ThermalPrinter Documentation

Mickaël Schoentgen

Jun 02, 2026

CONTENTS

Python module to manage the DP-EH600 thermal printer (the one sold by AdaFruit).

- **Python 3.9** minimum;
- This is a clean follow of the technical manual with few helpers;
- Get the source code on GitHub;
- And report a bug;
- **Contibutors** are welcome, check the *developer guide*!

CHAPTER

ONE

SUPPORTED PRINTERS

> **Tip**
>
> If you had success with another printer, please tell us

Known printers to be supported:

- DP-EH600
- DP-EH400/1 (PR #17)
- QR701 (issue #15)

CHAPTER

TWO

INSTALLATION

2.1 From a Package Manager

```
python -m pip install -U thermalprinter
```

2.2 From Sources

Alternatively, you can get a copy of the module from GitHub.

1. Clone the repository:

```
git clone --depth=1 https://github.com/BoboTiG/thermalprinter.git
cd thermalprinter
```

2. Install the module:

```
python -m pip install -e .
```

CHAPTER

THREE

SETUP

Tip

If you work with another hardware, please tell us

3.1 Raspberry Pi

Note

Tested on Raspberry Pi 2, and 3.

1. Ensure that `ttyAMA0` is not used for serial console access. Edit the file `/boot/cmdline.txt` to remove all name-value pairs containing `ttyAMA0`.
2. Add the user to the **dialout** group:

```
sudo usermod -a -G dialout $USER
```

3. Reboot.

3.2 Beagle Bone

Note

The part of the documentation was taken from luopio/py-thermal-printer, and has not been tested.

```
# Mux settings
echo 1 > /sys/kernel/debug/omap_mux/spi0_sclk
echo 1 > /sys/kernel/debug/omap_mux/spi0_d0
```

CHAPTER

FOUR

USAGE

4.1 Instantiate the Class

Import the module:

```
from thermalprinter import ThermalPrinter
```

So the module can be used as a context manager:

```
with ThermalPrinter() as printer:
    # ...
```

Or:

```
printer = ThermalPrinter()
```

Refer to the *ThermalPrinter()* documentation for potential keyword-arguments.

4.2 A World of Infinite Possibilities

An example is better than a thousand words:

```
with ThermalPrinter() as printer:
    printer.out("Show time, here comes the demonstration!")
    printer.feed()
    printer.demo()
```

Where you can see the *ThermalPrinter.demo()* source code right here:

```
def demo(self) -> None:
    """Show time!

    Demonstrate printer capabilities.

    .. versionadded:: 1.0.0
    """

    # Image (requires the PIL Image library)
    self.feed()
    self.image(GNU_FILE)
    self.feed()
```

(continues on next page)

(continued from previous page)

```
    # Barcode
    self.barcode(
        "012345678901",
        BarCode.EAN13,
        width=3,
        height=80,
        left_margin=2,
        position=BarCodePosition.BELOW,
    )

    # Style
    self.out("Bold", bold=True)
    self.out("Double height", double_height=True)
    self.out("Double width", double_width=True)
    self.out("Font B mode", font_b=True)
    self.out("Inverse", inverse=True)
    self.out("Rotate 90°", rotate=True, codepage=CodePage.ISO_8859_1)
    self.out("Left blank", left_blank=10)
    self.out("Left margin", left_margin=5)
    self.out("Size LARGE", size=Size.LARGE)
    self.out("Strike", strike=True)
    self.out("Underline", underline=Underline.THIN)
    self.out("Upside down", upside_down=True)

    # Chinese (almost all alphabets exist)
    self.out("", chinese=True, chinese_format=Chinese.UTF_8)

    # Greek (excepted the  character)
    self.out("  !", codepage=CodePage.CP737)

    # Persian (check the recipes documentation for this to work)
    self.out(".     \n   ", persian=True)

    # Other characters
    self.out(b"Cards \xe8 \xe9 \xea \xeb", codepage=CodePage.CP932)

    # Accent
    self.out(
        "Voilà !",
        codepage=CodePage.ISO_8859_1,
        justify=Justify.CENTER,
        strike=True,
        underline=Underline.THICK,
    )

    self.feed(2)
```

CHAPTER

FIVE

API

5.1 Class

class thermalprinter.**ThermalPrinter**(*port: str = '/dev/ttyAMA0', * (Keyword-only parameters separator (PEP 3102)), baudrate: int = 19200, byte_time: float = 5e-05, command_timeout: float = 0.05, dot_feed_time: float = 0.0021, dot_print_time: float = 0.03, heat_interval: int = 2, heat_time: int = 80, most_heated_point: int = 9, read_timeout: float = 0.0, run_setup_cmd: bool = True, use_stats: bool = True, write_timeout: float = 1.0*)

The class managing the thermal printer.

Parameters

- **port** (*str*) – Serial port to use, known as the device name (see *constants.Defaults.PORT*).
- **baudrate** (*int*) – Baud rate (see *constants.Defaults.BAUDRATE*).
- **byte_time** (*float*) – Number of seconds to issue one byte to the printer. 11 bits (not 8) to accommodate idle, start, and stop, bits. See *constants.Defaults.BYTE_TIME*.
- **command_timeout** (*float*) – Time to sleep after issuing a command to the printer, in seconds.
- **dot_feed_time** (*float*) – Printer feed time, in seconds (see *constants.Defaults.DOT_FEED_TIME*).
- **dot_print_time** (*float*) – Printer dot time, in seconds (see *constants.Defaults.DOT_PRINT_TIME*).
- **heat_interval** (*int*) – Printer heat time interval (see *constants.Defaults.HEAT_INTERVAL*).
- **heat_time** (*int*) – Printer heat time (see *constants.Defaults.HEAT_TIME*).
- **most_heated_point** (*int*) – Printer most heated point (see *constants.Defaults.MOST_HEATED_POINT*).
- **read_timeout** (*float*) – Serial read timeout, in seconds (see *constants.Defaults.READ_TIMEOUT*).
- **run_setup_cmd** (*bool*) – Set to `False` to disable the automatic one-shot run of the printer settings command (that ay be problematic on some devices).
- **use_stats** (*bool*) – Set to `False` to disable statistics persistence. See *tools* for its usage.

- **write_timeout** (*float*) – Serial write timeout, in seconds (see *constants.Defaults.WRITE_TIMEOUT*).

Raises
ThermalPrinterValueError – On incorrect argument's type, or value.

Added in version 0.3.0: The `command_timeout` keyword-argument.

Added in version 1.0.0: `byte_time`, `dot_feed_time`, `dot_print_time`, `run_setup_cmd`, `read_timeout`, `use_stats`, and `write_timeout`, keyword-arguments.

5.2 Methods

5.2.1 Printing

`ThermalPrinter.demo()` → None

Show time!

Demonstrate printer capabilities.

Added in version 1.0.0.

`ThermalPrinter.feed(number: int = 1)` → None

Feed by the specified number of lines.

Parameters
number (*int*) – The number of lines (min=0, max=255).

Raises
ThermalPrinterValueError – On incorrect `number`'s type, or value.

`ThermalPrinter.out(data: Any, line_feed: bool = True, **kwargs: Any)` → None

Send one line to the printer.

Parameters

- **data** (*mixed*) – The data to print.
- **line_feed** (*bool*) – Wether or not to issue a final line feed (\n character).
- **kwargs** (*dict*) – Additional styles to apply.

You can pass formatting instructions directly via arguments:

```
>>> printer.out(data, justify=Justify.CENTER, inverse=True)
```

This is a quicker way to do:

```
>>> printer.justify(Justify.CENTER)
>>> printer.inverse(True)
>>> printer.out(data)
>>> printer.inverse(False)
>>> printer.justify(Justify.LEFT)
```

Hint

A special boolean keyword-argument can be used to print Persian text: `persian=True`. It will reshape `data`, and set proper text styles to issue the final text.

See *recipes* for required dependencies.

Added in version 1.0.0: The `persian` keyword-argument.

5.2.2 Barcodes

`ThermalPrinter.`**`barcode`**(*data: str*, *barcode_type:* BarCode, ***kwargs: Any*) → None

Barcode printing. All checks are done to ensure the data validity.

Parameters

- **`data`** (*`str`*) – The data to print.
- **`barecode_type`** (`BarCode`) – The barcode type to use.
- **`kwargs`** (*`dict`*) – Additional barcode properties to apply.

Raises

`ThermalPrinterValueError` – On incorrect `data`'s type, or value.

You can set additional barcode properties via arguments:

```
>>> printer.barcode(
...     "012345678901",
...     BarCode.EAN13,
...     width=3,
...     height=80,
...     left_margin=2,
...     position=BarCodePosition.BELOW)
```

This is a quicker way to do:

```
>>> printer.barcode_width(3)
>>> printer.barcode_height(80)
>>> printer.barcode_left_margin(2)
>>> printer.barcode_position(BarCodePosition.BELOW)
>>> printer.barcode("012345678901", BarCode.EAN13)
```

Added in version 1.0.0: The `kwargs` keyword-argument to set additional barcode properties.

`ThermalPrinter.`**`barcode_height`**(*height: int = 162*) → None

Set the barcode height.

Parameters

`height` (*`int`*) – The barcode height (min=1, max=255).

Raises

`ThermalPrinterValueError` – On incorrect `height`'s type, or value.

`ThermalPrinter.`**`barcode_left_margin`**(*margin: int = 0*) → None

Set the left margin of the barcode.

Parameters

`margin` (*`int`*) – The barcode left margin (min=0, max=255).

Raises

`ThermalPrinterValueError` – On incorrect `margin`'s type, or value.

ThermalPrinter.**barcode_position**(*position:* BarCodePosition = *BarCodePosition.HIDDEN*) → None

Set the position of the text relative to the barcode.

Parameters
position (BarCodePosition) – The barcode position to use.

ThermalPrinter.**barcode_width**(*width:* *int* = *3*) → None

Set the barcode width.

Parameters
width (*int*) – The barcode with (min=2, max=6).

Raises
ThermalPrinterValueError – On incorrect `width`'s type, or value.

static ThermalPrinter.**validate_barcode**(*data:* *str*, *barcode_type:* BarCode) → None

Validate data against the barcode type.

Parameters

- **data** (*str*) – The data to print.
- **barecode_type** (BarCode) – The barcode type to validate.

Raises
ThermalPrinterValueError – On incorrect `data`'s type, or value.

Added in version 1.0.0.

5.2.3 Images

ThermalPrinter.**image**(*image: Any*) → None

Picture printing.

Requires the Python Imaging Library (Pillow). The image will be resized to 384 pixels width (*constants.MAX_IMAGE_WIDTH*) if necessary, and converted to 1-bit without diffusion dithering.

Parameters
image (*str | pathlib.Path | PIL.Image*) – The file, or PIL Image object, to print.

Examples:

```
>>> printer.image("picture.png")
```

```
>>> from pathlib import Path
>>> printer.image(Path.home() / "picture.png")
```

```
>>> from PIL import Image
>>> printer.image(Image.open("picture.png"))
```

Changed in version 1.0.0: `image` can also be a `str`, or `pathlib.Path`, in addition to the original `PIL.Image` object.

Tip

Since **v1.0.0** the image will be automatically resized when too wide.

> **Important**
>
> It works better with white background instead of transparence.

ThermalPrinter.**image_chunks**(*image: Any*) → bytearray

TODO Convert a given `image` to 1-bit without diffusion dithering, *if necessary*.

Parameters
image (*PIL.Image*) – The PIL Image object to handle.

Return type
bytearray

Returns
The converted image object, if converted, else the original `image`.

> **Hint**
>
> Usually you do not need to call this method manually. It is used automatically by the *image()* method.

Added in version 1.0.0.

ThermalPrinter.**image_convert**(*image: Any*) → Any

Convert a given `image` to 1-bit without diffusion dithering, *if necessary*.

Parameters
image (*PIL.Image*) – The PIL Image object to convert.

Return type
PIL.Image

Returns
The converted image object, if converted, else the original `image`.

> **Hint**
>
> Usually you do not need to call this method manually. It is used automatically by the *image()* method.

Added in version 1.0.0.

ThermalPrinter.**image_resize**(*image: Any*) → Any

Resize a given `image` to fit into the maximum width of 384 pixels (*constants.MAX_IMAGE_WIDTH*), *if necessary*. The size proportion will be respected.

Parameters
image (*PIL.Image*) – The PIL Image object to resize.

Return type
PIL.Image

Returns
The resized image object, if resized, else the original `image`.

> **Hint**
>
> Usually you do not need to call this method manually. It is used automatically by the *image()* method.

Added in version 1.0.0.

5.2.4 Text Styling

ThermalPrinter.**bold**(*state: bool = False*) → None

Turn on/off the emphasized mode.

Parameters
state (*bool*) – Enabled if `state` is `True`, else disabled.

ThermalPrinter.**char_spacing**(*spacing: int = 0*) → None

Set the character spacing.

Parameters
spacing (*int*) – The spacing to use (min=0, max=255).

Raises
ThermalPrinterValueError – On incorrect `spacing`'s type, or value.

ThermalPrinter.**double_height**(*state: bool = False*) → None

Turn on/off the double height mode.

Parameters
state (*bool*) – Enabled if `state` is `True`, else disabled.

ThermalPrinter.**double_width**(*state: bool = False*) → None

Turn on/off the double width mode.

Parameters
state (*bool*) – Enabled if `state` is `True`, else disabled.

ThermalPrinter.**font_b**(*state: bool = False*) → None

Turn on/off the font B mode.

Parameters
state (*bool*) – Enabled if `state` is `True`, else disabled.

Added in version 1.0.0.

ThermalPrinter.**inverse**(*state: bool = False*) → None

Turn on/off the white/black reverse printing mode.

Parameters
state (*bool*) – Enabled if `state` is `True`, else disabled.

ThermalPrinter.**justify**(*value: Justify = Justify.LEFT*) → None

Set the text justification.

Parameters
value (`Justify`) – The justification to use.

Changed in version 1.0.0: The `value` keyword-argument was converted from a `str` to *constants.Justify*.

ThermalPrinter.**left_blank**(*value: int = 0*) → None

Set the left margin, in points.

Parameters
value (*int*) – Value to pass to the printer (min=0, max=255).

Raises
ThermalPrinterValueError – On incorrect `value`'s type, or value.

Added in version 1.0.0.

ThermalPrinter.**left_margin**(*margin: int = 0*) → None

Set the left margin, in 8-points.

Parameters
margin (*int*) – The new margin (min=0, max=47).

Raises
ThermalPrinterValueError – On incorrect `margin`'s type, or value.

ThermalPrinter.**line_spacing**(*spacing: int = 30*) → None

Set the line spacing.

Parameters
spacing (*int*) – The new spacing (min=0, max=255).

Raises
ThermalPrinterValueError – On incorrect `spacing`'s type, or value.

ThermalPrinter.**rotate**(*state: bool = False*) → None

Turn on/off 90° clockwise rotation.

Parameters
state (*bool*) – Enabled if `state` is `True`, else disabled.

ThermalPrinter.**size**(*value: Size = Size.SMALL*) → None

Set the text size.

Parameters
value (`Size`) – The size to use.

Note

This method affects *max_column*.

Changed in version 1.0.0: The `value` keyword-argument was converted from a `str` to *constants.Size*.

ThermalPrinter.**strike**(*state: bool = False*) → None

Turn on/off the double-strike mode.

Parameters
state (*bool*) – Enabled if `state` is `True`, else disabled.

ThermalPrinter.**underline**(*weight: Underline = Underline.OFF*) → None

Set the underline mode.

Parameters
weight (`Underline`) – The underline weight to use.

Changed in version 1.0.0: The `weight` keyword-argument was converted from an `int` to *constants.Underline*.

ThermalPrinter.**upside_down**(*state: bool = False*) → None

Turns on/off the upside-down printing mode.

Parameters
state (*bool*) – Enabled if `state` is `True`, else disabled.

5.2.5 Encoding and Charsets

ThermalPrinter.**charset**(*charset: CharSet = CharSet.USA*) → None

Set the character set.

Parameters
charset (`CharSet`) – The new charset to use.

ThermalPrinter.**codepage**(*codepage: CodePage = CodePage.CP437*) → None

Set the character code table.

Parameters
codepage (`CodePage`) – The new code page to use.

5.2.6 Chinese

ThermalPrinter.**chinese**(*state: bool = False*) → None

Turn on/off Chinese mode.

Parameters
state (*bool*) – Enabled if `state` is `True`, else disabled.

ThermalPrinter.**chinese_format**(*fmt: Chinese = Chinese.GBK*) → None

Set the Chinese format.

Parameters
fmt (`Chinese`) – The new Chinese format to use.

5.2.7 Printer State

ThermalPrinter.**offline**() → None

Take the printer offline. Upcoming print commands issued will be ignored until *online()* is called.

ThermalPrinter.**online**() → None

Take the printer online. Subsequent print commands will be obeyed.

ThermalPrinter.**sleep**(*seconds: int = 1*) → None

Put the printer into a low-energy state.

Parameters
seconds (*int*) – Value to pass to the printer (min=0).

Raises
ThermalPrinterValueError – On incorrect `seconds`'s type, or value.

ThermalPrinter.**status**(*, *raise_on_error: bool = True*) → dict[str, bool]

Return the printer status.

Parameters
raise_on_error (*bool*) – Raise on error.

Raises
ThermalPrinterCommunicationError – If the RX pin is not connected, and if `raise_on_error` is `True`.

Return type
dict[str, bool]

Returns
See *status_to_dict()*.

Added in version 0.2.0: The `raise_on_error` keyword-argument.

Removed in version 1.0.0: The `movement` key as it would always be `False`.

static ThermalPrinter.**status_to_dict**(*stat: int*) → dict[str, bool]

Return the printer status as a `dict`.

Return type
dict[str, bool]

Returns

Contains those keys:

- `paper`: `False` if no paper
- `temp`: `False` if the temperature exceeds 60°C
- `voltage`: `False` if the voltage is higher than 9.5V

Added in version 1.0.0.

ThermalPrinter.**reset**() → None

Reset the printer to factory defaults.

ThermalPrinter.**test**() → None

Print the test page (including printer's settings).

ThermalPrinter.**wake**() → None

Wake up the printer.

5.2.8 Special Methods

ThermalPrinter.**close**() → None

Persist statistics, *if desired*, and close the serial port.

ThermalPrinter.**flush**(*clear: bool = False*) → None

Remove the print data from the output buffer.

Parameters
clear (*bool*) – Set to `True` to also clear the input buffer.

ThermalPrinter.**init**(*heat_time: int*) → None

Set printer heat properties.

Parameters
heat_time (*int*) – Printer heat time.

Added in version 1.0.0.

ThermalPrinter.**print_char**(*char: str*) → None

Test one character with all supported code pages.

Parameters
char (*str*) – The character to print.

Say you are looking for the good code page to print a sequence, you can print it using every code pages:

```
>>> printer.print_char("")
```

Added in version 1.0.0.

ThermalPrinter.**send_command**(*command:* Command, **args: int*) → None

Send a command to the printer.

Parameters

- **command** (Command) – The command to send to the printer.
- **args** (*list[int]*) – Eventual command arguments.

ThermalPrinter.**to_bytes**(*data: Any*) → bytes

Convert data before sending to the printer.

Parameters
data (*mixed*) – Any type of data to print.

Return type
bytes

Returns
The converted data in bytes.

5.3 Attributes

All these attributes are **read-only**.

property ThermalPrinter.**feeds: int**

Number of printed line feeds.

property ThermalPrinter.**has_paper: bool**

Return `True` if there is paper.

property ThermalPrinter.**is_online: bool**

The printer online status.

property ThermalPrinter.**is_sleeping: bool**

The printer sleep status.

property ThermalPrinter.**lines: int**

Number of printed lines.

property ThermalPrinter.**max_column**: int

Number of printable characters on one line.

5.4 Constants

5.4.1 Barcode Types

enum thermalprinter.constants.**BarCode**(*value*)

Barcode types.

Valid values are as follows:

UPC_A = <BarCode.UPC_A: 65, range: 11 <= len(data) <= 12>

UPC_E = <BarCode.UPC_E: 66, range: 11 <= len(data) <= 12>

JAN13 = <BarCode.JAN13: 67, range: 12 <= len(data) <= 13>

JAN8 = <BarCode.JAN8: 68, range: 7 <= len(data) <= 8>

CODE39 = <BarCode.CODE39: 69, range: 1 <= len(data) <= 255>

ITF = <BarCode.ITF: 70, range: 1 <= len(data) <= 255>

CODABAR = <BarCode.CODABAR: 71, range: 1 <= len(data) <= 255>

CODE93 = <BarCode.CODE93: 72, range: 1 <= len(data) <= 255>

CODE128 = <BarCode.CODE128: 73, range: 2 <= len(data) <= 255>

5.4.2 Barcode Positions

enum thermalprinter.constants.**BarCodePosition**(*value*)

Barcode positions.

Valid values are as follows:

HIDDEN = <BarCodePosition.HIDDEN: 0>

ABOVE = <BarCodePosition.ABOVE: 1>

BELOW = <BarCodePosition.BELOW: 2>

BOTH = <BarCodePosition.BOTH: 3>

5.4.3 Characters Sets

enum thermalprinter.constants.**CharSet**(*value*)

Character sets.

Valid values are as follows:

USA = <CharSet.USA: 0>

FRANCE = <CharSet.FRANCE: 1>

GERMANY = <CharSet.GERMANY: 2>

UK = <CharSet.UK: 3>

DENMARK = <CharSet.DENMARK: 4>

SWEDEN = <CharSet.SWEDEN: 5>

ITALY = <CharSet.ITALY: 6>

SPAIN = <CharSet.SPAIN: 7>

JAPAN = <CharSet.JAPAN: 8>

NORWAY = <CharSet.NORWAY: 9>

DENMARK2 = <CharSet.DENMARK2: 10>

SPAIN2 = <CharSet.SPAIN2: 11>

LATIN_AMERICAN = <CharSet.LATIN_AMERICAN: 12>

KOREA = <CharSet.KOREA: 13>

SLOVENIA = <CharSet.SLOVENIA: 14>

CHINA = <CharSet.CHINA: 15>

5.4.4 Chinese Formats

enum thermalprinter.constants.**Chinese**(*value*)

Chinese formats.

Valid values are as follows:

GBK = <Chinese.GBK: 0>

UTF_8 = <Chinese.UTF_8: 1>

BIG5 = <Chinese.BIG5: 3>

5.4.5 Code Pages

enum thermalprinter.constants.**CodePage**(*value*)

Character code tables.

Valid values are as follows:

CP437 = <CodePage.CP437: (0, 'the United States of America, European standard')>

CP932 = <CodePage.CP932: (1, 'Katakana')>

CP850 = <CodePage.CP850: (2, 'Multi language')>

CP860 = <CodePage.CP860: (3, 'Portuguese')>

CP863 = <CodePage.CP863: (4, 'Canada, French')>

CP865 = <CodePage.CP865: (5, 'Western Europe')>

CYRILLIC = <CodePage.CYRILLIC: (6, 'The Slavic language')>

```
CP866 = <CodePage.CP866:  (7, 'The Slavic 2')>
MIK = <CodePage.MIK: (8, 'The Slavic / Bulgaria')>
CP755 = <CodePage.CP755:  (9, 'Eastern Europe, Latvia 2')>
IRAN = <CodePage.IRAN: (10, 'Iran, Persia')>
CP862 = <CodePage.CP862:  (15, 'Hebrew')>
CP1252 = <CodePage.CP1252:  (16, 'Latin 1 [WCP1252]')>
CP1253 = <CodePage.CP1253:  (17, 'Greece [WCP1253]')>
CP852 = <CodePage.CP852:  (18, 'Latina 2')>
CP858 = <CodePage.CP858:  (19, 'A variety of language Latin 1 + Europe')>
IRAN2 = <CodePage.IRAN2:  (20, 'Persian')>
LATVIA = <CodePage.LATVIA: (21, '')>
CP864 = <CodePage.CP864:  (22, 'Arabic')>
ISO_8859_1 = <CodePage.ISO_8859_1:  (23, 'Western Europe')>
CP737 = <CodePage.CP737:  (24, 'Greece')>
CP1257 = <CodePage.CP1257:  (25, 'The Baltic Sea')>
THAI = <CodePage.THAI: (26, 'Thai Wen')>
CP720 = <CodePage.CP720:  (27, 'Arabic')>
CP855 = <CodePage.CP855:  (28, '')>
CP857 = <CodePage.CP857:  (29, 'Turkish')>
CP1250 = <CodePage.CP1250:  (30, 'Central Europe [WCP1250]')>
CP775 = <CodePage.CP775:  (31, '')>
CP1254 = <CodePage.CP1254:  (32, 'Turkish [WCP1254]')>
CP1255 = <CodePage.CP1255:  (33, 'Hebrew [WCP1255]')>
CP1256 = <CodePage.CP1256:  (34, 'Arabic [WCP1256]')>
CP1258 = <CodePage.CP1258:  (35, 'Vietnamese [WCP1258]')>
ISO_8859_2 = <CodePage.ISO_8859_2:  (36, 'Latin 2')>
ISO_8859_3 = <CodePage.ISO_8859_3:  (37, 'Latin 3')>
ISO_8859_4 = <CodePage.ISO_8859_4:  (38, 'Baltic languages')>
ISO_8859_5 = <CodePage.ISO_8859_5:  (39, 'The Slavic language')>
ISO_8859_6 = <CodePage.ISO_8859_6:  (40, 'Arabic')>
ISO_8859_7 = <CodePage.ISO_8859_7:  (41, 'Greece')>
```

ISO_8859_8 = <CodePage.ISO_8859_8: (42, 'Hebrew')>

ISO_8859_9 = <CodePage.ISO_8859_9: (43, 'Turkish')>

ISO_8859_15 = <CodePage.ISO_8859_15: (44, 'Latin 9')>

THAI2 = <CodePage.THAI2: (45, 'Thai Wen 2')>

CP856 = <CodePage.CP856: (46, '')>

CP874 = <CodePage.CP874: (47, '')>

5.4.6 Code Pages Fallback

enum thermalprinter.constants.**CodePageConverted**(*value*)

Some code pages are not available on Python, so we use a little translation table.

Valid values are as follows:

MIK = <CodePageConverted.MIK: 'ISO-8859-5'>

CP755 = <CodePageConverted.CP755: 'UTF-8'>

IRAN = <CodePageConverted.IRAN: 'CP1256'>

LATVIA = <CodePageConverted.LATVIA: 'ISO-8859-4'>

THAI = <CodePageConverted.THAI: 'ISO-8859-11'>

> **Note**
>
> If you find a better fit for one of the code page below, open an issue please (or better: send a patch)

5.4.7 Commands

enum thermalprinter.constants.**Command**(*value*)

Codes used to send commands.

Valid values are as follows:

NONE = <Command.NONE: 0>

DC2 = <Command.DC2: 18>

ESC = <Command.ESC: 27>

FS = <Command.FS: 28>

GS = <Command.GS: 29>

5.4.8 Defaults Parameters

enum thermalprinter.constants.**Defaults**(*value*)

Default printer parameters.

> **Hint**
>
> Most of them can be tweaked by special environment variables following the given syntax: `TP_XXX`. Example: `export TP_BAUDRATE=9600` will set `Defaults.BAUDRATE` to `9600`.

Valid values are as follows:

BARCODE_HEIGHT = <Defaults.BARCODE_HEIGHT: 162>

BARCODE_WIDTH = <Defaults.BARCODE_WIDTH: 3>

BAUDRATE = <Defaults.BAUDRATE: 19200>

BYTE_TIME = <Defaults.BYTE_TIME: 5e-05>

DOT_FEED_TIME = <Defaults.DOT_FEED_TIME: 0.0021>

DOT_PRINT_TIME = <Defaults.DOT_PRINT_TIME: 0.03>

HEAT_INTERVAL = <Defaults.HEAT_INTERVAL: 2>

HEAT_TIME = <Defaults.HEAT_TIME: 80>

LINE_SPACING = <Defaults.LINE_SPACING: 30>

MOST_HEATED_POINT = <Defaults.MOST_HEATED_POINT: 9>

PORT = <Defaults.PORT: '/dev/ttyAMA0'>

READ_TIMEOUT = <Defaults.READ_TIMEOUT: 0.0>

WRITE_TIMEOUT = <Defaults.WRITE_TIMEOUT: 1.0>

5.4.9 Text Justification

enum thermalprinter.constants.**Justify**(*value*)

Text justifications.

Valid values are as follows:

LEFT = <Justify.LEFT: 0>

CENTER = <Justify.CENTER: 1>

RIGHT = <Justify.RIGHT: 2>

5.4.10 Text Size

enum thermalprinter.constants.**Size**(*value*)

Text sizes.

- `MEDIUM` will double the text height.
- `LARGE` will both double the text height, and its width.

Valid values are as follows:

SMALL = <Size.SMALL: (0, 24, 32)>

MEDIUM = <Size.MEDIUM: (1, 48, 32)>

LARGE = <Size.LARGE: (17, 48, 16)>

5.4.11 Text Underline

enum thermalprinter.constants.**Underline**(*value*)

Text underline weights.

Valid values are as follows:

OFF = <Underline.OFF: 0>

THIN = <Underline.THIN: 1>

THICK = <Underline.THICK: 2>

5.4.12 Other

thermalprinter.constants.**MAX_IMAGE_WIDTH = 384**

Max image width.

thermalprinter.constants.**STATS_FILE = '~/.thermalprinter.json'**

Printer statistics file. See *tools* for its usage.

5.5 Exceptions

exception thermalprinter.exceptions.**ThermalPrinterError**

Base class for thermal printer exceptions.

exception thermalprinter.exceptions.**ThermalPrinterCommunicationError**

Raised on communication error with the printer.

exception thermalprinter.exceptions.**ThermalPrinterValueError**

Raised on incorrect type, or value, passed to any method.

CHAPTER

SIX

RECIPES

Added in version 1.0.0: Recipes are extras features that you can install on-demand.

Hint

When an executable is made available, you can tweak printer properties using `TP_*` environment variables. See *thermalprinter.constants.Defaults*.

6.1 Calendar

Print daily stuff from your calendar, and birthdays as a bonus!

Note

All texts are in French by default. You can tweak those constants to better fit your needs:

`thermalprinter.recipes.calendar.BIRTHDAY = "C'est l'anniversaire de..."`
Birthdays introduction.

`thermalprinter.recipes.calendar.MONTH_NAMES = ['Janvier', 'Février', 'Mars', 'Avril', 'Mai', 'Juin', 'Juillet', 'Août', 'Septembre', 'Octobre', 'Novembre', 'Décembre']`
Months names.

`thermalprinter.recipes.calendar.NICE_DAY = 'Belle journée :)'`
Nice words printed at the end.

`thermalprinter.recipes.calendar.TOMORROW = 'demain'`
Tomorrow.

Added in version 2.0.0.

thermalprinter.recipes.calendar.**UNTIL = "Jusqu'à"**

Until.

Added in version 2.0.0.

thermalprinter.recipes.calendar.**WHOLE_DAY = 'Toute la journée'**

When an event takes the whole day.

Installation:

```
sudo apt install libcairo2
python -m pip install -U 'thermalprinter[calendar]'
```

There is an executable made available:

```
print-calendar --help
```

Here is the API:

class thermalprinter.recipes.calendar.**Calendar**(*url: str*, *printer:* ThermalPrinter | *None* = *None*)

Print daily stuff from your calendar, and birthdays as a bonus!

Parameters

- **url** (*str*) – The calendar URL.
- **printer** (thermalprinter.ThermalPrinter | *None*) – Optional printer to use.

Calendar.**start**() → None

Where all the magic happens.

thermalprinter.recipes.calendar.**forge_header_image**(*now: datetime*) → <module 'PIL.Image' from '/home/docs/checkouts/readthedocs.org/user_builds/thermalprinter/en packages/PIL/Image.py'>

Create the image object containing the nice image with current month, and day.

Parameters

now (*datetime.datetime*) – The current date to compare event's dates to.

Return type

PIL.Image

Returns

A PNG file-like PIL.Image object.

Added in version 2.0.0: It was a class method in v1.0.0.

thermalprinter.recipes.calendar.**format_event_date**(*now: datetime*, *event: Event*) → str

Given the current date, and an iCal event, return the formated event's start, and end.

Parameters

- **now** (*datetime.datetime*) – The current date to compare event's dates to.
- **event** (*icalendar.cal.Event*) – The iCal event.

Return type

str

Returns
The formated event duration.

Added in version 2.0.0.

thermalprinter.recipes.calendar.**localize**(*event_date: datetime*, *dst_tz: ZoneInfo*) → datetime

Given an event date, and a timezone, return the localized date.

Parameters

- **event_date** (*datetime.datetime*) – The event date.
- **dst_tz** (*zoneinfo.ZoneInfo*) – The destination timezone to localize the event date to.

Return type
datetime.datetime

Returns
The localized event date.

Added in version 2.1.0.

thermalprinter.recipes.calendar.**TIMEZONE = 'Europe/Paris'**

The timezone to display proper hours.

thermalprinter.recipes.calendar.**BIRTHDAYS_FILE = '~/.birthdays.lst'**

File containing birthdays.

Hint

The content of this file is as follow:

```
YYYY-MM-DD = Alice
YYYY-MM-DD = Bob
```

6.2 Persian

Persian text uses non standard codes, and it is quite painful to print it out-of-the-box.

So this extra allowes you to print Persian text as easy as follow (you provide the `persian` keyword-argument, and the magic happens under the wood):

```
printer.out("...", persian=True)
```

Installation:

```
python -m pip install -U 'thermalprinter[persian]'
```

Here is the API:

thermalprinter.recipes.persian.**IRAN_SYSTEM_MAP**

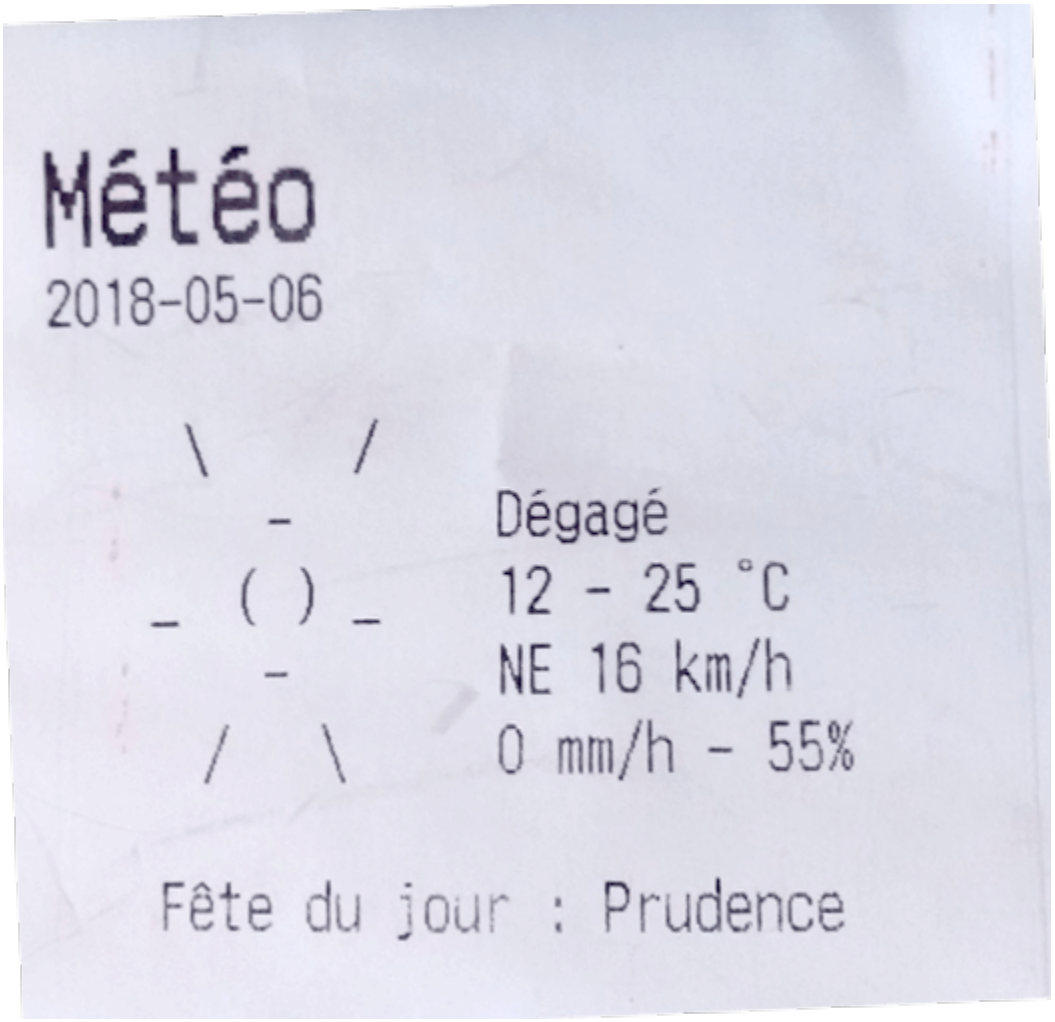
Unicode translations for the Iran code page.

Note

Credits go to @ghorbanpirizad in issue #4.

6.3 Weather

How cool is it to have the weather printed every morning, alongside with the saint of the day?



Note

All texts are in French by default. You can tweak those constants to better fit your needs:

`thermalprinter.recipes.weather.DESCRIPTIONS`

That is a big one, kept synced with OWM weather-conditions.

thermalprinter.recipes.weather.**TITLE** = 'Météo'

Title.

thermalprinter.recipes.weather.**SAINT_OF_THE_DAY** = 'Fête du jour : {}'

Prefix before the saint of the day.

thermalprinter.recipes.weather.**NORTH** = 'N'

The North cord point abbreviation.

thermalprinter.recipes.weather.**EAST** = 'E'

The East cord point abbreviation.

thermalprinter.recipes.weather.**SOUTH** = 'S'

The South cord point abbreviation.

thermalprinter.recipes.weather.**WEST** = 'O'

The West cord point abbreviation.

Installation:

```
python -m pip install -U 'thermalprinter[weather]'
```

There is an executable made available:

```
print-weather --help
```

Here is the API:

class thermalprinter.recipes.weather.**Weather**(*lat: float*, *lon: float*, *appid: str*, *printer:* ThermalPrinter | *None = None*)

Print the weather of the day alongside with the saint of the day. Using the data from OpenWeatherMap.

Parameters

- **lat** (*float*) – Location latitude.
- **lon** (*float*) – Location longitude.
- **appid** (*str*) – OpenWeatherMap API key.
- **printer** (thermalprinter.ThermalPrinter | *None*) – Optional printer to use.

Weather.**start**() → None

Where all the magic happens.

thermalprinter.recipes.weather.**mps_to_kmph**(*mps: float*) → int

Convert the wind unity from m/sec to km/h.

Parameters
mps (*float*) – The input value in m/sec.

Return type
int

Returns
The converted value to km/h.

thermalprinter.recipes.weather.**wind_dir**(*angle: float*) → bytes | str

Get the corresponding wind direction arrow, or the cord point abbreviation.

Parameters
angle (*float*) – The wind angle.

Return type
bytes | str

Returns

The cord point abbreviation. Either bytes for arrows, or plain string.

> **Note**
>
> Bytes values are *thermalprinter.constants.CodePage* THAI2 characters.

thermalprinter.recipes.weather.**ASCII_ARTS**

Beautiful weather ASCII arts, copied from schachmat/wego.

thermalprinter.recipes.weather.**SAINTS_FILE = './saints.lst'**

File containing calendar saints.

Added in version 2.0.0.

thermalprinter.recipes.weather.**URL = 'https://api.openweathermap.org/data/2.5/onecall?lat={lat}&lon={lon}&units=metric&appid={appid}'**

OpenWeatherMap API URL.

thermalprinter.recipes.weather.**USER_AGENT = 'Mozilla/5.0 (X11; Linux x86_64; rv:133.0) Gecko/20100101 Firefox/133.0'**

User-Agent HTTP header used to fetch OpenWeatherMap data.

Added in version 2.0.0.

thermalprinter.recipes.weather.**TIMEZONE = 'Europe/Paris'**

The timezone to display proper dates.

CHAPTER

SEVEN

TOOLS

7.1 Constants

thermalprinter.tools.**ls**(**consts: type[enum.Enum]*) → None

Print constants values.

Parameters
constants (*list[type[enum.Enum]]*) – Constant(s) to print.

Print all constants:

```
>>> ls()
```

Print *thermalprinter.constants.Chinese* constant values:

```
>>> ls(Chinese)
```

Print *thermalprinter.constants.Chinese*, and *thermalprinter.constants.CodePage*, constant values:

```
>>> ls(Chinese, CodePage)
```

7.2 Statistics

Simple printer statistics can be persisted into the *thermalprinter.constants.STATS_FILE* file. They are enabled by default unless `use_stats` is set to `False` (see *thermalprinter.ThermalPrinter()*).

thermalprinter.tools.**stats_file**() → Path

Return the full path to the statistics file.

thermalprinter.tools.**stats_load**() → dict[str, int]

Load statistics from the *thermalprinter.constants.STATS_FILE* file.

Return type
dict[str, int]

Returns

Contains those keys:

- `feeds`: total count of printed feeds
- `lines`: total count of printed lines

thermalprinter.tools.**stats_save**(*printer:* ThermalPrinter) → None

Save printer statistics to the *thermalprinter.constants.STATS_FILE* file.

Parameters

printer (ThermalPrinter) – The Printer.

CHAPTER

EIGHT

DEVELOPERS

8.1 Setup

1. First, you need to fork the GitHub repository.

   Note

   Always work on a **specific branch** dedicated to your patch.

2. Then, you could need the `Embedded printer DP-EH600 Technical Manual`, and take a look at the features advancement.
3. Finally, be sure to add/update tests, and documentation, within your patch.

8.2 Special Serial Device

Hint

Whenever you need a dummy serial device for your tests, you can use the `loop://` port.

8.3 Testing

8.3.1 Dependencies

Install required packages:

```
python -m pip install -e '.[tests]'
```

8.3.2 How to test?

```
python -m pytest
```

And you can enhance the *demo* if you introduced a styling method.

8.4 Validating the code

It is important to keep a clean base code.

8.4.1 Dependencies

Install required packages:

```
python -m pip install -e '.[lint]'
```

8.4.2 How to validate?

```
./checks.sh
```

8.5 Documentation

8.5.1 Dependencies

Install required packages:

```
python -m pip install -e '.[docs]'
```

8.5.2 How to build?

```
sphinx-build --color -W -bhtml docs/source docs/output
```

CHAPTER

NINE

MIGRATION

9.1 From v0.3 to v1.0

- *ThermalPrinter.justify()*:

```
+ from thermalprinter import Justify

- printer.justify("L")
+ printer.justify(Justify.LEFT)

- printer.justify("C")
+ printer.justify(Justify.CENTER)

- printer.justify("R")
+ printer.justify(Justify.RIGHT)
```

- *ThermalPrinter.size()*:

```
+ from thermalprinter import Size

- printer.size("S")
+ printer.size(Size.SMALL)

- printer.size("M")
+ printer.size(Size.MEDIUM)

- printer.size("L")
+ printer.size(Size.LARGE)
```

- *ThermalPrinter.underline()*:

```
+ from thermalprinter import Underline

- printer.underline(0)
+ printer.underline(Underline.OFF)

- printer.underline(1)
+ printer.underline(Underline.THIN)

- printer.underline(2)
+ printer.underline(Underline.THICK)
```

- tools.print_char() → *ThermalPrinter.print_char()*:

```
- from thermalprinter.tools import print_char

- print_char("")
+ printer.print_char("")
```

- tools.printer_tests() → *ThermalPrinter.demo()*:

```
- from thermalprinter.tools import printer_tests

- printer_tests()
+ printer.demo()
```

9.2 From v0.2 to v0.3

- tools.test_char() → tools.print_char():

```
- from thermalprinter.tools import test_char
+ from thermalprinter.tools import print_char

- test_char("")
+ print_char("")
```

- tools.testing() → tools.printer_tests():

```
- from thermalprinter.tools import testing
+ from thermalprinter.tools import printer_tests

- testing()
+ printer_tests()
```

CHAPTER

TEN

CHANGELOG

10.1 2.1.1-dev

Release date: `202x-xx-xx`

10.1.1 Bug Fixes

-

10.1.2 Features

-

10.1.3 Technical Changes

-

10.2 2.1.0

Release date: `2025-03-03`

10.2.1 Bug Fixes

- Calendar recipe: prevent whole-day events from day N-1 to also appear at day N.
- Calendar recipe: whole-day event date formatting.
- Calendar recipe: consistent spacing between the calendar picture and the next item.
- Weather recipe: shortened descriptions to fit into the remaining 18 characters wide.
- Tests: missing code adaptation in `test_calendar.py::test_get_events_on_multi_days()`.

10.2.2 Features

- Calendar recipe: support in-between recurring events.

10.2.3 Technical Changes

- Calendar recipe: removed the `icalevents` requirement.
- Calendar recipe: added `recurring-ical-events`, and `requests`, requirements.

10.3 2.0.0

Release date: 2025-01-10

10.3.1 Bug Fixes

- Weather recipe: fixed special characters handling (PR #49).
- Weather recipe: added the missing `byteorder` (keyword-)argument to `int.from_bytes()` calls for Python 3.9, and 3.10.

10.3.2 Features

- Calendar recipe: improve multi-days event display (issue #43).
- Calendar recipe: improve printing performances.
- Calendar recipe: feed before printing birthdays.
- Weather recipe: now using a custom User-Agent HTTP header to fetch OpenWeatherMap data.

10.3.3 Technical Changes

- Drop support for Python 3.7, and 3.8.
- Calendar recipe: added the *recipes.calendar.format_event_date()* function.
- Calendar recipe: moved the `recipes.calendar.Calendar.forge_header_image()` method to its own *recipes.calendar.forge_header_image()* function.
- Weather recipe: added the *recipes.weather.USER_AGENT* constant.
- Weather recipe: removed the `recipes.weather.UNKNOWN` constant.

10.4 1.0.0

Release date: 2024-12-17

10.4.1 Bug Fixes

- Reworked images printing to, hopefully, fix all issues.
- Fixed printed lines counting in *ThermalPrinter.out()*.

10.4.2 Features

- Support for QR701 printers is confirmed (issue #15).
- New extra: `calendar`, to print daily stuff from your calendar, and birthdays as a bonus! See *recipes*.
- New extra: `persian`, to make your life easier when printing Persian text, see *recipes*.
- New extra: `weather`, to print the weather alongside with the saint of the day! See *recipes*.
- New text styles: *ThermalPrinter.font_b()*, and *ThermalPrinter.left_blank()*.
- New options to tweak printer behaviors: `byte_time`, `dot_feed_time`, `dot_print_time`, `read_timeout`, and `write_timeout`. See *ThermalPrinter*.
- New option to control printer settings at initialization to *ThermalPrinter*: `run_setup_cmd=bool` (issue #15).

- It is now possible to pass barcode styling instructions in *ThermalPrinter.barcode()*, in the same way it's done for *ThermalPrinter.out()*.
- Introduced statistics persisted at exit. This behavior can be disabled by passing `use_stats=False` to *ThermalPrinter*.
- Enhanced the demonstration code.
- Rewrote the entire documentation to cover all possible stuff, and it is way prettier now, (thanks to the awesome Shibuya theme).
- Improved the documentation by fixing issues found with Harper.
- 100% tests coverage!
- Added lot of logs.

10.4.3 Technical Changes

- Added the *constants.Justify* constant to use in the *ThermalPrinter.justify()* method (**breaking change**).
- Added the *constants.Size* constant to use in the *ThermalPrinter.size()* method (**breaking change**).
- Added the *constants.Underline* constant to use in the *ThermalPrinter.underline()* method (**breaking change**).
- Added the *constants.Defaults* constant. And they can be tweaked via `TP_*` environment variables.
- Added the `ThermalPrinter.__exit__()` method to properly close the printer when leaving the context manager.
- Added the *ThermalPrinter.has_paper()* property.
- Added the *ThermalPrinter.close()* method.
- Added the *ThermalPrinter.demo()* method.
- Added the *ThermalPrinter.font_b()* method.
- Added the *ThermalPrinter.image_chunks()* method.
- Added the *ThermalPrinter.image_convert()* method.
- Added the *ThermalPrinter.image_resize()* method.
- Added the *ThermalPrinter.init()* method.
- Added the *ThermalPrinter.left_blank()* method.
- Added the *ThermalPrinter.status_to_dict()* method.
- Added the *tools.stats_file()* function.
- Added the *tools.stats_load()* function.
- Added the *tools.stats_save()* function.
- Moved the `tools.printer_tests()` function to the *ThermalPrinter.demo()* method (**breaking change**).
- Moved the `tools.print_char()` function to the *ThermalPrinter.print_char()* method (**breaking change**).
- Moved the `validate.validate_barcode()` function to the *ThermalPrinter.validate_barcode()* method.
- Removed the `validate.py` file, and most `validate_*()` functions.

- Removed the `exceptions.ThermalPrinterConstantError` class.

10.5 0.3.0

Release date: `2024-11-02`

10.5.1 Features

- Added support for Python 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, and 3.14.
- Support for DP-EH400/1 printers is confirmed (PR #17).
- Added type annotations.
- New option to specify commands timeout via `command_timeout=float)` in *ThermalPrinter* (PR #17).
- Documentation is now generated from the source code to never miss signature changes.
- Moved the CI from Travis-CI to GitHub actions.
- Run ruff on the entire source code.
- Added more quality checks.

10.5.2 Technical Changes

- Drop support for Python 3.5, and 3.6.
- Renamed `tools.test_char()` → `tools.print_char()`.
- Renamed `tools.testing()` → `tools.printer_tests()`.
- No longer checks that the provided `image` argument to *ThermalPrinter.image()* is a `PIL.Image` object.

10.5.3 Contributors

Thanks to our beloved contributors: @uniphil, @d21d3q

10.6 0.2.0

Release date: `2019-01-10`

10.6.1 Bug Fixes

- Fixed image printing in `tools.printer_tests()` when the module is installed. Will now raise an exception if `raise_on_error` argument is `True` (default).

10.6.2 Features

- Add communication error in the *ThermalPrinter.status()* (issue #3). Will now raise an exception if `raise_on_error` argument is `True` (default).
- Use `setup.cfg` instead of `setup.py`.

10.6.3 Technical Changes

- Removed `exceptions.ThermalPrinterAttributeError` exception.
- Attributes *ThermalPrinter.is_online*, *ThermalPrinter.is_sleeping*, *ThermalPrinter.lines*, *ThermalPrinter.feeds* and *ThermalPrinter.max_column* now raise `AttributeError` when trying to set them (previously raising `ThermalPrinterAttributeError`).
- Changed the signature of `tools.printer_tests(port='/dev/ttyAMA0', heat_time=80)()` → `tools.printer_tests(printer=None, raise_on_error=True)()`.
- Changed the signature of `tools.print_char(char)()` → `tools.print_char(char, printer=None)()`.

10.6.4 Contributors

Thanks to our beloved contributors: @d21d3q

10.7 0.1.0

Release date: `2016-05-24`

10.7.1 Features

- First working version.

10.7.2 Contributors

Thanks to our beloved contributors: @phillipthelen, and @AKokkalas

- genindex
- modindex
- search

PYTHON MODULE INDEX

t

INDEX

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W