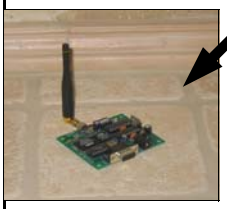
Ultra Series Expansion Modules

Designed to Grow with your Needs

One of the most powerful features of the Ultra Series controllers is their ability to grow as your needs grow. Instead of replacing hardware, Ultra Controllers allow you to continuously add new hardware by chaining a large variety of expansion modules to the expansion ports (also known as Ultra Ports). By chaining expansion modules, you can build the control system that best suit your specific application. Our initial release of seven expansion modules, supported by version 2.0 firmware, will handle most control applications you may require. But rest assured, many more expansion modules are planned for release in the coming years as we standardize our products and product line to an expansion oriented architecture.

It All Starts Here: A TranSendES is Connected to a PC, Offering a Wireless Connection to NCD Devices such as the WIOADXR.



Ultra Series Controllers Communicate to the TranSendES via Wireless Communications. The WIOADXR Shown Below becomes the Central "Hub" for all of the expansion modules. This device can also be used by itself, with 16 I/O Lines and an RS-232 Direct Connection to your PC.

And to think the controller shown above is using only 6.8% of its total expansion capabilities...

With the capacity to control 512 relays, read 4096 contact closure/voltage detection inputs, 96 channels of 12 Bit A/D, the WIOAXR is shown above with only 112 relays attached to its output, 96 voltage detection inputs, 96 contact closure inputs, and 16 Channels 12 Bit A/D...and the network is ready for more anytime you are. But if that still isn't enough, you can communicate with 1.67 Million WIOADXR controllers.

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

Version 2.0 Firmware, Release Date, Aug. 22, 2005

Firmware Version 2.0
Ultra Series Firmware Release Version 2.0 Begins Shipping on August 22, 2005. All products ordered on or after this date will receive version 2.0 firmware. Firmware upgrades for existing controllers are available by contacting ryan@controlanything.com.

Version 2.0 Firmware Identification

Version 2.0 firmware is identified by showing V2.0 on the optional character LCD display. The firmware version is also displayed using the Ultra Configuration utility (Firmware version is shown on the bottom center of the configuration window). Version 2.0 has three components:

V2.0 Product Enhancements:

TranSend Encoder Chip Update

The TranSend Encoder chip was updated to communicate data slightly faster. In addition, new default parameters have been established. SID error checking is not recommended for version 2.0 firmware, but is still usable. The greatest speed benefit and reliability is achieved by keeping SID error checking off. Data transmission routines were updated to improve range. Effective communication distance has been tested at over 1000 feet, up from 500 feet using version 1.0 firmware.

Software Updates

The Configuration and Test software also received a new parameter that allows you to adjust how many times data packets are sent back to the computer. A good value is 2 or 3, but you can choose to transmit return data packets up to 10 times. This parameter greatly increases communication range, but there is a serious speed penalty for every value you choose, so keep the number as low as possible.

In addition, new Version 2.0 test software is included in the archive. Please see below for additional information on test software.

Ultra Series Firmware Controller Updates

The Ultra Series Firmware has been updated for better communication range back to a PC. Effective communication range is now over 1000 feet with improved speed and reliability. Firmware upgrades also allow you to send multiple data packets back to the remote computer, improving range and reliability, but this parameter increase comes at a sacrifice of communication speed.

Version 2.0 Firmware Now supports many new expansion modules, allowing you to read up to 96 12-Bit Analog Channels and up to 4096 Switch Closure and/or Voltage Detector inputs (in any combination). The Ultra Series controllers can now be relied upon for very serious security and monitoring applications. With the new power of highly expandable inputs, Ultra Series controllers can read incoming data just as well as it is able to control large arrays of relays.

The following new expansion devices can now be plugged into expansion ports A or B to significantly increase the input monitoring capabilities of the Ultra Series controllers:

UAD1216	***16-Channel 12-Bit A/D Converter. You can chain up to 3 of these on each I/O expansion port, allowing you to read up to 96 12-Bit or 8-Bit Analog Voltages from 0-5VDC.***
USCV16x	***Ultra Expansion Module ScanVolt 16 Channel. This device is used to detect the presence of voltages. Each input is 100% optoisolated with a wide user-controlled input voltage range. Daisy Chain multiple expansion modules to the same I/O buss. This device counts as 2 Input Banks on the I/O Expansion port. Up to 256 Banks allowed per port.***
USCV32x	***Same as Above, but with 32 Channels, Counts as 4 Input Banks.***
USCV48x	***Same as Above, but with 48 Channels, Counts as 6 Input Banks.***
USCS16x	***Ultra Expansion Module ScanSwitch 16 Channel. This device is used to detect switch closures, useful for motion detection and light switch monitoring. Daisy Chain multiple expansion modules to the same I/O buss. This device counts as 2 Input Banks on the I/O Expansion port. Up to 256 Banks allowed per port.***
USCS32x	***Same as Above, but with 32 Channels, Counts as 4 Input Banks.***
USCS48x	***Same as Above, but with 48 Channels, Counts as 6 Input Banks.***

Understanding Banks

Understanding the Concept of Banks will become increasingly important as you expand your Ultra Series controller. By this point, you have run across the term "Bank" as it applies to a group of eight relays. Similarly, "Banks" will be used as a generic term for "a grouping of eight". A bank of inputs is a group of eight inputs. Using some of our new expansion modules, a "Bank" parameter will be required. In many cases, you can read inputs from 256 banks (which translates into 2048 individual inputs). Using banks, you will be able to read switch closure inputs using our ScanSwitch expansion modules. Our ScanVolt expansion module is used to detect the presence of a voltage (which is very useful for confirming the actual switch closure of a relay). Complete details will be discussed as we introduce you to our line of Ultra Series expansion modules.

Version 2.0 Firmware is Required for All Programming and Examples and Expansion Modules Shown Beyond this Point

UAD1216: 16-Channel 8-Bit / 12-Bit Analog to Digital Conversion

UAD1216 Introduction

The UAD1216 is an Ultra Series expansion module designed to plug into the Ultra Series Expansion Port A or B. Up to three UAD1216 expansion modules can be connected to a single port, allowing up to six UAD1216 expansions per ultra series controller (for a total of 96 Channels of Analog to Digital conversion).

What is Analog to Digital Conversion?
A/D conversion is the process of converting a voltage to a numeric value. A/D conversion is useful for reading light or sound levels in a room, temperature (using a thermistor), water levels (using only a couple of wires), not to mention all kinds of analog sensors such as pressure sensors, thermo-couples, and much more. Think of the AD1216 as sixteen volt meters, each capable of reading voltages from 0-5 DC volts. Instead of each volt meter having a display, the voltage values are delivered to your PC for manipulation by your own program.

8-Bit Conversion
The user may request an 8-Bit A/D conversion. When the appropriate command is executed, the Ultra Controller will deliver a value from 0-255 indicating voltage. A value of 0 will be returned if no voltage is present on the input. A value of 255 will be return if the voltage is at 5 volts on the input. A value of 128 will be returned if the voltage is at 2.5 volts. An 8-Bit A/D conversion breaks apart the input into 256 equal steps, and is sensitive to voltage changes as low as .192 volts.

12-Bit Conversion
The user may also request a 12-Bit A/D conversion. When the appropriate command is execute, the Ultra Controller will deliver a value from 0-4095 indicating voltage. A value of 0 will be returned if no voltage is present on the input. A value of 4095 will be returned if the voltage is at 5 volts on the input. A value of 2048 will be returned if the voltage is at 2.5 volts. A 12-Bit A/D conversion breaks apart the input into 4096 equal steps, and is sensitive to voltage changes as low as .00122 volts. A 12-Bit A/D conversion takes just as long as an 8-Bit A/D conversion. However, 12-Bit conversions appear to take longer because data has to be delivered back to the computer using two bytes instead of just one for 8-Bit conversions. These two bytes are then "re-assembled" into a 0-4095 value. More on this later.

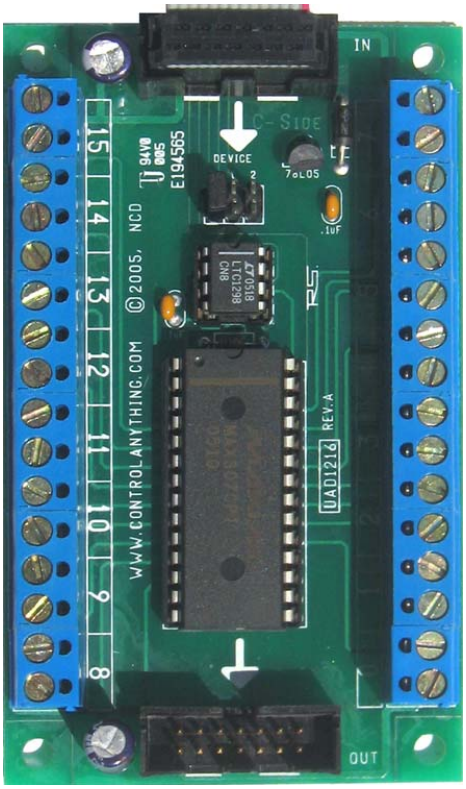
Multi-Channel Conversion
Ultra Series controllers allow you to read a single channel or all 16 inputs using a single request. Requesting all 16 input channels at one time can significantly improve communication speed.

Expansion modules are designed to be chained off the I/O port buss. You can easily mix voltage detectors, switch scanners, and analog to digital converters off the same data port. No power supply is required for this expansion module, the I/O port provides the required power supply.

This Connector Attaches to the Ultra Series Controller

White Arrows Printed on the Circuit Board Should ALWAYS Point AWAY from the Main Controller

Ground	Input Channel 7
Input Channel 15	Ground
Ground	Input Channel 6
Input Channel 14	Ground
Ground	Input Channel 5
Input Channel 13	Ground
Ground	Input Channel 4
Input Channel 12	Ground
Ground	Input Channel 3
Input Channel 11	Ground
Ground	Input Channel 2
Input Channel 10	Ground
Ground	Input Channel 1
Input Channel 9	Ground
Ground	Input Channel 0
Input Channel 8	Ground



This Connector Attaches to More Expansion Modules
(Up to 2 more UAD1216 Expansion Boards per Ultra Port)

All Inputs are 0-5VDC. Power to this Device is Derived from the Ultra Expansion Port.

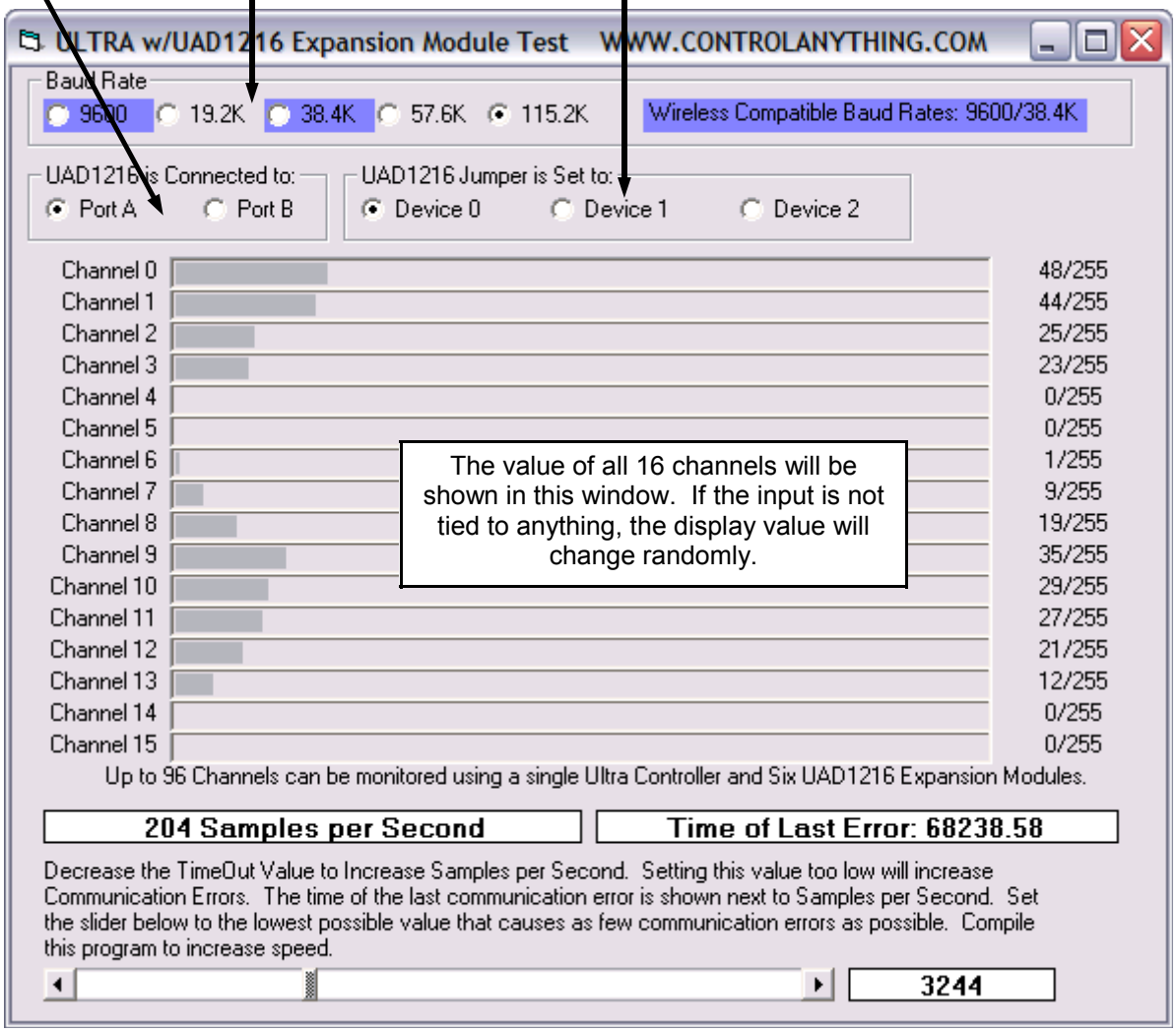
The UAD1216 Must Be Connected to the I/O Data Buss (Ultra Port) First. Voltage Detectors and Contact Closure Input Expansions Must be at the End of the Chain.

UAD1216: 16-Channel 8-Bit / 12-Bit Analog to Digital Conversion

The UAD1216 Can be Connected to Port A or Port B, Select the Port it is Connected to.

Select a Baud Rate

Up to 3 UAD1216s can Connect to a Single Port, Select the Device you would Like to Communicate with. Make Sure the Device Jumper on the UAD1216 Matches this Setting.



The value of all 16 channels will be shown in this window. If the input is not tied to anything, the display value will change randomly.

Program Variations in the Examples.ZIP File:

UAD1216 16-Channel One at a Time 8-Bit (COM1-COM6 Supported)
Reads One Channel at a Time, All 16 Channels, 8-Bits per Channel.

UAD1216 16-Channel One at a Time 12-Bit (COM1-COM6 Supported)
Reads One Channel at a Time, All 16 Channels, 12-Bits per Channel.

UAD1216 16-Channel Requests 8-Bit (COM1-COM6 Supported)
Reads All 16 Channels at a Time, 8-Bits per Channel.

UAD1216 16-Channel Requests All 12-Bit (COM1-COM6 Supported)
Reads All 16 Channels at a Time, 12-Bits per Channel.

UAD1216: 16-Channel 8-Bit / 12-Bit Analog to Digital Conversion

Programming Examples

Version 2.0 Firmware includes a new set of commands that allow you to communicate to the UAD1216 on Port A or Port B. You can also request data one channel at a time or all 16 channels at once. You may also request 8-Bit values (0-255), or high resolution 12-Bit values (0-4095). Up to 3 devices are supported per port, for a total of 96 analog inputs per Ultra controller.

254, 12, Device, Channel: 8-Bit Single Channel Port A

This command reads a single 8-Bit channel from the UAD1216 Expansion Module connected to Expansion Port A. This command requires a Device parameter (0, 1, or 2), which should match the Device jumper setting of the UAD1216 you would like to read from. This command also requires a Channel parameter, indicating which input channel to read from (0-15). This command will return a single byte of data, from 0 to 255, indicating the analog value on the specified input channel.

254, 13, Device, Channel: 8-Bit Single Channel Port B

This command reads a single 8-Bit channel from the UAD1216 Expansion Module connected to Expansion Port B. This command requires a Device parameter (0, 1, or 2), which should match the Device jumper setting of the UAD1216 you would like to read from. This command also requires a Channel parameter, indicating which input channel to read from (0-15). This command will return a single byte of data, from 0 to 255, indicating the analog value on the specified input channel.

254, 14, Device: 8-Bit All Channels Port A

This command reads all 16 8-Bit channels from the UAD1216 Expansion Module connected to Expansion Port A. This command requires a Device parameter (0, 1, or 2), which should match the Device jumper setting of the UAD1216 you would like to read from. This command will return sixteen bytes of data, from 0 to 255, indicating the analog values of all input channels, beginning with channel 0.

254, 15, Device: 8-Bit All Channels Port B

This command reads all 16 8-Bit channels from the UAD1216 Expansion Module connected to Expansion Port B. This command requires a Device parameter (0, 1, or 2), which should match the Device jumper setting of the UAD1216 you would like to read from. This command will return sixteen bytes of data, from 0 to 255, indicating the analog values of all input channels, beginning with channel 0.

254, 16, Device, Channel: 12-Bit Single Channel Port A

This command reads a single 12-Bit channel from the UAD1216 Expansion Module connected to Expansion Port A. This command requires a Device parameter (0, 1, or 2), which should match the Device jumper setting of the UAD1216 you would like to read from. This command also requires a Channel parameter, indicating which input channel to read from (0-15). This command will return two bytes of data (LSB, then MSB), which will be re-assembled by your program to a value of 0-4095 using the equation: Value = (MSB * 256) + LSB.

254, 17, Device, Channel: 12-Bit Single Channel Port B

This command reads a single 12-Bit channel from the UAD1216 Expansion Module connected to Expansion Port A. This command requires a Device parameter (0, 1, or 2), which should match the Device jumper setting of the UAD1216 you would like to read from. This command also requires a Channel parameter, indicating which input channel to read from (0-15). This command will return two bytes of data (LSB, then MSB), which will be re-assembled by your program to a value of 0-4095 using the equation: Value = (MSB * 256) + LSB.

8-Bit Single Channel Port A

```
MSComm1.Output = Chr$(254) 'Command Mode
MSComm1.Output = Chr$(12)  'UAD1216 Port A 8-Bit Commands
MSComm1.Output = Chr$(0)   'UAD1216 Device 0
MSComm1.Output = Chr$(0)   'Request 8-Bit A/D Channel 0
GetI                       'GetI Contains the 8-Bit A/D Value
```

8-Bit Single Channel Port B

```
MSComm1.Output = Chr$(254) 'Command Mode
MSComm1.Output = Chr$(13)  'UAD1216 Port B 8-Bit Commands
MSComm1.Output = Chr$(2)   'UAD1216 Device 2
MSComm1.Output = Chr$(15)  'Request 8-Bit A/D Channel 15
GetI                       'GetI Contains the 8-Bit A/D Value
```

8-Bit All Channels Port A

```
MSComm1.Output = Chr$(254)     'Command Mode
MSComm1.Output = Chr$(14)      'UAD1216 Port A 8-Bit Commands
MSComm1.Output = Chr$(1)       'UAD1216 Device 1
For Channel = 0 To 15          'Count Through Returned Data
   Debug.print Channel;GetI    'Get Data Byte from Controller
Next Channel
```

8-Bit All Channels Port B

```
MSComm1.Output = Chr$(254)     'Command Mode
MSComm1.Output = Chr$(15)      'UAD1216 Port B 8-Bit Commands
MSComm1.Output = Chr$(1)       'UAD1216 Device 1
For Channel = 0 To 15          'Count Through Returned Data
   Debug.print Channel;GetI    'Get Data Byte from Controller
Next Channel
```

12-Bit Single Channel Port A

```
MSComm1.Output = Chr$(254) 'Command Mode
MSComm1.Output = Chr$(16)  'UAD1216 Port A 12-Bit Commands
MSComm1.Output = Chr$(0)   'UAD1216 Device 0
MSComm1.Output = Chr$(7)   'Request 12-Bit A/D Channel 7
LSB = GetI                 'GetI Contains the LSB 12-Bit A/D Value
LSB = GetI                 'GetI Contains the MSB 12-Bit A/D Value
Value = (MSB*256)+LSB      'This Equation Converts to 12-Bit Value
Debug.Print Value          'Show 12-Bit A/D Value in the Debug Window
```

12-Bit Single Channel Port B

```
MSComm1.Output = Chr$(254) 'Command Mode
MSComm1.Output = Chr$(17)  'UAD1216 Port B 12-Bit Commands
MSComm1.Output = Chr$(0)   'UAD1216 Device 0
MSComm1.Output = Chr$(7)   'Request 12-Bit A/D Channel 7
LSB = GetI                 'GetI Contains the LSB 12-Bit A/D Value
LSB = GetI                 'GetI Contains the MSB 12-Bit A/D Value
Value = (MSB*256)+LSB      'This Equation Converts to 12-Bit Value
Debug.Print Value          'Show 12-Bit A/D Value in the Debug Window
```

Reading Data from the Controller

The ***GetI*** Function Below waits for data from the controller. If data is not received before the timeout period, the function will exit, containing no value.

```
Public Function GetI()
    T = 0
    Do
        T = T + 1
        If T > 10000 then Exit Function
        DoEvents
    Loop Until MSComm1.InBufferCount > 0
    GetI = Asc(MSComm1.Input)
End Function
```

UAD1216: 16-Channel 8-Bit / 12-Bit Analog to Digital Conversion

254, 18, Device: 12-Bit All Channels Port A

This command reads all 16 8-Bit channels from the UAD1216 Expansion Module connected to Expansion Port A. This command requires a Device parameter (0, 1, or 2), which should match the Device jumper setting of the UAD1216 you would like to read from. This command will return 32 bytes of data indicating the 12-Bit analog values of all 16 input channels, beginning with channel 0. The first two bytes of data indicate the analog values for channel 0. The next two bytes of data indicate the analog values for channel 1. The process ends by sending the final two bytes of data indicating the 12-Bit analog value for channel 15. Each pair of data bytes are send in the order LSB, then MSB, which stands for the Least Significant Byte and Most Significant byte. The 12-Bit value is re-assembled by your program to a value of 0-4095 using the equation: Value = (MSB * 256) + LSB. This equation will be used 16 times (for each pair of 32 bytes), during the conversion process.

12-Bit All Channels Port A

```
MSComm1.Output = Chr$(254) 'Command Mode
MSComm1.Output = Chr$(18)  'UAD1216 Port A 12-Bit Commands
MSComm1.Output = Chr$(0)   'UAD1216 Device 0
For Ch = 0 to 15           'Count Through Channels
   LSB = GetI              'GetI Contains the LSB 12-Bit A/D Value
   LSB = GetI              'GetI Contains the MSB 12-Bit A/D Value
   Value = (MSB*256)+LSB   'This Equation Converts to 12-Bit Value
   Debug.Print Ch;Value    'Show 12-Bit A/D Value in the Debug Window
Next Ch
```

254, 19, Device: 12-Bit All Channels Port B

This command reads all 16 8-Bit channels from the UAD1216 Expansion Module connected to Expansion Port B. This command requires a Device parameter (0, 1, or 2), which should match the Device jumper setting of the UAD1216 you would like to read from. This command will return 32 bytes of data indicating the 12-Bit analog values of all 16 input channels, beginning with channel 0. The first two bytes of data indicate the analog values for channel 0. The next two bytes of data indicate the analog values for channel 1. The process ends by sending the final two bytes of data indicating the 12-Bit analog value for channel 15. Each pair of data bytes are send in the order LSB, then MSB, which stands for the Least Significant Byte and Most Significant byte. The 12-Bit value is re-assembled by your program to a value of 0-4095 using the equation: Value = (MSB * 256) + LSB. This equation will be used 16 times (for each pair of 32 bytes), during the conversion process.

12-Bit All Channels Port B

```
MSComm1.Output = Chr$(254) 'Command Mode
MSComm1.Output = Chr$(19)  'UAD1216 Port B 12-Bit Commands
MSComm1.Output = Chr$(0)   'UAD1216 Device 0
For Ch = 0 to 15           'Count Through Channels
   LSB = GetI              'GetI Contains the LSB 12-Bit A/D Value
   LSB = GetI              'GetI Contains the MSB 12-Bit A/D Value
   Value = (MSB*256)+LSB   'This Equation Converts to 12-Bit Value
   Debug.Print Ch;Value    'Show 12-Bit A/D Value in the Debug Window
Next Ch
```

Reading Data from the Controller

The ***GetI*** Function Below waits for data from the controller. If data is not received before the timeout period, the function will exit, containing no value.

```
Public Function GetI()
    T = 0
    Do
        T = T + 1
        If T > 10000 then Exit Function
        DoEvents
    Loop Until MSComm1.InBufferCount > 0
    GetI = Asc(MSComm1.Input)
End Function
```

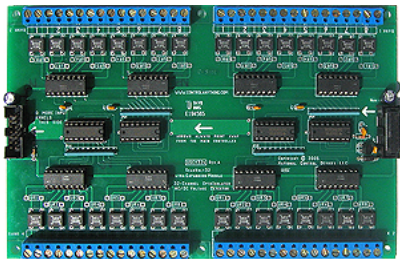
USCVxxx: 16/32/48-Channel Voltage Detector (ScanVolt)

Is it Really On?

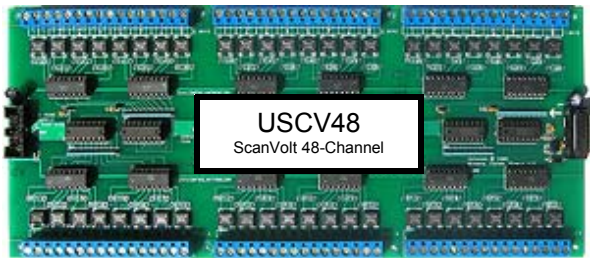
New Expansion Detects Voltages

So... you turned it on, but did it actually come on? Our new voltage detector expansion modules can help you find up to 4096 potential voltage problems.

USCV16
ScanVolt 16-Channel



USCV32
ScanVolt 32-Channel



USCV48
ScanVolt 48-Channel

ScanVolt USCVxxx Introduction

Ultra ScanVolt 16, 32, and 48 Expansion Modules allow you to detect the presence of a voltage. This is useful for determining if there is ever an electrical failure, or to confirm the actual activation of a relay. ScanVolt inputs are NOT polarity sensitive and are compatible with AC or DC voltages. Each input is rectified with a 400V bridge rectifier AND optoisolated up to 5,000 volts from your Ultra Series controller and computer.

ScanVolt expansion controllers count as 2, 4, or 6 banks on a Port A or Port B expansion buss. Each bank provides you with 8 optically isolated voltage input detectors. You can chain multiple expansion controllers in various combinations to a single port buss. Up to 256 input banks are allowed on either or both data ports, allowing you to detect the presence of 4096 voltages using a single Ultra Controller with ScanVolt expansion modules attached.

Expansion modules are designed to be chained off the I/O port buss. You can easily mix voltage detectors, switch scanners, and analog to digital converters off the same data port. No power supply is required, the I/O port provides the required power supply.

IMPORTANT NOTE:
If you are using this device in combination with the UAD1216 expansion module, the UAD1216 MUST be connected to the Ultra Controller FIRST, you can then chain multiple input expansion boards from the UAD1216.

ScanVolt is available with 16, 32, or 48 voltage input channels, and 7 varieties to help you detect many voltage ranges.

BORDER LINE VOLTAGE WARNING:
Voltages below the Minimum Detection Voltage ranges shown below may not appear to stay on. Logic circuits may read border line voltages as on or off with a lot of inconsistency. Your software may choose to sample the voltage detect circuit a few times to make sure the voltage is within the detection range. The voltages shown above are the guaranteed ON voltage values.

UD Version
The UD Version of the ScanVolt expansion module does not include optoisolator resistors. This allows you to easily build your own ScanVolt expansion. This is particularly helpful when you need to use one ScanVolt reading a wide variety of voltage ranges.

Using the UD version, you will need to install your own resistors before the ScanVolt will function, the following information will be helpful:

Inputs can be AC or DC and are NOT polarity sensitive.
Inputs are full-bridge rectified prior to optoisolation.
The on-board bridge rectifier drops the incoming voltage by 1.2V.
The on-board bridge rectifier has a 400V input rating.
The Optoisolator has a Minimum ON voltage of 1.0VDC.
The Optoisolator has a Typical ON voltage of 1.2VDC.
The Optoisolator has a Maximum ON voltage of 1.4VDC.
The Optoisolator consumes 50ma at 1.2VDC.
The Quad Optoisolator is www.digikey.com part number 160-1364-5-ND manufactured by Lite-On Inc, Part Number LTV846.

Resistors are installed in **User Resistor** locations **UR1-UR8, 1 resistor will be required for each input, resistors are not supplied with the UD version of this controller**. The table below shows some common **User Resistor** values and their associated voltage ranges.

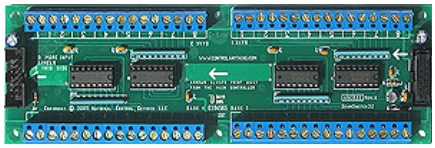
Ending Part Number	Description	UR Resistor Value	Minimum Detection Voltage	Maximum Detection Voltage
UD	User Defined: Board Comes Unpopulated with Optoisolation Resistors. Resistors MUST be installed by the user prior to use. This allows the user to specify the detection voltage for each individual channel.	User Installed	User Defined	User Defined
LV	Low Voltage Detection		3V	30V
MV1	Medium Voltage Detection Range 1		25V	50V
MV2	Medium Voltage Detection Range 2		45V	70V
HV1	High Voltage Detection Range 1		65V	90V
HV2	High Voltage Detection Range 2		85V	110V
HV3	High Voltage Detection Range 3		105V	130V

USCSxx: 16/32/48-Channel Switch Closure Input Scanner (ScanSwitch)

Ultra Series Controllers can Now Read up to 4096 Switch Closure Inputs

We developed this device specifically for use at our new design & manufacturing facility...it manages all the light switches, motion detectors, glass breakage detectors, and open window detectors with ease.

USCS16
ScanSwitch 16-Channel



USCS32
ScanSwitch 32-Channel

USCS48
ScanSwitch 48-Channel

USCSxx Introduction

Ultra ScanSwitch 16, 32, and 48 Expansion Modules allow you to detect switch closure inputs, allowing the Ultra controller to easily monitor light switches, motion detectors, door sensors, limit switches, and any other kind of sensor with a switch closure output.

ScanSwitch expansion controllers count as 2, 4, or 6 banks on a Port A or Port B expansion buss. Each bank provides you with 8 switch closure input detectors. You can chain multiple expansion controllers in various combinations to a single port buss. Up to 256 input banks are allowed on either or both data ports, for a total monitoring capability of 4096 switch closure inputs.

Expansion modules are designed to be chained off the I/O port buss. You can easily mix voltage detectors, switch scanners, and analog to digital converters off the same data port. No power supply is required, the I/O port provides the required power supply.

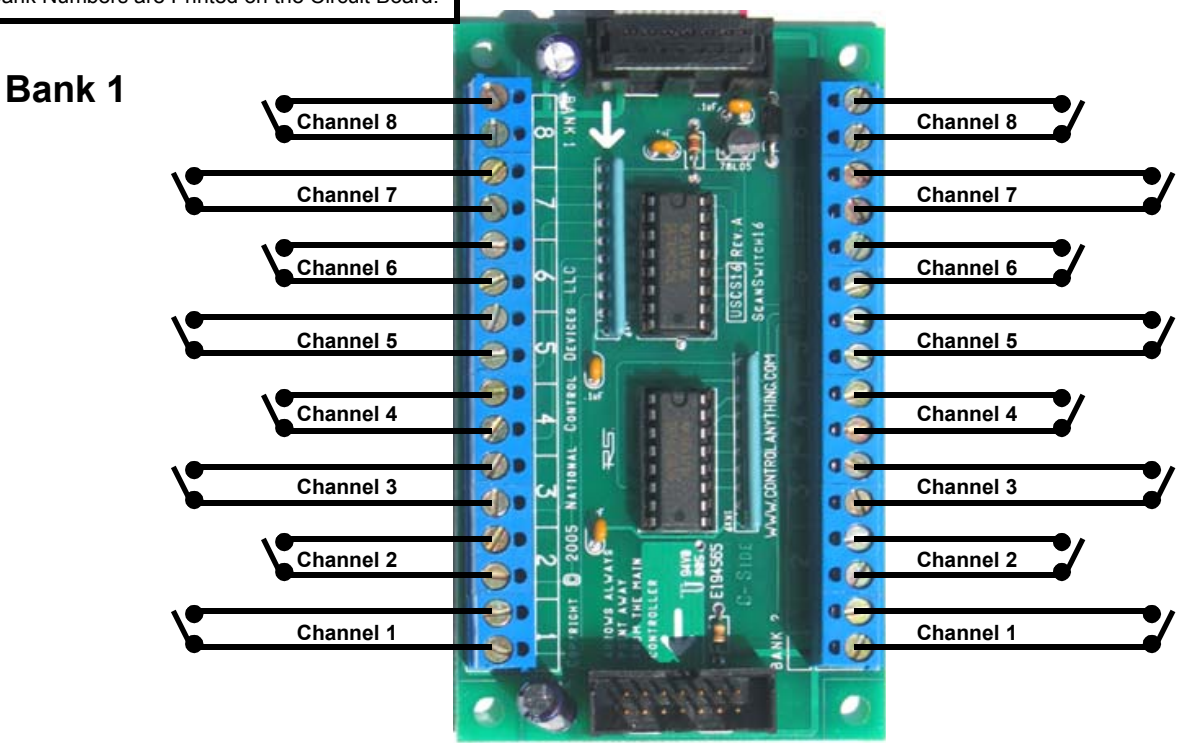
IMPORTANT NOTE:
If you are using this device in combination with the UAD1216 expansion module, the UAD1216 MUST be connected to the Ultra Controller FIRST, you can then chain multiple input expansion boards from the UAD1216.

IMPORTANT WARNING:
ScanSwitch controllers should only be used to detect the on/off status of various kinds of switches. ScanSwitch is NOT compatible with any devices that delivers a voltage to the controller (Use our ScanVolt series controller for this application). Applying any kind of voltage to the ScanSwitch inputs can cause severe damage to all attached controllers and computers. Make absolutely certain that ONLY switch closure connections are made to the ScanSwitch expansion module.

Connections are the Same for All Models, there is simply more input banks on the other versions. Bank Numbers are Printed on the Circuit Board.

This Connector Attaches to the Ultra Series Controller

White Arrows Printed on the Circuit Board Should ALWAYS Point AWAY from the Main Controller



All Inputs are 0-5VDC. Power to this Device is Derived from the Ultra Expansion Port.

This Connector Attaches to More Expansion Modules
(Up to 256 Banks of 8 per Ultra Port)

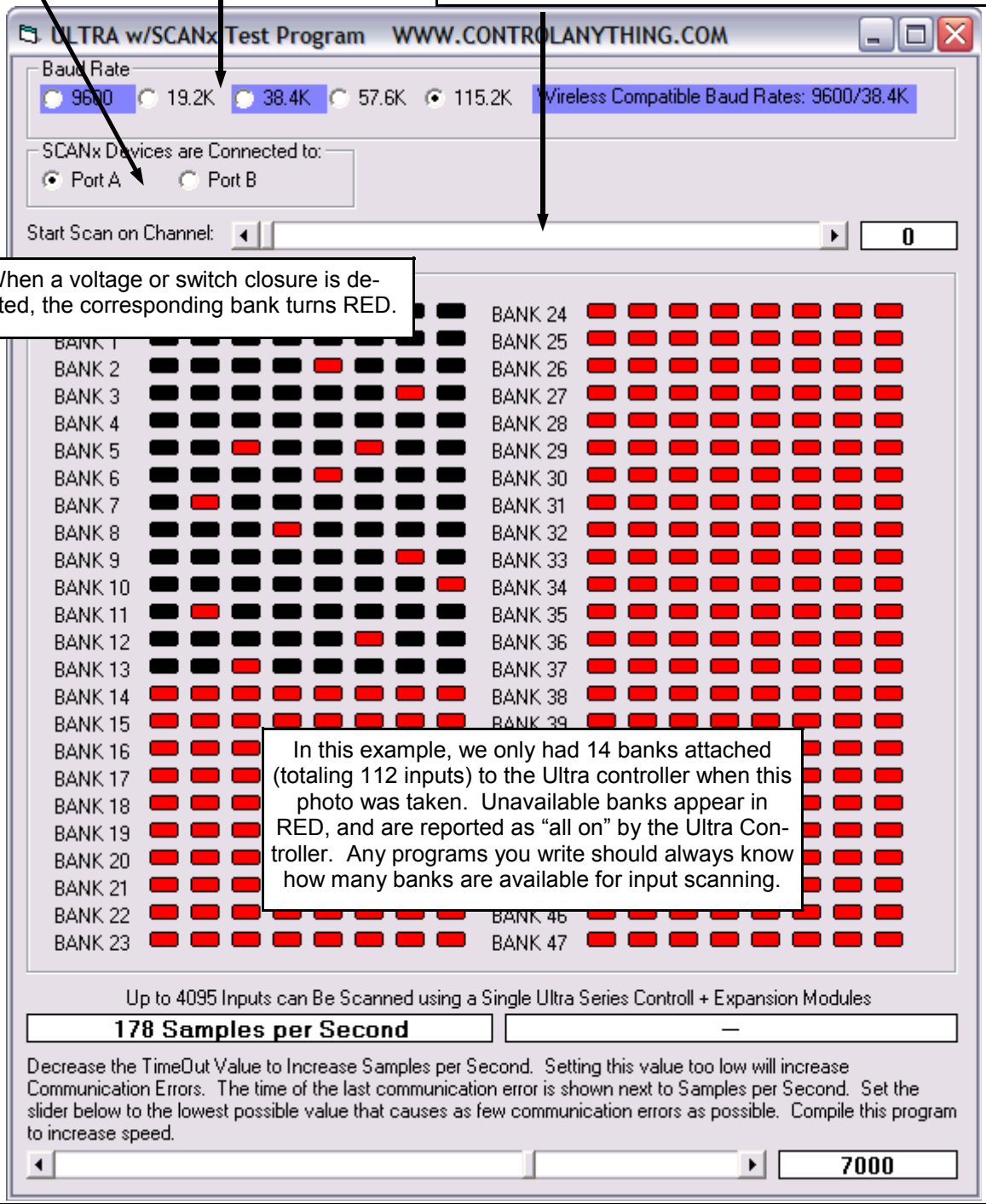
Voltage Detectors and Contact Closure Input Expansions Must be at the End of the Ultra Port Chain.

Example Software for Input Scanning Devices

Input Scanning Devices can be Connected to Port A and/or Port B. Make sure this parameter is properly set.

Select a Baud Rate

The SCANx program is capable of displaying 48 channels of inputs at one time (you can mix the ScanSwitch or the ScanVolt expansion modules in any combination). You can control the range of inputs that are scanned using this slider.



When a voltage or switch closure is detected, the corresponding bank turns RED.

In this example, we only had 14 banks attached (totaling 112 inputs) to the Ultra controller when this photo was taken. Unavailable banks appear in RED, and are reported as "all on" by the Ultra Controller. Any programs you write should always know how many banks are available for input scanning.

Program Variations in the Examples.ZIP File:

SCANx Input Scanning Single Bank (COM1-COM6 Supported)
Reads One Bank at a Time

Expansion Modules: Programming for Input Scanning Devices

Input Module Banks

The ScanSwitch and the ScanVolt expansion modules use the same command set. The logic circuits on these devices are 100% identical, the only difference is the interface circuit. You can use these commands interchangeably for the ScanVolt and the ScanSwitch. Groups of eight inputs are addressed by their "bank". The more devices you have connected to the expansion port, the more banks you have available to read from. Banks are assigned to these devices incrementally. Each time you add a bank of inputs to the expansion port, you increase the potential bank value. The lowest number bank values correspond to the banks that are closest to the input expansion connector. Bank values increase as you get further away from the expansion port on the Ultra controller. You must specify a "Bank" value when using these commands so it is important to understand the numbering sequence of banks. See diagram at right for a visual explanation of banks.

254, 20, Bank: Input Bank Read Port A

This command reads a specified input bank (connected to Port A), and is compatible with ScanVolt and ScanSwitch input expansion modules. This command requires a Bank parameter, with a valid range of 0-255, indicating which input bank you would like to read. This command returns a single byte of data, from 0-255, indicating the on/off status in binary format for all eight inputs (of the selected bank). This byte can then be bit tested to determine if individual inputs are active. See Bit Testing below.

254, 21, Bank: Input Bank Read Port B

This command reads a specified input bank (connected to Port B), and is compatible with ScanVolt and ScanSwitch input expansion modules. This command requires a Bank parameter, with a valid range of 0-255, indicating which input bank you would like to read. This command returns a single byte of data, from 0-255, indicating the on/off status in binary format for all eight inputs (of the selected bank). This byte can then be bit tested to determine if individual inputs are active. See Bit Testing below.

Bit Testing

The programming example below shows you how to "Bit Test" a byte of data to determine if individual inputs are on/off. It is at least 14 times faster to offload this task on your computer, rather than actual device, especially when you are working with large arrays of inputs. Bit Testing uses the mathematical "And" function to determine if bits are active or not. Virtually all programming languages that support math functions support the "And" function, as it is an essential function for most logical operations.

Note that VALUE below is the value reported back to your computer from the Input Expansion controller. You specified the Bank you wanted to read, now the controller is reporting back the VALUE generated by your inputs on the selected bank.

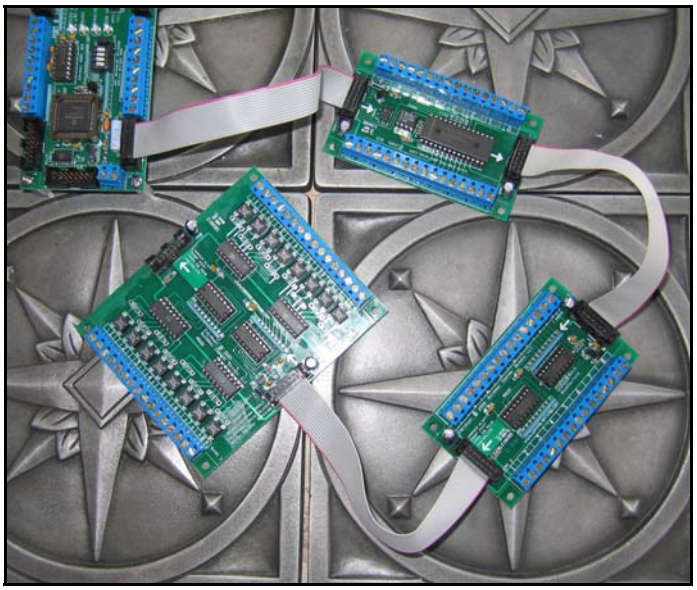
```
VALUE = GetI                    'Get a Data Byte from the Controller
If (VALUE AND 1) = 1 then       'If Input Bit 0 is ON Then
                                'Execute these lines of code
Else                            'Otherwise the Bit is OFF
                                'Execute these lines of code
endif
```

Here is an example that tests all input bits:

```
VALUE = GetI                    'Get a Data Byte from the Controller

If (VALUE AND 1) = 1 then       'If Bit 0 (Input 0) is in the ON State
If (VALUE AND 2) = 2 then       'If Bit 1 (Input 1) is in the ON State
If (VALUE AND 4) = 4 then       'If Bit 2 (Input 2) is in the ON State
If (VALUE AND 8) = 8 then       'If Bit 3 (Input 3) is in the ON State
If (VALUE AND 16) = 16 then     'If Bit 4 (Input 4) is in the ON State
If (VALUE AND 32) = 32 then     'If Bit 5 (Input 5) is in the ON State
If (VALUE AND 64) = 64 then     'If Bit 6 (Input 6) is in the ON State
If (VALUE AND 128) = 128 then   'If Bit 7 (Input 7) is in the ON State

If (VALUE AND 1) = 0 then       'If Bit 0 (Input 0) is in the OFF State
If (VALUE AND 2) = 0 then       'If Bit 1 (Input 1) is in the OFF State
If (VALUE AND 4) = 0 then       'If Bit 2 (Input 2) is in the OFF State
If (VALUE AND 8) = 0 then       'If Bit 3 (Input 3) is in the OFF State
If (VALUE AND 16) = 0 then      'If Bit 4 (Input 4) is in the OFF State
If (VALUE AND 32) = 0 then      'If Bit 5 (Input 5) is in the OFF State
If (VALUE AND 64) = 0 then      'If Bit 6 (Input 6) is in the OFF State
If (VALUE AND 128) = 0 then     'If Bit 7 (Input 7) is in the OFF State
```



The photo above shows how expansion modules can be chained to the Ultra Port, adding features to the main controller. Expansion modules simply connect end to end, and are designed to work together. The UAD1216 is shown first in the chain. For proper operation, it is important to keep input scanning devices, such as the ScanSwitch and ScanVolt at the end of the chain. These devices modify data that passes along the data port. The UAD1216 is a pass-through device, it does not modify any data on the port. Pass-through devices should be connected first, then data port modification devices.

Input Bank Read Port A

```
MSComm1.Output = Chr$(254) 'Command Mode
MSComm1.Output = Chr$(20)  'Read Input on Port A
MSComm1.Output = Chr$(1)   'Read Bank 1
VALUE = GetI               'Get Value from Controller
If (VALUE AND 1) = 1 then     'If Bit 0 (Input 0) is in the ON State
If (VALUE AND 2) = 2 then     'If Bit 1 (Input 1) is in the ON State
If (VALUE AND 4) = 4 then     'If Bit 2 (Input 2) is in the ON State
If (VALUE AND 8) = 8 then     'If Bit 3 (Input 3) is in the ON State
If (VALUE AND 16) = 16 then   'If Bit 4 (Input 4) is in the ON State
If (VALUE AND 32) = 32 then   'If Bit 5 (Input 5) is in the ON State
If (VALUE AND 64) = 64 then   'If Bit 6 (Input 6) is in the ON State
If (VALUE AND 128) = 128 then 'If Bit 7 (Input 7) is in the ON State
```

Input Bank Read Port B

```
MSComm1.Output = Chr$(254) 'Command Mode
MSComm1.Output = Chr$(21)  'Read Input on Port B
MSComm1.Output = Chr$(0)   'Read Bank 0
VALUE = GetI               'Get Value from Controller
If (VALUE AND 1) = 1 then     'If Bit 0 (Input 0) is in the ON State
If (VALUE AND 2) = 2 then     'If Bit 1 (Input 1) is in the ON State
If (VALUE AND 4) = 4 then     'If Bit 2 (Input 2) is in the ON State
If (VALUE AND 8) = 8 then     'If Bit 3 (Input 3) is in the ON State
If (VALUE AND 16) = 16 then   'If Bit 4 (Input 4) is in the ON State
If (VALUE AND 32) = 32 then   'If Bit 5 (Input 5) is in the ON State
If (VALUE AND 64) = 64 then   'If Bit 6 (Input 6) is in the ON State
If (VALUE AND 128) = 128 then 'If Bit 7 (Input 7) is in the ON State
```

Reading Data from the Controller

The ***GetI*** Function Below waits for data from the controller. If data is not received before the timeout period, the function will exit, containing no value.

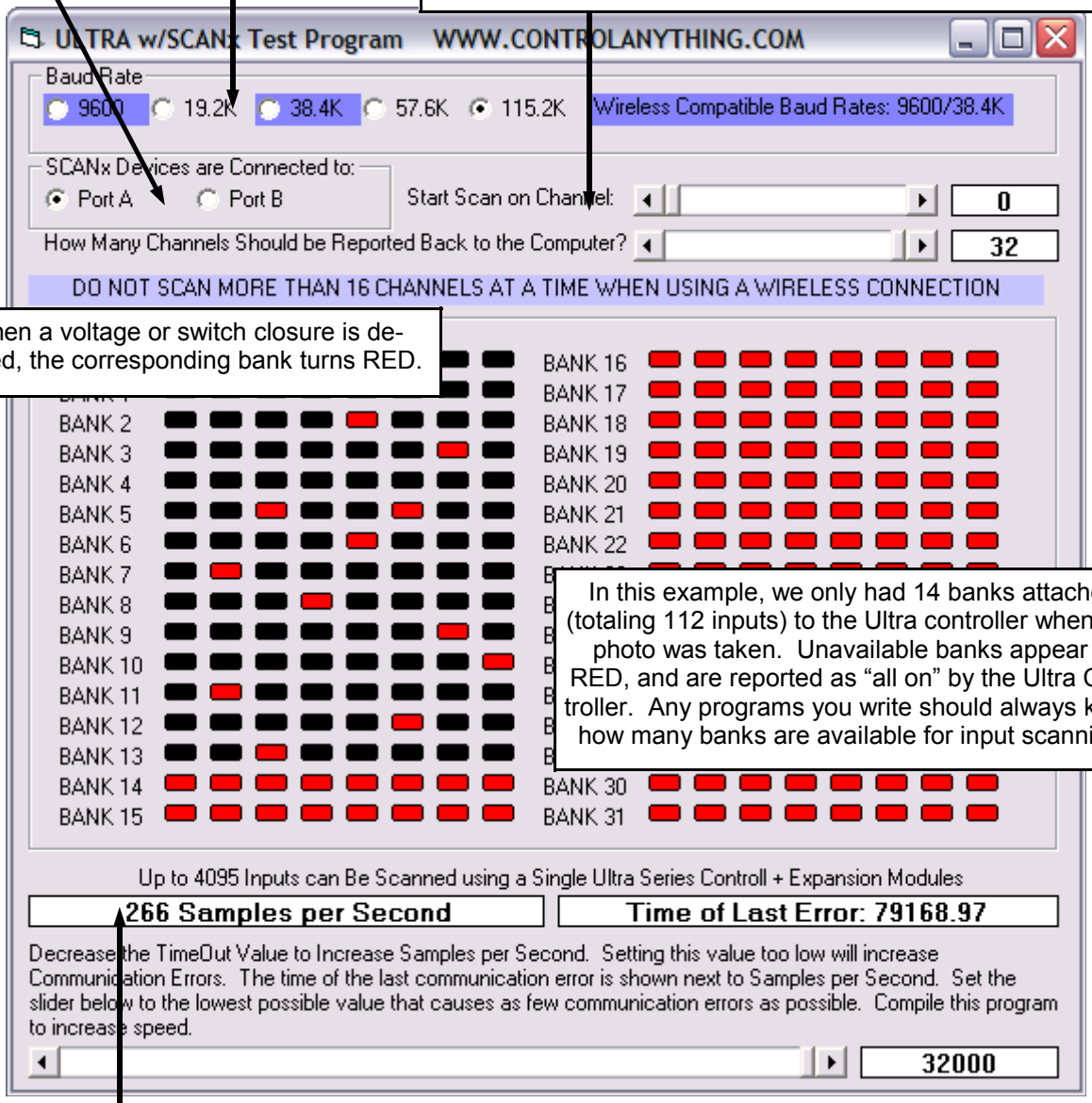
```
Public Function GetI()
    T = 0
    Do
        T = T + 1
        If T > 10000 then Exit Function
        DoEvents
    Loop Until MSComm1.InBufferCount > 0
    GetI = Asc(MSComm1.Input)
End Function
```

Example Software for Input Scanning Devices

Input Scanning Devices can be Connected to Port A and/or Port B. Make sure this parameter is properly set.

Select a Baud Rate

"Input Spanning" allows you to read a group of inputs at one time (up to 32). Spanning is significantly faster when reading large arrays of inputs because there is less communication to the Ultra controller. One request is made for up to 32 channels of data rather than 32 requests for 32 channels. These sliders are used to set the range of inputs you would like to "Span", beginning with the starting bank. The "Span" is set by the second slider, it controls how many banks will be read. Each bank will report back one byte of data. You can span consecutive inputs beginning with any bank.



When a voltage or switch closure is detected, the corresponding bank turns RED.

In this example, we only had 14 banks attached (totaling 112 inputs) to the Ultra controller when this photo was taken. Unavailable banks appear in RED, and are reported as "all on" by the Ultra Controller. Any programs you write should always know how many banks are available for input scanning.

Note the significant increase in samples per second when requesting inputs in groups rather than individually.

Program Variations in the Examples.ZIP File:

SCANx Input Scanning Multi-Bank (COM1-COM6 Supported)

Reads a Span of Banks at a One Time and Sends Multiple Bytes of Data Back to your Computer

Expansion Modules: Programming for Input Scanning Devices

Input Module Banks

The ScanSwitch and the ScanVolt expansion modules use the same command set. The logic circuits on these devices are 100% identical, the only difference is the interface circuit. You can use these commands interchangeably for the ScanVolt and the ScanSwitch. Groups of eight inputs are addressed by their "bank". The more devices you have connected to the expansion port, the more banks you have available to read from. Banks are assigned to these devices incrementally. Each time you add a bank of inputs to the expansion port, you increase the potential bank value. The lowest number bank values correspond to the banks that are closest to the input expansion connector. Bank values increase as you get further away from the expansion port on the Ultra controller. You must specify a "Bank" value when using these commands so it is important to understand the numbering sequence of banks. See diagram at right for a visual explanation of banks.

254, 22, Start, Span: Input Banks Multi-Read Port A

This command reads a range of inputs (or "Span") on an input module connected to Port A. This command requires two parameters. The first parameter is the Starting bank you would like the controller to read. The Span parameter is used to indicate how many banks you would like the controller to return (up to 32). The Start + Span Values should never exceed 255. The Span value should not exceed 32 on a direct RS-232 connection and 16 on a wireless connection.

Example Usage:

254, 22, 10, 5	Returns 5 Bytes, Staring with Bank 10
254, 22, 0, 32	Returns 32 Bytes, Starting with Bank 0
254, 22, 245, 10	Returns 10 Bytes, Starting with Bank 245

254, 23, Start, Span: Input Banks Multi-Read Port B

This command reads a range of inputs (or "Span") on an input module connected to Port B. This command requires two parameters. The first parameter is the Starting bank you would like the controller to read. The Span parameter is used to indicate how many banks you would like the controller to return (up to 32). The Start + Span Values should never exceed 255. The Span value should not exceed 32 on a direct RS-232 connection and 16 on a wireless connection.

Bit Testing

The programming example below shows you how to "Bit Test" a byte of data to determine if individual inputs are on/off. It is at least 14 times faster to offload this task on your computer, rather than actual device, especially when you are working with large arrays of inputs. Bit Testing uses the mathematical "And" function to determine if bits are active or not. Virtually all programming languages that support math functions support the "And" function, as it is an essential function for most logical operations.

Note that VALUE below is the value reported back to your computer from the Input Expansion controller. You specified the Bank you wanted to read, now the controller is reporting back the VALUE generated by your inputs on the selected bank.

```
VALUE = GetI                    'Get a Data Byte from the Controller
If (VALUE AND 1) = 1 then       'If Input Bit 0 is ON Then
                                'Execute these lines of code
Else                            'Otherwise the Bit is OFF
                                'Execute these lines of code
endif
```

Here is an example that tests all input bits:

```
VALUE = GetI                    'Get a Data Byte from the Controller

If (VALUE AND 1) = 1 then       'If Bit 0 (Input 0) is in the ON State
If (VALUE AND 2) = 2 then       'If Bit 1 (Input 1) is in the ON State
If (VALUE AND 4) = 4 then       'If Bit 2 (Input 2) is in the ON State
If (VALUE AND 8) = 8 then       'If Bit 3 (Input 3) is in the ON State
If (VALUE AND 16) = 16 then     'If Bit 4 (Input 4) is in the ON State
If (VALUE AND 32) = 32 then     'If Bit 5 (Input 5) is in the ON State
If (VALUE AND 64) = 64 then     'If Bit 6 (Input 6) is in the ON State
If (VALUE AND 128) = 128 then   'If Bit 7 (Input 7) is in the ON State

If (VALUE AND 1) = 0 then       'If Bit 0 (Input 0) is in the OFF State
If (VALUE AND 2) = 0 then       'If Bit 1 (Input 1) is in the OFF State
If (VALUE AND 4) = 0 then       'If Bit 2 (Input 2) is in the OFF State
If (VALUE AND 8) = 0 then       'If Bit 3 (Input 3) is in the OFF State
If (VALUE AND 16) = 0 then      'If Bit 4 (Input 4) is in the OFF State
If (VALUE AND 32) = 0 then      'If Bit 5 (Input 5) is in the OFF State
If (VALUE AND 64) = 0 then      'If Bit 6 (Input 6) is in the OFF State
If (VALUE AND 128) = 0 then     'If Bit 7 (Input 7) is in the OFF State
```

Input Banks Multi-Read Port A

```
MSComm1.Output = Chr$(254) 'Command Mode
MSComm1.Output = Chr$(22)  'Read Multi-Bank Input on Port A
MSComm1.Output = Chr$(0)   'Read Start at Bank 0 (Start Bank)
MSComm1.Output = Chr$(10)  'Read 10 Banks of Inputs (Span)
For Bank = 0 to 9          'Count Through Input Banks
  VALUE = GetI             'Get Value from Controller
  Debug.Print Bank;Value   'Show Bank Data in Debug Window
Next Ch
```

Input Banks Multi-Read Port B

```
MSComm1.Output = Chr$(254) 'Command Mode
MSComm1.Output = Chr$(23)  'Read Multi-Bank Input on Port A
MSComm1.Output = Chr$(5)   'Read Start at Bank 5 (Start Bank)
MSComm1.Output = Chr$(15)  'Read 15 Banks of Inputs (Span)
For Bank = 5 to 20         'Count Through Input Banks
  VALUE = GetI             'Get Value from Controller
  Debug.Print Bank;Value   'Show Bank Data in Debug Window
Next Ch
```

Reading Data from the Controller

The ***GetI*** Function Below waits for data from the controller. If data is not received before the timeout period, the function will exit, containing no value.

```
Public Function GetI()
    T = 0
    Do
        T = T + 1
        If T > 10000 then Exit Function
        DoEvents
    Loop Until MSComm1.InBufferCount > 0
    GetI = Asc(MSComm1.Input)
End Function
```