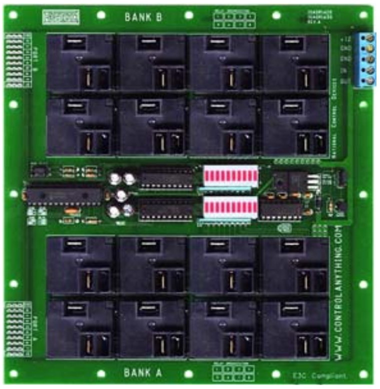
IOADR8x IOADR16x

RS-232 E3C Networkable *Hybrid* I/O + A/D + LCD + Keypad + Terminal + Relay Controller



Hybrid Relay + I/O Controllers

NCD Hybrid Controllers combine the features and command set of our IOAD168 and R8x Pro relay controllers. A Hybrid Controller is a Digital Input/Output controller with a 5-channel 8 or 10-Bit A/D converter with 8 or 16 integrated relays (5, 10, 20, and 30-Amp Version are available). NCD Hybrid Controllers connect to the serial port of any computer or microcontroller and responds to commands from the user at baud rates up to 38.4K.

NCD Hybrid Controllers were designed to offer desktops, laptops, and embedded computers access to low-level digital I/O and A/D functions integrated with powerful relay control commands, with minimal user-programming. Visual Basic program examples are provided in this manual and are available for download at ***www.controlanything.com***.

NCD Hybrid Controllers supports commands for setting the on/off state of individual I/O lines, setting the state of all I/O lines at once (serial-to-parallel conversion), reading the state of I/O lines (single input or parallel-to-serial encoding), reading analog values, and mixing inputs and outputs on a single I/O data port. In addition, a large number of character LCD, Text Terminal, and Keypad Entry commands make it easy to interface a Hybrid controller in just about any application. Integrated with the features and command set of our Popular R8x Pro Relay Controllers, NCD Hybrid Controllers offer the best of all worlds in a single, easy-to-use, controller.

NCD Hybrid Controllers are E3C compliant, allowing 256 NCD devices to be attached to a single RS-232 serial port in any combination. The NCD Hybrid Controller firmware is upgradeable, allowing for future upgrades to add new features at no additional cost.

NCD Hybrid Controllers are compatible with many expansion modules (available separately) for adding I/O functions to the hybrid controller. See below for current compatibility list.

This Manual Covers the Following NCD Hybrid Controllers:
IOADR85: Hybrid + 8-Relay 5-Amp Controller
IOADR810: Hybrid + 8-Relay 10-Amp Controller
IOADR820: Hybrid + 8-Relay 20-Amp Controller
IOADR830: Hybrid + 8-Relay 30-Amp Controller
IOADR165: Hybrid + 16-Relay 5-Amp Controller
IOADR1610: Hybrid + 16-Relay 10-Amp Controller
IOADR1620: Hybrid + 16-Relay 20-Amp Controller
IOADR1630: Hybrid + 16-Relay 30-Amp Controller

These Devices will be referred to as Hybrid Controllers in this manual. The devices above are a Hybrid Combination of our popular IOAD168 and our R8x Pro Relay Controllers. The electronics and firmware of these designs have been merged into a single controller to offer the most powerful features available in a single low-cost design. Existing customer software can be re-written in a matter of minutes to take advantage of our new Hybrid controllers because of a highly standardized command set.

E3C Compliant Command Set: Control 256 Devices from a Single Serial Port
User-Selectable Communication Rates from 2400, 9600, 19.2K, and 38.4K Baud
Integrated 8-Relay or 16-Relay Controller with 5, 10, 20, or 30-Amp Relays
IOAD168 Command Set + R8x Pro Command Set x 2
Device Enabled/Power LED
+12 Volt DC Operation
Data Receive LED
Dual 8-Bit I/O Data Port
5 Channel 8-Bit or 10-Bit Software Selectable A/D Converter
O.C. RS-232 Communication for Networking Multiple Devices
Powerful ASCII Character Code Based Command Set
Compatible with ANY Computer or Microcontroller
Expansion Module Upgradeable
Firmware Upgradeable FLASH Memory

Compatible Expansion Modules

8FET High Power 8-Channel FET Driver
Useful for Direct Control of Relays, Stepper Motors, or Other High-Power Devices. Connect one of these to the IO Data bus on any controller, or Connect 3 of these to a single PAR24 Expansion Module. Compatible with All 3 8-bit Data Ports on the IOAD168.

AD1216 12-Bit 16-Channel Analog to Digital Converter Expansion Module
Up to 3 AD1216s can share a Single IO Data Bus. Connect up to 6 AD1216s on a Single IOAD168 controller, allowing software monitoring of 96 analog data channels. Not compatible with the Output Only port on the IOAD168.

PAR24 24-Bit Parallel Output Expansion Module
Each PAR24 Adds 24 TTL Outputs to the IO Data Bus. Connect up to 4 of these to a single IOAD168 controller. Not compatible with the Output Only port on the IOAD168.

TRDVR Low Power 8-Channel O.C. Transistor Driver
Useful for Direct Control of small Relays, LEDs, or other low current devices. Connect up to three of these to the IOAD168, or Connect 3 of these to a single PAR24 Expansion Module.

TESTL 10-LED Output Test Module
Connect up to three of these to the IOAD168, or Connect 3 of these to a single PAR24 Expansion Module.

TESTD 8-DIP Switch Input Test Module
Connect up to two of these to the IOAD168. Not compatible with the Output Only port on the IOAD168.

Hybrid:

Merging Technologies

NCD Hybrid Controllers are an electronics and firmware amalgamation of our popular IOAD168 and R8x Pro Relay controllers. The command set and capabilities of both devices have been integrated into a single design effectively combining the best of both designs into a single integrated package. The command structure of a Hybrid controller is 99% identical to that of our IOAD168 and R8x Pro relay controllers. The R8x Pro Relay Controller Command set is accessed by “Branching”, which is fully described in the relay controller portion of this manual. Simply Put, a single IOAD168 command has been added to allow the user access to the R8x Pro set of commands.

Some commands have been added and omitted to eliminate command “overlap” between the IOAD168 and the R8x Pro command set. Omitted commands no-longer exist in firmware. If an omitted command is issued, the firmware will exit command mode and begin waiting for another command.

Manual Editing Notes:

This manual is a mix of the IOAD168 and R8x Pro relay controllers. Some photos of these controllers may be used for illustration purposes only, but will apply directly to our Hybrid Controllers. Page 3 illustrates actual locations of I/O data ports and Relay Banks on a Hybrid controller. The structure of our data ports and relay banks have been retained as closely as possible to existing designs.

Hardware Configurations:

NCD Hybrid controllers with eight integrated relays have the following Ports available:
Port A: 8-Bit I/O + 8-Bit or 10-Bit A/D
Port B: 8-Bit I/O Only
Port C: 8-Bit TTL Output Port
BANK A: 8-Relays
R8x Pro Command Set for Controlling 8 Relays

NCD Hybrid controllers with sixteen integrated relays have the following Ports available:
Port A: 8-Bit I/O + 5-Channel 8-Bit or 10-Bit A/D
Port B: 8-Bit I/O Only
BANK A: 8 Relays
BANK B: 8 Relays
One additional command that directs R8x Pro Command Set to Bank A or Bank B Relays has been added. Note the R8x Pro Command Set cannot control all 16 relays using a single command. Commands must be directed to a bank of eight relays. Extended commands have been added to control all relays and to read the status of all relays at once.
Note: Port C has been omitted from hardware and firmware on all Hybrid controllers with sixteen integrated relays.

5-Year Repair or Replace Warranty

Warranty
NCD Warrants its products against defects in materials and workmanship for a period of 5 years. If you discover a defect, NCD will, at its option, repair, replace, or refund the purchase price. Simply return the product with a description of the problem and a copy of your invoice (if you do not have your invoice, please include your name and telephone number). We will return your product, or its replacement, via UPS Ground Service in the US and Canada Only. Additional shipping charges will apply to international customers.

This warranty does not apply if the product has been modified or damaged by accident, abuse, or misuse.

30-Day Money-Back Guarantee
If, within 30 days of having received your product, you find that it does not suit your needs, you may return it for a refund. NCD will refund the purchase price of the product, excluding shipping/handling costs. This guarantee does not apply if the product has been altered or damaged.

Copyrights and Trademarks

Disclaimer of Liability
NCD is not responsible for special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, goodwill, damage to or replacement of equipment or property, and any costs or recovering, reprogramming, or reproducing any data stored in or used with NCD products.

Technical Assistance
Technical questions should be e-mailed to Ryan Sheldon at ryan@controlanything.com. Technical questions submitted via e-mail are answered up to 20 times daily. Technical support is also available by calling (417) 646-5644 from 9:00 A.M. to 4:00 P.M. Central Standard Time.

NCD Contact Information

Mailing Address:
National Control Devices
P.O. Box 455
Osceola, MO 64776

Telephone:
(417) 646-5644

FAX:
(417) 646-8302

Internet:
ryan@controlanything.com
www.controlanything.com
www.controleverything.com

IMPORTANT POWER SUPPLY REQUIREMENTS

1) DO NOT USE A WALL WART TYPE UNREGULATED POWER SUPPLY.
2) USE ONLY A COMPUTER GRADE REGULATED SWITCHER SUPPLY RATED AT +5 or 12 VOLTS DC, 200 ma OR GREATER.
3) USE A SUPPLY RATED FOR MORE AMPERAGE WHEN POWERING MULTIPLE BOARDS.
4) DC POWER SHOULD NEVER TRAVEL GREATER THAN 20 FEET. A SEPARATE POWER SUPPLY SHOULD BE USED FOR EACH CONTROLLER IF CONTROLLERS ARE NOT LOCATED WITHIN 20 FEET OF EACH OTHER.
5) THE IOAD168 CAN BE USED IN 12 VOLT AUTOMOTIVE ELECTICAL SYSTEMS WHEN SET TO +12 VOLT MODE. SOME EXPANSION MODULES SHOULD NOT BE USED WHEN SET TO +12 VOLTS.

Chaining Multiple IOAD168s on a Single RS-232 Serial Port

Step 1:
Connect Each IOAD168 to your computer by itself and program each board with a unique device number from 0-255.

Step 2:
It would be helpful to use an NCD Quick Start Kit (Part#: QS5-F6) to Apply Power and Data to the First IOAD168.

Step 3:
Connect the V+, GND, In and OUT lines (and optionally the RESET line) of the first board to the V+, GND, In and OUT lines of next board. Keep chaining IOAD168s together in this manner until you have enough controllers attached to meet your needs.

Step 4:
Remove the TERM jumper on all IOAD168 boards except for the board that is wired closest to your computer.

Step 5:
Use E3C command 252 to select an IOAD168 board to control using the programmed device number. All subsequent commands will be seen only by the board you have selected with this command. Use command 252 again to switch to a different board.

Status LED:

When power is first applied to the IOAD168, the power LED will glow brightly. If an E3C command is used to disable the IOAD168, the power LED will glow dimly. If the LED is not on at all, the IOAD168 is not getting any power. If the LED glows dimly when power is first applied to the board, the IOAD168 CPU is damaged.

RS-232 Communications

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

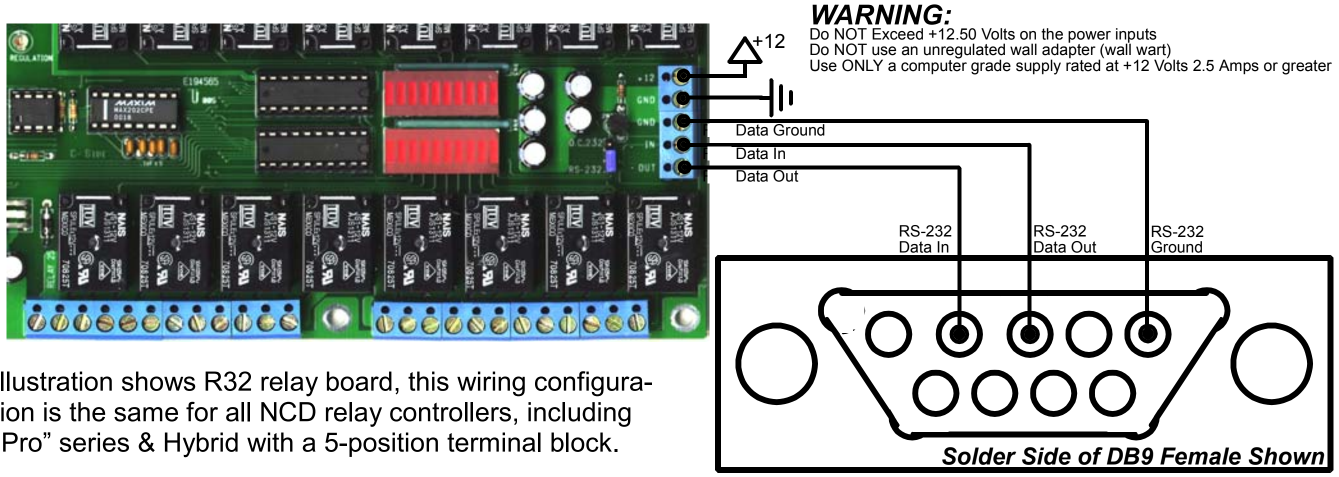
Dimension Information Begins on Page 23 of this Manual.

Two-Way Communication:

NCD Hybrid Controllers support two-way communication for confirming the receipt of commands and for reporting the status of the relays back to the host computer.

NCD Hybrid Controller should be connected as shown below when using this device for the first time. Even if you plan to connect several controllers to a single serial port, this wiring diagram must first be used to program the device number into the controller.

Visual Basic Example Programs expect this wiring configuration.



WARNING:

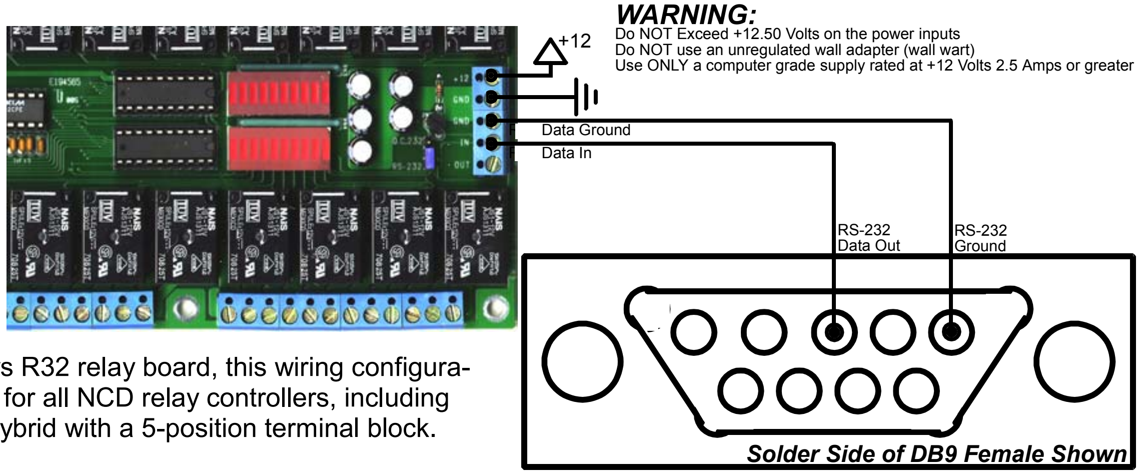
Do NOT Exceed +12.50 Volts on the power inputs
Do NOT use an unregulated wall adapter (wall wart)
Use ONLY a computer grade supply rated at +12 Volts 2.5 Amps or greater

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

One-Way Communication:

NCD Hybrid Controllers 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 Hybrid much faster, but it is impossible to ask the controller for the status of relays when wired as shown below.



WARNING:

Do NOT Exceed +12.50 Volts on the power inputs
Do NOT use an unregulated wall adapter (wall wart)
Use ONLY a computer grade supply rated at +12 Volts 2.5 Amps or greater

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

Sending Commands to the Hybrid

NCD Hybrid Controllers are capable of sending and receiving data via RS-232 serial communications. A Hybrid controller 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 most NCD devices. Commands should be sent using ASCII character codes 0-255 rather than ASCII characters (A, B, C etc.). See "ASCII Codes vs. Characters" on this 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 Hybrid for the current status of the relays, just to confirm their 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 our Hybrid controllers. 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 our Hybrid controllers 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.

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 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 ASCII 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.

The E3C Command Set: Software Control of Multiple NCD Devices

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. NCD Hybrid Controllers support 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. NCD Hybrid Controllers 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 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 at right.

The E3C Command Set

248 Enable All Devices:
Tells all devices to respond to your commands.

249 Disable All Devices:
Tells all devices to ignore your commands.

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

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

252 Enable Selected Device Only:
Tells a specific device to listen to your commands, all other devices will ignore your commands.

253 Disable a Selected Device Only:
Tells a specific device to ignore your commands, all others will listen.

E3C Visual Basic Programming Examples

The E3C command set is easily used from any programming language that supports serial communication. The following Visual Basic 6 Example source code demonstrates subroutines that can be used to control which devices will listen and which devices will ignore your commands.

Most commands issued to the Hybrid controler are acknowledged by sending ASCII character code 85 back to the host computer (when reporting is turned on). E3C commands are not acknowledged regardless of the reporting mode.

Sample Code: The E3C Command Set

```
Public Sub EnableAllDevices()
    'Enable All E3C Devices
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(248)    'E3C Enable All Device Command
End Sub

Public Sub DisableAllDevices()
    'Disable All E3C Devices
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(249)    'E3C Disable All Device Command
End Sub

Public Sub EnableSpecificDevice(Device)
    'Enable A Specific E3C Devices, Other Devices will be unchanged
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(250)    'E3C Disable Specific Device Command
    MSComm1.Output = Chr$(Device) 'Device Number that will be Disabled
End Sub

Public Sub DisableSpecificDevice(Device)
    'Disable A Specific E3C Devices, 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 Disabled
End Sub

Public Sub DisableAllDevicesExcept(Device)
    '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
End Sub

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

The Hybrid Command Set and Controller Configuration

NCD Hybrid Controllers supports an extensive command set, used to control IO lines, set operation modes, and store and recall IO status lines. 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 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 at right.

Test 2-Way Communications

0, 0 Test 2-Way Communication

This command can be used to test 2-way communication between the host computer and the relay controller. When executed, the relay controller will send ASCII 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 used to detect the presence of a relay controller on the serial port.

Default Settings

0, 1 Default Settings

This command can be used to restore all configuration parameters to factory default settings. DO NOT ISSUE THIS COMMAND IF MORE THAN ONE HYBRID OR IOAD168 CONTROLLER IS ATTACHED TO THE SAME SERIAL PORT OR ALL CONTROLLERS WILL BE PROGRAMMED WITH FACTORY DEFAULT VALUES. When this command is sent, parameters are changed to the following default values:

Default Keypad Settings

1	2	3	A
4	5	6	B
7	8	9	C
*	0	#	D

Default PIN Character = X

The Hybrid has the ability to control character LCD and VF displays. In addition, keystrokes from the Keypad can be sent directly to the display screen. In some cases, such as a PIN or PASSWORD request, the characters should be hidden from other viewers, but acknowledged by the display. The default character used to hide the users input is "X".

Default Character Delay = 150

The ***Character Delay*** value sets the delay between bytes of data sent from the controller to the user. A value of 150 is very safe for slow computers. If you are using a fast computer, two-way communication speed can be significantly increased by lowering the character delay value. However, a value set too low can generate communication errors.

Default E3C Device Number = 0

Up to 256 E3C compliant devices can share a single serial port in any combination. The E3C device number is used to define the "location" of the Hybrid on the E3C network.

When this command has completed programming of default values, the controller will send ASCII character code 85 back to the user. At this point, power should be removed and re-applied to the controller for all changes to take effect.

Visual Basic Programming Examples

A Visual Basic 6 programming examples are provided in the following pages to assist in the development of software for controlling the IOAD168. Additional source code can be found on our web site at www.controlanything.com.

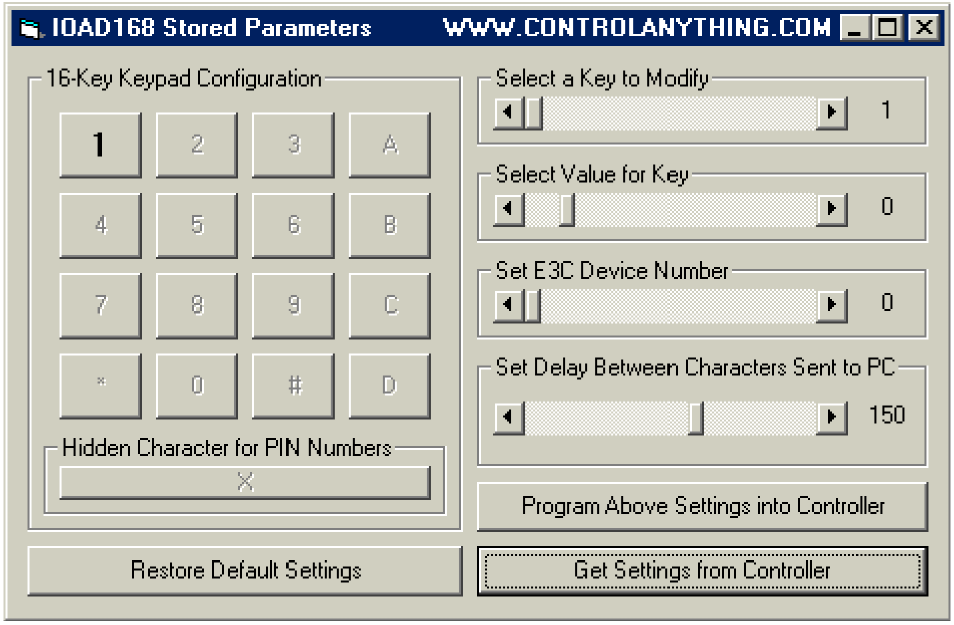
Sample Code: Test 2-Way Communications

```
Public Function Test2Way
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(0)            'Enter Setup Mode
    MSComm1.Output = Chr$(0)            '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 Controller
    Debug.Print Test2Way                'Display in Immediate Window
End Sub
```

Sample Code: Default Settings

```
Public Sub Default
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(0)            'Enter Setup Mode
    MSComm1.Output = Chr$(1)            '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 Controller
    Debug.Print Test2Way                'Display in Immediate Window
End Sub
```

The Hybrid Command Set and Controller Configuration


IOAD168 Stored Parameters
WWW.CONTROLANYTHING.COM
16-Key Keypad Configuration
1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
Hidden Character for PIN Numbers
Program Above Settings into Controller
Restore Default Settings
Get Settings from Controller

Select a Key to Modify: 1

Select Value for Key: 0

Set E3C Device Number: 0

Set Delay Between Characters Sent to PC: 150

Hidden Character for PIN Numbers: X

The Hybrid can be programmed with several parameters either by the user or by using our compiled Hybrid (IOAD168) configuration program. The easiest way to configure the controller is to use our compiled configuration program, which provides a simple visual interface for permanently configuring the IOAD168 or Hybrid controller.

Maximizing Performance

The "Set Delay Between Characters" slider should be decreased for better performance. High speed compiled programs may even allow for a "0" setting, making communication from controller to PC MUCH FASTER.

Programming Keypad Keys:

Use the "Select a Key to Modify" Slider to choose a button to change. Next, use the "Select Value for Key" slider to change the button. The "Hidden Character" is the character this displayed in place of actual key presses to hide the feedback from other viewers (when used with a character display).

When finished, select "Program Settings into Controller".

Running Compiled Program Examples

Please examine our complete library of software examples for the IOAD168 or Hybrid Controllers at www.controlanything.com. The examples shown are code fragments only and require other components to operate properly. The configuration program used to set the parameters of the Hybrid (or IOAD168) is provided with complete source code (viewable in any text editor), and as a set of four compiled applications (executable .EXE programs). If you choose to run the COMPILED (.EXE) version, set the Hybrid (or IOAD168) controller for 38.4K baud. Then select the application suited for your system (one for each of the four COM ports). If you have difficulty running the compiled programs, go to www.controlanything.com and visit the graphic displays section of our web site. Next, locate, download, and install the file: Image Loader Utility Version 1.01 or newer (ILUV101.ZIP). This will install the necessary components on your system required to run all Visual Basic program examples that we have compiled into .EXE applications.

Programming and Retrieving Configuration

The code fragments below show a general example for storing and retrieving configuration data into and out of the Hybrid (or IOAD168). All parameters are stored and retrieved sequentially. Pay attention to the "GetData" functions in these code examples. These functions are used to keep data synchronized between the computer and the controller. Any time a "GetData" function is encountered below, the program will wait for the Hybrid (or IOAD168) to send ASCII character code 85 back to the computer. The easiest way to configure the Hybrid (or IOAD168) is to use a Windows based computer and a Quick Start kit. Make sure only ONE controller is attached to the serial port when programming parameters or all devices will be programmed with the same configuration data.

Sample Code Fragment: Programming Settings

```
Private Sub ProgramSettings_Click()

    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(0)    'Enter Setup Command
    MSComm1.Output = Chr$(2)    'Program Settings Command
    Debug.Print GetData         'Wait for Command to Begin
    '1 = E3C Device Number
    MSComm1.Output = Chr$(HScroll3.Value)

    GetData                     'Wait for Parameter to Store
    '2 = Character Delay Value (0-255) 255 = slow
    MSComm1.Output = Chr$(CDEL.Value)
    GetData                     'Wait for Parameter to Store
    'Program the Keypad and Hidden PIN Characters
    For n = 0 To 16
       MSComm1.Output = Chr$(Asc(Key(n).Caption))
       For del = 0 To 250: DoEvents: Next del
       GetData                  'Wait for Parameter to Store
    Next n
End Sub

Public Function GetData()
       t = 0
        Do
            t = t + 1
            If t > 10000 Then GoTo ext

            DoEvents
        Loop Until MSComm1.InBufferCount > 0
        GetData = Asc(MSComm1.Input)
        Debug.Print GetData
        Exit Function
ext:
        GetData = 88
End Function
```

Sample Code Fragment: Retrieve Settings

```
Public Sub GetSettings_Click()
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(0)            'Enter Setup Command
    MSComm1.Output = Chr$(3)            'Get Settings Command
    Debug.Print "Waiting..."
    If GetData = 254 Then               'Begin Retrieval of Settings
        HScroll3.Value = GetData        '1 = E3C Device Number
        CDEL.Value = GetData            '2 = Character Delay
        For n = 0 To 16                 'Get Keypad Character Codes
            Key(n).Caption = Chr$(GetData)
            'MSComm1.Output = Chr$(Asc(Key(n).Caption))
            GetData
        Next n
        Debug.Print GetData
    End If
End Sub

Public Function GetData()
       t = 0
        Do
            t = t + 1
            If t > 10000 Then GoTo ext

            DoEvents
        Loop Until MSComm1.InBufferCount > 0
        GetData = Asc(MSComm1.Input)
        Debug.Print GetData
        Exit Function
ext:
        GetData = 88
End Function
```

The Hybrid Command Set

Sending a Byte of Data to a Data Port

1, 0-255 Put Byte on Port A
This command is used to send a byte of data directly to the Port A data port on the Hybrid. All lines on Port A become an output when this command is issued. This command is very similar to a serial-to-parallel converter. The parameter of this command, 0-255, is written directly to the port and the on/off status of each of the 8 lines appears in the equivalent binary pattern of the parameter.

2, 0-255 Put Byte on Port B
This command is used to send a byte of data directly to the Port B data port on the Hybrid. All lines on Port B become an output when this command is issued. This command is very similar to a serial-to-parallel converter. The parameter of this command, 0-255, is written directly to the port and the on/off status of each of the 8 lines appears in the equivalent binary pattern of the parameter.

3, 0-255 Put Byte on Port C (8-Relay Hybrid ONLY)
This command is used to send a byte of data directly to the Port C data port on the Hybrid. Port C is an Output-Only port. This command is very similar to a serial-to-parallel converter. The parameter of this command, 0-255, is written directly to the port and the on/off status of each of the 8 lines appears in the equivalent binary pattern of the parameter.

Sample Code: Send Byte to Port

```
Public Sub PortAByte(DatByte)           'DatByte Parameter = 0 to 255
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(1)            'Send Data Byte to Port A
    MSComm1.Output = Chr$(DatByte)      'Data Byte to Appear on Port
End Sub
```

```
Public Sub PortBByte(DatByte)           'DatByte Parameter = 0 to 255
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(2)            'Send Data Byte to Port B
    MSComm1.Output = Chr$(DatByte)      'Data Byte to Appear on Port
End Sub
```

8-Relay Hybrid ONLY:

```
Public Sub PortCByte(DatByte)           'DatByte Parameter = 0 to 255
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(3)            'Send Data Byte to Port C
    MSComm1.Output = Chr$(DatByte)      'Data Byte to Appear on Port
End Sub
```

Setting Individual Data Bits

4, 0-47 Set Port Bits
This command is used to set the on/off status of individual I/O lines on each of the 3 data ports. This command requires a parameter of 0-47.
Parameter Values 0-7 Turn Off Port A Bits
Parameter Values 8-15 Turn On Port A Bits
Parameter Values 16-23 Turn Off Port B Bits
Parameter Values 24-31 Turn On Port B Bits
Parameter Values 32-39 Turn Off Port C Bits ***8-Relay Hybrid ONLY***
Parameter Values 40-47 Turn On Port C Bits ***8-Relay Hybrid ONLY***

Sample Code: Setting Status of Port Bits

```
Public Sub PortBitSet(BitSet)           'BitSet Parameter = 0 to 47
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(4)            'Send BitSet Command
    MSComm1.Output = Chr$(BitSet)       'Set Bit Status
End Sub
```

Reading Port Bits

5, 0-23 Get Status of Port Bits
This command is used to read the status of data bits on each of the 3 data ports. This command requires a parameter value of 0-23. This command returns an ASCII character code 0 or 1 indicating bit status.
Parameter Values 0-7 Set Bit to Input and Read Port A Data Bit
Parameter Values 8-15 Set Bit to Input and Read Port B Data Bit
8-Relay Hybrid ONLY:
Parameter Values 16-23 Read Port C Bits, Cannot Be Used as Input

Sample Code: Read Port Bit Status

```
Public Function PortBitGet(GetBit)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(5)            'Request Status of Port Bit
    MSComm1.Output = Chr$(GetBit)       'Port Bit to Read
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetBit = Asc(MSComm1.Input)         'Get Status from Controller
End Sub
```

The Hybrid Command Set

Reading Port Bytes

The Hybrid is capable of reading data from each of the various data ports. The Read Port Byte command is rather extensive because it is used to report any data generated by the Hybrid back to the user. These commands include reading I/O status information, analog values, and data generated by various expansion modules. For this reason, each parameter for the Read Port Byte command will be thoroughly explained.

Read Port Byte from Data Port

6, 0 Read Port A Data Byte

This command is used to read a binary value from Port A, acting like a parallel-to-serial encoder. This command switching all data bits on Port A to an input, takes a reading, and reports a value of 0-255 back to the user.

6, 1 Read Port B Data Byte

This command is used to read a binary value from Port B, acting like a parallel-to-serial encoder. This command switching all data bits on Port B to an input, takes a reading, and reports a value of 0-255 back to the user.

6, 2 Read Port C Data Byte (8-Relay Hybrid ONLY)

This command is used to report the current output status of Port C. Port C cannot be used to read inputs, therefore, it can only tell you what the current output status is currently set to. This command reports a value of 0-255 back to the user.

Sample Code: Read Port Byte

```
Public Function GetPortAStatus
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(6)            'Request a Port Byte
    MSComm1.Output = Chr$(0)            'Request Port A Byte
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetPortAStatus = Asc(MSComm1.Input) 'Get Status from Controller
End Sub

Public Function GetPortBStatus
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(6)            'Request a Port Byte
    MSComm1.Output = Chr$(1)            'Request Port B Byte
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetPortAStatus = Asc(MSComm1.Input) 'Get Status from Controller
End Sub
```

8-Relay Hybrid ONLY:

```
Public Function GetPortCStatus
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(6)            'Request a Port Byte
    MSComm1.Output = Chr$(2)            'Request Port C Byte
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetPortAStatus = Asc(MSComm1.Input) 'Get Status from Controller
End Sub
```

Reading 8-Bit Analog Values

6, 3-7 Read 8-Bit Analog Data on Channels 1-5

The first 5 lines (labeled 0-4 on Hybrid circuit board) on Port A can be used to read 8-bit analog values from a variable 0-5VDC data source. This command requests the analog data value from the user-specified A/D input, and reports back an analog value of 0-255.

Sample Code: Reading 8-Bit Analog Values

```
Public Function GetAnalog8(Channel)     'Channel = 0 to 4
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(6)            'Request 8-Bit Analog Value
    MSComm1.Output = Chr$(3+Channel)    'Request from Channels 0-4
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetAnalog8 = Asc(MSComm1.Input)     'Get Analog Value from IOAD168
End Sub
```

Reading 10-Bit Analog Values

6, 8-12 Read 10-Bit Analog Data on Channels 1-5

The first 5 lines (labeled 0-4 on Hybrid circuit board) on Port A can be used to read 10-bit analog values from a variable 0-5VDC data source. This command requests the analog data value from the user-specified A/D input, and reports back two bytes that are computed to indicate an analog value from 0-2047.

Sample Code: Reading 10-Bit Analog Values

```
Public Function GetAnalog8(Channel)     'Channel = 0 to 4
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(6)            'Request 10-Bit Analog Value
    MSComm1.Output = Chr$(8+Channel)    'Request from Channels 0-4
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    MSB = Asc(MSComm1.Input)            'Get Most Significant Byte
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    LSB = Asc(MSComm1.Input)            'Get Least Significant Byte
    GetAnalog10 = (MSB*256)+LSB         'Compute Analog Value
End Sub
```

The Hybrid Command Set

Expansion Modules

The Hybrid is capable of interfacing to a number of 3rd party products and expansion modules. This section demonstrates the use of the Hybrid with various other products such as Keypads, LCD Displays, and of course, NCD Expansion Modules.

PAR24: 24-Bit Parallel Output

The PAR24 is a 24-Bit Parallel Output Expansion module that can be connected to Port A or Port B of the Hybrid controller. The PAR24 provides three sets of 8 TTL/CMOS compatible outputs. Each set of eight outputs is called a “Channel”. If you are using ONE PAR24 connected to PORTA, then channels 1-3 are available. If you are using TWO PAR24s connected to PORTA, channels 1-6 are available. Up to FOUR PAR24 expansion modules can be connected to a single Hybrid controller. Two on Port A and two on Port B. This will provide 96 TTL/CMOS outputs (+ 8 outputs on Port C 8-Relay Version Only) of the Hybrid. The following commands are used to send bytes of data to the outputs of PAR24. A single byte of data is easily routed to the user selected Port and Channel using the integrated PAR24 command set.

PAR24: PORT A and PORT B

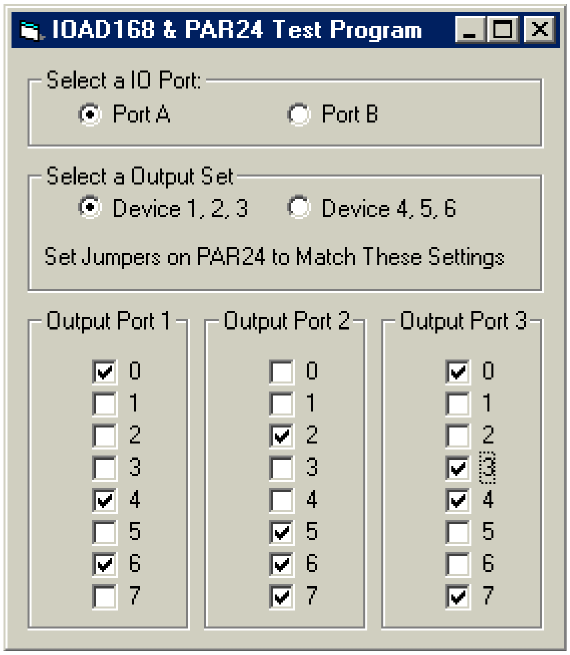
11, 1-6, 0-255 PAR24 Output Byte PortA
This command sends a byte of data (0-255) to one of the PAR24 output channels (1-6) on Port A (Command 11). The data appears as binary/parallel data on the 8-bit output of the PAR24.

12, 1-6, 0-255 PAR24 Output Byte PortB
This command sends a byte of data (0-255) to one of the PAR24 output channels (1-6) on Port B (Command 12). The data appears as binary/parallel data on the 8-bit output of the PAR24.

Sample Code: PAR24

```
Public Sub Par24A(Channel,Data)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(11)           'PAR24 on Port A Command
    MSComm1.Output = Chr$(Channel)      'Select Output Channel 1-6
    MSComm1.Output = Chr$(Data)         'Send Data 0-255
End Sub

Public Sub Par24B(Channel,Data)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(12)           'PAR24 on Port B Command
    MSComm1.Output = Chr$(Channel)      'Select Output Channel 1-6
    MSComm1.Output = Chr$(Data)         'Send Data 0-255
End Sub
```



IOAD168 & PAR24 Test Program

Select a IO Port:
● Port A ○ Port B

Select a Output Set
● Device 1, 2, 3 ○ Device 4, 5, 6

Set Jumpers on PAR24 to Match These Settings

Output Port 1
☑ 0
□ 1
□ 2
□ 3
☑ 4
□ 5
☑ 6
□ 7

Output Port 2
□ 0
□ 1
☑ 2
□ 3
□ 4
☑ 5
☑ 6
☑ 7

Output Port 3
☑ 0
□ 1
□ 2
☑ 3
☑ 4
□ 5
□ 6
☑ 7

We also provide a PAR24 demo program in the form of Visual Basic 6 Source Code at www.controlanything.com.

The Hybrid Command Set

AD1216: 16-Channel 12-Bit A/D Converter

The AD1216 is an expansion module with a 16-Channel 12-Bit A/D Converter that is compatible with Port A or Port B on the Hybrid controller. Up to Six AD1216s can be connected to a single Hybrid, three on Port A and three on Port B. This allows monitoring of 96 analog data sources at 12-bit resolution. The AD1216 command set makes it very easy to request the analog value on any port, device, and channel. Two bytes of data are sent back to the user after a successful A/D conversion. The Least Significant Byte (LSB) is sent first, followed by the Most Significant Byte (MSB). To convert the MSB and LSB to an analog value of 0-4095, use the following equation:

ANALOG_VALUE = (MSB x 256) + LSB

HINT: You can increase the samples per second far above what is shown in the picture below by setting the baud rate to 38.4K baud and by decreasing the delay between characters. A fast computer is required. See Page 8 for Details.

6, 13, 0-2, 0-15 Get 12-Bit Analog on Port A

This command returns a 12-Bit analog value from Channel 0-15, Defined as Device 0-2 on a AD1216 Connected to Port A.

6, 14, 0-2, 0-15 Get 12-Bit Analog on Port B

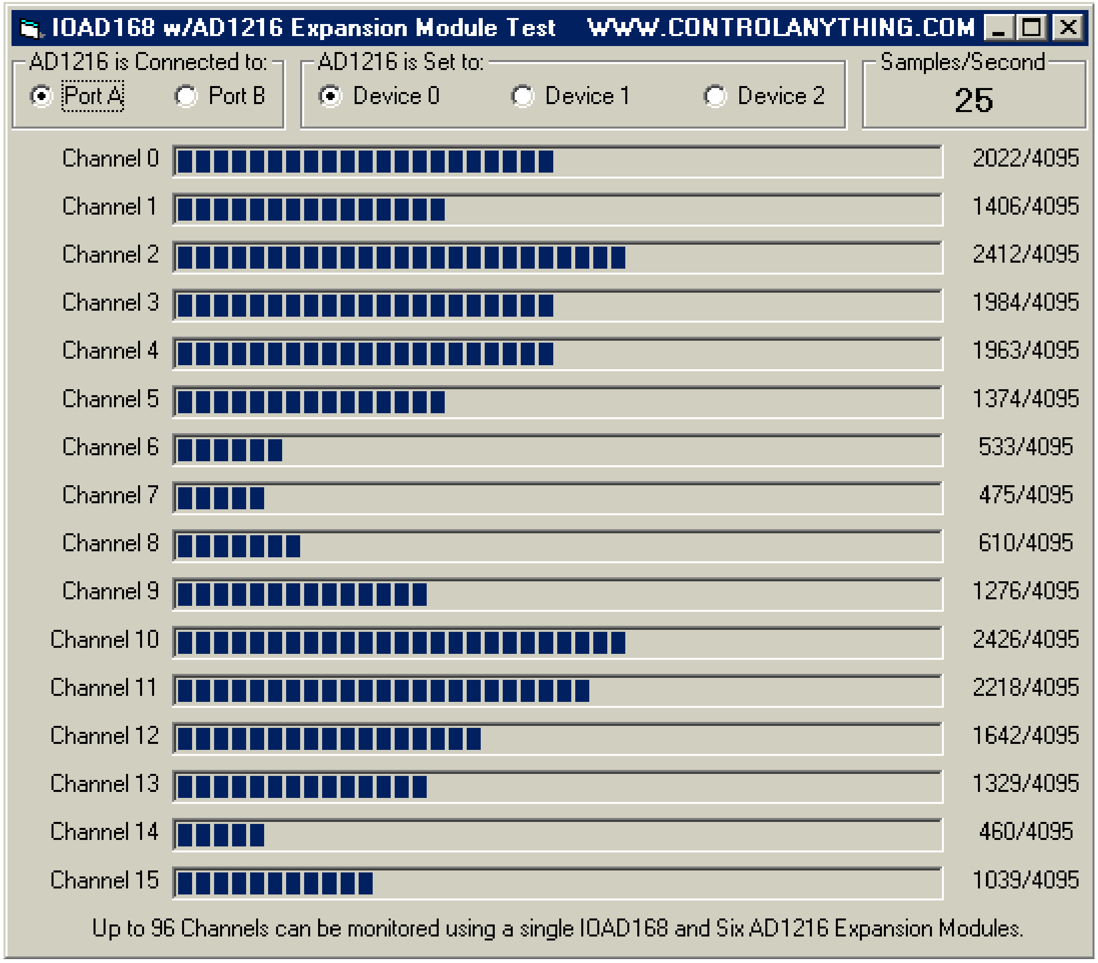
This command returns a 12-Bit analog value from Channel 0-15, Defined as Device 0-2 on a AD1216 Connected to Port B.

Sample Code: AD1216

```
Public Function AD1216A(Device,Channel)
   MSComm1.Output = Chr$(254)     'Enter Command Mode
   MSComm1.Output = Chr$(6)       'Get Port Byte
   MSComm1.Output = Chr$(13)      'from AD1216 on Port A
   MSComm1.Output = Chr$(Device)  'AD1216 Device Number 0-2
   MSComm1.Output = Chr$(Channel) 'Analog Input Channel 0-15
   LSB = SerialIn
   MSB = SerialIn
   AD1216A = (MSB * 256) + LSB
End Function


Public Function AD1216B(Device,Channel)
   MSComm1.Output = Chr$(254)     'Enter Command Mode
   MSComm1.Output = Chr$(6)       'Get Port Byte
   MSComm1.Output = Chr$(14)      'from AD1216 on Port B
   MSComm1.Output = Chr$(Device)  'AD1216 Device Number 0-2
   MSComm1.Output = Chr$(Channel) 'Analog Input Channel 0-15
   LSB = SerialIn
   MSB = SerialIn
   AD1216B = (MSB * 256) + LSB
End Function

Public Function SerialIn()
        Timeout = 100
        Do
            Timeout = Timeout - 1
            If Timeout <= 0 Then
                SerialIn = -1
                Exit Function
            End If
            DoEvents
        Loop Until MSComm1.InBufferCount > 0
        SerialIn = Asc(MSComm1.Input)
End Function
```



IOAD168 w/AD1216 Expansion Module Test WWW.CONTROLANYTHING.COM

AD1216 is Connected to: ● Port A ○ Port B

AD1216 is Set to: ● Device 0 ○ Device 1 ○ Device 2

Samples/Second 25

Channel 0 2022/4095
Channel 1 1406/4095
Channel 2 2412/4095
Channel 3 1984/4095
Channel 4 1963/4095
Channel 5 1374/4095
Channel 6 533/4095
Channel 7 475/4095
Channel 8 610/4095
Channel 9 1276/4095
Channel 10 2426/4095
Channel 11 2218/4095
Channel 12 1642/4095
Channel 13 1329/4095
Channel 14 460/4095
Channel 15 1039/4095

Up to 96 Channels can be monitored using a single IOAD168 and Six AD1216 Expansion Modules.

We also provide a AD1216 demo program in the form of Visual Basic 6 Source Code at www.controlanything.com.

The Hybrid Command Set

Introduction to the LCD and VF Character Display Command Set

The Hybrid also includes several powerful commands for controlling up to three character LCD and VF displays. A character display may be attached to Port A, B, and/or C of the Hybrid. Each port has slightly different timing characteristics. Port C seems to be the most compatible (8-Relay version of Hybrid ONLY), followed closely by port A. Port B character display routines operate slightly faster, and may suffer from incompatibility with some displays. Port B should NOT be used if you intend to use the Hybrid as a terminal since Port B is the only port capable of reading a 16-button keypad.

Character displays are controlled in 4-bit mode using 7 of the 8 TTL outputs. The 8th TTL output can be used to software control the on/off status of a backlight.

It is important to understand that the CPU on the Hybrid runs MUCH faster than a character display can keep up with. Therefore, some provisions have been taken to ensure the display is never overrun with text or commands. Most commands and data send ASCII character code 85 back to the user to indicate completion. This serves to slow communications just enough to ensure compatibility with just about every character display ever made up to 80 characters (such as 40x2 and 20x4 displays).

Commands are sent directly to the display at the hardware level, therefore, the user has complete control over the use of write-only commands to the character display. This guide will discuss the most popular character display commands. Other commands may be issued using the functions we provide. A detailed data sheet on character displays should be obtained to use the advanced command set of character displays.

There are Seven character display commands currently supported by the Hybrid. Here is a brief overview of the command set. A detailed explanation will be provided on the following pages.

LCD Initialize
This command issues a general purpose initialization for all character displays. This command was written specifically to initialize a 16x2 display, but the init sequence is compatible with just about every character display produced. This command returns 85 back to the user when complete.

LCD Data
This command is used to send data to the display. Data is used for text and command parameters. This command returns 85 back to the user when complete.

LCD Command
This command is used to send a hardware command to the character display. Commands are used to position the cursor and other such functions. This command returns 85 back to the user when complete.

LCD Text Buffer
Due to speed limitations of a character display, text cannot be written directly to the screen from a serial output. Instead, text must be stored in a text buffer and then dumped to the display. This is managed entirely by the Hybrid firmware. Simply fill the text buffer and terminate the command with ASCII character code 255 (more on this later). These data will then be rationed to the display from the text buffer. When the operation is complete, ASCII character 85 will be sent back to the user.

LCD LED Off
Turn Off Backlight LED. Initialization may be required after this command has be issued for some displays.

LCD LED On
Turn On Backlight LED. Initialization may be required after this command has be issued for some displays.

LCD Port
The Hybrid has 3 parallel data ports (two on 16-Relay Versions of the Hybrid), all of which are compatible with a character display. The LCD Port command is used to direct LCD commands to one of these data ports. By default, Port C is used (8-Relay Hybrid, Port A on 16-Relay Hybrid). Port A may be selected by issuing this command with a parameter of 0. Port B may be selected by issuing this command with a parameter of 1. All subsequent LCD commands will be directed to the selected port. This allows an Hybrid to control up to 3 character displays of different sizes at one time.

Character Display Related Commands

The Hybrid also supports commands for controlling a keypad on Port B. This allows the Hybrid to be used as a terminal interface to users in 256 different locations using a single serial port.

In some applications, it is necessary to direct key presses from the integrated keypad directly to the character display screen. The Hybrid also supports a command that we call “Key-to-Screen”. Key-to-screen allows any key press on the keypad to be viewed directly on the character display.

In the case of PIN number and Pass Code requests, the Hybrid can be configured to display an “X” (or any other character) to hide the key presses from other potential viewers.

Complete details on the Key-to-Screen features can be found in the Keypad section of this manual. The next few pages will help you utilize the character display capabilities of the Hybrid.

The Hybrid Command Set

Connecting a Display to the IOAD168

The IOAD168 is capable of connecting to three different character display modules at one time. The user has complete software control of all display write functions as well as a software controlled backlight. The controller firmware has been tuned for compatibility with most character LCD and Vacuum Florescent displays.

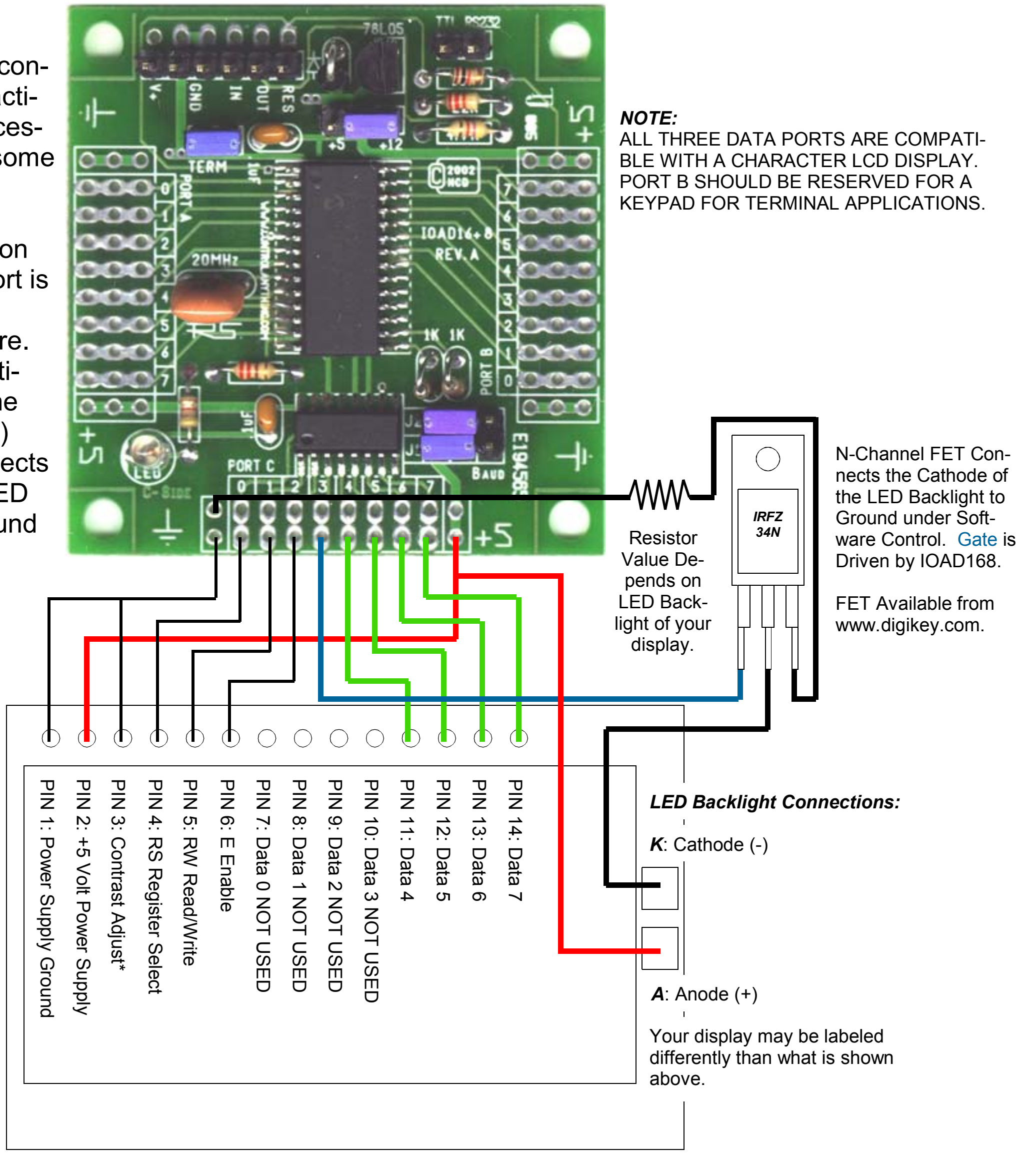
Power for the display is derived from the IOAD168 controller. When controlling a Vacuum Florescent Display, you MUST provide a regulated +5 volt supply to the IOAD168 AND the power supply jumper below MUST be set to +5 volts to avoid damage to the on-board voltage regulator.

For Hybrid Controllers:

This page illustrates the connection of a LCD to a IOAD168. NCD Hybrid 8-Relay Controllers have 3 Data Ports that are identical to the IOAD168 except for the location. NCD Hybrid 16-Relay Controllers do NOT have a Port C. For these controllers, Use Port A (default) for controlling a character LCD Display.

When the software controlled backlight is activated, it may be necessary to re-initialize some display models.

Data pin 3 (labeled on the board) of any port is used to control the backlight via software. When this line is activated, the gate of the FET (shown at right) goes high and connects the ground of the LED backlight to the ground of the controller.



NOTE:
ALL THREE DATA PORTS ARE COMPATIBLE WITH A CHARACTER LCD DISPLAY. PORT B SHOULD BE RESERVED FOR A KEYPAD FOR TERMINAL APPLICATIONS.

Resistor Value Depends on LED Backlight of your display.

N-Channel FET Connects the Cathode of the LED Backlight to Ground under Software Control. Gate is Driven by IOAD168.

FET Available from www.digikey.com.

LED Backlight Connections:

K: Cathode (-)

A: Anode (+)

Your display may be labeled differently than what is shown above.

The Hybrid Command Set

Hybrid Character Display Native Commands

When controlling a character LCD display with the Hybrid, it is important to realize there are two distinct command sets. There are commands that are we have built into the Hybrid to control a character display (Hybrid Native Commands), and there are commands built into the character display itself (LCD Native Commands).

The first type of commands we will discuss are the commands that we have designed to control a character display (Hybrid Native Commands). There are seven commands used to control all character display write-only functions.

By default, all commands are routed to Port C on Hybrid 8-Relay Controllers, Port A on Hybrid 16-Relay Controllers. It is possible to route LCD commands to any of the data ports on the Hybrid using the LCD Port command on the next page. Port B should not be used for character display functions if you intend to use a Keypad.

LCD Initialize

7, 0 LCD Initialize
This command is used to send an initialization sequence to the character display. While this command was written specifically to initialize a 16x2 display, this command is compatible with all displays we tested. When this command has finished, a flashing cursor should appear in the upper left corner of the display screen. This command sends ASCII character code 85 back to the user indicating the command has finished execution.

Sample Code: LCD Initialize

```
Public Sub LCD_Init
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(0)            'Initialization Command
    GetData                             'Wait for Command to Finish
End Sub
```

LCD Data

7, 1, 0-255 LCD Data
This command is used to send data directly to the display hardware. This command can be used to display text on the screen, however, it is much more efficient to use the LCD Text Buffer command. This command is not used in most applications, but it is provided so just incase you need to adjust some display parameters. This command requires a parameter from 0-255 indicating the data that it is to be sent to the display. This command sends ASCII character code 85 back to the user indicating the command has finished execution.

Sample Code: LCD Data

```
Public Sub LCD_Data(Data)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(1)            'LCD Data Command
    MSComm1.Output = Chr$(Data)         'Send Data 0-255
    GetData                             'Wait for Command to Finish
End Sub
```

LCD Command

7, 2, 0-255 LCD Command
This command is used to send a command directly to the display hardware. This command should be used to issue all commands native to the character display. Such commands allow you to be hide and position the cursor, program your own fonts, and clear the display. This command will be demonstrated on the following pages. This command requires a parameter from 0-255 indicating the command that it is to be sent to the display. Misuse of this command can cause unusual and erratic behavior. A complete set of commands is found in your LCD data sheet. Only the most common commands will be discussed in this manual. This command sends ASCII character code 85 back to the user indicating the command has finished execution.

Sample Code: LCD Command

```
Public Sub LCD_Command(Command)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(2)            'LCD Hardware Command
    MSComm1.Output = Chr$(Command)      'Send Command 0-255
    GetData                             'Wait for Command to Finish
End Sub
```

Sample Code: Wait for Command to Finish

```
Public Function GetData
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetData = Asc(MSComm1.Input)        'Get Status from Controller
    Debug.Print GetData                 'Display in Immediate Window
End Sub
```

The Hybrid Command Set

LCD Buffer

7, 3, Text$, 255 LCD Buffer (64 Byte Text Buffer)
The Hybrid has a very powerful text buffer built in that is very easy to use. A text buffer is required because the processor on the Hybrid easily outruns the capabilities of a character display. Due to these speed limitations, text is stored in a buffer and slowly rationed to the display. To use the text buffer command, simply send: 254, 7, 3, “Text that you want to send.”, 255. The 255 is a “Termination” command. It is used to signal the end of your text to the controller. Up to 64 characters can be stored in the text buffer. If the text buffer is filled, the display will update with the current contents of the text buffer and ignore excess data. Once text data has been written to the display, this command will send ASCII character code 85 back to the user indicating completion of this command.

Sample Code: LCD Buffer

```
Public Sub LCD_Buffer
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(3)            'Text Buffer Command
    MSComm1.Output = “Hello World”      'Text to Display
    MSComm1.Output = Chr$(255)          'Terminate and Display
    GetData                             'Wait for Command to Finish
End Sub
```

Backlight OFF

7, 4 LED Off
Turns off the backlight LED. Some displays need to be re-initialized after this command has finished execution. By default, the backlight LED is Off.

Sample Code: Backlight Off

```
Public Sub LCD_Backlight_Off
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(4)            'LED Backlight Off Command
End Sub
```

Backlight ON

7, 5 LED On
Turns on the backlight LED. Some displays need to be re-initialized after this command has finished execution.

Sample Code: Backlight On

```
Public Sub LCD_Backlight_On
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(5)            'LED Backlight On Command
End Sub
```

Routing Commands to Hybrid Data Ports

7, 6, 0-2 LCD Port
It is possible to attach up to 3 character displays to a single Hybrid 8-Relay Controller (Two on Hybrid 16-Relay Controller). One display is allowed per data port (A, B, or C). This command is used to route all subsequent LCD commands to the selected port. By default, LCD commands are sent to Port C on Hybrid 8-Relay Controllers. On Hybrid 16-Relay Controllers, all LCD commands are directed to Port A.

You can route LCD commands to Port A using 254, 7, 6, 0.
All subsequent LCD commands will be sent to Port A.
You can route LCD commands to Port B using 254, 7, 6, 1.
All subsequent LCD commands will be sent to Port B.
Hybrid 8-Relay Controller Only:
You can route LCD commands to Port C using 254, 7, 6, 2.
All subsequent LCD commands will be sent to Port C.

Sample Code: Command Port Selection

```
Public Sub LCD_Port(Port)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(6)            'LCD Hardware Port
    MSComm1.Output = Chr$(Port)         'Select Port A-C (0-2)
End Sub
```

Sample Code: Wait for Command to Finish

```
Public Function GetData
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetData = Asc(MSComm1.Input)        'Get Status from Controller
    Debug.Print GetData                 'Display in Immediate Window
End Sub
```

The Hybrid Command Set

Native LCD Commands

A character display has its own built-in commands used to control cursor position, cursor appearance, fonts, and much more. Here are a few commands that can be sent to control these display functions. Other commands can be sent, please consult the manual for your character display for a complete set of commands and usage. The Hybrid does not support any commands that read data from the display.

Basic HD44780 Command Set:

Command	Description
1	Clear Screen
2	Home (move cursor to top/left character position)
8	Blank the display (without clearing)
12	Make cursor invisible
12	Restore the display (with cursor hidden)
14	Turn on visible underline cursor
15	Turn on visible blinking-block cursor
16	Move cursor one character left
20	Move cursor one character right
24	Scroll display one character left (all lines)
28	Scroll display one character right (all lines)
64 + addr	Set pointer in character-generator RAM (CG RAM address)
128 + addr	Set cursor position (DDRAM address)

Sample Code: Native LCD Commands

```
Public Sub Clear_Screen()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(1)    'Clear Screen Command
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Home_Cursor()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(2)    'Home Cursor Command
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Blank_Display()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(8)    'Blank Display Command (Does Not Clear)
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Invisible_Cursor()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(12)   'Invisible Cursor Command
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Restore_Display()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(12)   'Restore Display Command (Cursor Invisible)
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Underline_Cursor()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(14)   'Underline Cursor
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Block_Cursor()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(15)   'Block Cursor Command
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Cursor_Left()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(16)   'Move Cursor Left
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Cursor_Right()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(20)   'Move Cursor Right
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Set_Cursor_Position(Pos)
    MSComm1.Output = Chr$(254)      'Enter Command Mode
    MSComm1.Output = Chr$(7)        'LCD Function
    MSComm1.Output = Chr$(2)        'Send Command to LCD
    MSComm1.Output = Chr$(128+Pos)  'Set Cursor Position (0-127)
    GetStatus                       'Wait for Command to Finish
End Sub
```

The Hybrid Command Set

Keypad Encoder

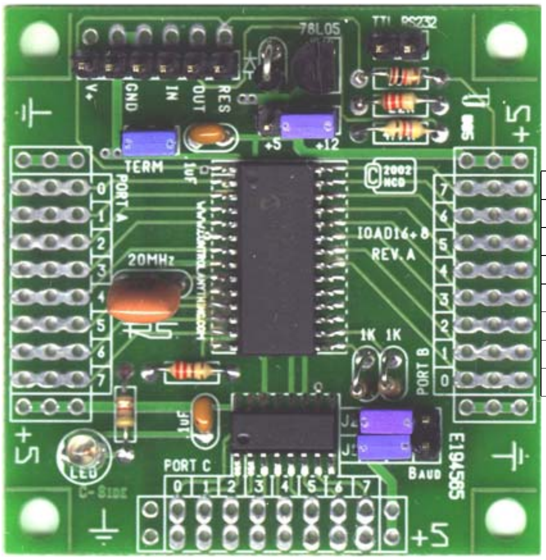
The Hybrid incorporates a very powerful programmable keypad encoder function. Any ASCII value from 0 to 255 can be assigned to each of 16 keys. See page 8 for assigning new values for each of the keys on the keypad.

Connecting a Keypad to the Hybrid is very easy. All that is required is a suitable keypad (Matrix Type up to 16 Keys), and a Pull-Down Resistor Network (1K to 4.7K is suitable, 3.9K shown below).

Photo Illustrates Resistor Network Connection on IOAD168. Using a Resistor Network on a Hybrid controller is identical other than physical location of the Data Port B on the Hybrid.

The Hybrid incorporates scan-based encoding algorithm to determine which keys are pressed. For this reason, 4 data bits are TTL level inputs and 4 are TTL level outputs. The encoder function detects an intersection between input and outputs and assigns a numeric value from 0-15. This numeric value is then cross referenced with the user-defined character set and sent out the serial port of the Hybrid.

A keypad can only be attached to Port B of the Hybrid. The photo below shows how to connect a keypad to the IOAD168. ***Connection to a Hybrid is identical except for the physical location of data port B.***



Row A	Output
Row B	Output
Col 1	Input
Col 2	Input
Col 3	Input
Col 4	Input
Row D	Output
Row C	Output

Part Numbers:
The Hybrid is compatible with matrix type keypads from many manufacturers. We used the Storm 700 and 900 series keypads from www.digikey.com for development. Part number MGR1113-ND. Part number 7731011102-ND is also a suitable 1K 10-Pin 9-Resistor network.

Keypad Functions

8, 0-127 Keypad
The Keypad function is used to acquire key presses from the user. This command requires a parameter from 0-127 indicating the number of key presses you would like the user to enter.

9, 0-127 Keypad to Screen
This command is the same as above, except that key presses are directed to a character display connected to Port A or Port C of the Hybrid. The keys that are stored in the controller (see page 8) will appear on the character display screen.

9,128-255 Keypad Hidden to Screen
This command is the same as above, except the key presses are not displayed on the screen. Rather, the hidden PIN character is used to acknowledge key presses from the user. The parameter value of (128-255) - 127 indicates the number of key presses. To acquire one key press from the user, send the command: 254, 9, 128. By default, the requested character will appear as an “X” on the character display screen. To acquire 4 key presses, send the command: 254, 9, 131. Four X’s will appear on the screen as the key is pressed. The user will be sent the actual character value assigned to the keys (assignment is shown on page 8 of this manual).

Sample Code: Keypad Functions

```
Public Sub Keypad(Keys)
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(8)    'Keypad Function
    MSComm1.Output = Chr$(Keys) 'Get Key Presses (0-127)
    For N = 1 to Keys           'Display Key presses from User
       GetStatus
    Next N
End Sub

Public Sub KeyToScreen(Keys)
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(9)    'Keypad Function Keys Displayed on LCD
    MSComm1.Output = Chr$(Keys) 'Get Key Presses (0-127)
    For N = 1 to Keys           'Display Key presses from User
       GetStatus
    Next N
End Sub

Public Sub KeyToScreenHidden(Keys)
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(9)    'Keypad Function Keys Displayed on LCD
    MSComm1.Output = Chr$(Keys) 'Get Key Presses (128-255)
    For N = 1 to Keys—127       'Display Key presses from User
       GetStatus
    Next N
End Sub
```

The Hybrid Relay Controller Command Set

Introduction

The Hybrid integrates most of the features of the R8x Pro Series Command Set. This command set is designed to support 8 relays. Since there are 16 relays on the IOADR16x, commands must be "applied" or "directed to" a bank of relays. The IOADR16x is composed of two "Banks" of relays, Bank A and Bank B. It is possible to apply relay commands to either bank, but it is not possible to apply R8x Pro relay control commands to both banks using a single command.

Selecting Relay Banks

The R8x Pro Command Set was designed to manipulate up to 8 relays with a single command. Since there are two banks of relays on the IOADR16x, commands must be directed to Bank A or Bank B relays. Commands cannot be directed to BOTH banks of relays. The Bank Select function is used to tell the IOADR16x which Bank of Relays (A or B), will receive the relay control commands. All subsequent relay control commands will be directed to this relay bank. By default, when power is first applied to the IOADR16x, commands are directed to Bank A relays. It is not possible to manipulate Bank B relays without using this Bank Select command. Likewise, it is not possible to manipulate Bank A relays without using this command to switch back to Bank A relays. ***NOTE: This command is NOT Available on Hybrid 8-Relay Models.***

About the Relay Controller Command Set

The following pages were copied directly out of the R8x Pro manual and were modified to the specifications of this Hybrid Relay Controller. Some commands have been removed from this command set because they are handled directly by the I/O controller portion of the firmware.

The only real difference between the command structures of the R8x Pro and the Hybrid is the addition of the "Relay Controller Commands Branch" before every R8x command.

Here is an example of a command that is compatible with the R8x Pro Relay Controller:

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

The same command is available on a Hybrid Controller, but the "Relay Controllers Command Branch" must be used to access the "R8x" Relay Controller Command Set:

```
MSComm1.Output = Chr$(254)  'Enter Command Mode
MSComm1.Output = Chr$(14)   'R8x Pro Relay Controller Command Branch
MSComm1.Output = Chr$(30)   'Turn On All Relays
```

IMPORTANT:

Relay Control Commands MUST be Directed to Bank A or Bank B Relays. It is NOT POSSIBLE to use the R8x Pro Command Set on Both Banks of Relays using a Single Command. Command 13 is NOT Supported by Hybrid 8-Relay Controllers.

Directing Commands to Relay Bank A

```
Public Sub RelayBankA
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(13)   'Direct Relay Commands to Relay Bank
    MSComm1.Output = Chr$(0)    'Send Relay Commands to Bank A Relays
End Sub
```

NOTE: ALL SUBSEQUENT RELAY CONTROL COMMANDS WILL BE SEEN BY BANK A RELAYS ONLY.

Directing Commands to Relay Bank B

```
Public Sub RelayBankB
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(13)   'Direct Relay Commands to Relay Bank
    MSComm1.Output = Chr$(1)    'Send Relay Commands to Bank B Relays
End Sub
```

NOTE: ALL SUBSEQUENT RELAY CONTROL COMMANDS WILL BE SEEN BY BANK B RELAYS ONLY.

The Hybrid Relay Controller Command Set

The Hybrid 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 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. ASCII character code 14 must then be sent to access the R8x Pro Command Set. See examples at right.

Controlling Individual Relays

0-7 Turing Off Individual Relays

8-15 Turing On Individual Relays

Reading the Status of Relays

16-23 Get the Status of an Individual Relay

This command allows you to read the on/off status of an individual relay. 16 corresponds 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.

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.

Power-Up Default Relay Pattern

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.

26 Get the Power-Up Default Relay Pattern

This command allows you read the stored power-up default relay pattern. The binary pattern of the value returned directly corresponds to the on/off status of each relay.

Reporting Mode

27 Turn Reporting Mode ON

This Command Has Been Omitted from the Hybrid Command Set.

28 Turn Reporting Mode OFF

This Command Has Been Omitted from the Hybrid Command Set.

All On/Off

29 Turns All Relays OFF

30 Turns All Relays ON

Visual Basic Programming Examples

Many Visual Basic 6 programming examples are provided in the following pages to assist in the development of software for controlling Hybrid controllers. Additional source code can be found on our web site at www.controleverything.com.

Sample Code: Controlling Individual Relays

```
Public Sub SetRelayStatus(Relay,Stat)   'Relay Parameter = 1 to 8
                                        'Stat Parameter = 1 or 9
    If Stat = 0                         'Turn Off Relay
       MSComm1.Output = Chr$(254)       'Enter Command Mode
       MSComm1.Output = Chr$(14)        'R8x Pro Command Branch
       MSComm1.Output = Chr$(Relay-1)   'Relay to Turn Off
    Else                                'Turn On Relay
       MSComm1.Output = Chr$(254)       'Enter Command Mode
       MSComm1.Output = Chr$(14)        'R8x Pro Command Branch
       MSComm1.Output = Chr$(Relay+7)   'Relay to Turn On
    Endif
End Sub
```

Sample Code: Reading Status of Relays

```
Public Function GetRelayStatus(Relay)   'Relay Parameter = 1 to 8
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(14)           'R8x Pro Command Branch
    MSComm1.Output = Chr$(Relay+15)     'Get Status of One Relay
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetRelayStatus = Asc(MSComm1.Input) 'Get Status from Serial Buffer
    Debug.Print GetRelayStatus          'Display in Immediate Window
End Sub
Public Function GetAllRelayStatus(Relay)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(14)           'R8x Pro Command Branch
    MSComm1.Output = Chr$(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
End Sub
```

Sample Code: Power-Up Relay Pattern

```
Public Sub StoreDefault
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(14)           'R8x Pro Command Branch
    MSComm1.Output = Chr$(25)           'Store Powerup Default Status
End Sub

Public Function GetDefaultStatus
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(14)           'R8x Pro Command Branch
    MSComm1.Output = Chr$(26)           'Get Status of all Relays
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetDefaultStatus=Asc(MSComm1.Input) 'Get Status from Serial Buffer
    Debug.Print GetDefaultStatus        'Display in Immediate Window
End Sub
```

Sample Code: All On/Off

```
Public Sub AllRelaysOff
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(14)           'R8x Pro Command Branch
    MSComm1.Output = Chr$(29)           'Turn Off All Relays
End Sub

Public Sub ReportingOn
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(14)           'R8x Pro Command Branch
    MSComm1.Output = Chr$(30)           'Turn On All Relays
End Sub
```

The R4x/R8x Pro Command Set

Relay Pattern Inversion and Reversal

31 Invert All Relays

All relays that are currently off turn on, all relays that were on turn off.

32 Reverse Relay Order

The Status of Relays 12345678 are reversed to 87654321. This command does not permanently reassign relays, it only copies the status of the relays when executed.

Sample Code: Relay Inversion and Reversal

```
Public Sub InvertAllRelays
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(31)           'Invert All Relays Command
End Sub

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

Testing 2-Way Communication

33 Test 2-Way Communication

This command can be used to test 2-way communication between the host computer and the relay controller. When executed, the relay controller will send ASCII 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 used to detect the presence of a relay controller on the serial port.

Sample Code: Test 2-Way Communication

```
Public Function Test2Way
    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 Relay Board
    Debug.Print Test2Way                'Display in Immediate Window
End Sub
```

Commands with Parameters:
Set Status of All Relays

40,0-255 Set Status of All Relays

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.

Sample Code: Set Status of All Relays

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

Program Emulation Device Number

41 Not Supported

This Command Does Not Apply to the IOADR Series, and is not supported. If this command is issued, the controller will wait for a new command.

Sample Code: Set Emulation Device Number

```
Not Supported
```

Relay Pattern Banks

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.

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 settings defined by command 42 above.

Sample Code: Memory Storage Functions

```
Public Sub StorePatterninBank(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
End Sub

Public Sub RecallPatterninBank(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
End Sub
```

Relay Select and De-Select

44,0-7 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.

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 de-activated at the same time.

Sample Code: Relay Select/De-Select

```
Public Sub RelaySelect(Relay)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(44)           'Select Relay Command
    MSComm1.Output = Chr$(Relay)        'Relay to Select
End Sub

Public Sub RelayDeselect(Relay)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(45)           'Deselect Relay Command
    MSComm1.Output = Chr$(Relay)        'Relay to Deselect
End Sub
```

The R4x/R8x Pro Command Set

Toggle Relay

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

This command reverses the current on/off status of the selected relay.

Relay Timing Functions

The Hybrid Series R8x Pro Command Set has a timer function used to activate one or more relays for a user specified period of time from 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 Feedback
Time=192	.5 Second Timer with Feedback
Time=73	5 Second Timer with No Feedback
Time=201	5 Second Timer with Feedback

Activate a Single Relay on a Timer

47, Time(0-255), Relay(0-7) Set Relay Timer (R8x Pro)

This command is used to activate a relay for a user-defined period of time. All other relays will remain unchanged. If the selected relay is already on, this function will have no effect, so make sure the relay is off before using this command. This command will send ASCII character code 85 back to the host computer if the timing function is enabled by the ***Time*** parameter.

Relay Pattern Select on a Timer

48, Time(0-255), RPOn(0-255), RPOff(0-255) - R8x Pro

This command is used set the status of all relays (RPOn), apply a timer (Time 0-255), and then set all Relays to a new state once the timer has completed (RPOff). This command will send ASCII character code 85 back to the host computer if the timing function is enabled by the ***Time*** parameter.

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.

Sample Code: Toggle Relay

```
Public Sub ToggleRelay(Relay)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(14)           'R8x Pro Command Branch
    MSComm1.Output = Chr$(46)           'Toggle Relay Command
    MSComm1.Output = Chr$(Relay)        'Relay to Toggle
End Sub
```

Sample Code: Relay Timing Functions

```
Public Sub SetRelayTimer(Tymer,Relay)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(14)           'R8x Pro Command Branch
    MSComm1.Output = Chr$(47)           'Set Timer for a Relay Command
    MSComm1.Output = Chr$(Tymer)        'Specify Time Period
    MSComm1.Output = Chr$(Relay)        'Relay to Activate
    'Debug.print GetFeedback            'Optional if Feedback is Used
End Sub

Public Sub SetMultiRelayTimer(Tymer,RPOn,RPOff)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(14)           'R8x Pro Command Branch
    MSComm1.Output = Chr$(48)           'Set Timer for All Relays
    MSComm1.Output = Chr$(Tymer)        'Specify Time Period
    MSComm1.Output = Chr$(RPOn)         'Timer Start Relay Pattern
    MSComm1.Output = Chr$(RPOff)        'Timer Stop Relay Pattern
    'Debug.print GetFeedback            'Optional if Feedback is Used
End Sub
```

If feedback is enabled, the relay controller will send ASCII character code 85 back to the host computer to indicate the completion of the timer. Call the function below after you have issued command 47 or 48 if the feedback function is enabled. This routine will capture the 85 generated by the relay board.

```
Public Function GetFeedback
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    Feedback = Asc(MSComm1.Input)       'Get Status from Relay Board
End Function
```

Troubleshooting

Nothing Happens when Relay Control Commands are Sent
The most common cause is simply 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.

Unreliable or Unpredictable Operation
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 if using an Desktop PC.

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.

Still Having Problems?
Please call us at (417) 646-5644 between 9AM and 4PM central standard time. Or, e-mail us at ryan@controlanything.com. We typically respond to e-mail requests as soon as they are received, even during the evenings and on weekends.

Electrical Ratings and Switching Characteristics

Model	Ratings	Configuration	Notes
IOADR85	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
IOADR810	10A 250VAC / 8A 30VDC	SPDT	
IOADR820	20A 240VAC / 20A 28VDC	SPDT	Common to Normally Closed Terminal is rated at 10A 240VAC / 10A 28VDC Flange Type Connections to Relay
IOADR830	30A 250VAC / 20A 28VDC	SPST	SPST: Common and Normally Open Terminals Only Flange Type Connections to Relay
IOADR165	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
IOADR1610	10A 250VAC / 8A 30VDC	SPDT	
IOADR1620	20A 240VAC / 20A 28VDC	SPDT	Common to Normally Closed Terminal is rated at 10A 240VAC / 10A 28VDC Flange Type Connections to Relay
IOADR1630	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.

Voltage and Current Consumption

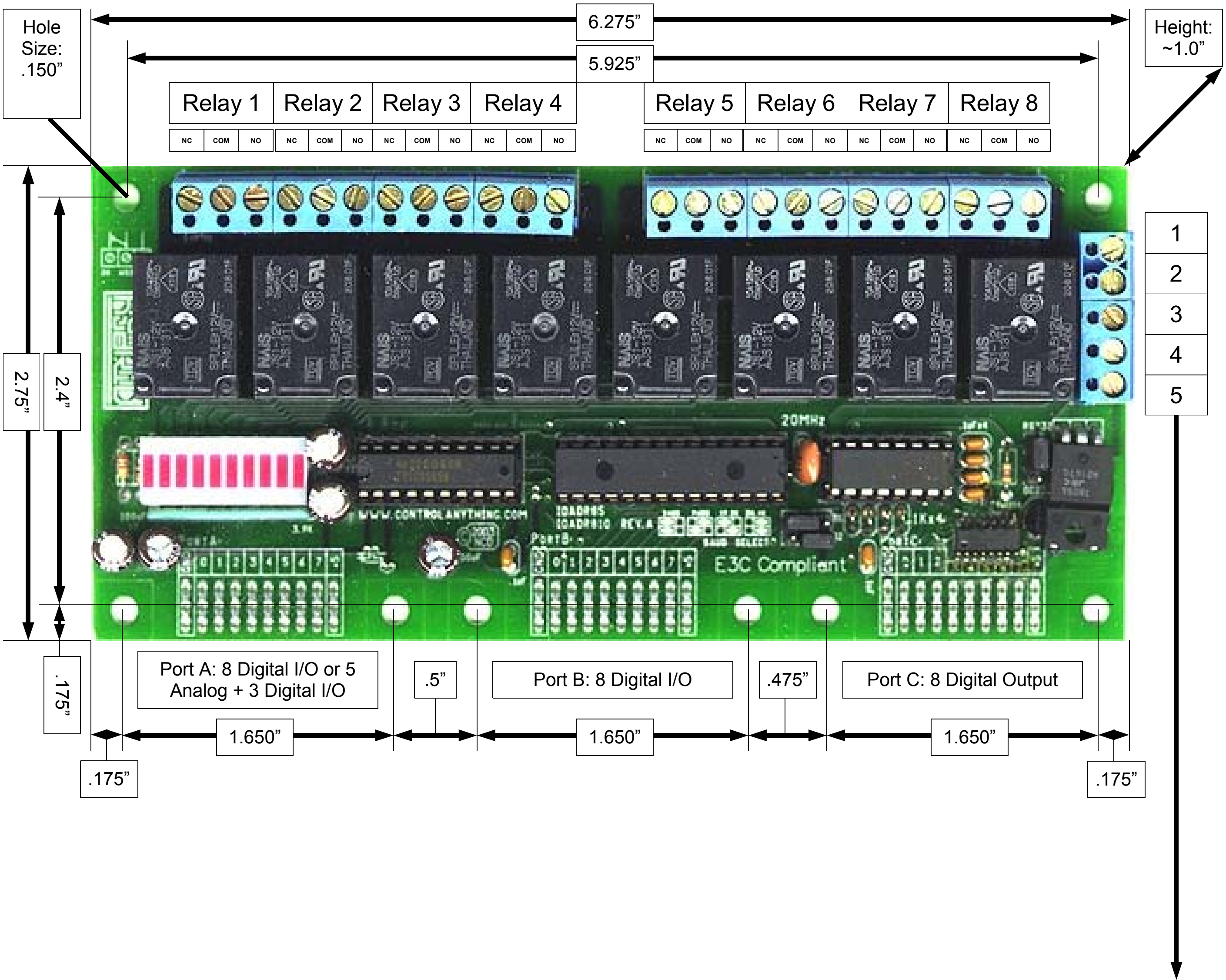
NCD Hybrid Controllers Require a +12 Volt Power Supply capable of delivering 200 ma to power the logic circuits of the relay controller ***PLUS*** 100 ma per activated relay. All NCD relay controllers may be used in standard 13.8V automotive applications without modification. Note that NCD relay controllers operate more efficiently with a computer grade regulated power supply. For this reason, it is safe to use a ***computer grade regulated*** +12 power supply with a current rating of 1.25 amps or more to power any of the Hybrid relay controllers.

Model	Max. Current Consumption Un-Regulated Power Supply
IOADR8x	200 ma MIN to 1,000 ma MAX
IOADR16x	200 ma MIN to 1,800 ma MAX

Characteristics Data	Minimum	Maximum
Relay Activation Time	>5 ms	<15 ms
Relay Deactivation Time	>5 ms	<20 ms
Activations per Second at 2400 Baud using “Pro” Command Set	N/A	80
Activations per Second at 9600 Baud using “Pro” Command Set	N/A	320
Activations per Second at 19.2K Baud using “Pro” Command Set	N/A	640
Activations per Second at 38.4K Baud using “Pro” Command Set	N/A	1280
Communication Distance from PC Without Boosting Signal 2400 Baud*	N/A	Aprox. 2400 Feet
Communication Distance from PC Without Boosting Signal 9600 Baud*	N/A	Aprox. 1200 Feet
Communication Distance from PC Without Boosting Signal 19.2K Baud*	N/A	Aprox. 400 Feet
Communication Distance from PC Without Boosting Signal 38.4K Baud*	N/A	Aprox. 200 Feet
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
Typical Operational Cycles Per Minute	N/A	1,800

* assumes good quality low-capacitive wire, twisted pair preferred.

Outline Dimensions IOADR85, IOADR810



NOTE: Port A can be used as 8-Bit Digital I/O OR 5-Channel A/D (8-Bit or 10-Bit) + 3-Bit Digital I/O. Port A Compatible with Character Display. Port B Compatible with Character Display OR Matrix Type Keypad up to 16 Keys. Port C Compatible with Character Display.

Relay Contact	Description
NC	Normally Closed: This Line Connects to Common when Relay is Turned Off
COM	Common: This is the Moving Portion of the Switch that Swings between NC and NO
NO	Normally Open: This Line Connects to Common ONLY When the Relay is Turned On

Power and Data Interface

1) +5 or 12 Volt Input
This pin supplies power to the board. ***MAKE SURE THE POWER JUMPER IS SET PROPERLY OR PERMANENT DAMAGE WILL RESULT***.

2) Power Ground
3) RS-232 Ground
These ground lines are tied together on the IOAD168 board.

4) RS-232 Data Input
Connect this line to the RS-232 Data OUTPUT of your Computer or Micro.

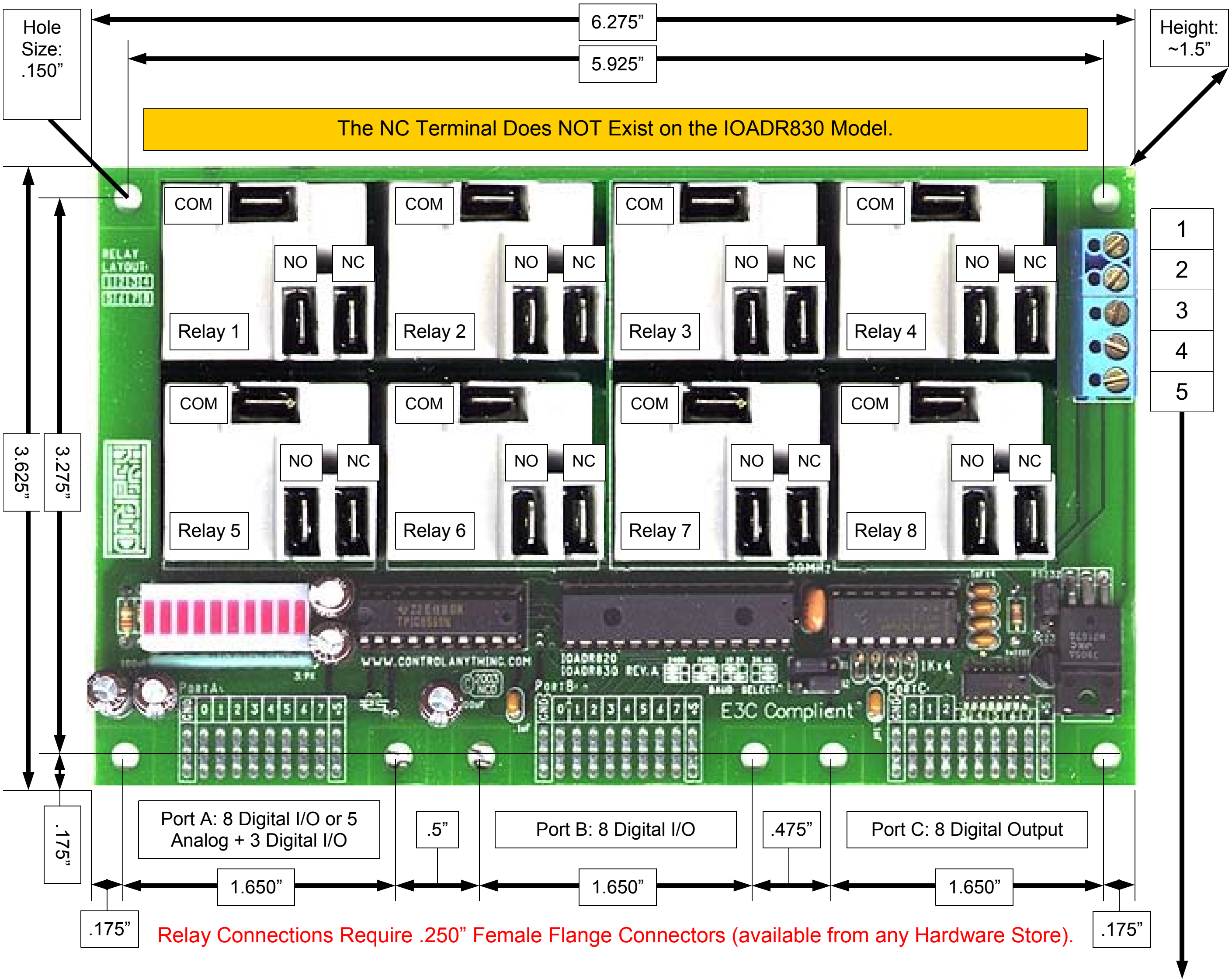
5) RS-232 Data Output
Connect this line to the RS-232 Data INPUT of your Computer or Microcontroller.

Jumper Settings

Set Jumpers to Configure Baud Rate. Settings are Printed on Circuit Board. Supported Baud Rates: 2400, 9600, 19.2K, or 38.4K Baud. Serial Configuration is 8 Data Bits, 1 Stop Bit, No Parity. This Controller Reads Jumper Settings when Power is First Applied to the Board ONLY.

Supported Output Standards: RS-232 (for 1 unit per serial port), or OC-232 (when connecting Multiple Controllers to Serial Port, Requires RSB Booster).

Outline Dimensions IOADR820, IOADR830



The NC Terminal Does NOT Exist on the IOADR830 Model.

Relay Connections Require .250" Female Flange Connectors (available from any Hardware Store).

NOTE: Port A can be used as 8-Bit Digital I/O OR 5-Channel A/D (8-Bit or 10-Bit) + 3-Bit Digital I/O. Port A Compatible with Character Display. Port B Compatible with Character Display OR Matrix Type Keypad up to 16 Keys. Port C Compatible with Character Display.

Relay Contact	Description
NC	Normally Closed: This Line Connects to Common when Relay is Turned Off
COM	Common: This is the Moving Portion of the Switch that Swings between NC and NO
NO	Normally Open: This Line Connects to Common ONLY When the Relay is Turned On

Power and Data Interface

1) +5 or 12 Volt Input
This pin supplies power to the board. ***MAKE SURE THE POWER JUMPER IS SET PROPERLY OR PERMANENT DAMAGE WILL RESULT***.

2) Power Ground
3) RS-232 Ground
These ground lines are tied together on the IOAD168 board.

4) RS-232 Data Input
Connect this line to the RS-232 Data OUTPUT of your Computer or Micro.

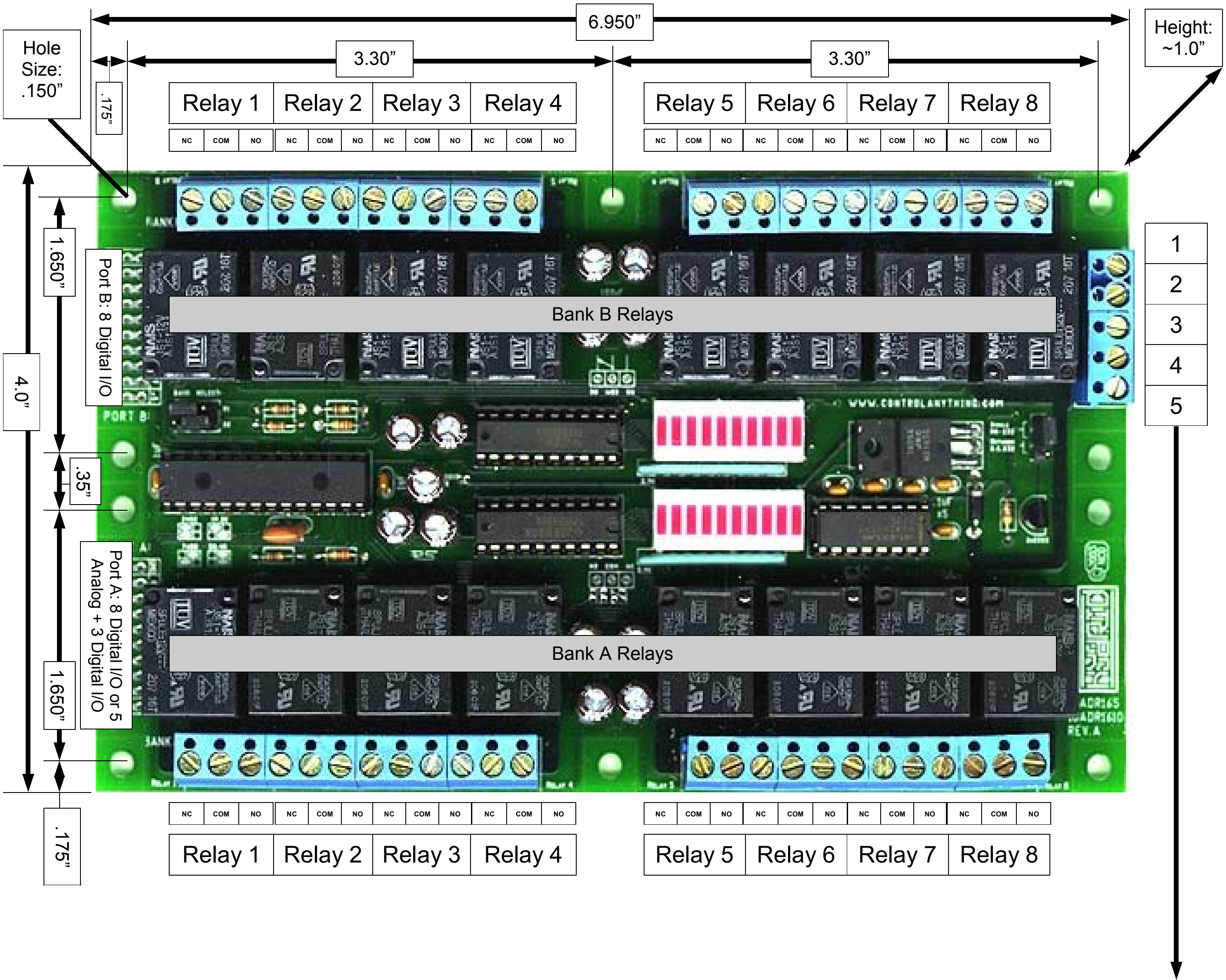
5) RS-232 Data Output
Connect this line to the RS-232 Data INPUT of your Computer or Microcontroller.

Jumper Settings

Set Jumpers to Configure Baud Rate. Settings are Printed on Circuit Board. Supported Baud Rates: 2400, 9600, 19.2K, or 38.4K Baud. Serial Configuration is 8 Data Bits, 1 Stop Bit, No Parity. This Controller Reads Jumper Settings when Power is First Applied to the Board ONLY.

Supported Output Standards: RS-232 (for 1 unit per serial port), or OC-232 (when connecting Multiple Controllers to Serial Port, Requires RSB Booster).

Outline Dimensions IOADR165, IOADR1610



NOTE: Port A can be used as 8-Bit Digital I/O OR 5-Channel A/D (8-Bit or 10-Bit) + 3-Bit Digital I/O. Port A Compatible with Character Display. Port B Compatible with Character Display OR Matrix Type Keypad up to 16 Keys.

Relay Contact	Description
NC	Normally Closed: This Line Connects to Common when Relay is Turned Off
COM	Common: This is the Moving Portion of the Switch that Swings between NC and NO
NO	Normally Open: This Line Connects to Common ONLY When the Relay is Turned On

Power and Data Interface

1) +5 or 12 Volt Input

This pin supplies power to the board. ***MAKE SURE THE POWER JUMPER IS SET PROPERLY OR PERMANENT DAMAGE WILL RESULT***.

2) Power Ground

3) RS-232 Ground

These ground lines are tied together on the IOAD168 board.

4) RS-232 Data Input

Connect this line to the RS-232 Data OUTPUT of your Computer or Micro.

5) RS-232 Data Output

Connect this line to the RS-232 Data INPUT of your Computer or Microcontroller.

Jumper Settings

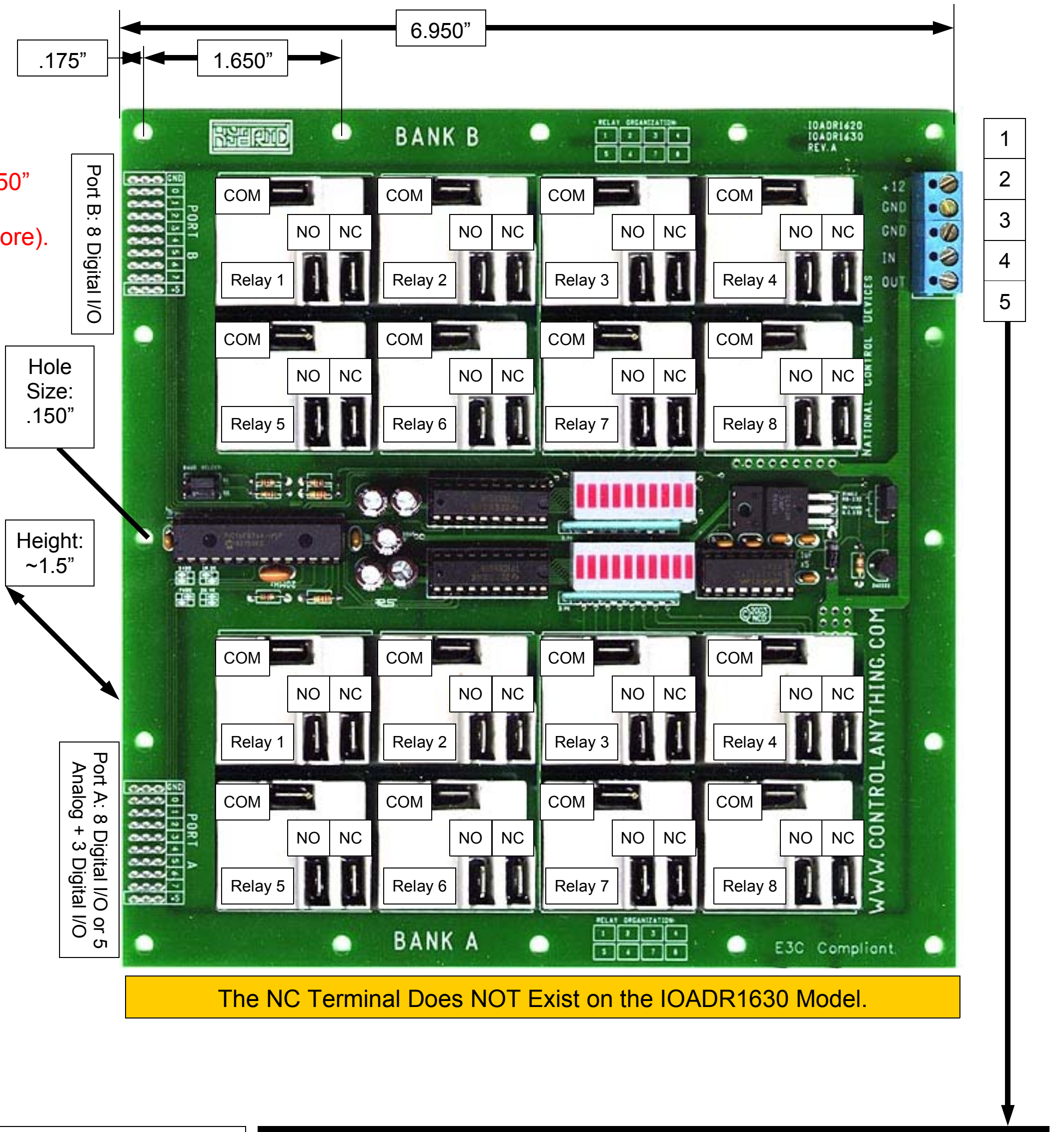
Set Jumpers to Configure Baud Rate. Settings are Printed on Circuit Board. Supported Baud Rates: 2400, 9600, 19.2K, or 38.4K Baud. Serial Configuration is 8 Data Bits, 1 Stop Bit, No Parity. This Controller Reads Jumper Settings when Power is First Applied to the Board ONLY.

Supported Output Standards: RS-232 (for 1 unit per serial port), or OC-232 (when connecting Multiple Controllers to Serial Port, Requires RSB Booster).

Outline Dimensions IOADR1620, IOADR1630

This Board is Exactly 6.95" x 6.95" Square. All 16 Mounting Holes are Equally Spaced at 1.650" Apart. All Holes are Inset .175"

Relay Connections Require .250" Female Flange Connectors (available from any Hardware Store).



The NC Terminal Does NOT Exist on the IOADR1630 Model.

NOTE: Port A can be used as 8-Bit Digital I/O OR 5-Channel A/D (8-Bit or 10-Bit) + 3-Bit Digital I/O. Port A Compatible with Character Display. Port B Compatible with Character Display OR Matrix Type Keypad up to 16 Keys.

Relay Contact	Description
NC	Normally Closed: This Line Connects to Common when Relay is Turned Off
COM	Common: This is the Moving Portion of the Switch that Swings between NC and NO
NO	Normally Open: This Line Connects to Common ONLY When the Relay is Turned On

Power and Data Interface

1) +5 or 12 Volt Input

This pin supplies power to the board. ***MAKE SURE THE POWER JUMPER IS SET PROPERLY OR PERMANENT DAMAGE WILL RESULT***.

2) Power Ground

3) RS-232 Ground

These ground lines are tied together on the IOAD168 board.

4) RS-232 Data Input

Connect this line to the RS-232 Data OUTPUT of your Computer or Micro.

5) RS-232 Data Output

Connect this line to the RS-232 Data INPUT of your Computer or Microcontroller.

Jumper Settings

Set Jumpers to Configure Baud Rate. Settings are Printed on Circuit Board. Supported Baud Rates: 2400, 9600, 19.2K, or 38.4K Baud. Serial Configuration is 8 Data Bits, 1 Stop Bit, No Parity. This Controller Reads Jumper Settings when Power is First Applied to the Board ONLY.

Supported Output Standards: RS-232 (for 1 unit per serial port), or OC-232 (when connecting Multiple Controllers to Serial Port, Requires RSB Booster).

Last Minute Design Notes and Changes

A/D Conversion Notes:

Port A may be used in one of two modes: 8-Bit Digital I/O or 5-Channel A/D + 3-Bit Digital I/O. When reading an Analog value from a single A/D input, all 5 A/D Channels become Analog Inputs (Port A[0-4]).

Extended Commands: 16 Relay Models Only

The R8x Pro Command set is used to control Banks of eight relays as discussed earlier in this manual. However, it is not possible to use the Pro Command Set to read or set the status of all 16 relays one time. We have added two additional commands for controllers with 16 relays that allow users to control/ read relays from all 16 relays using single command. These commands apply ONLY to the IOADR165, IOADR1610, IOADR1620, and IOADR1630 controllers.

Extended Commands: Setting 16 Relays

15, 0, 0-255, 0-255 Set Status of 16 Relays
This Command is used to set the Status of ALL 16 relays at one time. This command requires two parameters. The first parameter sets the status of Relay Bank A, the second parameter sets the status of Relay Bank B. Both Parameters have a valid range of 0-255. Relays will be activated in the equivalent binary pattern of the parameter value.

Extended Commands: Reading 16 Relays

15, 1 Get Status of 16 Relays
This Command is used to read the Status of ALL 16 relays at one time. This command sends two bytes of data back to the host computer. The first byte of data returned indicates the status of Bank A relays. The second byte of data returned indicates the status of Bank B relays. Each byte of data returned to the host computer has a valid range of 0-255. The binary pattern of the returned valued indicates the on/off status of each relay. The Least Significant Bit of Each Returned Value indicates the Status of Relay 1 on Bank A and Bank B Respectively.

Environmental Requirements:

NCD Hybrid relay controllers should be used in an environment of less than 80% Non-Condensing Humidity at a temperature range of 0-70* Celsius. Extended temperature range versions are available by special order.

Example Code: Setting 16 Relays

```
Public Sub SetAll16(BankA, BankB)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(15)           'Control 16 Relays Command
    MSComm1.Output = Chr$(0)            'Set the Status of 16 Relays
    MSComm1.Output = Chr$(BankA)        'Set Status of Bank A Relay
    MSComm1.Output = Chr$(BankB)        'Set Status of Bank B Relay
End Sub
```

Example Code: Setting 16 Relays

```
Public Sub GetAll16(BankA, BankB)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(15)           'Control 16 Relays Command
    MSComm1.Output = Chr$(1)            'Read the Status of 16 Relays
    'Controller will send two bytes of data back to the user
    BankA = GetData                     'Status of Bank A Relays
    BankB = GetData                     'Status of Bank B Relays
End Sub
```