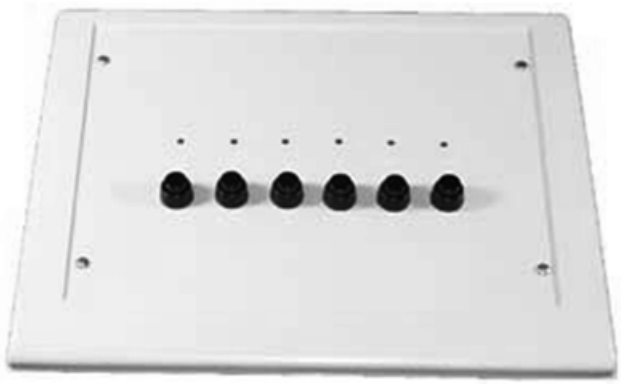
BBOX Button Box



BBOX Overview

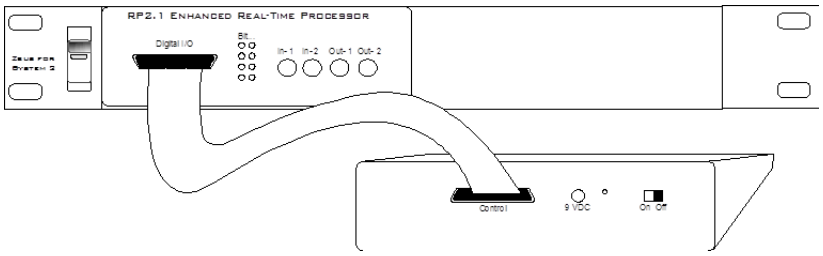
The button box is a complete subject response interface. It is an excellent system for psychoacoustics, including n-alternative forced choice, GO NO GO, Bekesy style presentation, and modified method of limits experiments. The button box provides accurate reliable performance. All inputs are debounced in the button box and a built-in rechargeable lithium-ion battery provides power for up to 24 hours of continuous use per charge.

The standard button box configuration includes six buttons and six high intensity LEDs. However, the button and LED organization can be configured to user specification. The button box can have up to eight buttons and 32 LEDs. The button box design allows experimenters a great deal of flexibility to control feedback based on subject response, reinforcing behavior for correct and incorrect choices.

The button box can be controlled from an RP2.1 or RV8 processor with button response acquisition and LED control through the digital input/output port of these modules. Data can be latched and then read from specialized RPvdsEx circuits using ActiveX and Matlab, or other programming languages. RPvdsEx circuits designed for button box control can be used with all TDT software.

Connecting the Button Box to the RP2.1 or RV8

The button box is controlled using the RP2.1 or RV8 processor. The button box connects from the DB25 connector (Control) directly to the digital input/output port on the RP2.1 or RV8 with the supplied ribbon cable. The button box is configured at the factory for the RP2.1. It can be configured for the RV8 by installing a jumper pin (Jumper for RV8) on the back of the button box.



RP2.1 to BBOX Connection

Power Requirements

The button box is supplied with a 3.3 Volt lithium-ion battery pack. This high current battery should provide up to 24 hours of continuous use per charge. The lithium-ion battery charges in under three hours with the supplied 9 Volt battery charger. The ON/OFF switch, the power connection for the battery charger, and a power indicator light are found on the back of the button box. The Power/(Low Bat) LED lights when the button box is on and flashes if the battery is low.

Important! To operate any features of the button box the power must be turned on and the device must be connected to an RP2.1 or RV8 that is powered on and connected to a PC.

Caution !

A low battery may give erroneous results. If the battery is low, the battery charger can be connected to the device. This will charge the battery and power the box at the same time.

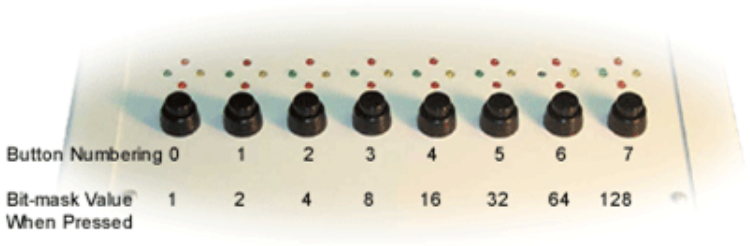
BBox Control

LEDs can be controlled and button presses can be acquired by including the necessary circuit segments in the RPvdsEx circuit that will be run on the controlling device. The button box can also be controlled using ActiveX and Matlab, or any programming language that supports ActiveX. Before designing or debugging circuits for the button box, ensure that the button box is connected to the RP2.1 or RV8 that will be used for control and that the button box power is turned on. The buttons will only operate when the button box is powered.

The remaining button box help topics provide the necessary information for basic button box control, including circuits that acquire button responses and test for correct or incorrect responses to button presses. The information provided assumes some knowledge of RPvdsEx and possibly ActiveX. Users with custom built button boxes should modify circuits based on the configuration of the buttons.

Acquiring BBox Button Presses

The most efficient way to acquire button presses is with the WordIn component in RPvdsEx. The WordIn checks all the digital input lines and returns a 16-bit value from the digital line addressed. Input values are generated as a bit-mask that determines which buttons were pressed. Users can also record the inputs from the individual digital I/O lines. The RPvdsEx examples in this topic use the WordIn method.

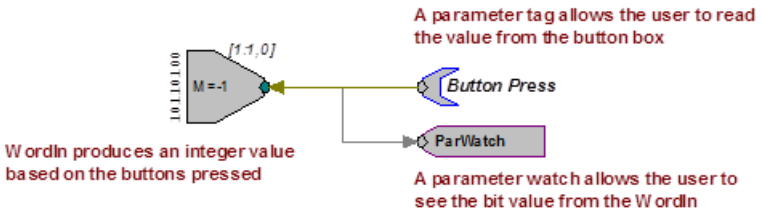


BBox Organization of Buttons

Note: The button box power supply must be turned on for the buttons to operate.

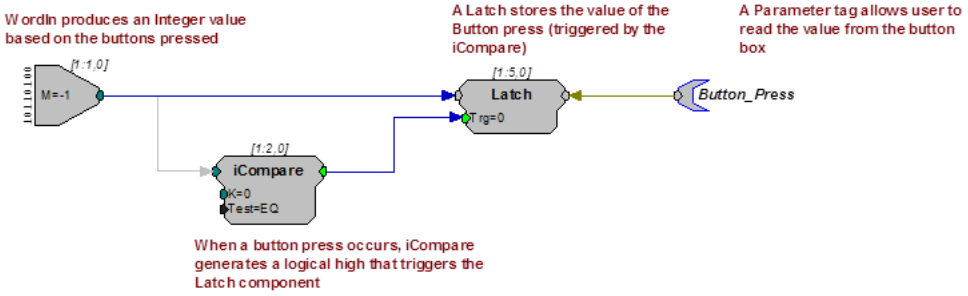
Many of the circuits shown below, as well as some MATLAB examples for use with ActiveX controls, are included with RPvdsEx (RPvdsEX | Examples | ButtonBox).

A simple circuit for acquiring button presses...



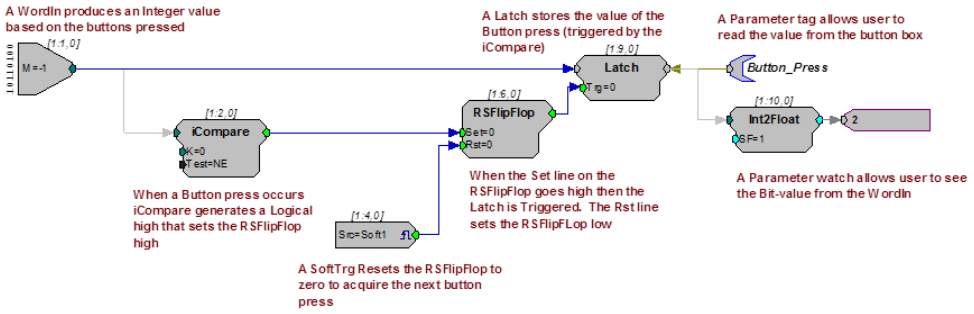
In this example, the user would continuously poll the component, from a program that acquired the value from the ButtonPress parameter, to determine which buttons are pressed. A simple circuit like this may be required if the RP2 that controls the button box is also used for stimulus presentation.

A more likely circuit design for button acquisition...



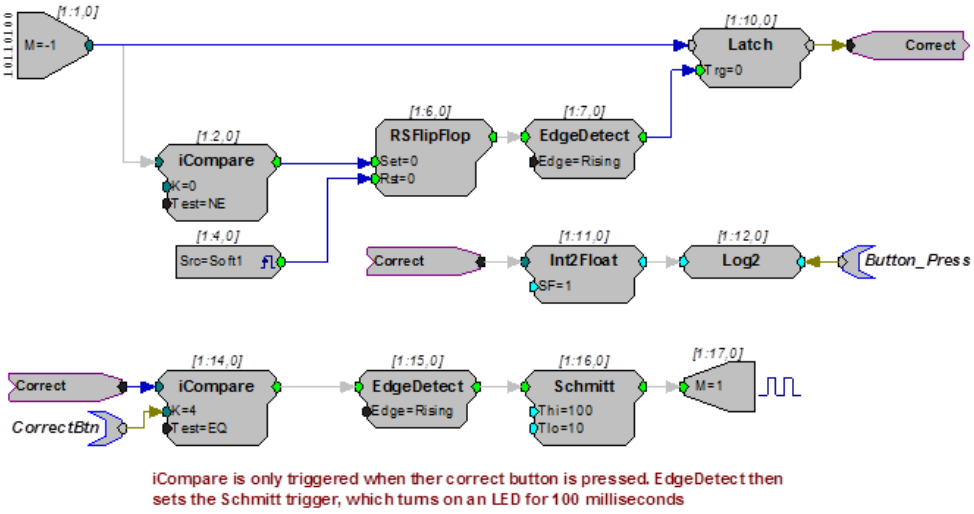
In this example, the WordIn produces an integer value based on the buttons pressed. When a button press occurs, an iCompare generates a logical high that triggers the Latch component. The Latch stores the value of the button press until the next button press occurs. The Button_Press parameter tag allows the user to read the value from the button box. If only the first button press is important then a reset line should be included in the circuit to rest the Latch.

Resetting the Latch...



In the previous examples all button presses are acquired, that is, if a person presses buttons simultaneously there is the chance that both responses will be obtained. This will happen infrequently with circuits that use an iCompare and Latch, but it is still possible. In some cases the user will want to determine if the proper button press was acquired or wait until a particular button press has happened. Additional circuitry can be added that checks for this.

Identifying the correct button press...



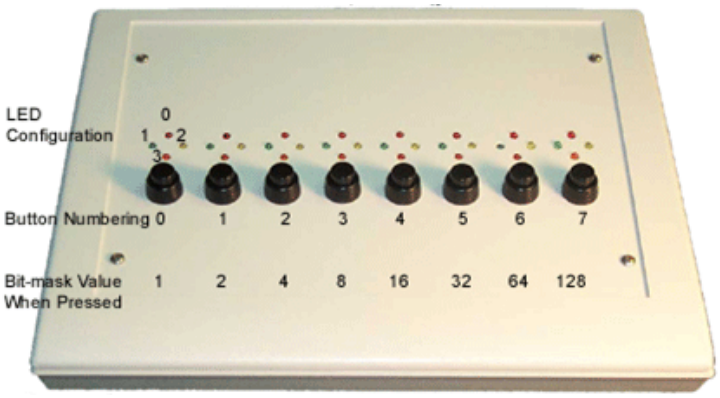
In this example, the top part of the circuit detects if a button is pressed. The button press value is also translated into a value representing which bit was read. For example, if the bit in bitmask value is 16, then Log2 converts the value to 4. This lets the user determine, via the Button_Press parameter tag, that bit 4 was high.

The lower part of the circuit tests to determine if the correct button was pressed. If so, an LED is flashed. A parameter tag is used to identify the correct button press. The iCompare is only triggered when the correct button is pressed. The EdgeDetect component then sets the Schmitt that turns on the first LED for 100 milliseconds.

Button box circuits can be incorporated in to all TDT System 3 software. For information on using the button box with other applications please see that application's documentation. If you have questions about how to design your own applications for the button box call 386-462-9622 for technical assistance.

Controlling the LEDs

Their are several methods to control LEDs. The button box may have up to four LEDs for each button and each LED can be turned on and off independently of any other. Using the LEDs involves two steps: 1) designating the LED to turn on or off and 2) turning the LED on and off. LEDs are designated by specifying the column (button number) and position (LED number).



BBox Organization of LEDs and Buttons

Bit Patterns Table

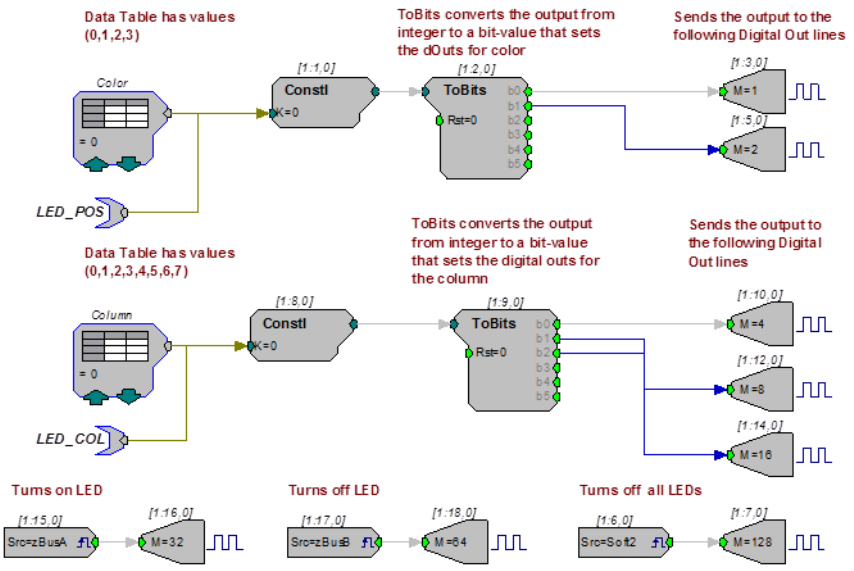
Selecting an LED	
Bits 0, 1:	Control the position within a column
Bits 2, 3, 4:	Control which column is selected
Turning on/off LEDs	
Bit 5:	Turns on selected LED
Bit 6:	Turns off selected LED
Bit 7:	Turns off all LEDs

Note: Because the button box has its own power supply, the LED's will remain on until they are turned off via the RP2 or RV8 or until the power is turned off.

The circuits shown below, as well as some MATLAB examples for use with ActiveX controls, are included with RPvdsEx (RPvdsEX\Examples\ButtonBox). In the first design the user designates the LED and the button number or column position in two separate steps. In the second the steps are combined. In the final design LED designation and on/off information are combined in a single word.

Designating the LED and button number or column position in two separate steps...

In the example below there are two sets of inputs used to specify the LED. The first controls which LED (LED position within a grouping) is lit while the second controls the column (button location) in which the LED is located. DataTables are used to test and run the circuit within RPvdsEx and parameter tags (LED_POS and LED_COL) are included to allow users to control the position and column values from another application.



To follow along with this example:

- Open the LED1 RPvdsEx file in the ButtonBox example folder (TDT\RPvdsEX\Examples\ButtonBox).

To designating and turn on/off an LED and button:

1. To set the color or position of the LED (0 = Top, 1 = Left, 2 = Right, 3 = Bottom), click the **green up and down arrows** on the DataTable labeled Color.
2. To determine which column the LED is in (0 = Far Left... 7 = Far Right), click the **green up and down arrows** on the DataTable marked **Column**.
3. To turn on the LED, press the **zBusA** trigger button in RPvdsEx. Make sure to click the **pulse**  button for the zTrig. To turn off the LED press the **zBusB**  trigger.

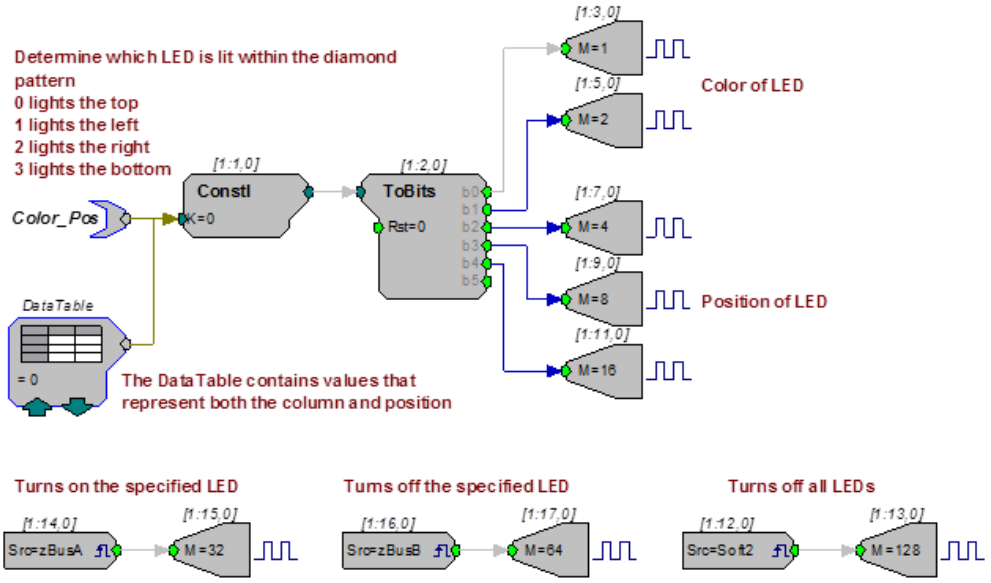
You can select (one at a time) several lights to turn on and off.

For example, to light the top LED in the first column and the bottom LED in the last column perform the following steps:

1. Set the **Color** DataTable to **0** and the **Column** DataTable to 0.
2. Turn on the LED by clicking the **zBusA** trigger button in RPvdsEx. This will turn on the top LED in the first column.
3. Set the **Color** DataTable to **3** and the **Column** DataTable to 7.
4. Click the **zBusB** trigger button in RPvdsEx. Both LED's should now be on.
5. To turn off the latter LED, click the **zBusB** trigger button.
6. To turn off all LEDs, click the **Soft2**  button in RPvdsEx.
7. To turn on all LED's in succession, set the **zBusA** trigger line high  and then cycle through the DataTable values.
8. To reverse the operation set the **zBusA** trigger low  , set the **zBusB** trigger high  , then cycle through the DataTable values.

Combining the position and column setup...

The following example combines the two data tables and uses one ToBits component to control the button box's LEDs.



The single data table used in this example contains values that combine the column and position.

For example:

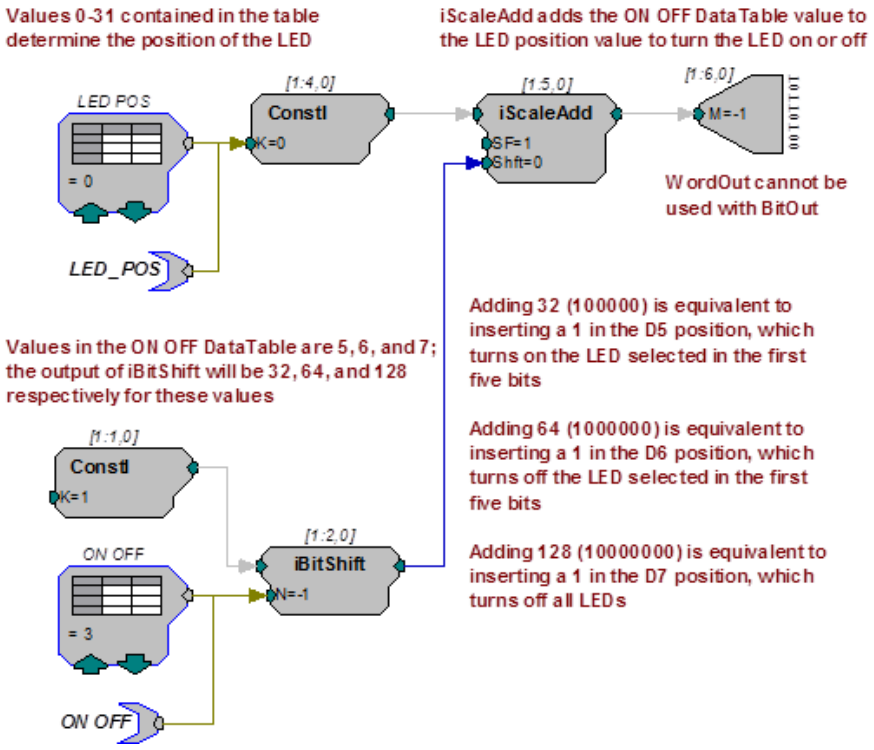
If 28 is used in the data table, the circuit selects the top LED in the seventh column. That's because the top position in the seventh column is represented by the digital number 11100 (as shown below), which equals 28.

Column Select Lines			LED Position Select Lines	
D4	D3	D2	D1	D0
1	1	1	0	0

To learn more about this example, open the LED2 RPvdsEx file in the ButtonBox example folder (TDT | RPvdsEX | Examples | ButtonBox).

Using a WordOut with a DataTable/ParTag for on/off actions...

The following example uses the WordOut component similarly to the way the WordIn is used in the button press example. As before, a DataTable is used to determine which LED to light. In the LED POS DataTable, values 0 - 31 are used to determine the position of the LED. In addition, another DataTable is used to set whether the LED is turned ON or OFF, all LED's are turned OFF, or if nothing is done when the LED is selected. This value gets added to the LED position value and is sent out via the WordOut component. The values for the second DataTable are 0 = 0 (nothing done), 1 = 32 (LED ON), 2 = 64 (LED OFF), and 3 = 128 (all LEDs OFF). The cycle usage for this example is half the cycle usage for the one above it. Notice that there are no BitOut components used. The WordOut and BitOut components cannot be used in the same circuit.



Note: See the Bit Pattern Table for a review of how each bit position is used.

This example is found in the LED3 RPvdsEx file in the ButtonBox example folder (TDT | RPvdsEX | Examples | ButtonBox).