

---

# Adafruit HID Library Documentation

*Release 1.0*

**Scott Shawcroft**

**Jul 27, 2021**



---

## Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Usage Example</b>	<b>5</b>
<b>3</b>	<b>Contributing</b>	<b>9</b>
<b>4</b>	<b>Documentation</b>	<b>11</b>
<b>5</b>	<b>Table of Contents</b>	<b>13</b>
5.1	Simple test . . . . .	13
5.2	Keyboard Shortcuts . . . . .	14
5.3	Simple Gamepad . . . . .	14
5.4	HID Joywing . . . . .	15
5.5	Consumer Control Brightness . . . . .	17
5.6	adafruit_hid.keyboard.Keyboard . . . . .	18
5.7	adafruit_hid.keycode.Keycode . . . . .	19
5.8	adafruit_hid.keyboard_layout_us.KeyboardLayoutUS . . . . .	26
5.9	adafruit_hid.mouse.Mouse . . . . .	26
5.10	adafruit_hid.consumer_control.ConsumerControl . . . . .	28
5.11	adafruit_hid.consumer_control_code.ConsumerControlCode . . . . .	28
<b>6</b>	<b>Indices and tables</b>	<b>31</b>
	<b>Python Module Index</b>	<b>33</b>
	<b>Index</b>	<b>35</b>



This driver simulates USB HID devices. Currently keyboard and mouse are implemented.



# CHAPTER 1

---

## Dependencies

---

This driver depends on:

- [Adafruit CircuitPython](#)

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the [Adafruit library and driver bundle](#).





## CHAPTER 2

---

### Usage Example

---

The Keyboard class sends keypress reports for a USB keyboard device to the host.

The Keycode class defines USB HID keycodes to send using Keyboard.

```
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode

# Set up a keyboard device.
kbd = Keyboard(usb_hid.devices)

# Type lowercase 'a'. Presses the 'a' key and releases it.
kbd.send(Keycode.A)

# Type capital 'A'.
kbd.send(Keycode.SHIFT, Keycode.A)

# Type control-x.
kbd.send(Keycode.CONTROL, Keycode.X)

# You can also control press and release actions separately.
kbd.press(Keycode.CONTROL, Keycode.X)
kbd.release_all()

# Press and hold the shifted '1' key to get '!' (exclamation mark).
kbd.press(Keycode.SHIFT, Keycode.ONE)
# Release the ONE key and send another report.
kbd.release(Keycode.ONE)
# Press shifted '2' to get '@'.
kbd.press(Keycode.TWO)
# Release all keys.
kbd.release_all()
```

The KeyboardLayoutUS sends ASCII characters using keypresses. It assumes the host is set to accept keypresses from a US keyboard.

If the host is expecting a non-US keyboard, the character to key mapping provided by `KeyboardLayoutUS` will not always be correct. Different keypresses will be needed in some cases. For instance, to type an 'A' on a French keyboard (AZERTY instead of QWERTY), `Keycode.Q` should be pressed.

Currently this package provides only `KeyboardLayoutUS`. More `KeyboardLayout` classes could be added to handle non-US keyboards and the different input methods provided by various operating systems.

```
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS

kbd = Keyboard(usb_hid.devices)
layout = KeyboardLayoutUS(kbd)

# Type 'abc' followed by Enter (a newline).
layout.write('abc\n')

# Get the keycodes needed to type a '$'.
# The method will return (Keycode.SHIFT, Keycode.FOUR).
keycodes = layout.keycodes('$')
```

The `Mouse` class simulates a three-button mouse with a scroll wheel.

```
import usb_hid
from adafruit_hid.mouse import Mouse

m = Mouse(usb_hid.devices)

# Click the left mouse button.
m.click(Mouse.LEFT_BUTTON)

# Move the mouse diagonally to the upper left.
m.move(-100, -100, 0)

# Roll the mouse wheel away from the user one unit.
# Amount scrolled depends on the host.
m.move(0, 0, -1)

# Keyword arguments may also be used. Omitted arguments default to 0.
m.move(x=-100, y=-100)
m.move(wheel=-1)

# Move the mouse while holding down the left button. (click-drag).
m.press(Mouse.LEFT_BUTTON)
m.move(x=50, y=20)
m.release_all()          # or m.release(Mouse.LEFT_BUTTON)
```

The `ConsumerControl` class emulates consumer control devices such as remote controls, or the multimedia keys on certain keyboards.

```
import usb_hid
from adafruit_hid.consumer_control import ConsumerControl
from adafruit_hid.consumer_control_code import ConsumerControlCode

cc = ConsumerControl(usb_hid.devices)

# Raise volume.
cc.send(ConsumerControlCode.VOLUME_INCREMENT)
```

(continues on next page)

(continued from previous page)

```
# Pause or resume playback.  
cc.send(ConsumerControlCode.PLAY_PAUSE)
```



## CHAPTER 3

---

### Contributing

---

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.



## CHAPTER 4

---

### Documentation

---

For information on building library documentation, please check out [this guide](#).





## 5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/hid\_simpletest.py

```
1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  import time
5  import board
6  import digitalio
7  import usb_hid
8  from adafruit_hid.mouse import Mouse
9
10 mouse = Mouse(usb_hid.devices)
11
12 # define buttons. these can be any physical switches/buttons, but the values
13 # here work out-of-the-box with a CircuitPlayground Express' A and B buttons.
14 up = digitalio.DigitalInOut(board.D4)
15 up.direction = digitalio.Direction.INPUT
16 up.pull = digitalio.Pull.DOWN
17
18 down = digitalio.DigitalInOut(board.D5)
19 down.direction = digitalio.Direction.INPUT
20 down.pull = digitalio.Pull.DOWN
21
22 while True:
23     # scroll up one unit (varies with host/OS)
24     if up.value:
25         mouse.move(wheel=1)
26
27     # scroll down one unit (varies with host/OS)
```

(continues on next page)

(continued from previous page)

```

28     elif down.value:
29         mouse.move(wheel=-1)
30
31     time.sleep(0.1)

```

## 5.2 Keyboard Shortcuts

Send ALT+Tab for swapping windows, and CTRL+K for searching in a browser.

Listing 2: examples/hid\_keyboard\_shortcuts.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  import time
5  import board
6  import digitalio
7  import usb_hid
8  from adafruit_hid.keyboard import Keyboard
9  from adafruit_hid.keycode import Keycode
10
11 kbd = Keyboard(usb_hid.devices)
12
13 # define buttons. these can be any physical switches/buttons, but the values
14 # here work out-of-the-box with a CircuitPlayground Express' A and B buttons.
15 swap = digitalio.DigitalInOut(board.D4)
16 swap.direction = digitalio.Direction.INPUT
17 swap.pull = digitalio.Pull.DOWN
18
19 search = digitalio.DigitalInOut(board.D5)
20 search.direction = digitalio.Direction.INPUT
21 search.pull = digitalio.Pull.DOWN
22
23 while True:
24     # press ALT+TAB to swap windows
25     if swap.value:
26         kbd.send(Keycode.ALT, Keycode.TAB)
27
28     # press CTRL+K, which in a web browser will open the search dialog
29     elif search.value:
30         kbd.send(Keycode.CONTROL, Keycode.K)
31
32     time.sleep(0.1)

```

## 5.3 Simple Gamepad

Send gamepad buttons and joystick to the host.

Listing 3: examples/hid\_simple\_gamepad.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT

```

(continues on next page)

(continued from previous page)

```

3
4 import board
5 import digitalio
6 import analogio
7 import usb_hid
8
9 from gamepad import Gamepad
10
11 gp = Gamepad(usb_hid.devices)
12
13 # Create some buttons. The physical buttons are connected
14 # to ground on one side and these and these pins on the other.
15 button_pins = (board.D2, board.D3, board.D4, board.D5)
16
17 # Map the buttons to button numbers on the Gamepad.
18 # gamepad_buttons[i] will send that button number when buttons[i]
19 # is pushed.
20 gamepad_buttons = (1, 2, 8, 15)
21
22 buttons = [digitalio.DigitalInOut(pin) for pin in button_pins]
23 for button in buttons:
24     button.direction = digitalio.Direction.INPUT
25     button.pull = digitalio.Pull.UP
26
27 # Connect an analog two-axis joystick to A4 and A5.
28 ax = analogio.AnalogIn(board.A4)
29 ay = analogio.AnalogIn(board.A5)
30
31 # Equivalent of Arduino's map() function.
32 def range_map(x, in_min, in_max, out_min, out_max):
33     return (x - in_min) * (out_max - out_min) // (in_max - in_min) + out_min
34
35
36 while True:
37     # Buttons are grounded when pressed (.value = False).
38     for i, button in enumerate(buttons):
39         gamepad_button_num = gamepad_buttons[i]
40         if button.value:
41             gp.release_buttons(gamepad_button_num)
42             print(" release", gamepad_button_num, end="")
43         else:
44             gp.press_buttons(gamepad_button_num)
45             print(" press", gamepad_button_num, end="")
46
47     # Convert range[0, 65535] to -127 to 127
48     gp.move_joysticks(
49         x=range_map(ax.value, 0, 65535, -127, 127),
50         y=range_map(ay.value, 0, 65535, -127, 127),
51     )
52     print(" x", ax.value, "y", ay.value)

```

## 5.4 HID Joywing

Use Joy FeatherWing to drive Gamepad.

Listing 4: examples/hid\_joywing\_gamepad.py

```
1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  # Use Joy FeatherWing to drive Gamepad.
5  # https://www.adafruit.com/product/3632
6  # https://learn.adafruit.com/joy-featherwing
7
8  import time
9
10 import board
11 import busio
12 from micropython import const
13 import adafruit_seesaw
14 import usb_hid
15 from gamepad import Gamepad
16
17
18 def range_map(value, in_min, in_max, out_min, out_max):
19     return (value - in_min) * (out_max - out_min) // (in_max - in_min) + out_min
20
21
22 BUTTON_RIGHT = const(6)
23 BUTTON_DOWN = const(7)
24 BUTTON_LEFT = const(9)
25 BUTTON_UP = const(10)
26 BUTTON_SEL = const(14)
27 button_mask = const(
28     (1 << BUTTON_RIGHT)
29     | (1 << BUTTON_DOWN)
30     | (1 << BUTTON_LEFT)
31     | (1 << BUTTON_UP)
32     | (1 << BUTTON_SEL)
33 )
34
35 i2c = busio.I2C(board.SCL, board.SDA)
36
37 ss = adafruit_seesaw.Seesaw(i2c)
38
39 ss.pin_mode_bulk(button_mask, ss.INPUT_PULLUP)
40
41 last_game_x = 0
42 last_game_y = 0
43
44 g = Gamepad(usb_hid.devices)
45
46 while True:
47     x = ss.analog_read(2)
48     y = ss.analog_read(3)
49
50     game_x = range_map(x, 0, 1023, -127, 127)
51     game_y = range_map(y, 0, 1023, -127, 127)
52     if last_game_x != game_x or last_game_y != game_y:
53         last_game_x = game_x
54         last_game_y = game_y
55         print(game_x, game_y)
```

(continues on next page)

(continued from previous page)

```

56         g.move_joysticks(x=game_x, y=game_y)
57
58     buttons = (BUTTON_RIGHT, BUTTON_DOWN, BUTTON_LEFT, BUTTON_UP, BUTTON_SEL)
59     button_state = [False] * len(buttons)
60     for i, button in enumerate(buttons):
61         buttons = ss.digital_read_bulk(button_mask)
62         if not (buttons & (1 << button) and not button_state[i]):
63             g.press_buttons(i + 1)
64             print("Press", i + 1)
65             button_state[i] = True
66         elif button_state[i]:
67             g.release_buttons(i + 1)
68             print("Release", i + 1)
69             button_state[i] = False
70
71     time.sleep(0.01)

```

## 5.5 Consumer Control Brightness

Send brightness up and down consumer codes to the host.

Listing 5: examples/hid\_consumer\_control\_brightness.py

```

1  # SPDX-FileCopyrightText: 2021 Tim C for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  import time
5  import board
6  import digitalio
7  import usb_hid
8  from adafruit_hid.consumer_control import ConsumerControl
9  from adafruit_hid.consumer_control_code import ConsumerControlCode
10
11  cc = ConsumerControl(usb_hid.devices)
12
13  # define buttons. these can be any physical switches/buttons, but the values
14  # here work out-of-the-box with a FunHouse UP and DOWN buttons.
15  button_up = digitalio.DigitalInOut(board.BUTTON_UP)
16  button_up.switch_to_input(pull=digitalio.Pull.DOWN)
17
18  button_down = digitalio.DigitalInOut(board.BUTTON_DOWN)
19  button_down.switch_to_input(pull=digitalio.Pull.DOWN)
20
21  while True:
22      if button_up.value:
23          print("Button up pressed!")
24          # send brightness up button press
25          cc.send(ConsumerControlCode.BRIGHTNESS_INCREMENT)
26
27      if button_down.value:
28          print("Button down pressed!")
29          # send brightness down button press
30          cc.send(ConsumerControlCode.BRIGHTNESS_DECREMENT)
31

```

(continues on next page)

```
time.sleep(0.1)
```

## 5.6 adafruit\_hid.keyboard.Keyboard

- Author(s): Scott Shawcroft, Dan Halbert

**class** `adafruit_hid.keyboard.Keyboard` (*devices*)  
Send HID keyboard reports.

**LED\_CAPS\_LOCK** = 2  
LED Usage ID for Caps Lock

**LED\_COMPOSE** = 8  
LED Usage ID for Compose

**LED\_NUM\_LOCK** = 1  
LED Usage ID for Num Lock

**LED\_SCROLL\_LOCK** = 4  
LED Usage ID for Scroll Lock

**led\_on** (*led\_code*)  
Returns whether an LED is on based on the led code

Examples:

```
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode
import time

# Initialize Keyboard
kbd = Keyboard(usb_hid.devices)

# Press and release CapsLock.
kbd.press(Keycode.CAPS_LOCK)
time.sleep(.09)
kbd.release(Keycode.CAPS_LOCK)

# Check status of the LED_CAPS_LOCK
print(kbd.led_on(Keyboard.LED_CAPS_LOCK))
```

**led\_status**  
Returns the last received report

**press** (*\*keycodes*)  
Send a report indicating that the given keys have been pressed.

**Parameters** **keycodes** – Press these keycodes all at once.

**Raises** **ValueError** – if more than six regular keys are pressed.

Keycodes may be modifiers or regular keys. No more than six regular keys may be pressed simultaneously.

Examples:

```

from adafruit_hid.keycode import Keycode

# Press ctrl-x.
kbd.press(Keycode.LEFT_CONTROL, Keycode.X)

# Or, more conveniently, use the CONTROL alias for LEFT_CONTROL:
kbd.press(Keycode.CONTROL, Keycode.X)

# Press a, b, c keys all at once.
kbd.press(Keycode.A, Keycode.B, Keycode.C)

```

**release** (\*keycodes)

Send a USB HID report indicating that the given keys have been released.

**Parameters** **keycodes** – Release these keycodes all at once.

If a keycode to be released was not pressed, it is ignored.

Example:

```

# release SHIFT key
kbd.release(Keycode.SHIFT)

```

**release\_all** ()

Release all pressed keys.

**send** (\*keycodes)

Press the given keycodes and then release all pressed keys.

**Parameters** **keycodes** – keycodes to send together

## 5.7 adafruit\_hid.keycode.Keycode

- Author(s): Scott Shawcroft, Dan Halbert

**class** adafruit\_hid.keycode.**Keycode**

USB HID Keycode constants.

This list is modeled after the names for USB keycodes defined in [https://usb.org/sites/default/files/hut1\\_21\\_0.pdf#page=83](https://usb.org/sites/default/files/hut1_21_0.pdf#page=83). This list does not include every single code, but does include all the keys on a regular PC or Mac keyboard.

Remember that keycodes are the names for key *positions* on a US keyboard, and may not correspond to the character that you mean to send if you want to emulate non-US keyboard. For instance, on a French keyboard (AZERTY instead of QWERTY), the keycode for ‘q’ is used to indicate an ‘a’. Likewise, ‘y’ represents ‘z’ on a German keyboard. This is historical: the idea was that the keycaps could be changed without changing the keycodes sent, so that different firmware was not needed for different variations of a keyboard.

**A = 4**

a and A

**ALT = 226**

Alias for LEFT\_ALT; Alt is also known as Option (Mac)

**APPLICATION = 101**

Application: also known as the Menu key (Windows)

**B = 5**

b and B

**BACKSLASH = 49**  
  \ and |

**BACKSPACE = 42**  
  Delete backward (Backspace)

**C = 6**  
  c and C

**CAPS\_LOCK = 57**  
  Caps Lock

**COMMA = 54**  
  , and <

**COMMAND = 227**  
  Labeled as Command on Mac keyboards, with a clover glyph

**CONTROL = 224**  
  Alias for LEFT\_CONTROL

**D = 7**  
  d and D

**DELETE = 76**  
  Delete forward

**DOWN\_ARROW = 81**  
  Move the cursor down

**E = 8**  
  e and E

**EIGHT = 37**  
  8 and \*

**END = 77**  
  End (often moves to end of line)

**ENTER = 40**  
  Enter (Return)

**EQUALS = 46**  
  = ` and `` +

**ESCAPE = 41**  
  Escape

**F = 9**  
  f and F

**F1 = 58**  
  Function key F1

**F10 = 67**  
  Function key F10

**F11 = 68**  
  Function key F11

**F12 = 69**  
  Function key F12



**F13 = 104**  
Function key F13 (Mac)

**F14 = 105**  
Function key F14 (Mac)

**F15 = 106**  
Function key F15 (Mac)

**F16 = 107**  
Function key F16 (Mac)

**F17 = 108**  
Function key F17 (Mac)

**F18 = 109**  
Function key F18 (Mac)

**F19 = 110**  
Function key F19 (Mac)

**F2 = 59**  
Function key F2

**F20 = 111**  
Function key F20

**F21 = 112**  
Function key F21

**F22 = 113**  
Function key F22

**F23 = 114**  
Function key F23

**F24 = 115**  
Function key F24

**F3 = 60**  
Function key F3

**F4 = 61**  
Function key F4

**F5 = 62**  
Function key F5

**F6 = 63**  
Function key F6

**F7 = 64**  
Function key F7

**F8 = 65**  
Function key F8

**F9 = 66**  
Function key F9

**FIVE = 34**  
5 and %

**FORWARD\_SLASH = 56**

/ and ?

**FOUR = 33**

4 and \$

**G = 10**

g and G

**GRAVE\_ACCENT = 53**

` and ~

**GUI = 227**

Alias for LEFT\_GUI; GUI is also known as the Windows key, Command (Mac), or Meta

**H = 11**

h and H

**HOME = 74**

Home (often moves to beginning of line)

**I = 12**

i and I

**INSERT = 73**

Insert

**J = 13**

j and J

**K = 14**

k and K

**KEYPAD\_ASTERISK = 85**

Keypad \*

**KEYPAD\_BACKSLASH = 100**

Keypad \ and | (Non-US)

**KEYPAD\_EIGHT = 96**

Keypad 8 and Up Arrow

**KEYPAD\_ENTER = 88**

Keypad Enter

**KEYPAD\_EQUALS = 103**

Keypad = (Mac)

**KEYPAD\_FIVE = 93**

Keypad 5

**KEYPAD\_FORWARD\_SLASH = 84**

Keypad /

**KEYPAD\_FOUR = 92**

Keypad 4 and Left Arrow

**KEYPAD\_MINUS = 86**

Keypad -

**KEYPAD\_NINE = 97**

Keypad 9 and PgUp

**KEYPAD\_NUMLOCK = 83**  
Num Lock (Clear on Mac)

**KEYPAD\_ONE = 89**  
Keypad 1 and End

**KEYPAD\_PERIOD = 99**  
Keypad . and Del

**KEYPAD\_PLUS = 87**  
Keypad +

**KEYPAD\_SEVEN = 95**  
Keypad 7 and Home

**KEYPAD\_SIX = 94**  
Keypad 6 and Right Arrow

**KEYPAD\_THREE = 91**  
Keypad 3 and PgDn

**KEYPAD\_TWO = 90**  
Keypad 2 and Down Arrow

**KEYPAD\_ZERO = 98**  
Keypad 0 and Ins

**L = 15**  
l and L

**LEFT\_ALT = 226**  
Alt modifier left of the spacebar

**LEFT\_ARROW = 80**  
Move the cursor left

**LEFT\_BRACKET = 47**  
[ and {

**LEFT\_CONTROL = 224**  
Control modifier left of the spacebar

**LEFT\_GUI = 227**  
GUI modifier left of the spacebar

**LEFT\_SHIFT = 225**  
Shift modifier left of the spacebar

**M = 16**  
m and M

**MINUS = 45**  
-` and ``\_

**N = 17**  
n and N

**NINE = 38**  
9 and (

**O = 18**  
o and O

**ONE = 30**  
1 and !

**OPTION = 226**  
Labeled as Option on some Mac keyboards

**P = 19**  
p and P

**PAGE\_DOWN = 78**  
Go forward one page

**PAGE\_UP = 75**  
Go back one page

**PAUSE = 72**  
Pause (Break)

**PERIOD = 55**  
. and >

**POUND = 50**  
# and ~ (Non-US keyboard)

**POWER = 102**  
Power (Mac)

**PRINT\_SCREEN = 70**  
Print Screen (SysRq)

**Q = 20**  
q and Q

**QUOTE = 52**  
' and "

**R = 21**  
r and R

**RETURN = 40**  
Alias for ENTER

**RIGHT\_ALT = 230**  
Alt modifier right of the spacebar

**RIGHT\_ARROW = 79**  
Move the cursor right

**RIGHT\_BRACKET = 48**  
] and }

**RIGHT\_CONTROL = 228**  
Control modifier right of the spacebar

**RIGHT\_GUI = 231**  
GUI modifier right of the spacebar

**RIGHT\_SHIFT = 229**  
Shift modifier right of the spacebar

**S = 22**  
s and S

**SCROLL\_LOCK = 71**

Scroll Lock

**SEMICOLON = 51**

; and :

**SEVEN = 36**

7 and &

**SHIFT = 225**

Alias for LEFT\_SHIFT

**SIX = 35**

6 and ^

**SPACE = 44**

Alias for SPACEBAR

**SPACEBAR = 44**

Spacebar

**T = 23**

t and T

**TAB = 43**

Tab and Backtab

**THREE = 32**

3 and #

**TWO = 31**

2 and @

**U = 24**

u and U

**UP\_ARROW = 82**

Move the cursor up

**V = 25**

v and V

**W = 26**

w and W

**WINDOWS = 227**

Labeled with a Windows logo on Windows keyboards

**X = 27**

x and X

**Y = 28**

y and Y

**Z = 29**

z and Z

**ZERO = 39**

0 and )

**classmethod modifier\_bit** (*keycode*)

Return the modifier bit to be set in an HID keycode report if this is a modifier key; otherwise return 0.

## 5.8 adafruit\_hid.keyboard\_layout\_us.KeyboardLayoutUS

- Author(s): Dan Halbert

**class** `adafruit_hid.keyboard_layout_us.KeyboardLayoutUS` (*keyboard*)

Map ASCII characters to appropriate keypresses on a standard US PC keyboard.

Non-ASCII characters and most control characters will raise an exception.

**keycodes** (*char*)

Return a tuple of keycodes needed to type the given character.

**Parameters** **char** (*str of length one.*) – A single ASCII character in a string.

**Returns** tuple of Keycode keycodes.

**Raises** **ValueError** – if **char** is not ASCII or there is no keycode for it.

Examples:

```
# Returns (Keycode.TAB,)
keycodes(' ')
# Returns (Keycode.A,)
keycode('a')
# Returns (Keycode.SHIFT, Keycode.A)
keycode('A')
# Raises ValueError because it's a accented e and is not ASCII
keycode('é')
```

**write** (*string*)

Type the string by pressing and releasing keys on my keyboard.

**Parameters** **string** – A string of ASCII characters.

**Raises** **ValueError** – if any of the characters are not ASCII or have no keycode (such as some control characters).

Example:

```
# Write abc followed by Enter to the keyboard
layout.write('abc\n')
```

## 5.9 adafruit\_hid.mouse.Mouse

- Author(s): Dan Halbert

**class** `adafruit_hid.mouse.Mouse` (*devices*)

Send USB HID mouse reports.

**LEFT\_BUTTON** = 1

Left mouse button.

**MIDDLE\_BUTTON** = 4

Middle mouse button.

**RIGHT\_BUTTON** = 2

Right mouse button.

**click** (*buttons*)

Press and release the given mouse buttons.

**Parameters `buttons`** – a bitwise-or'd combination of `LEFT_BUTTON`, `MIDDLE_BUTTON`, and `RIGHT_BUTTON`.

Examples:

```
# Click the left button.
m.click(Mouse.LEFT_BUTTON)

# Double-click the left button.
m.click(Mouse.LEFT_BUTTON)
m.click(Mouse.LEFT_BUTTON)
```

**`move(x=0, y=0, wheel=0)`**

Move the mouse and turn the wheel as directed.

**Parameters**

- **`x`** – Move the mouse along the x axis. Negative is to the left, positive is to the right.
- **`y`** – Move the mouse along the y axis. Negative is upwards on the display, positive is downwards.
- **`wheel`** – Rotate the wheel this amount. Negative is toward the user, positive is away from the user. The scrolling effect depends on the host.

Examples:

```
# Move 100 to the left. Do not move up and down. Do not roll the scroll wheel.
m.move(-100, 0, 0)
# Same, with keyword arguments.
m.move(x=-100)

# Move diagonally to the upper right.
m.move(50, 20)
# Same.
m.move(x=50, y=-20)

# Roll the mouse wheel away from the user.
m.move(wheel=1)
```

**`press(buttons)`**

Press the given mouse buttons.

**Parameters `buttons`** – a bitwise-or'd combination of `LEFT_BUTTON`, `MIDDLE_BUTTON`, and `RIGHT_BUTTON`.

Examples:

```
# Press the left button.
m.press(Mouse.LEFT_BUTTON)

# Press the left and right buttons simultaneously.
m.press(Mouse.LEFT_BUTTON | Mouse.RIGHT_BUTTON)
```

**`release(buttons)`**

Release the given mouse buttons.

**Parameters `buttons`** – a bitwise-or'd combination of `LEFT_BUTTON`, `MIDDLE_BUTTON`, and `RIGHT_BUTTON`.

**`release_all()`**

Release all the mouse buttons.

## 5.10 adafruit\_hid.consumer\_control.ConsumerControl

- Author(s): Dan Halbert

**class** adafruit\_hid.consumer\_control.ConsumerControl (*devices*)

Send ConsumerControl code reports, used by multimedia keyboards, remote controls, etc.

**press** (*consumer\_code*)

Send a report to indicate that the given key has been pressed. Only one consumer control action can be pressed at a time, so any one that was previously pressed will be released.

**Parameters** **consumer\_code** – a 16-bit consumer control code.

Examples:

```
from adafruit_hid.consumer_control_code import ConsumerControlCode

# Raise volume for 0.5 seconds
consumer_control.press(ConsumerControlCode.VOLUME_INCREMENT)
time.sleep(0.5)
consumer_control.release()
```

**release** ()

Send a report indicating that the consumer control key has been released. Only one consumer control key can be pressed at a time.

Examples:

```
from adafruit_hid.consumer_control_code import ConsumerControlCode

# Raise volume for 0.5 seconds
consumer_control.press(ConsumerControlCode.VOLUME_INCREMENT)
time.sleep(0.5)
consumer_control.release()
```

**send** (*consumer\_code*)

Send a report to do the specified consumer control action, and then stop the action (so it will not repeat).

**Parameters** **consumer\_code** – a 16-bit consumer control code.

Examples:

```
from adafruit_hid.consumer_control_code import ConsumerControlCode

# Raise volume.
consumer_control.send(ConsumerControlCode.VOLUME_INCREMENT)

# Advance to next track (song).
consumer_control.send(ConsumerControlCode.SCAN_NEXT_TRACK)
```

## 5.11 adafruit\_hid.consumer\_control\_code.ConsumerControlCode

- Author(s): Dan Halbert

**class** adafruit\_hid.consumer\_control\_code.ConsumerControlCode

USB HID Consumer Control Device constants.



This list includes a few common consumer control codes from [https://www.usb.org/sites/default/files/hut1\\_21\\_0.pdf#page=118](https://www.usb.org/sites/default/files/hut1_21_0.pdf#page=118).

**BRIGHTNESS\_DECREMENT = 112**

Decrease Brightness

**BRIGHTNESS\_INCREMENT = 111**

Increase Brightness

**EJECT = 184**

Eject

**FAST\_FORWARD = 179**

Fast Forward

**MUTE = 226**

Mute

**PLAY\_PAUSE = 205**

Play/Pause toggle

**RECORD = 178**

Record

**REWIND = 180**

Rewind

**SCAN\_NEXT\_TRACK = 181**

Skip to next track

**SCAN\_PREVIOUS\_TRACK = 182**

Go back to previous track

**STOP = 183**

Stop

**VOLUME\_DECREMENT = 234**

Decrease volume

**VOLUME\_INCREMENT = 233**

Increase volume



## CHAPTER 6

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### a

`adafruit_hid.consumer_control`, [27](#)  
`adafruit_hid.consumer_control_code`, [28](#)  
`adafruit_hid.keyboard`, [18](#)  
`adafruit_hid.keyboard_layout_us`, [25](#)  
`adafruit_hid.keycode`, [19](#)  
`adafruit_hid.mouse`, [26](#)



## A

A (*adafruit\_hid.keycode.Keycode* attribute), 19  
 adafruit\_hid.consumer\_control (*module*), 27  
 adafruit\_hid.consumer\_control\_code (*module*), 28  
 adafruit\_hid.keyboard (*module*), 18  
 adafruit\_hid.keyboard\_layout\_us (*module*), 25  
 adafruit\_hid.keycode (*module*), 19  
 adafruit\_hid.mouse (*module*), 26  
 ALT (*adafruit\_hid.keycode.Keycode* attribute), 19  
 APPLICATION (*adafruit\_hid.keycode.Keycode* attribute), 19

## B

B (*adafruit\_hid.keycode.Keycode* attribute), 19  
 BACKSLASH (*adafruit\_hid.keycode.Keycode* attribute), 19  
 BACKSPACE (*adafruit\_hid.keycode.Keycode* attribute), 20  
 BRIGHTNESS\_DECREMENT (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29  
 BRIGHTNESS\_INCREMENT (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29

## C

C (*adafruit\_hid.keycode.Keycode* attribute), 20  
 CAPS\_LOCK (*adafruit\_hid.keycode.Keycode* attribute), 20  
 click() (*adafruit\_hid.mouse.Mouse* method), 26  
 COMMA (*adafruit\_hid.keycode.Keycode* attribute), 20  
 COMMAND (*adafruit\_hid.keycode.Keycode* attribute), 20  
 ConsumerControl (*class* in *adafruit\_hid.consumer\_control*), 28  
 ConsumerControlCode (*class* in *adafruit\_hid.consumer\_control\_code*), 28  
 CONTROL (*adafruit\_hid.keycode.Keycode* attribute), 20

## D

D (*adafruit\_hid.keycode.Keycode* attribute), 20  
 DELETE (*adafruit\_hid.keycode.Keycode* attribute), 20  
 DOWN\_ARROW (*adafruit\_hid.keycode.Keycode* attribute), 20

## E

E (*adafruit\_hid.keycode.Keycode* attribute), 20  
 EIGHT (*adafruit\_hid.keycode.Keycode* attribute), 20  
 EJECT (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29  
 END (*adafruit\_hid.keycode.Keycode* attribute), 20  
 ENTER (*adafruit\_hid.keycode.Keycode* attribute), 20  
 EQUALS (*adafruit\_hid.keycode.Keycode* attribute), 20  
 ESCAPE (*adafruit\_hid.keycode.Keycode* attribute), 20

## F

F (*adafruit\_hid.keycode.Keycode* attribute), 20  
 F1 (*adafruit\_hid.keycode.Keycode* attribute), 20  
 F10 (*adafruit\_hid.keycode.Keycode* attribute), 20  
 F11 (*adafruit\_hid.keycode.Keycode* attribute), 20  
 F12 (*adafruit\_hid.keycode.Keycode* attribute), 20  
 F13 (*adafruit\_hid.keycode.Keycode* attribute), 20  
 F14 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F15 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F16 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F17 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F18 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F19 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F2 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F20 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F21 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F22 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F23 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F24 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F3 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F4 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F5 (*adafruit\_hid.keycode.Keycode* attribute), 21  
 F6 (*adafruit\_hid.keycode.Keycode* attribute), 21

F7 (*adafruit\_hid.keycode.Keycode* attribute), 21

F8 (*adafruit\_hid.keycode.Keycode* attribute), 21

F9 (*adafruit\_hid.keycode.Keycode* attribute), 21

FAST\_FORWARD (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29

FIVE (*adafruit\_hid.keycode.Keycode* attribute), 21

FORWARD\_SLASH (*adafruit\_hid.keycode.Keycode* attribute), 21

FOUR (*adafruit\_hid.keycode.Keycode* attribute), 22

## G

G (*adafruit\_hid.keycode.Keycode* attribute), 22

GRAVE\_ACCENT (*adafruit\_hid.keycode.Keycode* attribute), 22

GUI (*adafruit\_hid.keycode.Keycode* attribute), 22

## H

H (*adafruit\_hid.keycode.Keycode* attribute), 22

HOME (*adafruit\_hid.keycode.Keycode* attribute), 22

## I

I (*adafruit\_hid.keycode.Keycode* attribute), 22

INSERT (*adafruit\_hid.keycode.Keycode* attribute), 22

## J

J (*adafruit\_hid.keycode.Keycode* attribute), 22

## K

K (*adafruit\_hid.keycode.Keycode* attribute), 22

Keyboard (class in *adafruit\_hid.keyboard*), 18

KeyboardLayoutUS (class in *adafruit\_hid.keyboard\_layout\_us*), 26

Keycode (class in *adafruit\_hid.keycode*), 19

keycodes () (*adafruit\_hid.keyboard\_layout\_us.KeyboardLayoutUS* method), 26

KEYPAD\_ASTERISK (*adafruit\_hid.keycode.Keycode* attribute), 22

KEYPAD\_BACKSLASH (*adafruit\_hid.keycode.Keycode* attribute), 22

KEYPAD\_EIGHT (*adafruit\_hid.keycode.Keycode* attribute), 22

KEYPAD\_ENTER (*adafruit\_hid.keycode.Keycode* attribute), 22

KEYPAD\_EQUALS (*adafruit\_hid.keycode.Keycode* attribute), 22

KEYPAD\_FIVE (*adafruit\_hid.keycode.Keycode* attribute), 22

KEYPAD\_FORWARD\_SLASH (*adafruit\_hid.keycode.Keycode* attribute), 22

KEYPAD\_FOUR (*adafruit\_hid.keycode.Keycode* attribute), 22

KEYPAD\_MINUS (*adafruit\_hid.keycode.Keycode* attribute), 22

KEYPAD\_NINE (*adafruit\_hid.keycode.Keycode* attribute), 22

KEYPAD\_NUMLOCK (*adafruit\_hid.keycode.Keycode* attribute), 22

KEYPAD\_ONE (*adafruit\_hid.keycode.Keycode* attribute), 23

KEYPAD\_PERIOD (*adafruit\_hid.keycode.Keycode* attribute), 23

KEYPAD\_PLUS (*adafruit\_hid.keycode.Keycode* attribute), 23

KEYPAD\_SEVEN (*adafruit\_hid.keycode.Keycode* attribute), 23

KEYPAD\_SIX (*adafruit\_hid.keycode.Keycode* attribute), 23

KEYPAD\_THREE (*adafruit\_hid.keycode.Keycode* attribute), 23

KEYPAD\_TWO (*adafruit\_hid.keycode.Keycode* attribute), 23

KEYPAD\_ZERO (*adafruit\_hid.keycode.Keycode* attribute), 23

## L

L (*adafruit\_hid.keycode.Keycode* attribute), 23

LED\_CAPS\_LOCK (*adafruit\_hid.keyboard.Keyboard* attribute), 18

LED\_COMPOSE (*adafruit\_hid.keyboard.Keyboard* attribute), 18

LED\_NUM\_LOCK (*adafruit\_hid.keyboard.Keyboard* attribute), 18

led\_on () (*adafruit\_hid.keyboard.Keyboard* method), 18

LED\_SCROLL\_LOCK (*adafruit\_hid.keyboard.Keyboard* attribute), 18

led\_status (*adafruit\_hid.keyboard.Keyboard* attribute), 18

LEFT\_ALT (*adafruit\_hid.keycode.Keycode* attribute), 23

LEFT\_ARROW (*adafruit\_hid.keycode.Keycode* attribute), 23

LEFT\_BRACKET (*adafruit\_hid.keycode.Keycode* attribute), 23

LEFT\_BUTTON (*adafruit\_hid.mouse.Mouse* attribute), 26

LEFT\_CONTROL (*adafruit\_hid.keycode.Keycode* attribute), 23

LEFT\_GUI (*adafruit\_hid.keycode.Keycode* attribute), 23

LEFT\_SHIFT (*adafruit\_hid.keycode.Keycode* attribute), 23

## M

M (*adafruit\_hid.keycode.Keycode* attribute), 23

MIDDLE\_BUTTON (*adafruit\_hid.mouse.Mouse* attribute), 26

MINUS (*adafruit\_hid.keycode.Keycode* attribute), 23



`modifier_bit()` (*adafruit\_hid.keycode.Keycode* class method), 25

`Mouse` (class in *adafruit\_hid.mouse*), 26

`move()` (*adafruit\_hid.mouse.Mouse* method), 27

`MUTE` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29

## N

`N` (*adafruit\_hid.keycode.Keycode* attribute), 23

`NINE` (*adafruit\_hid.keycode.Keycode* attribute), 23

## O

`O` (*adafruit\_hid.keycode.Keycode* attribute), 23

`ONE` (*adafruit\_hid.keycode.Keycode* attribute), 23

`OPTION` (*adafruit\_hid.keycode.Keycode* attribute), 24

## P

`P` (*adafruit\_hid.keycode.Keycode* attribute), 24

`PAGE_DOWN` (*adafruit\_hid.keycode.Keycode* attribute), 24

`PAGE_UP` (*adafruit\_hid.keycode.Keycode* attribute), 24

`PAUSE` (*adafruit\_hid.keycode.Keycode* attribute), 24

`PERIOD` (*adafruit\_hid.keycode.Keycode* attribute), 24

`PLAY_PAUSE` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29

`POUND` (*adafruit\_hid.keycode.Keycode* attribute), 24

`POWER` (*adafruit\_hid.keycode.Keycode* attribute), 24

`press()` (*adafruit\_hid.consumer\_control.ConsumerControl* method), 28

`press()` (*adafruit\_hid.keyboard.Keyboard* method), 18

`press()` (*adafruit\_hid.mouse.Mouse* method), 27

`PRINT_SCREEN` (*adafruit\_hid.keycode.Keycode* attribute), 24

## Q

`Q` (*adafruit\_hid.keycode.Keycode* attribute), 24

`QUOTE` (*adafruit\_hid.keycode.Keycode* attribute), 24

## R

`R` (*adafruit\_hid.keycode.Keycode* attribute), 24

`RECORD` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29

`release()` (*adafruit\_hid.consumer\_control.ConsumerControl* method), 28

`release()` (*adafruit\_hid.keyboard.Keyboard* method), 19

`release()` (*adafruit\_hid.mouse.Mouse* method), 27

`release_all()` (*adafruit\_hid.keyboard.Keyboard* method), 19

`release_all()` (*adafruit\_hid.mouse.Mouse* method), 27

`RETURN` (*adafruit\_hid.keycode.Keycode* attribute), 24

`REWIND` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29

`RIGHT_ALT` (*adafruit\_hid.keycode.Keycode* attribute), 24

`RIGHT_ARROW` (*adafruit\_hid.keycode.Keycode* attribute), 24

`RIGHT_BRACKET` (*adafruit\_hid.keycode.Keycode* attribute), 24

`RIGHT_BUTTON` (*adafruit\_hid.mouse.Mouse* attribute), 26

`RIGHT_CONTROL` (*adafruit\_hid.keycode.Keycode* attribute), 24

`RIGHT_GUI` (*adafruit\_hid.keycode.Keycode* attribute), 24

`RIGHT_SHIFT` (*adafruit\_hid.keycode.Keycode* attribute), 24

## S

`S` (*adafruit\_hid.keycode.Keycode* attribute), 24

`SCAN_NEXT_TRACK` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29

`SCAN_PREVIOUS_TRACK` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29

`SCROLL_LOCK` (*adafruit\_hid.keycode.Keycode* attribute), 24

`SEMICOLON` (*adafruit\_hid.keycode.Keycode* attribute), 25

`send()` (*adafruit\_hid.consumer\_control.ConsumerControl* method), 28

`send()` (*adafruit\_hid.keyboard.Keyboard* method), 19

`SEVEN` (*adafruit\_hid.keycode.Keycode* attribute), 25

`SHIFT` (*adafruit\_hid.keycode.Keycode* attribute), 25

`SIX` (*adafruit\_hid.keycode.Keycode* attribute), 25

`SPACE` (*adafruit\_hid.keycode.Keycode* attribute), 25

`SPACEBAR` (*adafruit\_hid.keycode.Keycode* attribute), 25

`STOP` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29

## T

`T` (*adafruit\_hid.keycode.Keycode* attribute), 25

`TAB` (*adafruit\_hid.keycode.Keycode* attribute), 25

`THREE` (*adafruit\_hid.keycode.Keycode* attribute), 25

`TWO` (*adafruit\_hid.keycode.Keycode* attribute), 25

## U

`U` (*adafruit\_hid.keycode.Keycode* attribute), 25

`UP_ARROW` (*adafruit\_hid.keycode.Keycode* attribute), 25

## V

`V` (*adafruit\_hid.keycode.Keycode* attribute), 25

`VOLUME_DECREMENT` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29

`VOLUME_INCREMENT` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 29

## W

`W` (*adafruit\_hid.keycode.Keycode attribute*), [25](#)

`WINDOWS` (*adafruit\_hid.keycode.Keycode attribute*), [25](#)

`write()` (*adafruit\_hid.keyboard\_layout\_us.KeyboardLayoutUS method*), [26](#)

## X

`X` (*adafruit\_hid.keycode.Keycode attribute*), [25](#)

## Y

`Y` (*adafruit\_hid.keycode.Keycode attribute*), [25](#)

## Z

`Z` (*adafruit\_hid.keycode.Keycode attribute*), [25](#)

`ZERO` (*adafruit\_hid.keycode.Keycode attribute*), [25](#)