The Black Box ToolKit

TTL to USB Keys Module v1 User Guide

Credits:

Author: Dr. Richard R. Plant, C.Psychol, CSci, AFBPsS

Covers the following hardware:

The Black Box ToolKit TTL to USB Keys Module v1

For the following platforms:

Microsoft Windows XP SP3, Vista SP2 (32/64), Windows 7 SP1 (32/64)
Windows 8 (32/64), Windows 8.1 (32/64), Windows 10 (32/64), Windows 11 (64)

Mac OS 8/9, OS-X

Linux 2.4 and greater

Contact details:

Phone: +44 (0)114 303 00 56
Fax: +44 (0)114 303 46 56

Email: info@blackboxtoolkit.com
support@blackboxtoolkit.com
sales@blackboxtoolkit.com

Web address: www.blackboxtoolkit.com

Contents

Please read this manual fully before using the TTL to USB Keys Module. Improper connections to the input ports can cause the Module to reset, or even cause permanent damage. Damage caused by over, or reverse, voltage conditions resulting from improper wiring to the ports is not covered under the warranty.

When not in use you are advised to disconnect it from your equipment.

1. Introduction

The Black Box ToolKit TTL to USB Keys Module (TTLtoKeys) enables you to quickly and easily create USB keyboard events from TTL signals. When a TTL signal, or event trigger, is detected it is converted into a key press, or keystroke, as though a key on a standard keyboard had been pressed. Once plugged-in the TTLtoKeys Module appears as a standard USB HID keyboard and requires no drivers.

Any Experiment Generator, or other software, that can accept standard USB keyboard inputs, on any platform, e.g. Windows, MacOS, Linux etc, can be used to read TTL signals. For example, it can be used to read TTL event markers in EEG and Eye Tracking studies as key presses. Because standard key presses, or keystrokes, are used this means that you can also receive TTL input into environments which are normally sandboxed, e.g. Web Browser based experiments.

The Module itself is designed to output key presses, or keystrokes, in response to 8 TTL inputs on a standard 25-way female D connector on the front panel (pins 2 to 9) and two shared TTL inputs on a 2.5 mm stereo socket on the rear. The two rear inputs are shared with pins 2 & 3 on the front 25-way D. As TTL1 & 2 share inputs with both the front 25-way D and the rear connector, you should take care not to confuse the source of the inputs on those pins.

There are a total of three operating modes which can be selected by pressing the mode button on the rear of the Module. The current operating mode is indicated by the colour of the LED next to the mode button. Each operating mode differs in the way it translates and then “types” the TTL signals back to your computer. This allows for maximum flexibility and ease of use. In each mode the TTL input lines are sampled every millisecond (1 kHz) and the corresponding USB HID key presses, or keystrokes, are transmitted on the next available USB packet.

The TTLtoKeys Module can run in two latency modes: Quantitative and Qualitative. In Quantitative mode all keyboard output is millisecond accurate by default (Mode 1). In Qualitative modes because multiple letters, or numbers, are typed as a string of key downs and key ups you should treat typed output as not being millisecond accurate by default as output is buffered (Modes 2 & 3).

1.1 Quantitative Versus Qualitative Latency Modes

In mode 1, the default, all 8 TTL lines are sampled every 1 mS and output also occurs at 1 mS by default. Mode 1 is defined as a Quantitative mode where timing is designed to be accurate to 1 mS.

Modes 2 and 3 are Qualitative modes where all 8 TTL lines are sampled every 1 mS, but as output is typed over multiple keystrokes from an internal FIFO buffer, the time when the value of the port is typed my not be millisecond accurate. This is because it takes some time to type the value of the port to the host computer as if you were physically typing using a real keyboard.

In Modes 2 & 3 you can also configure the delay between key presses and releases using our configuration App. By default this is set at 1 mS or 1 kHz. In addition the internal FIFO buffer itself is 125 characters long so if there are a lot of very fast line changes as the buffer fills up this may mean that typed output could be up to 125 mS later than the physical TTL line change.

Mode	Mode Type	Output	Source
Mode 1	Quantitative	1 kHz / 1,000 Hz – Output every 1 mS	Direct
Mode 2 & 3	Qualitative	250 Hz to 166 Hz – Output every 4-6 mS from on board buffer	Internal FIFO buffer

1.2 Mode 1: Unique KeyDown in Single USB Keyboard HID Packet for Each TTL Line – Red LED / Quantitative

The 8 TTL input lines on the 25-way female D connector (TTL1 through TTL8, or pins 2~9) are sampled at 1 kHz and each input line is mapped to an equivalent USB keyboard key stroke. This means the TTL lines are sampled each millisecond, or 1,000 times per second

When an input line is HIGH, or +5 V, the associated key is pressed. When the input line is LOW, or 0V, the associated key is released. The key press duration corresponding the amount of time the TTL line is HIGH. The two TTL input lines on the rear 2.5 mm connector are shared with TTL1 & TTL2 so can be used in tandem as they are OR'd together.

For example, if TTL1 detects the onset of an event, HIGH or 5V, the number "1" will be pressed and held down. When TTL1 detects the offset of an event, LOW or 0V, the number "1" key will be released. This is equivalent to pressing and holding the "1" key down on the keyboard until you release it. By holding a key down while the event is active this lets you easily assess duration.

If multiple lines go HIGH, more numbers will be added to the keystroke output, e.g. if TTL1 and TTL2 were HIGH at the same time then, "12" keystrokes would be held down and output on the same USB HID packet within a 1 mS timeframe. Each time there is a change on the 8 TTL input lines, keystrokes will be added and removed from the USB HID packet thus assuring that the changes can still be reported within a 1 mS timeframe.

Number keys 1~6 and LEFT SHIFT and LEFT CTRL are used to represent input lines TTL1~TTL8 respectively.

1.3 Mode 2: Unique KeyPress (Down & Up) for Each TTL line – Green LED / Qualitative

The 8 TTL input lines on the 25-way female D & 2.5 mm connector (TTL1 through TTL8, or pins 2~9) are sampled at 1 kHz. As each TTL input line transitions HIGH, or 5 V, the associated TTL HIGH key is pressed and quickly released. As each TTL input line transitions LOW, or 0 V, the associated TTL low key is pressed and quickly released. If there are no changes nothing will be typed. The two TTL input lines on the rear 2.5 mm connector are shared with TTL1 & TTL2 so can be used in tandem as they are OR'd together.

For example, if TTL1 detects the onset of an event, HIGH or +5 V, the number "1" will be pressed and quickly released. When TTL1 detects the offset of the event, LOW or 0 V, the letter "a" key will be pressed and released. This is equivalent to pressing and releasing the "1" key on a keyboard for the onset and pressing and releasing the "a" key for the offset.

1.4 Mode 3: Port Value Representing the Status of all TTL Lines in Either Hexadecimal or Decimal Characters – Orange LED / Qualitative

The 8 TTL input lines on the 25-way female D & 2.5 mm connector (TTL1 through TTL8, or pins 2~9) are sampled at 1 kHz. The two TTL input lines on the rear 2.5 mm connector are shared with TTL1 & TTL2 so can be used in tandem as they are OR'd together.

By default a two byte hexadecimal value representing the TTL port state is typed as though from a USB keyboard each time there is a change on any of the TTL input lines. If there is no change then nothing will be typed.

For example, the Hex value, "01", would be typed if TTL1 or pin 2 on the 25-way D went HIGH. Effectively this is the same as pressing and releasing each of the following keys in sequence, "0", "1", i.e. 2 key presses. Hex values range from "00", for no lines active, i.e. 0 V, to "ff" for all 8 TTL lines HIGH, or +5 V.

Whereas if decimal output is selected in the configuration software a decimal value representing the TTL port state as an 8-bit number is typed as though from a USB keyboard. Each time there is a change on any of the TTL input lines a new value will be typed. If there is no change then nothing will be typed.

For example, the decimal value, "001", would be typed if TTL1, or pin 2, on the 25-way D went HIGH. Effectively this is the same as pressing and releasing each of the following keys in sequence, "0", "0", "1", i.e. three key presses and releases.

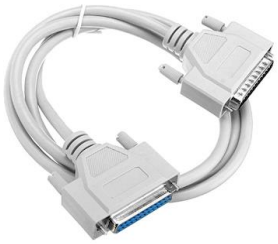
NOTE: In all modes TTL input lines are sampled every millisecond (1 kHz) and the corresponding USB HID key presses, or keystrokes, are transmitted on the next available USB packet. This means that in mode 1, all TTL events are delivered within 1 mS as the status of all TTL lines can be delivered in a single USB HID data packet.

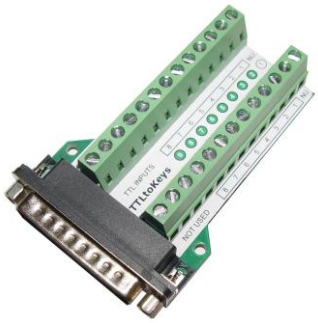
However, in modes 2 & 3, activity on the TTL input lines is stacked in an event buffer and transmitted sequentially as individual USB HID keyboard key presses. Consequently, during high levels of TTL input change there could be significant delays to the transmitted USB data and keys typed in the sequence the TTL lines changed.

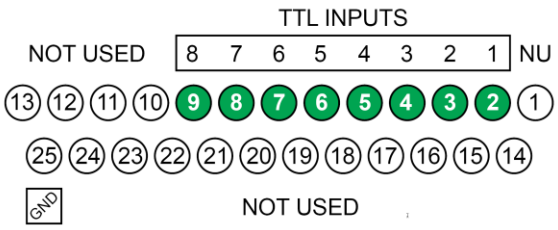
2. TTLtoKeys Module Pinouts & Connectors

Using the BBTK TTLtoKeys Module is simple. The module plugs into a USB port of any computer without the need for additional drivers and appears as a standard USB HID keyboard. Depending on the operating mode TTL inputs are represented as keystrokes or key presses.

If you have chosen not to purchase the optional breakout board shown on the right below you may need to make up your own parallel cable which maps from your specific devices TTL output pins to the TTLtoKeys Modules front facing 25-way female D.







The front of the module has a 25-way female D connector.

Pins 2~9 correspond to TTL input lines 1~8. Pin 25 is ground.

The rear of the module has standard USB type B connector by which it is connected to the PC which will receive the USB HID keyboard key presses and keystrokes.

A 2.5 mm stereo socket mirrors TTL1 & TTL2 on the front 25-way D (tip & ring respectively, sleeve GND), i.e. OR'd with TTL1 & TTL2.

A multi-color LED indicates the operating mode of the Module. To switch mode simply press the button once.

Available modes:

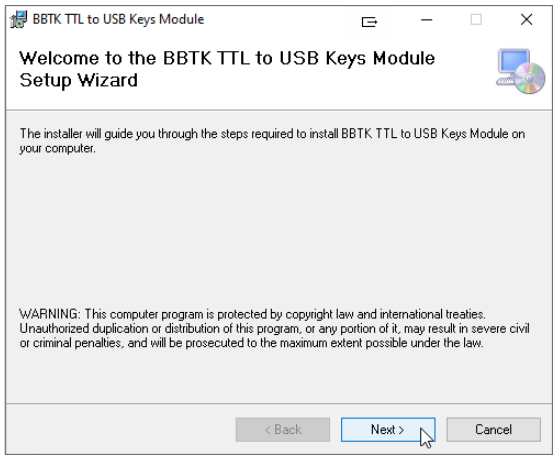
1. Mode 1: Unique KeyDown in single USB Keyboard HID packet for each TTL Line (1 mS transmit latency[#])
2. Mode 2: Unique KeyPress (Down & Up) for each TTL line (2 mS transmit latency[#])
3. Mode 3: Port Value representing the stats of all TTL Lines in a either Hex or Decimal characters (4 or 6 mS transmit latency depending on number system used[#])

[#]Dependent on specific USB subsystem and using the USB Configuration utility to check for correct installation.

3. The BBTK TTLtoKeys Module Configuration Utility & Basic Functional Checks

3.1 Installing the Configuration Utility

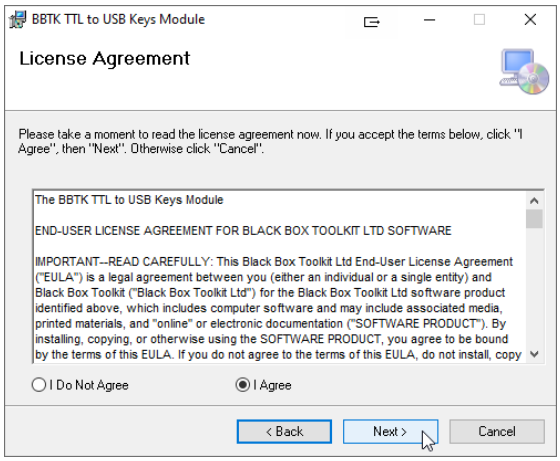
To configure the keys presses that the module responds with and configure the module other settings you will need to make use of the BBTK TTLtoKeys Configuration Utility.



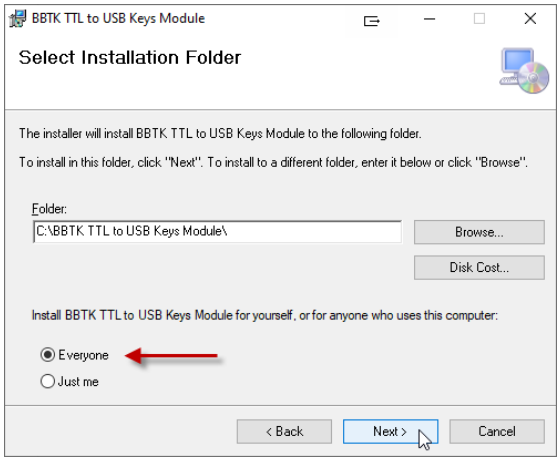
Ensure you are logged into the local PC with Administrator rights and insert the CD into your CD drive. The setup should automatically run. If not you may need to manually run setup.exe.

If the Microsoft .NET 4 Framework is not installed on your PC you may be prompted to download and install it.

An offline version is included on the CD under the DotNetFX40 folder. If you wish to use this press Cancel, install the Framework and the rerun the BBTK TTLtoKeys Module setup and continue as below.

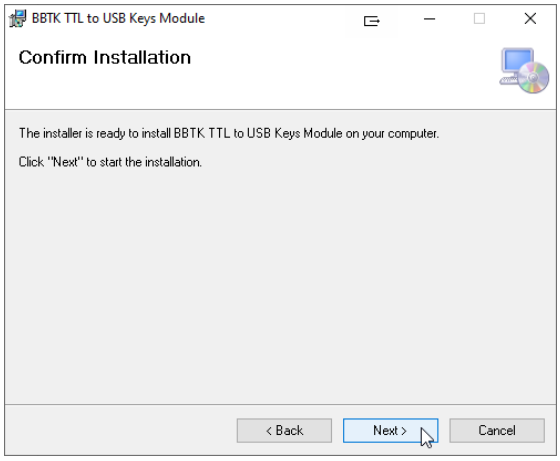


Before you can install you will need to Agree to the License Agreement.



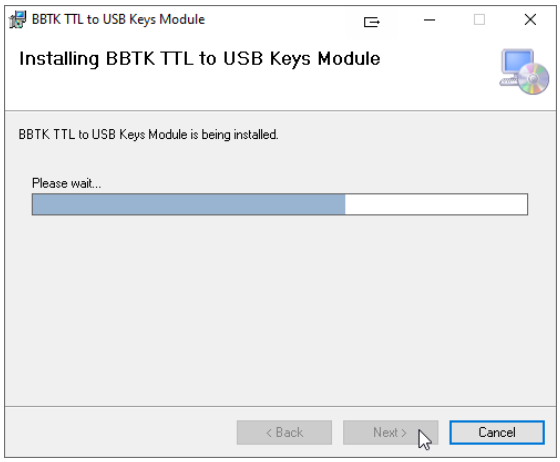
Select the folder where you want to install the BBTK TTLtoKeys Module.

In general you should select Everyone who uses this PC so that everyone who has an account on the PC can use the Configuration Utility.

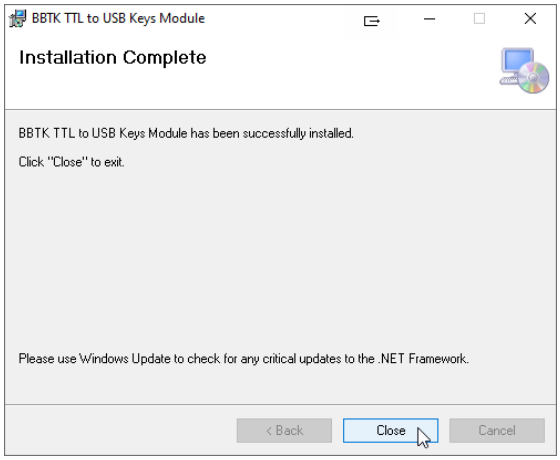


Next confirm installation.

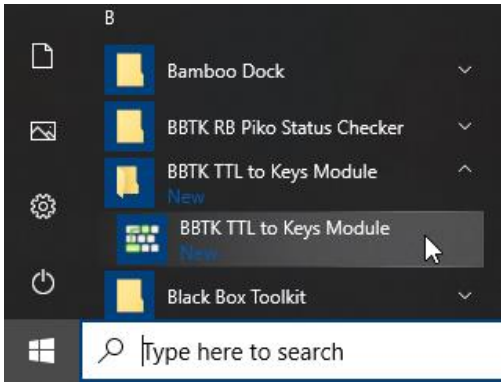
At this stage you may be prompted by UAC to confirm the application Publisher. Click on Yes to continue.



The BBTK TTLtoKeys Module Configuration Utility will now be installed.



To finalise the installation click on close.



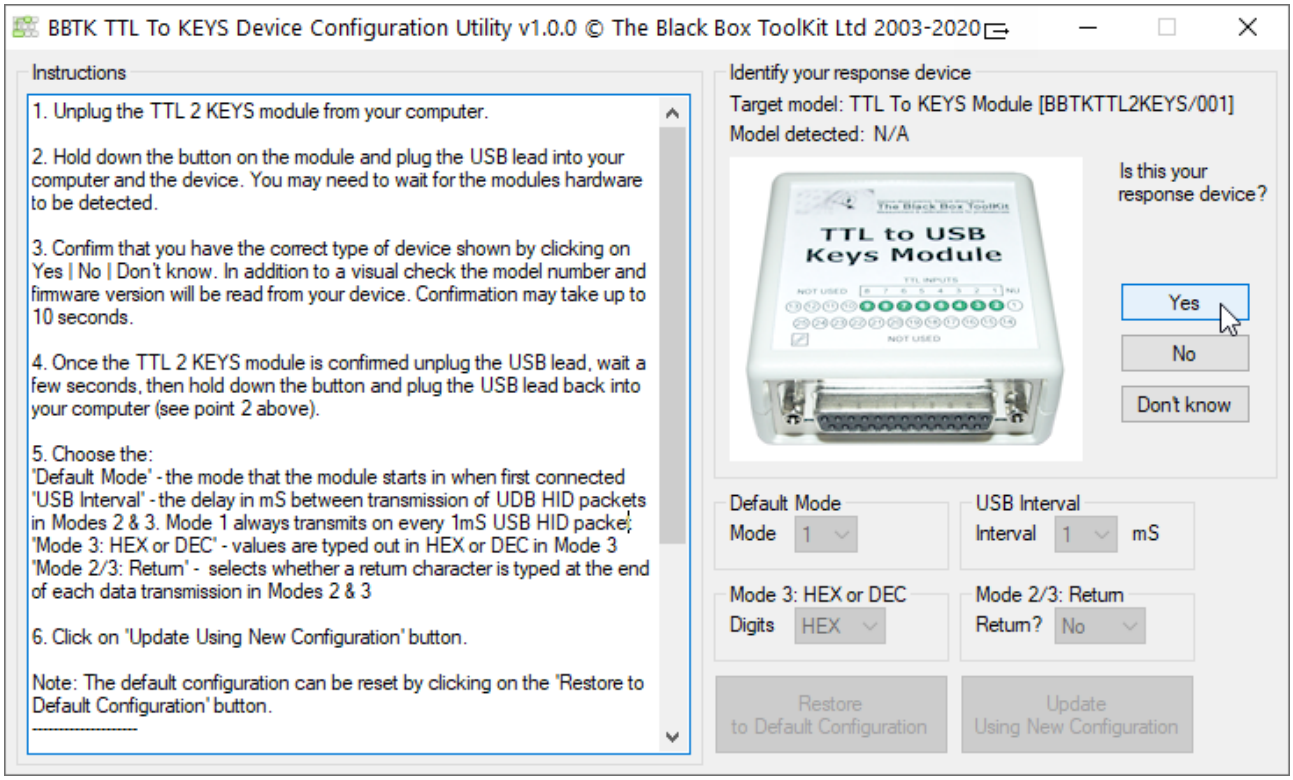
To start the BBTK TTLToKEYS Module Configuration Utility click on its icon from the Start Menu.

3.2 Using the Configuration Utility

The TTLtoKeys configuration utility lets you configure the default start-up mode of the Module and whether Mode 3 outputs, or types, the TTL port value in Hexademical or Decimal. Once configured these are saved into the Module and will be retained even when unplugged and not powered.

In Modes 2 & 3 you can also configure the delay between key presses and releases. By default this is set at 1 mS or 1 kHz.

In Mode 3 you also have the option to configure whether a CR, or Return, is sent after each Hexadecimal or Decimal number has been typed, i.e. as if the Enter key had been pressed on a keyboard.



To enable you to configure the TTLtoKeys Module start the configuration App and should hold down the button on the rear of the unit at the same time as plugging the USB lead in. Once the USB lead is plugged in you should release the button to go into configuration mode.

To enable the configuration options, confirm you have the module shown by clicking on the “Yes” button.

Once you have configured the Module click on “Update Using New Configuration” button. If you do not wish to save your changes into the Module quit the App.

Should you wish to restore the Module to its shipping state, click on “Restore to Default Configuration”.

3.3 Carrying out a Basic Functional Check in a Text Editor

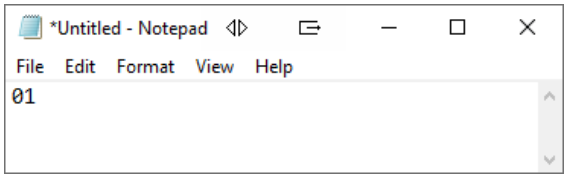
Often it's useful to check that your equipment is connected to the TTLtoKeys Module correctly and that the module is sending the appropriate USB HID key presses and keystrokes in response to changes on the TTL input lines.

Remember: TTL Inputs on the Module are +5 V TTL outputs FROM your equipment.

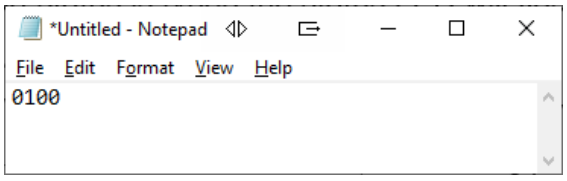
A basic functional check can be carried out using a simple text editor:

1. Open NotePad, or equivalent text editor on your computer and ensure this has focus by clicking the mouse on the application.
2. Plug the TTLtoKeys Module into a USB port on your device using the USB cable supplied. This should be a motherboard socket and not a USB hub or switch.
3. Set the TTLToKEYS Module to Mode 3 using the mode button on the rear of the Module by pressing it twice. By default this mode produces a two character Hexidecimal port value. Once you have switched to Mode 3 this will be indicated by a solid orange indicator LED.
4. Connect the TTL lines that you wish to monitor to the TTL inputs of the TTLtoKeys Module using either the front 25-way D or rear 2.5 mm socket noting the connections as detailed in section, "7.2 TTLtoKeys Module Pinouts & Connectors", page 41 of this manual. Connect the ground of your TTL device to pin 25 of the 25-way D connector of the module, or the common ground of the 2.5 mm socket.
5. A change on any of the TTL input lines should trigger a Hexadecimal value to be typed out into the text editor that corresponds to the binary value of the TTL input lines. As the TTL port status is typed the status LED will flash.

   For example, "01" means that the TTL1 line has gone HIGH, or +5 V. When this drops to LOW, or 0 V, "00" will be typed straight afterwards with no spaces.

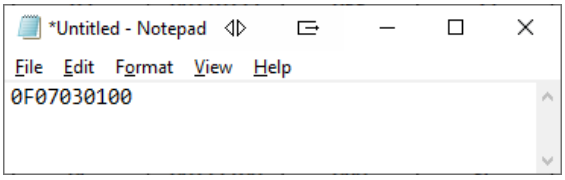


- "01" = TTL1 or pin 2 is HIGH, +5 V.



- "01" = TTL1 or pin 2 is HIGH, +5 V
- "00" = All TTL pins go low, 0 V

A more complicated example where more than one TTL input line is active at the same time is shown below:



- “0F” = TTL1, TTL 2, TTL3 & TTL4 HIGH, or +5 V
- “07” = TTL1, TTL2 & TTL3 HIGH, or +5 V
- “03” = TTL 1 & TTL2 HIGH, or +5 V
- “01” = TTL1 HIGH or +5 V
- “00” = All lines LOW, or 0 V

4. Using the TTLtoKeys Module

4.1 Worked Examples Using the TTL to USB Keys Module in Each Mode

There are a total of three modes the TTLtoKeys Module can operate in. The first, Mode 1 is millisecond accurate and is Quantitative. Whereas Modes 2 & 3 are Qualitative and may not be millisecond accurate as typed output takes time to arrive at your computer. This is because each letter, or number, is typed as though on a standard USB keyboard and each key press needs to go down and up.

In addition the internal FIFO buffer itself is 125 characters long so if there are a lot of very fast line changes as the buffer fills up this may mean that typed output could be up to 125 mS later than the physical TTL line change.

4.2 Mode Selection & Activity LED

Each time there is activity on the USB HID, i.e. a keystroke or key press is sent to the computer, the LED between the USB socket and mode selection button will flash on and off.

The color of the LED is dependent on which mode the Module is in.

LED Colour & Mode		Mode Type	Output	Source	Description
	Mode 1 RED	Quantitative	1 kHz – Output every 1 mS	Direct	Single keystroke for each TTL input line. Keystroke held down for duration each TTL Line is HIGH, or +5 V. Released when LOW, or 0 V.
	Mode 2 GREEN	Qualitative	500 Hz – Output every 2 mS from on board buffer	Internal FIFO buffer	Single key press typed as each TTL line changes (unique key press for each TTL Line going HIGH/LOW, or +5V/0V).
	Mode 3 ORANGE	Qualitative	250 Hz or 166 Hz – Output every 4 mS (Hex) or 6 mS (Dec) from on board buffer	Internal FIFO buffer	Hexadecimal or Decimal TTL Port Value typed. Two letters or numbers typed when in Hexadecimal mode or three numbers typed when in Decimal mode.

A worked example covering each mode is outlined below. This assumes that the TTLtoKeys Module is plugged in and you have connected it to your equipment correctly so that it can receive TTL inputs from your equipment's outputs, i.e. your equipment's digital trigger out port. For testing basic functionality, refer to section 3.

4.3 Using Mode 1: Unique KeyDown in Single USB Keyboard HID Packet for Each TTL Line – Red LED / Quantitative

Ensure the TTLtoKeys Module is plugged into a USB port. Set the Module to Mode 1 using the mode button, as indicated by the indicator LED being solid red. By default after first plug in the Module should be in this mode and the LED solid red.

Connect the TTL lines that you wish to monitor to the TTL inputs of the TTLtoKeys Module, either via the front 25-way D or rear 2.5mm connector.

The 8 TTL input lines on the 25- way female D connector (TTL1 ~ TTL8) are sampled at 1 kHz, i.e. each millisecond or a 1,000 times a second and each input line is mapped to a USB keyboard keystroke. The two TTL line inputs from the rear 2.5 mm connector are OR'd with TTL1 & TTL2 so can be used in tandem.

When a TTL input line, or pin, is HIGH, or +5 V, an associated keystroke sent to your computer and a corresponding USB keyboard response will be received by your application. When a TTL input line is LOW, or 0 V, the associated keystroke is released. Each key press duration corresponds to the amount of time each TTL line is HIGH, or +5 V.

In Mode 1, up to 8 keystrokes are transmitted in a single USB HID packet and consequently TTL inputs 1 to 6 are represented by normal keyboard keys and TTL inputs 7 & 8 are represented by a keyboard modifier keys SHIFT & CTRL, as per the table below.

As we are sending up to 8 keystrokes in one packet this means that up to 8 TTL lines can HIGH, or +5 V, at a time and typed to the PC in one go. So in one keyboard HID packet there might be up to 8 keyboard keys flagged as being held down. Where keys are removed and flagged as being up, this is where a given TTL input line has gone LOW, or 0 V.

TTL Input Line	1	2	3	4	5	6	7	8
25-way D Pin	2	3	4	5	6	7	8	9
Key	1	2	3	4	5	6	SHIFT	CTRL

The Module will check that your USB bus is free before attempting to send a single data packet containing up to 8 keystrokes. If your computer conforms to the USB specifications it should accept this data every millisecond or at 1,000 Hz.

Note: An 'x' will be typed by the module if for whatever reason it is unable to send this data packet to your computer at the industry standard rates.

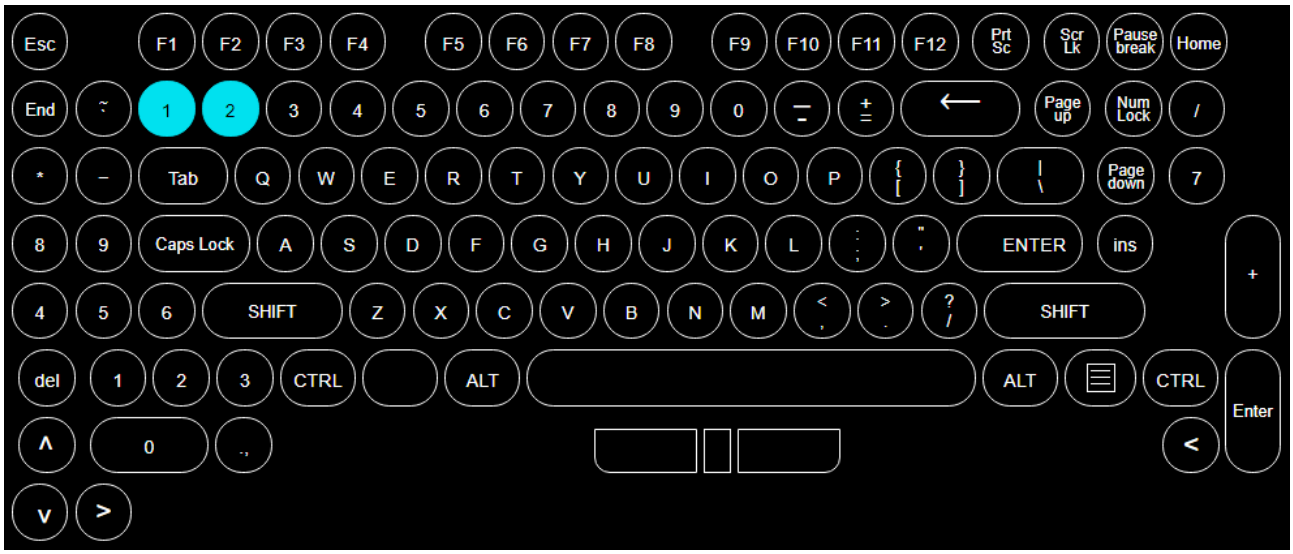
As we need to detect keystrokes you will need to set up your Experiment Generator, or other software, to look for KeyDown's and KeyUp's. These represent individual TTL Lines

going HIGH, or +5 V and LOW, 0 V, respectively. You can test that your Experiment Generator or other software is working correctly by using the keyboard you normally use for typing. For example, to simulate TTL Lines 1 & 2 (pins 2 & 3) being HIGH, or +5 V, simply press and hold down your 1 & 2 keys. Whist you hold them down this simulates receiving a “1” & “2” key from the TTLtoKeys Module.

Using the TTLtoKeys keyboard checking tool this is easy to accomplish.

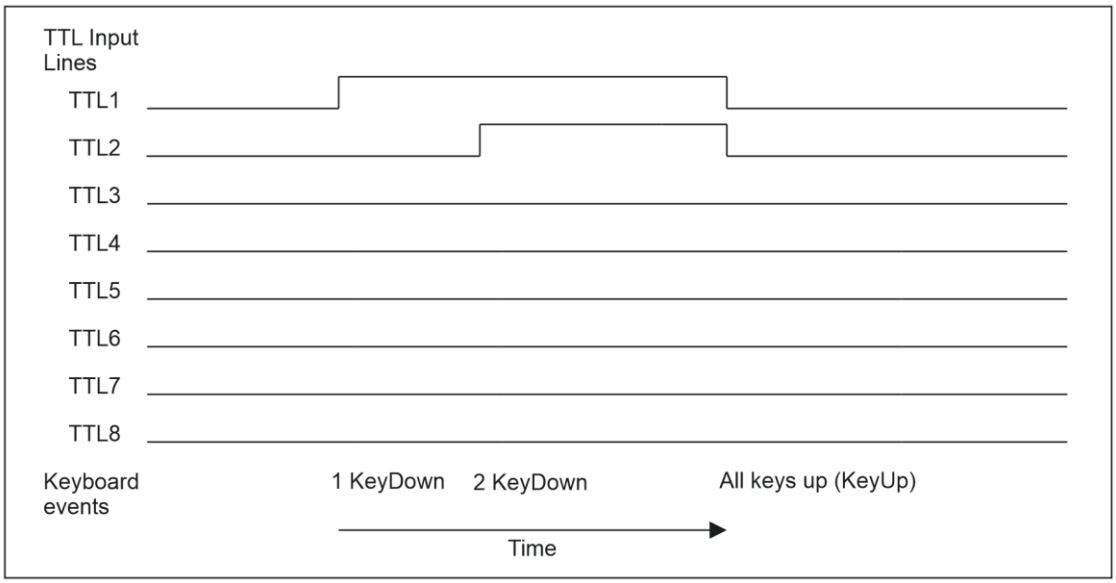
www.blackboxtoolkit.com/tools/ttltokeys/linechecker.html

Note: You should not use a text editor to check this as you will not be able to see the state of each keystroke, or TTL Line, in the output as you can with a keyboard key checker such as the one shown below.

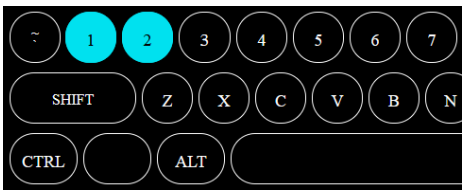


In terms of a concrete example, imagine we have an Eye Tracker that signals via TTL1 that the participant gaze has entered an Area of Interest (AoI) and when they make a response on a response pad, the Eye Tracker signals the button onset and duration via TTL2.

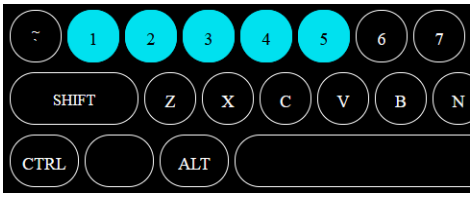
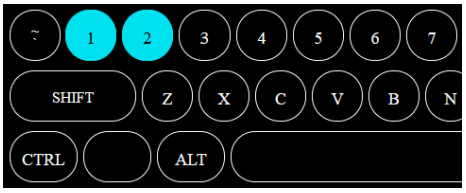
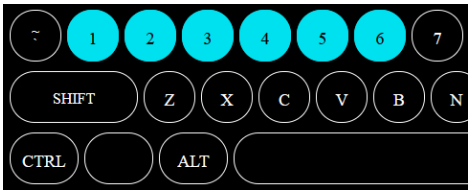
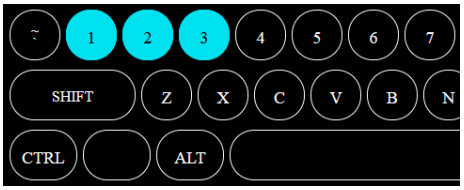
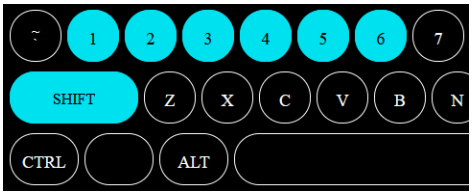
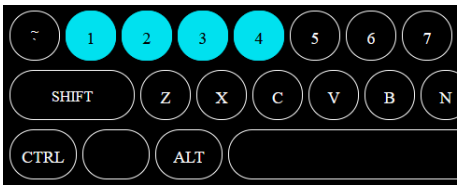
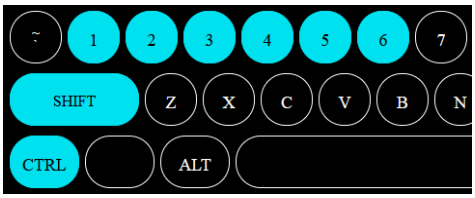
On a simulated TTL trace we can see that there is a “1” keystroke, or KeyDown event, to indicate the onset of the visual stimulus on TTL1 (HIGH, or +5V). After so many milliseconds there is a second “2” keystroke, or KeyDown event, on TTL2 (HIGH, or +5 V) to indicate a button being pressed on a response pad. Remember these are the signals sent FROM your own equipment to the TTLtoKeys Module. When the response button is released TTL1 & 2 both go LOW, or 0 V and a KeyUp event is generated showing that all keys have been released and all TTL lines are inactive.



When using the TTLtoKeys keyboard checking tool the TTL sequence above would appear as shown below. TTL Line 1 (pin 2) is active HIGH, or +5V, then TTL Line 2 (pin 3) goes HIGH and finally both TTL lines go LOW, or 0 V. The KeyDown keyboard keystrokes indicate when TTL Lines went active HIGH, or +5 V and the duration of those keystrokes matches the duration the TTL Lines were active for. Activity on a given keystroke ceased when there we a KeyUp event for that key, or keys.

Binary Line Input	Keystrokes Pressed, i.e. keys down	Description
00000000	No keys are pressed	To begin with no keys are pressed as there is no activity on any of the TTL Lines.
00000001 TTL1 = 1 Pin 2		The “1” key is pressed and held down as TTL1 is active. The duration this is held down is the duration TTL1 is HIGH, or +5 V for. Each number should be checked as a KeyDown event in your Experiment Generator or other software.
00000011 TTL2 = 2 Pin 3		TTL2 goes to active HIGH, or +5V along with TTL1. Both “1” & “2” keys are now pressed, i.e. there are two down keystrokes in the USB Keyboard HID packet sent from the TTLtoKeys Module.
00000000 No Keystroke		Both keys are released as TTL1 & TTL2 both go LOW to 0V. Your Experiment Generator or other software should check for KeyUp events. The duration each individual key was down for and was generating a KeyDown event from should equal the duration the TTL Line was active HIGH, or +5 V, for.

A table with all possible keystroke values is shown below.

Binary Line Input	Keystrokes Pressed, i.e. keys down	Binary Line Input	Keystrokes Pressed, i.e. keys down
00000000 No Keystroke			
00000001 TTL1 = 1 Pin 2		00011111 TTL5 = 5 Pin 6	
00000011 TTL2 = 2 Pin 3		00111111 TTL6 = 6 Pin 7	
00000111 TTL3 = 3 Pin 4		01111111 TTL7 = SHIFT Pin 8	
00001111 TTL4 = 4 Pin 5		11111111 TTL8 = CTRL Pin 9	

4.4 Using Mode 2: Unique KeyPress (Down & Up) for Each TTL line – Green LED / Qualitative

Ensure the TTLtoKeys Module is plugged into a USB port. Set the Module to Mode 2 using the mode button, as indicated by the indicator LED being solid green. By default after first plug in the Module will be in Mode 1 and the LED solid red.

Connect the TTL lines that you wish to monitor to the TTL inputs of the TTLtoKeys Module, either via the front 25-way D or rear 2.5 mm connector.

The 8 TTL input lines on the 25- way female D connector (TTL1 ~ TTL8) are sampled at 1 kHz, i.e. each millisecond or a 1,000 times a second. Each line is mapped to two USB key presses. That is, one keypress, or character, will be typed when the TTL Line is HIGH, or +5 V and a different character will be typed when LOW, or 0 V. The two TTL line inputs from the rear 2.5mm connector are OR'd with TTL1 & TTL2 so can be used in tandem.

When a TTL input line, or pin, is HIGH, or +5 V, an associated key press is sent to your computer and a corresponding USB keyboard response will be received by your application. When a TTL input line is LOW, or 0 V, another associated key press will be sent.

In Mode 2, there are a bank of 16 separate characters representing the HIGH or LOW state of each TTL Line (8 x 2 for each TTL line state). As there is a change on a given line a character is typed and immediately released. That is, there is a KeyDown sent on one USB HID packet followed on the next packet by a KeyUp event. By default there are 1mS between USB packets so the fastest a single event can be reported as on is 2mS, i.e. KeyDown and KeyUp. If this level of switching is too fast and proves unreliable for your computer this can be made slower using the TTLtoKeys PC App. This is why this mode is described as being Qualitative as timings are not guaranteed to be millisecond accurate as opposed to Mode 1 which is and is Quantitative.

In Modes 2 & 3 you can also configure the delay between key presses and releases. By default this is set at 1 mS or 1 kHz. In addition the internal FIFO buffer itself is 125 characters long so if there are a lot of very fast line changes as the buffer fills up this may mean that typed output could be up to 125 mS later than the physical TTL line change.

Changes on the TTL Lines are reported in the order that the changes occur, i.e. they are buffered and stacked as they are typed out by the Module.

But if there were more than one change on the TTL lines then they would be typed in order and would not be completely in sync with the TTL line status at the millisecond accurate reporting level.

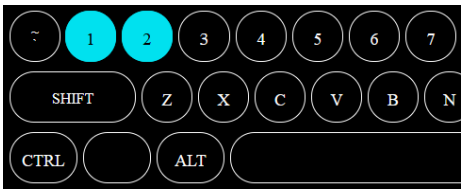
For example, if TTL1 goes HIGH, or +5 V, then a number "1" will be typed, i.e. "1" KeyDown on one USB HID packet and KeyUp on the next. So if there are no proceeding inputs on any TTL line then this might be considered as being millisecond accurate as the "1" KeyDown was millisecond accurate.

For example if TTL1 & TTL 2 were active HIGH, or +5 V, at the same time the following would be typed:

Characters Typed to PC	Keystroke sequence to produce key presses
12	• “1” KeyDown • “1” KeyUp • “2” KeyDown • “2” KeyUp

So typing “1” and “2” back to the PC would need to be done over 4 USB HID packets which would take 4 milliseconds even though the onset of TTL1 & TTL 2 were actually in sync on the TTL port a 1 mS. That is, they occurred simultaneously at the same time.

Contrast this with mode 1, where the status of both TTL1 & TTL2 would be sent in the same USB HID packet with 1mS resolution as shown:

Binary Line Input	Keystrokes Pressed, i.e. keys down
00000011 TTL1 = 1, Pin 2 TTL2 = 2, Pin 3	

In mode 2, you can imagine that if there were a lot of changes happening simultaneously or very quickly one after each other, then each change would take 2 mS to type, i.e. 2 x HID packets, one for a KeyDown and one for a KeyUp. For example, if all 8 TTL Lines became active HIGH, or +5 V, at the same time this would take 16mS to type over the keyboard HID despite the fact that all 8 TTL Lines became active at the same time, i.e. the first millisecond.

So TTL8 would only be flagged as being HIGH at 16mS after TTL1 was as that is the order the characters are typed back to the PC in.

A table with all possible key press values is shown below.

TTL Line Input	HIGH / +5V	LOW / 0V
TTL 1 / Pin 2	1	a
TTL 2 / Pin 3	2	b
TTL 3 / Pin 4	3	c
TTL 4 / Pin 5	4	d

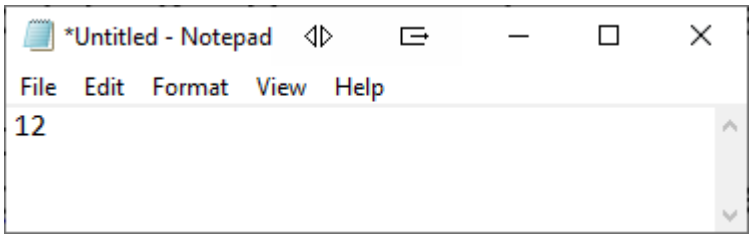
TTL Line Input	HIGH / +5V	LOW / 0V
TTL 1 / Pin 6	5	e
TTL 2 / Pin 7	6	f
TTL 3 / Pin 8	7	g
TTL 4 / Pin 9	8	h

By default the Module will check that your USB bus is free before attempting to send a data packet containing either the single KeyDown or KeyUp. The characters typed will be taken from the table above. If your computer conforms to the USB specifications it should accept this data every millisecond or at 1,000Hz.

Note: An 'x' will be typed by the module if for whatever reason it is unable to send this data packet to your computer at the industry standard rates.

As we need to detect key presses you will need to set up your Experiment Generator, or other software, to look for KeyPress events, i.e. pairs of KeyDowns and KeyUps.

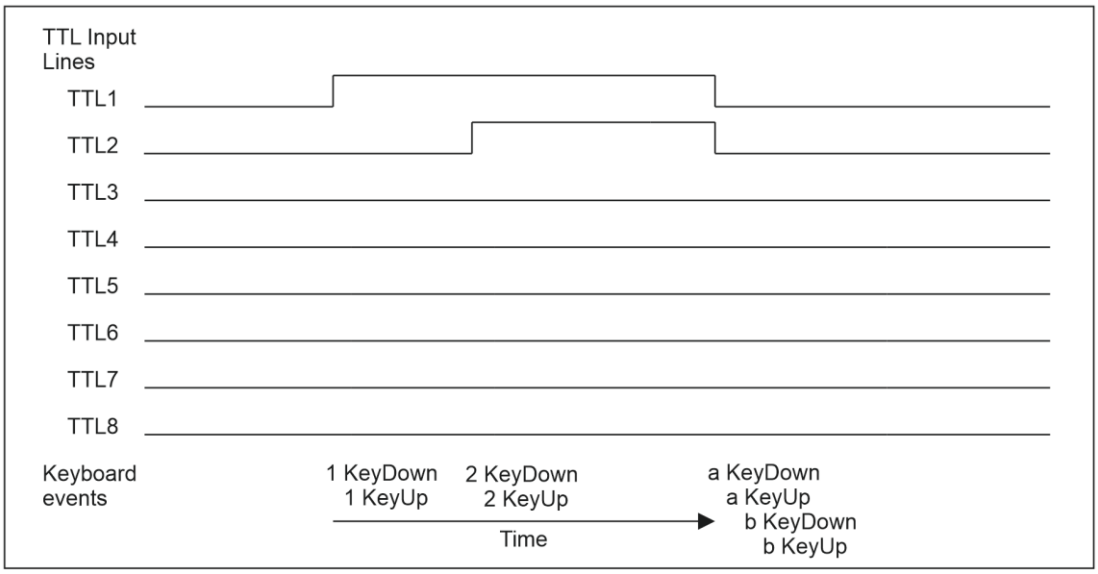
These represent individual TTL Lines going HIGH, or +5 V and LOW, 0 V, respectively. You can test that your Experiment Generator or other software is working correctly by using the keyboard you normally use for typing. For example, to simulate TTL Lines 1 & 2 (pins 2 & 3) being HIGH, or +5 V, simply press and release "1", then "2" as shown below.



In terms of a concrete example, imagine we have an Eye Tracker that signals via TTL1 that the participant gaze has entered an Area of Interest (AoI) and when they make a response on a response pad, the Eye Tracker signals the button onset and duration via TTL2.

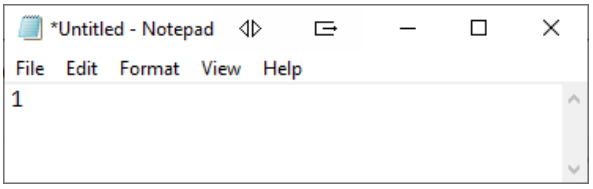
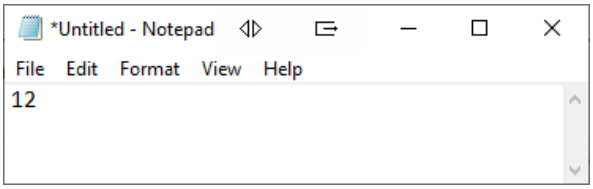
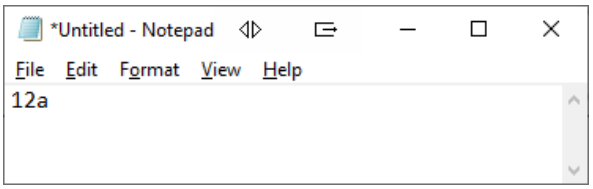
On a simulated TTL trace we can see that there is an "1" key press, which is actually an "1" KeyDown followed by an "1" KeyUp event pair, to indicate the onset of the visual stimulus on TTL1 (HIGH, or +5 V). After so many milliseconds there is a second "2" key press, or "2" KeyDown followed by a "2" KeyUp event, on TTL2 (HIGH, or +5 V) to indicate a button being pressed on a response pad. Remember these are the signals sent FROM your own equipment to the TTLtoKeys Module. Then when the response button is released TTL1 & 2 both go LOW, or 0 V, and the key presses "a" and "b" are typed in the order that the TTL lines have become inactive, i.e. "a" KeyDown, "a" KeyUp, "b" KeyDown and "b" KeyUp.

If you examine the x-axis this shows how the key presses might not match the exact time something happened on the TTL port. This is why you should not rely on the either Mode 2 or Mode 3 (Qualitative) to provide accurate timings via the keyboard HID. If you need to accurate timings via a keyboard HID interface you should use Mode 1 (Quantitative) exclusively.

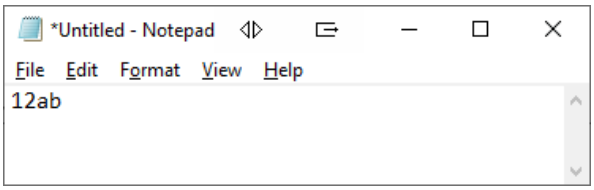


When using a text editor like Notepad, the TTL sequence above would appear as shown below.

TTL Line 1 (pin2) is active HIGH, or +5 V, then TTL Line 2 (pin 3) goes HIGH and finally both TTL Lines go LOW, or 0 V. The keyboard key presses indicate when TTL lines went active HIGH, or +5 V and the numbers represent when the TTL lines went LOW, 0 V.

Binary Line Input	Key presses, i.e. KeyDown | KeyUp pairs	Description
00000000	No keys are pressed	To begin with no keys are pressed as there is no activity on any of the TTL Lines.
00000001 TTL1 = 1 Pin 2		The “1” key is pressed, i.e. “1” KeyDown followed by “a” KeyUp as TTL1 is active HIGH, or +5 V. Each letter should be checked for as a KeyPress event in your Experiment Generator or other software.
00000011 TTL2 = 2 Pin 3		TTL2 goes to active HIGH, or +5 V along with TTL1. But as there already has been a key press for TTL1 now there is a “2” key press.
00000000		TTL1 & TTL2 go LOW, or 0 V, at the same time. An “a” key press is typed first, i.e. “a” KeyDown followed by an “a” KeyUp.

00000000



After the “a” key press has been typed the next key press is a “b”.

4.5 Mode 3: Port Value representing the Status of all TTL Lines in Either Hexadecimal or Decimal characters – OrangeLED / Qualitative

Ensure the TTLtoKeys Module is plugged into a USB port. Set the Module to Mode 3 using the mode button, as indicated by the indicator LED being solid orange. By default after first plug in the Module will be in mode 1 and the LED solid red.

Connect the TTL lines that you wish to monitor to the TTL inputs of the TTLtoKeys Module, either via the front 25-way D or rear 2.5 mm connector.

The 8 TTL input lines on the 25- way female D connector (TTL1 ~ TTL8) are sampled at 1 kHz, i.e. each millisecond or a 1,000 times a second. Each time there is a line change a Hexadecimal or Decimal TTL Port Value will be typed. This means that either two alphanumeric characters will be typed for Hexadecimal Port Values or three numeric values for Decimal Port Values. The two TTL line inputs from the rear 2.5 mm connector are OR'd with TTL1 & TTL2 so can be used in tandem.

Whether Hexadecimal or Decimal values are typed will depend on how you have configured the module in the PC application. By default Hex values are used. You may also have chosen for a CR, or Return, to be typed after Hex or Dec values have been typed.

In Mode 3, each time there is a change on any line either two hex or three decimal characters will be typed. These values represent each individual lines status across the port and range from 0~255, or all lines LOW (0 V) through to all lines HIGH (+5 V). That is, 00~ff for Hex or 000~255 for Dec. You can use a programmer's calculator to work out which individual lines are HIGH or LOW at the pin level or use the look-up table at the end of this section.

As there is a change on any line, two or three characters are typed as a series of key presses, i.e. KeyDown followed by KeyUp. That is, there is a KeyDown sent on one USB HID packet followed on the next packet by a KeyUp event. By default there are 1 mS between USB packets so the fastest a port change can be reported as is 4mS, i.e. KeyDown and KeyUp pairs repeated twice for Hexadecimal. For Decimal values this will be 6 mS as there is an extra character. In both modes, if you have chosen to have a CR, or Return, typed after the value this will add another pair of KeyDowns and KeyUps, or 2 mS. This is why this mode is described as being Qualitative as timings are not guaranteed to be millisecond accurate as opposed to Mode 1 which is and is Quantitative.

In Modes 2 & 3 you can also configure the delay between key presses and releases. By default this is set at 1 mS or 1 kHz. In addition the internal buffer itself is 125 characters long so if there are a lot of very fast line changes as the buffer fills up this may mean that typed output could be up to 125 mS later than the physical TTL line change. Changes on the TTL Lines are reported in the order that the changes occur, i.e. they are buffered and stacked as they are typed out by the Module.

Remember if there were a series of very fast changes on the TTL Lines they would be typed in order and would not be completely insync with the TTL line status at the millisecond accurate reporting level for the reason explained above, i.e. it takes time to type the TTL Port Value as a series of key presses. For example, if TTL1 goes HIGH, or +5 V, then the value "01" would be typed in Hex mode or "001" if running in Dec mode.

Mode	Characters Typed to PC	Keystroke sequence to produce key presses
Hex	01	• "0" KeyDown • "0" KeyUp • "1" KeyDown • "1" KeyUp
Decimal	001	• "0" KeyDown • "0" KeyUp • "0" KeyDown • "0" KeyUp • "1" KeyDown • "1" KeyUp

So typing "0" and "1" back to the PC in Hex would need to be done over 4 USB HID packets which would take 4 milliseconds and in Dec 6, USB HID packets which would take 6 milliseconds.

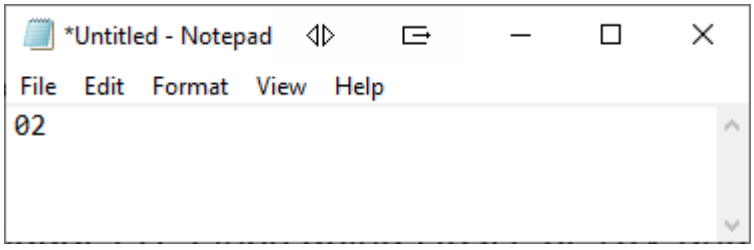
In mode 3, you can imagine that if there were a lot of changes happening very quickly one after each other there could be a lag when they were typed back to the PC. In reality there will always be a lag as you need to wait for the second or third character to be typed back before you can reliably determine the value of the TTL Port.

By default the Module will check that your USB bus is free before attempting to send a data packet containing either the single KeyDown or KeyUp. If your computer conforms to the USB specifications it should accept this data every millisecond or at 1,000Hz.

Note: An 'x' will be typed by the module if for whatever reason it is unable to send this data packet to your computer at the industry standard rates.

As you need to detect key presses you will need to set up your Experiment Generator, or other software, to look for either two characters being typed in Hexadecimal or three in Decimal.

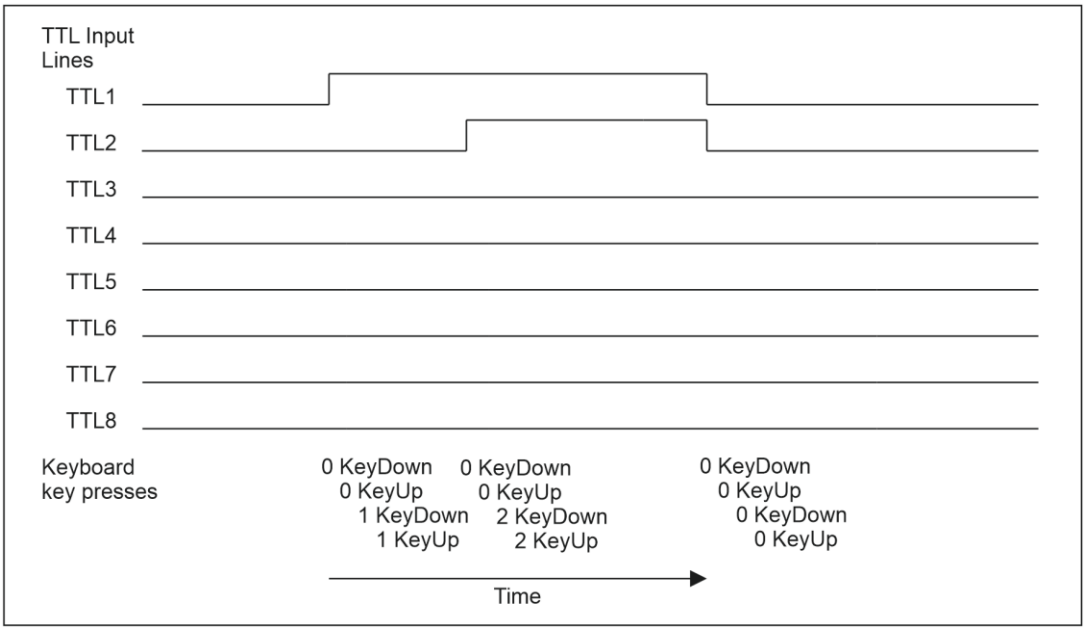
These represent individual TTL lines going HIGH, or +5 V and LOW, 0 V, respectively. You can test that your Experiment Generator or other software is working correctly by using the keyboard you normally use for typing. For example, to simulate TTL line 1 & 2 (pins 2 & 3) being HIGH, or +5 V, simply press and release "0", then "2" as shown below.



In terms of a concrete example, imagine we have an Eye Tracker that signals via TTL1 that the participant gaze has entered an Area of Interest (AoI) and when they make a response on a response pad, the Eye Tracker signals the button onset and duration via TTL2.

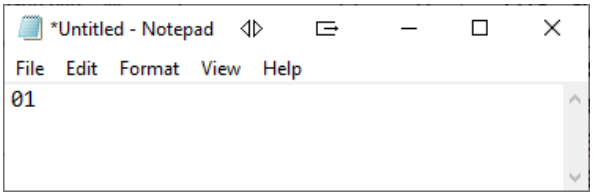
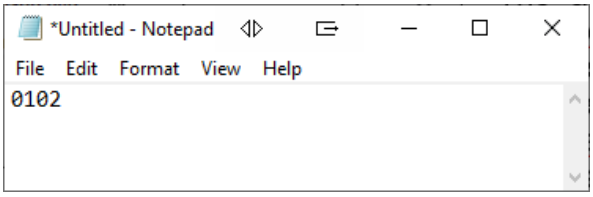
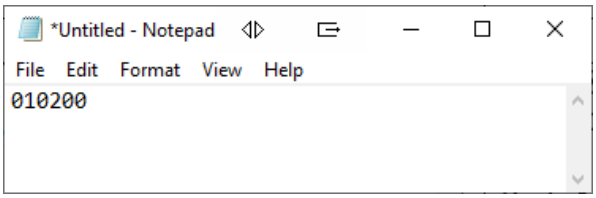
On a simulated TTL trace we can see that there are “01” key presses to indicate the onset of the visual stimulus on TTL1 (HIGH, or +5 V). After so many Milliseconds there is a second set of “02” key presses for TTL2 (HIGH, or +5 V) to indicate a button being pressed on a response pad. Remember these are the signals sent FROM your own equipment to the TTLtoKeys Module. Then when the response button is released TTL1 & 2 both go LOW, or 0V, and the key presses “00” are typed as all TTL Lines are inactive.

If you examine the x-axis this shows how the key presses might not match the exact time something happened on the TTL port. This is why you should not rely on the either Mode 2 or Mode 3 (Qualitative) to provide accurate timings via the keyboard HID. If you need to accurate timings via a keyboard HID interface you should use Mode 1 (Quantitative) exclusively.



When using a text editor like Notepad, the TTL sequence above would appear as shown below.

TTL line 1 (pin2) is active HIGH, or +5 V, then TTL Line 2 (pin 3) goes HIGH and finally both TTL Lines go LOW, or 0 V. The key presses represent the TTL Port Value at the time a sample was taken but you should remember that as this takes time to type back to the PC this will not be millisecond accurate as discussed above.

Binary Line Input	Keystrokes Pressed, i.e. KeyDown followed by KeyUp	Description
00000000	No keys are pressed	To begin with no keys are pressed as there is no activity on any of the TTL Lines.
00000001 TTL1 = 1 Pin 2		The keys “01” are pressed, i.e. “0” KeyDown, “0” KeyUp, “1” KeyDown, “1” KeyUp as TTL1 is active HIGH, or +5 V. Each number should be checked as a KeyPress event in your Experiment Generator or other software.
00000011 TTL2 = 2 Pin 3		TTL2 goes to active HIGH, or +5 V along with TTL1. The TTL Port Value, “02” is typed as in Hex that means TTL1 & TTL2 are HIGH, or +5 V, i.e. “0” KeyDown, “0” KeyUp, “2” KeyDown, “2” KeyUp.
00000000		TTL1 & TTL2 go LOW, or 0 V, at the same time and the TTL Port Value, “00” is typed as in Hex that means TTL1 & TTL2 are LOW, or 0V, i.e. “0” KeyDown, “0” KeyUp, “0” KeyDown, “0” KeyUp.

A table with all possible keystroke values in Hexadecimal and Decimal is shown below.

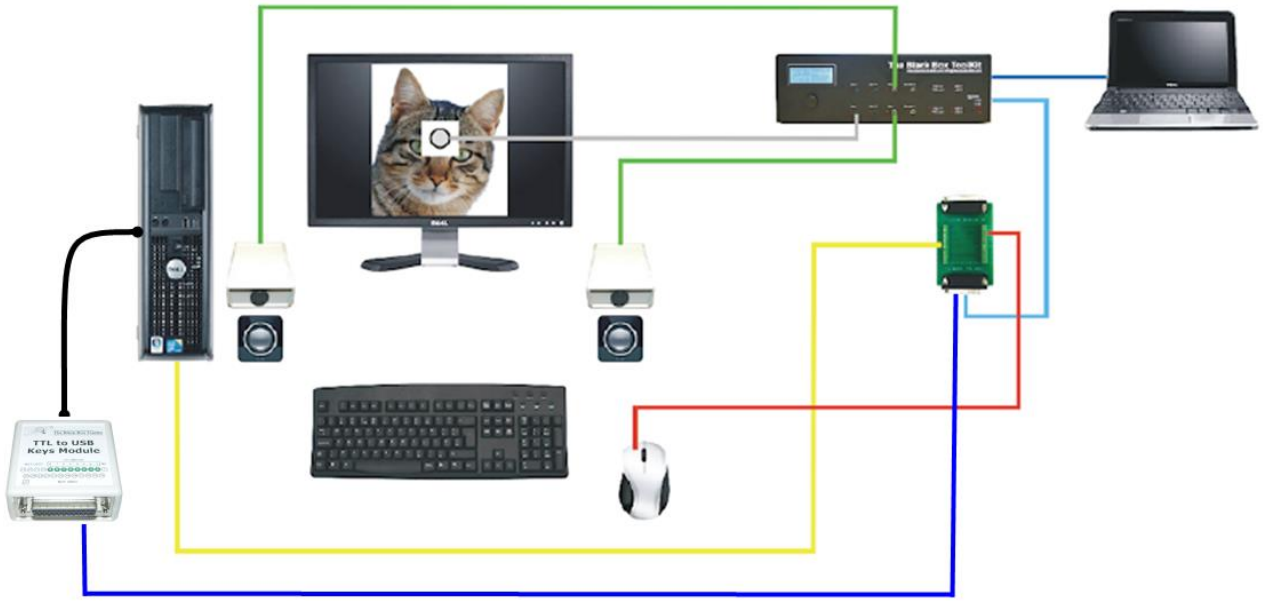
Binary Line Input	2 character Hex typed in Mode 3 (no CR or LF)	3 character Decimal typed in Mode 3 (no CR or LF)	Binary Line Input	2 character Hex typed in Mode 3 (no CR or LF)	3 character Decimal typed in Mode 3 (no CR or LF)	Binary Line Input	2 character Hex typed in Mode 3 (no CR or LF)	3 character Decimal typed in Mode 3 (no CR or LF)
00000000	00	000	00110000	30	048	01100000	60	096
00000001	01	001	00110001	31	049	01100001	61	097
00000010	02	002	00110010	32	050	01100010	62	098
00000011	03	003	00110011	33	051	01100011	63	099
00000100	04	004	00110100	34	052	01100100	64	100
00000101	05	005	00110101	35	053	01100101	65	101
00000110	06	006	00110110	36	054	01100110	66	102
00000111	07	007	00110111	37	055	01100111	67	103
00001000	08	008	00111000	38	056	01101000	68	104
00001001	09	009	00111001	39	057	01101001	69	105
00001010	0a	010	00111010	3a	058	01101010	6a	106
00001011	0b	011	00111011	3b	059	01101011	6b	107
00001100	0c	012	00111100	3c	060	01101100	6c	108
00001101	0d	013	00111101	3d	061	01101101	6d	109
00001110	0e	014	00111110	3e	062	01101110	6e	110
00001111	0f	015	00111111	3f	063	01101111	6f	111
00010000	10	016	01000000	40	064	01110000	70	112
00010001	11	017	01000001	41	065	01110001	71	113
00010010	12	018	01000010	42	066	01110010	72	114
00010011	13	019	01000011	43	067	01110011	73	115
00010100	14	020	01000100	44	068	01110100	74	116
00010101	15	021	01000101	45	069	01110101	75	117
00010110	16	022	01000110	46	070	01110110	76	118
00010111	17	023	01000111	47	071	01110111	77	119
00011000	18	024	01001000	48	072	01111000	78	120
00011001	19	025	01001001	49	073	01111001	79	121
00011010	1a	026	01001010	4a	074	01111010	7a	122
00011011	1b	027	01001011	4b	075	01111011	7b	123
00011100	1c	028	01001100	4c	076	01111100	7c	124
00011101	1d	029	01001101	4d	077	01111101	7d	125
00011110	1e	030	01001110	4e	078	01111110	7e	126
00011111	1f	031	01001111	4f	079	01111111	7f	127
00100000	20	032	01010000	50	080	10000000	80	128
00100001	21	033	01010001	51	081	10000001	81	129
00100010	22	034	01010010	52	082	10000010	82	130
00100011	23	035	01010011	53	083	10000011	83	131
00100100	24	036	01010100	54	084	10000100	84	132
00100101	25	037	01010101	55	085	10000101	85	133
00100110	26	038	01010110	56	086	10000110	86	134
00100111	27	039	01010111	57	087	10000111	87	135
00101000	28	040	01011000	58	088	10001000	88	136
00101001	29	041	01011001	59	089	10001001	89	137
00101010	2a	042	01011010	5a	090	10001010	8a	138
00101011	2b	043	01011011	5b	091	10001011	8b	139
00101100	2c	044	01011100	5c	092	10001011	8b	139
00101101	2d	045	01011101	5d	093	10001100	8c	140
00101110	2e	046	01011110	5e	094	10001101	8d	141
00101111	2f	047	01011111	5f	095	10001110	8e	142

Binary Line Input	2 character Hex typed in Mode 3 (no CR or LF)	3 character Decimal typed in Mode 3 (no CR or LF)	Binary Line Input	2 character Hex typed in Mode 3 (no CR or LF)	3 character Decimal typed in Mode 3 (no CR or LF)	Binary Line Input	2 character Hex typed in Mode 3 (no CR or LF)	3 character Decimal typed in Mode 3 (no CR or LF)
10001111	8f	143	10111110	be	190	11101110	ee	238
10010000	90	144	10111111	bf	191	11101111	ef	239
10010001	91	145	11000000	c0	192	11110000	f0	240
10010010	92	146	11000001	c1	193	11110001	f1	241
10010011	93	147	11000010	c2	194	11110010	f2	242
10010100	94	148	11000011	c3	195	11110011	f3	243
10010101	95	149	11000100	c4	196	11110100	f4	244
10010110	96	150	11000101	c5	197	11110101	f5	245
10010111	97	151	11000110	c6	198	11110110	f6	246
10011000	98	152	11000111	c7	199	11110111	f7	247
10011001	99	153	11001000	c8	200	11111000	f8	248
10011010	9a	154	11001001	c9	201	11111001	f9	249
10011011	9b	155	11001010	ca	202	11111010	fa	250
10011100	9c	156	11001011	cb	203	11111011	fb	251
10011101	9d	157	11001100	cc	204	11111100	fc	252
10011110	9e	158	11001101	cd	205	11111101	fd	253
10011111	9f	159	11001110	ce	206	11111110	fe	254
10100000	a0	160	11001111	cf	207	11111111	ff	255
10100001	a1	161	11010000	d0	208			
10100010	a2	162	11010001	d1	209			
10100011	a3	163	11010010	d2	210			
10100100	a4	164	11010011	d3	211			
10100101	a5	165	11010100	d4	212			
10100110	a6	166	11010101	d5	213			
10100111	a7	167	11010110	d6	214			
10101000	a8	168	11010111	d7	215			
10101001	a9	169	11011000	d8	216			
10101010	aa	170	11011001	d9	217			
10101010	aa	170	11011010	da	218			
10101011	ab	171	11011011	db	219			
10101100	ac	172	11011100	dc	220			
10101101	ad	173	11011101	dd	221			
10101110	ae	174	11011110	de	222			
10101111	af	175	11011111	df	223			
10110000	b0	176	11100000	e0	224			
10110001	b1	177	11100001	e1	225			
10110010	b2	178	11100010	e2	226			
10110011	b3	179	11100011	e3	227			
10110100	b4	180	11100100	e4	228			
10110101	b5	181	11100101	e5	229			
10110110	b6	182	11100110	e6	230			
10110111	b7	183	11100111	e7	231			
10111000	b8	184	11101000	e8	232			
10111001	b9	185	11101001	e9	233			
10111010	ba	186	11101010	ea	234			
10111011	bb	187	11101011	eb	235			
10111100	bc	188	11101100	ec	236			
10111101	bd	189	11101101	ed	237			

1. Using the TTLtoKeys Module with the Black Box ToolKit v2, mBBTK v2 (Event Marking Version) or BBTK USB TTL Module

5.1 Working With the Black Box ToolKit v2 to Test Timing Accuracy

The Black Box Toolkit v2 can be used to independently check the presentation, synchronization & response timing of your own hardware and software when running experiments. It can automatically respond to stimuli with sub-millisecond accuracy as though a human participant had made the response. This enables you to correct timing errors or tune your experiments timing accuracy.

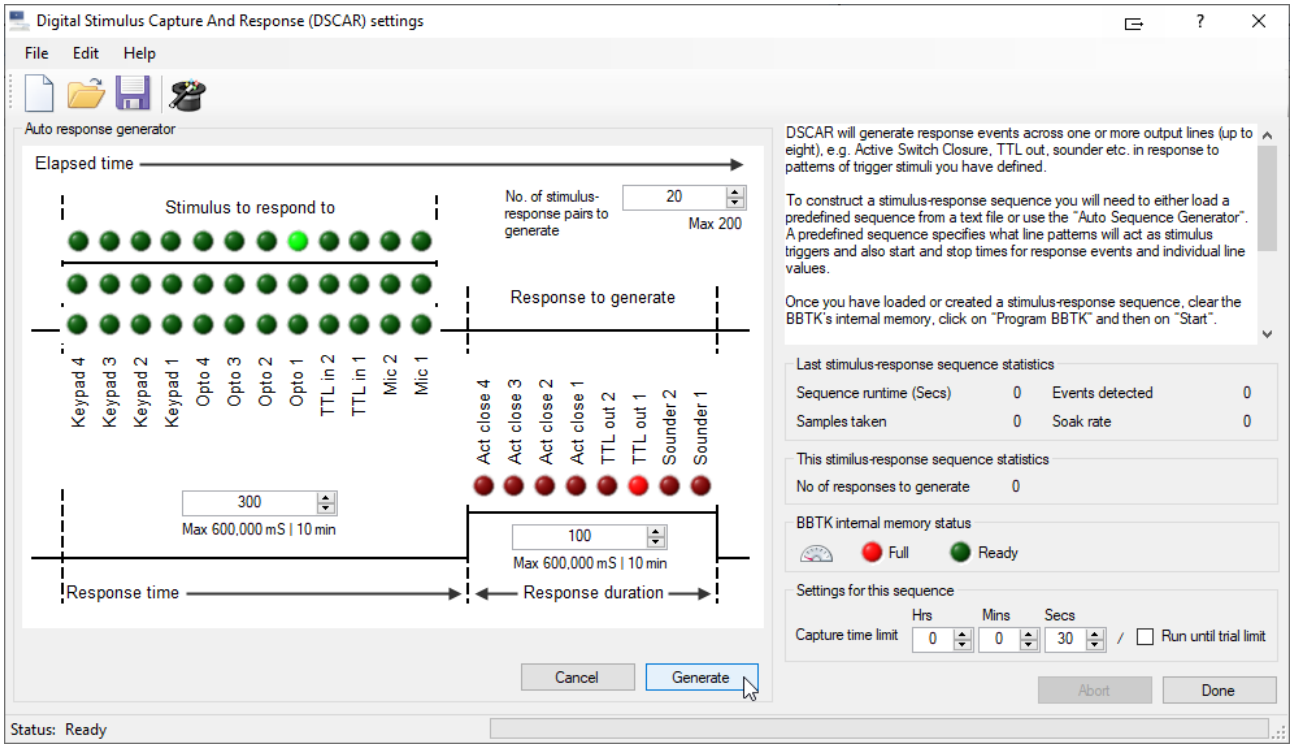


By using the TTLtoKeys module with a BBTK v2 this lets you use its TTL output lines to generate key presses, or keystrokes, as though they had been made by a human participant using a keyboard when making responses in your experiment.

For example, TTL Out 1 FROM the BBTK v2 could be wired TO input pin 2 of the TTLtoKeys Module. A GND from the BBTK v2 should be wired to pin 25 GND of the TTLtoKeys Module.

Once wired-up you could tell the BBTK v2 to look for a visual stimulus using an Opto-detector and a white/black marker. In response to trigger TTL Out 1 300 mS after detecting the stimulus and holding the TTL line HIGH for 100 mS. If the TTLtoKeys Module was in Mode 1 this should produce a 300 mS RT with a duration of 100 mS in your experiment. Simply, the “1” key will be pressed down at 300 mS and held down for 100 mS and then released.

This could be easily accomplished using the Digital Stimulus Capture And Response (DSCAR) functions of the BBTK v2 as shown below.

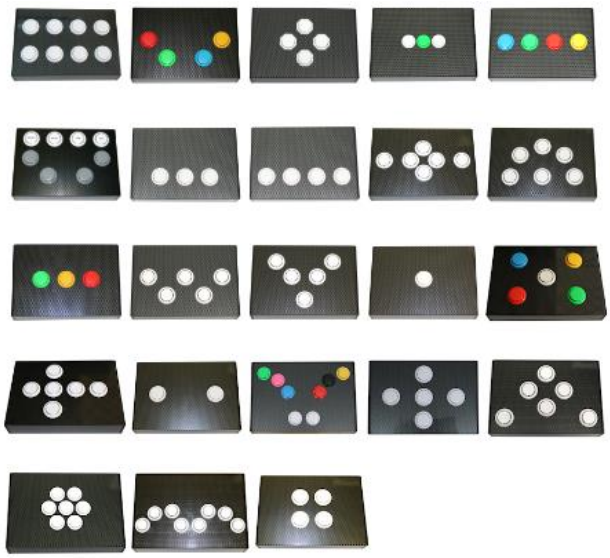


Once the sequence had been run you would then be able to evaluate the timing accuracy of your experiment.

This provides an alternative to using the BBTK Robotic Key Actuator (RKA) to physically press the keys on your own keyboard.

However, you should remember that the TTLtoKeys Module is not a substitute for your own keyboard as there is no guarantee that your own keyboard can respond with millisecond accuracy. In actual fact it is very unlikely to be that accurate and studies have shown that potentially it may take tens of milliseconds to respond and be extremely variable after a key has been pressed down.

Instead you should think of this setup as letting you mirror what would happen with regards to timing accuracy if you used on of our USB Response Pads as a drop in replacement.

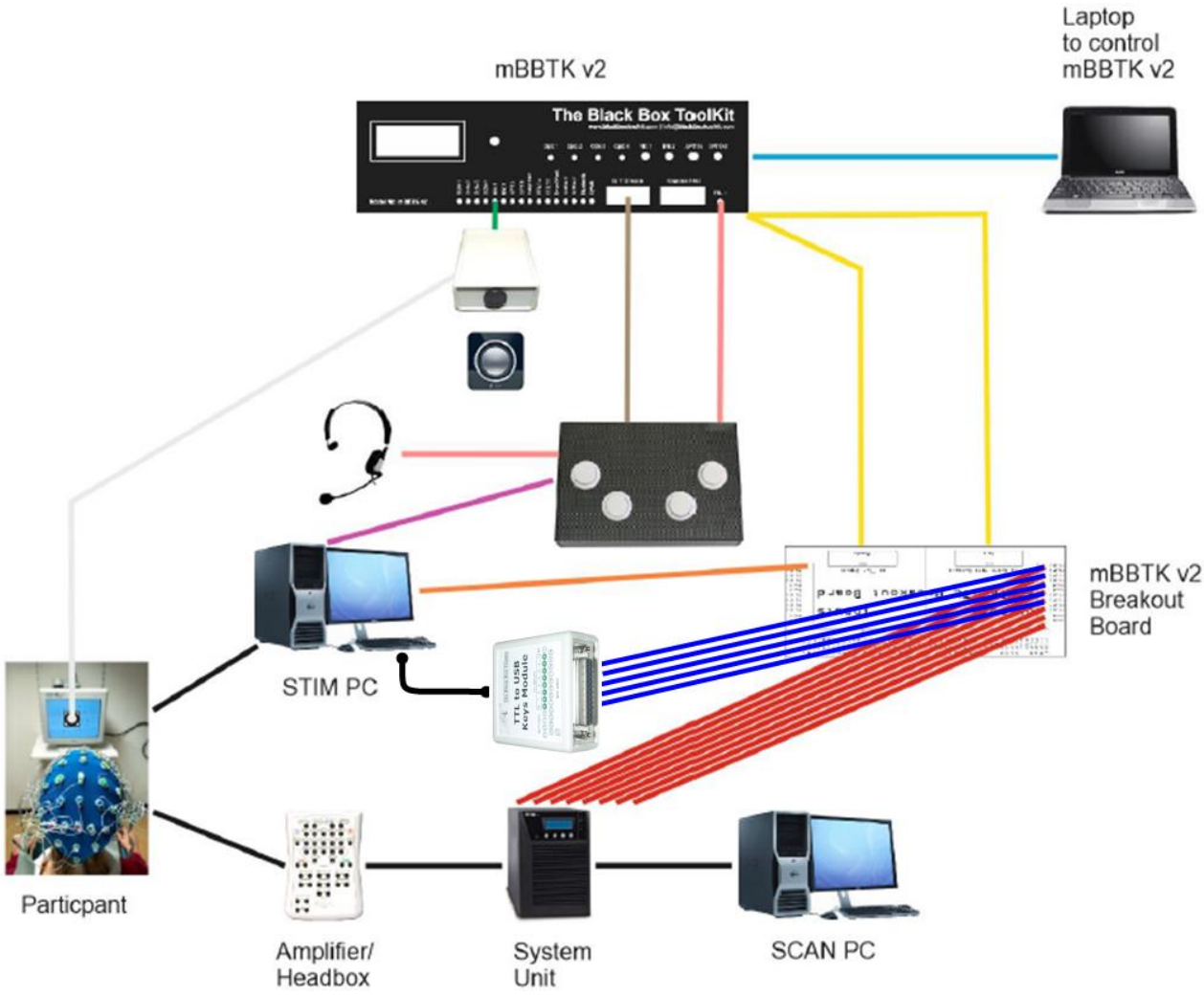


For more information on the BBTK v2 and our USB Response Pads you should visit out website, www.blackboxtoolkit.com.

5.2 Working With the Black Box ToolKit mBBTK to Event Mark Using Software That Cannot Normally Accept TTL Inputs

The mBBTK v2 is an easy to use fit and forget solution for TTL event marking/TTL triggering in psychology experiments that use EEG, fMRI, Eye Tracking or any study where you need to event mark with sub-millisecond accuracy. The mBBTK v2 (event marking version) is designed to act as your eyes and ears and event mark exactly when a stimulus appears, a sound is made or a response button is pressed.

With up to 48 input channels there is a sensor to cover every type of stimulus. Any input channel or sensor can quickly be paired with any of the 24 TTL event marking/TTL trigger output lines using our PC Software. What's more all event marking data is simultaneously recorded in real time for later analysis.

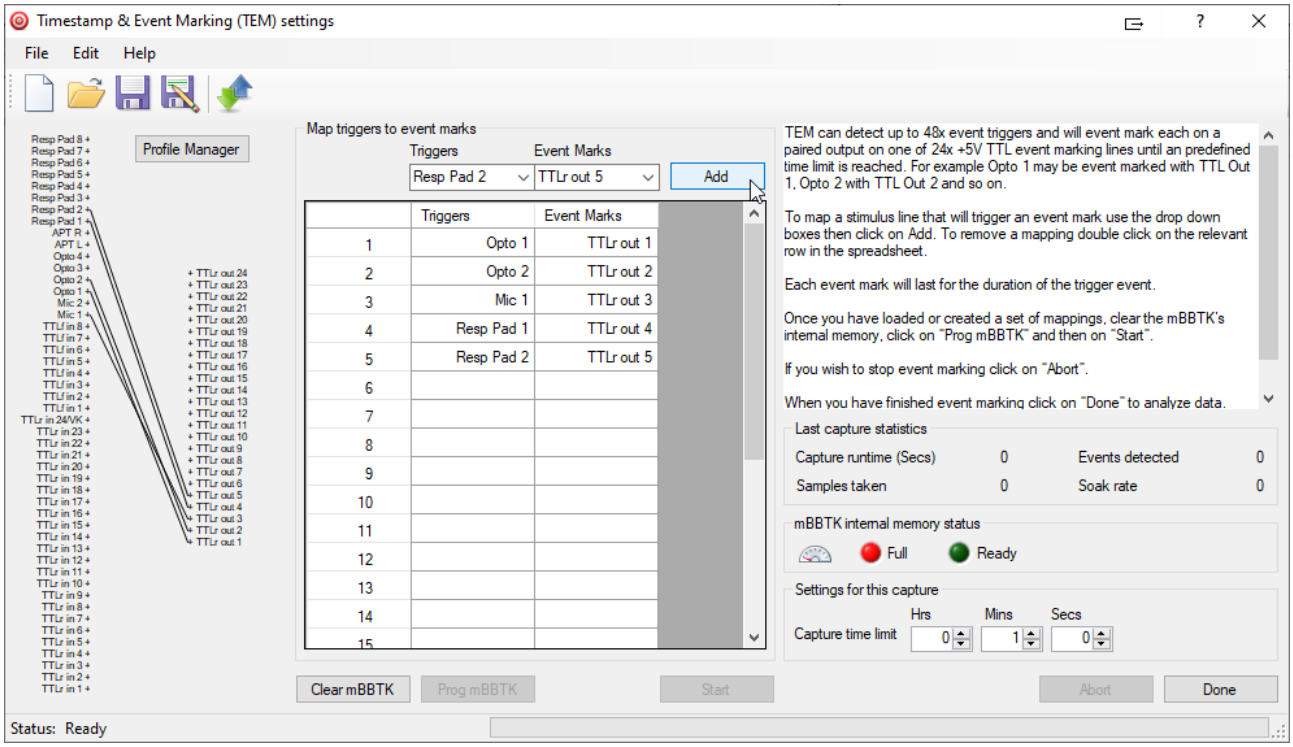


By using the TTLtoKeys module with a mBBTK v2 this lets you use its TTL output lines to generate key presses, or keystrokes, in lieu of being able to accept TTL event marking triggers into a platform incapable of receiving them directly due to it being sandboxed, e.g. a web based experiment generator.

For example, the first 8 TTL Out lines FROM the mBBTK could be wired TO input pins 2~9 of the TTLtoKeys Module. A GND from the mBBTK should be wired to pin 25 GND of the TTLtoKeys Module.

Once wired-up you could tell the mBBTK to look for various events as a trigger and pair them with TTLr outs 1-8. Thus in Mode 1 each TTL event mark or trigger would trigger the TTLtoKeys Module and cause a key to be held down for the duration of the event mark. That is, where up to 8 keys are held down in one USB HID packet to represent the state of all 8 TTL input lines.

This could be easily accomplished using the Timestamp & Event Marking (TEM) functions of the mBBTK v2 as shown below. In this example one visual stimulus under Opto-detector 1 would trigger TTL 1 and make the TTLtoKeys Module hold down the “1” key for the duration the stimulus was shown. A “2” KeyDown for a visual stimulus at another location, a “3” for a sound and “4” and a “5” KeyDowns for either of two response pad buttons pressed.



mBBTK	mBBTK Output Line	TTLtoKeys Module Key
Opto 1	TTLr out 1	“1”
Opto 2	TTLr out 2	“2”
Mic 1	TTLr out 3	“3”
Resp Pad 1	TTLr out 4	“4”
Resp Pad 2	TTLr out 5	“5”

Once the sequence had been run you would then have a fully auditable set of timings for events and triggers the mBBTK sent and also have a set of keyboard key based event marks in your own experiment.

5.3 Working With the Black Box ToolKit USB TTL Module to Event Mark as Keystrokes

Designed as a parallel port replacement for simple event marking the BBTK USB TTL Module converts serial output into TTL event marks. When plugged into a USB port on a PC our USB TTL Module appears as a Virtual Com Port (VCP). To event mark you would send two hex bytes from an experiment generator or other software.

Our TTL Module allows you to quickly and easily TTL event mark/TTL trigger with any experiment generator via standard one line serial commands, e.g. task events in E-Prime. Works out of the box with E-Prime, SuperLab, Presentation, Inquisit, PsychoPy or any other software that can write to a standard serial port.

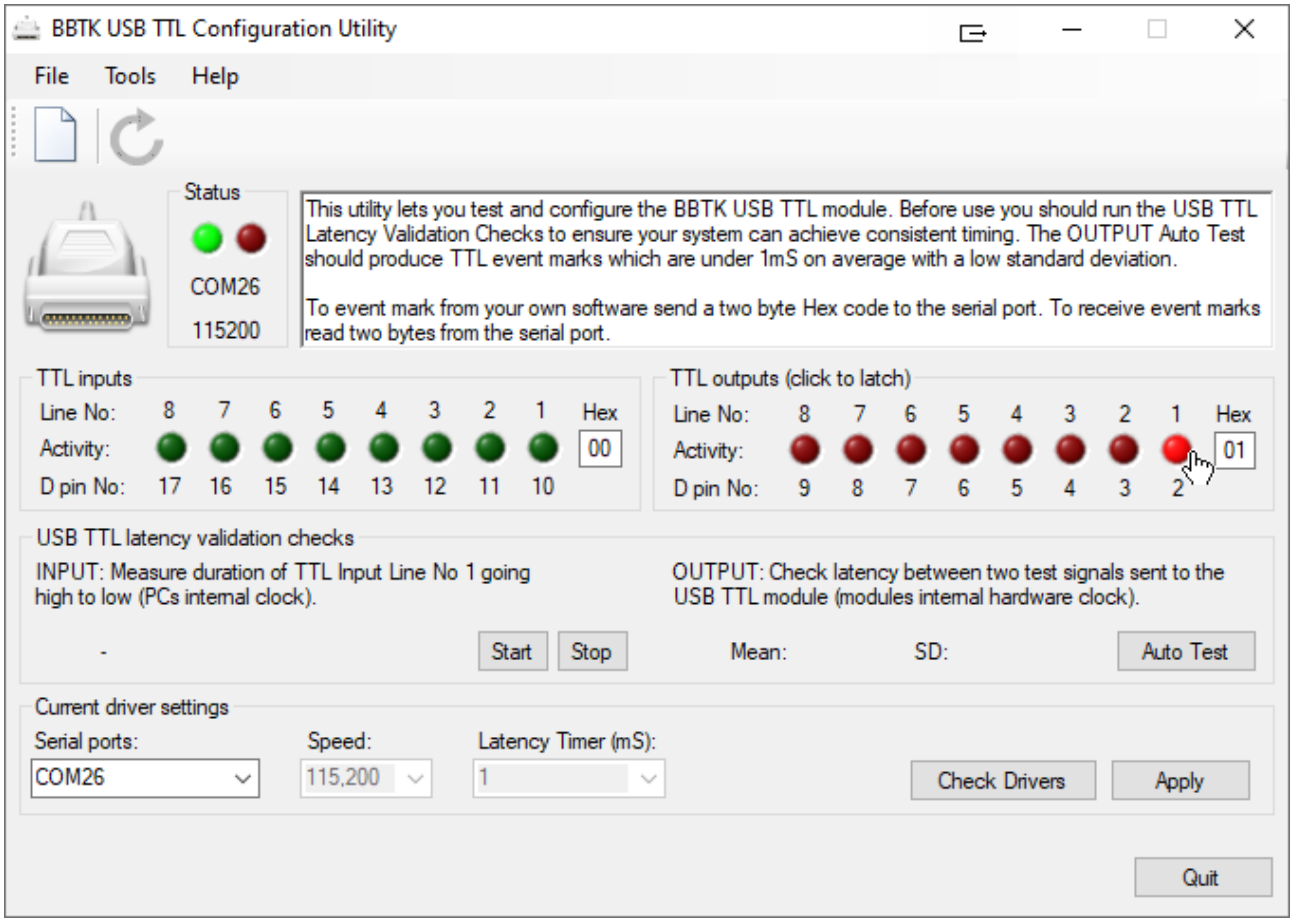
In common with the TTLtoKeys Module our USB TTL Module also has 8 lines which means that they can be connected together easily. That is, TTL Module output pins 2~9 (red) can be connected directly to TTLtoKeys input pins 2~9 (green).

Serial Command in Hex	USB TTL Module	Pins	TTLtoKeys Module	Keystroke Produced (Mode 1)
“01”		2 → 2		“1”
“02”		3 → 3		“2”
“04”		4 → 4		“3”
“08”		5 → 5		“4”
“10”		6 → 6		“5”
“20”		7 → 7		“6”
“40”		8 → 8		SHIFT
“80”		9 → 9		CTRL
		25 → 25		

When in Mode 1 each time a TTL line was HIGH, or 5 V, on the TTL Module the corresponding keys would be held down on the TTLtoKeys Module. When LOW, or 0 V, they would be released.

You can quite easily see that if the TTLtoKeys Module was switched to working on Mode 3 and Hex was typed it would mirror the value of the TTL Import Port, i.e. the same Hex value used to trigger the TTL Module initially would be typed!

The BBTK USB TTL Configuration Utility shows how you could send TTL triggers into the TTLtoKeys Module as a test.



6. Tips for Using the TTLtoKeys Module With Your Own Hardware and Experiment Generators

6.1 Using the TTLtoKeys Module With Your own Hardware

Using the TTLtoKeys Module with hardware capable out outputting TTL of between 3.3~5 V is generally straightforward. You should look for ports on your own hardware labelled “Trigger Out”, “Digital Out” or “TTL Out” for example. The pintout for these should be detailed in your devices manual. They should switch to a positive voltage when active or HIGH and 0 V when inactive or LOW.

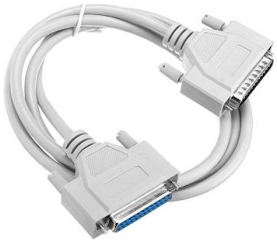
The easiest way to connect your equipment to the TTLtoKeys Module is to order a custom made cable from the Black Box ToolKit when you purchase your Module. This gives you a cable with one end that terminates in a male 25-way D which plugs into the Module and the other end which plugs into whatever format port on your own equipment, e.g. 9-way D, DIN socket, BNC etc.

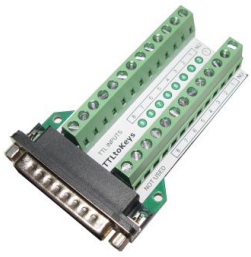
Alternatively you can make use of the breakout board to wire your own cable temporarily. Pins 2~9 on the 25-way D correspond to TTL input lines 1~8. Pin 25 is ground or GND.

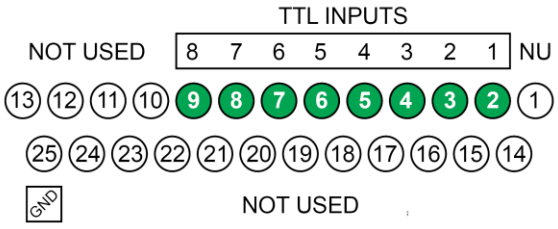
The 2.5 mm stereo socket on the rear of the Module mirrors TTL1 & TTL 2 on the front 25-way D (tip & ring respectively, sleeve GND).

To check your equipment is connected correctly and to ensure you can output TTL singals as you expect you should refer to sections “3.3 Carrying out a Basic Functional Check in a Text Editor” and “4.1 Worked Examples Using the TTL to USB Keys Module in Each Mode” of this manual.

WARNING: The Black Box ToolKit Ltd cannot be held liable for any damage caused to devices through use of the TTLtoKeys Module howsoever caused. By using the TTLtoKeys Module you accept sole responsibility for any subsequent damage. The end user should evaluate fitness for purpose prior to using the TTLtoKeys Module with any given device. The Module is designed for use in research environments.







The front of the module has a 25-way female D connector.

Pins 2~9 correspond to TTL input lines 1~8. Pin 25 is ground.

The rear of the module has standard USB type B connector by which it is connected to the PC which will receive the USB HID keyboard key presses and keystrokes.

A 2.5 mm stereo socket mirrors TTL1 & TTL 2 on the front 25-way D (tip & ring respectively, sleeve GND).

A multi-color LED indicates the operating mode of the Module. To switch mode simply press the button once.

Physical connection:

- Use a USB port which is directly connected to the motherboard
- Try and always use the same USB port
- Do not use USB hubs or switches to connect the Module
- Do not use high bandwidth USB devices together with the Module and do not leave them connected to any USB ports on your system, e.g. USB memory sticks, external hard drives etc.
- Use the supplied high quality USB cable
- Remove any TTL connections from the module when not in use

6.2 What Keys to Check for When Using an Experiment Generator or Other Software

Using the TTLtoKeys Module should be simple as you only need to check for either key presses or keystrokes, i.e. pairs of KeyDowns followed by a KeyUps in modes 2 & 3, or key strokes in mode 1, i.e. KeyDown for each individual TTL Line.

In short you can use your normal keyboard to check whether your software is working correctly. In Mode 1 for example, press and hold a key the USBtoKeys Module uses and then release. Your software should respond to the KeyDowns and also when keys go up within a single USB HID packet.

TTL Input Line	1	2	3	4	5	6	7	8
25-way D Pin	2	3	4	5	6	7	8	9
Key	1	2	3	4	5	6	SHIFT	CTRL

Using the TTLtoKeys keyboard checking tool this is easy to accomplish.

www.blackboxtoolkit.com/tools/ttltokeys/linechecker.html

Note: You should not use a text editor to check this as you will not be able to see the state of each keystroke, or TTL Line, in the output as you can with a keyboard key checker such as the one shown below.

In Modes 2 & 3 simply press the keys the Module uses.

In Mode 2 for example, press and release each key immediately.

TTL Line Input	HIGH / +5V	LOW / 0V
TTL 1 / Pin 2	1	a
TTL 2 / Pin 3	2	b
TTL 3 / Pin 4	3	c
TTL 4 / Pin 5	4	d

TTL Line Input	HIGH / +5V	LOW / 0V
TTL 1 / Pin 6	5	e
TTL 2 / Pin 7	6	f
TTL 3 / Pin 8	7	g
TTL 4 / Pin 9	8	h

In Mode 3, type either to Hex characters, i.e. press and release keys 00~ff, or in decimal 000~255, as though you are typing them.

Hex	Dec
0~ff	000~255

Please note that the Black Box ToolKit Ltd cannot provide support for third party software. Example scripts can be downloaded from our support website but are provided on an "as is" basis.

7. Technical Specifications

7.1 Hardware Specifications

- Dimensions (W x H x D): 67.1 x 28.2 x 67.1 mm
- Weight: approx 100 g
- Shipping carton dimensions (W x H x D): 100 x 60 x 80 mm
- Shipping carton weight: approx 200 g
- 28-Pin, High-Performance, Enhanced Flash, USB 2.0 Microcontroller with nanoWatt Technology
- Single chip USB to asynchronous serial data transfer interface
- USB 2.0 Full Speed compatible
- Low USB bandwidth consumption
- Operating Temperature: 0 °C to 50 °C
- Operating Voltage Range: 3.3 V to 5.25 V
- High-Current Sink/Source: 25 mA/25 mA
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory Typical for firmware updates
- Flash/Data EEPROM Retention: > 40 Years

- 10 Digital +5 V TTL Lines
    - 1x 25-way female D connector for connecting 8x TTL output lines from your equipment (pins 2~9). Plus 1x GND pin (pin 25)
    - 1x 2.5mm stereo socket for connecting 2x TTL output lines from your equipment. Plus 1x GND (tip pin 1, ring pin 2, sleeve GND)
- Accessed via a standard USB Keyboard HID
- TTL Input Lines configured as an 8 bit port
- Change detection on TTL Input Lines
- TTL Input converted to key presses or keystrokes that represent either individual line status or Port Value for all TTL Lines (255 possible states):
    - Mode 1: Unique KeyDown in single USB Keyboard HID packet for each TTL Line (1mS transmit latency[#])
    - Mode 2: Unique KeyPress (Down & Up) for each TTL line (2mS transmit latency[#])
    - Mode 3: Port Value representing the stats of all TTL Lines in a either Hex or Decimal characters (4 or 6mS transmit latency depending on number system used[#])

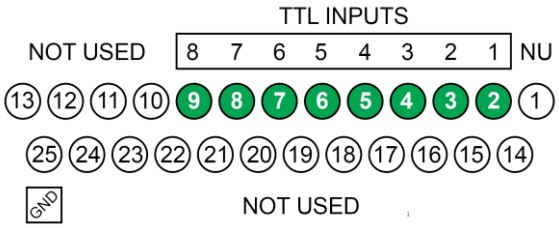
- Custom designed firmware
    - Checks for a TTL event mark or trigger event 1,000 times a second (1 kHz sampling rate)
    - Fastest time to transmit as TTL Line change as a keyboard event: 1mS in Mode 1[#]
    - Supplied with configuration and latency validation software for Windows XP, Vista, Win 7, Win 8/8.1 and Win 10

#Dependent on specific USB subsystem and using the USB Configuration utility to check correct for correct installation.

- Moulded grey ABS plastic enclosure
- Stainless Steel front and rear panels with DP1 finish

- Custom designed PCB
    - Material: FR4
    - Solder Resist Coating: CAWN_1331 green resist and CAWN_1291 green hardener
- Lead free solder: MULTICORE / LOCTITE - 96SC 400 5C 1.00MM - SOLDER WIRE (Henkel 60EN 362 5C)

7.2 TTLtoKeys Module Pinouts & Connectors



The front of the module has a 25-way female D connector. Pins 2~9 correspond to TTL input lines 1~8. Pin 25 is ground.

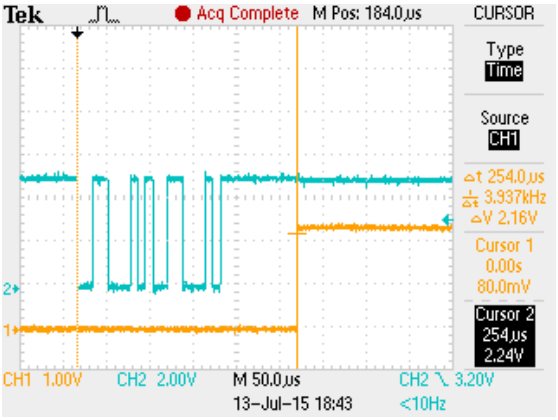
The rear of the module has standard USB type B connector by which it is connected to the PC which will receive the USB HID keyboard key presses and keystrokes. A 2.5 mm stereo socket mirrors TTL1 & TTL 2 on the front 25-way D (tip & ring respectively, sleeve GND).

A multi-color LED indicates the operating mode of the Module. To switch mode simply press the button once.

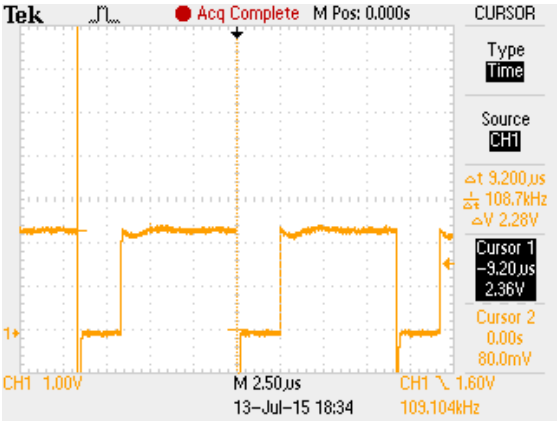
7.3 Timing Specifications - TBD

- Checks for a TTL event mark or trigger event, every millisecond, or 1,000 times a second (1 kHz sampling rate)*
- Typical time* to send a single TTL Line change to PC as a key press or key stroke in each operating mode:
  - Mode 1: 1 mS
  - Mode 2: 2 mS
  - Mode 3 Hex: 4 mS / Mode 3 Dec: 6 mS
- Supplied with configuration software for Windows XP, Vista, Win 7, Win 8/8.1 and Win 10

*Dependent on specific USB subsystem and how configured via configuration utility.



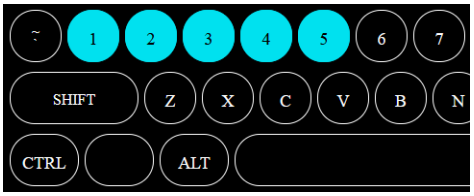
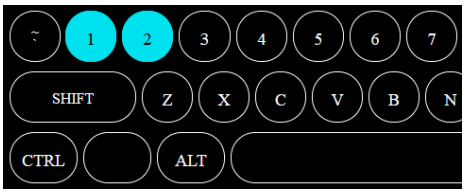
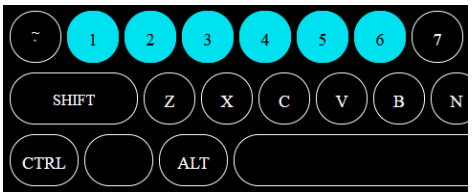
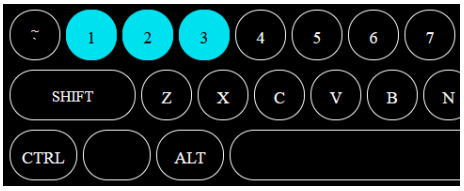
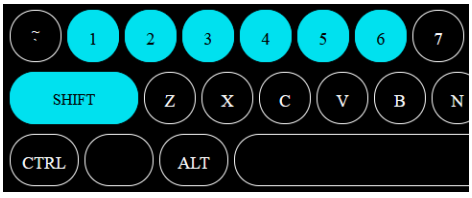
Typical time to detect a TTL input or event mark (orange) to sending out a KeyDown HID keystroke in Mode 1



TTL sampling rate, i.e. how many times a second TTL I/O status is checked

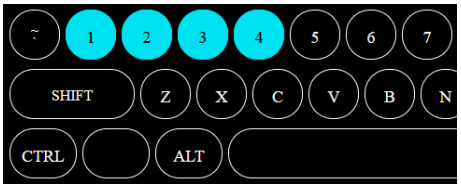
Appendix A:TTL Line decoding for Each Operating Mode

Mode 1: Keystrokes in one USB HID Packet

Binary Line Input	Keystrokes Pressed, i.e. keys down	Binary Line Input	Keystrokes Pressed, i.e. keys down
00000000 No Keystroke			
00000001 TTL1 = 1 Pin 2		00011111 TTL5 = 5 Pin 6	
00000011 TTL2 = 2 Pin 3		00111111 TTL6 = 6 Pin 7	
00000111 TTL3 = 3 Pin 4		01111111 TTL7 = SHIFT Pin 8	

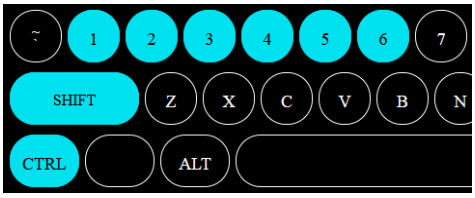
00001111

TTL4 = 4
Pin 5



11111111

TTL8 =
CTRL
Pin 9



Mode 2: Key Presses Typed as Each Line Changes (Down and up for Each Letter or Number)

TTL Line Input	HIGH / +5V	LOW / 0V
No TTL Input		0
TTL 1 / Pin 2	1	a
TTL 2 / Pin 3	2	b
TTL 3 / Pin 4	3	c
TTL 4 / Pin 5	4	d

TTL Line Input	HIGH / +5V	LOW / 0V
TTL 1 / Pin 6	5	e
TTL 2 / Pin 7	6	f
TTL 3 / Pin 8	7	g
TTL 4 / Pin 9	8	h

Mode 3: Hex or Decimal TTL Port Value

Binary Line Input	2 character Hex typed in Mode 3 (no CR or LF)	3 character Decimal typed in Mode 3 (no CR or LF)	Binary Line Input	2 character Hex typed in Mode 3 (no CR or LF)	3 character Decimal typed in Mode 3 (no CR or LF)	Binary Line Input	2 character Hex typed in Mode 3 (no CR or LF)	3 character Decimal typed in Mode 3 (no CR or LF)
00000000	00	000	00110000	30	048	01100000	60	096
00000001	01	001	00110001	31	049	01100001	61	097
00000010	02	002	00110010	32	050	01100010	62	098
00000011	03	003	00110011	33	051	01100011	63	099
00000100	04	004	00110100	34	052	01100100	64	100
00000101	05	005	00110101	35	053	01100101	65	101
00000110	06	006	00110110	36	054	01100110	66	102
00000111	07	007	00110111	37	055	01100111	67	103
00001000	08	008	00111000	38	056	01101000	68	104
00001001	09	009	00111001	39	057	01101001	69	105
00001010	0a	010	00111010	3a	058	01101010	6a	106
00001011	0b	011	00111011	3b	059	01101011	6b	107
00001100	0c	012	00111100	3c	060	01101100	6c	108
00001101	0d	013	00111101	3d	061	01101101	6d	109
00001110	0e	014	00111110	3e	062	01101110	6e	110
00001111	0f	015	00111111	3f	063	01101111	6f	111
00010000	10	016	01000000	40	064	01110000	70	112
00010001	11	017	01000001	41	065	01110001	71	113
00010010	12	018	01000010	42	066	01110010	72	114
00010011	13	019	01000011	43	067	01110011	73	115
00010100	14	020	01000100	44	068	01110100	74	116
00010101	15	021	01000101	45	069	01110101	75	117
00010110	16	022	01000110	46	070	01110110	76	118
00010111	17	023	01000111	47	071	01110111	77	119
00011000	18	024	01001000	48	072	01111000	78	120
00011001	19	025	01001001	49	073	01111001	79	121
00011010	1a	026	01001010	4a	074	01111010	7a	122
00011011	1b	027	01001011	4b	075	01111011	7b	123
00011100	1c	028	01001100	4c	076	01111100	7c	124
00011101	1d	029	01001101	4d	077	01111101	7d	125
00011110	1e	030	01001110	4e	078	01111110	7e	126
00011111	1f	031	01001111	4f	079	01111111	7f	127
00100000	20	032	01010000	50	080	10000000	80	128
00100001	21	033	01010001	51	081	10000001	81	129
00100010	22	034	01010010	52	082	10000010	82	130
00100011	23	035	01010011	53	083	10000011	83	131
00100100	24	036	01010100	54	084	10000100	84	132
00100101	25	037	01010101	55	085	10000101	85	133
00100110	26	038	01010110	56	086	10000110	86	134
00100111	27	039	01010111	57	087	10000111	87	135
00101000	28	040	01011000	58	088	10001000	88	136
00101001	29	041	01011001	59	089	10001001	89	137
00101010	2a	042	01011010	5a	090	10001010	8a	138
00101011	2b	043	01011011	5b	091	10001011	8b	139
00101100	2c	044	01011100	5c	092	10001011	8b	139
00101101	2d	045	01011101	5d	093	10001100	8c	140
00101110	2e	046	01011110	5e	094	10001101	8d	141
00101111	2f	047	01011111	5f	095	10001110	8e	142

Binary Line Input	2 character Hex typed in Mode 3 (no CR or LF)	3 character Decimal typed in Mode 3 (no CR or LF)	Binary Line Input	2 character Hex typed in Mode 3 (no CR or LF)	3 character Decimal typed in Mode 3 (no CR or LF)	Binary Line Input	2 character Hex typed in Mode 3 (no CR or LF)	3 character Decimal typed in Mode 3 (no CR or LF)
10001111	8f	143	10111110	be	190	11101110	ee	238
10010000	90	144	10111111	bf	191	11101111	ef	239
10010001	91	145	11000000	c0	192	11110000	f0	240
10010010	92	146	11000001	c1	193	11110001	f1	241
10010011	93	147	11000010	c2	194	11110010	f2	242
10010100	94	148	11000011	c3	195	11110011	f3	243
10010101	95	149	11000100	c4	196	11110100	f4	244
10010110	96	150	11000101	c5	197	11110101	f5	245
10010111	97	151	11000110	c6	198	11110110	f6	246
10011000	98	152	11000111	c7	199	11110111	f7	247
10011001	99	153	11001000	c8	200	11111000	f8	248
10011010	9a	154	11001001	c9	201	11111001	f9	249
10011011	9b	155	11001010	ca	202	11111010	fa	250
10011100	9c	156	11001011	cb	203	11111011	fb	251
10011101	9d	157	11001100	cc	204	11111100	fc	252
10011110	9e	158	11001101	cd	205	11111101	fd	253
10011111	9f	159	11001110	ce	206	11111110	fe	254
10100000	a0	160	11001111	cf	207	11111111	ff	255
10100001	a1	161	11010000	d0	208			
10100010	a2	162	11010001	d1	209			
10100011	a3	163	11010010	d2	210			
10100100	a4	164	11010011	d3	211			
10100101	a5	165	11010100	d4	212			
10100110	a6	166	11010101	d5	213			
10100111	a7	167	11010110	d6	214			
10101000	a8	168	11010111	d7	215			
10101001	a9	169	11011000	d8	216			
10101010	aa	170	11011001	d9	217			
10101010	aa	170	11011010	da	218			
10101011	ab	171	11011011	db	219			
10101100	ac	172	11011100	dc	220			
10101101	ad	173	11011101	dd	221			
10101110	ae	174	11011110	de	222			
10101111	af	175	11011111	df	223			
10110000	b0	176	11100000	e0	224			
10110001	b1	177	11100001	e1	225			
10110010	b2	178	11100010	e2	226			
10110011	b3	179	11100011	e3	227			
10110100	b4	180	11100100	e4	228			
10110101	b5	181	11100101	e5	229			
10110110	b6	182	11100110	e6	230			
10110111	b7	183	11100111	e7	231			
10111000	b8	184	11101000	e8	232			
10111001	b9	185	11101001	e9	233			
10111010	ba	186	11101010	ea	234			
10111011	bb	187	11101011	eb	235			
10111100	bc	188	11101100	ec	236			
10111101	bd	189	11101101	ed	237			