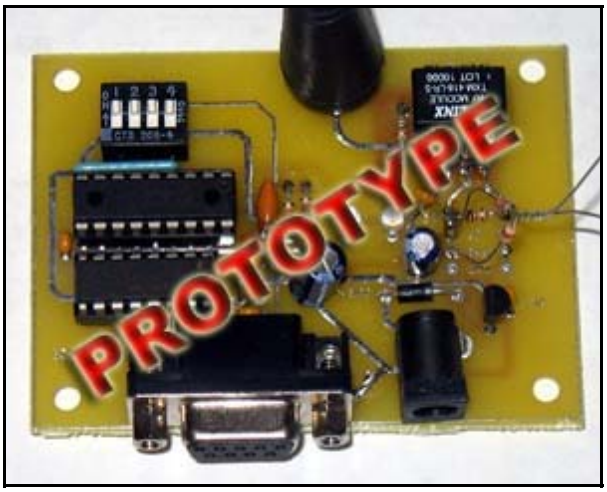
AirControl AC418LR

RS-232 to Wireless RF Upstream Data Encoder Transmitter

5-Year Repair or Replace Warranty!!!



Control an Unlimited Number of Remote Devices Individually or Simultaneously
Supports 2400 to 38.4K Baud Operation
Supports 2.5 to 10KBPS Data Transmission Speeds
Capable of Sending Commands up to 3,000 Feet
Transmit and Ready Status LEDs Indicate Device Operation
Software Configured Redundancy can Send Each Control Packet up to 255 Times
Superior 48/72-Bit Noise Rejection Protocol Ensures Reliable Consistent Operation
Includes Power Supply, Serial Cable, and Antenna

THIS DEVICE IS NOT FCC APPROVED.
THIS DEVICE SHOULD NEVER BE USED IN LIFE SAVING APPLICATIONS.
THIS DEVICE SHOULD NEVER BE USED IN APPLICATIONS THAT COULD RESULT IN INJURY OR LOSS OF LIFE IF THE DEVICE WERE TO FAIL

The AirControl AC418LR gathers control commands from your computer, attaches a noise rejection protocol, and transmits you control commands up to 3,000 feet to a remote wireless NCD device, such as a relay controller, light dimmer, or other future computer control products we may offer. The AirControl is very easy to use and requires no configuration when received; however, there are ways to increase performance, transmission range, and overall communications reliability by reading the information in the manual. The AirControl, for the most part, works very much like any other NCD device that is attached to the serial port of your computer. Simply send it a command and the device responds. In this case, simply send it a command, the command is transmitted, and the device responds. The AirControl knows nothing of the command set for the device you are trying to control, instead, it forwards your command to a remote device, where it is interpreted. This allows the AirControl to work with future devices with no firmware upgrades or compatibility issues.

FAST FACTS

PLEASE READ THIS SECTION IF YOU READ NOTHING ELSE!

- The Mechanical Drawing can be found on the Product Description Page for the AirControl on our web site at www.controlanything.com.
- A Terminal Program such as HyperTerminal is NOT SUITABLE for Sending Wireless Commands.
- This device transmits data to a remote device, communication is one way only. Use the AirMonitor to Receive Data from a Remote Device.
- Your program must be written to begin all data transmissions with the ASCII Character Code 254.
- Every time you send a command to the AirControl, ASCII Character code 85 will be sent back to you once transmission has finished.
- Your program must be written so that it waits for the “85” response code after every command sent.
- The command set for a remote device resides in the remote device, the AirControl knows nothing of the commands being sent.
- You can increase noise rejection by using “Keys” to remotely communicate with a remote device.
- You can use keys to control an unlimited number of devices, individually or simultaneously.
- A key is a device number. There are three keys that make up a complete device number.
- The AirControl Supports 2.5KBPS, 5KBPS, 7.5KBPS, and 10KBPS communication speeds.
- Bit rates can be controlled in software at any time, regardless of the default DIP switch settings.
- Changing bit rates in software does NOT change the default bit rate, which can only be set by the DIP switches.
- You can increase performance by lowering the Time Out and Send (TOS value), but this can increase communication errors if set too low.
- You can increase reliability and range significantly by sending each data packet 5 or more times.
- You can overcome noise significantly while boosting range by sending each data packet 50 or more times.
- You can change the number of packet transmissions at any time in software, and a default value can be programmed into non-volatile memory.
- The remote device accepts only one data packet and rejects all others, preventing commands from executing multiple times.
- A new command is defined as a new serial transmission from your computer to the AirControl transmitter.
- The AirControl has a 64-Byte Serial Receive Buffer.
- ***Applications written for communication with the AirControl attempt to read settings from the AirControl when executed. If there is a communication problem between the serial port and the AirControl, you will get a Type Mismatch Runtime Error.***

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, using the same shipping method used to ship the product to NCD.

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.

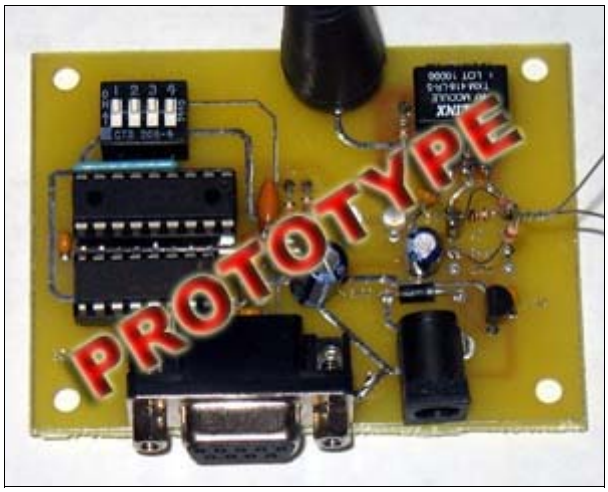
NCD Contact Information

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



BAUD RATE SELECTION:		
DIP 1	DIP 2	
OFF	OFF	2400
ON	OFF	9600
OFF	ON	19.2K
ON	ON	38.4K

DATA TRANSMISSION BIT RATE:		
DIP 3	DIP 4	
OFF	OFF	2.5K BPS
ON	OFF	5K BPS
OFF	ON	7.5K BPS
ON	ON	10K BPS

LED STATUS LIGHTS:	
LED1: LEFT SIDE OF PCB	LED2: RIGHT SIDE OF PCB
READY LED STAYS LIT UNTIL A PACKET OF DATA IS TRANSMITTED	TRANSMIT LED LIGHTS ONLY DURING RF TRANSMISSION

INCLUDED ACCESSORIES:
Antenna Serial Cable Power Supply 120VAC to 12VDC Computer Grade Regulated Switcher Type

Electrical Ratings	**Min**	**Typical**	**Max**
Input Voltage 2.1mm Barrel Connector, Center Positive	9VDC	12VDC	15VDC
Current Requirements	-	100 ma	-
Transmission Range	-	1,500 ft	3,000 ft
Transmission Frequency	-	418 MHz	-

Sending Commands to the AirControl

The AirControl is capable of sending and receiving data via RS-232 serial communications. The AirControl 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 sending data to the AirControl. 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 will need to make provisions for reading data from the serial port. Your programming language will support commands for reading data from the AirControl.

For your convenience, we have provided several programming examples in Visual Basic 6 for communicating with the AirControl. These examples should greatly speed development time. You may want to visit **www.controleverything.com** for the latest software and programming examples.

Programming examples for the AirControl 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.

Sending Wireless Commands

The AirControl works just like any other NCD serial device. It waits for a command then processes the command. The only difference it, the command isn't really processed by the AirControl. Instead, incoming commands are packaged up and transmitted wirelessly to a remote device. Once transmission has finished, the AirControl will send ASCII character code 85 back to your computer, signaling that it has finished processing your command.

Meanwhile, the remote device waits for valid data packets. Under normal operation, all kinds of noise is picked up by the receiver on the remote device. The processor on the remote device is constantly filtering though the incoming noise, waiting for a data packet that follows our proprietary format.

Once the data packet is received, the remote device decodes the data packet, checks it for errors, and then attempts to process the packet as a command. In other words, the remote device understands the command set, the AirControl only understands how to package up your command and send it out.

There are only a few rules you MUST follow when using the AirControl.

1) It is NOT possible to send random data to the AirControl, all data must start with 254, followed by a command sequence that fits the format of the remote device. We will show you some examples, this is just an introduction.
2) You can send commands directly to the AirControl, setting various parameters at any time. Commands that are directed to the AirControl must begin with ASCII character code 253. Again, we will show you some examples of this.
3) The AirControl needs time to transmit your data packets. It will tell you when it is finished by sending ASCII character code 85 upon completion of your transmission. Don't try to send more commands while the AirControl is transmitting, your commands will be ignored. So for every command you send, you will need to wait for an 85 response.
4) There is a 64-byte limit to the number of bytes you can send, which includes the 254 header byte. NCD devices typically have a command structure that is no more than 10-bytes, but we do have plans to make use of this extra buffer with future products.

Visual Basic Programming Examples

The following Visual Basic 6 Example source code demonstrates subroutines that work well with the AirControl. Keep in mind, all commands sent to the AirControl are acknowledged by sending ASCII character code 85 back to the host computer once processing and/or transmission has finished.

Sample Code: Receiving Responses from the AirControl

The following routine works well for reading data from the AirControl. When a command is sent, the GetData function should be called. GetData will contain the value 85 when the command has successfully completed. A Timeout value of 25,000 shown below may need to be increased or decreased for better performance. Example usage will be shown in other examples, but this is an essential routine for making sure the AirControl has finished it's last command.

```
Public Function GetData()
    Timeout = 0
    Do                                      'Wait for Device to Reply
      If Timeout > 25000 then Exit Function
      Timeout = Timeout + 1
    DoEvents                                'Allow Windows to MultiTask
    Until MSComm1.InBufferCount > 0         'If the Device Replies
    GetData = Asc(MSComm1.Input)            'Return Response Code
    Debug.Print GetData
End Sub
```

Sample Code: Sending Commands to a Wireless Relay Controller

The following commands work with our RxxW1LR series relay controllers. Keep in mind, this is just an example of how to send a wireless command to activate a relay. Actual Code may be a little different. The actual command set is defined by the remote device, NOT the AirControl.

```
Public Sub SetTheStatusOfAllRelays_Bank1()
    MSComm1.Output = Chr$(254) 'Enter Command Mode
    MSComm1.Output = Chr$(1)   'Control Relay Bank 1
    MSComm1.Output = Chr$(40)  'Set the Status of All Relays
    MSComm1.Output = Chr$(255) 'Activates All Relays on Remote Device
    GetData                    'Wait for Transmission to Finish
                               'The AirControl will Return 85
End Sub
```

Sample Code: Using the "Required Keys" Option

If the Remote Device has the "Required Keys" Option Enabled, you can send commands to a specific device. Here is an example of sending keys to "unlock" a command in a remote device. This routine assumes the remote device has been programmed with the Keys 25, 10, and 35. You can program each of the three keys to be anything from 0 to 255.

```
Public Sub SetTheStatusOfAllRelays_Bank1()
    MSComm1.Output = Chr$(254) 'Enter Command Mode
    MSComm1.Output = Chr$(25)  'Send Key 1
    MSComm1.Output = Chr$(10)  'Send Key 2
    MSComm1.Output = Chr$(35)  'Send Key 3
    MSComm1.Output = Chr$(1)   'Control Relay Bank 1
    MSComm1.Output = Chr$(40)  'Set the Status of All Relays
    MSComm1.Output = Chr$(255) 'Activates All Relays on Remote Device
    GetData                    'Wait for Transmission to Finish
                               'The AirControl will Return 85
End Sub
```

The EWC Command Set: Software Control of Multiple Wireless NCD Devices

Using the EWC Command Set, it is possible to remotely control any number of remote devices individually or simultaneously. EWC commands are used to “Enable” and “Disable” remote devices. Each remote device is identified by a unique Key, which is a set of three numbers that you store into the remote device. Once these three keys have been stored, you can use EWC commands to enable and disable the remote device. Using this simple protocol, it is very easy to control which devices are responding to your commands and which devices are ignoring your commands. Keep in mind, EWC commands are the only set of commands that are NEVER ignored. NOTE: THE EWC COMMAND SET DOES NOT WORK ON REMOTE DEVICES THAT HAVE THE “KEYS REQUIRED” OPTION ENABLED. IN GENERAL, THE “KEYS REQUIRED” OPTION IS A LITTLE EASIER TO USE, BUT COMMUNICATIONS IS SLOWER, AND IS LIMITED TO CONTROLLING ONLY ONE DEVICE AT A TIME.

The EWC command set is virtually identical to the E3C command set found on our directly wired RS-232 devices. There are only a few minor differences. Three keys are used to replace a single device number. The AirControl does not really know anything about EWC commands, it just transmits the command codes to all wireless devices within range. The remote device then determines if it should be enabled or disabled. Since the AirControl is used for communication of EWC commands, the AirControl will report back an 85 after processing, which is another notable difference from standard E3C commands. Since our wireless LR Series controllers are only one way, it is not possible to retrieve the keys or other settings from the remote device. So take note of your keys (device number) settings stored within your remote devices. Lastly, E3C command 253 was eliminated from the EWC command set due to its lack of use.

Networking Comparison

As you probably already know, it is possible to control an unlimited number of devices remotely. There are two ways of doing this, depending on your preference. Devices are identified by a device number, which consists of three number, each set to a value between 0 and 255. You will need to program the remote device with a device number, usually done when the remote device is in “Program” mode. When the remote device is in “Run” mode, it is ready to respond to your commands.

Require Keys Method
This is a very convenient protocol, very easy to implement that allows you to control one remote device at a time. Communications is slower, but this method offer superior 72-bit noise rejection. When the “Require Keys” options is enabled on a remote device, keys must be immediately transmitted after the 254 command mode header. Only the device with correct key will respond. This is one our favorite methods of remote control because of it’s ease of implementation….just add the three key numbers after the 254 and before the actual command code. It really doesn’t get any easier than to use the Required Keys Method.

EWC Networking
EWC cuts down on the number of data bytes sent out the serial port, which greatly increases communication speed. Noise rejection is limited to 42 bits using this method. However, it is possible to control multiple devices at one time by enabling and disabling remote devices prior to sending commands. Only the enabled devices will accept and process your commands. The apropriate command and three keys can be transmitted to control which devices are enabled (listening to your control commands) and which devices are disabled (ignoring your control commands).

The EWC 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.
You will need to send 3 keys, identifying the device you would like to enable.

251 Disable Selected Device:
Tells a specific device to ignore your commands.
You will need to send 3 keys, identifying the device you would like to disable.

252 Enable Selected Device Only:
Tells a specific device to listen to your commands, all other devices will ignore your commands. You will need to send 3 keys, identifying the device you would like to enable.

Sample Code: The EWC Command Set

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

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

Public Sub EnableSpecificDevice(Device)
    'Enable A Specific EWC Devices, Other Devices will be unchanged
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(250)    'EWC Disable Specific Device Command
    MSComm1.Output = Chr$(Key1)   'Device Number (Key 1) to Enable
    MSComm1.Output = Chr$(Key2)   'Device Number (Key 2) to Enable
    MSComm1.Output = Chr$(Key3)   'Device Number (Key 3) to Enable
    GetData
End Sub

Public Sub DisableSpecificDevice(Device)
    'Disable A Specific EWC Devices, Other Devices will be unchanged
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(251)    'EWC Disable Specific Device Command
    MSComm1.Output = Chr$(Key1)   'Device Number (Key 1) to Disable
    MSComm1.Output = Chr$(Key2)   'Device Number (Key 2) to Disable
    MSComm1.Output = Chr$(Key3)   'Device Number (Key 3) to Disable
    GetData
End Sub

Public Sub DisableAllDevicesExcept(Device)
    'Disable All EWC Devices Except (Device)
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(252)    'EWC Disable All Device Except Command
    MSComm1.Output = Chr$(Key1)   'Device Number (Key 1) to Enable
    MSComm1.Output = Chr$(Key2)   'Device Number (Key 2) to Enable
    MSComm1.Output = Chr$(Key3)   'Device Number (Key 3) to Enable
    GetData
End Sub
```

AirControl Advanced Configuration Settings

It should be stated that the AirControl is ready to run, right out of the box. You don't really need to do anything but start sending commands to remote devices. Having said that, we recognize that there are users who are going to want the best possible performance, or to be able to change settings on the fly, either to increase performance or improve reliability. The AirControl has a built in command set that you may find very useful for on the fly setting changes, allowing you to greatly improve performance or long range communication. It should be stated that this section is completely optional, if you have no interest in tweaking some performance parameters, then skip this section.

If you are familiar with the command structure of NCD devices, you will find that up until this point, you will always use ASCII character code 254 to enter command mode. This system works great for communicating to a device. But communications devices need to have their own command structure, leaving the "254" system untouched. You can speak to the AirControl at any time using the "253" command structure. The "253" command structure enters command mode for the AirControl, leaving the "254" system alone.

To simplify, you will use 254 to enter command mode on NCD devices. You will use 253 to enter command mode on NCD communication devices.

Here you will find some examples of issuing different commands to the AirControl. These commands are never transmitted, they are used exclusively to communicate to the AirControl hardware memory.

Advanced Communication Tips

- Remote devices that are outside normal communications range may become accessible if the remote device is set to a lower bit rate. Use Command 4 (below) to lower the bit rate at any time to communicate to distant devices. Use command 4 to change the bit rate back up to 10KBPS for faster communications with remote devices that are close by.
- Use Command 5 (below) with a parameter value of 255 for distant devices to improve communication range. You can lower the value at any time to 5 or 10 to communicate to nearby devices at a faster speed.
- Use Command 4 and 5 together to significantly increase communication range or speed. In some environments, it may be more beneficial to only change Command 5. It all depends on how much electrical interference you have in your environment.
- Use Command 5 (below) with a parameter value of 255 for commands that you consider to be "important". For instance, if you want to turn all the lights off in a building at the end of a day, your commands will be transmitted 255 times, helping to insure all devices respond to your commands.

AirControl Command Set

0 Test 2-Way Communication:
This command sends 85 back to the computer, used for checking 2-way communication between the AirControl and the Computer.

1 Get Settings:
This Command retrieves two non-volatile values out of the AirControl memory. The first value returned will contain the stored "Redundancy" value, which contains the default number of times each command packet is sent. The second byte returned contains the Time Out and Send (TOS) value. Which can be used to increase communication speed (lowering this value increase speed, but can cause communication errors).

2 Store Settings:
This command requires two parameters in the following order: Redundancy and TOS, explained above.

3 Restore Default Settings:
This command restores the AirControl to known Safe factory default settings.

4 Set Bit Rate:
This command allows you to change the bit rate of data transmissions at any time. This setting over rides the dip switch settings, but it does not change the power up default settings. This command requires a parameter of 0-3, setting bit rates of 2.5K, 5K, 7.5K, and 10KBPS respectively.

5 Set Redundancy:
This command overrides, but does not alter the power up default setting. This command requires a parameter of 1 to 255, indicating how many times each command is transmitted. A greater value increases communication range and reliability while a lower value increases communication speed.

Sample Code: The AirControl Command Set

```
Public Sub Test2Way()
    MSComm1.Output = Chr$(253)    'Enter AirControl Command Mode
    MSComm1.Output = Chr$(0)      'Test 2-Way Communications
    GetData                       'AirControl Responds with 85
End Sub

Public Sub GetStoredSettings()
    MSComm1.Output = Chr$(253)    'Enter AirControl Command Mode
    MSComm1.Output = Chr$(1)      'Get Settings
    GetData                       'AirControl Responds with Redundancy
    GetData                       'AirControl Responds with TOS Value
End Sub

Public Sub StoreDefaultSettings(Redun,TOS)
    MSComm1.Output = Chr$(253)    'Enter AirControl Command Mode
    MSComm1.Output = Chr$(2)      'Store Default Settings Command
    MSComm1.Output = Chr$(Redun)  'Store Redundancy Value
    MSComm1.Output = Chr$(TOS)    'Store TOS Value
End Sub

Public Sub RestoreDefaultSettings()
    MSComm1.Output = Chr$(253)    'Enter AirControl Command Mode
    MSComm1.Output = Chr$(3)      'Set to Factory Default Settings
    GetData                       'AirControl Responds with 85
End Sub

Public Sub ChangeBitRate(BitRate)
    MSComm1.Output = Chr$(253)    'Enter AirControl Command Mode
    MSComm1.Output = Chr$(4)      'Change BitRate Command
    MSComm1.Output = Chr$(BitRate)'BitRate = 0-3 0=2.5K 3=10K
    GetData                       'AirControl Responds with 85
End Sub

Public Sub ChangeRedundancy(Redun)
    MSComm1.Output = Chr$(253)    'Enter AirControl Command Mode
    MSComm1.Output = Chr$(5)      'Change Redundancy Value
    MSComm1.Output = Chr$(Redun)  'Commands are Sent 1-255 Times
    GetData                       'AirControl Responds with 85
End Sub
```

AirControl Applications

We have not developed any software for controlling the AirControl directly. Since the AirControl is used in combination with remote devices, we decided it would be better to integrate AirControl configuration into applications that apply to a specific remote device. This will allow you to experiment with the remote device and the AirControl at the same time. ***Applications written for communication with the AirControl attempt to read settings from the AirControl when executed. If there is a communication problem between the serial port and the AirControl, you may receive an error message.***

To keep our software simple, we did NOT include any error checking into our programs. Keep in mind, the AirControl MUST be connected to a Valid COM Port (1-6), and set at 38.4K Baud for proper operation.

NOTICE: OUR VISUAL BASIC EXAMPLE PROGRAMS AND SOFTWARE ARE ALWAYS AVAILABLE FOR DOWNLOAD ON OUR WEB SITE.

Below:
AirSwitch is a program you can install on your computer to remotely control relays. AirSwitch includes a full range of AirControl Functions, allowing you to tweak the AirControl for maximum performance and communications range. AirSwitch also supports EWC and Keys Required networking protocols. Source Code is Included in the Archive for Visual Basic 6.

Note that some settings can cause loss of communication between the AirControl and the remote device, we encourage our customers to experiment with the different settings to see how they function. Take note, the AirControl MUST be connected to a valid COM port and set for 38.4K Baud operation for our software to function.