adafruit_thermal_printer Library Documentation

Release 1.0

Tony DiCola

Oct 21, 2019

Contents

1 Dependencies **3**

2 Usage Example **5**

3 Contributing **7**
3.1 Sphinx documentation 7

4 Table of Contents **9**
4.1 Simple test 9
4.2 adafruit_thermal_printer.thermal_printer - Thermal Printer Driver 11
4.2.1 Implementation Notes 12
4.3 adafruit_thermal_printer.thermal_printer_264.ThermalPrinter 13
4.4 adafruit_thermal_printer.thermal_printer_legacy.ThermalPrinter 13

5 Indices and tables **15**

Python Module Index **17**

Index **19**

CircuitPython module for control of various small serial thermal printers.

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.

NOTE: This library is not supported for smaller non-Express boards like the Trinket M0, Gemma M0, etc.

CHAPTER 2

Usage Example

See examples/thermal_printer_simpletest.py for a demo of basic printer usage.

CHAPTER 3

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

3.1 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the index.html in your browser to view them. It will also (due to -W) error out on any warning like Travis will. This is a good way to locally verify it will pass.

CHAPTER 4

Table of Contents

4.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/thermal_printer_simpletest.py

```
# Simple demo of printer functionality.
# Author: Tony DiCola
import board
import busio

import adafruit_thermal_printer

# Pick which version thermal printer class to use depending on the version of
# your printer.  Hold the button on the printer as it's powered on and it will
# print a test page that displays the firmware version, like 2.64, 2.68, etc.
# Use this version in the get_printer_class function below.
ThermalPrinter = adafruit_thermal_printer.get_printer_class(2.69)

# Define RX and TX pins for the board's serial port connected to the printer.
# Only the TX pin needs to be configued, and note to take care NOT to connect
# the RX pin if your board doesn't support 5V inputs.  If RX is left unconnected
# the only loss in functionality is checking if the printer has paper--all other
# functions of the printer will work.
RX = board.RX
TX = board.TX

# Create a serial connection for the printer.  You must use the same baud rate
# as your printer is configured (print a test page by holding the button
# during power-up and it will show the baud rate).  Most printers use 19200.
uart = busio.UART(TX, RX, baudrate=19200)

# For a computer, use the pyserial library for uart access.
```

(continues on next page)

(continued from previous page)

```
# import serial
# uart = serial.Serial("/dev/serial0", baudrate=19200, timeout=3000)

# Create the printer instance.
printer = ThermalPrinter(uart, auto_warm_up=False)

# Initialize the printer.  Note this will take a few seconds for the printer
# to warm up and be ready to accept commands (hence calling it explicitly vs.
# automatically in the initializer with the default auto_warm_up=True).
printer.warm_up()

# Check if the printer has paper.  This only works if the RX line is connected
# on your board (but BE CAREFUL as mentioned above this RX line is 5V!)
if printer.has_paper():
    print('Printer has paper!')
else:
    print('Printer might be out of paper, or RX is disconnected!')

# Print a test page:
printer.test_page()

# Move the paper forward two lines:
printer.feed(2)

# Print a line of text:
printer.print('Hello world!')

# Print a bold line of text:
printer.bold = True
printer.print('Bold hello world!')
printer.bold = False

# Print a normal/thin underline line of text:
printer.underline = adafruit_thermal_printer.UNDERLINE_THIN
printer.print('Thin underline!')

# Print a thick underline line of text:
printer.underline = adafruit_thermal_printer.UNDERLINE_THICK
printer.print('Thick underline!')

# Disable underlines.
printer.underline = None

# Print an inverted line.
printer.inverse = True
printer.print('Inverse hello world!')
printer.inverse = False

# Print an upside down line.
printer.upside_down = True
printer.print('Upside down hello!')
printer.upside_down = False

# Print a double height line.
printer.double_height = True
printer.print('Double height!')
printer.double_height = False
```

(continues on next page)

(continued from previous page)

```
85
86  # Print a double width line.
87  printer.double_width = True
88  printer.print('Double width!')
89  printer.double_width = False
90
91  # Print a strike-through line.
92  printer.strike = True
93  printer.print('Strike-through hello!')
94  printer.strike = False
95
96  # Print medium size text.
97  printer.size = adafruit_thermal_printer.SIZE_MEDIUM
98  printer.print('Medium size text!')
99
100 # Print large size text.
101 printer.size = adafruit_thermal_printer.SIZE_LARGE
102 printer.print('Large size text!')
103
104 # Back to normal / small size text.
105 printer.size = adafruit_thermal_printer.SIZE_SMALL
106
107 # Print center justified text.
108 printer.justify = adafruit_thermal_printer.JUSTIFY_CENTER
109 printer.print('Center justified!')
110
111 # Print right justified text.
112 printer.justify = adafruit_thermal_printer.JUSTIFY_RIGHT
113 printer.print('Right justified!')
114
115 # Back to left justified / normal text.
116 printer.justify = adafruit_thermal_printer.JUSTIFY_LEFT
117
118 # Print a UPC barcode.
119 printer.print('UPCA barcode:')
120 printer.print_barcode('123456789012', printer.UPC_A)
121
122 # Feed a few lines to see everything.
123 printer.feed(2)
```

4.2 adafruit_thermal_printer.thermal_printer - Thermal Printer Driver

Thermal printer control module built to work with small serial thermal receipt printers. Note that these printers have many different firmware versions and care must be taken to select the appropriate module inside this package for your firmware printer:

- thermal_printer = The latest printers with firmware version 2.68+
- thermal_printer_264 = Printers with firmware version 2.64 up to 2.68.
- thermal_printer_legacy = Printers with firmware version before 2.64.
- Author(s): Tony DiCola

4.2.1 Implementation Notes

Hardware:

- Mini Thermal Receipt Printer (Product ID: 597)

Software and Dependencies:

- Adafruit CircuitPython firmware for the ESP8622 and M0-based boards: https://github.com/adafruit/circuitpython/releases

class adafruit_thermal_printer.thermal_printer.**ThermalPrinter**(*uart*, *, *byte_delay_s=0.00057346*, *dot_feed_s=0.0021*, *dot_print_s=0.03*, *auto_warm_up=True*)

Thermal printer for printers with firmware version 2.68 or higher.

feed(*lines*)
Advance paper by specified number of blank lines.

feed_rows(*rows*)
Advance paper by specified number of pixel rows.

flush()
Flush data pending in the printer.

has_paper()
Return a boolean indicating if the printer has paper. You MUST have the serial RX line hooked up for this to work. NOTE: be VERY CAREFUL to ensure your board can handle a 5V serial input before hooking up the RX line!

inverse
Set the inverse printing mode boolean to enable or disable inverse printing.

justify
Set the justification of text, must be a value of JUSTIFY_LEFT, JUSTIFY_CENTER, or JUSTIFY_RIGHT.

offline()
Put the printer into an offline state. No other commands can be sent until an online call is made.

online()
Put the printer into an online state after previously put offline.

print(*text*, *end='\n'*)
Print a line of text. Optionally specify the end keyword to override the new line printed after the text (set to None to disable the new line entirely).

print_barcode(*text*, *barcode_type*)
Print a barcode with the specified text/number (the meaning varies based on the type of barcode) and type. Type is a value from the datasheet or class-level variables like UPC_A, etc. for convenience. Note the type value changes depending on the firmware version so use class-level values where possible!

reset()
Reset the printer.

send_command(*command*)
Send a command string to the printer.

set_defaults()
Set default printing and text options. This is useful to reset back to a good state after printing different size, weight, etc. text.

size
Set the size of text, must be a value of SIZE_SMALL, SIZE_MEDIUM, or SIZE_LARGE.

tab()
Print a tab (i.e. move to next 4 character block). Note this is only supported on more recent firmware printers!

test_page()
Print a test page.

underline
Set the underline state of the text, must be None (off), UNDERLINE_THIN, or UNDERLINE_THICK.

warm_up(*heat_time=120*)
Initialize the printer. Can specify an optional heat_time keyword to override the default heating timing of 1.2 ms. See the datasheet for details on the heating time value (duration in 10uS increments). Note that calling this function will take about half a second for the printer to intialize and warm up.

4.3 adafruit_thermal_printer.thermal_printer_264.ThermalPrinter

Thermal printer control module built to work with small serial thermal receipt printers. Note that these printers have many different firmware versions and care must be taken to select the appropriate module inside this package for your firmware printer:

- thermal_printer = The latest printers with firmware version 2.68+
- thermal_printer_264 = Printers with firmware version 2.64 up to 2.68.
- thermal_printer_legacy = Printers with firmware version before 2.64.
- Author(s): Tony DiCola

class adafruit_thermal_printer.thermal_printer_264.**ThermalPrinter**(*uart*, *byte_delay_s=0.00057346*, *dot_feed_s=0.0021*, *dot_print_s=0.03*)
Thermal printer for printers with firmware version 2.64 up to (but NOT including) 2.68.

4.4 adafruit_thermal_printer.thermal_printer_legacy.ThermalPrinter

Thermal printer control module built to work with small serial thermal receipt printers. Note that these printers have many different firmware versions and care must be taken to select the appropriate module inside this package for your firmware printer:

- thermal_printer = The latest printers with firmware version 2.68+
- thermal_printer_264 = Printers with firmware version 2.64 up to 2.68.
- thermal_printer_legacy = Printers with firmware version before 2.64.
- Author(s): Tony DiCola

class adafruit_thermal_printer.thermal_printer_legacy.**ThermalPrinter**(*uart*, *byte_delay_s=0.00057346*, *dot_feed_s=0.0021*, *dot_print_s=0.03*)

Thermal printer for printers with firmware version before 2.64.

feed(*lines*)

Advance paper by specified number of blank lines.

has_paper()

Return a boolean indicating if the printer has paper. You MUST have the serial RX line hooked up for this to work.

Note: be VERY CAREFUL to ensure your board can handle a 5V serial input before hooking up the RX line!

---

print_barcode(*text*, *barcode_type*)

Print a barcode with the specified text/number (the meaning varies based on the type of barcode) and type. Type is a value from the datasheet or class-level variables like UPC_A, etc. for convenience. Note the type value changes depending on the firmware version so use class-level values where possible!

reset()

Reset the printer.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

adafruit_thermal_printer.thermal_printer, 11
adafruit_thermal_printer.thermal_printer_264, 13
adafruit_thermal_printer.thermal_printer_legacy, 13

Index

A

adafruit_thermal_printer.thermal_printer (*module*), 11
adafruit_thermal_printer.thermal_printer_264 (*module*), 13
adafruit_thermal_printer.thermal_printer_legacy (*module*), 13

F

feed() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 12
feed() (*adafruit_thermal_printer.thermal_printer_legacy.ThermalPrinter method*), 14
feed_rows() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 12
flush() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 12

H

has_paper() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 12
has_paper() (*adafruit_thermal_printer.thermal_printer_legacy.ThermalPrinter method*), 14

I

inverse (*adafruit_thermal_printer.thermal_printer.ThermalPrinter attribute*), 12

J

justify (*adafruit_thermal_printer.thermal_printer.ThermalPrinter attribute*), 12

O

offline() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 12
online() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 12

P

print() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 12
print_barcode() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 12
print_barcode() (*adafruit_thermal_printer.thermal_printer_legacy.ThermalPrinter method*), 14

R

reset() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 12
reset() (*adafruit_thermal_printer.thermal_printer_legacy.ThermalPrinter method*), 14

S

send_command() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 12
set_defaults() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 12
size (*adafruit_thermal_printer.thermal_printer.ThermalPrinter attribute*), 13

T

tab() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 13
test_page() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 13
ThermalPrinter (*class in adafruit_thermal_printer.thermal_printer*), 12
ThermalPrinter (*class in adafruit_thermal_printer.thermal_printer_264*), 13
ThermalPrinter (*class in adafruit_thermal_printer.thermal_printer_legacy*), 13

U

underline (*adafruit_thermal_printer.thermal_printer.ThermalPrinter attribute*), 13

W

warm_up() (*adafruit_thermal_printer.thermal_printer.ThermalPrinter method*), 13