
Adafruit HID Library Documentation

Release 1.0

Scott Shawcroft

May 04, 2021

Contents

1	Dependencies	3
2	Usage Example	5
3	Contributing	9
4	Documentation	11
5	Table of Contents	13
5.1	Simple test	13
5.2	adafruit_hid.keyboard.Keyboard	17
5.3	adafruit_hid.keycode.Keycode	18
5.4	adafruit_hid.keyboard_layout_us.KeyboardLayoutUS	24
5.5	adafruit_hid.mouse.Mouse	25
5.6	adafruit_hid.consumer_control.ConsumerControl	26
5.7	adafruit_hid.consumer_control_code.ConsumerControlCode	27
6	Indices and tables	29
	Python Module Index	31
	Index	33

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_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 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 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.

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

# 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

5.3 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. 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)
```

(continues on next page)

(continued from previous page)

```

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.

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
```

(continues on next page)

(continued from previous page)

```
# 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)
```

5.7 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.

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`, [26](#)
`adafruit_hid.consumer_control_code`, [27](#)
`adafruit_hid.keyboard`, [16](#)
`adafruit_hid.keyboard_layout_us`, [24](#)
`adafruit_hid.keycode`, [18](#)
`adafruit_hid.mouse`, [25](#)

A

A (*adafruit_hid.keycode.Keycode attribute*), 18
adafruit_hid.consumer_control (*module*), 26
adafruit_hid.consumer_control_code (*module*), 27
adafruit_hid.keyboard (*module*), 16
adafruit_hid.keyboard_layout_us (*module*), 24
adafruit_hid.keycode (*module*), 18
adafruit_hid.mouse (*module*), 25
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
COMMA (*adafruit_hid.keycode.Keycode attribute*), 18
COMMAND (*adafruit_hid.keycode.Keycode attribute*), 19
ConsumerControl (*class in adafruit_hid.consumer_control*), 26
ConsumerControlCode (*class in adafruit_hid.consumer_control_code*), 27
CONTROL (*adafruit_hid.keycode.Keycode attribute*), 19

D

D (*adafruit_hid.keycode.Keycode attribute*), 19
DELETE (*adafruit_hid.keycode.Keycode attribute*), 19
DOWN_ARROW (*adafruit_hid.keycode.Keycode attribute*), 19

E

E (*adafruit_hid.keycode.Keycode attribute*), 19
EIGHT (*adafruit_hid.keycode.Keycode attribute*), 19
EJECT (*adafruit_hid.consumer_control_code.ConsumerControlCode attribute*), 27
END (*adafruit_hid.keycode.Keycode attribute*), 19
ENTER (*adafruit_hid.keycode.Keycode attribute*), 19
EQUALS (*adafruit_hid.keycode.Keycode attribute*), 19
ESCAPE (*adafruit_hid.keycode.Keycode attribute*), 19

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*), 20
F19 (*adafruit_hid.keycode.Keycode attribute*), 20
F2 (*adafruit_hid.keycode.Keycode attribute*), 20
F3 (*adafruit_hid.keycode.Keycode attribute*), 20
F4 (*adafruit_hid.keycode.Keycode attribute*), 20
F5 (*adafruit_hid.keycode.Keycode attribute*), 20
F6 (*adafruit_hid.keycode.Keycode attribute*), 20
F7 (*adafruit_hid.keycode.Keycode attribute*), 20
F8 (*adafruit_hid.keycode.Keycode attribute*), 20
F9 (*adafruit_hid.keycode.Keycode attribute*), 20
FAST_FORWARD (*adafruit_hid.consumer_control_code.ConsumerControlCode attribute*), 27
FIVE (*adafruit_hid.keycode.Keycode attribute*), 20
FORWARD_SLASH (*adafruit_hid.keycode.Keycode attribute*), 20
FOUR (*adafruit_hid.keycode.Keycode attribute*), 20

G

G (*adafruit_hid.keycode.Keycode* attribute), 20
 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), 21
 Keyboard (class in *adafruit_hid.keyboard*), 17
 KeyboardLayoutUS (class *adafruit_hid.keyboard_layout_us*), 24
 Keycode (class in *adafruit_hid.keycode*), 18
 keycodes () (*adafruit_hid.keyboard_layout_us.KeyboardLayoutUS* attribute), 17
 method), 24
 KEYPAD_ASTERISK (*adafruit_hid.keycode.Keycode* attribute), 21
 KEYPAD_BACKSLASH (*adafruit_hid.keycode.Keycode* attribute), 21
 KEYPAD_EIGHT (*adafruit_hid.keycode.Keycode* attribute), 21
 KEYPAD_ENTER (*adafruit_hid.keycode.Keycode* attribute), 21
 KEYPAD_EQUALS (*adafruit_hid.keycode.Keycode* attribute), 21
 KEYPAD_FIVE (*adafruit_hid.keycode.Keycode* attribute), 21
 KEYPAD_FORWARD_SLASH (*adafruit_hid.keycode.Keycode* attribute), 21
 KEYPAD_FOUR (*adafruit_hid.keycode.Keycode* attribute), 21
 KEYPAD_MINUS (*adafruit_hid.keycode.Keycode* attribute), 21
 KEYPAD_NINE (*adafruit_hid.keycode.Keycode* attribute), 21
 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
 LED_CAPS_LOCK (*adafruit_hid.keyboard.Keyboard* attribute), 17
 LED_COMPOSE (*adafruit_hid.keyboard.Keyboard* attribute), 17
 LED_NUM_LOCK (*adafruit_hid.keyboard.Keyboard* attribute), 17
 in led_on () (*adafruit_hid.keyboard.Keyboard* method), 17
 LED_SCROLL_LOCK (*adafruit_hid.keyboard.Keyboard* attribute), 17
 led_status (*adafruit_hid.keyboard.Keyboard* attribute), 17
 LEFT_ALT (*adafruit_hid.keycode.Keycode* attribute), 22
 LEFT_ARROW (*adafruit_hid.keycode.Keycode* attribute), 22
 LEFT_BRACKET (*adafruit_hid.keycode.Keycode* attribute), 22
 LEFT_BUTTON (*adafruit_hid.mouse.Mouse* attribute), 25
 LEFT_CONTROL (*adafruit_hid.keycode.Keycode* attribute), 22
 LEFT_GUI (*adafruit_hid.keycode.Keycode* attribute), 22
 LEFT_SHIFT (*adafruit_hid.keycode.Keycode* attribute), 22

M

M (*adafruit_hid.keycode.Keycode* attribute), 22
 MIDDLE_BUTTON (*adafruit_hid.mouse.Mouse* attribute), 25
 MINUS (*adafruit_hid.keycode.Keycode* attribute), 22
 modifier_bit () (*adafruit_hid.keycode.Keycode* class method), 24
 Mouse (class in *adafruit_hid.mouse*), 25
 move () (*adafruit_hid.mouse.Mouse* method), 25
 MUTE (*adafruit_hid.consumer_control_code.ConsumerControlCode* attribute), 27

N

N (*adafruit_hid.keycode.Keycode* attribute), 22

NINE (*adafruit_hid.keycode.Keycode* attribute), 22

O

O (*adafruit_hid.keycode.Keycode* attribute), 22

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), 27

POUND (*adafruit_hid.keycode.Keycode* attribute), 22

POWER (*adafruit_hid.keycode.Keycode* attribute), 22

press() (*adafruit_hid.consumer_control.ConsumerControl* method), 26

press() (*adafruit_hid.keyboard.Keyboard* method), 17

press() (*adafruit_hid.mouse.Mouse* method), 26

PRINT_SCREEN (*adafruit_hid.keycode.Keycode* attribute), 22

Q

Q (*adafruit_hid.keycode.Keycode* attribute), 23

QUOTE (*adafruit_hid.keycode.Keycode* attribute), 23

R

R (*adafruit_hid.keycode.Keycode* attribute), 23

RECORD (*adafruit_hid.consumer_control_code.ConsumerControlCode* attribute), 27

release() (*adafruit_hid.consumer_control.ConsumerControl* method), 27

release() (*adafruit_hid.keyboard.Keyboard* method), 17

release() (*adafruit_hid.mouse.Mouse* method), 26

release_all() (*adafruit_hid.keyboard.Keyboard* method), 18

release_all() (*adafruit_hid.mouse.Mouse* method), 26

RETURN (*adafruit_hid.keycode.Keycode* attribute), 23

REWIND (*adafruit_hid.consumer_control_code.ConsumerControlCode* attribute), 27

RIGHT_ALT (*adafruit_hid.keycode.Keycode* attribute), 23

RIGHT_ARROW (*adafruit_hid.keycode.Keycode* attribute), 23

RIGHT_BRACKET (*adafruit_hid.keycode.Keycode* attribute), 23

RIGHT_BUTTON (*adafruit_hid.mouse.Mouse* attribute), 25

RIGHT_CONTROL (*adafruit_hid.keycode.Keycode* attribute), 23

RIGHT_GUI (*adafruit_hid.keycode.Keycode* attribute), 23

RIGHT_SHIFT (*adafruit_hid.keycode.Keycode* attribute), 23

S

S (*adafruit_hid.keycode.Keycode* attribute), 23

SCAN_NEXT_TRACK (*adafruit_hid.consumer_control_code.ConsumerControlCode* attribute), 28

SCAN_PREVIOUS_TRACK (*adafruit_hid.consumer_control_code.ConsumerControlCode* attribute), 28

SCROLL_LOCK (*adafruit_hid.keycode.Keycode* attribute), 23

SEMICOLON (*adafruit_hid.keycode.Keycode* attribute), 23

send() (*adafruit_hid.consumer_control.ConsumerControl* method), 27

send() (*adafruit_hid.keyboard.Keyboard* method), 18

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), 28

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), 24

U

U (*adafruit_hid.keycode.Keycode* attribute), 24

UP_ARROW (*adafruit_hid.keycode.Keycode* attribute), 24

V

V (*adafruit_hid.keycode.Keycode* attribute), 24

VOLUME_DECREMENT (*adafruit_hid.consumer_control_code.ConsumerControlCode* attribute), 28

VOLUME_INCREMENT (*adafruit_hid.consumer_control_code.ConsumerControlCode* attribute), 28

W

W (*adafruit_hid.keycode.Keycode* attribute), 24

WINDOWS (*adafruit_hid.keycode.Keycode* attribute), 24

write() (*adafruit_hid.keyboard_layout_us.KeyboardLayoutUS* method), 25

X

X (*adafruit_hid.keycode.Keycode* attribute), 24

Y

`Y` (*adafruit_hid.keycode.Keycode* attribute), [24](#)

Z

`Z` (*adafruit_hid.keycode.Keycode* attribute), [24](#)

`ZERO` (*adafruit_hid.keycode.Keycode* attribute), [24](#)