RTBox v5/6: USTC Response Time Box

http://lobes.osu.edu/rt-box.php

What is it for?

Computer keyboard and mouse can be used to record response time (RT) to an event, such as visual or auditory stimulus. However, the accuracy depends on many factors, such as computer hardware specification, operating system, programming software, and user code etc. Even if the user code is well written, these devices are inaccurate, and even worse, often introduce bias. The major reason is that, the user program can get the time when the program reads the key or mouse response, not the time when the key or mouse is pressed.

The USTC Response Time Box (RTBox) is designed to measure response time with high accuracy. The microcontroller in the device will record the event time and button identity. The user code can read them anytime when convenient. Unlike keyboard or mouse response, the timing is independent of time your program reads the response.

Features

- Compatible to major computer systems, including Windows, MAC and Linux
- USB 1.1 and 2.0 compatible
- Measure both the button press and release time
- Built-in light port and pulse/sound port for trigger and calibration
- Built-in firmware upgrade so the device will never expire
- External buttons or button-driven TTL input for specialized keypad, such as MRI compatible keypad
- TTL output registering all input events and user event code, useful for EEG
- Extra TTL input for TR (repetition time) trigger from MRI scanner
- Analog-to-Digital Converter function
- Many features, such as TTL input and output, are customizable

Improvements over previous version

- Larger button space and buttons are along an arc
- Either high or low active TR signal (or TTL) can be received
- Software controlled reference signal for light and sound input

- Receive 8 external button/TTL input
- Event code TTL can have 8 bits
- Hardware is more flexible, such as above TTL input and output can be assigned for different purpose

Specification

- Four buttons allowing user to label with descriptive names
- Detection time resolution: about 6 μs (90 μs prior to v5.1)
- Dimensions: 5.5 x 4.5 x 1 (h) inches, 14 x 11 x 2.5 (h) cm
- Weight: ~5 oz

Parts of the device

- Photodiode with rubber suction and cable
- Four buttons
- USB port: connect to computer USB port with the provided cable
- Light port: receive light signal from the provided light sensor
- Sound/Pulse port: receive pulse signal from audio/stimulator device
- DA-15 input port: external switches or button-driven TTL, and TR
- DB-25 output port: output TTL for input events and 8-bit event code

How it works?

When the device is connected to a USB port of a computer, it will be recognized as a serial port. The device is powered by the USB port.

For principle about how the device works, you can check the paper on *Behavior Research Methods*.

Basically, the device detects button and port events with interval about 6 μs. When it detects an event, it records the event code and time based on its own clock, and sends them to the computer serial port. Each event contains 7 bytes of data. The first byte is the event code, and the rest 6 bytes contain the absolute time since the device is powered.

At the computer side, the device driver reads the data from serial buffer, identifies the event type, and calculates the response time based on device clock.

Install driver

When the device is connected to a USB port of the computer for the first time, the computer system may need the driver to recognize the device. You can download the latest version of the driver for your operating system from http://www.ftdichip.com/Drivers/VCP.htm.

For MAC OSX, the downloaded file is a dmg file. Run it and the driver will be installed. You may be asked to restart the computer.

For Linux, the driver should be included in your system. In case it is not, you can download the driver and follow the instruction in the ReadMe file.

How to use it?

There are two ways to use the device to measure response time. If the user code has accurate stimulus onset timestamp, we need only to get the response time with the same timestamp, and then a subtraction to get the response time. This is the recommended method. This method relies on the method we developed to synchronize the device clock with computer clock.

The second way is to provide a trigger to the device to indicate the onset of stimulus. The device will detect both trigger and button events. We get the response time by calculating the time difference between the two events. This method is only needed when the user code doesn't have accurate stimulus onset time.

If there is a TTL pulse synchronized with stimulus, you can connect it to the pulse port with a cable (not provided). Some stimulus equipment, such as an electrical stimulator and audio stimulator, has built-in trigger output for this purpose.

For computer-controlled visual stimulus, it may not be easy to generate an accurate trigger. The timing error could be caused by many factors, such as the time difference between code and real display, time difference due to the stimulus location on computer screen etc. We provide a photodiode as light trigger for visual experiments. You can mount the suction onto your screen, and program a light square which is within the same frame as the onset of your stimulus. If you use only grayscale visual stimulus, you can use our video switcher to provide a pulse trigger (http://lobes.usc.edu/videoswitcher).

The second method requires additional hardware connection. Although it is the most accurate solution, it is less convenient. Also the photodiode and the trigger light may be a distractor for the subjects.

Driver code based on PsychToolbox in MatLab

We provided a MatLab/Octave code (RTBox.m) based on PsychToolbox. The code can detect the device automatically, and use all its functionality. If you don't use MatLab/Octave, you may "translate" this code into your own language, suppose your software environment has similar functions for serial communication and timestamp.

There are also some demo codes, showing how to use RTBox in an experiment.

How to do calibration?

The timing of device is very accurate and you normally don't need to calibrate it. However, there may be a time difference between the nominal stimulus onset and the trigger marking the onset of the stimulus, and you need to measure this difference so you can correct the measured response time. Ideally, this difference should be fixed for a certain setup.

The light and pulse/sound ports can be used for calibration purpose, especially for the computer-based visual stimulus

The demo code RTBoxdemo.m can also be used for calibration. It will show a flashing light. You can connect the photodiode to the light port so it can detect the flash. The measured response time will be the difference between Screen('flip') time and the onset of flash, taking all time delays into account. Then you can simply correct response time by the measured difference. Note that you need to enable the detection of 'light', since it is disabled by default.

How to test synchronization of computer and device clocks?

We synchronize the computer and device clocks by a serial trigger. When we send a trigger to the device, we record the computer time when the trigger is sent, and get the device time when the device receives the trigger. Then we get the difference between the two clocks. Later when we have device time for an event, we can calculate its computer time based on the difference. This is not guaranteed to work with all system setup. When

possible inaccurate clock synchronization is detected, you will see a warning message. This typically indicates your system is not accurate on timing, not only for the response box, but also for other timing related measurement.

The speed of the computer and device clocks may be slightly different. The RTBox('ClockRatio') command will measure it and apply correction automatically. You'd better to run it when suggested by the driver code, although this is not mandatory.

How to use the ADC function of RTBox?

The RTBox can be used as a simple analog-to-digital converter (ADC). For details and instruction, check the code RTBoxADCDemo.m. For v5+ hardware, the light signal is connected to ADC channel 8, so once the light sensor is connected to the RTBox, channel 8 will show the light signal. The ADC channels are pins 1~7 of DA-15 port.

Note that the device will stop its event report during ADC function.

Frequently asked questions and answers

1. I am warned to do RTBox('ClockRatio'). What is it for, and do I have to do it?

   Answer: the short answer is no. However, clock ratio correction will further improve timing accuracy.

2. I am warned "Failed to change latency timer due to insufficient privilege" or asked sudo password to change it. What does that mean, and how do I proceed?

   Answer: the warning indicates you are running as a restricted user. The code tries to change the latency timer of the USB serial port to the shortest 1 ms, so the reading of device will be faster. For example, on Windows and MACI systems, if the latency time is default 16 ms, RTBox('clear') will take about 380 ms, while with 1 ms, it takes about only 60 ms. The failure to change the value affects only the speed of reading. As the warning message suggests, you can run the code as administrator once (or simply input sudo password for MAC), and it will change the latency timer. After the change, you can run as restricted user. On Windows system, if you plug the box into a different USB port, you might need to make the change again for that port. You may need to right-click the Matlab icon or exe file, and *Run as administrator* so Matlab can change the LatencyTimer. You need to do this only once. On Linux, it seems that you have to run as super user anyway to access the USB-serial port. On MAC, the change will take effect after you reboot the computer, and the change is permanent for all USB ports.

3. Does it make difference if I plug the RTBox to different USB port?

Answer: from our test, we didn't see time accuracy difference for different USB ports. However, we do see, on some computers, that reading speed is a little slower if the RTBox is connected to a port from a USB hub. As a rule of thumb, the USB ports at the back of a desktop computer may be better.

4. Why some of the TTL outputs are inverted, while others are not?

   Answer: the diagram of RTBox with TTL outputs is shown in Appendix 2. By default, the output pins 1~8 are normal TTL (high active), while the output pins 17~24 of DB-25 port are inverted TTL (low active). This is compatible to the default polarity of Neuroscan EEG system. For other system, such as BrainProduct EEG, you can either set the polarity in the EEG software so it can receive the TTL with correct polarity, or you can change the RTBox TTL polarity to meet your system's requirement.

Contact us

If you have question or suggestion about the device, please contact us at:

Xiangrui Li
B71 Psychology Building
The Ohio State University
1835 Neil Ave
Columbus, OH 43210

Phone: (614) 292-1847
Email: xiangrui.li@gmail.com

Appendix 1: Serial Commands

This lists all the serial commands, and the returned data, if any, by the device. All commands are a single byte data.

Uint8	Char	Returned	function	Comments
01			Next byte will be TTL byte	5.0+
63	?	[63 state]	Ask button states	Return [state 63] for 1.4-
66	B		Enter boot mode from simple	R to return from boot
69	E	[69 state]	Ask Enable state	1.4+
101	e	e	Wait for enable byte	4.1+
16			Send data to EEPROM	4.3+
17			Read data from EEPROM	4.3+
88	X	USTCRTBOX,921600,v5.x	Ask device identity	Also switch to advance
120	x		Switch to simple mode	1.4+
89	Y	[89 6-byte time]	Ask device time	
71	G		Go to ADC firmware	4.2+

Command uint8(1) indicates next byte will be 8-bit TTL. The device will wait for 71 ms for the next bytes. If timeout occurs, it will return 1, indicating error.

For command "?", bits 0~3 of the returned state byte are for buttons 1 to 4.

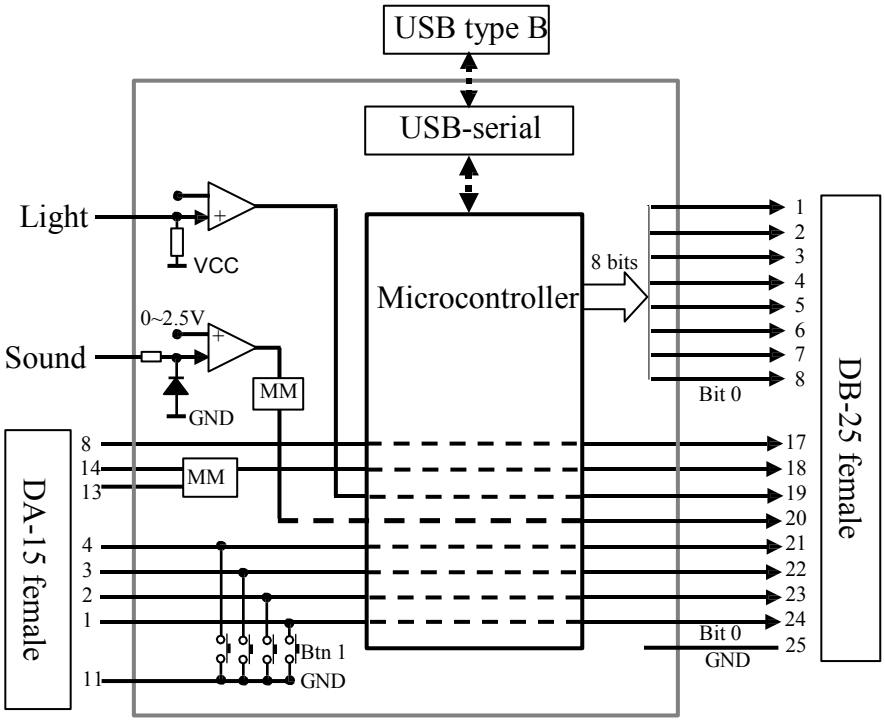
Command "B" is used for firmware update, so you should not use it.

For command "E", the 1~6th bits of the returned state byte are for button press, button release, pulse/sound, light, TR and aux respectively.

Command "T" is used by firmware developers to test the round trip speed, and is not available for normal device.

Command "X" switches device into advanced mode, and returns the device ID "USTCRTBOX", device clock frequency 921600, and version of firmware.

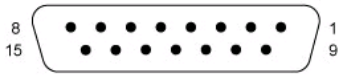
Appendix 2: Schematic of RTBox 5.x and 6.x



MM is monostable multivibrator. Both MM generate low active output.

Appendix 3: Pinout of DA-15 port

Looking at the female DA-15 port on the RTBox, the pins appear as the following picture. This port is designed to receive external button-driven TTL, or to connect external buttons (switches). There are two pins to receive MRI scanner TR trigger.

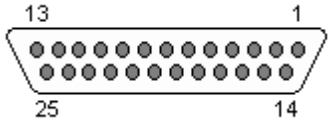


Pin	Signal	Comments
1~8	Button 1~8	5~8 may be OR'ed with other signal
9	Sound	After removing negative component
10	No connection	
11	GND	
12	No connection	
13	TR	Low active pulse
14	TR	High active pulse
15	VCC	For special purpose

If you want to connect your external buttons/switches or button-driven TTL input to RTBox, use pin 11 as ground, and connect your signal to pins 1 to 4. You could also connect switches/TTL to pins 5~8. In other words, the device can connect to up to 8 TTL/switches.

Appendix 4: Pinout of DB-25 port

Looking at the female DB-25 port on the RTBox, the pins appear as the following picture. This port is designed to output TTL signal to EEG system. For now, we know it is compatible with Neuroscan, BrainProduct and Biosemi EEG system.



Pin	Signal	Comments
1~8	TTL Bits 7~0	8-bit event code (PC 7~0)
9~14	No connection	
15	RST	For firmware flash only
16	VCC	Normally you should not use this
17	Aux	Button 8, signal from DA-15 pin 8.
18	TR	Button 7, or signal from DA-15 pin 7, 13 or 14.
19	Light	Button 6, signal from light port or DA-15 pin 6
20	Sound/Pulse	Button 5, signal from sound port or DA-15 pin 5.
21~24	Buttons 4~1	Signal from DA-15 pins 4~1.
25	GND	

A light, sound, TR and aux event will disable the later detection of itself. TR TTL output can be always on if needed. By default, the 8-bit event code is high active and 8-bit signal at pins 17~24 is low active. But the levels are configurable.