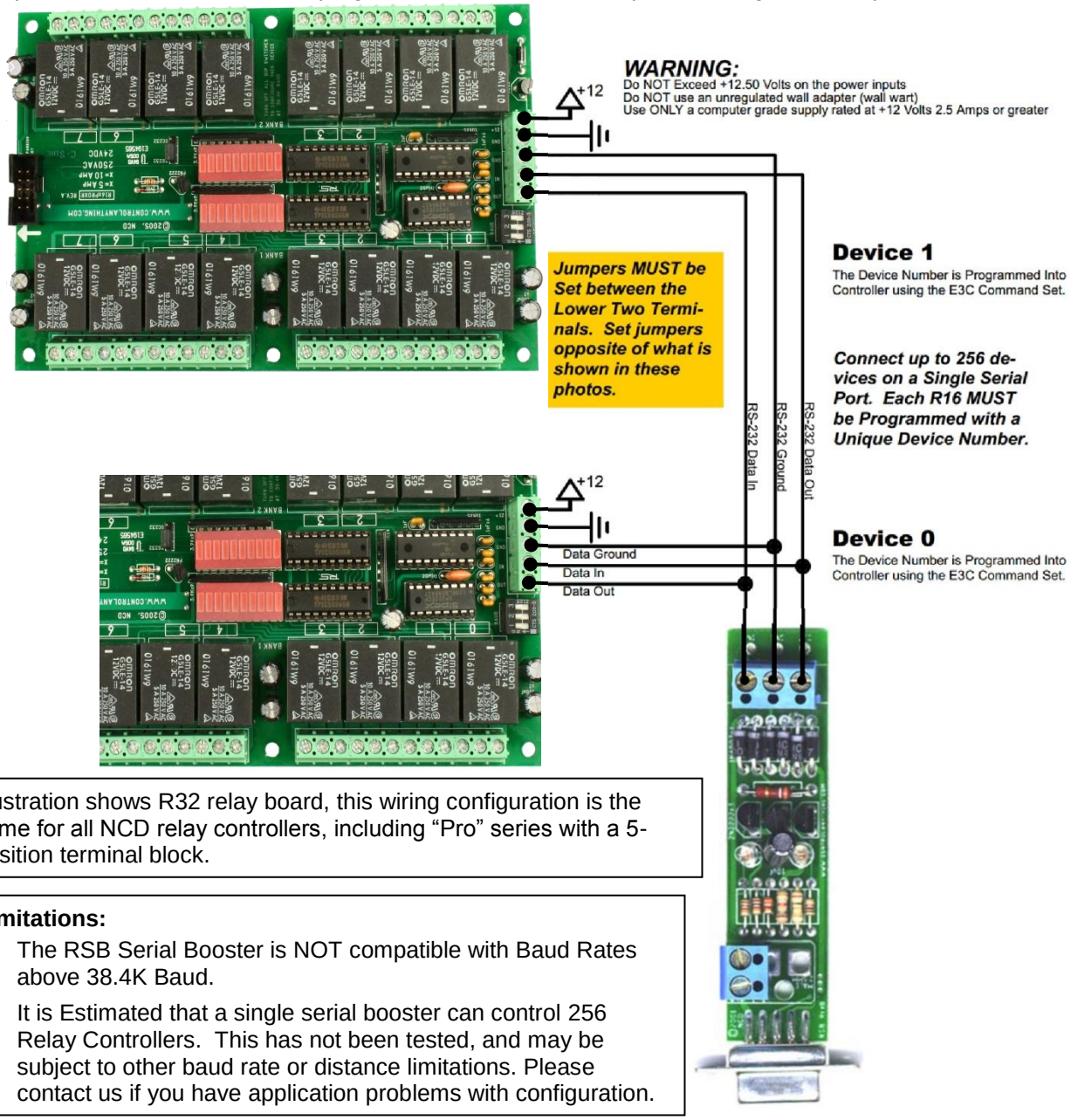


Illustration shows R32 relay board, this wiring configuration is the same for all NCD relay controllers, including "Pro" series with a 5-position terminal block.

Limitations:

1. The RSB Serial Booster is NOT compatible with Baud Rates above 38.4K Baud.
2. It is Estimated that a single serial booster can control 256 Relay Controllers. This has not been tested, and may be subject to other baud rate or distance limitations. Please contact us if you have application problems with configuration.

Sending Commands to the R4x/R8x Pro Relay Controllers

The R4x/R8x Pro is capable of sending and receiving data via RS-232 serial communications. The R4x/R8x Pro is compatible with just about any computer or microcontroller ever produced, including the Macintosh, Amiga, Basic Stamp, and of course, Windows and 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 the R4x/R8x Pro. Commands should be sent using ASCII character codes 0-255 rather than ASCII characters (A, B, C, etc.). *See "ASCII Codes vs. Characters" on the next page.* Most systems require you to open the appropriate serial port (COM port) prior to sending or receiving 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. In your program, you may want to ask the R4x/R8x Pro for the current status of the relays, just to confirm the activation. If so, your programming language will support commands for reading data from the serial port. For your convenience, we have provided several programming examples in Visual Basic 6 for controlling the R4x/R8x Pro. These examples should greatly speed development time. You may want to visit www.controleverything.com for the latest software and programming examples. Programming examples for the R4x/R8x Pro 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, at 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.

Chapter 4

ASCII Codes vs. Characters

The difference 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 work with numbers. With regard RS-232 data, the computer is only capable of sending and receiving numbers from 0-255. What confuses people is the simple idea that numbers 0-255 are assigned letters. For instance, the number 65 represent 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 character codes can be clearly defined as number 0-255. ASCII character 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 and numbers. If you wanted to send the word "hello" out of the serial port, it is easier to send the letters H, e, I, I, 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 where you want to speak to lots of outputs (which are numbered), inputs (which are also numbered), or control specific devices using their device number (from 0 to 255). 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 SECII character based, it may be difficult to generate the proper series of keystrokes that would be necessary to activate a particular function. Therefore, they 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.

Chapter 5

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. The R4x/R8x Pro relay controllers support the full set of E3C commands, plus a set of extended commands, plus a set of extended commands for storing and recalling the device number.

How does E3C Work?

First of all, each device must be assigned a device number from 0 to 255. The R4x/R8x Pro must be programmed with a device number, which is accomplished using the "Store Device Number" command shown below. 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 will ignore your commands. Note that E3C commands are never ignored by any device, regardless of the commands you send to the controller. The number to the left of each command indicated 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:

Enable All Devices

Tell 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
```

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 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
```

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
```

Control Anything...Anywhere.

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                          'Wait for R16 to acknowledge command
```

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
        DoEvent                               'Allow Windows to MultiTask
Until MSComm1 . InBufferCount > 0             'If the device Replies
GetDevice Number=Asc(MSComm1 . Input)         'Get Device Number from Buffer
```

Chapter 6

R4x/R8x Pro Command Set

The R4x/R8x supports an extensive command set, used to control relays, set operation modes, and store and recall relay status. Most users will not use many of the functions built into this controller. The best way to familiarize yourself with the capabilities is to carefully read through the command set in this section. The "plain English" examples provide a quick, easy to understand definition of what each command does.

The number to the left of each command indicated 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 at right. Not the command set for the R4x Pro is very similar to the command set for the R8x Pro. R8x Pro commands issue to the R4x will be ignored, allowing for easy future upgrade to the R8x Pro.

0-7 Turning Off Individual Relays

Turn Off an individual relay.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	0-7
Hex Values:	0xFE	0x00-0x07

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code:

```
Set Relay Status (Relay, Stat)      'Relay Parameter = 1 to 8
                                    'Stat Parameter = 1 to 9
If Stat = 0                         'Turn Off Relay
        MSComm1 . Output = Chr$(254)    'Enter Command Mode
        MSComm1 . Output = Chr$(Relay-1)    'Relay to Turn Off
```

8-15 Turning On Individual Relays

Turn On an individual relay.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	8-15
Hex Values:	0xFE	0x08-0x0F

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code:

```
Set Relay Status (Relay, Stat)        'Relay Parameter = 1 to 8
                                      'Stat Parameter = 1 to 9
Else (other than 0)                   'Relay to Turn On
    MSComm1 . Output = Chr$(254)      'Enter Command Mode
    MSComm1 . Output = Chr$(Relay+7)  'Relay to Turn On
```

16-23 Get the Status of an Individual Relay

This command allows you to read the On/Off status of an individual relay. 16 correspond to relay 1, 23 corresponds to relay 8. This command will return a 1 indicating the relay is ON or a 0 indicating the relay is OFF.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Value:	254	16-23
Hex Values:	0xFE	0x10 – 0x17

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code:

```
Get Relay Status (Relay)              'Relay Parameter – 1 to 8
MSComm1 . Output = Chr$(254)          'Enter Command Mode
MSComm1 . Output = Chr$(Relay+15)     'Get Status of One Relay
Do                                    'Wait for Device of Reply
    Do Events                         'Allow Windows to Multitask
Until MSComm1.InBufferCount > 0       'If the Device Replies
GetRelayStatus = Asc(MSComm1.Input)   'Get Status from Serial Buffer
Debug.Print GetRelayStatus            'Display in Immediate Window
```

24 Get the Status of All Relays

This Command allows you to get the status all relays at one time. A value of 0-255 is returned indicating the status of all 8 relays from the R8x Pro. A value of 0-15 is returned from the R4x Pro. The binary pattern of the value returned directly corresponds to the On/Off status of each relay.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	24
Hex Values:	0xFE	0x18

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Get All Relay Status (Relay)

```
MSComm1 . Output = Chr$(254)                    'Enter Command Mode
MSComm1 . Output = Chr$(24)                     'Get Status of all Relays
Do                                              'Wait for Device to Reply
        DoEvents                                'Allow Windows to Multitask
Until MSComm1.InBufferCount > 0                 'If the Device Replies
GetAllRelayStat = Asc(MSComm1.Input)            'Get Status from Serial Buffer
Debug.Print GetAllRelayStat                     'Display in Immediate Window
```

25 Store Relay Pattern as Power-up Default

This command allows you to define the On/Off status of all relays when power is first applied to the board. Use other commands to set the relays in the desired power-up state, then issue this command to store the current status of the relays as the power-up default.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	25
Hex Values:	0xFE	0x19

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Store Default

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(25)       'Store Power-up Default Status
```

26 Get the Power-up Default Relay Pattern

This command allows you to read the stored power-up default relay pattern. R4x Pro boards return a value of 0-15, R8x Pro board return a value of 0-255. The binary pattern of the value returned directly corresponds to the On/Off status of each relay.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	26
Hex Values:	0xFE	0x1A

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Get Default Status

```
MSComm1.Output = Chr$(254)                  'Enter Command Mode
MSComm1.Output = Chr$(26)                   'Get Status of all Relays
Do                                          'Wait for Device to Reply
        DoEvents                            'Allow Windows to Multitask
Until MSComm1.BufferCount > 0               'If the Device Replies
GetDefaultStatus = Asc(MSComm1.Input)       'Get Status from Serial Buffer
Debug.Print GetDefaultStatus                'Display in Immediate Window
```

27 Turn Reporting Mode ON

By default, Reporting Mode is Off. Reporting mode confirms the completion of most commands by sending an ASCII character code 85 back to the user. Reporting mode does NOT confirm the completion of E3C commands, or any command that sends data back to the user. Reporting mode is stored in non-volatile memory. Reporting mode slows communications and is only recommended in high-reliability installations.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	27
Hex Values:	0xFE	0x1B

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Store default

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(27)       'Turn On Reporting Mode
```

28 Turn Reporting Mode OFF

This command turns off the reporting function. Reporting mode is stored in non-volatile memory. This is the default setting.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	28
Hex Values:	0xFE	0x1C

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Get Default Status

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(28)       'Turn Off Reporting Mode
```

29 Turns All Relays OFF

Turns ALL relays Off.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	29
Hex Values:	0xFE	0x1D

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: All Relays Off

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(29)       'Turn Off All Relays
```

30 Turns All Relays ON

Turns ALL relays On.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	30
Hex Values:	0xFE	0x1E

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: All Relays On

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(30)       'Turn On All Relays
```

32 Reverse Relay Order

The Status of Relays 1-2-3-4-5-6-7-8 are reversed to 8-7-6-5-4-3-2-1. This command does not permanently reassign relays; it only copies the status of the relays when executed.

Send Bytes:	**Byte 1:**	**Byte2:**
Function:	Command	Command
Decimal Values:	254	32
Hex Values:	0xFE	0x20

Receive Byte: Decimal: 85
Hex: 0x55

Sampler Code: All Relays Off

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(32)       'Reverse Relay Order Command
```

33 Test Two Way Communication

This command can be used to test 2-way communication between the hose computer and the relay controller. When executed, the relay controller will send ACII character code 85 back to the user. This command should be used for initial installations if 2-way communication is required. It can also be sued to detect the presence of a relay controller on the serial port.

Send Bytes:	**Byte 1:**	**Byte 2:**
Function:	Command	Command
Decimal Values:	254	33
Hex Values:	0xFE	0x1A

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Get Default Status

```
MSComm1.Output = Chr$(254)              'Enter Command Mode
MSComm1.Output = Chr$(33)               'Request 2-Way Comm. Test
Do                                      'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
Until MSComm1.InBufferCount > 0         'If the Device Replies
Test2Way = Asc(MSComm1.Input)           'Get Status from Serial Buffer
Debug.Pring Test 2Way                   'Display in Immediate Window
```

40, 0-15 Set Status of All Relays (R4x Pro)

This command is used to set the status of all relays at one time. A single parameter is required. The equivalent binary pattern of the parameter is copied directly to the relays, instantly setting the On/Off status of all relays on the board.

Send Bytes:	**Byte 1:**	**Byte 2:**	**Byte 3:**
Function:	Command	Command	Parameter
Decimal Values:	254	40	0 -15
Hex Values:	0xFE	0x28	0x00 – 0x0F

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Set All Relays (Relay)

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(40)       'Set All Status Relay Command
MSComm1.Output = Chr$(Relay)    'Pattern to Set Relays To
```

40, 0-255 Set Status of All Relays (R8x Pro)

This command is used to set the status of all relays at one time. A single parameter is required. The equivalent binary pattern of the parameter is copied directly to the relays, instantly setting the On/Off status of all relays on the board.

Send Bytes:	**Byte 1:**	**Byte 2:**	**Byte 3:**
Function:	Command	Command	Parameter
Decimal Values:	254	40	0 – 255
Hex Values:	0xFE	0x28	0x00 – 0xFF

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Set All Relays (Relay)

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(32)       'Set All Status Relay Command
MSComm1.Output = Chr$(Relay)    'Pattern to Set Relays To
```

41, 0-15 Program Emulation device Number

When the R4x/R8x Pro is NOT in emulation mode, this command can be used, along with its parameter of 0-15, to define the device number for use in emulation mode. One programmed, you must remove power from the board and set the board to Emulation mode.

Send Bytes:	**Byte 1:**	**Byte 2:**	**Byte 3:**
Function:	Command	Command	Parameter
Decimal Values:	254	41	0 – 15
Hex Values:	0xFE	0x29	0x00 – 0x0F

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Program Emulation Device Number (Device)

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(33)       'Program Emu Device Number
MSComm1.Output = Chr$(Device)   'Device Number to Program
```

42, 0-15 Store Relay Pattern in Memory Bank

This command stores the current On/Off setting of all relays into a memory bank (0-15). This command is useful for creating macros or for making sure certain relays are never activated simultaneously.

Send Bytes:	**Byte 1:**	**Byte 2:**	**Byte 3:**
Function:	Command	Command	Parameter
Decimal Values:	254	42	0 – 15
Hex Values:	0xFE	0x2A	0x00 – 0x0F

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Store Patten In Bank (Bank)

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(42)       'Store Pattern in Bank
MSComm1.Output = Chr$(Bank)     'Mem. Bank to Store Pattern In
```

43, 0-15 Recall Relay Pattern from Memory Bank

This command recalls a stored relay pattern from the user selected memory bank (0-15) and update all relays on the board to the setting defined by command 42 above.

Send Bytes:	**Byte 1:**	**Byte 2:**	**Byte 3:**
Function:	Command	Command	Parameter
Decimal Values:	254	43	0 – 15
Hex Values:	0xFE	0x2B	0x00 – 0x0F

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Recall Pattern in Bank (Bank)

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(43)       'Recall Pattern from Bank
MSComm1.Output = Chr$(Bank)     'Mem. Bank to Get Pattern From
```

44, 0-3 (R4x Pro);
44, 0-7 R8x Pro)Select a Relay for Activation

This command turns Off all relays and then turns On the selected relay only. This command performs a safe "Break Before Make", ensuring that no two relays are ever activated at the same time.

Send Bytes:	**Byte 1:**	**Byte 2:**	**Byte3:**	**Byte 3:**
Function:	Command	Command	Parameter (R4x Pro)	Parameter
Decimal Values:	254	44	0 – 3	0 – 7
Hex Values:	0xFE	0x2C	0x00 – 0x03	0x00 – 0x07

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Relay Select (Relay)

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(44)       'Select Relay Command
MSComm1.Output = Chr$(Bank)     'Relay to Select
```

45, 0-3 (R4x Pro);
45, 0-7 Select a Relay for De-Activation

This command turns On all relays and then turns Off the selected relay only. This command performs a safe "Make Before Brake", ensuring that no two relays are ever deactivated at the same time.

Send Bytes:	**Byte 1:**	**Byte 2:**	**Byte 3:**	**Byte 3:**
Function:	Command	Command	Parameter (R4x Pro)	Parameter (R8x Pro)
Decimal Values:	254	45	0 – 3	0 – 7
Hex Values:	0xFE	0x2D	0x00 – 0x03	0x00 – 0x07

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Relay Deselect (Relay)

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(45)       'Deselect Relay Command
MSComm1.Output = Chr$(Bank)     'Relay to Deselect
```

46, 0-3 (R4x Pro);
45, 0-7 (R8x Pro) Toggle the Status of a Relay

This command reverses the current On/Off status of the selected relay.

Send Bytes:	**Byte 1:**	**Byte 2:**	**Byte 3:**	**Byte 4:**
Function:	Command	Command	Parameter (R4x Pro)	Parameter (R8x Pro)
Decimal Values:	254	46	0 – 3	0 – 7
Hex Values:	0xFE	0x2E	0x00 – 0x03	0x00 – 0x07

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Relay Deselect (Relay)

```
MSComm1.Output = Chr$(254)      'Enter Command Mode
MSComm1.Output = Chr$(46)       'Deselect Relay Command
MSComm1.Output = Chr$(Bank)     'Relay to Deselect
```

Relay Timing Functions

The R4x/R8x Pro Series Relay controllers have a timer function used to activate one or more relays for user specified period of time form 10 Milliseconds to 32 Seconds. Timing is accurate to within 5% of the User specified period. The timer functions require a Time parameter in the commands shown below. Use the following guide to determine the appropriate value for the Time parameter.

The Time Variable sets 3 functions:

Feedback:	0=Off
	128=On, Sends 85 to Host when Timer is Finished
Duration Interval:	0=(10 milliseconds x Duration) + 10
	64=(.5 seconds x Duration) + .5
Duration:	0 to 63

To use the different modes of the timer, simply add together the values for each parameter. Feed the total into the TIME variable above. Then select the relay to apply the timer to.

Examples:

Time=0	10 Millisecond Timer with No Feedback
Time=4	50 Millisecond Timer with No Feedback
Time=132	50 Millisecond Timer with No Feedback
Time=192	5 Second Timer with Feedback
Time=73	5 Second Timer with No Feedback
Time=201	5 Second Timer with Feedback

Note: Timer uses All CPU Resources

Commands cannot be sent to the relay controller while the timer is in operation. Any commands received during this period of time will be ignored. Use the feedback function (which is part of the time parameter) to signal the host when the timer has completed its cycle or use the Test 2 Way command to query the relay controller.

47, Time (0-255), Relay (0-3) – R4x Pro;
47, Time (0-255), Relay (0-7) – R8x Pro Activate a Single Relay on a Timer

This command turns On all relays and then turns off the selected relay only. This command performs a safe "Make Before Brake", ensuring that no two relays are ever de-activated at the same time.

Send Bytes:	***Byte 1:***	***Byte 2:***	***Byte 3:***	***Byte 4:***	***Byte 4:***
Function:	Command	Command	Timer	Parameter (R4x Pro)	Parameter (R8x Pro)
Decimal Values:	254	47	0 – 255	0 – 3	0 – 7
Hex Values:	0xFE	0x2F	0x00 – 0xFF	0x00 – 0x03	0x00 – 0x07

Receive Byte: Decimal: 85
Hex: 0x55

Sample Code: Set Relay Timer (Timer, Relay)

```
MSComm1.Output = Chr$(254)          'Enter Command Mode
MSComm1.Output = Chr$(47)           'Set Timer for a Relay Command
MSComm1.Output = Chr$(Bank)         'Specify Time Period
MSComm1.Output = Chr$(Relay)        'Replay to Activate
   'Debug. Print GetFeedback        'Optional if Feedback is Used
```

48, Timer (0-255), RPOn (0-15), RP Off (0-15) – R4x Pro;
48, Timer (0-255), RPOn (0-255) – R8x Pro Relay Patten Select on a Timer

This command reverses the current On/Off status of the selected relay.

Send Bytes:	**Byte 1:**	**Byte 2:**	**Byte 3:**	**Byte 4:**	**Byte 4:**
Function:	Command	Command	Timer	RP On/Off (R4x Pro)	RP On/Off (R8x Pro)
Decimal Values:	254	48	0 – 255	0 – 15	0 – 255
Hex Values:	0xFE	0x30	0x00 – 0xFF	0x00 – 0x0F	0x00 – 0xFF

Receive Byte: Decimal: 85
Hex 0x55

Sample Code: Set Multi Relay Timer (Timer, RP On, RP Off)

```
MSComm1.Output = Chr$(254)          'Enter Command Mode
MSComm1.Output = Chr$(48)           'Set Timer for All Relays
MSComm1.Output = Chr$(Timer)        'Specify Time Period
MSComm1 Out = Chr$(RPOn)            'Timer Start Relay Pattern
MSComm1.Output = Chr$(RPOff)        'Timer Stop Relay Pattern
   'Debug. Print GetFeedback        'Optional if Feedback is Used
```

Control Anything...Anywhere.

Electrical Ratings and Switching Characteristics			
Model	**Rating**	**Relay Type**	**Notes**
R41DPDT	3A 120VAC / 3A 20VDC	DPDT	DPDT: Two Switches Per Relay Not for Use with Inductive Loads
R43DPDT	3A 250VAC / 3A 30VDC	DPDT	DPDT: Two Switches Per Relay
R45DPDT	5A 250VAC / 5A 30VDC	DPDT	DPDT: Two Switches Per Relay
R45 Pro	10A 250VAC / 5A 100VDC	SPDT	Relay ratings for this model are Absolute Maximum, Not for Sustained Constant Use. Divide all ratings by 2 or constant operation.
R410 Pro	10A 250VAC / 8A 30VDC	SPDT	
R420	20A 240VAC / 20A 28VDC	SPDT	Common to Normally Closed Terminal is rated at 10A 240VAC / 10A 28 VDC Flange type connections to relay.
R430	30A 250VAC / 20A 28VDC	SPST	SPST: Common and Normally Open Terminals Only Flange Type Connections to Relay
R81DPDT	3A 120VAC / 3A 20VDC	DPDT	DPDT: Two Switches per Relay Not for use with Inductive Loads
R83DPDT	3A 250VAC / 3A 30VDC	DPDT	DPDT: Two Switches per Relay
R85DPDT	5A 250VAC / 5A 30VDC	DPDT	DPDT: Two Switches per Relay
R85 Pro	10A 250VAC / 5A 100VDC	SPDT	Relay ratings for this model are Absolute Maximum, Not for Sustained Constant Use. Divide all ratings by 2 for Constant Operation.
R810 Pro	10A 254VAC / 8A 30VDC	SPDT	
R820	20A 240VAC / 20A 28VDC	SPDT	Common to Normally Closed Terminal is rated at 10A 240VAC / 10A 28VDC Flange Type Connections to Relay.
R830	30A 250VAC / 20A 28VDC	SPST	SPST Common and Normally Open Terminals Only Flange Type Connections to Relay.
All ratings above are for Resistive Loads. Divide current switching capabilities by 2 for Inductive Loads. Inductive loads not recommended for 1A DPDT relay controller models			

Relay Types	
SPST	Single Pole Single Throw: This relay is the most basic type of switch available. SPST relays have only two connections. Typically, an SPST relay shorts these two connections together when this relay is activated. When the relay is OFF, this connections return to their original OFF state (disconnected).
SPDT	Single Pole Double Throw: This is the most popular type of relay used in our relay controller designs. Each relay consists of a Normally Open (NO), Normally Closed (NC), and a Common (COM). When the relay is OFF, NC and COM are connected together. When the relay is turned ON, NC is disconnected from COM and NO and COM are connected together.
DPDT	Same as SPDT above, but there are two of these switches in a single relay that are activated/deactivated at the same time.

Characteristic Data		
Relay Activation Time	>5 ms	<15 ms
Relay Deactivation Time	>5 ms	<20 ms
Activations Per Second at 9600 Baud using Emulation Mode	N/A	960
Activations Per Second at 9600 Baud using “Pro” Command Set	N/A	480
Activations Per Second at 19.2K Baud using Emulation Mode	N/A	1,920
Activations Per Second at 19.2K Baud using “Pro” Command Set	N/A	960
Communication Distance from PC Without Boosting Signal 120 Baud*	N/A	Approx. 2,000 ft.
Communication Distance from PC Without Boosting Signal 2400 Baud*	N/A	Approx. 1,200 ft.
Communication Distance from PC Without Boosting Signal 9600 Baud*	N/A	Approx. 600 ft.
Communication Distance from PC Without Boosting Signal 19.2K Baud *	N/A	Approx. 200 ft.
Maximum allowed Activation Time per Relay (Relay held in ON state)	N/A	Unlimited
Expected Operational Life, Non-DPDT Models	>10,000,000 Cycles	N/A
Expected Operational Life, DPDT Models	>2,000,000 Cycles	N/A
Typical Cycles Per Minute	N/A	1,800
*Assumes good quality low-capacitive wire, twisted pair preferred. Note that distances are estimated. Typically, longer distances are easily achieved.		

Chapter 7

Troubleshooting

Unreliable or Unpredictable Operation or No Response After Sent Commands

- In nearly 100% of the cases we have seen, unreliable or unpredictable operation is caused by improper setting of the PC/MAC jumper. Make sure this jumper is set in the MAC position if you are using a laptop computer of any kind, a microcontroller, or an Apple Macintosh product. Set it in the PC setting is using a Desktop PC. Refer to page 4 for correct settings and information on the PC/MAC jumper.

Nothing Happens when Relay Control Commands are Sent

- Another simple cause can be incorrect connection of the relay controller to the serial port of your computer or improper COM settings in software. Keep in mind, this manual indicates the RS-232 data INPUT of the relay controller. This means you must connect the RS-232 Data Output of your computer to the RS-232 Data INPUT of the relay controller. Note that the Data Receive LED on this device is intelligent. It will ONLY light up if it receives a valid ASCII character code 254 (enter command mode) preceding each command. Send 254 constantly to the board to make the LED flash.

No 2 Way Communication

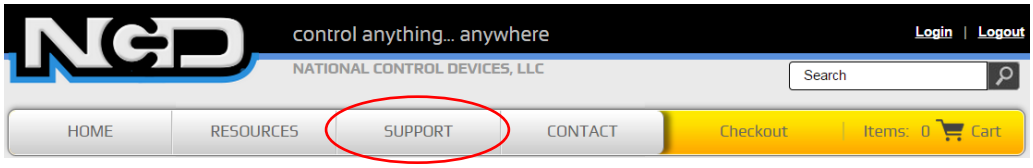
- Two-Way communication can be compromised by an incorrect jumper setting and/or improper wiring. Make sure the TTL/OC jumper is set in the TTL position for any kind of PC system (desktop or laptop). Also, note that the R4x/R8x Pro relay controllers use a hybrid optoisolation design. For two-way communication to work properly, you MUST connect the RS-232 ground to the Power Supply ground of the board.

Control Anything...Anywhere.

Chapter 8

Technical Support

Technical support is available through our website, controlanything.com. **Support** is the way we connect NCD engineers to our customers.



*Click on the **Support** tab at the top of any page on our website to be taken to the **Forum** page. Here you can publicly post or review problems that customers have had, and learn about 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 **Support** 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

Like "National Control Devices" on Facebook, and follow us on Twitter @ControlAnything.

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

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