

---

# **Adafruit HID Library Documentation**

***Release 1.0***

**Scott Shawcroft**

**Mar 03, 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	adafruit_hid.keyboard.Keyboard . . . . .	17
5.3	adafruit_hid.keycode.Keycode . . . . .	17
5.4	adafruit_hid.keyboard_layout_us.KeyboardLayoutUS . . . . .	24
5.5	adafruit_hid.mouse.Mouse . . . . .	24
5.6	adafruit_hid.consumer_control.ConsumerControl . . . . .	26
5.7	adafruit_hid.consumer_control_code.ConsumerControlCode . . . . .	26
5.8	adafruit_hid.gamepad.Gamepad . . . . .	27
<b>6</b>	<b>Indices and tables</b>	<b>29</b>
	<b>Python Module Index</b>	<b>31</b>
	<b>Index</b>	<b>33</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.

*New in CircuitPython 3.0.*

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

cc = ConsumerControl(usb_hid.devices)
```

(continues on next page)

(continued from previous page)

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

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

The Gamepad class emulates a two-joystick gamepad with 16 buttons.

*New in CircuitPython 3.0.*

```
import usb_hid
from adafruit_hid.gamepad import Gamepad

gp = Gamepad(usb_hid.devices)

# Click gamepad buttons.
gp.click_buttons(1, 7)

# Move joysticks.
gp.move_joysticks(x=2, y=0, z=-20)
```



## 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\_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
```

(continues on next page)

(continued from previous page)

```

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)

```

Listing 2: 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)
28     elif down.value:
29         mouse.move(wheel=-1)
30
31     time.sleep(0.1)

```

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

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  import board
5  import digitalio
6  import analogio
7  import usb_hid
8
9  from adafruit_hid.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.

```

(continues on next page)

(continued from previous page)

```

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)

```

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
6 import time
7
8 import board
9 import busio
10 from micropython import const
11 import adafruit_seesaw
12 import usb_hid
13 from adafruit_hid.gamepad import Gamepad
14
15

```

(continues on next page)

(continued from previous page)

```

16 def range_map(value, in_min, in_max, out_min, out_max):
17     return (value - in_min) * (out_max - out_min) // (in_max - in_min) + out_min
18
19
20 BUTTON_RIGHT = const(6)
21 BUTTON_DOWN = const(7)
22 BUTTON_LEFT = const(9)
23 BUTTON_UP = const(10)
24 BUTTON_SEL = const(14)
25 button_mask = const(
26     (1 << BUTTON_RIGHT)
27     | (1 << BUTTON_DOWN)
28     | (1 << BUTTON_LEFT)
29     | (1 << BUTTON_UP)
30     | (1 << BUTTON_SEL)
31 )
32
33 i2c = busio.I2C(board.SCL, board.SDA)
34
35 ss = adafruit_seesaw.Seesaw(i2c)
36
37 ss.pin_mode_bulk(button_mask, ss.INPUT_PULLUP)
38
39 last_game_x = 0
40 last_game_y = 0
41
42 g = Gamepad(usb_hid.devices)
43
44 while True:
45     x = ss.analog_read(2)
46     y = ss.analog_read(3)
47
48     game_x = range_map(x, 0, 1023, -127, 127)
49     game_y = range_map(y, 0, 1023, -127, 127)
50     if last_game_x != game_x or last_game_y != game_y:
51         last_game_x = game_x
52         last_game_y = game_y
53         print(game_x, game_y)
54         g.move_joysticks(x=game_x, y=game_y)
55
56     buttons = (BUTTON_RIGHT, BUTTON_DOWN, BUTTON_LEFT, BUTTON_UP, BUTTON_SEL)
57     button_state = [False] * len(buttons)
58     for i, button in enumerate(buttons):
59         buttons = ss.digital_read_bulk(button_mask)
60         if not (buttons & (1 << button) and not button_state[i]):
61             g.press_buttons(i + 1)
62             print("Press", i + 1)
63             button_state[i] = True
64         elif button_state[i]:
65             g.release_buttons(i + 1)
66             print("Release", i + 1)
67             button_state[i] = False
68
69     time.sleep(0.01)

```

## 5.2 adafruit\_hid.keyboard.Keyboard

- Author(s): Scott Shawcroft, Dan Halbert

**class** `adafruit_hid.keyboard.Keyboard` (*devices*)

Send HID keyboard reports.

**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

## 5.3 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

**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.

## 5.4 `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')
```

## 5.5 `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.

## 5.6 `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.

**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)
```

## 5.7 `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).

**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

## 5.8 adafruit\_hid.gamepad.Gamepad

- Author(s): Dan Halbert

**class** adafruit\_hid.gamepad.**Gamepad** (*devices*)

Emulate a generic gamepad controller with 16 buttons, numbered 1-16, and two joysticks, one controlling `x`` and ``y`` values, and the other controlling `z` and `r_z` (z rotation or Rz) values.

The joystick values could be interpreted differently by the receiving program: those are just the names used here. The joystick values are in the range -127 to 127.

**click\_buttons** (*\*buttons*)

Press and release the given buttons.

**move\_joysticks** (*x=None, y=None, z=None, r\_z=None*)

Set and send the given joystick values. The joysticks will remain set with the given values until changed

One joystick provides `x` and `y` values, and the other provides `z` and `r_z` (z rotation). Any values left as `None` will not be changed.

All values must be in the range -127 to 127 inclusive.

Examples:

```
# Change x and y values only.
gp.move_joysticks(x=100, y=-50)

# Reset all joystick values to center position.
gp.move_joysticks(0, 0, 0, 0)
```

**press\_buttons** (*\*buttons*)

Press and hold the given buttons.

**release\_all\_buttons** ()

Release all the buttons.

**release\_buttons** (*\*buttons*)

Release the given buttons.

**reset\_all** ()

Release all buttons and set joysticks to zero.





## CHAPTER 6

---

### Indices and tables

---

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



### a

`adafruit_hid.consumer_control`, [26](#)  
`adafruit_hid.consumer_control_code`, [26](#)  
`adafruit_hid.gamepad`, [27](#)  
`adafruit_hid.keyboard`, [16](#)  
`adafruit_hid.keyboard_layout_us`, [24](#)  
`adafruit_hid.keycode`, [17](#)  
`adafruit_hid.mouse`, [24](#)



## A

A (*adafruit\_hid.keycode.Keycode attribute*), 18  
adafruit\_hid.consumer\_control (*module*), 26  
adafruit\_hid.consumer\_control\_code (*module*), 26  
adafruit\_hid.gamepad (*module*), 27  
adafruit\_hid.keyboard (*module*), 16  
adafruit\_hid.keyboard\_layout\_us (*module*), 24  
adafruit\_hid.keycode (*module*), 17  
adafruit\_hid.mouse (*module*), 24  
ALT (*adafruit\_hid.keycode.Keycode attribute*), 18  
APPLICATION (*adafruit\_hid.keycode.Keycode attribute*), 18

## B

B (*adafruit\_hid.keycode.Keycode attribute*), 18  
BACKSLASH (*adafruit\_hid.keycode.Keycode attribute*), 18  
BACKSPACE (*adafruit\_hid.keycode.Keycode attribute*), 18

## C

C (*adafruit\_hid.keycode.Keycode attribute*), 18  
CAPS\_LOCK (*adafruit\_hid.keycode.Keycode attribute*), 18  
click() (*adafruit\_hid.mouse.Mouse method*), 25  
click\_buttons() (*adafruit\_hid.gamepad.Gamepad method*), 27  
COMMA (*adafruit\_hid.keycode.Keycode attribute*), 18  
COMMAND (*adafruit\_hid.keycode.Keycode attribute*), 18  
ConsumerControl (*class in adafruit\_hid.consumer\_control*), 26  
ConsumerControlCode (*class in adafruit\_hid.consumer\_control\_code*), 26  
CONTROL (*adafruit\_hid.keycode.Keycode attribute*), 18

## D

D (*adafruit\_hid.keycode.Keycode attribute*), 18

DELETE (*adafruit\_hid.keycode.Keycode attribute*), 18  
DOWN\_ARROW (*adafruit\_hid.keycode.Keycode attribute*), 18

## E

E (*adafruit\_hid.keycode.Keycode attribute*), 18  
EIGHT (*adafruit\_hid.keycode.Keycode attribute*), 18  
EJECT (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode attribute*), 26  
END (*adafruit\_hid.keycode.Keycode attribute*), 18  
ENTER (*adafruit\_hid.keycode.Keycode attribute*), 18  
EQUALS (*adafruit\_hid.keycode.Keycode attribute*), 18  
ESCAPE (*adafruit\_hid.keycode.Keycode attribute*), 18

## F

F (*adafruit\_hid.keycode.Keycode attribute*), 19  
F1 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F10 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F11 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F12 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F13 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F14 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F15 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F16 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F17 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F18 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F19 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F2 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F3 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F4 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F5 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F6 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F7 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F8 (*adafruit\_hid.keycode.Keycode attribute*), 19  
F9 (*adafruit\_hid.keycode.Keycode attribute*), 19  
FAST\_FORWARD (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode attribute*), 26  
FIVE (*adafruit\_hid.keycode.Keycode attribute*), 19  
FORWARD\_SLASH (*adafruit\_hid.keycode.Keycode attribute*), 20

FOUR (*adafruit\_hid.keycode.Keycode* attribute), 20

## G

G (*adafruit\_hid.keycode.Keycode* attribute), 20

Gamepad (class in *adafruit\_hid.gamepad*), 27

GRAVE\_ACCENT (*adafruit\_hid.keycode.Keycode* attribute), 20

GUI (*adafruit\_hid.keycode.Keycode* attribute), 20

## H

H (*adafruit\_hid.keycode.Keycode* attribute), 20

HOME (*adafruit\_hid.keycode.Keycode* attribute), 20

## I

I (*adafruit\_hid.keycode.Keycode* attribute), 20

INSERT (*adafruit\_hid.keycode.Keycode* attribute), 20

## J

J (*adafruit\_hid.keycode.Keycode* attribute), 20

## K

K (*adafruit\_hid.keycode.Keycode* attribute), 20

Keyboard (class in *adafruit\_hid.keyboard*), 17

KeyboardLayoutUS (class *adafruit\_hid.keyboard\_layout\_us*), 24

Keycode (class in *adafruit\_hid.keycode*), 17

keycodes () (*adafruit\_hid.keyboard\_layout\_us.KeyboardLayoutUS* method), 24

KEYPAD\_ASTERISK (*adafruit\_hid.keycode.Keycode* attribute), 20

KEYPAD\_BACKSLASH (*adafruit\_hid.keycode.Keycode* attribute), 20

KEYPAD\_EIGHT (*adafruit\_hid.keycode.Keycode* attribute), 20

KEYPAD\_ENTER (*adafruit\_hid.keycode.Keycode* attribute), 20

KEYPAD\_EQUALS (*adafruit\_hid.keycode.Keycode* attribute), 20

KEYPAD\_FIVE (*adafruit\_hid.keycode.Keycode* attribute), 20

KEYPAD\_FORWARD\_SLASH (*adafruit\_hid.keycode.Keycode* attribute), 20

KEYPAD\_FOUR (*adafruit\_hid.keycode.Keycode* attribute), 20

KEYPAD\_MINUS (*adafruit\_hid.keycode.Keycode* attribute), 20

KEYPAD\_NINE (*adafruit\_hid.keycode.Keycode* attribute), 20

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

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

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

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

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

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

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

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

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

## L

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

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

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

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

LEFT\_BUTTON (*adafruit\_hid.mouse.Mouse* attribute), 24

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

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

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

## M

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

MIDDLE\_BUTTON (*adafruit\_hid.mouse.Mouse* attribute), 24

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

modifier\_bit () (*adafruit\_hid.keycode.Keycode* class method), 23

Mouse (class in *adafruit\_hid.mouse*), 24

move () (*adafruit\_hid.mouse.Mouse* method), 25

move\_joysticks () (*adafruit\_hid.gamepad.Gamepad* method), 27

MUTE (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 26

## N

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

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

## O

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

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

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

## P

`P` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`PAGE_DOWN` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`PAGE_UP` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`PAUSE` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`PERIOD` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`PLAY_PAUSE` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 26  
`POUND` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`POWER` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`press()` (*adafruit\_hid.keyboard.Keyboard* method), 17  
`press()` (*adafruit\_hid.mouse.Mouse* method), 25  
`press_buttons()` (*adafruit\_hid.gamepad.Gamepad* method), 27  
`PRINT_SCREEN` (*adafruit\_hid.keycode.Keycode* attribute), 22

## Q

`Q` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`QUOTE` (*adafruit\_hid.keycode.Keycode* attribute), 22

## R

`R` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`RECORD` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 26  
`release()` (*adafruit\_hid.keyboard.Keyboard* method), 17  
`release()` (*adafruit\_hid.mouse.Mouse* method), 25  
`release_all()` (*adafruit\_hid.keyboard.Keyboard* method), 17  
`release_all()` (*adafruit\_hid.mouse.Mouse* method), 26  
`release_all_buttons()` (*adafruit\_hid.gamepad.Gamepad* method), 27  
`release_buttons()` (*adafruit\_hid.gamepad.Gamepad* method), 27  
`reset_all()` (*adafruit\_hid.gamepad.Gamepad* method), 27  
`RETURN` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`REWIND` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 26  
`RIGHT_ALT` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`RIGHT_ARROW` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`RIGHT_BRACKET` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`RIGHT_BUTTON` (*adafruit\_hid.mouse.Mouse* attribute), 24  
`RIGHT_CONTROL` (*adafruit\_hid.keycode.Keycode* attribute), 22

`RIGHT_GUI` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`RIGHT_SHIFT` (*adafruit\_hid.keycode.Keycode* attribute), 22

## S

`S` (*adafruit\_hid.keycode.Keycode* attribute), 22  
`SCAN_NEXT_TRACK` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 26  
`SCAN_PREVIOUS_TRACK` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 26  
`SCROLL_LOCK` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`SEMICOLON` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`send()` (*adafruit\_hid.consumer\_control.ConsumerControl* method), 26  
`send()` (*adafruit\_hid.keyboard.Keyboard* method), 17  
`SEVEN` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`SHIFT` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`SIX` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`SPACE` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`SPACEBAR` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`STOP` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 27

## T

`T` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`TAB` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`THREE` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`TWO` (*adafruit\_hid.keycode.Keycode* attribute), 23

## U

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

## V

`V` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`VOLUME_DECREMENT` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 27  
`VOLUME_INCREMENT` (*adafruit\_hid.consumer\_control\_code.ConsumerControlCode* attribute), 27

## W

`W` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`WINDOWS` (*adafruit\_hid.keycode.Keycode* attribute), 23  
`write()` (*adafruit\_hid.keyboard\_layout\_us.KeyboardLayoutUS* method), 24

## X

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

## Y

`Y` (*adafruit\_hid.keycode.Keycode* attribute), [23](#)

## Z

`Z` (*adafruit\_hid.keycode.Keycode* attribute), [23](#)

`ZERO` (*adafruit\_hid.keycode.Keycode* attribute), [23](#)