Starting processes with a single push

User manual for programmable pushbuttons and pushboxes

Version 1, updated in: 09/2019

Imprint and contact

Grau GmbH Hardware & Software Solutions

Riemekestraße 11

33102 Paderborn

Germany

Web: www.grauonline.de

Table of contents

1. About this user manual

Please read this user manual thoroughly before you install, customize and use our hardware and software for USB pushbuttons and USB pushboxes. Our programmable pushbuttons and pushboxes are USB modules which serve as a substitution for mouse and keyboard if you want to initiate technical processes. Such USB modules are applied for a range of user scenarios, for presentations and trade shows, in museums, in photo studios and photo boxes, in a gaming environment and in industrial settings.

When you read this manual, please consider the following graphical markers:

	Safety instructions which are placed in relevant chapters of the user manual
	Tip for a successful and easy use of software and hardware
Ctrl	Bold and cursive font style in the context of commands, buttons and input fields concerning software and hardware

Contents and target groups of the user manual

The user manual addresses all user groups who want to start technical processes with a simple push on a button instead of entering hotkeys or clicking with a mouse. Users can be either more or less experienced with programming because it is possible to customize the USB modules both via command lines and a GUI (graphic user interface). In order to, install, customize and apply USB pushbuttons and pushboxes, users must be familiar with the basic functions of their computer and its file system (e.g. Windows, Mac).

Chapter 2 informs you about possible fields of application, user scenarios, and the hardware's and software's properties and limitations. The chapter also presents the differences between and common specifications of the two available USB modules. In chapter 3, you learn more about the hardware's composition and component parts with regard to both modules. Chapter 4 presents commonly used menu tags and general options of the app which comes with USB pushbuttons or pushboxes. Chapter 5 leads you through the processes of installing and customizing the different USB modules, step by step. Chapter 6 presents programming examples for common user scenarios, both via the delivered app and via command lines. In chapter 7, you get an overview of the USB modules' technical data. Chapter 8 contains an alphabetical index so you can easily find your topics by looking for relevant keywords.

2. Processes initiated by one push

USB modules as a simulation tool

USB pushbuttons and USB pushboxes were designed to substitute mouse clicks and hotkeys in order to initiate technical or industrial processes. Using a USB module instead of a mouse or keyboard can be helpful in many situations, for example if you need a robust tool in a production line or trade show. Using pushboxes or pushbuttons is also practical if you need to start processes quickly and intuitively like printing out a document or taking spontaneous snapshots in a photo box. Some common user scenarios in which a simple push can help you out are listed below.

- exhibitons, fairs and shows (commercial/professional)
- customer information terminals
- PowerPoint or video presentations
- start of machine processes by hand or foot
- gaming environment
- stage or TV shows
- fast and instant logout/login
- auto-typing of any, also complicated key-combinations “all in one”
- DJ settings
- photographer’s studios, mobile photo boxes

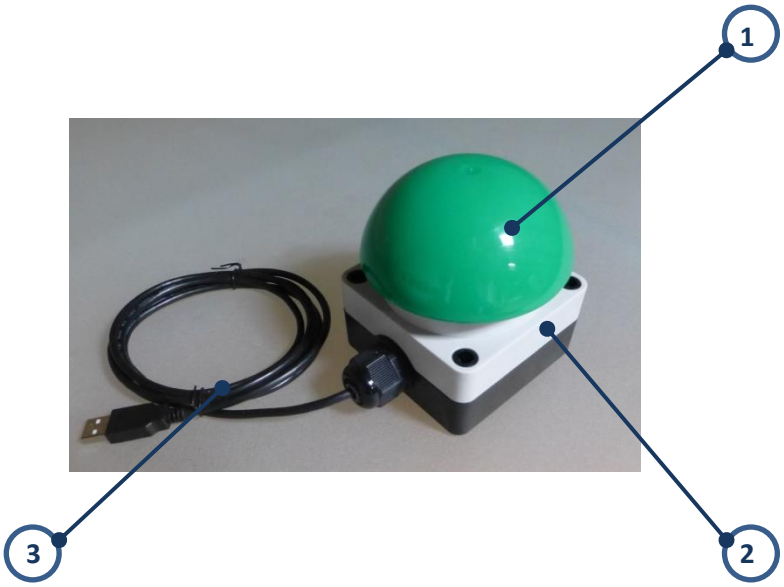
Both versions of the USB modules, which are USB pushbuttons (also: mushroom buzzers) and USB pushboxes, are currently serial terminal based when it comes to programming and configuration. They are able to substitute all sequences of mouse clicks and shortcut combinations with these devices “with a push” and robust enough to endure more than a million switching operations and about 100 kilogrammes in weight. Furthermore, they are splash-proof and compatible with all common file systems (Mac, Windows, Linux).

The only significant differences between buzzers and pushboxes are the number and designs of buttons integrated in the device. A buzzer, which includes a single, big and rounded button, is applied to substitute one shortcut or mouse click sequence at a time. In contrary, pushboxes were designed to substitute up to six mouse clicks or keyboard operations via several small flat buttons. Please consider that both buzzers and pushboxes are consumer products or tailored to scenarios in an industrial setting. Consequently, they were not designed for nuclear settings or any other environment which directly influences the life and health of human and other living beings.

Screen 1: USB pushbutton (left) and USB pushboxes (right)

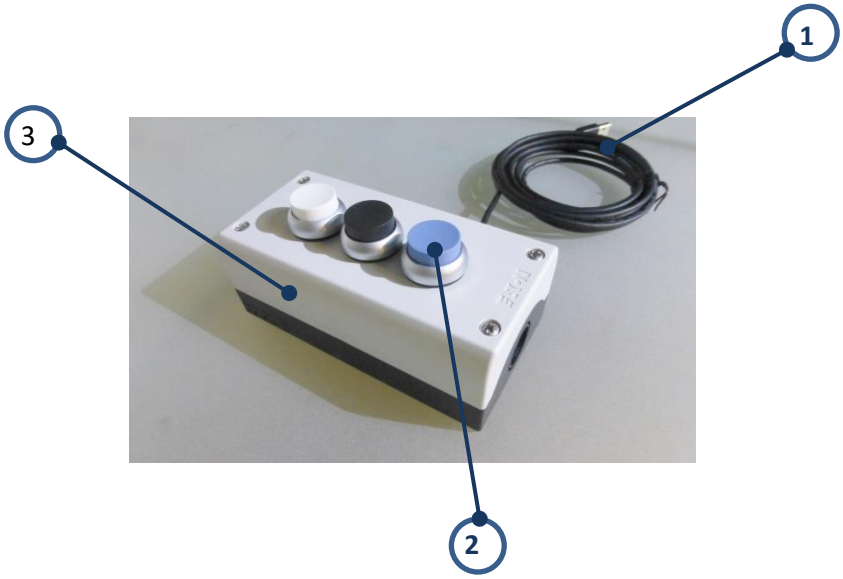
3. Hardware composition and component parts

3.1 Hardware composition of USB pushbuttons



Part number	Part name
1	mushroom buzzer
2	case with USB module inside
3	connection cable with USB plug

3.2 Hardware composition of USB pushboxes



Part number	Part name
1	connection cable with USB plug
2	button (small)
3	case with USB module inside

3.3 Hardware maintenance and disposal

USB modules, both pushbuttons and puxhboxes, were designed for long-term endurance. In case you need to dispose them, they are classified as electronic waste according to the RoHS guidelines Consider the following maintenance and disposal tips.

- Only mount and use your USB module on a firm and non-vibrating ground.
- Regularly clean your USB module with a soft and damp cloth if you used it in a dusty environment.
- Do not use any harsh detergence in order not to damage your USB module's surfaces.
- Do not arbitrarily repair your USB module if it is damaged, but ask for support.
- Do not dispose your USB module in the residential waste, but recycle it at your local collection station.

4. GUI: buttons, checkboxes and input fields

USB pushbuttons and pushboxes are delivered as a full package. This includes a CD with an application which helps the user to install and customize their hardware. This app will also be called GUI (graphic user interface) in the manual because. The chart in this chapter presents the main menu tabs, checkboxes and input fields on the GUI's start screen. Detailed operations will be illustrated in chapter 5.

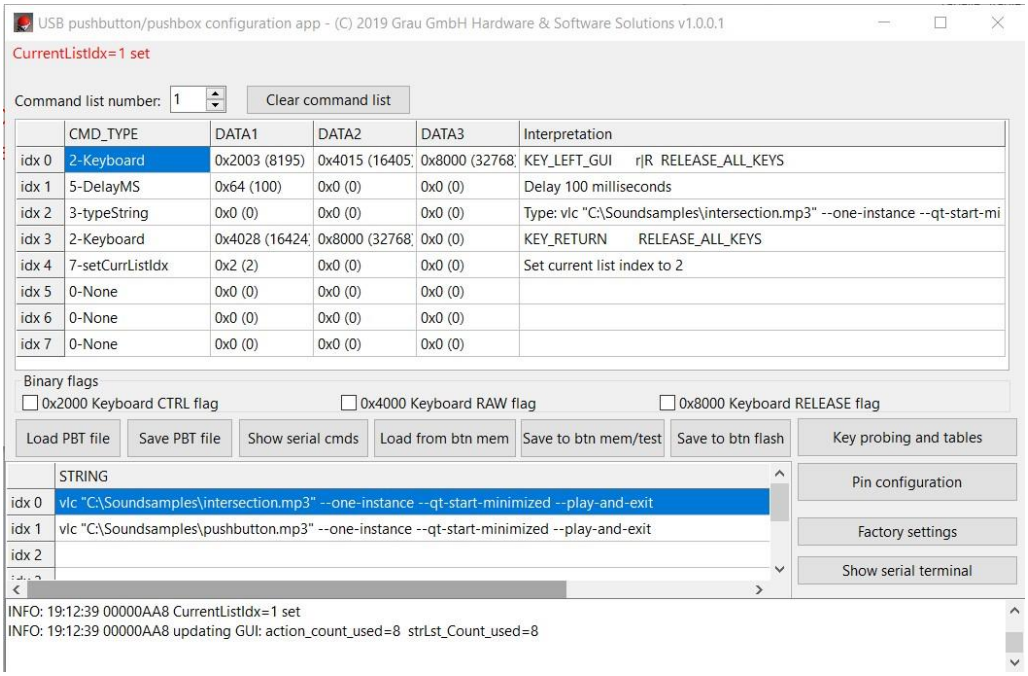
Functional aspects of the graphic user interface

Field name	Field type	Results of activation
Command list number	dropdown list	➲ The user can switch between different command list stored on the pushbutton or pushbox.
Clear command list	button	➲ The currently selected command list will be cleared/zeroed ➲ It is also possible to reload the former command list via ***"Load from btn mem".***
binary flags	checkboxes	➲ Users can set binary flags in order to determine different key behaviours/ access different key tables. ➲ ***0x2000***: is set for all functional keys (SHIFT, CTRL, ALT, GUI/OS-Key) located on a keyboard. ➲ ***0x4000***: is set for all HID-RAW table keys. ➲ ***0x8000***: is set if all currently pressed keys shall be released.
Load PBT file	button	➲ The GUI opens a new dialogue in which the user can select a stored pushbutton or pushbox template file.
Save PBT file	button	➲ The GUI opens a dialogue in which the user can save the current configuration to a pushbutton template file (also: PBT file).
Show serial cmds	button	➲ The GUI dumps the complete GUI configuration in serial programming commands text format. ➲ This includes the command lists,

		the strings and the pin configuration.
Load from btn mem	button	➲ The GUI loads command lists and files stored in the internal temporary button memory (RAM).
Save to btn mem/test	button	➲ The GUI configuration will be stored the button internal temporary memory (RAM). ➲ The GUI does NOT write the configuration to the button's permanent memory/flash memory. ➲ Unplugging and re-plugging the button reverts the button to the original configuration.
Save to btn flash	button	➲ The GUI stores the configuration to the button's permanent memory /flash memory.
Key probing and tables	button	➲ The GUI opens a new dialogue which helps to easily probe or select the shortcuts required by the user. ➲ Users can determine the keyboard layout for the given strings and ASCII chars. ➲ The dialogue includes an English keyboard view and an input field for showing the pressed keyboard shortcuts as button key codes.
Pin configuration	button	➲ The user defines the operations of physical pins of the USB board. ➲ Every pin is connected to a physical switch. ➲ Pins can trigger different command lists depending on their status: Trigger, Hold/Repeat, Release.
Factory settings	button	➲ This function resets the USB module's configuration and programming status. ➲ The GUI automatically and irrevocably reverts all recently stored commands on the USB

		module's permanent memory (flash) to factory settings.
Show serial terminal	button	➲ The GUI opens the dialogue in which the user can enter command lines and view the feedback the USB module provides. ➲ Users then can manually configurate their modules via command lines. This feature especially addresses users who are experienced with programming issues. ➲ The serial terminal serves as an alternative programming option for the GUI's selection table.
idx+number	field within selection table	➲ This interactive column in the GUI's selection table determines and displays the chronology of commands. ➲ The single operation in each command list in the index are defined and connected via the fields ***DATA1, DATA2*** and ***DATA3***. The complete command list is read like a book: From left to right and top to bottom.
CMD type+value	field within selection table	➲ This interactive column in the GUI's selection table displays command types and their allocations. ➲ Users can select a different command type in each field of the column, depending on their user scenarios. ➲ The available command types are: ***0-None, 1-HID System, 2-Keyboard***, ***3- typeString***, ***4-HIDCustomer***, ***5-DelayMS***, ***6-DelayS***, ***7-setCurrListIdx***, ***8-Mouse, 9-SerialTX***. ➲ Details on command types will be illustrated in chapter 6.
DATA1 - DATA3	fields within selection table	➲ By defining the fields ***DATA1, DATA2*** and ***DATA3***, the user determines how commands in the

		command lists are interpreted.
Interpretation	field within selection table	➲ This field displays the result of the entries in ***DATA1, DATA2*** and ***DATA3***. It thus shows the simulation sequence to be processed by the USB module. ➲ Consequently, user immediately can see if their configurations are correct. ➲ The field automatically adapts to changes within the selection table.



Screen 2: Start screen and main dialogue of the GUI for USB modules

5. Substituting mouse and keyboard with USB modules

USB pushbuttons and USB pushboxes simulate mouse-click sequences and key combination sequences. You've gotten familiar with the GUI's main functional elements in chapter 4 and with available hardware compositions in chapter 0. Now you will learn how to install and customize your USB modules. The chapters 5.1 to 5.2 lead you through all required processes by means of random user scenarios which serve as examples. In order to check specific programming settings, directly navigate to chapter 6.

Easy simulation of mouse, keys and type strings

Warning

Physical injury and environmental damage may occur if USB modules are used in a nuclear or medical setting.

USB modules were solely designed as consumer products and for industrial scenarios. Installing, customizing and applying USB modules in nuclear or medical settings can cause physical injury and environmental damage. Such critical environments are, for example, an x-ray apparatus, an emergency room or a nuclear power plant.

- ➲ Never use a USB module in a critical nuclear or medical setting where incorrect use can endanger the life and health of living beings or the natural environment.
- ➲ Use your USB module for the settings it was designed for, which cover a consumer-orientend or industrial use.

5.1 Installing a USB module

Before you can customize your USB module according to your wishes and purposes, you need to install. The following chapter describes the required installation processes with or without GUI step by step. The instructions were mainly designed for Windows users, but also include an installation guide for Mac and Linux users.

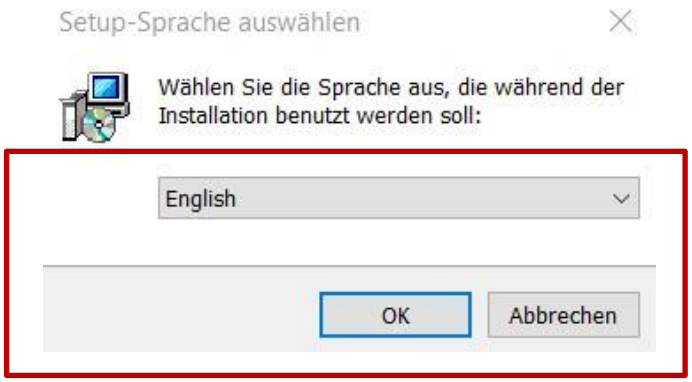
5.1.1 Installing the USB module's GUI

Installation of the GUI

1. Unpack the USB module and the mini CD included in the delivery.
2. Insert the mini CD into your computer's CD slot in order to install the programming app and a quick start manual.
    - Alternatively, you can download the quick start manual and the programming app on the company website.

3. Open the setup wizard on the mini CD or in your download folder in order to start the software setup (also: GUI installation).

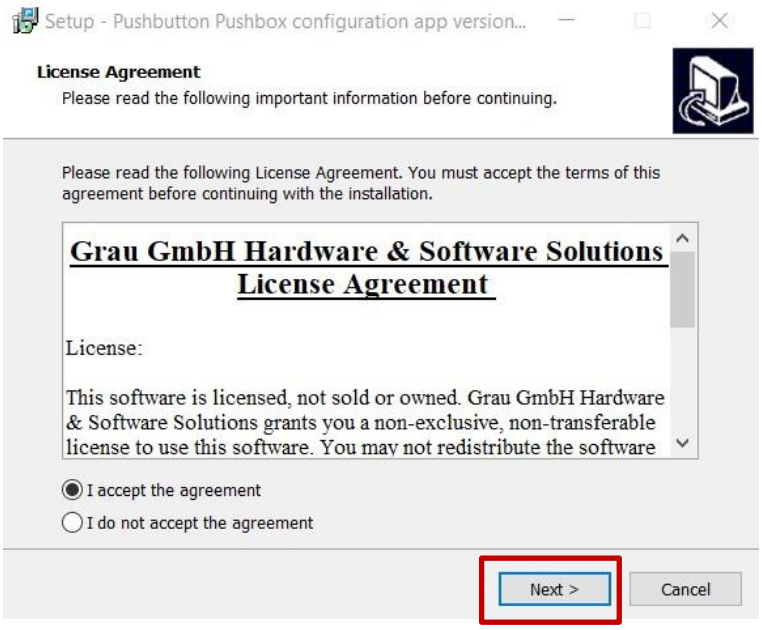
4. Select the setup language and confirm your selection with ***OK***.



Screen 3: GUI installation - selecting setup language

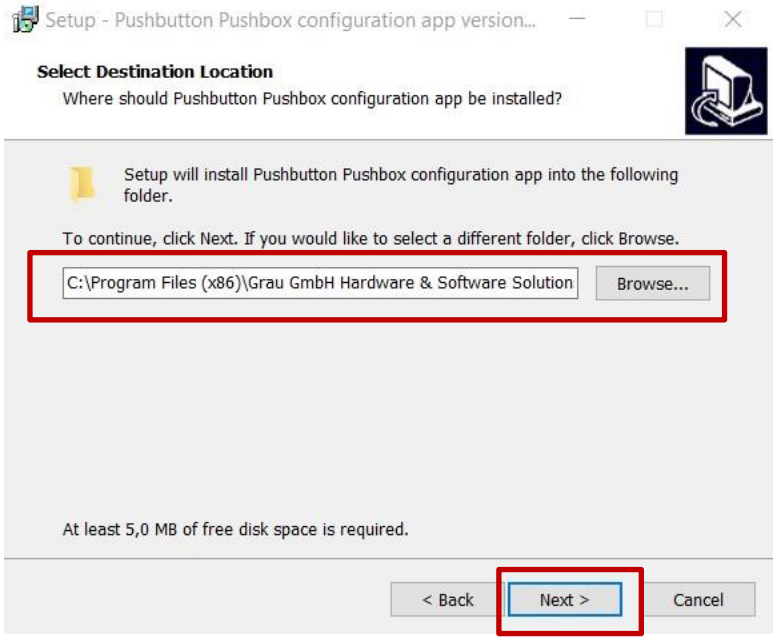
   - The setup wizard then opens a dialogue in which you can view, accept or deny the license agreement.

5. Accept the license agreement by clicking on ***Next >.***



Screen 4: GUI installation - confirming the license agreement

- The setup wizard opens a dialogue in which you can select a destination location for your programme folder.



Screen 5: GUI installation - selecting a destination folder

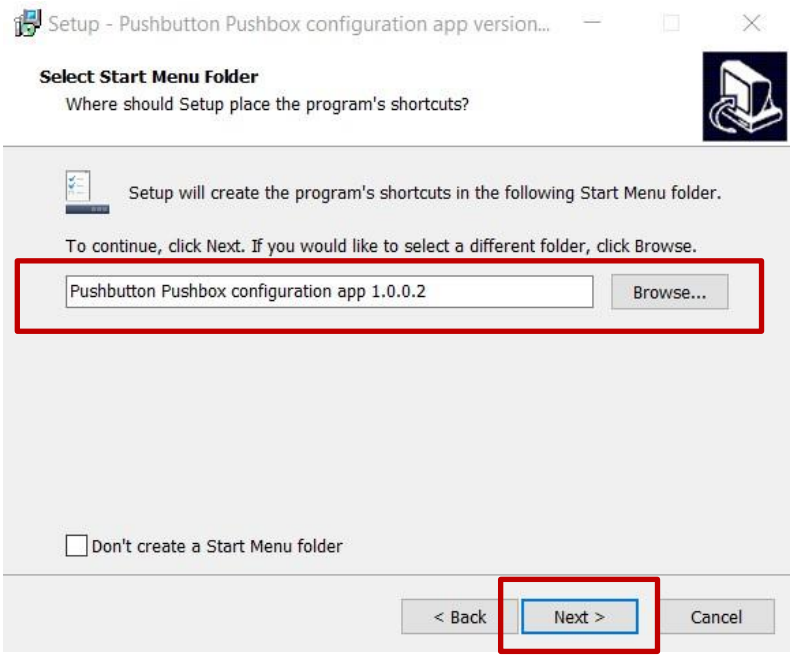
6. If you agree with the suggested destination location, confirm this selection by clicking on ***Next >.***

7. If you want to select another destination location, determine it via the ***Browse*** button. Then click on ***Next >.***

    - The setup wizard opens a dialogue in which you can select a start menu folder. You can also activate or deselect the creation of a start menu folder.

8. If you agree with the suggested start menu folder, confirm this selection by clicking on ***Next >.***

9. If you want to select another start menu folder, determine another one via the ***Browse*** button. Then click on ***Next >.***



Screen 6: GUI installation - creating a start menu folder

    - The setup wizard opens a dialogue in which you can activate or deselect the creation of a desktop shortcut.

10. Activate or deselect ***Create a desktop shortcut***, then click on ***Next >.***

11. The setup wizard opens a dialogue which sums up the selected setup configurations.

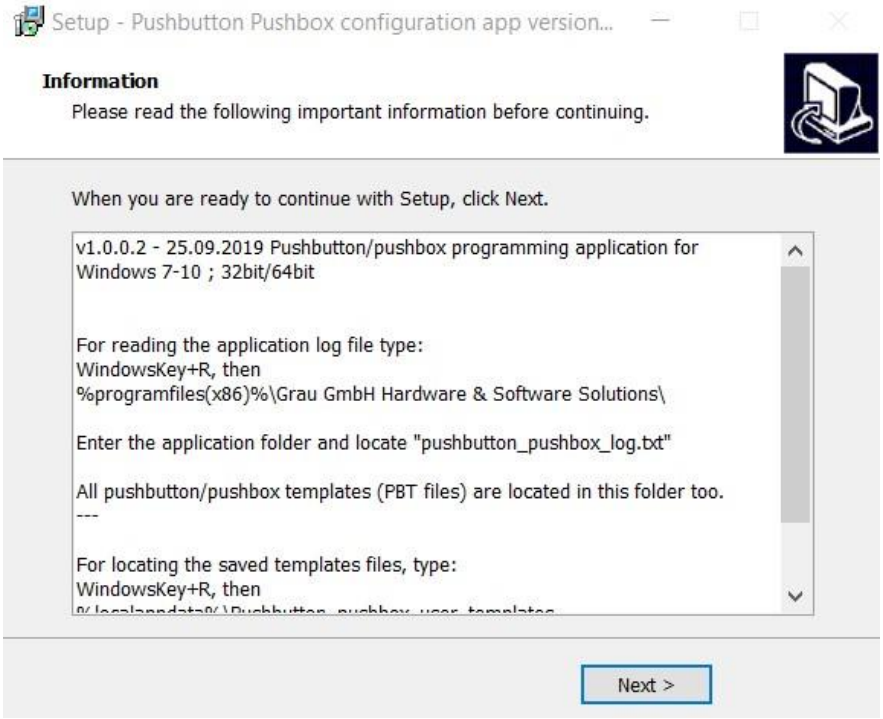
12. If you agree with the configurations, click on ***Install*** in order to start the GUI installation.

13. If you want to change the settings, click on ***< Back***, correct the settings in question and repeat all necessary processes. Then click on ***Install***.

    - You can also cancel the entire setup process and start it from the beginning.

    - The setup wizard opens another dialogue which informs the user about the configurations set for the installation process.

14. Click on ***Next >*** in order to proceed and start the installation.

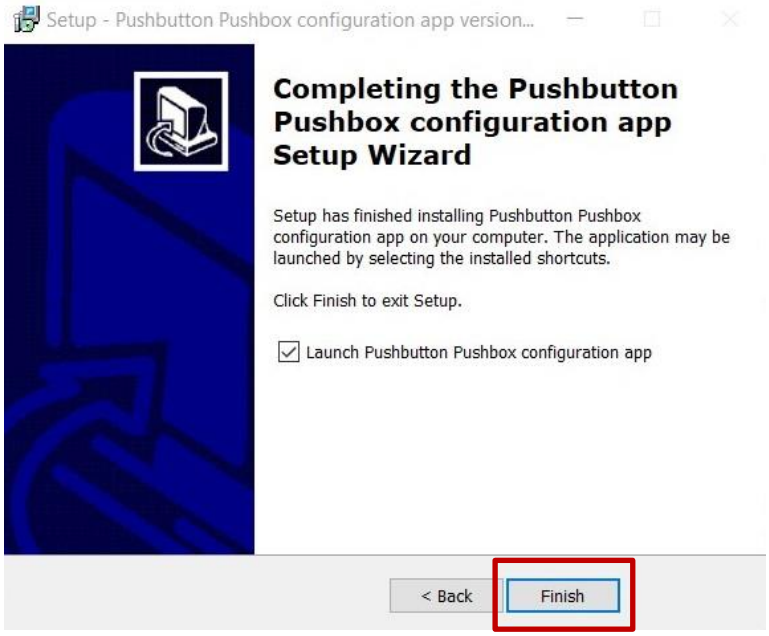


Screen 7: GUI installation – information dialogue before setup

    - The setup wizard now installs the GUI in the selected destination folder.

15. In the next dialogue, complete the setup by clicking on Finish.

    - You can also activate or deselect Launch Pushbutton Pushbox configuration app.



Screen 8: GUI installation – completing the setup process

✓ The setup wizard has completed the installation process. In case you activated the checkbox before, the GUI opens the launch window.

5.1.2 Installing the USB module's driver without GUI

Driver installation without using the GUI

1. Unpack the USB pushbutton and the mini CD included in the delivery.
2. Insert the mini CD into your computer's CD slot in order to install the programming app and a quick start manual.
    - Alternatively, you can download the quick start manual and the programming app on the company website.
3. If you use a Windows file system, open the ***control panel -> device manager*** in order to check your computer for the pushbutton's serial devices.
    - Mac users must open their computer's terminal and enter: ***sudo screen /dev/tty.usbmodemHIDA1***, type ***"screen /dev/tty."***, then press ***TAB***.
    - Linux users must open their computer's terminal and enter: ***sudo screen***. If the screen command is not found, they must type: "sudo apt-get install screen" for a common debian-based linux distribution

(like Ubuntu). If the screen has started properly, they must type “exit” to get back to terminal. They can then open the serial device with: ***sudo screen /dev/ttyACM0***. Therefore, they must type “screen /dev/ttyA” and then press ***TAB*** .

- If you see an “Arduino Leonardo” device in the section “Ports” (COM & LPT)’ without exclamation mark, the driver installation is completed. In all other cases you must download and install the Arduino IDE. Finally, it is necessary to repeat step 3.

  ✓ The driver for the serial port has been installed. You can now manually configure your USB module.

5.2 Customizing a USB module

After you successfully installed the driver or GUI for your USB module, you can start customizing it via the GUI or directly in the serial terminal CoolTerm. It depends on your own feeling of comfort and on your degree of programming knowledge how you configurate your USB pushbutton or pushbox. Generally, programming your USB module via GUI is the easiest way to go for programming beginners and possibly a shortcut to the target for programming experts. As the serial terminal is also included in the app, you can always open it and manually enter command lines designed for your purpose. In case you started programming without the GUI first and want to switch to the GUI configuration, you can load any saved PBT file and proceed via the app.

5.2.1 Customizing a USB module via GUI

There are generally two ways to configure a USB module via GUI. First, you can manually enter keys and key combinations under Key Probing and Tables. In many cases, especially with common key combinations including function keys, this is the easiest and most intuitive way to go. It is also possible to paste codes directly from integrated code tables into the selection table in the main dialogue. It is often required to paste codes into the fields ***DATA 1***, ***DATA 2*** and ***DATA 3*** if you want to perform more complicated operations like combining various command types. In this chapter, you wll get familiar with both methods.

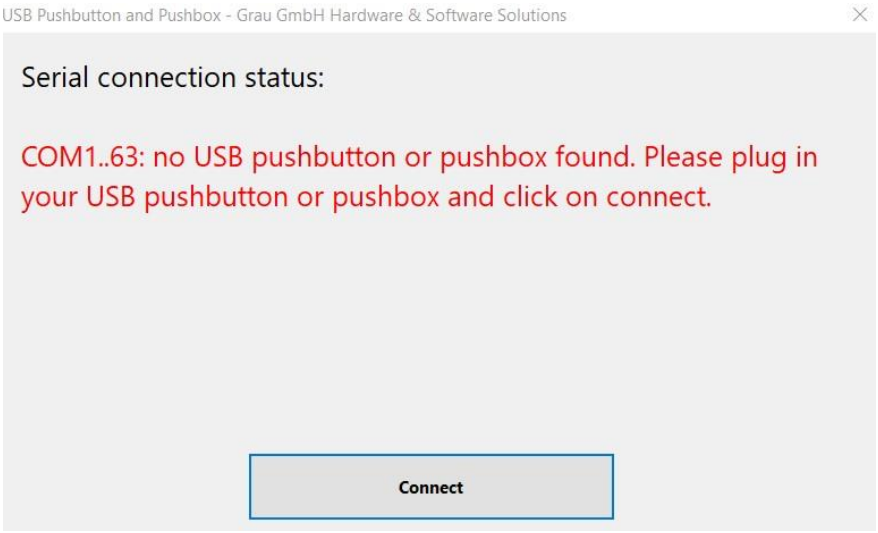
Connecting the module with the GUI (valid for all methods)

1. Start the GUI ***Pushbutton Pushbox configuration app*** on your computer.

- The GUI shows a dialogue which displays the serial connection status. Without any USB module plugged into your computer's USB port, the GUI cannot open any further dialogues or start any programming operations.

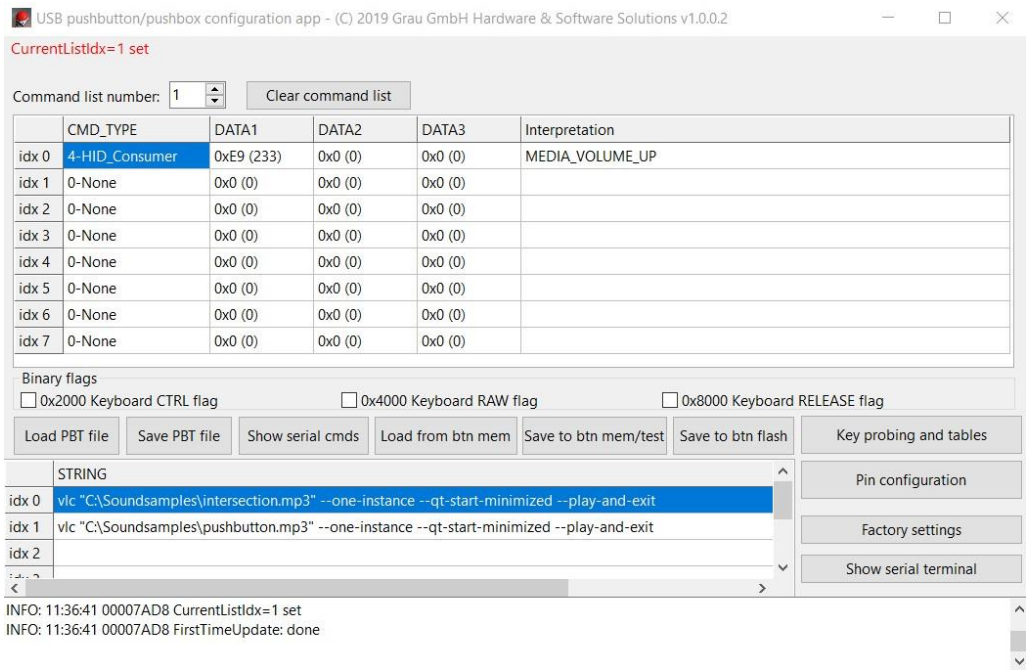
2. Connect your USB module to your computer by plugging it into an available USB port.

3. Click on ***Connect*** in the GUI dialogue mentioned in step one.



Screen 9: Configuration via GUI – connecting app and USB modules

- The GUI now opens the main dialogue which helps you customizing your USB module.



Screen 10: Configuration via GUI – start and main dialogue before programming

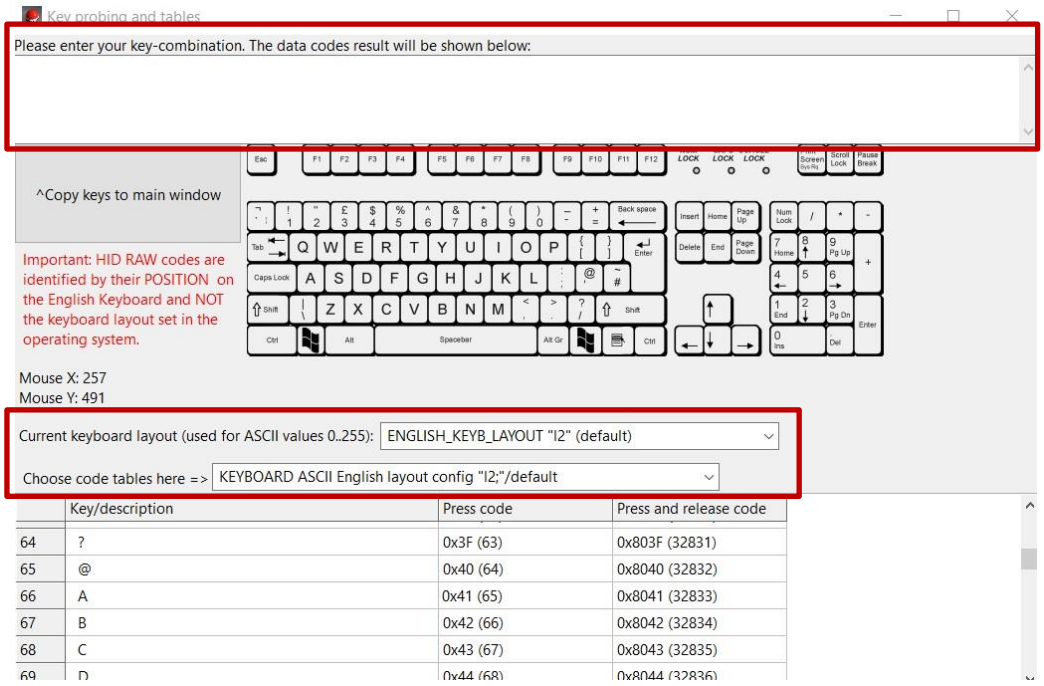
Entering keys directly under *Key Probing and Tables*

The following instructions show you how to programme the key combination “123” via the dialogue ***Key Probing and Tables***.

Example: configuration of the keys “123” via *Key Probing and Tables*

1. Select command list 1 (the default list on button trigger) in the main dialogue.
2. Click on ***Clear command lists*** and start with the first command entry “idx 0” if you want to want to enter a new configuration.
    - If you already programmed command lists in the GUI, you can also start at another command list.
3. Select the field ***DATA 1*** within a list entry (***idx 0*** – ***idx 7***). For a new configuration, start with idx 0.
    - If you already programmed entries in the command list, you can also start at another list entry.
4. Click on ***Key Probing and Tables*** in order to define your preferred keys.
    - The GUI opens a second dialogue. This dialogue includes an input field for keyboard shortcuts, the button ***Copy to main window***, an

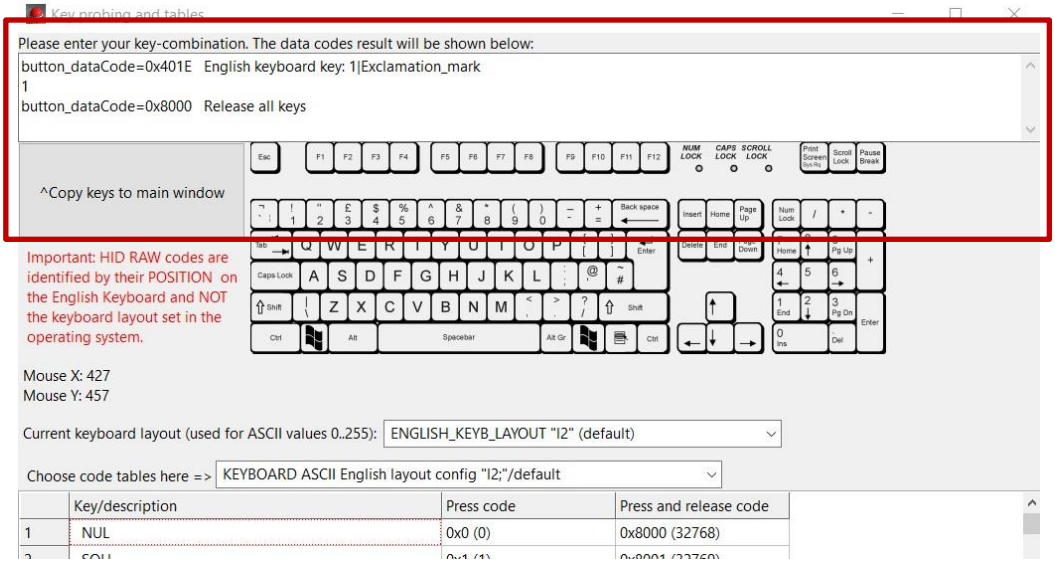
English keyboard image, a dropdown lists to select a keyboard layout and several tables with key codes.



Screen 11: Configuration via GUI – Key Probing and Tables: input field, keyboard layout and code tables

5. Manually enter the key “1” into the input field.

    - The GUI automatically displays the data codes result after all keys have been released.



Screen 12: Configuration via GUI – manually entering keys or key combinations

6. Click on ***Copy keys to main window*** to copy the data codes to the main dialogue at the selected position. Then return to the main dialogue by minimizing the window.

    - An auto-release data code (“0x8000”) be automatically added after each entered shortcut.

7. Repeat the steps 4 to 6 for the keys “2” and “3”.

    - Notice: If you only want to press the key, but do not release it before the next command, you must set the trailing data codes number “0x8000” (=”RELEASE ALL KEYS”) as “0x0 (0)” (=undefined).

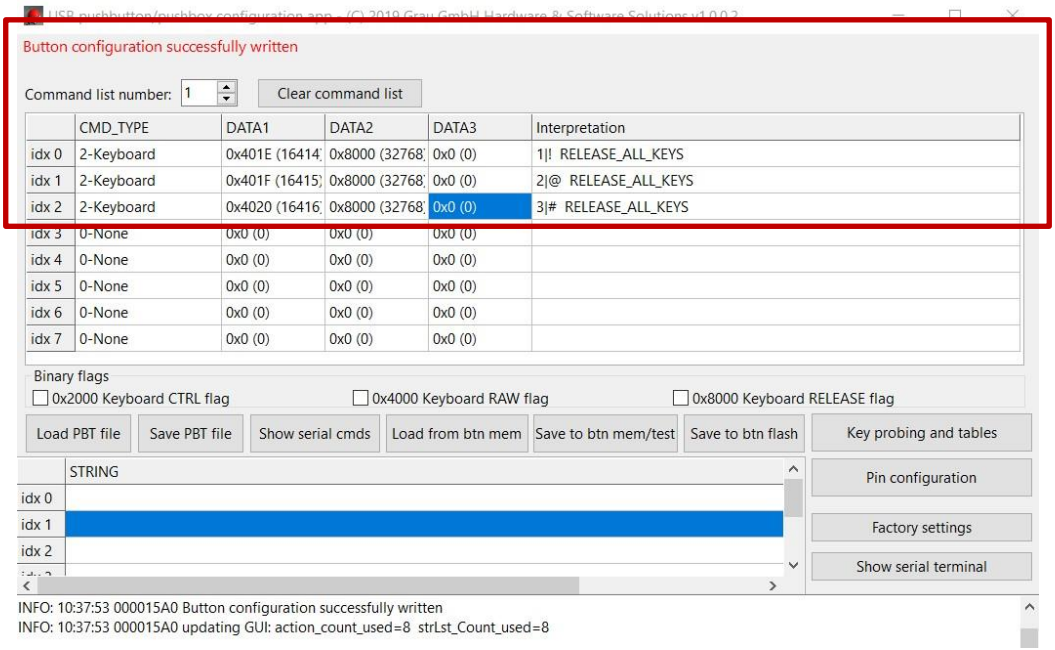
8. Check the concerned command list entry (idx+value).

    - The interactive chart with all command lists has adapted to the new configuration and displays it.

9. Check the field ***Interpretation*** in order to view and verify your programmed operations.

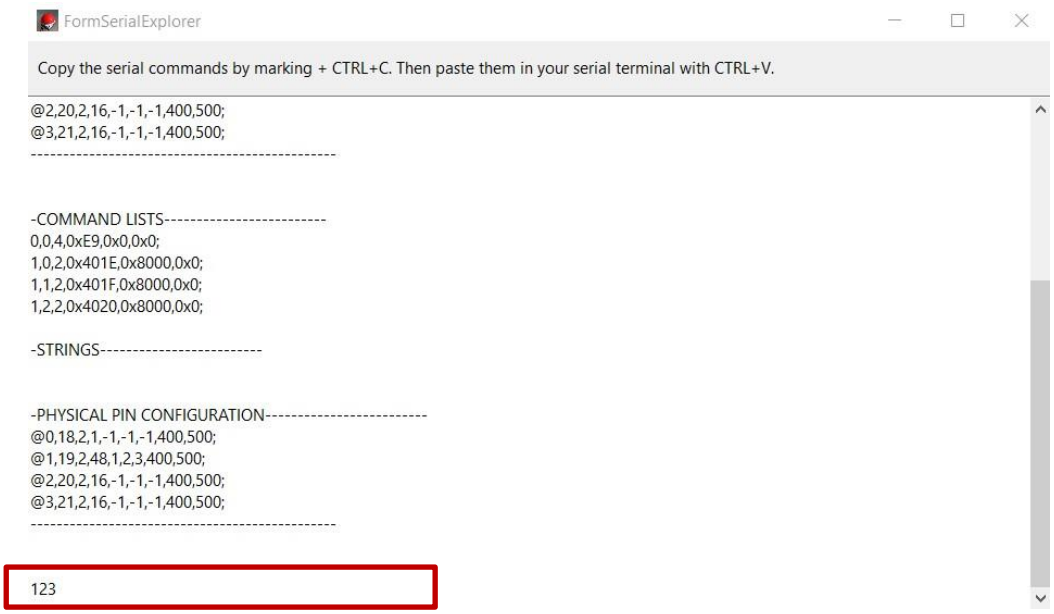
10. Click on ***Save to btn mem/test*** in order to temporarily store your USB module’s new configurations for testing.

- The GUI saves the newly set configurations in your USB module's internal memory without flashing the module. This means that you can still correct and adapt the configurations.
- If you want to revert to your original configuration you may also close the application, plug-out and re-plug the USB module.



Screen 13: Programming results in the main dialogue of the GUI

11. Click on ***Show serial terminal*** or ***Show serial cmds*** in order to test your USB module's new configurations.
    - ***Show serial cmds*** is a more compact terminal view which only contains the serial commands. Via ***Show serial terminal***, you can view all configurations as full strings.
12. Push the button(s) included in your USB module in order to verify your programming success.
    - In the best case, the module immediately executes the correct operations. Otherwise, you need to correct your configurations, save and test them again.



*Screen 14: Correct programming result "123" in the FormSerialExplorer (via **Show serial cmds**)*

13. Return to the main dialogue.

14. Click on ***Save to btn flash*** if you permanently want to save the new configurations in your USB module's internal flash memory. Do not immediately flash the module if you are going to initiatiate any changes regarding the stored configurations.

    ✓ You have successfully customized your USB module. The configurations for the key combination "123" have been stored in the module's internal flash memory.

Manually entering codes and strings in the main dialogue

Manual input of codes within the main dialogue

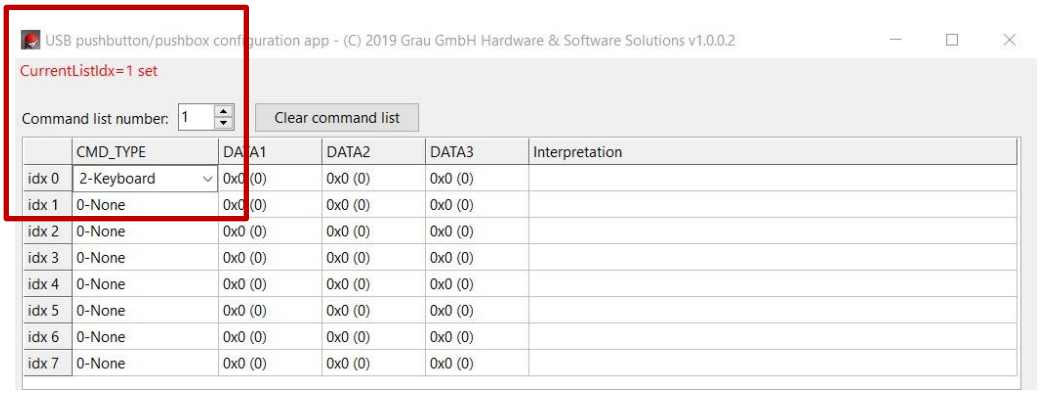
The following instructions show you how to programme the following key and string combination: Windows + r, 500 ms delay, "programme path". After programming, the GUI is supposed to open the programme "OpenOffice" on the computer. All values shall be manually entered into the selection table in the main dialogue.

1. Select a command list.

2. Start with the command list number 1 and click on ***Clear command lists*** if you want to want to enter a new configuration.

- If you already programmed command lists in the GUI, you can also start at another command list.

3. Select a command type for an entry in the command list (***idx + value***).



Screen 15: Configuration via GUI – defining a command list and command type

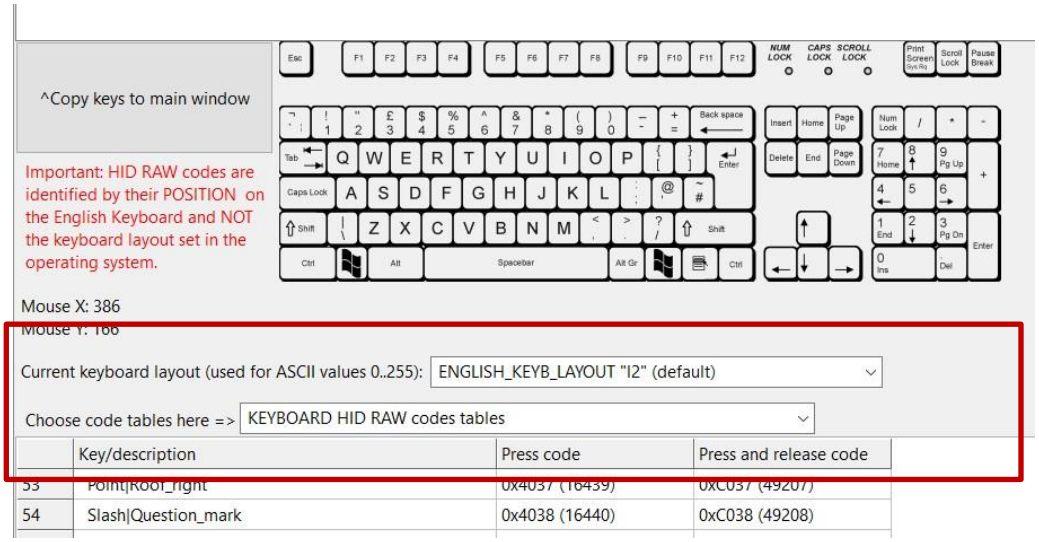
4. Within the column (also: index row) ***idx 0***, select the field ***DATA 1*** for configuration.

   - It is always recommended to start with ***DATA 1*** within an index row (left -right) and at ***idx 0*** (top-down) unless you already configured any index rows

5. Click on ***Key Probing and Tables*** in order to define the keys simulated by the field entries ***DATA1, DATA2*** and ***DATA3***.

   - The GUI opens a second dialogue (also see Screen 11:). This dialogue includes an input field for keyboard shortcuts, the button ***Copy to main window***, an English keyboard image, a dropdown list to select a keyboard layout and several tables with key codes.

   - Define a keyboard layout and a code table. For more information on keyboard layouts and code tables, see the chapters 4 and 6.1.



Screen 16: Configuration via GUI – defining a keyboard layout and code table

6. Select your preferred “Press code” or “Press and release code” from the code list. For this example, select the “Press” code for the Windows key “LEFT_GUI”.

    - If you select a “Press code”, the GUI does not automatically create a release code in the main window. You usually select press codes for parallel keystrokes.

    - If you select a “Press and release code”, the GUI automatically adds a hold and release code in the main dialogue. You usually select press and release codes for key sequences.

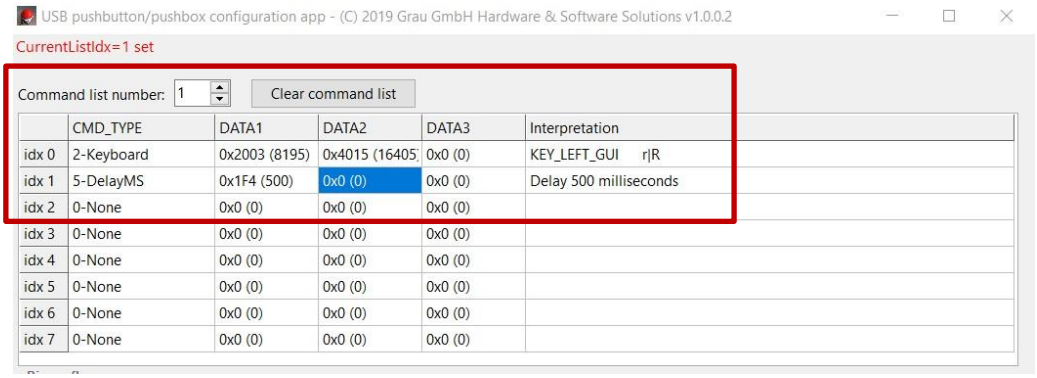
7. Copy the “Press code” for “LEFT_GUI” and return to the main window. Paste it into the field ***DATA 1***.

8. Now select the field ***DATA 2*** and repeat the steps 4 to 7 with the “Press code” for “r”.

9. Check the field ***Interpretation*** to verify your result.

10. Move on to the index row “idx 1”and select the command type ***5-DelayMS*** (delay in milliseconds).

11. Enter the value “500” into the field ***DATA 1*** and check the field ***Interpretation*** to verify your result.

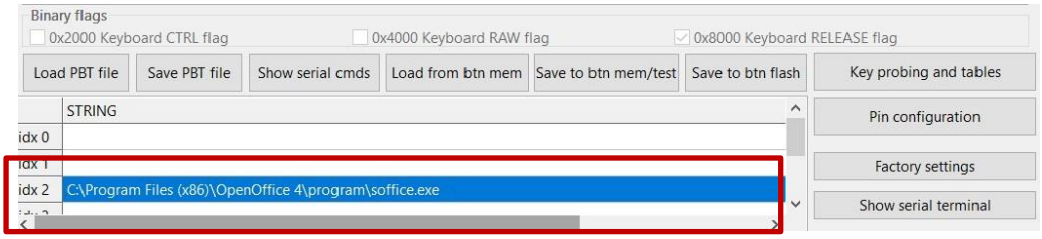


Screen 17: Configuration via GUI – setting a different command type in a new index row

12. Move on to the index row “idx 2”. Select the command type ***3- typeString*** in order to enter a string into the main window.

13. Copy your string (here: the programme path) into the input field at the bottom of the dialogue.

14. Make sure to choose the column with the correct index number (here: idx 2) and press ***Enter***.



Screen 18: Entering a type string (here: programme path) into the bottom input field

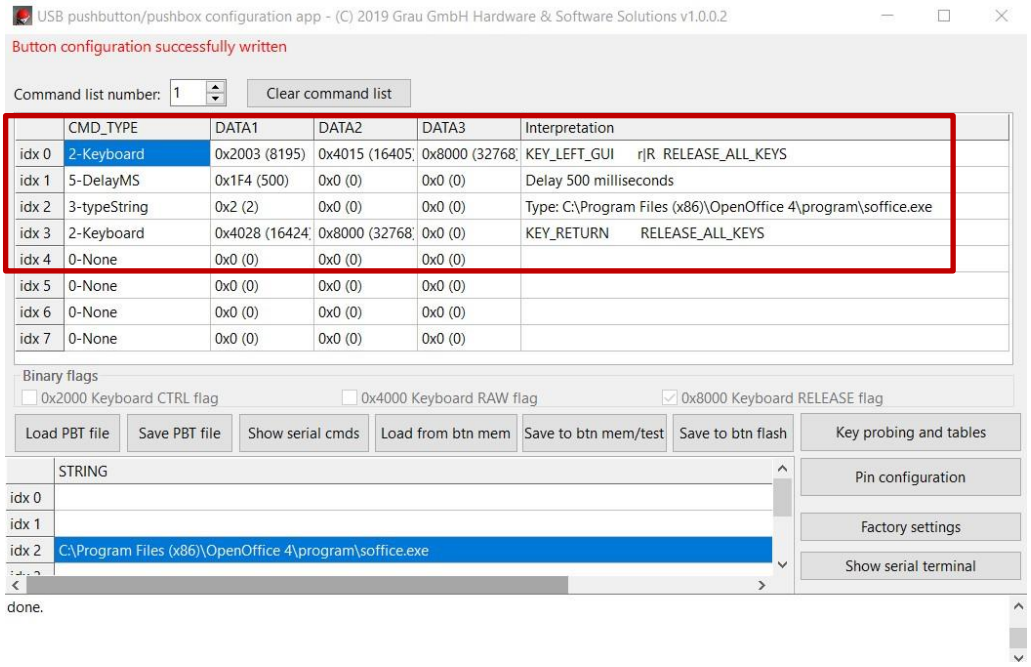
15. Enter the value “0x2 (2)” into the field DATA 2. “2” refers to the current index number.

16. Move on to the index row “idx 3”. Select the command type ***2- keyboard***.

17. Select the field ***DATA 1***.

18. Click on ***Key Probing and Tables***. Repeat the steps 4 to 7 with the “Press and release code” for “Return/Enter”.

19. Check the field ***Interpretation*** to verify your result.



Screen 19: Configuration via GUI – complete and actualized command list

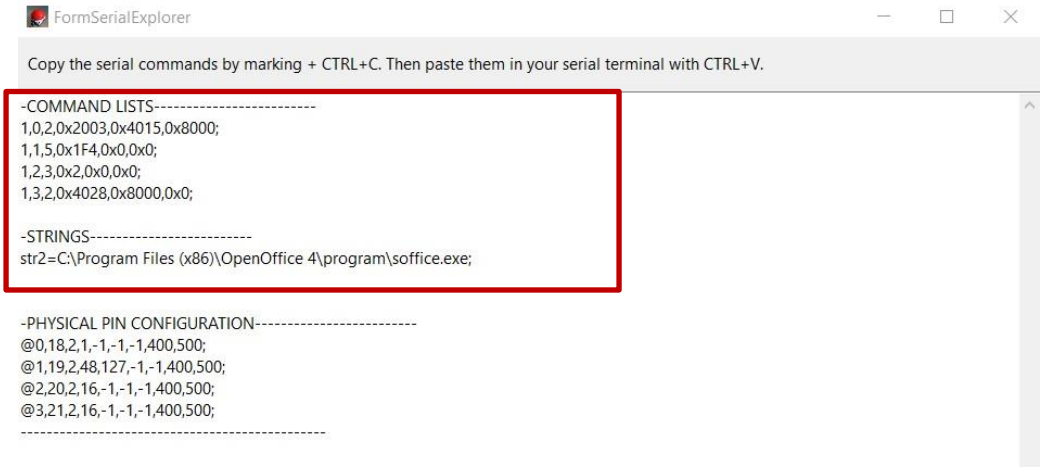
20. Click on ***Save to btn mem/test*** in order to temporarily store your USB module's new configurations for testing.

    - The GUI saves the newly set configurations in your USB module's internal memory without flashing the module. This means that you can still correct and adapt the configurations.

21. Push the button(s) included in your USB module in order to verify your programming success.

    - In the best case, the USB module opens the programme "OpenOffice" with 500 milliseconds delay on your computer. Otherwise you need to check your configurations, correct and save them again.

    - Your current configurations are now listed in the Form Serial Explorer (via ***Show serial cmds***) and can be viewed and modified in the serial terminal.



*Screen 20: Correct command list and string in the FormSerialExplorer (via **Show serial cmds**)*

22. Return to the main dialogue.

23. Click on ***Save to btn flash*** if you permanently want to save the new configurations in your USB module's internal flash memory. Do not immediately flash the module if you are going to initiatiate any changes regarding the stored configurations.

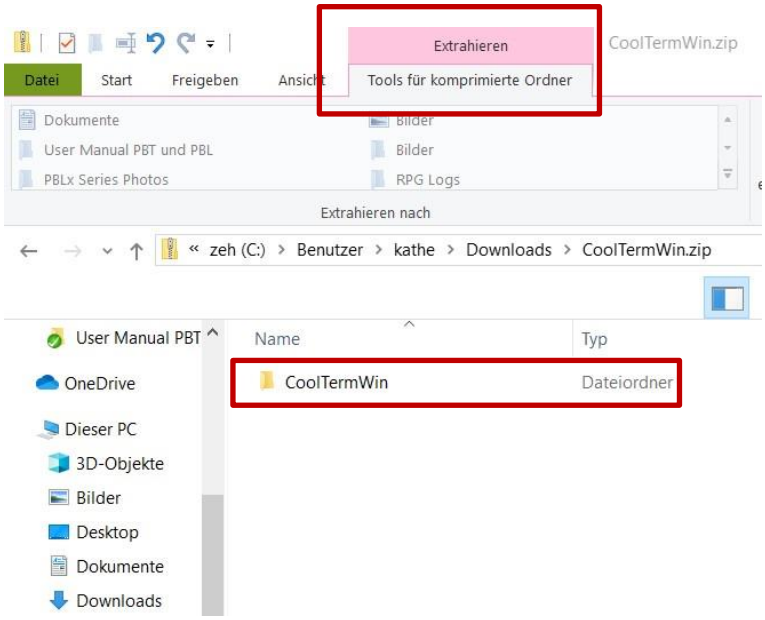
    ✓ You have successfully customized your USB module. The module now directly opens the programme OpenOffice on your computer through a push.

Tip: If you want to configurate one or several type strings, you can easily enter them directly into the serial terminal which integrated in the GUI. Always save your manually typed or copied string configurations via ***/s***. The GUI then automatically displays all configurations defined in the serial terminal in the interactive table.

5.2.2 Customizing a USB module without GUI

Module configuration via serial terminal

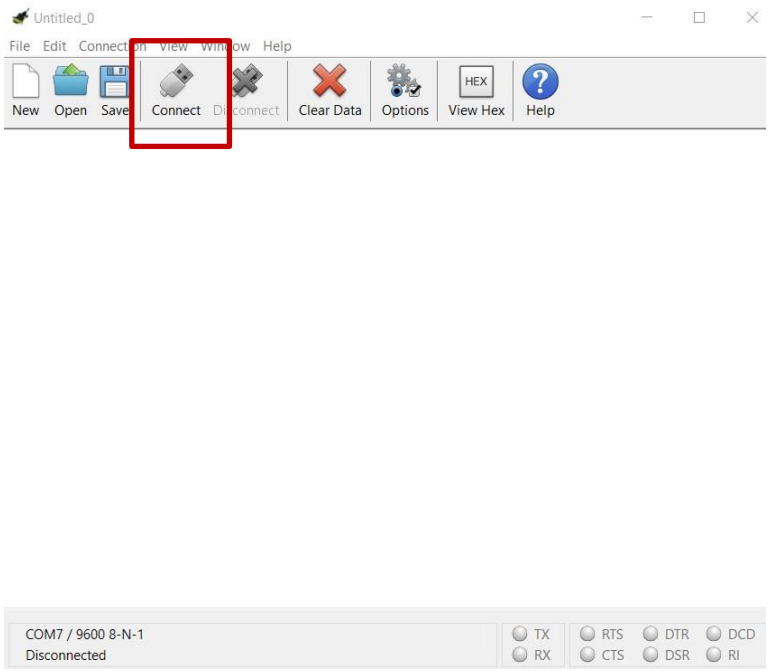
1. If you use a Windows file system, download and extract the software CoolTermWin.zip (Windows).



Screen 21: Customizing USB modules without GUI – CoolTerm download folder

2. Install CoolTerm as you usually install software in your file system.

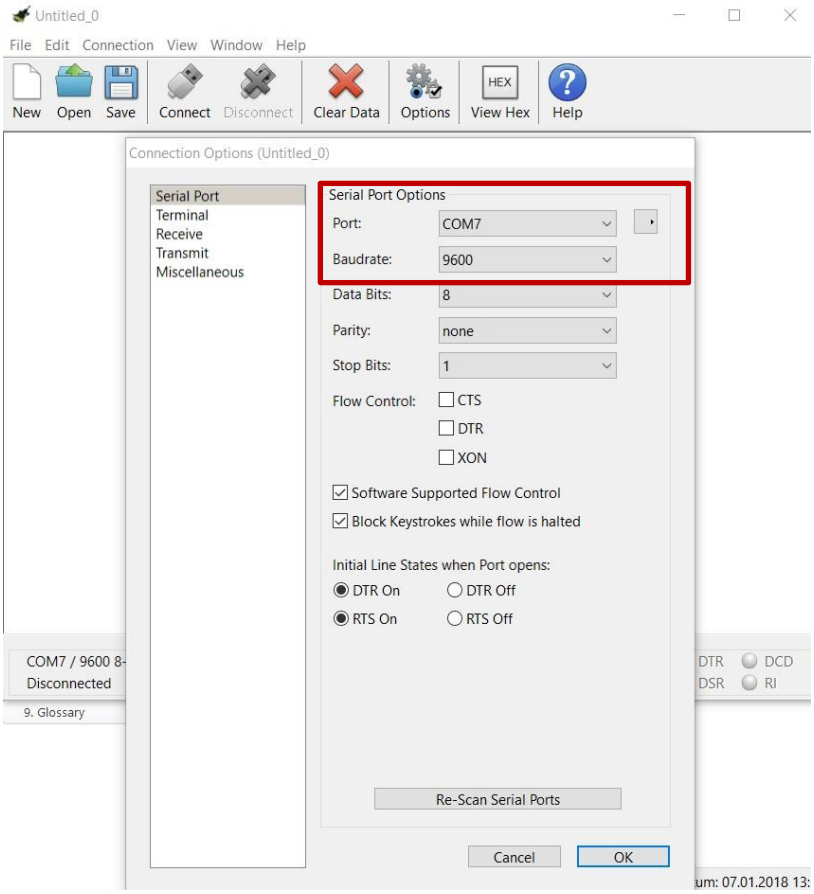
    - The serial terminal CoolTerm is installed on yor computer. You can also set your own preferences after setup.
    - CoolTerm opens a dialogue in which you can connect the terminal with your USB module. As long as the the module is not connected, the terminal window stays empty.



Screen 22: Customizing USB modules without GUI –serial terminal before connecting

3. In the menu bar of CoolTerm, click on Options and select Serial Port.

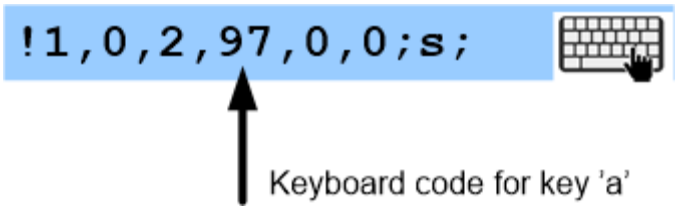
    - CoolTerm opens a selection dialogue in which you can adjustseveral parameters.



Screen 23: Customizing USB modules without GUI – serial port options in CoolTerm

4. Select the last COM-port in the dropdown list and set the baud rate on “115200”.

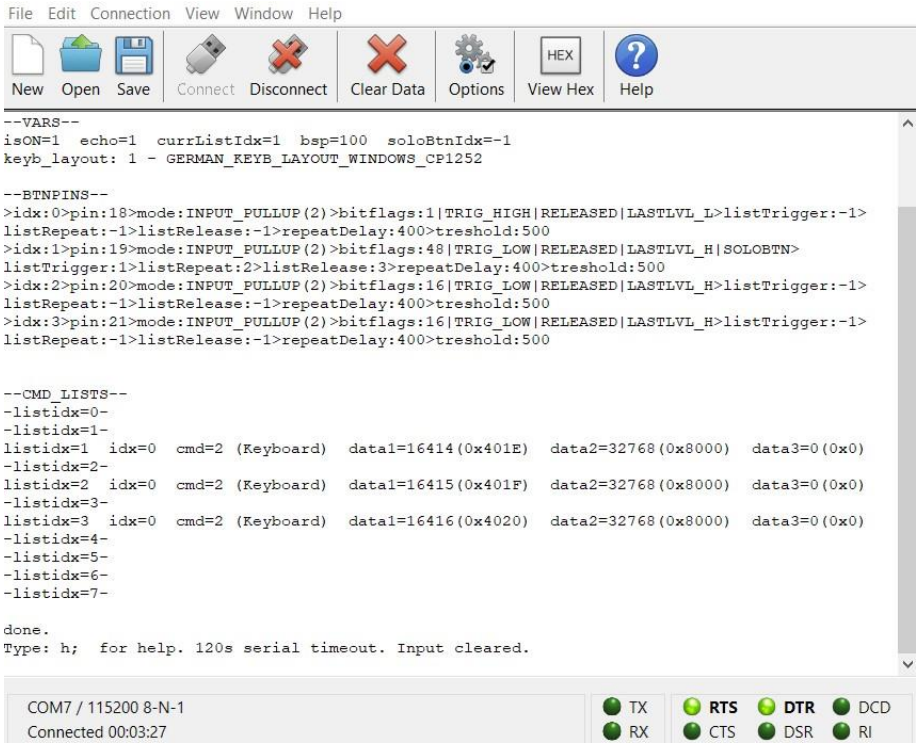
    - The button is configured via serial commands. You enter a command into the serial terminal, and send it to the push button by typing a ‘;’ (semicolon). You can use several commands in one line, each separated with semicolon.



Screen 24: Customizing USB modules without GUI – example of a command line

5. In order to view and adapt your USB module's configuration, enter "!h;" in the first line.

6. CoolTerm now displays the USB module's current configuration.



Screen 25: Customizing USB modules without GUI – command lines In CoolTerm

- **USB module connection for Mac users:** You must open your Mac's terminal and enter: ***sudo screen /dev/tty.usbmodemCHIDA1*** (type "screen /dev/tty." and then TAB key to see the connected tty pushbutton devices). If the GNU screen has not enough comfort for your, you can download CoolTermMac.zip (Mac). Then you must start CoolTerm, click on ***Options***, choose "serial-port = usbmodemCHIDA1" and "baud rate 115200". Finally, it is necessary to click on ***Connect***.

- **USB module connection for Linux users:** You must open your computer's terminal and enter: ***sudo screen /dev/ttyACM0 (type screen /dev/ttyA***, then press ***TAB***. If the GNU screen does not work for you, you can download CoolTermLinux.zip (Linux). Then you must start CoolTerm, click on ***Options***, activate the last serial port in the

dropdown list and “baud rate 115200”. Finally, it is necessary to click on ***Connect***.

7. Manually enter the command lines which you need for the planned module operations.

8. After you have entered all required command lines, click on ***Save*** in order to store them in the modules’ internal memory.

9. Push one or several buttons included in the module in order to verify your programming success.

    - In the best case, the module immediately executes the correct operations. Otherwise, you need to correct your command line inputs, save and test them again.

        ✓ You have successfully customized your USB module. All relevant commands and operations have been stored in the module’s internal memory.

Tip: The chapters 5.1.1 to 1.1.1 illustrate “first steps” with both the GUI and the serial terminal CoolTerm in a general context. As the purposes and user scenarios for USB modules are numerous and various, it is not possible to present each one step by step. Common user scenarios are presented in chapter 6. We can add your configuration scenario to our examples presented at www.grauonline.de. Also contact us for detailed support.

6. Common user scenarios: programming examples

In chapter 5, you have learned how to install your USB modules on the target device and how to initiate the required programme processes via the GUI. This chapter presents some common programming examples, both via the GUI and via command lines. Depending on your knowledge status, purpose and user scenario, both ways can lead to a successful module configuration.

6.1 Programming examples via GUI

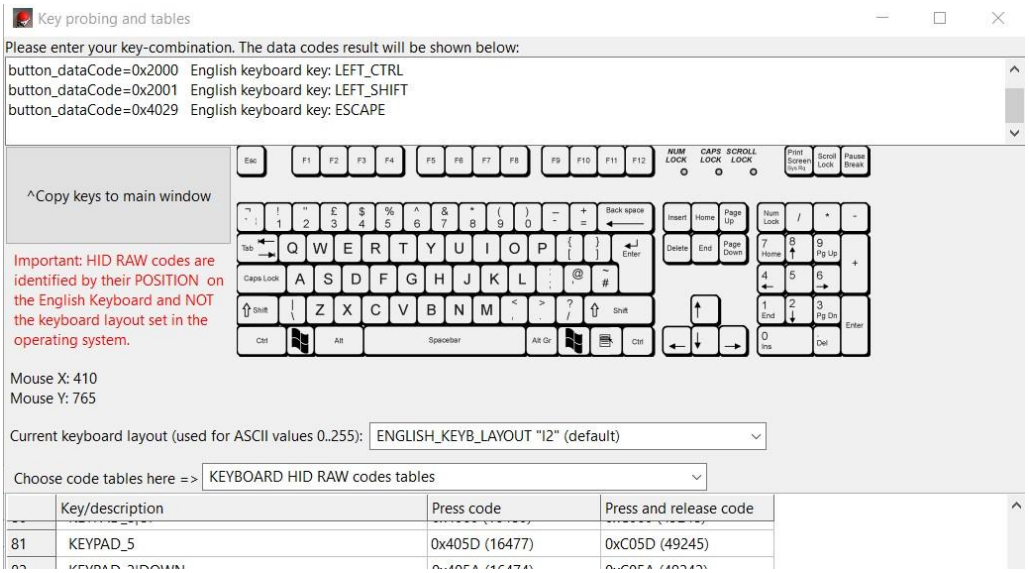
Practical user scenarios with GUI

With the GUI presented in the chapters 4 and 5, you can create a variety of module operations. The following examples shown in the screenshots below are some commonly required operations in a Windows scenario. For deeper and more detailed programming information, please check chapter 0.

Tips:

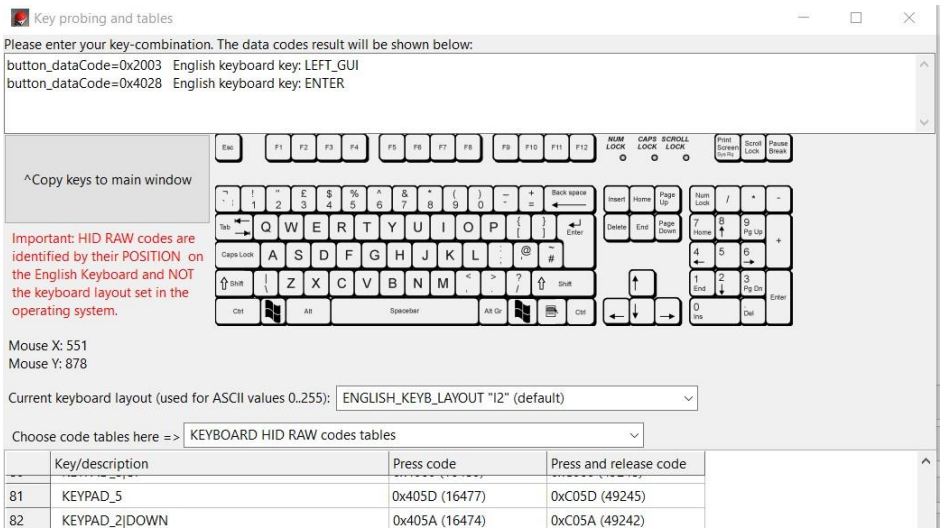
- If you want to be sure that all keyboards emulated by the USB module support your configurations, always select the code table “KEYBOARD HID RAW codes tables”.
- ASCII-based code tables do not include Windows function keys and thus cannot always be emulated by all USB modules. However, they work well with letters or figures.
- Also note that the USB module always defines a code according to its position, by its interpretation via an operating system. So enter the correct position code if a letter is located differently on another keyboard to be used. For example, there are different key positions if one compares an English keyboard to a German one.
- It is always the easiest way to use the GUI’s input field for probing (as explained in chapter 5.2.1).
- Do not forget to save your configurations both temporarily via ***Save to btn mem/test***, or, after testing, permanently via ***Save to btn mem/flash***.
- In case your GUI puts out wrongly programmed results and you cannot delete incorrect command lists via ***Clear command list***, click on ***Factory settings*** in order to reset your USB module.

Opening Windows Task Manager



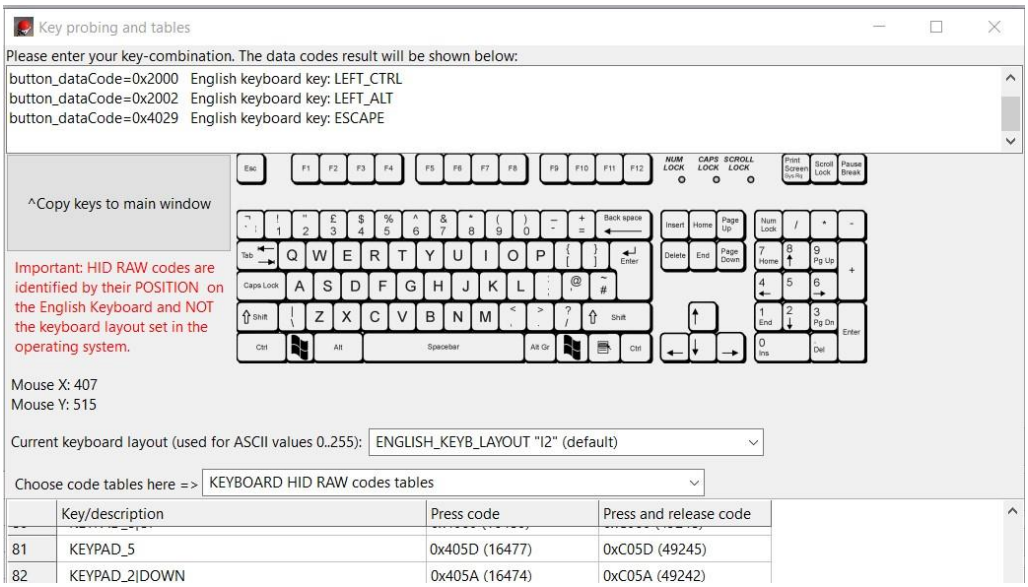
Screen 26: Simulating the key combination "Ctrl + Shift + Esc"

Three single "Enter" (with instant release)



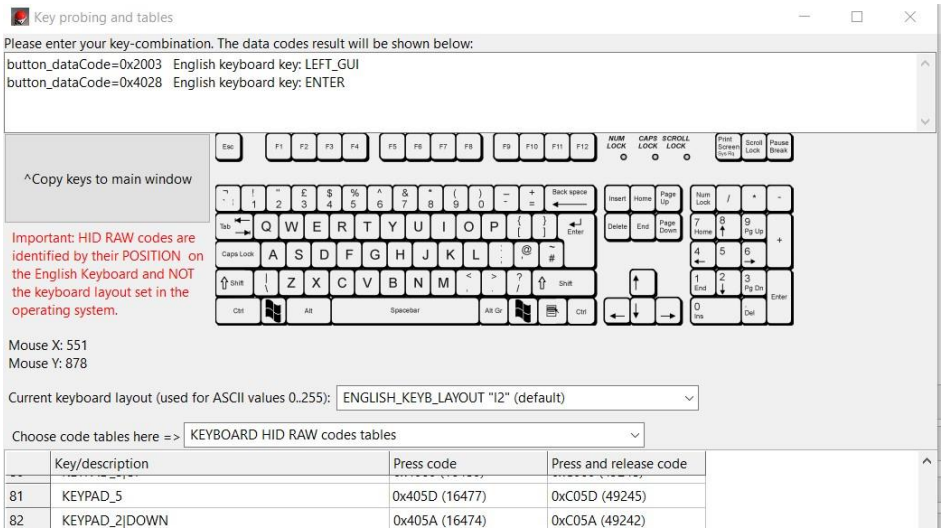
Screen 27: Simulating the key "Enter" three times

Cycling Windows



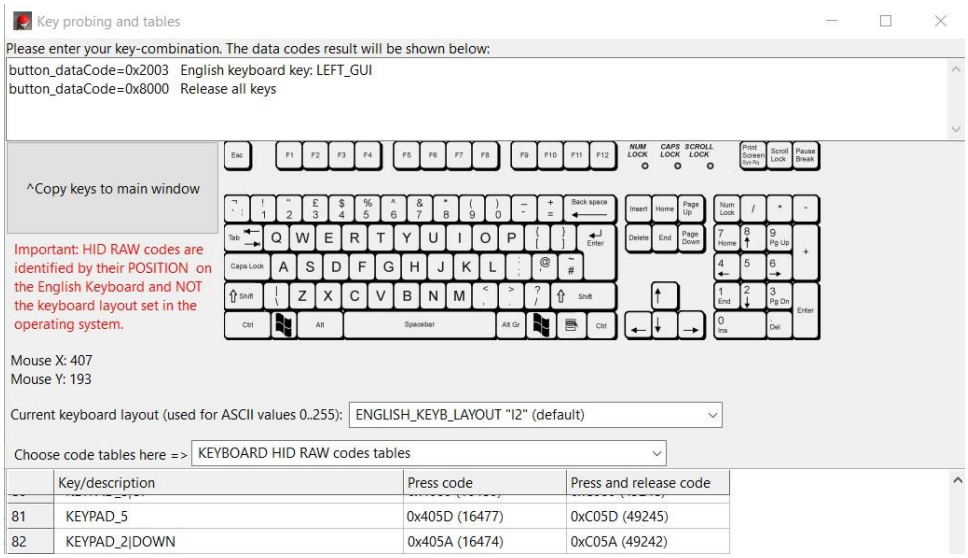
Screen 28: Simulating the key combination "Ctrl + Alt + Esc"

Three single "Enter" (with instant release)



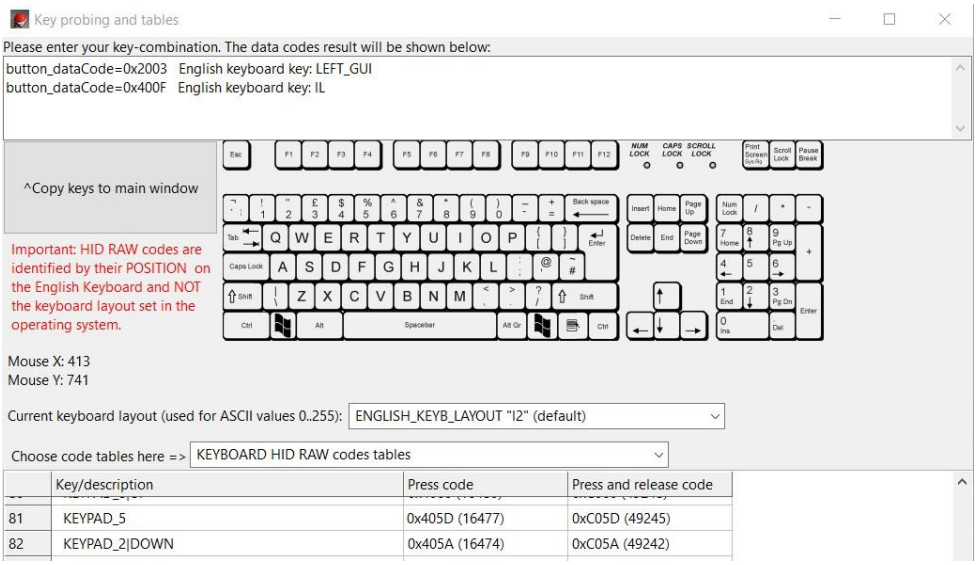
Screen 29: Simulating the key "Enter" three times

Left operating system key (hold and release)



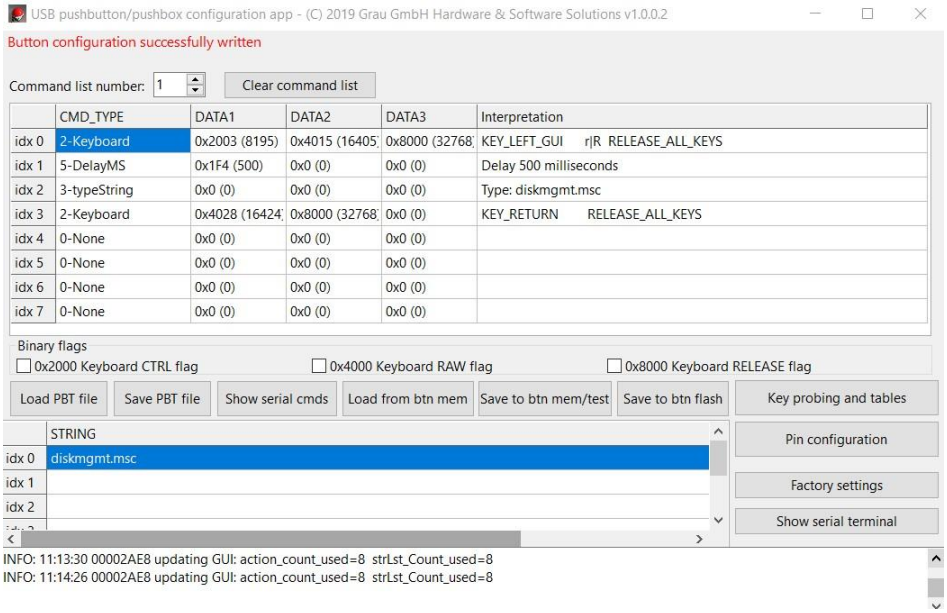
Screen 30: Simulating the Windows key and releasing it

Windows logout



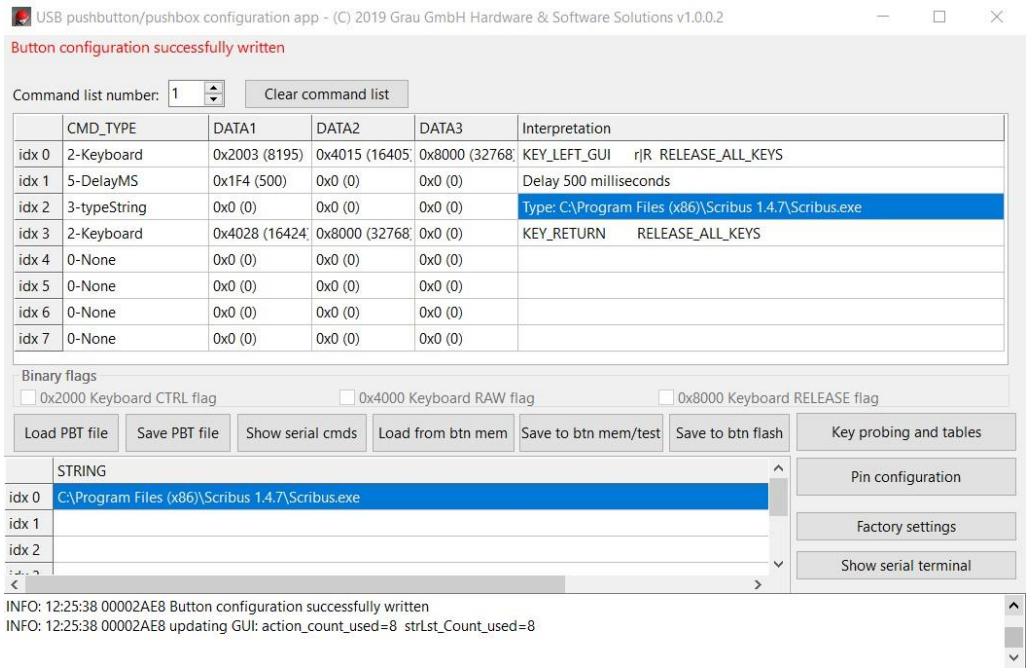
Screen 31: Simulating the key combination "Windows + L"

Opening Windows disk management



Screen 32: Simulating the the string and key combination "Windows key + r ; 500ms delay ; diskmgmt.msc ; Enter"

Opening programmes in a Windows file system

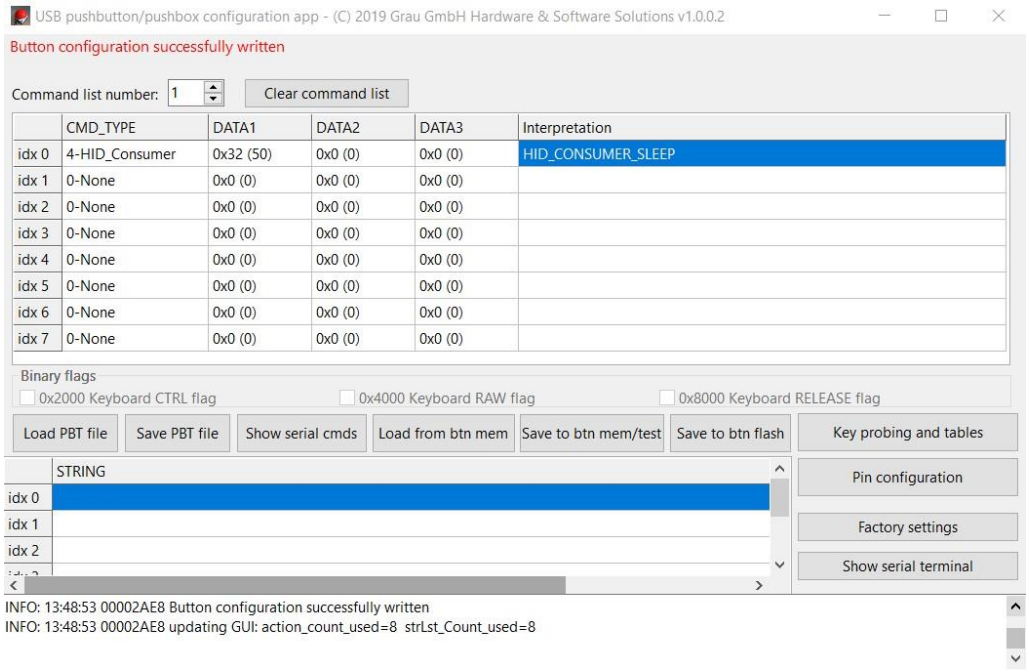


Screen 33: Simulating the the string and key combination "Windows key + r ; 500ms delay ; "programmePath" ; Enter"

Tips:

- In case the strings you typed in do not work, check the keyboard layout in your GUI and adjust it to your current keyboard type. Compatibility issues can occur if you have selected an English keyboard type, but type in keys on a German one. These compatibility issues are based on a different interpretation of ASCII signs depending on your operating system.
- Regarding programme paths, it can be necessary to manually add the programme file to the path in the format "\programme.exe" (Windows, different name ending for Mac).

Sleep mode (HID System/Consumer keys)

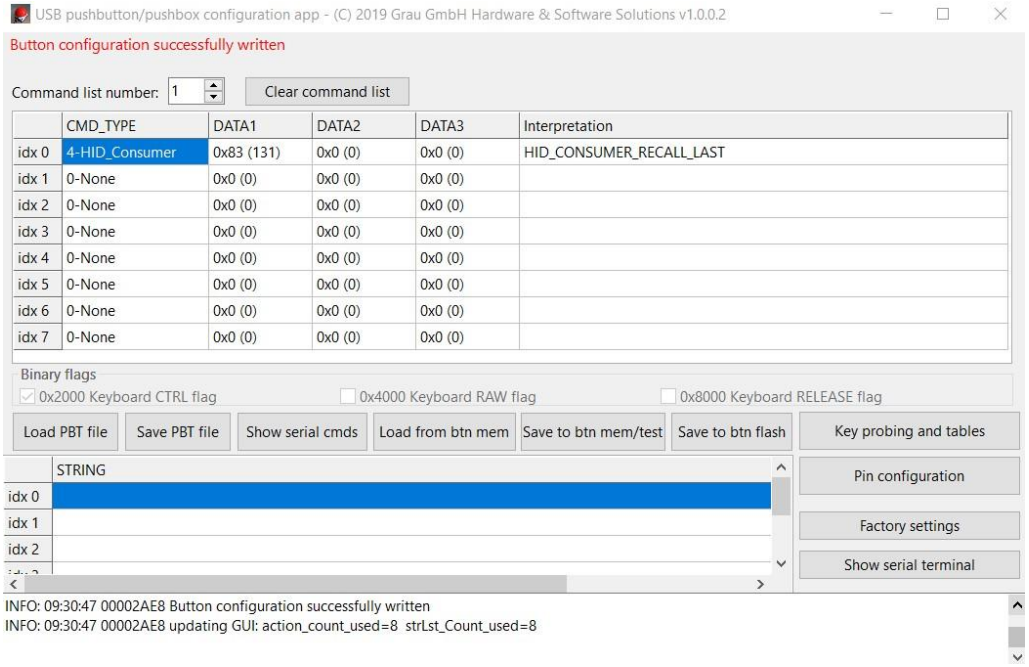


Screen 34: Failed trial HID Consumer Sleep key (depending on computer's file system and keyboard)

Tips:

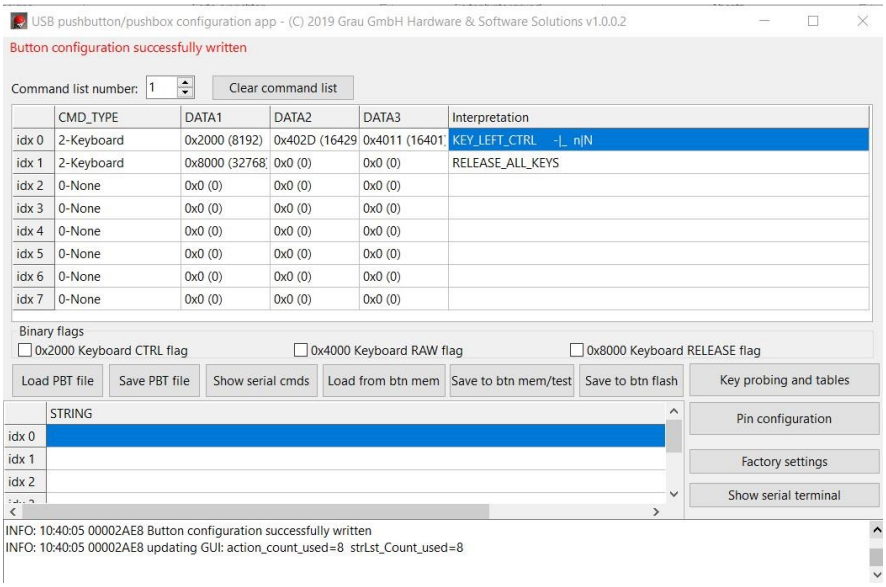
- It can happen that your GUI displays a different operation in the ***Interpretation*** column (like in the next example) although you entered the correct key combinations or codes. The HID key codes you selected can also be ***Unassigned***. This is not a GUI error – your keyboard simply might not dispose of such a key or the key is located at a different position, depending on your operating system.
- Check if you have selected the correct keyboard layout and code tablea under Key probing and tables.
- If your keyboard does not include a certain HID consumer button, also check and adapt your computer's energy management and try a work-around solution. For example, you can search for alternative keyboard shortcuts for Windows or Mac.

Waking up the system via HID key



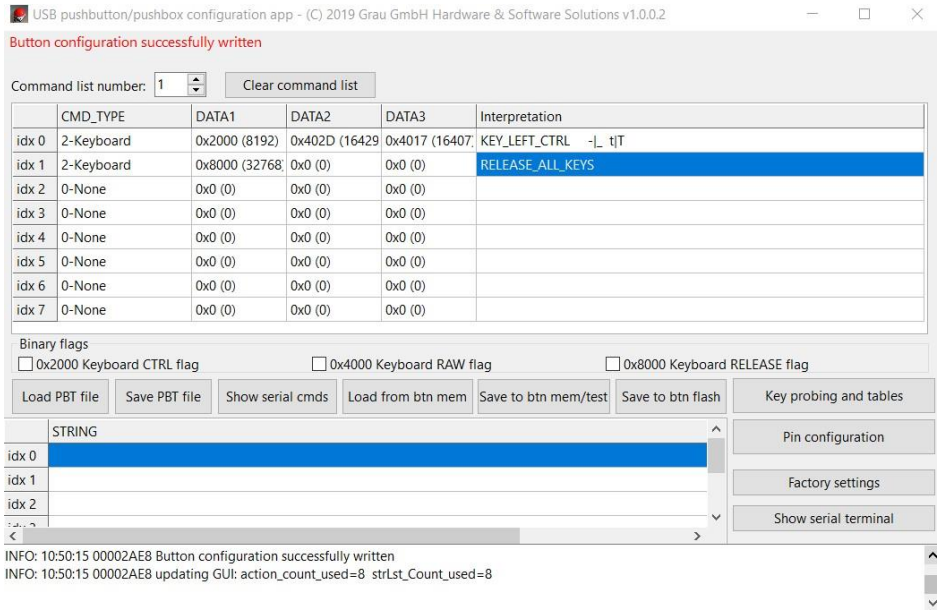
Screen 35: Enabling the HID Wake up key (depending on computer's file system and keyboard)

Opening a new window



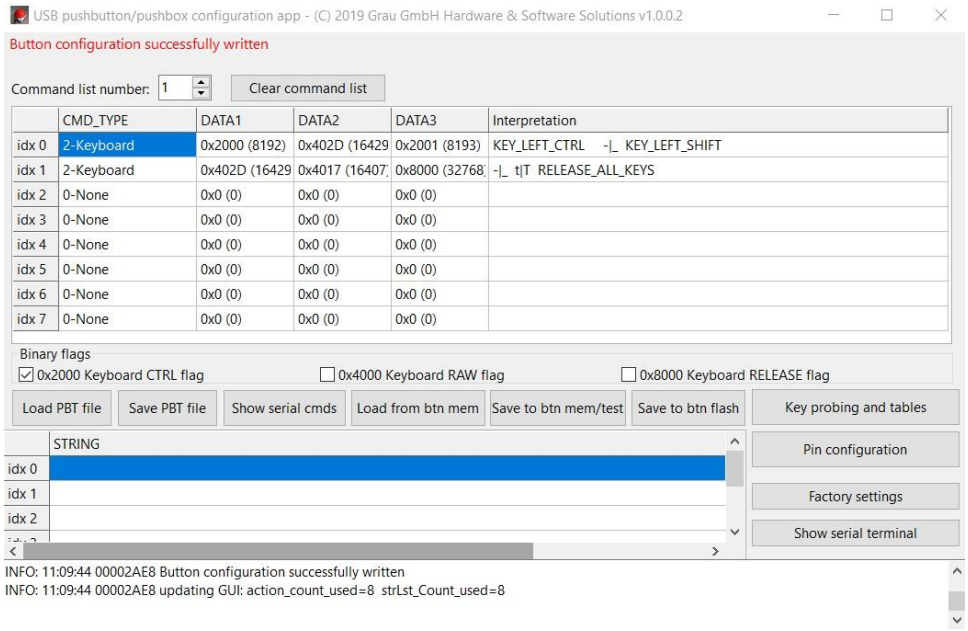
Simulating the key combination "Ctrl +-+ n"

Opening a new tab



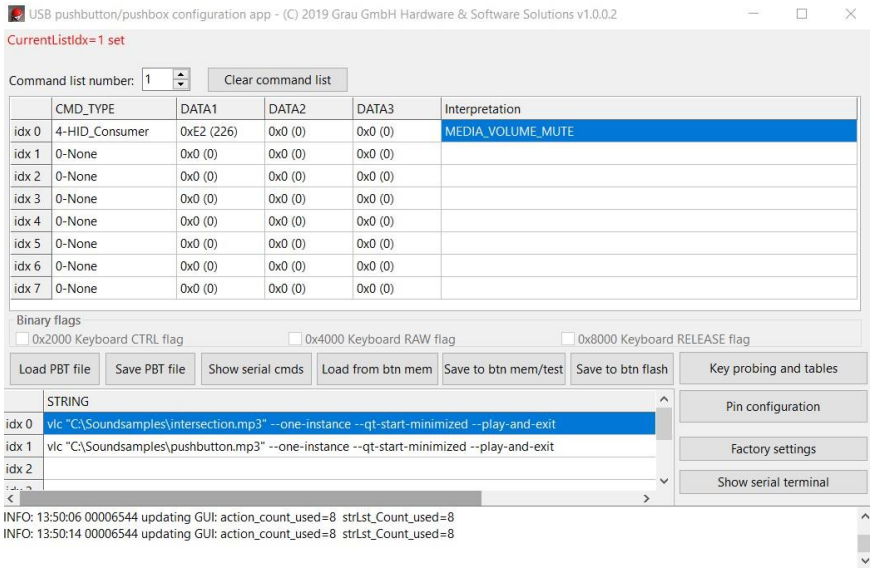
Simulating the key combination "Ctrl + - + t"

Re-opening a tab in Firefox



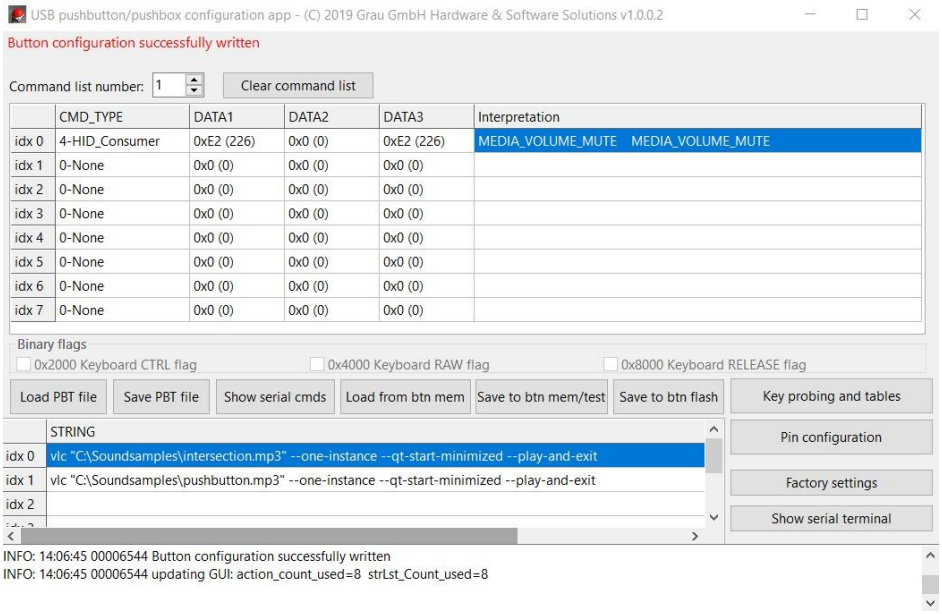
Simulating the key combination “Ctrl-Shift-t”

Muting and unmuting sound (1st/2nd trigger toggle)



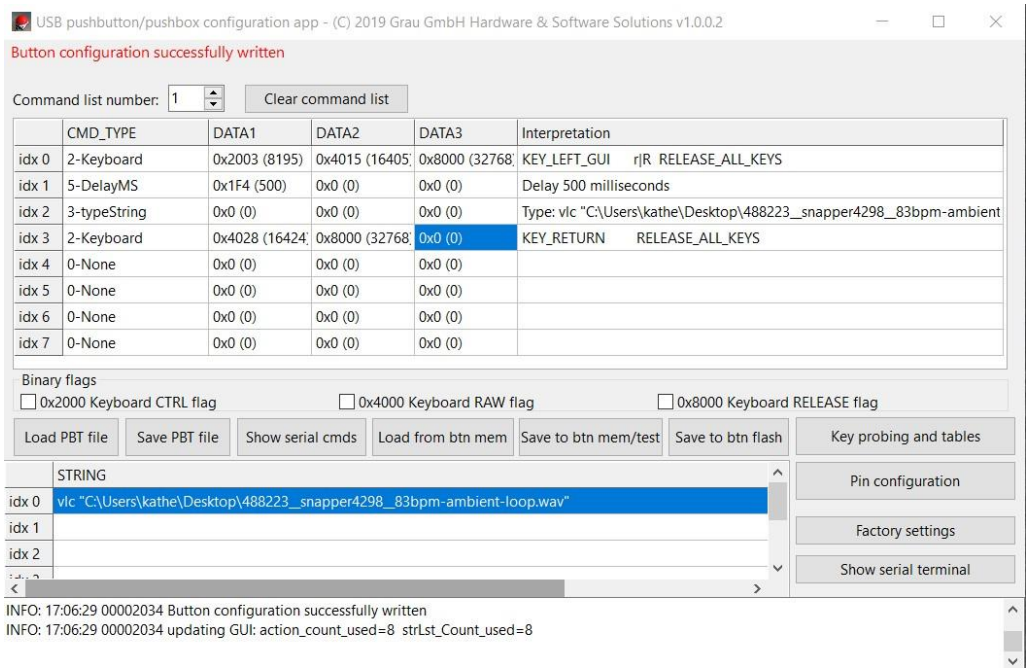
Simulating the key "HID Consumer mute"

Sound pause on button hold (mute on trigger, unmute on release)

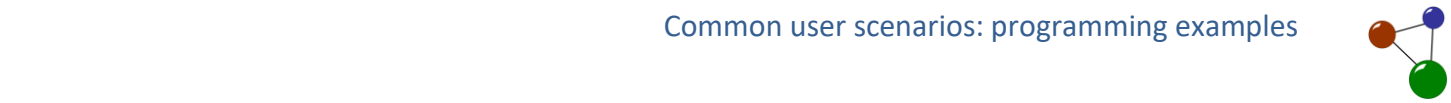


Screen 36: Simulating the key "HID Consumer mute" twice for a sound pause

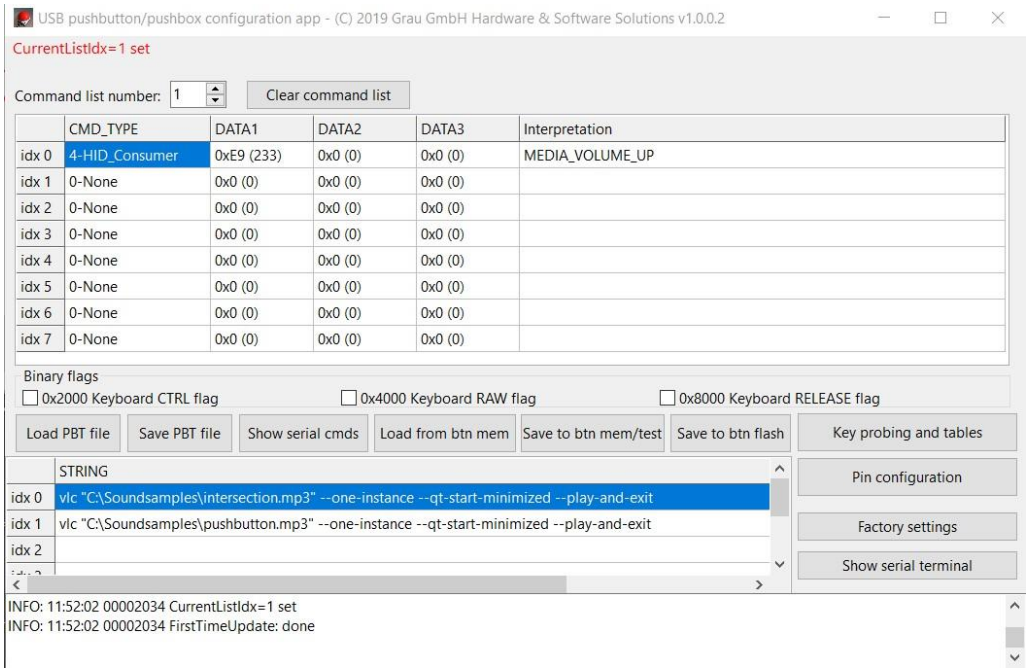
Playing and replaying sound with VLC



Screen 37: Simulating the key and string combination "Windows key+ r; 500ms delay; VLC "soundpath"; Enter; change command list to 2; p in command list 2"

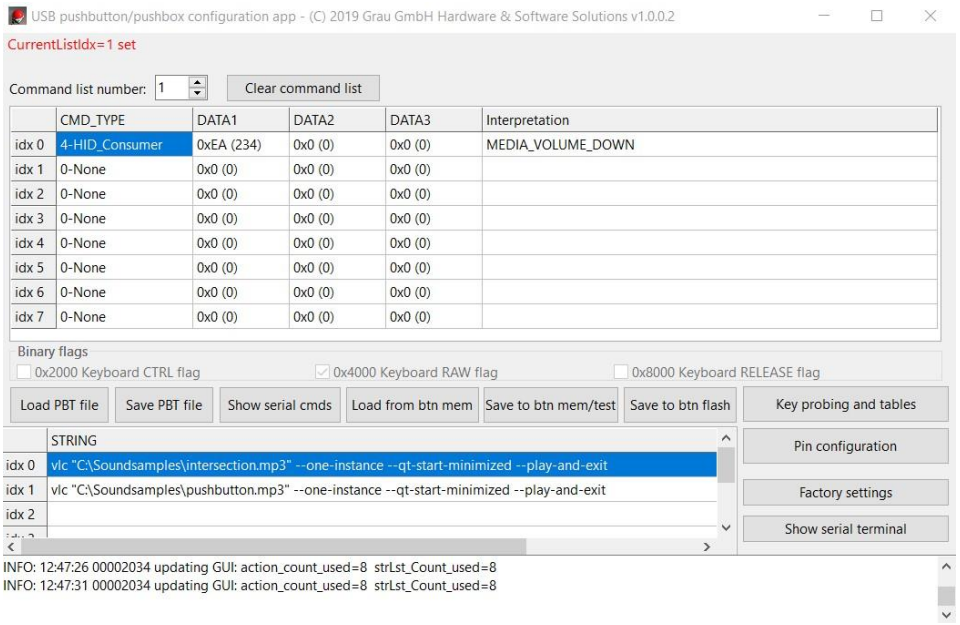


Increasing sound volume (with repetition on hold)



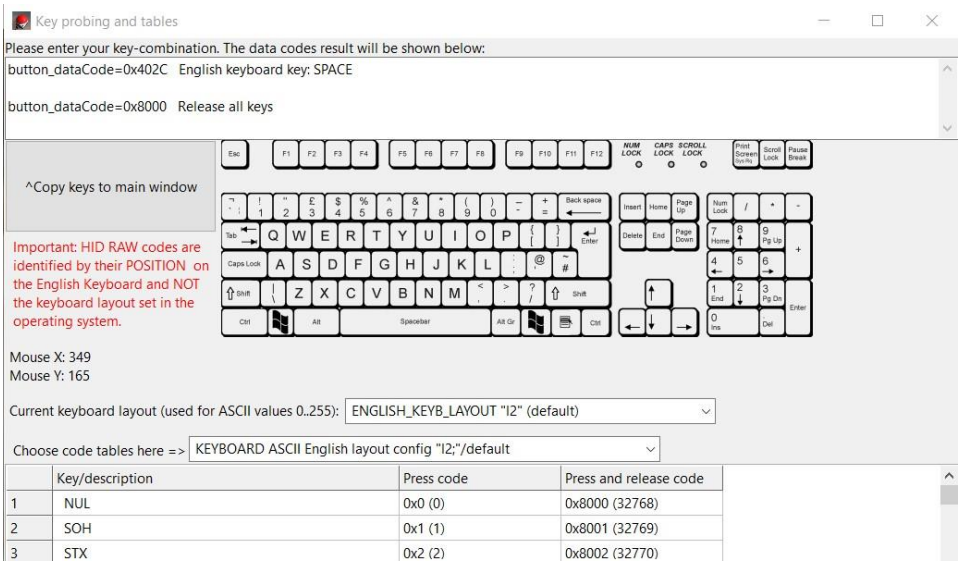
Screen 38: Simulating the key "HID Consumer Volume Increment"

Decreasing sound volume (with repetition on hold)



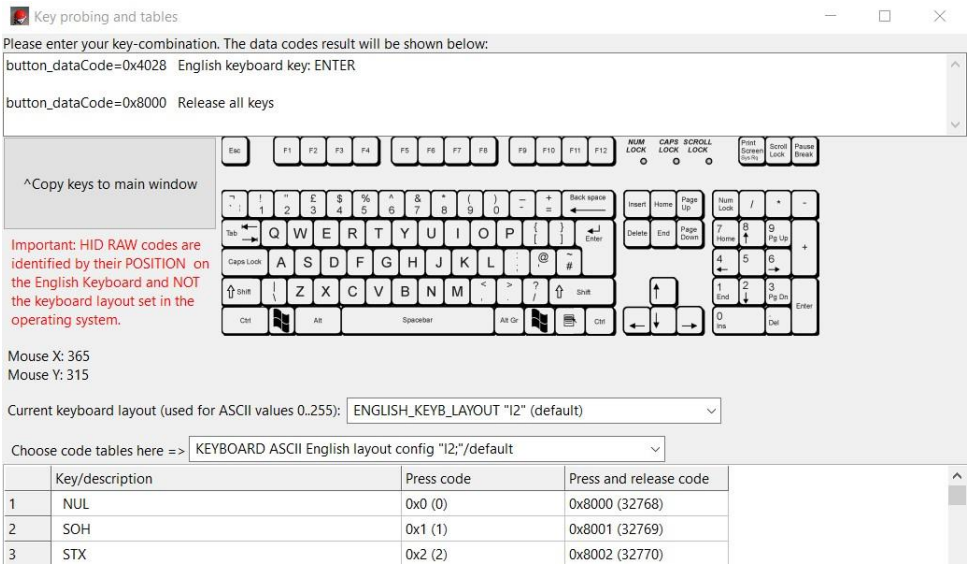
Screen 39: Simulating the key "HID Consumer Volume Decrement"

Starting presentations and switching slides in Power Point – option 1



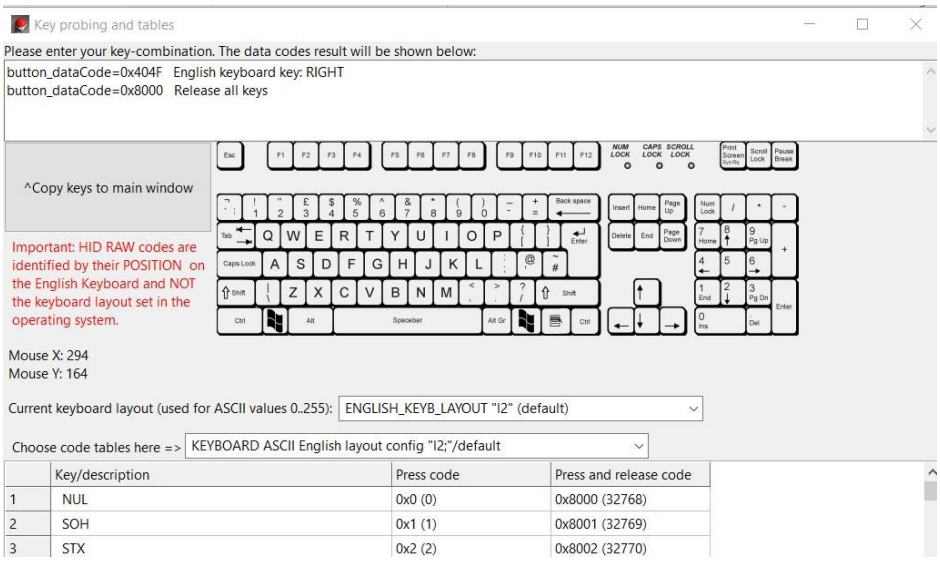
Screen 40: Simulating the key "Space"

Starting presentations or switching slides in PowerPoint – option 2



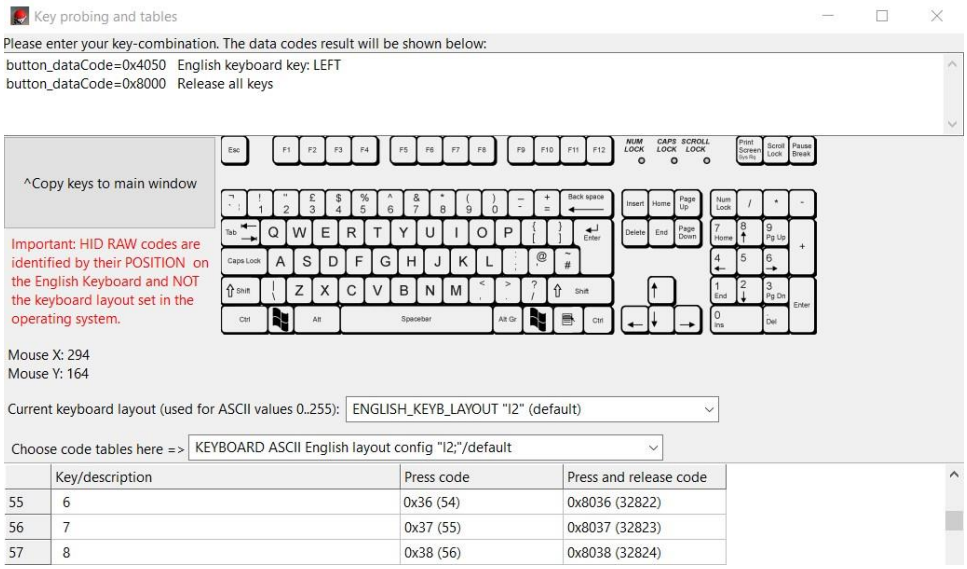
Screen 41: Simulating the key "Return/Enter"

Moving forward in a PowerPoint presentation



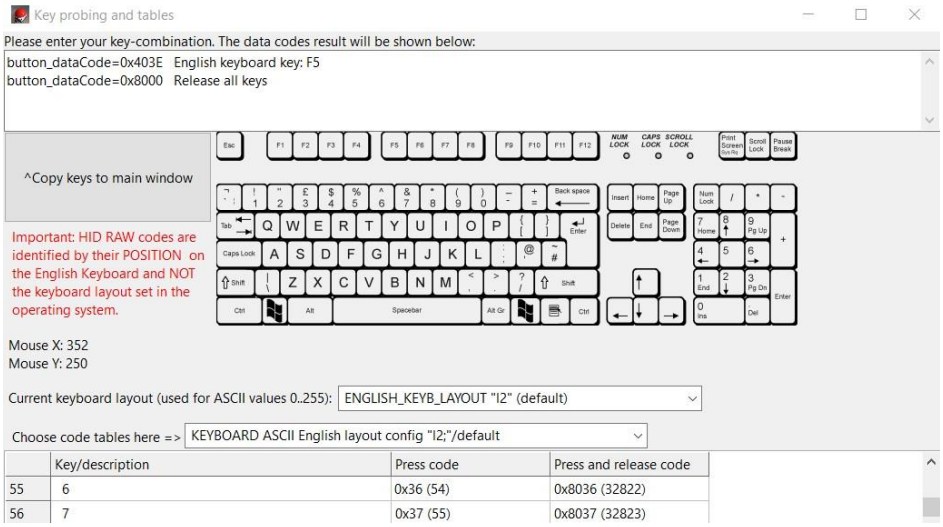
Screen 42: Simulating the key "Arrow right"

Moving backwards in a PowerPoint presentation



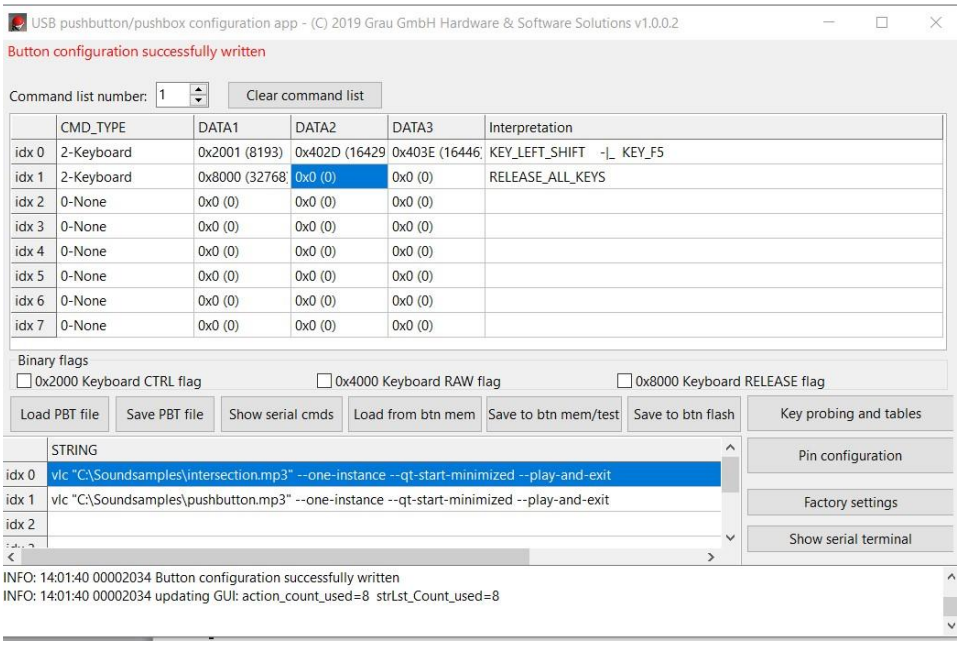
Screen 43: Simulating the key "Arrow left"

Starting or restarting a presentation in PowerPoint



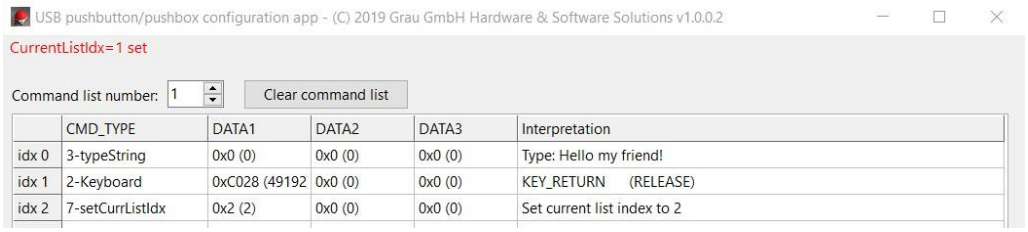
Screen 44: Simulating the key "F5"

Starting a PowerPoint presentation from the current slide

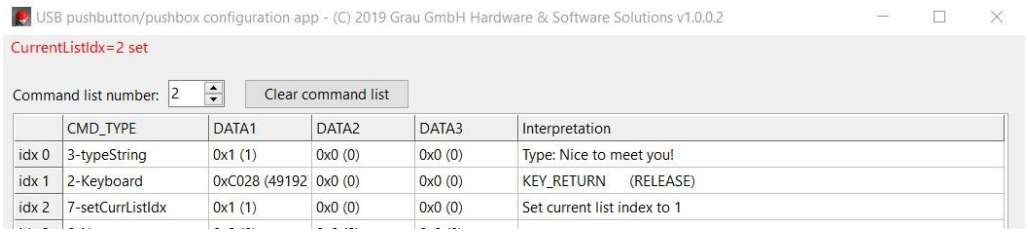


Screen 45: Simulating the key combination "Shift+-+F5"

Toggling two strings



Screen 46: Toggling two typed strings which were programmed in the integrated serial terminal (table view)l



Tips:

- You can save time and programming efforts if you use an existing pushbutton template. A pushbutton template is a functioning sample configuration for your USB module. All available pushbutton templates are located under ***Load PBT file***.
- If you want to use a certain configuration more often, you can also save it as a template under ***Save PBT file***. Doing this, you make the new pushbutton template available for further user scenarios.

6.2 Programming examples via command lines

Operation type or command type	Module operation, mouse clicks or key combination	Configuration via command lines
Serial commands	show all command- and string lists	?;
	save all lists to EEPROM	s; / save;
	show short user manual/ user guide	h; / help;
	reset to factory settings (includes saving to EEPROM)	factory;
	echo on/off	e1; / e0;
	clear command list no. 0-7	c0;.c7;
	set default button state polling interval in ms (default=100)	bsp=xxx;
	keyboard layout 0-2: 0=GERMAN_LATIN1, 1=GERMAN_CP1252, 2=ENGLISH	l0;..l2;
	store string number	str0=MyString;
	set single command at index 0 in command list	!1,0,x,x,x,x;

Practical user scenarios via serial terminal

	number 1 (default list)	
	command list syntax	
Structure of command rows and command lists	Attention: programming a “command row”, which is a single row/entry in a command list	!
	Programming command list number 1 (which is the default execution list). You can program command list 0..7. Each list can contain up to 8 execution command rows (indexed from 0..7). If the module’s button is pressed all rows in the command list are executed from index 0 to 7 (if they contain any valid command rows).	1 (or other number 2 -7)
	Programming command row/index number 0 (=the first one)	0
	Programming a HID-KEYBOARD-KEY command. Other command types are shown below.	2
	Complete list of all ASCII characters with their decimal/hex codes is here. The module accepts decimal (97) as well as the hex value (0x61) number format.	97 (example for ASCII code)
	zero padding/ no further keys shall be pressed	0,0
	Command row completed, you must the button process and verify your given command row.	;
	Saving the current settings/programming to EEPROM (=internal	s;

	permanent flash memory)	
	Clearing whole command lists (0-7): If you do not save your changes with “s;”, you can undo your clear command by simply un- and replugging the USB connector.	“c0;”..”c7;” for a selection of command rows, “c1;” for all command rows
	Deleting single command rows with zeros	e.g. !1,3,0,0,0,0; => (row number 3 as all lists start with index 0)
	Parallel key strokes, e.g. CTRL+ SHIFT+ESC (opening the Windows task manager shortcut) or CTRL+SHIFT+ALT+BACKSPACE	e.g. !1,0,2,0×2000,0x2001,0x4029; or c1; !1,0,2,0×2000,0x2001,0x2002; !1,1,2,0x402A,0x8000,0;
	Show all lists/get the complete configuration	beginning with ?; example: isON=1 echo=1 currListIdx=1 bsp=100 soloBtnIdx=-1 keyb_layout: 1 – GERMAN_KEYB_LAYOUT_WINDOWS_CP1252
Physical pin configuration	General syntax of physical pin configuration	@[listindex], [physPin], [pinmode], [bitflags], [listTrigger], [listRepeat], [listRelease], [repeatDelayMS], [treshold_1];
	Execute command list 1 on button press, execute command list 2 on button hold, execute command list 3 on button release	@1,19,2,0,1,-1,3,400,500; (for three different chronological operations)
	Execute command list 1 on trigger and allow changing the currListIdx via CMDTYPE_7	e.g. @1,19,2,0,127,-1,-1,400,500; s;
	ButtonStatePolling-Interval/ fast key repetition (only	e.g. @1,19,2,0,127,127,-1,50,500; bsp=50; s;

	pushbtton)	
	Default pin configurations (pushbox)	@0,18,2,32,1,-1,-1,400,500; @1,19,2,32,2,-1,-1,400,500; @2,20,2,32,3,-1,-1,400,500; @3,21,2,32,4,-1,-1,400,500; s;
Delayed command executions	waiting 300 msec. in action list list number 1, row 1 (5=command 'delayMicroseconds')	!1,1,5,300,0,0;
	waiting 3 sec. in action list number 1, row 2 (6=command 'delaySeconds')	!1,2,6,3,0,0;,
Mouse emulation	Command line syntax for the simulation of mouse-clicks	![list],[index],8, [x], [y], [buttonclick];
	Left mouse button click upper right corner / close window.	c1; !1,0,8,32256,256,1;
Testing screen coordinates (0---32768, x-axis and y-axis)	navigate x axis right – press the button to move navigate x axis left – press the button to move navigate y axis down – press the button to move navigate y axis up – press the button to move make larger/smaller steps in x/y stop test mode	mx+; mx-; my+; my-; ms+; / ms-; m0;
Keyboard layouts	German Windows CP1252 keyboard layout (set this for a German Windows OS). for German DOS-Latin-1 keyboard layout English keyboard layout	l1;, l0;, l2;

	(default setting)	
General functions for Windows operating systems	Opening Windows Task Manager (CTRL+SHIFT+ESC)	!1,0,2,0×2000,0x2001,0x4029;
	Cycling Windows (CTRL+ALT+ESC)	!1,0,2,0×2000,0x2002,0x4029;
	Three single Enter (with instant release)	!1,0,2,0xC028,0xC028,0xC028;
	left operating system key (hold)	!1,0,2,0×2003,0,0;
	Windows toggle desktop/app (Windows key + d)	!1,0,2,0×2003,100,0;
	Windows logout (Windows key + l)	!1,0,2,0×2003,108,0;
	Windows open disk management (Windows key + r ; 500ms delay ; diskmgmt.msc ; ENTER)	!1,0,2,0×2003,0x15,0x8000; !1,1,5,500,0,0; str0=diskmgmt.msc; !1,2,3,0,0,0; !1,3,2,0xC028,0,0;
	Windows open skype (OS-Key + r ; 500ms delay ; <skypePath> ; ENTER)	!1,0,2,0×2003,0x15,0x8000; !1,1,5,500,0,0; str1=”C:\Program Files (x86)\Microsoft\Skype for Desktop\Skype.exe”; !1,2,3,1,0,0; !1,3,2,0×4028,0,0;
System power, standby and wake functions	HID System/Consumer sleep	!1,0,1,0×82,0,0; !1,1,4,0×32,0,0;
	HID System wake up	!0,0,1,0×83,0,0;
	HID System/Consumer powerdown	!1,0,1,0×81,0,0; !1,1,4,0×30,0,0;
	OS-X powerdown	!1,0,2,0×2000,0,0;!1,1,1,0×81,0xffff,0;!1,2,5,500,0,0;!1,3,2,0xC028,0,0;

	OS-X standby/wake	!1,0,2,0×2000,0x2001,0; !1,1,1,0×81,0,0;!1,2,7,0, 0,0;!0,0,2,0×2000,0,0;!0, 1,7,1,0,0;
Browser functions	Opening a new window (CTRL-n)	!1,0,2,0×2000,110,0;
	Opening a new tab (CTRL-t)	!1,0,2,0×2000,116,0;
	Re-open the last tab in Firefox (CTRL-SHIFT-t)	!1,0,2,0×2000,0x2001,1 16;
Sound functions	Toggling: mute/unmute sound (1st/2nd trigger toogle)	!1,0,4,0xE2,0,0;
	Sound pause on button hold/ DJ break function (mute on trigger, unmute on release)	c1;c3; @1,19,2,0,1,-1,3,400,500; !1,0,4,0xE2,0,0; !3,0,4,0x E2,0,0;
	Playing and replaying sound with VLC: (OS key+ r; 500ms delay; VLC <soundpath>; Enter; change command list to 2; rewind in VLC with p)	!1,0,2,0×2003,0x15,0x80 00; !1,1,5,500,0,0; str0=vlc “C:\Soundsamples\push button.mp3”; !1,2,3,0,0,0; !1,3,2,0×40 28,0,0; !1,4,7,2,0,0; !2,0, 2,112,0,0;
	Volume step up/increase (with repetition on hold)	!1,0,4,0xE9,0,0; @1,19,2 ,0,127,127,-1,400,500;
	Volume step down/decrease (with repetition on hold)	!1,0,4,0xEA,0,0; @1,19,2 ,0,127,127,-1,400,500;
PowerPoint functions	Space key	!1,0,2,32,0,0;
	Return/Enter key	!1,0,2,0×4028,0,0;
	Arrow right	!1,0,2,0x404F,0,0;
	Arrow left	!1,0,2,0×4050,0,0;
	F5 to (re-)start a presentation	!1,0,2,0x403E,0,0;

	Shift-F5 to start a presentation from the current slide	!1,0,2,0×2001,0x403E,0;
Toggling strings	Toggling two strings	c1;c2; @1,19,2,0,127,-1,-1,400,500; str0=Hello my friend!; !1,0,3,0,0,0; !1,1,2,0xC028,0,0; !1,2,7,2,0,0; str1=Nice to meet you!; !2,0,3,1,0,0; !2,1,2,0xC028,0,0; !2,2,7,1,0,0;

Tip: Always test the programmed functions in your destination application by pushing one or several buttons. Add delays if necessary to allow the computer to perform the programmed operation. After programming, save to flash memory with ”s”. Then unplug and re-plug the USB module in order to test its new configurations.

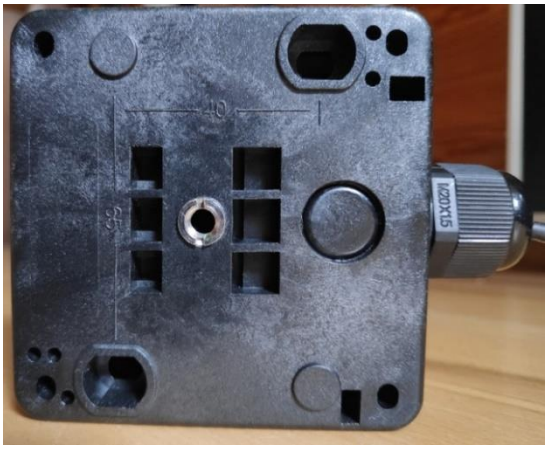
7. Technical data and features

Technical data: USB pushbutton

USB pushbutton	
Full name of product line:	USB Pushbutton / Mushroom buzzer (PBT-series, HID, programmable)
Standards applied:	IEC/EN 60947, VDE 0660, IEC/EN 60259; protection class: IP 67, IP69K
Size of bottom case in mm(length, width, height):	84mm x 84mm x 96mm
Diameter of mushroom cap:	93mm
Weight of pushbutton:	0,35kg
Length of USB connection cable	280cm, 480cm
Available colors for mushroom cap:	red, black, green
Mounting holes:	two available in the bottom case, distance: 40mm x 65mm; mounting of pushbuttons only on a firm and non-vibrating ground
Climate resistance:	humidity: up to 100%, temperature -15°C to 40°C
Lifespan and endurance:	vandalism-protected industrial quality, endures 100kg, minimum lifespan: 1 million switches (with 3600 switches/ h)
WEEE number:	DE87085967
German tax tariff number:	85365019
Compatible file systems:	all versions of Windows, Apple IOS, Linux, Android
Supported keyboard layouts:	English, German, Latin1/CP1252 (other layouts on demand)
Maximum number of parallel simulations	up to 6 keyboard keys in parallel (maximum of 6×8 = 48 keystrokes in sequence)
Technology included	USB board, internal flash memory, internal USB serial

in the product (hardware/software):	interface with terminal emulation, app/GUI as an auxiliary tool for programming
Required hardware drivers:	none

Screen 47: front view of pushbutton with USB connection cable



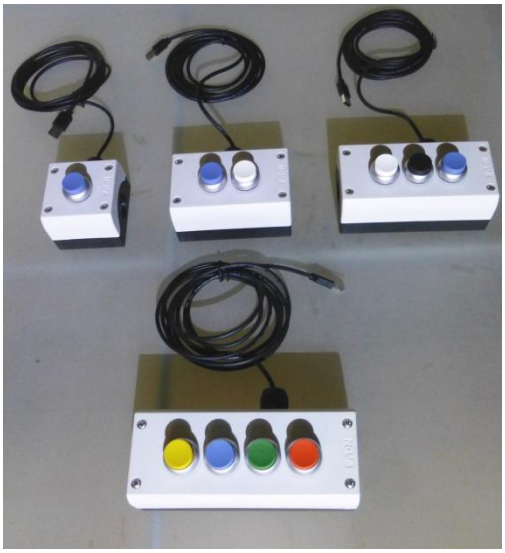
Screen 48: bottom view of pushbutton with mounting holes

Technical data: USB pushbox

USB pushbox	
Full name of product line:	USB Pushbox (PBL-series, HID, programmable)
Standards applied:	IEC/EN 60947-5; UL 508; CSA-C22.2 No. 14-05; CSA-C22.2 No. 94-91,CE, RoHS
Protection class:	IP66, IP67, IP69K ; UL/CSA Type 3R, 4X, 12, 13
UL numbers:	UL file No. E29184, UL category control No. NKCR
CSA certification:	CSA file No. 012528, CSA class No. 3211-03
WEEE Number:	92318050
German tax tariff number:	85381000
Size of bottom case in mm(length, width, height):	80,00 x 72,00 x 56,00 mm

Material and design of buttons and case:	buttons: RMQ-titan, either embedded or sticking out; case: hard-PVC with stainless steel screws
Weight of a pushbox:	0,3kg
Length of USB cable (cm)	280cm, 480cm
Possible number of buttons:	6
Colors of buttons:	six colors in an individual combination
Mounting holes	four available in the bottom case,more can be added; mounting of pushboxes only on a firm and non-vibrating ground
Climate resistance:	humidity: up to 100%, temperature -15°C to 40°C
Lifespan and endurance:	vandalism-protected industrial quality, endures 100kg, minimum lifespan: 1 million switches (with 3600 switches/ h)
Compatible file systems:	all versions of Windows, Apple IOS, Linux, Android
Supported keyboard layouts:	English, German, Latin1/CP1252 (other layouts on demand)
Maximum number of parallel simulations	up to 6 keyboard keys in parallel (maximum of 6×8 = 48 keystrokes in sequence); all buttons individually programmable
Technology included in the product (hardware/software):	USB board, internal flash memory, internal USB serial interface with terminal emulation, app/GUI as an auxiliary tool for programming
Required hardware drivers:	none

Screen 49: Individual programming for each button in the PBL series



Screen 50: The PBL series with a flexible amount of buttons

8. Index