Exhibit Control
ENGINEERING
102 Waterview Circle
Forest, VA 24551
(434) 385-4144
EMAIL: EXHIBIT_CONTROL@YAHOO.COM
WEB: WWW.EXHIBIT-CONTROL.NET

# *USB Commando*

**PN: ece-I-15-002, REv 3**

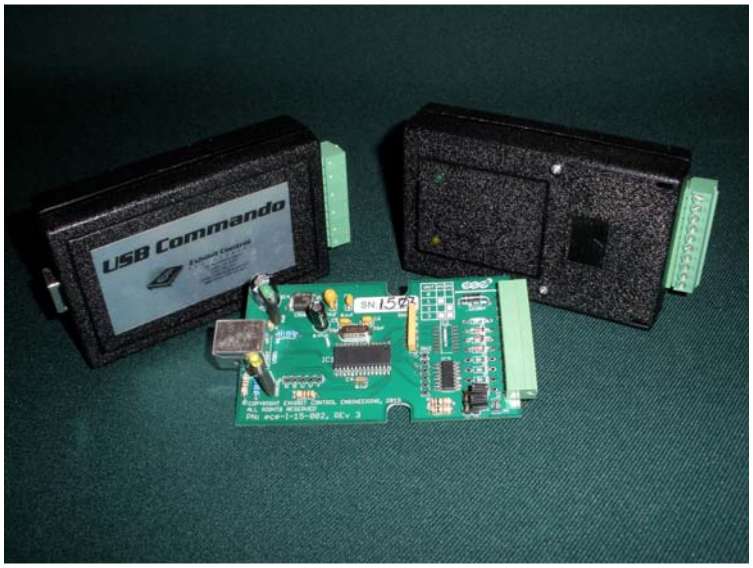


**Figure 1**

**Overview:**

This product is for those instances where you need to provide a PC program with external stimuli, either contact closure (switch input) or a voltage between 0-5vdc, as an input from outside the PC. Or, if you want the PC to control a bi-state device (on/off) such as a LED, relay, etc. Generally, the USB Commando comes with eight IOs. They could all be inputs for eight contact closures or they could be eight outputs (open collector) that can sink up to 500ma each to ground. They could also be four digital inputs and four digital outputs. Finally, four of the I/Os can be alternatively configured for analog inputs of 0-5vdc. These could be used for inputs from sliders, potentiometers, joysticks, and sensors of all types (voltage, current, temperature, pressure). The eighth I/O could be configured to provide 5vdc to external potentiometers for the analog inputs, making sure the input is never subjected to a voltage higher than 5 volts. As a matter of interest, the USB Commando does not use parasite power off of the USB server. It requires a 9-12 volt power supply making it self-powered. So here are some possible configurations we could customize your USB Commando to:

A. Eight digital inputs which could be contact closures switches, motion detectors, TTL signals, to name a few.

B. Eight digital outputs that can sink up to 500ma of current at 30vdc. This means you can control LEDs, 12-volt incandescent lights or relays. The relays could then control 120vdc power controllers and other higher current/voltage devices.

C. Four digital inputs and four digital outputs. This would be useful for four illuminated buttons on a reader rail or in a kiosk where the illumination could be controlled by the PC for feedback to the visitor.

D. Four analog inputs with the 5vdc source supplied by the Commando (I/O #8 or the tenth pin on the Phoenix connector). This could handle two analog joysticks (two analog inputs each) with three "fire" switch inputs.

E. Finally, we could probably devise any combination of inputs to outputs. Some of the possibilities include :
   a. 1 digital input, 7 digital outputs
   b. 2 digital inputs, 6 digital outputs
   c. 6 digital inputs, 2 digital outputs
   d. 7 digital inputs, 1 digital outputs
   e. 2 analog input, 2 digital inputs and 4 digital outputs.

There are two LEDs on the Commando: a green and a yellow. The green indicates when the device is powered and the yellow illuminates when the device is attached to a USB server.

## Technical Details:

Regarding identification of the USB Commando on the USB bus, its vendor ID will always be 3790 (0x0ECE). The product ID may change from device to device, depending if they are to be on the same computer. But, usually, for a single USB Commando in a system the product ID will be 257 (0X0101) and will increment up for additional units. It is hoped that the unit ID, which is configured by shorting blocks on the PCB, will also help in identifying several devices on the same computer. Unit #0 will have no shorting blocks on the jumpers. Unit #1 will have a shorting block on Jumper #2; Unit #2 will have a shorting block on Jumper #3; and, Unit #3 will have shorting blocks on both Jumpers.

For clarification: when referring to Bytes in a communication packet, the first byte is designated as Byte #1. However, when referring to a particular Bit in a byte, the first Bit is designated zero (Bit #0). This is kind of a programmer's bit banging reference that we have adopted for this document.

The communication between the Commando is on the USB bus and has two different packet sizes. When the Commando sends out a packet, it is 12 bytes long. However, the Commando only expects to get packets of 8-byte lengths back. Packets from the Commando will start with its 4-bit Unit address (0 – 3) based on the shorting blocks on the PCB, all other bits being zero (0). The tenth byte provides the information on the switch inputs. If switch one is closed, Bit #0 of the tenth byte will be 1. If switch two is closed, Bit #1 of the tenth byte will be 1. So if the Commando had its first four inputs as digital, we could expect the tenth byte to be a number between zero (0) and 15 (0x00 and 0x0F). If the Commando were configured for 8 digital inputs, we can expect the 10th Byte to be a number between zero (0) and 255 (0x00 and 0xFF). In all cases, the other ten Bytes (Bytes 2 through 9 and Byte 11 and Byte 12) would each be zero (0x00).

If the Commando were configured for analog inputs (a maximum of four), our 12-byte packet would look very different. Byte #1 would still be the 4-bit address byte. But the next two bytes would provide the value of the analog input for input #1. Byte #2 would provide the most significant byte value and the 3rd Byte would provide the least significant value. The Commando uses a 10-bit analog to digital converter, which means the largest decimal number we will get is 1,023 (or hex 0X03FF) for a 5vdc input. Similarly, Byte #4 and #5 will provide the value for analog value of input #2. The most significant byte is always first; the least significant second. So Byte #6 and #7 are the value for the third analog channel and Byte #8 and #9 for the fourth and last analog channel. If the other four IOs were used for button inputs, they would be in Byte #10 as describe in the paragraph above.

Regarding the ADC (analog) operation, when the Commando sends you a value for any particular channel, it is not a single sample of the signal. It actually samples the voltage 10 times very quickly. It then subtracts the highest value and the lowest value and then averages the remaining eight samples, which it sends as its evaluation of the voltage. It does these ten samples on every channel once every 0.25 of a second. Which is why, in any configuration, you can expect the Commando to blast you a 12-byte message update four times a second. And, it will never stop.

Regarding the input analog signal - it needs to be between 0 and 5vdc. The Commando cannot handle an input voltage greater than 5 volts. If your signal is greater you must scale it back so the max voltage is less than 5. This can easily be accomplished using a voltage divider network. Call us for help. The bottom line is this: if you apply a voltage greater than 5vdc, you will fry the input. One way you can ensure the max voltage is 5 volts or less is to take it from the Commando. This process will reduce the total IOs to seven, so we can use the last pin on the Phoenix connector for the 5 volts out. So, you could use the Commando's 5volt power on the two pots of a joy stick and the Commando could tell you joystick position four times every second. For the groundside of the potentiometer, you will always have to double up on the power supply ground position on the Phoenix connector. It should be evident from this discussion that you cannot use the Commando's input power supply for analog signals because it is greater than 5 volts (anywhere from 9-12vdc). By-the-by, this 5-volt limit is not true for digital inputs to the Commando. Digital inputs are diode protected, up to 30vdc.

That is it for communications from the Commando, now to discuss sending commands to it.  To tell the USB Commando to turn on or off a digital output, your PC program needs to send out an eight-byte packet to the Commando.  The first byte will always be zero (0x00).  The second byte is the unit address that you want to affect: the 4-bit address (0 – 3).  If you don't send out the right address, the Commando will ignore the communication.  The next byte, Byte #3, will use the same bit structure as described above for digital inputs.  If you want channel #1 output to sink current (turn on), make Bit #0 equal to one on Byte #3.  Want channel #2 on, make Bit #1 equal to one.  So you will be sending a byte with values between 0 and 255 in Byte #3.  All of the other bytes, three through eight, will be equal to zero (0x00).  Unlike the Commando, you only have to send the command once whenever you want to change the outputs' configuration; you don't have to repeatedly send out the command - unless you want to.  In any case, the outputs will hold until another command comes in or the Commando is turned off (which is different than being disconnected from the USB bus when the Commando will hold its current output states).

As was mentioned earlier, these outputs are open collector outputs.  They can handle 500ma of current from voltages as high as 30vdc.  So they can directly control LEDs and any other bi-state device that needs less than 500ma at 30vdc.  If you need to control larger currents or voltages, the Commando's outputs could control a relay coil for relays that can handle more severe current and voltage loads.

See the attached drawing for a typical application.

## Planned Enhancements:

In the original circuit design of the USB Commando, we intentionally ensured I/O #7 was on an I/O that could be pulse width modulated (PWM).  It is our intention at some time in the future to develop firmware to implement this feature on the Commando.  Right now that effort is on the back burner, but if you have an immediate requirement that could use a PWM output sinking 500ma to provide a pseudo-analog output, dim an LED or control a motor, call us to discuss.

We also realize that many media developers use programs that cannot deal directly at the hardware level.  By the end of 3QTR2015, we hope to have an interface applet that will enable those types of programs to communicate with the Commando via TCPIP.  Once again if you have an interest in this feature sooner, please call.

## ECE's USB Commando Input Simulator:

For those of you that might want to try before buy, we have a little demonstration kit that we could loan you.  The configuration of the Commando is the same as was provided as a typical application in the attached drawing.  However, we provide the input stimuli device so you don't have to breadboard anything.  It also comes with a little VB6.0 program we developed as a demonstration that provides a Window's PC user interface to communicate with the Commando.  See Figure 2 for the simulator plugged into a USB Commando and Figure 3 for a screen shot of the GUI.

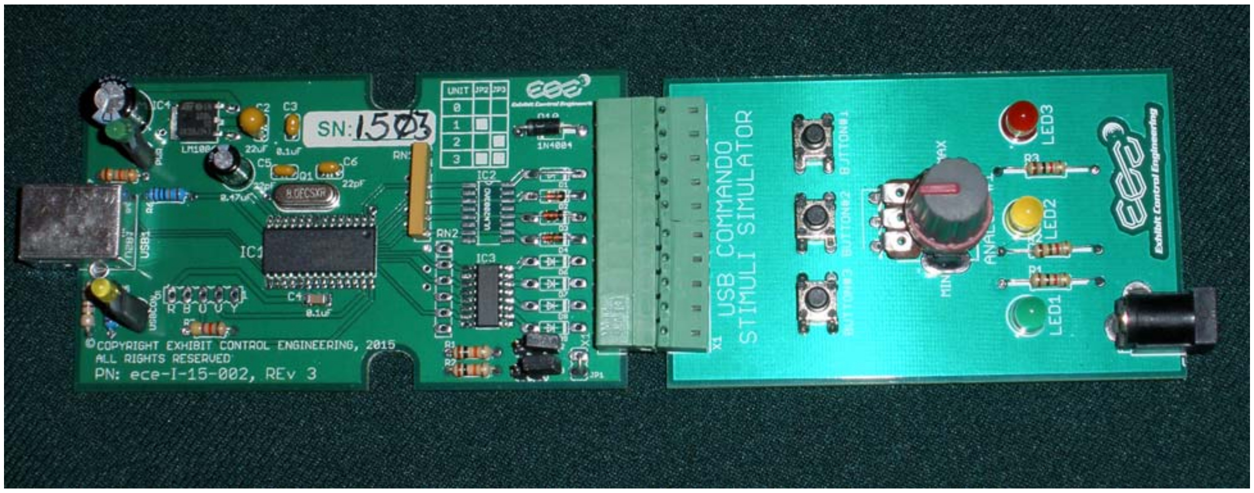
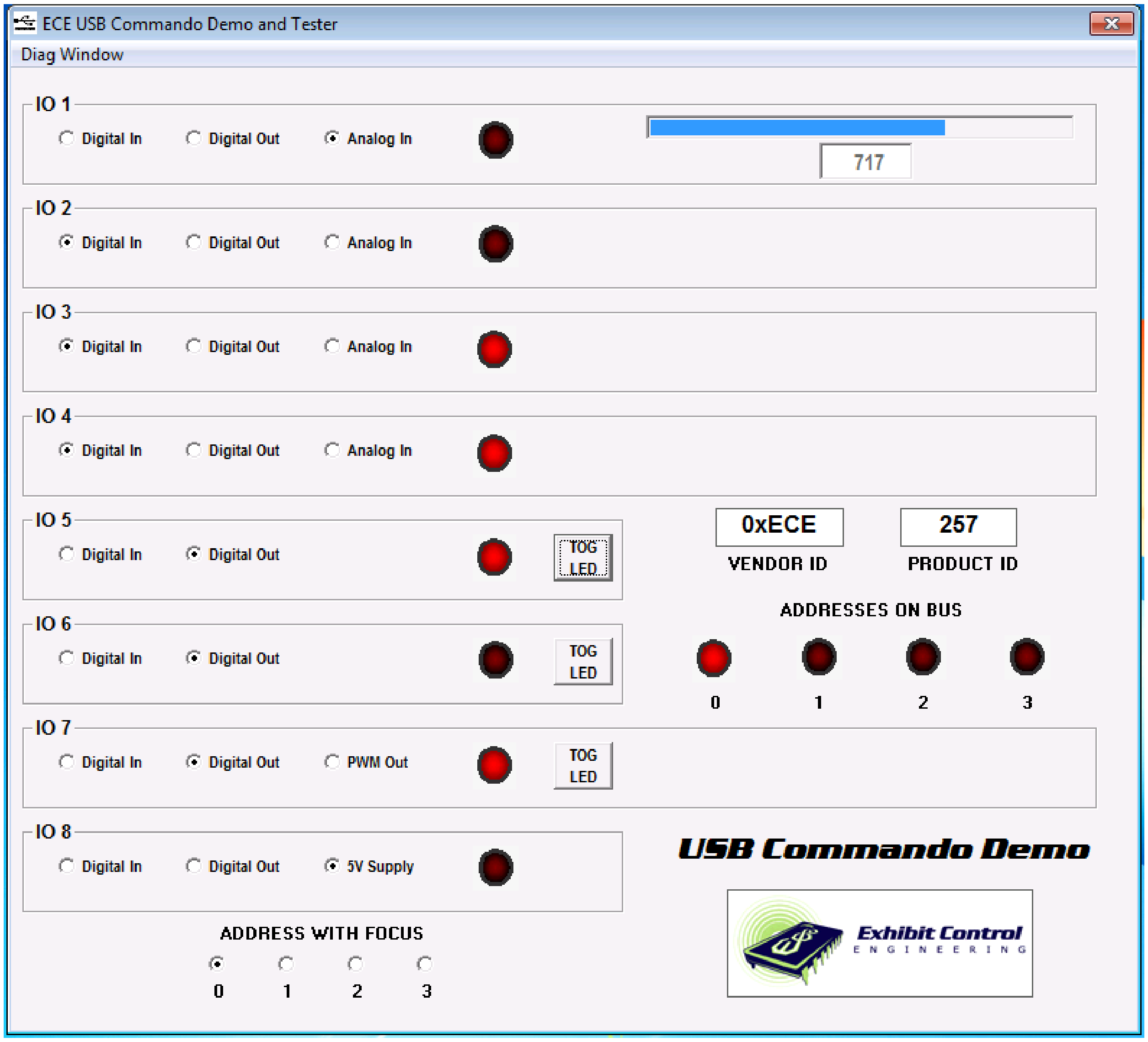

SN: 1503
USB COMMANDO STIMULI SIMULATOR
LED1
LED2
LED3
©COPYRIGHT EXHIBIT CONTROL ENGINEERING, 2015
ALL RIGHTS RESERVED
PN: ece-I-15-002, REv 3
Exhibit Control Engineering

**Figure 2**



IO 1: ( ) Digital In ( ) Digital Out [✓ Analog In] — 717

IO 2: [✓ Digital In] ( ) Digital Out ( ) Analog In

IO 3: [✓ Digital In] ( ) Digital Out ( ) Analog In

IO 4: [✓ Digital In] ( ) Digital Out ( ) Analog In

IO 5: ( ) Digital In [✓ Digital Out] — TOG LED

IO 6: ( ) Digital In [✓ Digital Out] — TOG LED

IO 7: ( ) Digital In [✓ Digital Out] ( ) PWM Out — TOG LED

IO 8: ( ) Digital In ( ) Digital Out [✓ 5V Supply]

VENDOR ID: 0xECE

PRODUCT ID: 257

ADDRESSES ON BUS: 0 1 2 3

ADDRESS WITH FOCUS: [✓ 0] 1 2 3

Figure 3

# USB Commando

CONFIGURATION:
I/O #1 – ANALOG INPUT
I/O #2 – DIGITAL INPUT
I/O #3 – DIGITAL INPUT
I/O #4 – DIGITAL INPUT
I/O #5 – DIGITAL OUTPUT
I/O #6 – DIGITAL OUTPUT
I/O #7 – DIGITAL OUTPUT
I/O #8 – 5VDC

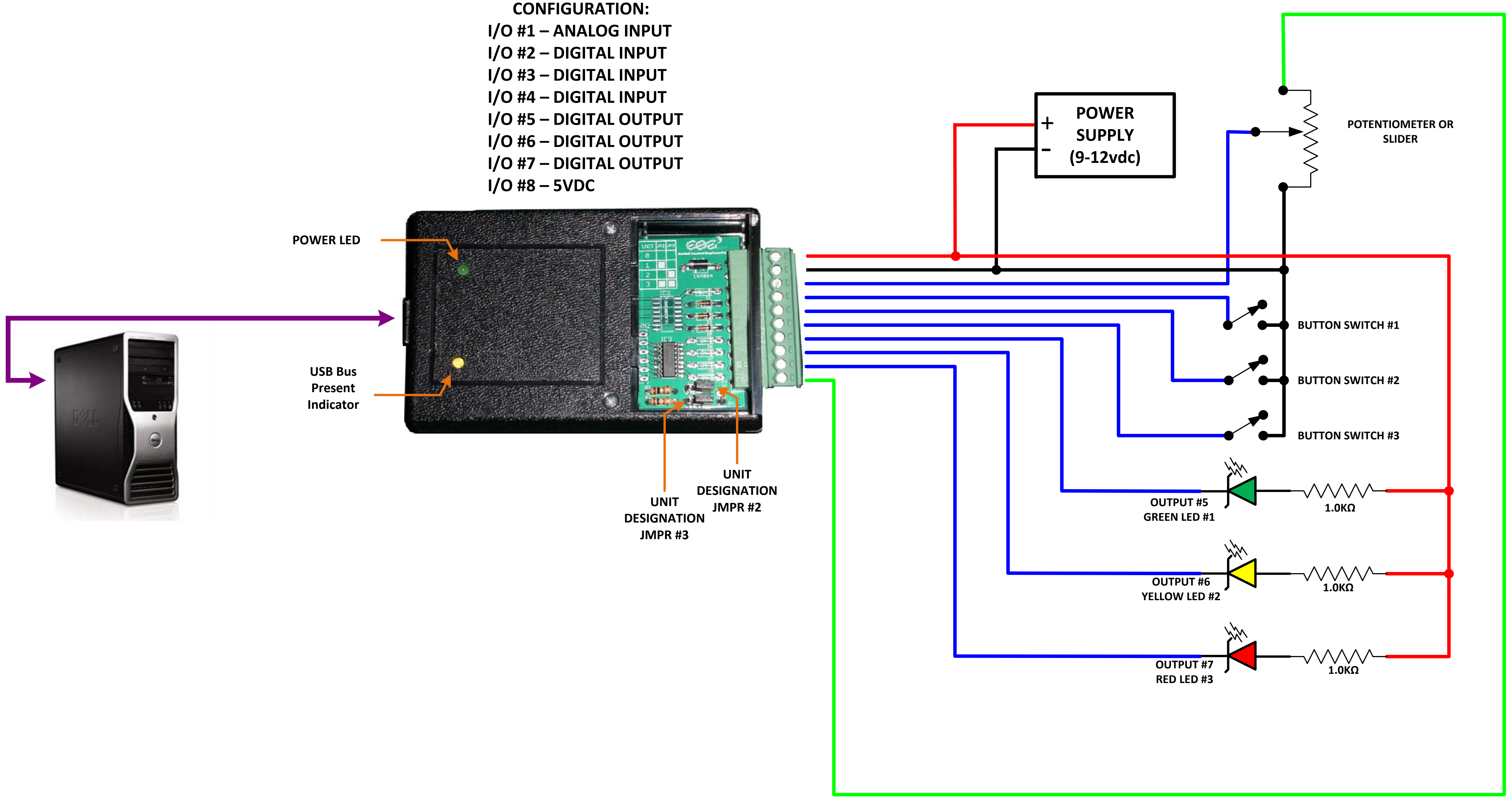


Exhibit Control
E N G I N E E R I N G
102 Waterview Circle, Forest, VA 24551
434.385.4144 (tel)
email : exhibit_control@yahoo.com
WEB Site : exhibit-control.net

LEGEND:

| | |
|---|---|
| 🟥 | B+ |
| ⬛ | GND |
| 🟦 | LOW VOLTAGE |
| 🟪 | USB Cable |
| 🟩 | 5vdc |

| REV | DATE | DESCRIPTION | APPROVED |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Exhibit Control
E N G I N E E R I N G

CLIENT:

PROJECT
TYPICAL APPLICATION

ITEM
USB Commando

| DATE | DRAWN | CHECKED |
|---|---|---|
| 4/22/2015 | FGP | |

| JOB NO. | SHEET |
|---|---|
| INTERNAL | 1.0 |

FILE NAME
USB Commando Typical Application.vsd