python-escpos Documentation

Release 3.2.dev9+g1e9adb8

python-escpos developers

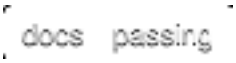
Apr 29, 2024

USER DOCUMENTATION

CHAPTER

ONE

DESCRIPTION



Python ESC/POS is a library which lets the user have access to all those printers handled by ESC/POS commands, as defined by Epson, from a Python application.

The library tries to implement the functions provided by the ESC/POS-command-set and supports sending text, images, barcodes and qr-codes to the printer.

Text can be aligned/justified and fonts can be changed by size, type and weight.

Also, this module handles some hardware functionalities like cutting paper, control characters, printer reset and similar functions.

Since supported commands differ from printer to printer the software tries to automatically apply the right settings for the printer that you set. These settings are handled by escpos-printer-db which is also used in escpos-php.

1.1 Dependencies

This library makes use of:

- pyusb for USB-printers
- Pillow for image printing
- qrcode for the generation of QR-codes
- pyserial for serial printers
- python-barcode for the generation of barcodes

1.2 Documentation and Usage

The basic usage is:

```
from escpos.printer import Usb

""" Seiko Epson Corp. Receipt Printer (EPSON TM-T88III) """
p = Usb(0x04b8, 0x0202, 0, profile="TM-T88III")
p.text("Hello World\n")
p.image("logo.gif")
p.barcode('4006381333931', 'EAN13', 64, 2, '', '')
p.cut()
```

Another example based on the Network printer class:

```
from escpos.printer import Network

kitchen = Network("192.168.1.100") #Printer IP Address
kitchen.text("Hello World\n")
kitchen.barcode('4006381333931', 'EAN13', 64, 2, '', '')
kitchen.cut()
```

Another example based on the Serial printer class:

```
from escpos.printer import Serial

""" 9600 Baud, 8N1, Flow Control Enabled """
p = Serial(devfile='/dev/tty.usbserial',
           baudrate=9600,
           bytesize=8,
           parity='N',
           stopbits=1,
           timeout=1.00,
           dsrdtr=True)

p.text("Hello World\n")
p.qr("You can readme from your smartphone")
p.cut()
```

The full project-documentation is available on Read the Docs.

1.3 Contributing

This project is open for any contribution! Please see CONTRIBUTING.rst for more information.

1.4 Disclaimer

None of the vendors cited in this project agree or endorse any of the patterns or implementations. Its names are used only to maintain context.

CHAPTER

TWO

CONTENT

2.1 User Documentation

This chapter describes the central points that are relevant to the user of this library.

2.1.1 Installation

Last Reviewed
2023-08-10

Installation with PIP

Installation should be rather straight-forward. python-escpos is on PyPi, so you can simply enter:

```
pip install python-escpos[all]
```

This should install all necessary dependencies. Apart from that python-escpos is for some versions also available as a Debian package. If you want to always benefit from the newest stable releases you should always install from PyPi. If you use the `--pre` parameter for `pip`, you will get the latest pre-release.

The following installation options exist:

- *all*: install all packages available for this platform
- *usb*: install packages required for USB printers
- *serial*: install packages required for serial printers
- *win32*: install packages required for win32 printing (only Windows)
- *cups*: install packages required for CUPS printing

Other installation methods

Officially, no other installation methods are supplied.

If you want to install nevertheless from another source, please make sure that you have received the correct package and that the capabilities data is properly integrated. When packaging from source please read the developer information in *Repository*.

If your packaging method breaks the resource system from setuptools, it might be necessary to supply the path to the capabilities file: *Advanced Usage: change capabilities-profile*.

Setup udev for USB-Printers

1. Get the *Product ID* and *Vendor ID* from the lsusb command `# lsusb Bus 002 Device 001: ID 1a2b:1a2b Device name`. (Or whichever way your system supplies to get the PID and VID.)
2. Create a udev rule to let users belonging to *dialout* group use the printer. You can create the file `/etc/udev/rules.d/99-escpos.rules` and add the following: `SUBSYSTEM=="usb", ATTRS{idVendor}=="1a2b", ATTRS{idProduct}=="1a2b", MODE="0664", GROUP="dialout"` Replace *idVendor* and *idProduct* hex numbers with the ones that you got from the previous step. Note that you can either, add yourself to "dialout" group, or use another group you already belongs instead "dialout" and set it in the `GROUP` parameter in the above rule.
3. Restart udev `# sudo service udev restart` In some new systems it is done with `# sudo udevadm control --reload`

Enabling tab-completion in CLI

python-escpos has a CLI with tab-completion. This is realized with `argcomplete`. In order for this to work you have to enable tab-completion, which is described in the manual of argcomplete.

If you only want to enable it for python-escpos, or global activation does not work, try this:

```
eval "$(register-python-argcomplete python-escpos)"
```

2.1.2 Methods

Last Reviewed
2023-08-10

Escpos class

The core part of the API of this library is the Escpos class. You use it by instantiating a *printer* which is a child of Escpos. The following methods are available:

class escpos.escpos.**Escpos**(*profile=None*, *magic_encode_args=None*, ***kwargs*)

ESC/POS Printer object.

This class is the abstract base class for an Esc/Pos-printer. The printer implementations are children of this class.

property device: Literal[None] | object

Implements a self-open mechanism.

An attempt to get the property before open the connection will cause the connection to open.

open()

Open a printer device/connection.

close()

Close a printer device/connection.

image(*img_source*, *high_density_vertical=True*, *high_density_horizontal=True*, *impl='bitImageRaster'*, *fragment_height=960*, *center=False*)

Print an image.

You can select whether the printer should print in high density or not. The default value is high density. When printing in low density, the image will be stretched.

Esc/Pos supplies several commands for printing. This function supports three of them. Please try to vary the implementations if you have any problems. For example the printer *IT80-002* will have trouble aligning images that are not printed in Column-mode.

The available printing implementations are:

- *bitImageRaster*: prints with the *GS v 0*-command
- *graphics*: prints with the *GS (L*-command
- *bitImageColumn*: prints with the *ESC* *-command

When trying to center an image make sure you have initialized the printer with a valid profile, that contains a media width pixel field. Otherwise the centering will have no effect.

Parameters

- **img_source** – PIL image or filename to load: *jpg*, *gif*, *png* or *bmp*
- **high_density_vertical** (bool) – print in high density in vertical direction *default:* True
- **high_density_horizontal** (bool) – print in high density in horizontal direction *default:* True
- **impl** (str) – choose image printing mode between *bitImageRaster*, *graphics* or *bitImageColumn*
- **fragment_height** (int) – Images larger than this will be split into multiple fragments *default:* 960
- **center** (bool) – Center image horizontally *default:* False

Return type

None

qr(*content*, *ec=0*, *size=3*, *model=2*, *native=False*, *center=False*, *impl=None*, *image_arguments=None*)

Print QR Code for the provided string.

Parameters

- **content** – The content of the code. Numeric data will be more efficiently compacted.
- **ec** – Error-correction level to use. One of QR_ECLEVEL_L (default), QR_ECLEVEL_M, QR_ECLEVEL_Q or QR_ECLEVEL_H. Higher error correction results in a less compact code.
- **size** – Pixel size to use. Must be 1-16 (default 3)
- **model** – QR code model to use. Must be one of QR_MODEL_1, QR_MODEL_2 (default) or QR_MICRO (not supported by all printers).

- **native** – True to render the code on the printer, False to render the code as an image and send it to the printer (Default)
- **center** – Centers the code *default:* False
- **impl** – Image-printing-implementation, refer to *image()* for details
- **image_arguments** (Optional[dict]) – arguments passed to *image()*. Replaces *impl* and *center*. If *impl* or *center* are set, they will overwrite *image_arguments*.

Return type
None

charcode(*code='AUTO'*)

Set Character Code Table.

Sets the control sequence from CHARCODE in *escpos.constants* as active. It will be sent with the next text sequence. If you set the variable code to AUTO it will try to automatically guess the right codepage. (This is the standard behavior.)

Parameters
code (str) – Name of CharCode

Raises
CharCodeError

Return type
None

static check_barcode(*bc*, *code*)

Check if barcode is OK.

This method checks if the barcode is in the proper format. The validation concerns the barcode length and the set of characters, but won't compute/validate any checksum. The full set of requirement for each barcode type is available in the ESC/POS documentation.

As an example, using EAN13, the barcode *12345678901* will be correct, because it can be rendered by the printer. But it does not suit the EAN13 standard, because the checksum digit is missing. Adding a wrong checksum in the end will also be considered correct, but adding a letter won't (EAN13 is numeric only).

Todo: Add a method to compute the checksum for the different standards

---

Todo: For fixed-length standards with mandatory checksum (EAN, UPC), compute and add the checksum automatically if missing.

Parameters

- **bc** (str) – barcode format, see *barcode()*
- **code** (str) – alphanumeric data to be printed as bar code, see *barcode()*

Returns
bool

barcode(*code*, *bc*, *height=64*, *width=3*, *pos='BELOW'*, *font='A'*, *align_ct=True*, *function_type=None*, *check=True*, *force_software=False*)

Print barcode.

Automatic hardwarelsoftware barcode renderer according to the printer capabilities.

Defaults to hardware barcode and its format types if supported. Automatically switches to software barcode renderer if hardware does not support a barcode type that is supported by software. (e.g. JAN, ISSN, etc.).

Set force_software=True to force the software renderer according to the profile. Set force_software=graphics|bitImageColumn|bitImageRaster to specify a renderer.

Ignores caps, special chars and whitespaces in barcode type names. So "EAN13", "ean-13", "Ean_13", "EAN 13" are all accepted.

Parameters

- **code** – alphanumeric data to be printed as bar code (payload).
- **bc** – barcode format type (EAN13, CODE128, JAN, etc.).
- **height** (int) – barcode module height (in printer dots), has to be between 1 and 255. *default*: 64
- **width** (int) – barcode module width (in printer dots), has to be between 2 and 6. *default*: 3
- **pos** (str) – text position (ABOVE, BELOW, BOTH, OFF) relative to the barcode (ignored in software renderer). *default*: BELOW
- **font** (str) – select font A or B (ignored in software renderer). *default*: A
- **align_ct** (bool) – If *True*, center the barcode. *default*: True
- **function_type** – ESCPOS function type A or B. None to guess it from profile (ignored in software renderer). *default*: None
- **check** (bool) – If *True*, checks that the code meets the requirements of the barcode type. *default*: True
- **force_software** (Union[bool, str]) – If *True*, force the use of software barcode renderer from profile. If *"graphics", "bitImageColumn" or "bitImageRaster"*, force the use of specific renderer.

Raises

BarcodeCodeError, *BarcodeTypeError*

Return type

None

Note:

Get all supported formats at:

- Hardware: *BARCODE_FORMATS*
- Software: Python barcode documentation

---

text(*txt*)

Print alpha-numeric text.

The text has to be encoded in the currently selected codepage. The input text has to be encoded in unicode.

Parameters

txt (str) – text to be printed

Raises

TextError

Return type
None

textln(*txt=''*)

Print alpha-numeric text with a newline.

The text has to be encoded in the currently selected codepage. The input text has to be encoded in unicode.

Parameters
txt (str) – text to be printed with a newline

Raises
TextError

Return type
None

ln(*count=1*)

Print a newline or more.

Parameters
count (int) – number of newlines to print

Raises
ValueError if count < 0

Return type
None

block_text(*txt*, *font='0'*, *columns=None*)

Print text wrapped to specific columns.

Text has to be encoded in unicode.

Parameters

- **txt** – text to be printed
- **font** – font to be used, can be a or b
- **columns** – amount of columns

Return type
None

Returns
None

set(*align=None*, *font=None*, *bold=None*, *underline=None*, *width=None*, *height=None*, *density=None*, *invert=None*, *smooth=None*, *flip=None*, *normal_textsize=None*, *double_width=None*, *double_height=None*, *custom_size=None*)

Set text properties by sending them to the printer.

If a value for a parameter is not supplied, nothing is sent for this type of format.

Parameters

- **align** (Optional[str]) – horizontal position for text, possible values are:
  - 'center'
  - 'left'
  - 'right'

- **font** (Optional[str]) – font given as an index, a name, or one of the special values 'a' or 'b', referring to fonts 0 and 1.
- **bold** (Optional[bool]) – text in bold
- **underline** (Optional[int]) – underline mode for text, decimal range 0-2
- **normal_textsize** (Optional[bool]) – switch to normal text size if True
- **double_height** (Optional[bool]) – doubles the height of the text
- **double_width** (Optional[bool]) – doubles the width of the text
- **custom_size** (Optional[bool]) – uses custom size specified by width and height parameters. Cannot be used with double_width or double_height.
- **width** (Optional[int]) – text width multiplier when custom_size is used, decimal range 1-8
- **height** (Optional[int]) – text height multiplier when custom_size is used, decimal range 1-8
- **density** (Optional[int]) – print density, value from 0-8, if something else is supplied the density remains unchanged
- **invert** (Optional[bool]) – True enables white on black printing
- **smooth** (Optional[bool]) – True enables text smoothing. Effective on 4x4 size text and larger
- **flip** (Optional[bool]) – True enables upside-down printing

Return type
None

set_with_default (*align='left'*, *font='a'*, *bold=False*, *underline=0*, *width=1*, *height=1*, *density=9*, *invert=False*, *smooth=False*, *flip=False*, *double_width=False*, *double_height=False*, *custom_size=False*)

Set default text properties by sending them to the printer.

This function has the behavior of the *set()*-method from before version 3. If a parameter to this method is not supplied, a default value will be sent. Otherwise this method forwards the values to the escpos.Escpos.set().

Parameters

- **align** (Optional[str]) – horizontal position for text, possible values are:
  - 'center'
  - 'left'
  - 'right'

  default: 'left'
- **font** (Optional[str]) – font given as an index, a name, or one of the special values 'a' or 'b', referring to fonts 0 and 1.
- **bold** (Optional[bool]) – text in bold, *default*: False
- **underline** (Optional[int]) – underline mode for text, decimal range 0-2, *default*: 0
- **double_height** (Optional[bool]) – doubles the height of the text
- **double_width** (Optional[bool]) – doubles the width of the text

- **custom_size** (Optional[bool]) – uses custom size specified by width and height parameters. Cannot be used with double_width or double_height.
- **width** (Optional[int]) – text width multiplier when custom_size is used, decimal range 1-8, *default*: 1
- **height** (Optional[int]) – text height multiplier when custom_size is used, decimal range 1-8, *default*: 1
- **density** (Optional[int]) – print density, value from 0-8, if something else is supplied the density remains unchanged
- **invert** (Optional[bool]) – True enables white on black printing, *default*: False
- **smooth** (Optional[bool]) – True enables text smoothing. Effective on 4x4 size text and larger, *default*: False
- **flip** (Optional[bool]) – True enables upside-down printing, *default*: False

Return type
None

line_spacing(*spacing=None*, *divisor=180*)

Set line character spacing.

If no spacing is given, we reset it to the default.

There are different commands for setting the line spacing, using a different denominator:

'+" line_spacing/360 of an inch, 0 <= line_spacing <= 255 '3' line_spacing/180 of an inch, 0 <= line_spacing <= 255 'A' line_spacing/60 of an inch, 0 <= line_spacing <= 85

Some printers may not support all of them. The most commonly available command (using a divisor of 180) is chosen.

Return type
None

cut(*mode='FULL'*, *feed=True*)

Cut paper.

Without any arguments the paper will be cut completely. With 'mode=PART' a partial cut will be attempted. Note however, that not all models can do a partial cut. See the documentation of your printer for details.

Parameters

- **mode** (str) – set to 'PART' for a partial cut. default: 'FULL'
- **feed** (bool) – print and feed before cutting. default: true

Raises
ValueError – if mode not in ('FULL', 'PART')

Return type
None

cashdraw(*pin*)

Send pulse to kick the cash drawer.

Kick cash drawer on pin 2 (*CD_KICK_2*) or pin 5 (*CD_KICK_5*) according to the default parameters. For non default parameters send a decimal sequence i.e. [27,112,48] or [27,112,0,25,255].

Parameters
pin – pin number, 2 or 5 or list of decimals

Raises
CashDrawerError

Return type
None

linedisplay_select(*select_display=False*)

Select the line display or the printer.

This method is used for line displays that are daisy-chained between your computer and printer. If you set *select_display* to true, only the display is selected and if you set it to false, only the printer is selected.

Parameters
select_display (bool) – whether the display should be selected or the printer

Return type
None

linedisplay_clear()

Clear the line display and resets the .

This method is used for line displays that are daisy-chained between your computer and printer.

Return type
None

linedisplay(*text*)

Display text on a line display connected to your printer.

You should connect a line display to your printer. You can do this by daisy-chaining the display between your computer and printer.

Parameters
text (str) – Text to display

Return type
None

hw(*hw*)

Hardware operations.

Parameters
hw (str) – hardware action, may be:

- INIT
- SELECT
- RESET

Return type
None

print_and_feed(*n=1*)

Print data in print buffer and feed *n* lines.

If n not in range (0, 255) then a ValueError will be raised.

Parameters
n (int) – number of n to feed. 0 <= n <= 255. default: 1

Raises
ValueError – if not 0 <= n <= 255

Return type
None

control(*ctl*, *count=5*, *tab_size=8*)

Feed control sequences.

Parameters

- **ctl** (str) – string for the following control sequences:
  - LF *for Line Feed*
  - FF *for Form Feed*
  - CR *for Carriage Return*
  - HT *for Horizontal Tab*
  - VT *for Vertical Tab*
- **count** (int) – integer between 1 and 32, controls the horizontal tab count. Defaults to 5.
- **tab_size** (int) – integer between 1 and 255, controls the horizontal tab size in characters. Defaults to 8

Raises
TabPosError

Return type
None

panel_buttons(*enable=True*)

Control the panel buttons on the printer (e.g. FEED).

When enable is set to False the panel buttons on the printer will be disabled. Calling the method with *enable=True* or without argument will enable the panel buttons.

If panel buttons are enabled, the function of the panel button, such as feeding, will be executed upon pressing the button. If the panel buttons are disabled, pressing them will not have any effect.

This command is effective until the printer is initialized, resetted or power-cycled. The default is enabled panel buttons.

Some panel buttons will always work, especially when the printer is opened. See for more information the manual of your printer and the escpos-command-reference.

Parameters
enable (bool) – controls the panel buttons

Return type
None

query_status(*mode*)

Query the printer for its status.

Returns byte array containing it.

Parameters
mode (bytes) – Integer that sets the status mode queried to the printer. - RT_STATUS_ONLINE: Printer status. - RT_STATUS_PAPER: Paper sensor.

Return type
bytes

is_online()

Query the online status of the printer.

Return type
`bool`

Returns
When online, returns `True`; `False` otherwise.

paper_status()

Query the paper status of the printer.

Returns 2 if there is plenty of paper, 1 if the paper has arrived to the near-end sensor and 0 if there is no paper.

Return type
`int`

Returns
2: Paper is adequate. 1: Paper ending. 0: No paper.

target(*type='ROLL'*)

Select where to print to.

Print to the thermal printer by default (ROLL) or print to the slip dot matrix printer if supported (SLIP)

Return type
`None`

eject_slip()

Eject the slip/cheque.

Return type
`None`

print_and_eject_slip()

Print and eject.

Prints data from the buffer to the slip station and if the paper sensor is covered, reverses the slip out the front of the printer far enough to be accessible to the operator. The impact station opens the platen in all cases.

Return type
`None`

use_slip_only()

Select the Slip Station for all functions.

The receipt station is the default setting after the printer is initialized or the Clear Printer (0x10) command is received

Return type
`None`

buzzer(*times=2*, *duration=4*)

Activate the internal printer buzzer on supported printers.

The 'times' parameter refers to the 'n' escpos command parameter, which means how many times the buzzer will be 'beeped'.

Parameters

- **times** (`int`) – Integer between 1 and 9, indicates the buzzer beeps.

- **duration** (int) – Integer between 1 and 9, indicates the beep duration.

Return type
None

Returns
None

2.1.3 Printers

Last Reviewed
2023-08-23

As of now there are 8 different types of printer implementations.

USB

The USB-class uses pyusb and libusb to communicate with USB-based printers.

Note: This driver is not suited for USB-to-Serial-adapters and similar devices, but only for those implementing native USB.

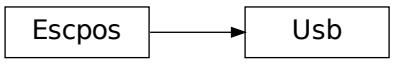
---

class escpos.printer.**Usb**(*idVendor=None*, *idProduct=None*, *usb_args={}*, *timeout=0*, *in_ep=130*, *out_ep=1*, **args*, ***kwargs*)

USB printer.

This class describes a printer that natively speaks USB.

inheritance:



static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

__init__(*idVendor=None*, *idProduct=None*, *usb_args={}*, *timeout=0*, *in_ep=130*, *out_ep=1*, **args*, ***kwargs*)

Initialize USB printer.

Parameters

- **idVendor** (Optional[int]) – Vendor ID
- **idProduct** (Optional[int]) – Product ID

- **usb_args** (Dict[str, Union[str, int]]) – Optional USB arguments (e.g. custom_match)
- **timeout** (Union[int, float]) – Is the time limit of the USB operation. Default without timeout.
- **in_ep** (int) – Input end point
- **out_ep** (int) – Output end point

open(*raise_not_found=True*)

Search device on USB tree and set it as escpos device.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Raises
USBNotFoundError

Return type
None

close()

Release USB interface.

Return type
None

Serial

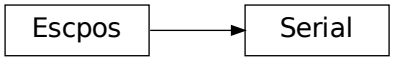
This driver uses pyserial in order to communicate with serial devices. If you are using an USB-based adapter to connect to the serial port, then you should also use this driver. The configuration is often based on DIP-switches that you can set on your printer. For the hardware-configuration please refer to your printer's manual.

class escpos.printer.**Serial**(*devfile=''*, *baudrate=9600*, *bytesize=8*, *timeout=1*, *parity=None*, *stopbits=None*, *xonxoff=False*, *dsrdtr=True*, **args*, ***kwargs*)

Serial printer.

This class describes a printer that is connected by serial interface.

inheritance:



static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

__init__(*devfile=''*, *baudrate=9600*, *bytesize=8*, *timeout=1*, *parity=None*, *stopbits=None*, *xonxoff=False*, *dsrdtr=True*, **args*, ***kwargs*)

Initialize serial printer.

Parameters

- **devfile** (str) – Device file under dev filesystem
- **baudrate** (int) – Baud rate for serial transmission
- **bytesize** (int) – Serial buffer size
- **timeout** (Union[int, float]) – Read/Write timeout
- **parity** (Optional[str]) – Parity checking
- **stopbits** (Optional[int]) – Number of stop bits
- **xonxoff** (bool) – Software flow control
- **dsrdtr** (bool) – Hardware flow control (False to enable RTS/CTS)

open(*raise_not_found=True*)

Set up serial port and set is as escpos device.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Return type
None

close()

Close Serial interface.

Return type
None

Network

This driver is based on the socket class.

class escpos.printer.**Network**(*host=''*, *port=9100*, *timeout=60*, **args*, ***kwargs*)

Network printer.

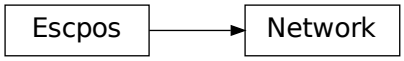
This class is used to attach to a networked printer. You can also use this in order to attach to a printer that is forwarded with socat.

If you have a local printer on parallel port /dev/usb/lp0 then you could start socat with:

```
socat -u TCP4-LISTEN:4242,reuseaddr,fork OPEN:/dev/usb/lp0
```

Then you should be able to attach to port 4242 with this class. Otherwise the normal use case would be to have a printer with Ethernet interface. This type of printer should work the same with this class. For the address of the printer check its manuals.

inheritance:



static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

__init__(*host=''*, *port=9100*, *timeout=60*, **args*, ***kwargs*)

Initialize network printer.

Parameters

- **host** (str) – Printer's host name or IP address
- **port** (int) – Port to write to
- **timeout** (Union[int, float]) – timeout in seconds for the socket-library

open(*raise_not_found=True*)

Open TCP socket with socket-library and set it as escpos device.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Return type
None

close()

Close TCP connection.

Return type
None

Troubleshooting

Problems with a network-attached printer can have numerous causes. Make sure that your device has a proper IP address. Often you can check the IP address by triggering the self-test of the device. As a next step try to send text manually to the device. You could use for example:

```
echo "OK\n" | nc IPADDRESS 9100
# the port number is often 9100
```

As a last resort try to reset the interface of the printer. This should be described in its manual.

File

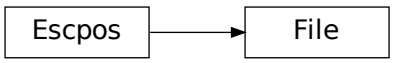
This printer "prints" just into a file-handle. Especially on *nix-systems this comes very handy.

class escpos.printer.**File**(*devfile=''*, *auto_flush=True*, **args*, ***kwargs*)

Generic file printer.

This class is used for parallel port printer or other printers that are directly attached to the filesystem. Note that you should stay away from using USB-to-Parallel-Adapter since they are unreliable and produce arbitrary errors.

inheritance:



static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

__init__(*devfile=''*, *auto_flush=True*, **args*, ***kwargs*)

Initialize file printer with device file.

Parameters

- **devfile** (str) – Device file under dev filesystem
- **auto_flush** (bool) – automatically call flush after every call of _raw()

open(*raise_not_found=True*)

Open system file.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Return type
None

flush()

Flush printing content.

Return type
None

close()

Close system file.

Return type
None

Dummy

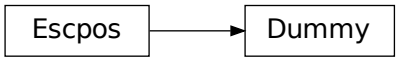
The Dummy-printer is mainly for testing- and debugging-purposes. It stores all of the "output" as raw ESC/POS in a string and returns that.

class escpos.printer.**Dummy**(**args*, ***kwargs*)

Dummy printer.

This class is used for saving commands to a variable, for use in situations where there is no need to send commands to an actual printer. This includes generating print jobs for later use, or testing output.

inheritance:



static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

property output: bytes

Get the data that was sent to this printer.

clear()

Clear the buffer of the printer.

This method can be called if you send the contents to a physical printer and want to use the Dummy printer for new output.

Return type
None

close()

Close not implemented for Dummy printer.

Return type
None

CUPS

This driver uses *pycups* in order to communicate with a CUPS server. Supports both local and remote CUPS printers and servers. The printer must be properly configured in CUPS administration. The connector generates a print job that is added to the CUPS queue.

class escpos.printer.**CupsPrinter**(*printer_name=''*, **args*, ***kwargs*)

Simple CUPS printer connector.

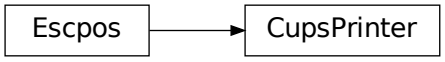
Note:

Requires `pycups` which in turn needs the cups development library package:

- Ubuntu/Debian: `libcups2-dev`
- OpenSuse/Fedora: `cups-devel`

---

inheritance:



static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
`bool`

property printers: dict

Available CUPS printers.

open(*job_name='python-escpos'*, *raise_not_found=True*)

Set up a new print job and target the printer.

A call to this method is required to send new jobs to the CUPS connection after close.

Defaults to default CUPS printer. Creates a new temporary file buffer.

By default raise an exception if device is not found.

Parameters
raise_not_found (`bool`) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Return type
`None`

send()

Send the print job to the printer.

Return type
None

close()

Close CUPS connection.

Send pending job to the printer if needed.

Return type
None

LP

This driver uses the UNIX command *lp* in order to communicate with a CUPS server. Supports local and remote CUPS printers. The printer must be properly configured in CUPS administration. The connector spawns a new sub-process where the command lp is executed.

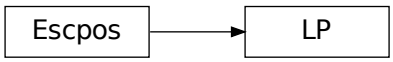
No dependencies required, but somehow the print queue will affect some print job such as barcode.

class escpos.printer.**LP**(*printer_name=''*, **args*, ***kwargs*)

Simple UNIX lp command raw printing.

Thanks to Oyami-Srk comment.

inheritance:



static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

__init__(*printer_name=''*, **args*, ***kwargs*)

LP class constructor.

Parameters

- **printer_name** (str) – CUPS printer name (Optional)
- **auto_flush** (*bool (Defaults False)*) – Automatic flush after every _raw() (Optional)

property printers: dict

Available CUPS printers.

open(*job_name='python-escpos'*, *raise_not_found=True*, *_close_opened=True*)

Invoke _lp_ in a new subprocess and wait for commands.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Return type
None

close()
Stop the subprocess.

Return type
None

flush()
End line and wait for new commands.

Return type
None

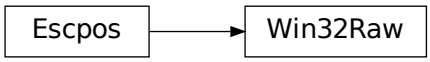
Win32Raw

This driver uses a native WIN32 interface of Windows in order to print. Please refer to the code for documentation as this driver is currently not included in the documentation build.

class escpos.printer.**Win32Raw**(*printer_name=''*, **args*, ***kwargs*)
Printer binding for win32 API.

Uses the module pywin32 for printing.

inheritance:



static is_usable()
Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

__init__(*printer_name=''*, **args*, ***kwargs*)
Initialize default printer.

property printers: dict
Available Windows printers.

open(*job_name='python-escpos'*, *raise_not_found=True*)
Open connection to default printer.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Return type
None

close()

Close connection to default printer.

Return type
None

2.1.4 Raspberry Pi

Last Reviewed
2023-08-10

Warning: You should **never** directly connect an printer with RS232-interface (serial port) directly to a Raspberry PI or similar interface (e.g. those simple USB-sticks without encasing). Those interfaces are based on 5V- or 3,3V-logic (the latter in the case of Raspberry PI). Classical RS232 uses 12V-logic and would **thus destroy your interface**. Connect both systems with an appropriate *level shifter*.

Installation

The installation should be performed as described in *Installation*.

Run

You can run this software as on any other Linux system.

Attach your printer and test it with the example code in the project's set of examples. You can find that in the project-repository.

For more details on this check the *installation-manual*.

2.1.5 Usage

Last Reviewed
2023-08-10

Define your printer

USB printer

Before creating your Python ESC/POS printer instance, consult the system to obtain the printer parameters. This is done with the 'lsusb' command.

Run the command and look for the "Vendor ID" and "Product ID" and write down the values. These values are displayed just before the name of the device with the following format:

```
xxxx:xxxx
```

Example:

```
# lsusb
Bus 002 Device 001: ID 04b8:0202 Epson ...
```

Write down the the values in question, then issue the following command so you can get the "Interface" number and "End Point"

```
# lsusb -vvv -d xxxx:xxxx | grep iInterface
    iInterface              0
# lsusb -vvv -d xxxx:xxxx | grep bEndpointAddress | grep OUT
      bEndpointAddress     0x01  EP 1 OUT
```

The first command will yield the "Interface" number that must be handy to have and the second yields the "Output Endpoint" address.

USB Printer initialization

```
p = printer.Usb(0x04b8,0x0202)
```

By default the "Interface" number is "0" and the "Output Endpoint" address is "0x01". If you have other values then you can define them on your instance. So, assuming that we have another printer, CT-S2000, manufactured by Citizen (with "Vendor ID" of 2730 and "Product ID" of 0fff) where in_ep is on 0x81 and out_ep=0x02, then the printer definition should look like:

Generic USB Printer initialization

```
p = printer.Usb(0x2730, 0x0fff, 0, 0x81, 0x02)
```

Network printer

You only need the IP of your printer, either because it is getting its IP by DHCP or you set it manually.

Network Printer initialization

```
p = printer.Network("192.168.1.99")
```

Serial printer

Most of the default values set by the DIP switches for the serial printers, have been set as default on the serial printer class, so the only thing you need to know is which serial port the printer is connected to.

Serial printer initialization

```
p = printer.Serial("/dev/tty0")

# on a Windows OS serial devices are typically accessible as COM
p = printer.Serial("COM1")
```

Other printers

Some printers under */dev* can't be used or initialized with any of the methods described above. Usually, those are printers used by printcap, however, if you know the device name, you could try to initialize by passing the device node name.

```
p = printer.File("/dev/usb/lp1")
```

The default is "/dev/usb/lp0", so if the printer is located on that node, then you don't necessary need to pass the node name.

Define your instance

The following example demonstrates how to initialize the Epson TM-TI88IV on a USB interface.

```
from escpos import *
""" Seiko Epson Corp. Receipt Printer M129 Definitions (EPSON TM-T88IV) """
p = printer.Usb(0x04b8,0x0202)
# Print text
p.text("Hello World\n")
# Print image
p.image("logo.gif")
# Print QR Code
p.qr("You can readme from your smartphone")
# Print barcode
p.barcode('4006381333931','EAN13',64,2,'','')
# Cut paper
p.cut()
```

Standard python constraints on libraries apply. This means especially that you should not name the script in which you implement these lines should not be named `escpos` as this would collide with the name of the library.

Configuration File

You can create a configuration file for python-escpos. This will allow you to use the CLI, and skip some setup when using the library programmatically.

The default configuration file is named `config.yaml` and uses the YAML format. For windows it is probably at:

```
%appdata%/python-escpos/config.yaml
```

And for linux:

```
$HOME/.config/python-escpos/config.yaml
```

If you are not sure, run:

```
from escpos import config
c = config.Config()
c.load()
```

If it can't find the configuration file in the default location, it will tell you where it's looking. You can always pass a path, or a list of paths, to the `load()` method.

To load the configured printer, run:

```
from escpos import config
c = config.Config()
printer = c.printer()
```

The printer section

The `printer` configuration section defines a default printer to create.

The only required parameter is `type`. The value of this has to be one of the printers defined in *Printers*.

The rest of the given parameters will be passed on to the initialization of the printer class. Use these to overwrite the default values as specified in *Printers*. This implies that the parameters have to match the parameter-names of the respective printer class.

An example file printer:

```
printer:
        type: File
        devfile: /dev/someprinter
```

And for a network printer:

```
printer:
        type: Network
        host: 127.0.0.1
        port: 9000
```

An USB-printer could be defined by:

```
printer:
        type: Usb
        idVendor: 0x1234
        idProduct: 0x5678
        in_ep: 0x66
        out_ep: 0x01
```

Printing text right

Python-escpos is designed to accept unicode.

For normal usage you can simply pass your text to the printers `text()`-function. It will automatically guess the right codepage and then send the encoded data to the printer. If this feature does not work, please try to isolate the error and then create an issue on the GitHub project page.

If you want or need to you can manually set the codepage. For this please use the `charcode()`-function. You can set any key-value that is in `CHARCODE`. If something is wrong, an `CharCodeError` will be raised. After you have manually set the codepage the printer won't change it anymore. You can revert to normal behavior by setting charcode to `AUTO`.

Advanced Usage: Print from binary blob

Imagine you have a file with ESC/POS-commands in binary form. This could be useful for testing capabilities of your printer with a known working combination of commands. You can print this data with the following code, using the standard methods of python-escpos. (This is an advantage of the fact that *_raw()* accepts binary strings.)

```
from escpos import printer
p = printer.Serial()  # adapt this to your printer model

file = open("binary-blob.bin", "rb")  # read in the file containing your commands in␣
↪binary-mode
data = file.read()
file.close()

p._raw(data)
```

That's all, the printer should then print your data. You can also use this technique to let others reproduce an issue that you have found. (Just "print" your commands to a File-printer on your local filesystem.) However, please keep in mind, that often it is easier and better to just supply the code that you are using.

Here you can download an example, that will print a set of common barcodes:

- `barcode.bin` by @mike42

Advanced Usage: change capabilities-profile

Packaged together with the escpos-code is a capabilities-file. This file in JSON-format describes the capabilities of different printers. It is developed and hosted in escpos-printer-db.

Certain applications like the usage of cx_freeze might change the packaging structure. This leads to the capabilities-profile not being found. In this case you can use the environment-variable *ESCPOS_CAPABILITIES_FILE*. The following code is an example.

```
# use packaged capabilities-profile
python-escpos cut

# use capabilities-profile that you have put in /usr/python-escpos
export ESCPOS_CAPABILITIES_FILE=/usr/python-escpos/capabilities.json
python-escpos cut

# use packaged file again
unset ESCPOS_CAPABILITIES_FILE
python-escpos cut
```

Hint: preprocess printing

Printing images directly to the printer is rather slow. One factor that slows down the process is the transmission over e.g. serial port.

Apart from configuring your printer to use the maximum baudrate (in the case of serial-printers), there is not much that you can do. However you could use the *escpos.printer.Dummy*-printer to preprocess your image. This is probably best explained by an example:

```
from escpos.printer import Serial, Dummy

p = Serial()
d = Dummy()

# create ESC/POS for the print job, this should go really fast
d.text("This is my image:\n")
d.image("funny_cat.png")
d.cut()

# send code to printer
p._raw(d.output)
```

This way you could also store the code in a file and print it later. You could then for example print the code from another process than your main-program and thus reduce the waiting time. (Of course this will not make the printer print faster.)

Troubleshooting

This section gathers various hints on troubleshooting.

Print with STAR TSP100 family

Printer of the STAR TSP100 family do not have a native ESC/POS mode, which is why you will not be able to directly print with this library to the printer.

More information on this topic can be found in the online documentation of Star Micronics and the discussion in the python-escpos project.

2.1.6 CLI

Last Reviewed
2023-09-25

Documentation of the command line interface, callable with `python-escpos`.

CLI for python-escpos

```
usage: python-escpos [-h] [-c CONFIG]
                     {qr,barcode,text,block_text,cut,cashdraw,image,fullimage,
↪charcode,set,hw,control,panel_buttons,raw,demo,version,version_extended}
                     ...
```

Named Arguments

-c, --config Alternate path to the configuration file

ESCPOS Command

parser Possible choices: qr, barcode, text, block_text, cut, cashdraw, image, fullimage, charcode, set, hw, control, panel_buttons, raw, demo, version, version_extended

Sub-commands

qr

Print a QR code

```
python-escpos qr [-h] --content CONTENT [--size SIZE]
```

Named Arguments

--content Text to print as a qr code

--size QR code size (1-16) [default:3]

barcode

Print a barcode

```
python-escpos barcode [-h] --code CODE --bc BC [--height HEIGHT]
                      [--width WIDTH] [--pos {BELOW,ABOVE,BOTH,OFF}]
                      [--font {A,B}] [--align_ct ALIGN_CT]
                      [--function_type {A,B}]
                      [--force_software {graphics,bitImageColumn,bitImageRaster}]
```

Named Arguments

--code Barcode data to print

--bc Barcode format

--height Barcode height in px

--width Barcode width

--pos Possible choices: BELOW, ABOVE, BOTH, OFF

Label position

--font Possible choices: A, B

Label font

--align_ct Align barcode center

--function_type	Possible choices: A, B ESCPOS function type
--force_software	Possible choices: graphics, bitImageColumn, bitImageRaster Force render and print barcode as an image

text

Print plain text

```
python-escpos text [-h] --txt TXT
```

Named Arguments

--txt	Plain text to print

block_text

Print wrapped text

```
python-escpos block_text [-h] --txt TXT [--columns COLUMNS]
```

Named Arguments

--txt	block_text to print
--columns	Number of columns

cut

Cut the paper

```
python-escpos cut [-h] [--mode {FULL,PART}]
```

Named Arguments

--mode	Possible choices: FULL, PART Type of cut

cashdraw

Kick the cash drawer

```
python-escpos cashdraw [-h] [--pin {2,5}]
```

Named Arguments

--pin Possible choices: 2, 5

Which PIN to kick

image

Print an image

```
python-escpos image [-h] --img_source IMG_SOURCE
                    [--impl {bitImageRaster,bitImageColumn,graphics}]
                    [--high_density_horizontal HIGH_DENSITY_HORIZONTAL]
                    [--high_density_vertical HIGH_DENSITY_VERTICAL]
```

Named Arguments

--img_source Path to image

--impl Possible choices: bitImageRaster, bitImageColumn, graphics

Implementation to use

--high_density_horizontal Image density (horizontal)

--high_density_vertical Image density (vertical)

fullimage

Print a fullimage

```
python-escpos fullimage [-h] --img IMG [--max_height MAX_HEIGHT]
                        [--width WIDTH] [--histeq HISTEQ]
                        [--bandsize BANDSIZE]
```

Named Arguments

--img Path to img

--max_height Max height of image in px

--width Max width of image in px

--histeq Equalize the histogram

--bandsize Size of bands to divide into when printing

charcode

Set character code table

```
python-escpos charcode [-h] --code CODE
```

Named Arguments

--code Character code

set

Set text properties

```
python-escpos set [-h] [--align {left,center,right}]
                  [--font {left,center,right}]
                  [--text_type {B,U,U2,BU,BU2,NORMAL}] [--width WIDTH]
                  [--height HEIGHT] [--density DENSITY] [--invert INVERT]
                  [--smooth SMOOTH] [--flip FLIP]
```

Named Arguments

--align Possible choices: left, center, right

Horizontal alignment

--font Possible choices: left, center, right

Font choice

--text_type Possible choices: B, U, U2, BU, BU2, NORMAL

Text properties

--width Width multiplier

--height Height multiplier

--density Print density

--invert White on black printing

--smooth Text smoothing. Effective on >: 4x4 text

--flip Text smoothing. Effective on >: 4x4 text

hw

Hardware operations

```
python-escpos hw [-h] --hw {INIT,SELECT,RESET}
```

Named Arguments

--hw Possible choices: INIT, SELECT, RESET

Operation

control

Control sequences

```
python-escpos control [-h] --ctl {LF,FF,CR,HT,VT} [--pos POS]
```

Named Arguments

--ctl Possible choices: LF, FF, CR, HT, VT

Control sequence

--pos Horizontal tab position (1-4)

panel_buttons

Controls panel buttons

```
python-escpos panel_buttons [-h] --enable ENABLE
```

Named Arguments

--enable Feed button enabled

raw

Raw data

```
python-escpos raw [-h] --msg MSG
```

Named Arguments

--msg Raw data to send

demo

Demonstrates various functions

```
python-escpos demo [-h] [--barcodes-a | --barcodes-b | --qr | --text]
```

Named Arguments

--barcodes-a Print demo barcodes for function type A

Default: False

--barcodes-b Print demo barcodes for function type B

Default: False

--qr Print some demo QR codes

Default: False

--text Print some demo text

Default: False

version

Print the version information of python-escpos

```
python-escpos version [-h]
```

version_extended

Print the extended version information of python-escpos (for bug reports)

```
python-escpos version_extended [-h]
```

Printer configuration is defined in the python-escpos configfile. See documentation for details.

2.1.7 Printing Barcodes

Last Reviewed
2023-08-10

Many printers implement barcode printing natively. These hardware rendered barcodes are fast but the supported formats are limited by the printer itself and different between models. However, almost all printers support printing images, so barcode rendering can be performed externally by software and then sent to the printer as an image. As a drawback, this operation is much slower and the user needs to know and choose the image implementation method supported by the printer's command-set.

barcode-method

Since version 3.0, the `barcode` method unifies the previous `barcode` (hardware) and `soft_barcode` (software) methods. It is able to choose automatically the best printer implementation for barcode printing based on the capabilities of the printer and the type of barcode desired. To achieve this, it relies on the information contained in the escpos-printer-db profiles. The chosen profile needs to match the capabilities of the printer as closely as possible.

`Escpos.`**`barcode`**(*code*, *bc*, *height=64*, *width=3*, *pos='BELOW'*, *font='A'*, *align_ct=True*, *function_type=None*, *check=True*, *force_software=False*)

Print barcode.

Automatic hardware|software barcode renderer according to the printer capabilities.

Defaults to hardware barcode and its format types if supported. Automatically switches to software barcode renderer if hardware does not support a barcode type that is supported by software. (e.g. JAN, ISSN, etc.).

Set force_software=True to force the software renderer according to the profile. Set force_software=graphics|bitImageColumn|bitImageRaster to specify a renderer.

Ignores caps, special chars and whitespaces in barcode type names. So "EAN13", "ean-13", "Ean_13", "EAN 13" are all accepted.

Parameters

- **`code`** – alphanumeric data to be printed as bar code (payload).
- **`bc`** – barcode format type (EAN13, CODE128, JAN, etc.).
- **`height`** (`int`) – barcode module height (in printer dots), has to be between 1 and 255. *default*: 64
- **`width`** (`int`) – barcode module width (in printer dots), has to be between 2 and 6. *default*: 3
- **`pos`** (`str`) – text position (ABOVE, BELOW, BOTH, OFF) relative to the barcode (ignored in software renderer). *default*: BELOW
- **`font`** (`str`) – select font A or B (ignored in software renderer). *default*: A
- **`align_ct`** (`bool`) – If *True*, center the barcode. *default*: True
- **`function_type`** – ESCPOS function type A or B. None to guess it from profile (ignored in software renderer). *default*: None
- **`check`** (`bool`) – If *True*, checks that the code meets the requirements of the barcode type. *default*: True
- **`force_software`** (`Union[bool, str]`) – If *True*, force the use of software barcode renderer from profile. If *"graphics", "bitImageColumn" or "bitImageRaster"*, force the use of specific renderer.

Raises
BarcodeCodeError, *BarcodeTypeError*

Return type
None

Note:

Get all supported formats at:

- Hardware: *BARCODE_FORMATS*
- Software: Python barcode documentation

CODE128

Code128 barcodes need a certain format. For now the user has to make sure that the payload is correct. For alphanumeric CODE128 you have to preface your payload with *{B*.

```
from escpos.printer import Dummy, Serial
p = Serial()
# print CODE128 012ABCDabcd
p.barcode("{B012ABCDabcd", "CODE128", function_type="B")
```

A very good description on CODE128 is also on Wikipedia.

2.2 Printer profiles

This chapter gives a listing of the available printer profiles. Details are described in *Capabilities*.

2.2.1 Capabilities

Last Reviewed
2023-08-10

Since the used command set often differs between printers, a model for supporting different printers is implemented. This feature is called *capabilities*.

The *capabilities*-feature allows this library to know which features are supported. If no further information is specified, python-escpos will try to automatically use features based on the supplied information.

In order to use the *capabilities*-database, the printer instance simply has to be created with the parameter *profile* set to the relevant identifier. The identifier can be found in *Available Profiles*.

This documentation describes the profiles in the database file that is bundled with this release. If another configuration is to be used, this chapter can be followed for information on how to side-load another *capabilities*-database: *Advanced Usage: change capabilities-profile*.

2.2.2 Available Profiles

Last Reviewed
2023-08-10

The following list describes which printer profiles are available in this release. The existence of a profile is a hint, but no guarantee that this printer actually can be controlled by this library.

If you find any issues with the described capabilities, please open an issue in the ESC/POS printer database. The data shown here is directly taken from there.

AF-240 Customer Display

This is a two-line, ESC/POS-aware customer display from Oxhoo. The ESC/POS command mode can be activated persistently by sending:

```
echo -ne "\n\x02\x05\x43\x31\x03" > /dev/ttyUSB0
```

You can select this profile in python-escpos with this identifier: `AF-240`. (Set parameter to *profile='AF-240'*.)

Basic information

Name	AF-240 Customer Display
Vendor	Oxhoo
Media width (mm)	120
Media width (pixels)	100
DPI	Unknown

Fonts

ID	Name	Columns
0	Font A	20

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	False
barcodeB	False
bitImageColumn	False
bitImageRaster	False
graphics	False
highDensity	False
paperFullCut	False
paperPartCut	False
pdf417Code	False
pulseBel	False
pulseStandard	False
qrCode	False
starCommands	False

Text code pages

ID	Encoding
0	*Oxhoo-specific European*

CT-S651

Citizen CT-S651 profile. This is a two-color thermal printer, supporting paper sizes from 58mm up to 83mm

You can select this profile in python-escpos with this identifier: `CT-S651`. (Set parameter to *profile='CT-S651'*.)

Basic information

Name	CT-S651
Vendor	Citizen
Media width (mm)	80
Media width (pixels)	640
DPI	203

Fonts

ID	Name	Columns
0	Font A	48
1	Font B	64
2	Font C	72

Colors

ID	Color
0	black
1	red

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	True
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*CP852*
7	*CP866*
8	*CP857*
9	*CP1252*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
20	*Unknown*
21	*Unknown*
25	*Unknown*
26	*Unknown*
30	*Vietnamese TCVN-3 1*
31	*Vietnamese TCVN-3 1*
40	*CP864*
255	*Unknown*

KR-306

Kefar KR-306 printer with 200mm/s speed

You can select this profile in python-escpos with this identifier: `KR-306`. (Set parameter to *profile='KR-306'*.)

Basic information

Name	KR-306
Vendor	Kefar
Media width (mm)	72
Media width (pixels)	576
DPI	Unknown

Fonts

ID	Name	Columns
0	Font A	48
1	Font B	64

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	False
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*Katakana (codepage 1)*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*Unknown*
7	*Unknown*
8	*Unknown*
9	*Unknown*
10	*Unknown*

continues on next page

Table 1 – continued from previous page

ID	Encoding
11	*CP1252*
12	*CP866*
13	*CP852*
14	*CP858*
15	*Unknown*
16	*Unknown*
17	*Unknown*
18	*Unknown*
19	*CP747*
20	*Unknown*
21	*Unknown*
22	*Unknown*
23	*CP864*
24	*Unknown*
25	*Unknown*
26	*Unknown*
27	*CP1255*
28	*CP437*
29	*Katakana (codepage 1)*
30	*CP437*
31	*CP858*
32	*CP852*
33	*CP860*
34	*CP861*
35	*CP863*
36	*CP865*
37	*CP866*
38	*CP855*
39	*CP857*
40	*CP862*
41	*CP864*
42	*CP737*
43	*Greek CP851*
44	*CP869*
45	*CP928*
46	*CP772*
47	*CP774*
48	*CP874*
49	*CP1252*
50	*CP1250*
51	*CP1251*
52	*Unimplemented Star-specific CP3840*
53	*Unimplemented Star-specific CP3841*
54	*Unimplemented Star-specific CP3843*
55	*Unimplemented Star-specific CP3844*
56	*Unimplemented Star-specific CP3845*
57	*Unimplemented Star-specific CP3846*
58	*Unimplemented Star-specific CP3847*
59	*Unimplemented Star-specific CP3848*
60	*Unimplemented Star-specific CP1001*

continues on next page

Table 1 – continued from previous page

ID	Encoding
61	*Unimplemented Star-specific CP2001*
62	*Unimplemented Star-specific CP3001*
63	*Unimplemented Star-specific CP3002*
64	*CP3011 Latvian*
65	*CP3012 Cyrillic*
66	*Unimplemented Star-specific CP3021*
67	*Unimplemented Star-specific CP3041*
68	*CP852*
69	*Unknown*
70	*CP1256*

NT-5890K

You can select this profile in python-escpos with this identifier: `NT-5890K`. (Set parameter to *profile='NT-5890K'*.)

Basic information

Name	NT-5890K
Vendor	Netum
Media width (mm)	57.5
Media width (pixels)	384
DPI	203

Fonts

ID	Name	Columns
0	Font A	32
1	Font B	42

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	False
barcodeB	False
bitImageColumn	True
bitImageRaster	True
graphics	False
highDensity	True
paperFullCut	False
paperPartCut	False
pdf417Code	False
pulseBel	False
pulseStandard	True
qrCode	False
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*Unknown*
7	*Unknown*
8	*Unknown*
9	*Unknown*
10	*Unknown*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
20	*Unknown*
21	*Unknown*
22	*Unknown*
23	*Unknown*
24	*CP747*
25	*CP1257*
27	*CP1258*
28	*CP864*
31	*Unknown*
32	*CP1255*
50	*CP437*
52	*CP437*
53	*CP858*

continues on next page

Table 2 – continued from previous page

ID	Encoding
54	*CP852*
55	*CP860*
56	*CP861*
57	*CP863*
58	*CP865*
59	*CP866*
60	*CP855*
61	*CP857*
62	*CP862*
63	*CP864*
64	*CP737*
65	*Greek CP851*
66	*CP869*
68	*CP772*
69	*CP774*
71	*CP1252*
72	*CP1250*
73	*CP1251*
74	*Unimplemented Star-specific CP3840*
76	*Unimplemented Star-specific CP3843*
77	*Unimplemented Star-specific CP3844*
78	*Unimplemented Star-specific CP3845*
79	*Unimplemented Star-specific CP3846*
80	*Unimplemented Star-specific CP3847*
81	*Unimplemented Star-specific CP3848*
83	*Unimplemented Star-specific CP2001*
84	*Unimplemented Star-specific CP3001*
85	*Unimplemented Star-specific CP3002*
86	*CP3011 Latvian*
87	*CP3012 Cyrillic*
88	*Unimplemented Star-specific CP3021*
89	*Unimplemented Star-specific CP3041*
90	*CP1253*
91	*CP1254*
92	*CP1256*
93	*CP720*
94	*CP1258*
95	*CP775*
96	*Unknown*
255	*Unknown*

80-V-UL

Netum 80-V-UL thermal printer series.

You can select this profile in python-escpos with this identifier: `NT-80-V-UL`. (Set parameter to *profile='NT-80-V-UL'*.)

Basic information

Name	80-V-UL
Vendor	Netum
Media width (mm)	80
Media width (pixels)	576
DPI	203

Fonts

ID	Name	Columns
0	Font A	12
1	Font B	9

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*Unknown*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*Unknown*
7	*Unknown*
8	*Unknown*
9	*Unknown*
10	*Unknown*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
20	*Unknown*
21	*Unknown*
22	*ISO_8859-6*
23	*Unknown*
24	*CP747*
25	*CP1257*
27	*Unknown*
28	*CP864*
31	*Unknown*
32	*CP1255*
50	*CP437*
52	*CP437*
53	*CP858*
54	*CP852*
55	*CP860*
56	*CP861*
57	*CP863*
58	*CP865*
59	*CP866*
60	*CP855*
61	*CP857*
62	*CP862*
63	*CP864*
64	*CP737*
65	*Greek CP851*
66	*CP869*
68	*CP772*
69	*CP774*
71	*CP1252*
72	*CP1250*
73	*CP1251*
74	*Unimplemented Star-specific CP3840*
76	*Unimplemented Star-specific CP3843*

continues on next page

Table 3 – continued from previous page

ID	Encoding
77	*Unimplemented Star-specific CP3844*
78	*Unimplemented Star-specific CP3845*
79	*Unimplemented Star-specific CP3846*
80	*Unimplemented Star-specific CP3847*
81	*Unimplemented Star-specific CP3848*
83	*Unimplemented Star-specific CP2001*
84	*Unimplemented Star-specific CP3001*
85	*Unimplemented Star-specific CP3002*
86	*CP3011 Latvian*
87	*CP3012 Cyrillic*
88	*Unimplemented Star-specific CP3021*
89	*Unimplemented Star-specific CP3041*
90	*CP1253*
91	*CP1254*
92	*CP1256*
93	*CP720*
94	*CP1258*
95	*CP775*
96	*Unknown*
255	*Unknown*

OCD-100 Customer Display

This is a two-line, ESC/POS-aware customer display from Aures. It has some graphics support via custom fonts, but is otherwise text-only. This profile is also suitable for the OCD-150 pole-mounted display.

You can select this profile in python-escpos with this identifier: `OCD-100`. (Set parameter to *profile='OCD-100'*.)

Basic information

Name	OCD-100 Customer Display
Vendor	Aures
Media width (mm)	180
Media width (pixels)	100
DPI	Unknown

Fonts

ID	Name	Columns
0	Font A	20

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	False
barcodeB	False
bitImageColumn	False
bitImageRaster	False
graphics	False
highDensity	False
paperFullCut	False
paperPartCut	False
pdf417Code	False
pulseBel	False
pulseStandard	False
qrCode	False
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*Unknown*
7	*Unknown*
8	*Unknown*
9	*CP852*
10	*CP862*
11	*CP866*
12	*CP1251*
13	*CP1254*
14	*CP1255*
15	*CP1257*
16	*CP1252*
17	*CP1253*
19	*CP858*

OCD-300 Customer Display

This is a two-line, ESC/POS-aware customer display from Aures. It has some graphics support via vendor-provided tools, but is otherwise text-only.

You can select this profile in python-escpos with this identifier: `OCD-300`. (Set parameter to *profile='OCD-300'*.)

Basic information

Name	OCD-300 Customer Display
Vendor	Aures
Media width (mm)	130.2
Media width (pixels)	240
DPI	Unknown

Fonts

ID	Name	Columns
0	Font A	20

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	False
barcodeB	False
bitImageColumn	False
bitImageRaster	False
graphics	False
highDensity	False
paperFullCut	False
paperPartCut	False
pdf417Code	False
pulseBel	False
pulseStandard	False
qrCode	False
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*Unknown*
7	*Unknown*
8	*Unknown*
9	*CP852*
10	*CP862*
11	*CP866*
12	*CP1251*
13	*CP1254*
14	*CP1255*
15	*CP1257*
16	*CP1252*
17	*CP1253*
18	*CP1250*
19	*CP858*
20	*Unknown*

P822D

You can select this profile in python-escpos with this identifier: `P822D`. (Set parameter to *profile='P822D'*.)

Basic information

Name	P822D
Vendor	PBM
Media width (mm)	79.5
Media width (pixels)	576
DPI	Unknown

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	False
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*Katakana (codepage 1)*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*Unknown*
7	*Unknown*
8	*Unknown*
9	*Unknown*
10	*Unknown*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
20	*Unknown*
21	*Unknown*
22	*Unknown*
23	*Unknown*
24	*CP747*

continues on next page

Table 4 – continued from previous page

ID	Encoding
25	*CP1257*
27	*Unknown*
28	*CP864*
29	*Unimplemented Star-specific CP1001*
30	*Unknown*
31	*Unknown*
32	*CP1255*
33	*CP720*
34	*CP1256*
35	*CP1257*
50	*CP437*
51	*Unknown*
52	*CP437*
53	*CP858*
54	*CP852*
55	*CP860*
56	*CP861*
57	*CP863*
58	*CP865*
59	*CP866*
60	*CP855*
61	*CP857*
62	*CP862*
63	*CP864*
64	*CP737*
65	*Greek CP851*
66	*CP869*
67	*CP928*
68	*CP772*
69	*CP774*
70	*CP874*
71	*CP1252*
72	*CP1250*
73	*CP1251*
74	*Unimplemented Star-specific CP3840*
75	*Unimplemented Star-specific CP3841*
76	*Unimplemented Star-specific CP3843*
77	*Unimplemented Star-specific CP3844*
78	*Unimplemented Star-specific CP3845*
79	*Unimplemented Star-specific CP3846*
80	*Unimplemented Star-specific CP3847*
81	*Unimplemented Star-specific CP3848*
82	*Unimplemented Star-specific CP1001*
83	*Unimplemented Star-specific CP2001*
84	*Unimplemented Star-specific CP3001*
85	*Unimplemented Star-specific CP3002*
86	*CP3011 Latvian*
87	*CP3012 Cyrillic*
88	*Unimplemented Star-specific CP3021*
89	*Unimplemented Star-specific CP3041*

continues on next page

Table 4 – continued from previous page

ID	Encoding
255	*Unknown*

POS5890 Series

POS-5890 thermal printer series, also marketed under various other names.

You can select this profile in python-escpos with this identifier: `POS-5890`. (Set parameter to *profile='POS-5890'*.)

Basic information

Name	POS5890 Series
Vendor	Zjiang
Media width (mm)	57.5
Media width (pixels)	384
DPI	203

Fonts

ID	Name	Columns
0	Font A	32
1	Font B	42

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	False
barcodeB	False
bitImageColumn	False
bitImageRaster	True
graphics	False
highDensity	True
paperFullCut	False
paperPartCut	False
pdf417Code	False
pulseBel	False
pulseStandard	True
qrCode	False
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*Unknown*
7	*Unknown*
8	*Unknown*
9	*Unknown*
10	*Unknown*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
20	*Unknown*
21	*Unknown*
22	*Unknown*
23	*Unknown*
24	*CP747*
25	*CP1257*
27	*CP1258*
28	*CP864*
31	*Unknown*
32	*CP1255*
50	*CP437*
52	*CP437*
53	*CP858*

continues on next page

Table 5 – continued from previous page

ID	Encoding
54	*CP852*
55	*CP860*
56	*CP861*
57	*CP863*
58	*CP865*
59	*CP866*
60	*CP855*
61	*CP857*
62	*CP862*
63	*CP864*
64	*CP737*
65	*Greek CP851*
66	*CP869*
68	*CP772*
69	*CP774*
71	*CP1252*
72	*CP1250*
73	*CP1251*
74	*Unimplemented Star-specific CP3840*
76	*Unimplemented Star-specific CP3843*
77	*Unimplemented Star-specific CP3844*
78	*Unimplemented Star-specific CP3845*
79	*Unimplemented Star-specific CP3846*
80	*Unimplemented Star-specific CP3847*
81	*Unimplemented Star-specific CP3848*
83	*Unimplemented Star-specific CP2001*
84	*Unimplemented Star-specific CP3001*
85	*Unimplemented Star-specific CP3002*
86	*CP3011 Latvian*
87	*CP3012 Cyrillic*
88	*Unimplemented Star-specific CP3021*
89	*Unimplemented Star-specific CP3041*
90	*CP1253*
91	*CP1254*
92	*CP1256*
93	*CP720*
94	*CP1258*
95	*CP775*
96	*Unknown*
255	*Unknown*

RP-F10

Seiko RP-F10 series with 58mm paper

You can select this profile in python-escpos with this identifier: `RP-F10-58mm`. (Set parameter to *profile='RP-F10-58mm'*.)

Basic information

Name	RP-F10
Vendor	Seiko
Media width (mm)	54
Media width (pixels)	432
DPI	203

Fonts

ID	Name	Columns
0	Font A	36
1	Font B	54

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*Katakana (codepage 1)*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
13	*CP857*
14	*CP737*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
34	*CP855*
37	*CP864*
45	*CP1250*
46	*CP1251*
47	*CP1253*
48	*CP1254*
255	*Unknown*

RP-F10

Seiko RP-F10 series with 80mm paper

You can select this profile in python-escpos with this identifier: `RP-F10-80mm`. (Set parameter to *profile='RP-F10-80mm'*.)

Basic information

Name	RP-F10
Vendor	Seiko
Media width (mm)	72
Media width (pixels)	576
DPI	203

Fonts

ID	Name	Columns
0	Font A	48
1	Font B	72

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*Katakana (codepage 1)*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
13	*CP857*
14	*CP737*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
34	*CP855*
37	*CP864*
45	*CP1250*
46	*CP1251*
47	*CP1253*
48	*CP1254*
255	*Unknown*

RP326

You can select this profile in python-escpos with this identifier: `RP326`. (Set parameter to *profile='RP326'*.)

Basic information

Name	RP326
Vendor	Rongta
Media width (mm)	Unknown
Media width (pixels)	Unknown
DPI	203

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	False
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*Katakana (codepage 1)*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*CP1251*
7	*CP866*
8	*Unknown*
9	*Unknown*
10	*Unknown*
15	*CP862*
16	*CP1252*
17	*CP1253*
18	*CP852*
19	*CP858*
20	*Unknown*
21	*Unknown*
22	*Unknown*
23	*ISO_8859-1*
24	*CP737*
25	*CP1257*
26	*Unknown*
27	*CP720*
28	*CP855*
29	*CP857*
30	*CP1250*
31	*CP775*
32	*CP1254*
33	*CP1255*
34	*CP1256*
35	*CP1258*
36	*ISO_8859-2*
37	*ISO_8859-3*
38	*ISO_8859-4*
39	*ISO_8859-5*
40	*ISO_8859-6*
41	*ISO_8859-7*
42	*ISO_8859-8*
43	*ISO_8859-9*
44	*ISO_8859-15*
45	*Unknown*
46	*CP856*
47	*CP874*

SP2000 Series

Star SP2000 impact printer series with ESC/POS emulation enabled

You can select this profile in python-escpos with this identifier: `SP2000`. (Set parameter to *profile='SP2000'*.)

Basic information

Name	SP2000 Series
Vendor	Star Micronics
Media width (mm)	Unknown
Media width (pixels)	Unknown
DPI	85

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	True

Text code pages

ID	Encoding
0	*CP437*
1	*CP437*
2	*CP932*
3	*CP437*
4	*CP858*
5	*CP852*
6	*CP860*
7	*CP861*
8	*CP863*
9	*CP865*
10	*CP866*
11	*CP855*
12	*CP857*
13	*CP862*
14	*CP864*
15	*CP737*
16	*Greek CP851*
17	*CP869*
18	*CP928*
19	*CP772*
20	*CP774*
21	*CP874*
32	*CP1252*
33	*CP1250*
34	*CP1251*
64	*Unimplemented Star-specific CP3840*
65	*Unimplemented Star-specific CP3841*
66	*Unimplemented Star-specific CP3843*
67	*Unimplemented Star-specific CP3844*
68	*Unimplemented Star-specific CP3845*
69	*Unimplemented Star-specific CP3846*
70	*Unimplemented Star-specific CP3847*
71	*Unimplemented Star-specific CP3848*
72	*Unimplemented Star-specific CP1001*
73	*Unimplemented Star-specific CP2001*
74	*Unimplemented Star-specific CP3001*
75	*Unimplemented Star-specific CP3002*
76	*CP3011 Latvian*
77	*CP3012 Cyrillic*
78	*Unimplemented Star-specific CP3021*
79	*Unimplemented Star-specific CP3041*
96	*Unknown*
97	*Unknown*
98	*Unknown*
99	*Unknown*
100	*Unknown*
101	*Unknown*
102	*Unknown*

continues on next page

Table 7 – continued from previous page

ID	Encoding
255	*Unknown*

Sunmi V2

Sunmi mini-POS Android device with a built-in Virtual Bluetooth thermal printer.

You can select this profile in python-escpos with this identifier: `Sunmi-V2`. (Set parameter to *profile='Sunmi-V2'*.)

Basic information

Name	Sunmi V2
Vendor	Sunmi
Media width (mm)	57.5
Media width (pixels)	384
DPI	Unknown

Fonts

ID	Name	Columns
0	Font A	32
1	Font B	42

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	False
bitImageRaster	True
graphics	False
highDensity	True
paperFullCut	False
paperPartCut	False
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
13	*CP857*
14	*CP737*
15	*ISO_8859-7*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
21	*CP874*
33	*CP775*
34	*CP855*
36	*CP862*
37	*CP864*
254	*CP855*

T-1

Metapace T-1 Thermal Printer Rev. 1.00

You can select this profile in python-escpos with this identifier: `T-1`. (Set parameter to *profile='T-1'*.)

Basic information

Name	T-1
Vendor	Metapace
Media width (mm)	80
Media width (pixels)	504
DPI	180

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*Katakana (codepage 1)*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
19	*CP858*
255	*Unknown*

TEP200M Series

You can select this profile in python-escpos with this identifier: `TEP-200M`. (Set parameter to *profile='TEP-200M'*.)

Basic information

Name	TEP200M Series
Vendor	EPOS
Media width (mm)	Unknown
Media width (pixels)	Unknown
DPI	203

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*Unknown*
7	*Unknown*
8	*Unknown*
11	*Greek CP851*
12	*CP853*
13	*CP857*
14	*CP737*
15	*ISO_8859-7*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
20	*Unknown*
21	*CP874*
22	*Unknown*
23	*Unknown*
24	*Unknown*
25	*Unknown*
26	*Unknown*
30	*Vietnamese TCVN-3 1*
31	*Vietnamese TCVN-3 1*
32	*CP720*

continues on next page

Table 8 – continued from previous page

ID	Encoding
33	*CP775*
34	*CP855*
35	*CP861*
36	*CP862*
37	*CP864*
38	*CP869*
39	*ISO_8859-2*
40	*ISO_8859-15*
41	*CP1098*
42	*CP774*
43	*CP772*
44	*CP1125*
45	*CP1250*
46	*CP1251*
47	*CP1253*
48	*CP1254*
49	*CP1255*
50	*CP1256*
51	*CP1257*
52	*CP1258*
53	*RK1048*
66	*Unknown*
67	*Unknown*
68	*Unknown*
69	*Unknown*
70	*Unknown*
71	*Unknown*
72	*Unknown*
73	*Unknown*
74	*Unknown*
75	*Unknown*
82	*Unknown*
254	*Unknown*
255	*Unknown*

TH230

Profile for TH230. Use bitImageColumn to print properly.

You can select this profile in python-escpos with this identifier: `TH230`. (Set parameter to *profile='TH230'*.)

Basic information

Name	TH230
Vendor	Wincor Nixdorf
Media width (mm)	72
Media width (pixels)	576
DPI	203

Fonts

ID	Name	Columns
0	Font A	44
1	Font B	57

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	False
pulseBel	False
pulseStandard	True
qrCode	False
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP850*
2	*CP852*
3	*CP860*
4	*CP863*
5	*CP865*
6	*CP858*
7	*CP866*
8	*CP1252*
9	*CP862*
10	*CP737*
11	*CP874*
12	*CP857*
16	*CP1254*
17	*CP1250*
18	*Unknown*
19	*Unknown*
20	*Unknown*
21	*Unknown*
22	*CP864*
23	*CP720*
24	*CP1256*
25	*Unknown*
26	*Katakana (codepage 1)*
27	*CP775*
28	*CP1257*
29	*Unknown*

TH230+

Profile for TH230+. Use bitImageColumn to print properly. TH230+ supports native qr codes and PDF417 codes

You can select this profile in python-escpos with this identifier: `TH230Plus`. (Set parameter to *profile='TH230Plus'*.)

Basic information

Name	TH230+
Vendor	Wincor Nixdorf
Media width (mm)	72
Media width (pixels)	576
DPI	203

Fonts

ID	Name	Columns
0	Font A	44
1	Font B	57

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP850*
2	*CP852*
3	*CP860*
4	*CP863*
5	*CP865*
6	*CP858*
7	*CP866*
8	*CP1252*
9	*CP862*
10	*CP737*
11	*CP874*
12	*CP857*
16	*CP1254*
17	*CP1250*
18	*Unknown*
19	*Unknown*
20	*Unknown*
21	*Unknown*
22	*CP864*
23	*CP720*
24	*CP1256*
25	*Unknown*
26	*Katakana (codepage 1)*
27	*CP775*
28	*CP1257*
29	*Unknown*

TM-L90

Epson TM-L90 profile. The standard 80mm paper width version was used here. The code page mapping is documented in the "TM-L90 Technical Reference Guide".

You can select this profile in python-escpos with this identifier: `TM-L90`. (Set parameter to *profile='TM-L90'*.)

Basic information

Name	TM-L90
Vendor	Epson
Media width (mm)	80
Media width (pixels)	576
DPI	203

Fonts

ID	Name	Columns
0	Font A	48
1	Font B	64

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
11	*Greek CP851*
12	*CP853*
13	*CP857*
14	*CP737*
15	*ISO_8859-7*

continues on next page

Table 9 – continued from previous page

ID	Encoding
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
20	*Unknown*
21	*CP874*
22	*Unknown*
23	*Unknown*
24	*Unknown*
25	*Unknown*
26	*Unknown*
30	*Vietnamese TCVN-3 1*
31	*Vietnamese TCVN-3 1*
32	*CP720*
33	*CP775*
34	*CP855*
35	*CP861*
36	*CP862*
37	*CP864*
38	*CP869*
39	*ISO_8859-2*
40	*ISO_8859-15*
41	*CP1098*
42	*CP774*
43	*CP772*
44	*CP1125*
45	*CP1250*
46	*CP1251*
47	*CP1253*
48	*CP1254*
49	*CP1255*
50	*CP1256*
51	*CP1257*
52	*CP1258*
53	*RK1048*
255	*Unknown*

TM-P80

Portable printer (48-column mode)

You can select this profile in python-escpos with this identifier: `TM-P80`. (Set parameter to *profile='TM-P80'*.)

Basic information

Name	TM-P80
Vendor	Epson
Media width (mm)	72
Media width (pixels)	576
DPI	203

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56
2	Kanji	24

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*Unknown*
7	*Unknown*
8	*Unknown*
11	*Greek CP851*
12	*CP853*
13	*CP857*
14	*CP737*
15	*ISO_8859-7*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
20	*Unknown*
21	*CP874*
22	*Unknown*
23	*Unknown*
24	*Unknown*
25	*Unknown*
26	*Unknown*
30	*Vietnamese TCVN-3 1*
31	*Vietnamese TCVN-3 1*
32	*CP720*
33	*CP775*
34	*CP855*
35	*CP861*
36	*CP862*
37	*CP864*
38	*CP869*
39	*ISO_8859-2*
40	*ISO_8859-15*
41	*CP1098*
42	*CP774*
43	*CP772*
44	*CP1125*
45	*CP1250*
46	*CP1251*
47	*CP1253*
48	*CP1254*
49	*CP1255*
50	*CP1256*
51	*CP1257*
52	*CP1258*

continues on next page

Table 10 – continued from previous page

ID	Encoding
53	*RK1048*
66	*Unknown*
67	*Unknown*
68	*Unknown*
69	*Unknown*
70	*Unknown*
71	*Unknown*
72	*Unknown*
73	*Unknown*
74	*Unknown*
75	*Unknown*
82	*Unknown*
254	*Unknown*
255	*Unknown*

TM-P80 (42 column mode)

Portable printer (42-column mode)

You can select this profile in python-escpos with this identifier: `TM-P80-42col`. (Set parameter to *profile='TM-P80-42col'*.)

Basic information

Name	TM-P80 (42 column mode)
Vendor	Epson
Media width (mm)	63.6
Media width (pixels)	546
DPI	Unknown

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	60
2	Kanji	21

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*Unknown*
7	*Unknown*
8	*Unknown*
11	*Greek CP851*
12	*CP853*
13	*CP857*
14	*CP737*
15	*ISO_8859-7*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
20	*Unknown*
21	*CP874*

continues on next page

Table 11 – continued from previous page

ID	Encoding
22	*Unknown*
23	*Unknown*
24	*Unknown*
25	*Unknown*
26	*Unknown*
30	*Vietnamese TCVN-3 1*
31	*Vietnamese TCVN-3 1*
32	*CP720*
33	*CP775*
34	*CP855*
35	*CP861*
36	*CP862*
37	*CP864*
38	*CP869*
39	*ISO_8859-2*
40	*ISO_8859-15*
41	*CP1098*
42	*CP774*
43	*CP772*
44	*CP1125*
45	*CP1250*
46	*CP1251*
47	*CP1253*
48	*CP1254*
49	*CP1255*
50	*CP1256*
51	*CP1257*
52	*CP1258*
53	*RK1048*
66	*Unknown*
67	*Unknown*
68	*Unknown*
69	*Unknown*
70	*Unknown*
71	*Unknown*
72	*Unknown*
73	*Unknown*
74	*Unknown*
75	*Unknown*
82	*Unknown*
254	*Unknown*
255	*Unknown*

TM-T20II

Epson TM-T20II profile

You can select this profile in python-escpos with this identifier: `TM-T20II`. (Set parameter to *profile='TM-T20II'*.)

Basic information

Name	TM-T20II
Vendor	Epson
Media width (mm)	72
Media width (pixels)	576
DPI	203

Fonts

ID	Name	Columns
0	Font A	48
1	Font B	64

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
11	*Greek CP851*
12	*CP853*
13	*CP857*
14	*CP737*
15	*ISO_8859-7*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
30	*Vietnamese TCVN-3 1*
31	*Vietnamese TCVN-3 1*
32	*CP720*
33	*CP775*
34	*CP855*
35	*CP861*
36	*CP862*
37	*CP864*
38	*CP869*
39	*ISO_8859-2*
40	*ISO_8859-15*
41	*CP1098*
44	*CP1125*
45	*CP1250*
46	*CP1251*
47	*CP1253*
48	*CP1254*
49	*CP1255*
50	*CP1256*
51	*CP1257*
52	*CP1258*
53	*RK1048*
255	*Unknown*

TM-T20II (42 column mode)

Epson TM-T20II profile (42 column mode)

You can select this profile in python-escpos with this identifier: `TM-T20II-42col`. (Set parameter to *profile='TM-T20II-42col'*.)

Basic information

Name	TM-T20II (42 column mode)
Vendor	Epson
Media width (mm)	68.3
Media width (pixels)	546
DPI	Unknown

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	60

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
11	*Greek CP851*
12	*CP853*
13	*CP857*
14	*CP737*
15	*ISO_8859-7*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
30	*Vietnamese TCVN-3 1*
31	*Vietnamese TCVN-3 1*
32	*CP720*
33	*CP775*
34	*CP855*
35	*CP861*
36	*CP862*
37	*CP864*
38	*CP869*
39	*ISO_8859-2*
40	*ISO_8859-15*
41	*CP1098*
44	*CP1125*
45	*CP1250*
46	*CP1251*
47	*CP1253*
48	*CP1254*
49	*CP1255*
50	*CP1256*
51	*CP1257*
52	*CP1258*
53	*RK1048*
255	*Unknown*

TM-T88II

Epson TM-T88II profile. The specs where taken from a TM-T88IIP machine (I assume the P just stands for parallel port). The standart 80mm paper width version was used here. There is also a custom 58mm factory option. If you are using the custom version change media width to 50.8mm and 360px accordingly. This printer is discontinued by the Vendor and has similar feature support to the TM-T88III. The code page mapping is documented in the "TM-T88II/T88III Technical Reference Guide".

You can select this profile in python-escpos with this identifier: `TM-T88II`. (Set parameter to *profile='TM-T88II'*.)

Basic information

Name	TM-T88II
Vendor	Epson
Media width (mm)	72
Media width (pixels)	512
DPI	180

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*Unknown*
7	*Unknown*
8	*Unknown*
11	*Unknown*
12	*Unknown*
13	*CP857*
14	*CP737*
15	*ISO_8859-7*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*Unknown*
20	*Unknown*
21	*CP874*
22	*Unknown*
23	*Unknown*
24	*Unknown*
25	*Unknown*
26	*Unknown*
30	*Vietnamese TCVN-3 1*
31	*Vietnamese TCVN-3 1*
32	*Unknown*

continues on next page

Table 14 – continued from previous page

ID	Encoding
33	*CP775*
34	*CP855*
35	*CP861*
36	*CP862*
37	*CP864*
38	*CP869*
39	*ISO_8859-2*
40	*ISO_8859-15*
41	*Unknown*
42	*CP774*
43	*CP772*
44	*CP1125*
45	*CP1250*
46	*CP1251*
47	*CP1253*
48	*CP1254*
49	*CP1255*
50	*CP1256*
51	*CP1257*
52	*CP1258*
53	*RK1048*
66	*Unknown*
67	*Unknown*
68	*Unknown*
69	*Unknown*
70	*Unknown*
71	*Unknown*
72	*Unknown*
73	*Unknown*
74	*Unknown*
75	*Unknown*
82	*Unknown*
254	*Unknown*
255	*Unknown*

TM-T88III

Epson TM-T88III profile. This printer has similar feature support to the TM-T88II. The code page mapping is documented in the "TM-T88II/T88III Technical Reference Guide".

You can select this profile in python-escpos with this identifier: `TM-T88III`. (Set parameter to *profile='TM-T88III'*.)

Basic information

Name	TM-T88III
Vendor	Epson
Media width (mm)	80
Media width (pixels)	512
DPI	180

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
16	*CP1252*
17	*CP866*
18	*CP862*
19	*CP858*
255	*Unknown*

TM-T88IV

Epson TM-T88IV profile

You can select this profile in python-escpos with this identifier: `TM-T88IV`. (Set parameter to *profile='TM-T88IV'*.)

Basic information

Name	TM-T88IV
Vendor	Epson
Media width (mm)	80
Media width (pixels)	512
DPI	180

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
255	*Unknown*

TM-T88IV South Asia

Epson TM-T88IV profile (South Asia models)

You can select this profile in python-escpos with this identifier: `TM-T88IV-SA`. (Set parameter to *profile='TM-T88IV-SA'*.)

Basic information

Name	TM-T88IV South Asia
Vendor	Epson
Media width (mm)	80
Media width (pixels)	512
DPI	180

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
20	*Unknown*
21	*CP874*
26	*Unknown*
30	*Vietnamese TCVN-3 1*
31	*Vietnamese TCVN-3 1*

TM-T88V

Epson TM-T88V profile

You can select this profile in python-escpos with this identifier: `TM-T88V`. (Set parameter to *profile='TM-T88V'*.)

Basic information

Name	TM-T88V
Vendor	Epson
Media width (mm)	80
Media width (pixels)	512
DPI	180

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
11	*Greek CP851*
12	*CP853*
13	*CP857*
14	*CP737*
15	*ISO_8859-7*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
30	*Vietnamese TCVN-3 1*
31	*Vietnamese TCVN-3 1*
32	*CP720*
33	*CP775*
34	*CP855*
35	*CP861*
36	*CP862*
37	*CP864*
38	*CP869*
39	*ISO_8859-2*
40	*ISO_8859-15*
41	*CP1098*
45	*CP1250*

continues on next page

Table 15 – continued from previous page

ID	Encoding
46	*CP1251*
47	*CP1253*
48	*CP1254*
49	*CP1255*
50	*CP1256*
51	*CP1257*
52	*CP1258*
53	*RK1048*
255	*Unknown*

TM-U220

Two-color impact printer with 80mm output

You can select this profile in python-escpos with this identifier: `TM-U220`. (Set parameter to *profile='TM-U220'*.)

Basic information

Name	TM-U220
Vendor	Epson
Media width (mm)	63.4
Media width (pixels)	400
DPI	Unknown

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black
1	alternate

Feature support

Feature	Supported
barcodeA	False
barcodeB	False
bitImageColumn	True
bitImageRaster	False
graphics	False
highDensity	False
paperFullCut	False
paperPartCut	False
pdf417Code	False
pulseBel	False
pulseStandard	True
qrCode	False
starCommands	False

Text code pages

ID	Encoding
0	*CP437*

TM-U220B

Two-color impact printer with 76mm output

You can select this profile in python-escpos with this identifier: `TM-U220B`. (Set parameter to *profile='TM-U220B'*.)

Basic information

Name	TM-U220B
Vendor	Epson
Media width (mm)	63.4
Media width (pixels)	400
DPI	Unknown

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black
1	alternate

Feature support

Feature	Supported
barcodeA	False
barcodeB	False
bitImageColumn	True
bitImageRaster	False
graphics	False
highDensity	False
paperFullCut	False
paperPartCut	True
pdf417Code	False
pulseBel	False
pulseStandard	True
qrCode	False
starCommands	False

Text code pages

ID	Encoding
0	*CP437*

TSP600 Series

Star TSP600 thermal printer series with ESC/POS emulation enabled

You can select this profile in python-escpos with this identifier: `TSP600`. (Set parameter to *profile='TSP600'*.)

Basic information

Name	TSP600 Series
Vendor	Star Micronics
Media width (mm)	72
Media width (pixels)	576
DPI	203

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	True

Text code pages

ID	Encoding
0	*CP437*
1	*CP437*
2	*CP932*
3	*CP437*
4	*CP858*
5	*CP852*
6	*CP860*
7	*CP861*
8	*CP863*
9	*CP865*
10	*CP866*
11	*CP855*
12	*CP857*
13	*CP862*
14	*CP864*
15	*CP737*
16	*Greek CP851*
17	*CP869*
18	*CP928*
19	*CP772*
20	*CP774*
21	*CP874*
32	*CP1252*
33	*CP1250*
34	*CP1251*
64	*Unimplemented Star-specific CP3840*
65	*Unimplemented Star-specific CP3841*
66	*Unimplemented Star-specific CP3843*

continues on next page

Table 16 – continued from previous page

ID	Encoding
67	*Unimplemented Star-specific CP3844*
68	*Unimplemented Star-specific CP3845*
69	*Unimplemented Star-specific CP3846*
70	*Unimplemented Star-specific CP3847*
71	*Unimplemented Star-specific CP3848*
72	*Unimplemented Star-specific CP1001*
73	*Unimplemented Star-specific CP2001*
74	*Unimplemented Star-specific CP3001*
75	*Unimplemented Star-specific CP3002*
76	*CP3011 Latvian*
77	*CP3012 Cyrillic*
78	*Unimplemented Star-specific CP3021*
79	*Unimplemented Star-specific CP3041*
96	*Unknown*
97	*Unknown*
98	*Unknown*
99	*Unknown*
100	*Unknown*
101	*Unknown*
102	*Unknown*
255	*Unknown*

TUP500 Series

Star TUP500 thermal printer series with ESC/POS emulation enabled

You can select this profile in python-escpos with this identifier: `TUP500`. (Set parameter to *profile='TUP500'*.)

Basic information

Name	TUP500 Series
Vendor	Star Micronics
Media width (mm)	80
Media width (pixels)	Unknown
DPI	203

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	True

Text code pages

ID	Encoding
0	*CP437*
1	*CP437*
2	*CP932*
3	*CP437*
4	*CP858*
5	*CP852*
6	*CP860*
7	*CP861*
8	*CP863*
9	*CP865*
10	*CP866*
11	*CP855*
12	*CP857*
13	*CP862*
14	*CP864*
15	*CP737*
16	*Greek CP851*
17	*CP869*
18	*CP928*
19	*CP772*

continues on next page

Table 17 – continued from previous page

ID	Encoding
20	*CP774*
21	*CP874*
32	*CP1252*
33	*CP1250*
34	*CP1251*
64	*Unimplemented Star-specific CP3840*
65	*Unimplemented Star-specific CP3841*
66	*Unimplemented Star-specific CP3843*
67	*Unimplemented Star-specific CP3844*
68	*Unimplemented Star-specific CP3845*
69	*Unimplemented Star-specific CP3846*
70	*Unimplemented Star-specific CP3847*
71	*Unimplemented Star-specific CP3848*
72	*Unimplemented Star-specific CP1001*
73	*Unimplemented Star-specific CP2001*
74	*Unimplemented Star-specific CP3001*
75	*Unimplemented Star-specific CP3002*
76	*CP3011 Latvian*
77	*CP3012 Cyrillic*
78	*Unimplemented Star-specific CP3021*
79	*Unimplemented Star-specific CP3041*
96	*Unknown*
97	*Unknown*
98	*Unknown*
99	*Unknown*
100	*Unknown*
101	*Unknown*
102	*Unknown*
255	*Unknown*

ZJ-5870 Thermal Receipt Printer

ESC/POS Profile for ZiJiang ZJ-5870 Thermal Receipt Printer, which may be branded AGPtEK or Esky, and identifies itself as a POS-58 Thermal Printer on selftest. This profile is suitable for alphanumberic character mode, but is untested on Chinese character mode. (Change modes by holding down feed button during power-on until the mode LED turns off, then release immediately.)

You can select this profile in python-escpos with this identifier: `ZJ-5870`. (Set parameter to *profile='ZJ-5870'*.)

Basic information

Name	ZJ-5870 Thermal Receipt Printer
Vendor	ZiJiang
Media width (mm)	48
Media width (pixels)	384
DPI	203

Fonts

ID	Name	Columns
0	Font A	32

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	False
barcodeB	False
bitImageColumn	True
bitImageRaster	True
graphics	False
highDensity	False
paperFullCut	False
paperPartCut	False
pdf417Code	False
pulseBel	False
pulseStandard	True
qrCode	False
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
16	*CP1252*
17	*CP866*
18	*CP852*

Default

Default ESC/POS profile, suitable for standards-compliant or Epson-branded printers. This profile allows the use of standard ESC/POS features, and can encode a variety of code pages.

You can select this profile in python-escpos with this identifier: `default`. (Set parameter to *profile='default'*.)

Basic information

Name	Default
Vendor	Generic
Media width (mm)	Unknown
Media width (pixels)	Unknown
DPI	Unknown

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	True
barcodeB	True
bitImageColumn	True
bitImageRaster	True
graphics	True
highDensity	True
paperFullCut	True
paperPartCut	True
pdf417Code	True
pulseBel	False
pulseStandard	True
qrCode	True
starCommands	False

Text code pages

ID	Encoding
0	*CP437*
1	*CP932*
2	*CP850*
3	*CP860*
4	*CP863*
5	*CP865*
6	*Unknown*
7	*Unknown*
8	*Unknown*
11	*Greek CP851*
12	*CP853*
13	*CP857*
14	*CP737*
15	*ISO_8859-7*
16	*CP1252*
17	*CP866*
18	*CP852*
19	*CP858*
20	*Unknown*
21	*CP874*
22	*Unknown*
23	*Unknown*
24	*Unknown*
25	*Unknown*
26	*Unknown*
30	*Vietnamese TCVN-3 1*
31	*Vietnamese TCVN-3 1*
32	*CP720*
33	*CP775*
34	*CP855*
35	*CP861*
36	*CP862*
37	*CP864*
38	*CP869*
39	*ISO_8859-2*
40	*ISO_8859-15*
41	*CP1098*
42	*CP774*
43	*CP772*
44	*CP1125*
45	*CP1250*
46	*CP1251*
47	*CP1253*
48	*CP1254*
49	*CP1255*
50	*CP1256*
51	*CP1257*
52	*CP1258*

continues on next page

Table 18 – continued from previous page

ID	Encoding
53	*RK1048*
66	*Unknown*
67	*Unknown*
68	*Unknown*
69	*Unknown*
70	*Unknown*
71	*Unknown*
72	*Unknown*
73	*Unknown*
74	*Unknown*
75	*Unknown*
82	*Unknown*
254	*Unknown*
255	*Unknown*

Simple

A profile for use in printers with unknown or poor compatibility. This profile indicates that a small number of features are supported, so that commands are not sent a printer that is unlikely to understand them.

You can select this profile in python-escpos with this identifier: `simple`. (Set parameter to *profile='simple'*.)

Basic information

Name	Simple
Vendor	Generic
Media width (mm)	Unknown
Media width (pixels)	Unknown
DPI	Unknown

Fonts

ID	Name	Columns
0	Font A	42
1	Font B	56

Colors

ID	Color
0	black

Feature support

Feature	Supported
barcodeA	False
barcodeB	False
bitImageColumn	False
bitImageRaster	True
graphics	False
highDensity	True
paperFullCut	False
paperPartCut	False
pdf417Code	False
pulseBel	False
pulseStandard	True
qrCode	False
starCommands	False

Text code pages

ID	Encoding
0	*CP437*

2.2.3 Available Encodings

Last Reviewed
2023-08-10

If you find any issues with the described encodings, please open an issue in the ESC/POS printer database. The data shown here is directly taken from there.

Unimplemented Star-specific CP1001

Mapping Information

identifier	CP1001
Name	Unimplemented Star-specific CP1001
Iconv Name	Unknown
`python_encode` Name	Unknown

CP1098

Mapping Information

identifier	CP1098
Name	CP1098
Iconv Name	Unknown
`python_encode` Name	Unknown

CP1125

Mapping Information

identifier	CP1125
Name	CP1125
Iconv Name	CP1125
`python_encode` Name	cp1125

CP1250

Mapping Information

identifier	CP1250
Name	CP1250
Iconv Name	CP1250
`python_encode` Name	cp1250

CP1251

Mapping Information

identifier	CP1251
Name	CP1251
Iconv Name	CP1251
`python_encode` Name	cp1251

CP1252

Mapping Information

identifier	CP1252
Name	CP1252
Iconv Name	CP1252
`python_encode` Name	cp1252

CP1253

Mapping Information

identifier	CP1253
Name	CP1253
Iconv Name	CP1253
`python_encode` Name	cp1253

CP1254

Mapping Information

identifier	CP1254
Name	CP1254
Iconv Name	CP1254
`python_encode` Name	cp1254

CP1255

Mapping Information

identifier	CP1255
Name	CP1255
Iconv Name	CP1255
`python_encode` Name	cp1255

CP1256

Mapping Information

identifier	CP1256
Name	CP1256
Iconv Name	CP1256
`python_encode` Name	cp1256

CP1257

Mapping Information

identifier	CP1257
Name	CP1257
Iconv Name	CP1257
`python_encode` Name	cp1257

CP1258

Mapping Information

identifier	CP1258
Name	CP1258
Iconv Name	CP1258
`python_encode` Name	cp1258

Unimplemented Star-specific CP2001

Mapping Information

identifier	CP2001
Name	Unimplemented Star-specific CP2001
Iconv Name	Unknown
`python_encode` Name	Unknown

Unimplemented Star-specific CP3001

Mapping Information

identifier	CP3001
Name	Unimplemented Star-specific CP3001
Iconv Name	Unknown
`python_encode` Name	Unknown

Unimplemented Star-specific CP3002

Mapping Information

identifier	CP3002
Name	Unimplemented Star-specific CP3002
Iconv Name	Unknown
`python_encode` Name	Unknown

CP3011 Latvian

Mapping Information

identifier	CP3011
Name	CP3011 Latvian
Iconv Name	Unknown
`python_encode` Name	Unknown

Code page data

[‘Çüéâäàåçêëèïî ÄÅ’, ‘ÉæÆôöòûùÿÖÜ¢£¥Pts ƒ’, ‘áíóúñÑ ºº¿⌐¬½¼¡«»’, ‘ ▒▓ | ┤ Ā╢ ņ╕ ╡╣ ║╗ ╝╜ ╛ ┐ ’, ‘ └┴┬ ├─┼ā ╟ ╚ ╔╩╦ ╠═╬╧ ’, ‘Š╤čČ ╙ ╒ģĪ╛ ┌ ▀ ūŪ■’, ‘αßΓπΣσμτΦΘΩδ∞φεՈ’, ‘ĒēĢķĶļĻžŽ•·√Ņš■ ‘]

CP3012 Cyrillic

Mapping Information

identifier	CP3012
Name	CP3012 Cyrillic
Iconv Name	Unknown
`python_encode` Name	Unknown

Code page data

[‘АБВГДЕЖЗИЙКЛМНОП’, ‘РСТУФХЦЧШЩЪЫЬЭЮЯ’, ‘абвгдежзийклмноп’, ‘ ▒▓ | ┤ Ā╢ ņ╕ ╡╣ ║╗ ╝╜ Ō╛ ┐ ’, ‘ └┴┬ ├─┼ā ╟ ╚ ╔╩╦ ╠═╬╧ ’, ‘Š╤čČ ╙ ╒ģĪ╛ ┌ ▀ ūŪ■’, ‘рстуфхцчшщъыьэюя’, ‘ĒēĢķĶļĻžŽ•·√Ņš■ ‘]

Unimplemented Star-specific CP3021

Mapping Information

identifier	CP3021
Name	Unimplemented Star-specific CP3021
Iconv Name	Unknown
`python_encode` Name	Unknown

Unimplemented Star-specific CP3041

Mapping Information

identifier	CP3041
Name	Unimplemented Star-specific CP3041
Iconv Name	Unknown
`python_encode` Name	Unknown

Unimplemented Star-specific CP3840

Mapping Information

identifier	CP3840
Name	Unimplemented Star-specific CP3840
Iconv Name	Unknown
`python_encode` Name	Unknown

Unimplemented Star-specific CP3841

Mapping Information

identifier	CP3841
Name	Unimplemented Star-specific CP3841
Iconv Name	Unknown
`python_encode` Name	Unknown

Unimplemented Star-specific CP3843

Mapping Information

identifier	CP3843
Name	Unimplemented Star-specific CP3843
Iconv Name	Unknown
`python_encode` Name	Unknown

Unimplemented Star-specific CP3844

Mapping Information

identifier	CP3844
Name	Unimplemented Star-specific CP3844
Iconv Name	Unknown
`python_encode` Name	Unknown

Unimplemented Star-specific CP3845

Mapping Information

identifier	CP3845
Name	Unimplemented Star-specific CP3845
Iconv Name	Unknown
`python_encode` Name	Unknown

Unimplemented Star-specific CP3846

Mapping Information

identifier	CP3846
Name	Unimplemented Star-specific CP3846
Iconv Name	Unknown
`python_encode` Name	Unknown

Unimplemented Star-specific CP3847

Mapping Information

identifier	CP3847
Name	Unimplemented Star-specific CP3847
Iconv Name	Unknown
`python_encode` Name	Unknown

Unimplemented Star-specific CP3848

Mapping Information

identifier	CP3848
Name	Unimplemented Star-specific CP3848
Iconv Name	Unknown
`python_encode` Name	Unknown

CP437

Mapping Information

identifier	CP437
Name	CP437
Iconv Name	CP437
`python_encode` Name	cp437

CP720

Mapping Information

identifier	CP720
Name	CP720
Iconv Name	Unknown
`python_encode` Name	cp720

CP737

Mapping Information

identifier	CP737
Name	CP737
Iconv Name	CP737
`python_encode` Name	cp737

CP747

Mapping Information

identifier	CP747
Name	CP747
Iconv Name	Unknown
`python_encode` Name	Unknown

CP772

Mapping Information

identifier	CP772
Name	CP772
Iconv Name	CP772
`python_encode` Name	Unknown

CP774

Mapping Information

identifier	CP774
Name	CP774
Iconv Name	CP774
`python_encode` Name	Unknown

CP775

Mapping Information

identifier	CP775
Name	CP775
Iconv Name	CP775
`python_encode` Name	cp775

CP850

Mapping Information

identifier	CP850
Name	CP850
Iconv Name	CP850
`python_encode` Name	cp850

Greek CP851

Not used, due to inconsistencies between implementations.

Mapping Information

identifier	CP851
Name	Greek CP851
Iconv Name	Unknown
`python_encode` Name	Unknown

CP852

Mapping Information

identifier	CP852
Name	CP852
Iconv Name	CP852
`python_encode` Name	cp852

CP853

Mapping Information

identifier	CP853
Name	CP853
Iconv Name	Unknown
`python_encode` Name	Unknown

CP855

Mapping Information

identifier	CP855
Name	CP855
Iconv Name	CP855
`python_encode` Name	cp855

CP856

Mapping Information

identifier	CP856
Name	CP856
Iconv Name	CP856
`python_encode` Name	cp856

CP857

Mapping Information

identifier	CP857
Name	CP857
Iconv Name	CP857
`python_encode` Name	cp857

CP858

Mapping Information

identifier	CP858
Name	CP858
Iconv Name	Unknown
`python_encode` Name	cp858

CP860

Mapping Information

identifier	CP860
Name	CP860
Iconv Name	CP860
`python_encode` Name	cp860

CP861

Mapping Information

identifier	CP861
Name	CP861
Iconv Name	CP861
`python_encode` Name	cp861

CP862

Mapping Information

identifier	CP862
Name	CP862
Iconv Name	CP862
`python_encode` Name	cp862

CP863

Mapping Information

identifier	CP863
Name	CP863
Iconv Name	CP863
`python_encode` Name	cp863

CP864

Mapping Information

identifier	CP864
Name	CP864
Iconv Name	CP864
`python_encode` Name	cp864

CP865

Mapping Information

identifier	CP865
Name	CP865
Iconv Name	CP865
`python_encode` Name	cp865

CP866

Mapping Information

identifier	CP866
Name	CP866
Iconv Name	CP866
`python_encode` Name	cp866

CP869

Mapping Information

identifier	CP869
Name	CP869
Iconv Name	CP869
`python_encode` Name	cp869

CP874

Mapping Information

identifier	CP874
Name	CP874
Iconv Name	CP874
`python_encode` Name	cp874

CP928

Mapping Information

identifier	CP928
Name	CP928
Iconv Name	Unknown
`python_encode` Name	Unknown

CP932

Mapping Information

identifier	CP932
Name	CP932
Iconv Name	CP932
`python_encode` Name	cp932

ISO_8859-1

Mapping Information

identifier	ISO_8859-1
Name	ISO_8859-1
Iconv Name	ISO_8859-1
`python_encode` Name	latin_1

ISO_8859-15

Mapping Information

identifier	ISO_8859-15
Name	ISO_8859-15
Iconv Name	ISO_8859-15
`python_encode` Name	iso8859-15

ISO_8859-2

Mapping Information

identifier	ISO_8859-2
Name	ISO_8859-2
Iconv Name	ISO_8859-2
`python_encode` Name	iso8859_2

ISO_8859-3

Mapping Information

identifier	ISO_8859-3
Name	ISO_8859-3
Iconv Name	ISO_8859-3
`python_encode` Name	iso8859_3

ISO_8859-4

Mapping Information

identifier	ISO_8859-4
Name	ISO_8859-4
Iconv Name	ISO_8859-4
`python_encode` Name	iso8859_4

ISO_8859-5

Mapping Information

identifier	ISO_8859-5
Name	ISO_8859-5
Iconv Name	ISO_8859-5
`python_encode` Name	iso8859_5

ISO_8859-6

Mapping Information

identifier	ISO_8859-6
Name	ISO_8859-6
Iconv Name	ISO_8859-6
`python_encode` Name	iso8859_6

ISO_8859-7

Mapping Information

identifier	ISO_8859-7
Name	ISO_8859-7
Iconv Name	ISO_8859-7
`python_encode` Name	iso8859_7

ISO_8859-8

Mapping Information

identifier	ISO_8859-8
Name	ISO_8859-8
Iconv Name	ISO_8859-8
`python_encode` Name	iso8859_8

ISO_8859-9

Mapping Information

identifier	ISO_8859-9
Name	ISO_8859-9
Iconv Name	ISO_8859-9
`python_encode` Name	iso8859_9

Katakana (codepage 1)

Mapping Information

identifier	KATAKANA
Name	Katakana (codepage 1)
Iconv Name	Unknown
`python_encode` Name	Unknown

Code page data

['▁▂▃▄▅▆▇█ ▌▐▀▄▅▆▇█┼', '└┬┤ ├╴─ ▏▕┌┐ ╙ ╭╮ ╰╯ ', ' □□□□□□□□□□□□□□□□', '□□□□□□□□□□□□□□□□□□□□', '□□□□□□□□□□□□□□□□□□□□', '□□□□□□□□□□□□□□□□□□□□', '═ ╞╪╡ ◢◣◥◤♠♥♦♣●○╱╲', '╳□□□□□□□□□□□□□□□▓xa0']

Oxhoo-specific European

Mapping Information

identifier	OXHOO-EUROPEAN
Name	Oxhoo-specific European
Iconv Name	Unknown
`python_encode` Name	Unknown

Code page data

['ÇüéâäàåçêëèïîìÄÅ', 'ÉæÆôöòûùÿÖÜñÑªº¿', 'áíóú¢£¥₧ƒ¡ÃãÕõØø', '·¨°˘˙½¼×÷≤≥«»≠√⁻', '∫⌠ ∞◤↵↑↓→← ┌┐ └┘ •®©', '™†§¶Γ◢Θ ', 'ß ε ', 'τ ']

RK1048

Mapping Information

identifier	RK1048
Name	RK1048
Iconv Name	RK1048
`python_encode` Name	Unknown

Vietnamese TCVN-3 1

Mapping Information

identifier	TCVN-3-1
Name	Vietnamese TCVN-3 1
Iconv Name	Unknown
`python_encode` Name	Unknown

Code page data

[’ ‘, ‘ ‘, ‘ ăâêôơđ ‘, ‘ àảãáạ ằẳẵắ ‘, ‘ ặầẩẫấậè ẻẽ’, ‘éẹèẻẽếệìỉ ĩíịò’, ‘ ỏõóọồổỗốộờởỡớợù’, ‘ ủũúụừửữứựỳỷỹýỵ ‘]

Vietnamese TCVN-3 1

Mapping Information

identifier	TCVN-3-2
Name	Vietnamese TCVN-3 1
Iconv Name	Unknown
`python_encode` Name	Unknown

Code page data

[’ ‘, ‘ ‘, ‘ ĂÂ Đ ÊÔƠƯ ‘, ‘ ÀẢÃÁẠ ẰẲẴẮ ‘, ‘ ẶẦẨẪẤẬÈ ẺẼ’, ‘ÉẸỀỂỄẾỆÌỈ ĨÍỊÒ’, ‘ ỎÕÓỌỒỔỖỐỘỜỞỠỚỢÙ’, ‘ỦŨÚỤỪỬỮỨỰỲỶỸÝỴ ‘]

Unknown

Code page that has not yet been identified.

Mapping Information

identifier	Unknown
Name	Unknown
Iconv Name	Unknown
`python_encode` Name	Unknown

2.3 Developer Documentation

This chapter summarizes information for developers of this library.

2.3.1 Release process

Last Reviewed
2023-08-10

- Update authors file
- Update changelog
- Set annotated tag for release and push to public github
- Build wheel
- Load wheel to PyPi
- Prepare project for next release with an empty changelog entry

2.3.2 Contributing

Last Reviewed
2023-08-10

This project is open to any kind of contribution. You can help with improving the documentation, adding fixes to the code, providing test cases in code or as a description or just spreading the word. Please feel free to create an issue or pull request. In order to reduce the amount of work for everyone please try to adhere to good practice.

The pull requests and issues will be prefilled with templates. Please fill in your information where applicable.

This project uses semantic versioning and tries to adhere to the proposed rules as well as possible.

Author-list

This project keeps a list of authors. This can be auto-generated by calling *./doc/generate-authors.sh*. When contributing the first time, please include a commit with the output of this script in place.

When you change your username or mail-address, please also update the *.mailmap* and the authors-list. You can find a good documentation on the mapping-feature in the documentation of git-shortlog.

Style-Guide

When writing code please try to stick to these rules.

Black Code Style

This project is formatted with the auto formatter black. Please format your contributions with black, otherwise they will be rejected.

GIT

The master-branch contains the main development of the project. A release to PyPi is marked with a tag corresponding to the version. Issues are closed when they have been resolved by merging into the master-branch. When you have a change to make, begin by creating a new branch from the HEAD of *python-escpos/master*.

Please try to group your commits into logical units. If you need to tidy up your branch, you can make use of a git feature called an 'interactive rebase' before making a pull request. A small, self-contained change-set is easier to review, and improves the chance of your code being merged. Please also make sure that before creating your PR, your branch is rebased on a recent commit or you merged a recent commit into your branch. This way you can ensure that your PR is without merge conflicts.

Docstrings

This project tries to have a good documentation. Please add a docstring to every method and class. Have a look at existing methods and classes for the style. We use basically standard rst-docstrings for Sphinx.

Test

Try to write tests whenever possible. Our goal for the future is 100% coverage. You can copy the structure from other testcases. Please remember to adapt the docstrings.

Further reading

For further best practices and hints on contributing please see the contribution-guide. Should there be any contradictions between this guide and the linked one, please stick to this text. Aside from that feel free to create an issue or write an email if anything is unclear.

Thank you for your contribution!

2.3.3 Repository

Last Reviewed
2023-09-05

This project uses sub-projects and retrieves its versioning information from version control. Therefore it is crucial that you follow these rules when working with the project (e.g. for packaging a development version).

- Make sure that the git project is complete. A call to git status for example should succeed.
- Make sure that you have checked out all available sub-projects.
- Proper initialization of submodules can be ensured with `git submodule update --init --recursive`

2.3.4 Changelog

202x-xx-xx - Version 3.x - “”

changes

contributors

2023-12-17 - Version 3.1 - “Rubric Of Ruin”

This is the minor release of the new version 3.1. It adds a modification of the API of the qr-method, hence the minor release.

changes

- extend API of the qr-method to allow passing image parameters in non-native mode
- use version 0.15 and upwards of python-barcode
- fix an issue in the config provider that prevented config files to be found when only a path was supplied
- Improve type annotations, usage of six and other packaging relevant parts.

contributors

- Patrick Kanzler
- Alexandre Detiste
- tuxmaster
- belono

2023-11-17 - Version 3.0 - "Quietly Confident"

This is the major release of the new version 3.0. A big thank you to @belono for their many contributions for the finalization of v3!

The third major release of this library drops support for Python 2 and requires a Python version of at least 3.8. The API has been reworked to be more consistent and two new concepts have been introduced:

Capabilities allow the library to know which features the currently used printer implements and send fitting commands. *Magic Encode* is a new feature that uses the capability information and encodes Unicode automatically with the correct code page while sending also the code page change commands.

The license of the project has been changed to MIT in accordance with its contributors.

The changes listed here are a summary of the changes of the previous alpha releases. For details please read the changelog of the alpha releases.

changes

- change the project's license to MIT in accordance with the contributors (see python-escpos/python-escpos#171)
- feature: add "capabilities" which are shared with escpos-php, capabilities are stored in escpos-printer-db
- feature: the driver tries now to guess the appropriate codepage and sets it automatically (called "magic encode")
- as an alternative you can force the codepage with the old API
- fix the encoding search so that lower encodings are found first
- automatically handle cases where full cut or partial cut is not available
- refactor of the set-method
- preliminary support of POS "line display" printing
- add support for software-based barcode-rendering
- make feed for cut optional
- implemented paper sensor querying command
- Include support for CUPS based printer interfaces
- add Win32Raw-Printer on Windows-platforms
- add and improve Windows support of USB-class
- pickle capabilities for faster startup

contributors

This is a list of contributors since the last v2 release.

- Ahmed Tahri
- akeonly
- Alexander Bougakov
- AlexandroJaez
- Asuki Kono
- belono

- brendanhowell
- Brian
- Christoph Heuel
- csoft2k
- Curtis // mashedkeyboard
- Dmytro Katyukha
- Foaly
- Gerard Marull-Paretas
- Justin Vieira
- kedare
- kennedy
- Lucy Linder
- Maximilian Wagenbach
- Michael Billington
- Michael Elsdörfer
- mrwunderbar666
- NullYing
- Omer Akram
- Patrick Kanzler
- primax79
- Ramon Poca
- reck31
- Romain Porte
- Sam Cheng
- Scott Rotondo
- Sergio Pulgarin
- Thijs Triemstra
- Yaisel Hurtado
- ysuolmai

and others

2023-05-11 - Version 3.0a9 - "Pride Comes Before A Fall"

This release is the 10th alpha release of the new version 3.0. After three years hiatus, a new release is in work in order to finally get a version 3.0 out.

changes

- Include support for CUPS based printer interfaces
- Move the build tool chain to GitHub

contributors

- belono
- brendanhowell
- AlexandroJaez
- NullYing
- kedare
- Foaly
- patkan
- and others

2020-05-12 - Version 3.0a8 - "Only Slightly Bent"

This release is the ninth alpha release of the new version 3.0. Please be aware that the API is subject to change until v3.0 is released.

This release drops support for Python 2, requiring at least version 3.5 of Python.

changes

- Drop support for Python 2 and mark in setuptools as only supporting 3.5 and upwards
- remove landscape.io badge
- replace viivakoodi with python-barcode which is maintained
- add configuration for Visual Studio Code
- use pkg_resources for the retrieval of the capabilities.json

contributors

- Romain Porte
- Patrick Kanzler

2020-05-09 - Version 3.0a7 - "No Fixed Abode"

This release is the eight alpha release of the new version 3.0. Please be aware that the API is subject to change until v3.0 is released.

This release also marks the point at which the project transitioned to having only a master-branch (and not an additional development branch).

changes

- add Exception for NotImplementedError in detach_kernel_driver
- update installation information
- update and improve documentation
- add error handling to image centering flag
- update and fix tox and CI environment, preparing drop of support for Python 2

contributors

- Alexander Bougakov
- Brian
- Yaisel Hurtado
- Maximilian Wagenbach
- Patrick Kanzler

2019-06-19 - Version 3.0a6 - "Mistake not..."

This release is the seventh alpha release of the new version 3.0. Please be aware that the API is subject to change until v3.0 is released.

changes

- fix inclusion of the capabilities-file
- execute CI jobs also on Windows and MacOS-targets
- improve documentation

contributors

- Patrick Kanzler

2019-06-16 - Version 3.0a5 - "Lightly Seared On The Reality Grill"

This release is the sixth alpha release of the new version 3.0. Please be aware that the API is subject to change until v3.0 is released.

changes

- allow arbitrary USB arguments in USB-class
- add Win32Raw-Printer on Windows-platforms
- add and improve Windows support of USB-class
- use pyyaml safe_load()
- improve doc
- implement _read method of Network printer class

contributors

- Patrick Kanzler
- Gerard Marull-Paretas
- Ramon Poca
- akeonly
- Omer Akram
- Justin Vieira

2018-05-15 - Version 3.0a4 - "Kakistocrat"

This release is the fifth alpha release of the new version 3.0. Please be aware that the API will still change until v3.0 is released.

changes

- raise exception when TypeError occurs in cashdraw (#268)
- Feature/clear content in dummy printer (#271)
- fix is_online() (#282)
- improve documentation
- Modified submodule to always pull from master branch (#283)
- parameter for implementation of nonnative qrcode (#289)
- improve platform independence (#296)

contributors

- Christoph Heuel
- Patrick Kanzler
- kennedy
- primax79
- reck31
- Thijs Triemstra

2017-10-08 - Version 3.0a3 - "Just Testing"

This release is the fourth alpha release of the new version 3.0. Please be aware that the API will still change until v3.0 is released.

changes

- minor changes in documentation, tests and examples
- pickle capabilities for faster startup
- first implementation of centering images and QR
- check barcodes based on regex

contributors

- Patrick Kanzler
- Lucy Linder
- Romain Porte
- Sergio Pulgarin

2017-08-04 - Version 3.0a2 - "It's My Party And I'll Sing If I Want To"

This release is the third alpha release of the new version 3.0. Please be aware that the API will still change until v3.0 is released.

changes

- refactor of the set-method
- preliminary support of POS "line display" printing
- improvement of tests
- added ImageWidthError
- list authors in repository
- add support for software-based barcode-rendering

- fix SerialException when trying to close device on __del__
- added the DLE EOT querying command for USB and Serial
- ensure QR codes have a large enough border
- make feed for cut optional
- fix the behavior of horizontal tabs
- added test script for hard an soft barcodes
- implemented paper sensor querying command
- added weather forecast example script
- added a method for simpler newlines

contributors

- csoft2k
- Patrick Kanzler
- mrwunderbar666
- Romain Porte
- Ahmed Tahri

2017-03-29 - Version 3.0a1 - "Headcrash"

This release is the second alpha release of the new version 3.0. Please be aware that the API will still change until v3.0 is released.

changes

- automatically upload releases to GitHub
- add environment variable ESCPOS_CAPABILITIES_FILE
- automatically handle cases where full cut or partial cut is not available
- add print_and_feed

contributors

- Sam Cheng
- Patrick Kanzler
- Dmytro Katyukha

2017-01-31 - Version 3.0a - "Grey Area"

This release is the first alpha release of the new version 3.0. Please be aware that the API will still change until v3.0 is released.

changes

- change the project's license to MIT in accordance with the contributors (see python-escpos/python-escpos#171)
- feature: add "capabilities" which are shared with escpos-php, capabilities are stored in escpos-printer-db
- feature: the driver tries now to guess the appropriate codepage and sets it automatically (called "magic encode")
- as an alternative you can force the codepage with the old API
- updated and improved documentation
- changed constructor of main class due to introduction of capabilities
- changed interface of method *blocktext*, changed behavior of multiple methods, for details refer to the documentation on python-escpos.readthedocs.io
- add support for custom cash drawer sequence
- enforce flake8 on the src-files, test py36 and py37 on travis

contributors

- Michael Billington
- Michael Elsdörfer
- Patrick Kanzler (with code by Frédéric Van der Essen)
- Asuki Kono
- Benito López
- Curtis // mashedkeyboard
- Thijs Triemstra
- ysuolmai

2016-08-26 - Version 2.2.0 - "Fate Amenable To Change"

changes

- fix improper API-use in qrcode()
- change setup.py shebang to make it compatible with virtualenvs.
- add constants for sheet mode and colors
- support changing the line spacing

contributors

- Michael Elsdörfer
- Patrick Kanzler

2016-08-10 - Version 2.1.3 - "Ethics Gradient"

changes

- configure readthedocs and travis
- update doc with hint on image preprocessing
- add fix for printing large images (by splitting them into multiple images)

contributors

- Patrick Kanzler

2016-08-02 - Version 2.1.2 - "Death and Gravity"

changes

- fix File-printer: flush after every call of _raw()
- fix lists in documentation
- fix CODE128: by adding the control character to the barcode-selection-sequence the barcode became unusable

contributors

- Patrick Kanzler

2016-08-02 - Version 2.1.1 - "Contents May Differ"

changes

- rename variable interface in USB-class to timeout
- add support for hypothesis and move pypy3 to the allowed failures (pypy3 is not supported by hypothesis)

contributors

- Patrick Kanzler
- Renato Lorenzi

2016-07-23 - Version 2.1.0 - "But Who's Counting?"

changes

- packaging: configured the coverage-analysis codecov.io
- GitHub: improved issues-template
- documentation: add troubleshooting tip to network-interface
- the module, CLI and documentation is now aware of the version of python-escpos
- the CLI does now support basic tab completion

contributors

- Patrick Kanzler

2016-06-24 - Version 2.0.0 - "Attitude Adjuster"

This version is based on the original version of python-escpos by Manuel F Martinez. However, many contributions have greatly improved the old codebase. Since this version does not completely match the interface of the version published on PyPi and has many improvements, it will be released as version 2.0.0.

changes

- refactor complete code in order to be compatible with Python 2 and 3
- modernize packaging
- add testing and CI
- merge various forks into codebase, fixing multiple issues with barcode-, QR-printing, cash-draw and structure
- improve the documentation
- extend support of barcode-codes to type B
- add function to disable panel-buttons
- the text-functions are now intended for unicode, the driver will automatically encode the string based on the selected codepage
- the image-functions are now much more flexible
- added a CLI
- restructured the constants

contributors

- Thomas van den Berg
- Michael Billington
- Nate Bookham
- Davis Goglin
- Christoph Heuel
- Patrick Kanzler
- Qian LinFeng

2016-01-24 - Version 1.0.9

- fix constant definition for PC1252
- move documentation to Sphinx

2015-10-27 - Version 1.0.8

- **Merge pull request #59 from zouppen/master**
  - Support for images vertically longer than 256 pixels
  - Sent by Joel Lehtonen <joel.lehtonen@koodilehto.fi>
- Updated README

2015-08-22 - Version 1.0.7

- Issue #57: Fixed transparent images

2015-07-06 - Version 1.0.6

- **Merge pull request #53 from ldos/master**
  - Extended params for serial printers
  - Sent by ldos <cafeteria.ldosalzira@gmail.com>

2015-04-21 - Version 1.0.5

- **Merge pull request #45 from Krispy2009/master**
  - Raising the right error when wrong charcode is used
  - Sent by Kristi <Krispy2009@gmail.com>

2014-05-20 - Version 1.0.4

- Issue #20: Added Density support (Sent by thomas.erbacher@ragapack.de)
- Added charcode tables
- Fixed Horizontal Tab
- Fixed code tabulators

2014-02-23 - Version 1.0.3

- Issue #18: Added quad-area characters (Sent by syncman1x@gmail.com)
- Added exception for PIL import

2013-12-30 - Version 1.0.2

- Issue #5: Fixed vertical tab
- Issue #9: Fixed indentation inconsistency

2013-03-14 - Version 1.0.1

- Issue #8: Fixed set font
- Added QR support

2012-11-15 - Version 1.0

- Issue #2: Added Ethernet support
- Issue #3: Added compatibility with libusb-1.0.1
- Issue #4: Fixed typo in escpos.py

2.3.5 TODO

Last Reviewed
2023-08-10

Open points and issues of the project are tracked in the GitHub issues. Some annotations still remain in the code and should be moved over time into the issue tracker.

Todos in the codebase

Todo: Resolve the encoding alias.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/python-escpos/checkouts/latest/src/escpos/codepages.py:docstring of escpos.codepages.CodePageManager.get_encoding_name, line 3.)

Todo: Add a method to compute the checksum for the different standards

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/python-escpos/checkouts/latest/src/escpos/escpos.py:docstring of escpos.escpos.Escpos.check_barcode, line 11.)

Todo: For fixed-length standards with mandatory checksum (EAN, UPC), compute and add the checksum automatically if missing.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/python-escpos/checkouts/latest/src/escpos/escpos.py:docstring of escpos.escpos.Escpos.check_barcode, line 13.)

Todo: Support encoding aliases: pc437 instead of cp437.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/python-escpos/checkouts/latest/src/escpos/magicencode.py:docstring of escpos.magicencode.Encoder.get_encoding_name, line 7.)

Todo: Add a method to compute the checksum for the different standards

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/python-escpos/checkouts/latest/src/escpos/escpos.py:docstring of escpos.escpos.Escpos.check_barcode, line 11.)

Todo: For fixed-length standards with mandatory checksum (EAN, UPC), compute and add the checksum automatically if missing.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/python-escpos/checkouts/latest/src/escpos/escpos.py:docstring of escpos.escpos.Escpos.check_barcode, line 13.)

2.4 API Documentation

This chapter contains an auto-generated documentation of the API of this library.

2.4.1 Esc/Pos

Module *escpos.escpos* Main class.

This module contains the abstract base class *Escpos*.

author
python-escpos developers

organization
Bashlinux and python-escpos

copyright
Copyright (c) 2012-2017 Bashlinux and python-escpos

license
MIT

class escpos.escpos.**Escpos**(*profile=None*, *magic_encode_args=None*, ***kwargs*)

Bases: object

ESC/POS Printer object.

This class is the abstract base class for an Esc/Pos-printer. The printer implementations are children of this class.

property device: Literal[None] | object

Implements a self-open mechanism.

An attempt to get the property before open the connection will cause the connection to open.

open()

Open a printer device/connection.

close()

Close a printer device/connection.

image(*img_source*, *high_density_vertical=True*, *high_density_horizontal=True*, *impl='bitImageRaster'*, *fragment_height=960*, *center=False*)

Print an image.

You can select whether the printer should print in high density or not. The default value is high density. When printing in low density, the image will be stretched.

Esc/Pos supplies several commands for printing. This function supports three of them. Please try to vary the implementations if you have any problems. For example the printer *IT80-002* will have trouble aligning images that are not printed in Column-mode.

The available printing implementations are:

- *bitImageRaster*: prints with the *GS v 0*-command
- *graphics*: prints with the *GS (L*-command
- *bitImageColumn*: prints with the *ESC **-command

When trying to center an image make sure you have initialized the printer with a valid profile, that contains a media width pixel field. Otherwise the centering will have no effect.

Parameters

- **img_source** – PIL image or filename to load: *jpg*, *gif*, *png* or *bmp*
- **high_density_vertical** (bool) – print in high density in vertical direction *default:* True

- **high_density_horizontal** (bool) – print in high density in horizontal direction *default:* True
- **impl** (str) – choose image printing mode between *bitImageRaster*, *graphics* or *bitImageColumn*
- **fragment_height** (int) – Images larger than this will be split into multiple fragments *default:* 960
- **center** (bool) – Center image horizontally *default:* False

Return type
None

qr(*content*, *ec=0*, *size=3*, *model=2*, *native=False*, *center=False*, *impl=None*, *image_arguments=None*)

Print QR Code for the provided string.

Parameters

- **content** – The content of the code. Numeric data will be more efficiently compacted.
- **ec** – Error-correction level to use. One of QR_ECLEVEL_L (default), QR_ECLEVEL_M, QR_ECLEVEL_Q or QR_ECLEVEL_H. Higher error correction results in a less compact code.
- **size** – Pixel size to use. Must be 1-16 (default 3)
- **model** – QR code model to use. Must be one of QR_MODEL_1, QR_MODEL_2 (default) or QR_MICRO (not supported by all printers).
- **native** – True to render the code on the printer, False to render the code as an image and send it to the printer (Default)
- **center** – Centers the code *default:* False
- **impl** – Image-printing-implementation, refer to *image()* for details
- **image_arguments** (Optional[dict]) – arguments passed to *image()*. Replaces *impl* and *center*. If *impl* or *center* are set, they will overwrite *image_arguments*.

Return type
None

charcode(*code='AUTO'*)

Set Character Code Table.

Sets the control sequence from CHARCODE in *escpos.constants* as active. It will be sent with the next text sequence. If you set the variable code to AUTO it will try to automatically guess the right codepage. (This is the standard behavior.)

Parameters
code (str) – Name of CharCode

Raises
CharCodeError

Return type
None

static check_barcode(*bc*, *code*)

Check if barcode is OK.

This method checks if the barcode is in the proper format. The validation concerns the barcode length and the set of characters, but won't compute/validate any checksum. The full set of requirement for each barcode type is available in the ESC/POS documentation.

As an example, using EAN13, the barcode *12345678901* will be correct, because it can be rendered by the printer. But it does not suit the EAN13 standard, because the checksum digit is missing. Adding a wrong checksum in the end will also be considered correct, but adding a letter won't (EAN13 is numeric only).

Todo: Add a method to compute the checksum for the different standards

---

Todo: For fixed-length standards with mandatory checksum (EAN, UPC), compute and add the checksum automatically if missing.

Parameters

- **bc** (str) – barcode format, see *barcode()*
- **code** (str) – alphanumeric data to be printed as bar code, see *barcode()*

Returns
bool

barcode (*code*, *bc*, *height=64*, *width=3*, *pos='BELOW'*, *font='A'*, *align_ct=True*, *function_type=None*, *check=True*, *force_software=False*)

Print barcode.

Automatic hardware|software barcode renderer according to the printer capabilities.

Defaults to hardware barcode and its format types if supported. Automatically switches to software barcode renderer if hardware does not support a barcode type that is supported by software. (e.g. JAN, ISSN, etc.).

Set force_software=True to force the software renderer according to the profile. Set force_software=graphics|bitImageColumn|bitImageRaster to specify a renderer.

Ignores caps, special chars and whitespaces in barcode type names. So "EAN13", "ean-13", "Ean_13", "EAN 13" are all accepted.

Parameters

- **code** – alphanumeric data to be printed as bar code (payload).
- **bc** – barcode format type (EAN13, CODE128, JAN, etc.).
- **height** (int) – barcode module height (in printer dots), has to be between 1 and 255. *default*: 64
- **width** (int) – barcode module width (in printer dots), has to be between 2 and 6. *default*: 3
- **pos** (str) – text position (ABOVE, BELOW, BOTH, OFF) relative to the barcode (ignored in software renderer). *default*: BELOW
- **font** (str) – select font A or B (ignored in software renderer). *default*: A
- **align_ct** (bool) – If *True*, center the barcode. *default*: True
- **function_type** – ESCPOS function type A or B. None to guess it from profile (ignored in software renderer). *default*: None

- **check** (bool) – If *True*, checks that the code meets the requirements of the barcode type. *default*: True
- **force_software** (Union[bool, str]) – If *True*, force the use of software barcode renderer from profile. If *"graphics", "bitImageColumn" or "bitImageRaster"*, force the use of specific renderer.

Raises
BarcodeCodeError, *BarcodeTypeError*

Return type
None

Note:

Get all supported formats at:

- Hardware: *BARCODE_FORMATS*
- Software: Python barcode documentation

---

text (*txt*)

Print alpha-numeric text.

The text has to be encoded in the currently selected codepage. The input text has to be encoded in unicode.

Parameters
txt (str) – text to be printed

Raises
TextError

Return type
None

textln (*txt=''*)

Print alpha-numeric text with a newline.

The text has to be encoded in the currently selected codepage. The input text has to be encoded in unicode.

Parameters
txt (str) – text to be printed with a newline

Raises
TextError

Return type
None

ln (*count=1*)

Print a newline or more.

Parameters
count (int) – number of newlines to print

Raises
ValueError if count < 0

Return type
None

block_text(*txt*, *font='0'*, *columns=None*)

Print text wrapped to specific columns.

Text has to be encoded in unicode.

Parameters

- **txt** – text to be printed
- **font** – font to be used, can be `a` or `b`
- **columns** – amount of columns

Return type
`None`

Returns
None

set(*align=None*, *font=None*, *bold=None*, *underline=None*, *width=None*, *height=None*, *density=None*, *invert=None*, *smooth=None*, *flip=None*, *normal_textsize=None*, *double_width=None*, *double_height=None*, *custom_size=None*)

Set text properties by sending them to the printer.

If a value for a parameter is not supplied, nothing is sent for this type of format.

Parameters

- **align** (`Optional[str]`) – horizontal position for text, possible values are:
  - 'center'
  - 'left'
  - 'right'
- **font** (`Optional[str]`) – font given as an index, a name, or one of the special values 'a' or 'b', referring to fonts 0 and 1.
- **bold** (`Optional[bool]`) – text in bold
- **underline** (`Optional[int]`) – underline mode for text, decimal range 0-2
- **normal_textsize** (`Optional[bool]`) – switch to normal text size if True
- **double_height** (`Optional[bool]`) – doubles the height of the text
- **double_width** (`Optional[bool]`) – doubles the width of the text
- **custom_size** (`Optional[bool]`) – uses custom size specified by width and height parameters. Cannot be used with double_width or double_height.
- **width** (`Optional[int]`) – text width multiplier when custom_size is used, decimal range 1-8
- **height** (`Optional[int]`) – text height multiplier when custom_size is used, decimal range 1-8
- **density** (`Optional[int]`) – print density, value from 0-8, if something else is supplied the density remains unchanged
- **invert** (`Optional[bool]`) – True enables white on black printing
- **smooth** (`Optional[bool]`) – True enables text smoothing. Effective on 4x4 size text and larger
- **flip** (`Optional[bool]`) – True enables upside-down printing

Return type
None

set_with_default (*align='left'*, *font='a'*, *bold=False*, *underline=0*, *width=1*, *height=1*, *density=9*, *invert=False*, *smooth=False*, *flip=False*, *double_width=False*, *double_height=False*, *custom_size=False*)

Set default text properties by sending them to the printer.

This function has the behavior of the *set()*-method from before version 3. If a parameter to this method is not supplied, a default value will be sent. Otherwise this method forwards the values to the `escpos.Escpos.set()`.

Parameters

- **align** (`Optional[str]`) – horizontal position for text, possible values are:
  - 'center'
  - 'left'
  - 'right'

  default: 'left'
- **font** (`Optional[str]`) – font given as an index, a name, or one of the special values 'a' or 'b', referring to fonts 0 and 1.
- **bold** (`Optional[bool]`) – text in bold, *default*: False
- **underline** (`Optional[int]`) – underline mode for text, decimal range 0-2, *default*: 0
- **double_height** (`Optional[bool]`) – doubles the height of the text
- **double_width** (`Optional[bool]`) – doubles the width of the text
- **custom_size** (`Optional[bool]`) – uses custom size specified by width and height parameters. Cannot be used with double_width or double_height.
- **width** (`Optional[int]`) – text width multiplier when custom_size is used, decimal range 1-8, *default*: 1
- **height** (`Optional[int]`) – text height multiplier when custom_size is used, decimal range 1-8, *default*: 1
- **density** (`Optional[int]`) – print density, value from 0-8, if something else is supplied the density remains unchanged
- **invert** (`Optional[bool]`) – True enables white on black printing, *default*: False
- **smooth** (`Optional[bool]`) – True enables text smoothing. Effective on 4x4 size text and larger, *default*: False
- **flip** (`Optional[bool]`) – True enables upside-down printing, *default*: False

Return type
None

line_spacing (*spacing=None*, *divisor=180*)

Set line character spacing.

If no spacing is given, we reset it to the default.

There are different commands for setting the line spacing, using a different denominator:

'+" line_spacing/360 of an inch, 0 <= line_spacing <= 255 '3' line_spacing/180 of an inch, 0 <= line_spacing <= 255 'A' line_spacing/60 of an inch, 0 <= line_spacing <= 85

Some printers may not support all of them. The most commonly available command (using a divisor of 180) is chosen.

Return type
None

cut(*mode='FULL'*, *feed=True*)

Cut paper.

Without any arguments the paper will be cut completely. With 'mode=PART' a partial cut will be attempted. Note however, that not all models can do a partial cut. See the documentation of your printer for details.

Parameters

- **mode** (str) – set to 'PART' for a partial cut. default: 'FULL'
- **feed** (bool) – print and feed before cutting. default: true

Raises
ValueError – if mode not in ('FULL', 'PART')

Return type
None

cashdraw(*pin*)

Send pulse to kick the cash drawer.

Kick cash drawer on pin 2 (*CD_KICK_2*) or pin 5 (*CD_KICK_5*) according to the default parameters. For non default parameters send a decimal sequence i.e. [27,112,48] or [27,112,0,25,255].

Parameters
pin – pin number, 2 or 5 or list of decimals

Raises
CashDrawerError

Return type
None

linedisplay_select(*select_display=False*)

Select the line display or the printer.

This method is used for line displays that are daisy-chained between your computer and printer. If you set *select_display* to true, only the display is selected and if you set it to false, only the printer is selected.

Parameters
select_display (bool) – whether the display should be selected or the printer

Return type
None

linedisplay_clear()

Clear the line display and resets the .

This method is used for line displays that are daisy-chained between your computer and printer.

Return type
None

linedisplay(*text*)

Display text on a line display connected to your printer.

You should connect a line display to your printer. You can do this by daisy-chaining the display between your computer and printer.

Parameters
text (str) – Text to display

Return type
None

hw(*hw*)

Hardware operations.

Parameters
hw (str) – hardware action, may be:

- INIT
- SELECT
- RESET

Return type
None

print_and_feed(*n=1*)

Print data in print buffer and feed *n* lines.

If n not in range (0, 255) then a ValueError will be raised.

Parameters
n (int) – number of n to feed. 0 <= n <= 255. default: 1

Raises
ValueError – if not 0 <= n <= 255

Return type
None

control(*ctl*, *count=5*, *tab_size=8*)

Feed control sequences.

Parameters

- **ctl** (str) – string for the following control sequences:
  - LF *for Line Feed*
  - FF *for Form Feed*
  - CR *for Carriage Return*
  - HT *for Horizontal Tab*
  - VT *for Vertical Tab*
- **count** (int) – integer between 1 and 32, controls the horizontal tab count. Defaults to 5.
- **tab_size** (int) – integer between 1 and 255, controls the horizontal tab size in characters. Defaults to 8

Raises
TabPosError

Return type
None

panel_buttons(*enable=True*)

Control the panel buttons on the printer (e.g. FEED).

When enable is set to False the panel buttons on the printer will be disabled. Calling the method with *enable=True* or without argument will enable the panel buttons.

If panel buttons are enabled, the function of the panel button, such as feeding, will be executed upon pressing the button. If the panel buttons are disabled, pressing them will not have any effect.

This command is effective until the printer is initialized, resetted or power-cycled. The default is enabled panel buttons.

Some panel buttons will always work, especially when the printer is opened. See for more information the manual of your printer and the escpos-command-reference.

Parameters
enable (`bool`) – controls the panel buttons

Return type
`None`

query_status(*mode*)

Query the printer for its status.

Returns byte array containing it.

Parameters
mode (`bytes`) – Integer that sets the status mode queried to the printer. - RT_STATUS_ONLINE: Printer status. - RT_STATUS_PAPER: Paper sensor.

Return type
`bytes`

is_online()

Query the online status of the printer.

Return type
`bool`

Returns
When online, returns `True`; `False` otherwise.

paper_status()

Query the paper status of the printer.

Returns 2 if there is plenty of paper, 1 if the paper has arrived to the near-end sensor and 0 if there is no paper.

Return type
`int`

Returns
2: Paper is adequate. 1: Paper ending. 0: No paper.

target(*type='ROLL'*)

Select where to print to.

Print to the thermal printer by default (ROLL) or print to the slip dot matrix printer if supported (SLIP)

Return type
`None`

eject_slip()

Eject the slip/cheque.

Return type
None

print_and_eject_slip()

Print and eject.

Prints data from the buffer to the slip station and if the paper sensor is covered, reverses the slip out the front of the printer far enough to be accessible to the operator. The impact station opens the platen in all cases.

Return type
None

use_slip_only()

Select the Slip Station for all functions.

The receipt station is the default setting after the printer is initialized or the Clear Printer (0x10) command is received

Return type
None

buzzer(*times=2*, *duration=4*)

Activate the internal printer buzzer on supported printers.

The 'times' parameter refers to the 'n' escpos command parameter, which means how many times the buzzer will be 'beeped'.

Parameters

- **times** (int) – Integer between 1 and 9, indicates the buzzer beeps.
- **duration** (int) – Integer between 1 and 9, indicates the beep duration.

Return type
None

Returns
None

class escpos.escpos.**EscposIO**(*printer*, *autocut=True*, *autoclose=True*, ***kwargs*)

Bases: object

ESC/POS Printer IO object.

Allows the class to be used together with the *with*-statement. You have to define a printer instance and assign it to the EscposIO class. This example explains the usage:

```
with EscposIO(printer.Serial('/dev/ttyUSB0')) as p:
    p.set(font='a', height=2, align='center', text_type='bold')
    p.printer.set(align='left')
    p.printer.image('logo.gif')
    p.writelines('Big line\\n', font='b')
    p.writelines('Привет')
    p.writelines('BIG TEXT', width=2)
```

After the *with*-statement the printer automatically cuts the paper if *autocut* is *True*.

set(***kwargs*)

Set the printer-parameters.

Controls which parameters will be passed to *Escpos.set()*. For more information on the parameters see the *set()*-methods documentation. These parameters can also be passed with this class' constructor or the *writelines()*-method.

Parameters
kwargs – keyword-parameters that will be passed to *Escpos.set()*

Return type
None

writelines(*text*, ***kwargs*)

Print text.

Return type
None

close()

Close printer.

Called upon closing the *with*-statement.

Return type
None

2.4.2 Printer implementations

Module *escpos.printer* printer implementations.

class escpos.printer.**Usb**(*idVendor=None*, *idProduct=None*, *usb_args={}*, *timeout=0*, *in_ep=130*, *out_ep=1*, **args*, ***kwargs*)

Bases: *Escpos*

USB printer.

This class describes a printer that natively speaks USB.

inheritance:

static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

open(*raise_not_found=True*)

Search device on USB tree and set it as escpos device.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Raises
USBNotFoundError

Return type
None

close()

Release USB interface.

Return type
None

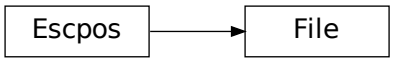
class escpos.printer.**File**(*devfile=''*, *auto_flush=True*, **args*, ***kwargs*)

Bases: *Escpos*

Generic file printer.

This class is used for parallel port printer or other printers that are directly attached to the filesystem. Note that you should stay away from using USB-to-Parallel-Adapter since they are unreliable and produce arbitrary errors.

inheritance:



static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

open(*raise_not_found=True*)

Open system file.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Return type
None

flush()

Flush printing content.

Return type
None

close()

Close system file.

Return type
None

class escpos.printer.**Network**(*host=''*, *port=9100*, *timeout=60*, **args*, ***kwargs*)

Bases: *Escpos*

Network printer.

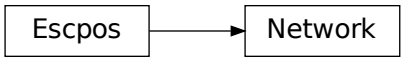
This class is used to attach to a networked printer. You can also use this in order to attach to a printer that is forwarded with `socat`.

If you have a local printer on parallel port `/dev/usb/lp0` then you could start `socat` with:

```
socat -u TCP4-LISTEN:4242,reuseaddr,fork OPEN:/dev/usb/lp0
```

Then you should be able to attach to port `4242` with this class. Otherwise the normal use case would be to have a printer with Ethernet interface. This type of printer should work the same with this class. For the address of the printer check its manuals.

inheritance:



static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

open(*raise_not_found=True*)

Open TCP socket with `socket`-library and set it as escpos device.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Return type
None

close()

Close TCP connection.

Return type
None

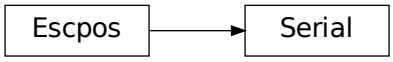
class escpos.printer.**Serial**(*devfile=''*, *baudrate=9600*, *bytesize=8*, *timeout=1*, *parity=None*, *stopbits=None*, *xonxoff=False*, *dsrdtr=True*, **args*, ***kwargs*)

Bases: *Escpos*

Serial printer.

This class describes a printer that is connected by serial interface.

inheritance:



static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

open(*raise_not_found=True*)

Set up serial port and set is as escpos device.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Return type
None

close()

Close Serial interface.

Return type
None

class escpos.printer.**LP**(*printer_name=''*, **args*, ***kwargs*)

Bases: *Escpos*

Simple UNIX lp command raw printing.

Thanks to Oyami-Srk comment.

inheritance:

static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

property printers: dict

Available CUPS printers.

open(*job_name='python-escpos'*, *raise_not_found=True*, *_close_opened=True*)

Invoke _lp_ in a new subprocess and wait for commands.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Return type
None

close()

Stop the subprocess.

Return type
None

flush()

End line and wait for new commands.

Return type
None

class escpos.printer.**Dummy**(**args*, ***kwargs*)

Bases: *Escpos*

Dummy printer.

This class is used for saving commands to a variable, for use in situations where there is no need to send commands to an actual printer. This includes generating print jobs for later use, or testing output.

inheritance:

static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

property output: bytes

Get the data that was sent to this printer.

clear()

Clear the buffer of the printer.

This method can be called if you send the contents to a physical printer and want to use the Dummy printer for new output.

Return type
None

close()

Close not implemented for Dummy printer.

Return type
None

class escpos.printer.**CupsPrinter**(*printer_name=''*, **args*, ***kwargs*)

Bases: *Escpos*

Simple CUPS printer connector.

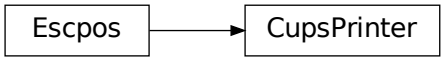
Note:

Requires `pycups` which in turn needs the cups development library package:

- Ubuntu/Debian: `libcups2-dev`
- OpenSuse/Fedora: `cups-devel`

---

inheritance:



static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

property printers: dict

Available CUPS printers.

open(*job_name='python-escpos'*, *raise_not_found=True*)

Set up a new print job and target the printer.

A call to this method is required to send new jobs to the CUPS connection after close.

Defaults to default CUPS printer. Creates a new temporary file buffer.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Return type
None

send()

Send the print job to the printer.

Return type
None

close()

Close CUPS connection.

Send pending job to the printer if needed.

Return type
None

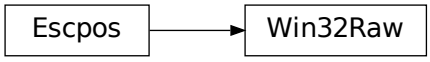
class escpos.printer.**Win32Raw**(*printer_name=''*, **args*, ***kwargs*)

Bases: *Escpos*

Printer binding for win32 API.

Uses the module pywin32 for printing.

inheritance:



static is_usable()

Indicate whether this printer class is usable.

Will return True if dependencies are available. Will return False if not.

Return type
bool

property printers: dict

Available Windows printers.

open(*job_name='python-escpos'*, *raise_not_found=True*)

Open connection to default printer.

By default raise an exception if device is not found.

Parameters
raise_not_found (bool) – Default True. False to log error but do not raise exception.

Raises
DeviceNotFoundError

Return type
None

close()

Close connection to default printer.

Return type
None

2.4.3 Constants

Module *escpos.constants* Set of ESC/POS Commands (Constants)

This module contains constants that are described in the Esc/Pos-documentation. Since there is no definitive and unified specification for all Esc/Pos-like printers the constants could later be moved to *capabilities* as in escpos-php by @mike42.

author
python-escpos developers

organization
Bashlinux and python-escpos

copyright
Copyright (c) 2012-2017 Bashlinux and python-escpos

license
MIT

escpos.constants.**CTL_LF: bytes = b'\n'**

Print and line feed

escpos.constants.**CTL_FF: bytes = b'\x0c'**

Form feed

escpos.constants.**CTL_CR: bytes = b'\r'**

Carriage return

escpos.constants.**CTL_HT: bytes = b'\t'**

Horizontal tab

escpos.constants.**CTL_SET_HT: bytes = b'\x1bD'**

Set horizontal tab positions

escpos.constants.**CTL_VT: bytes = b'\x0b'**

Vertical tab

escpos.constants.**CD_KICK_DEC_SEQUENCE**(*esc*, *p*, *m*, *t1=50*, *t2=50*)

decimal cash drawer kick sequence

escpos.constants.**CD_KICK_2: bytes = b'\x1bp\x0022'**

Sends a pulse to pin 2 []

escpos.constants.**CD_KICK_5: bytes = b'\x1bp\x0122'**

Sends a pulse to pin 5 []

escpos.constants.**PAPER_FULL_CUT: bytes = b'\x1dV\x00'**

Full cut paper

escpos.constants.**PAPER_PART_CUT: bytes = b'\x1dV\x01'**

Partial cut paper

escpos.constants.**TXT_STYLE: ConstTxtStyleClass = {'align': {'center': b'\x1ba\x01', 'left': b'\x1ba\x00', 'right': b'\x1ba\x02'}, 'bold': {False: b'\x1bE\x00', True: b'\x1bE\x01'}, 'color': {'black': b'\x1br\x00', 'red': b'\x1br\x01'}, 'density': {0: b'\x1d|\x00', 1: b'\x1d|\x01', 2: b'\x1d|\x02', 3: b'\x1d|\x03', 4: b'\x1d|\x04', 5: b'\x1d|\x08', 6: b'\x1d|\x07', 7: b'\x1d|\x06', 8: b'\x1d|\x05'}, 'flip': {False: b'\x1b{\x00', True: b'\x1b{\x01'}, 'font': {'a': b'\x1bM\x00', 'b': b'\x1bM\x00'}, 'height': {1: 0, 2: 1, 3: 2, 4: 3, 5: 4, 6: 5, 7: 6, 8: 7}, 'invert': {False: b'\x1dB\x00', True: b'\x1dB\x01'}, 'size': {'2h': b'\x1b!\x00\x1b!\x10', '2w': b'\x1b!\x00\x1b! ', '2x': b'\x1b!\x00\x1b!0', 'normal': b'\x1b!\x00\x1b!\x00'}, 'smooth': {False: b'\x1db\x00', True: b'\x1db\x01'}, 'underline': {0: b'\x1b-\x00', 1: b'\x1b-\x01', 2: b'\x1b-\x02'}, 'width': {1: 0, 2: 16, 3: 32, 4: 48, 5: 64, 6: 80, 7: 96, 8: 112}}**

text style dictionary for *escpos.escpos.Escpos.set()*

escpos.constants.**SET_FONT**(*n*)

escpos.constants.**TXT_FONT_A: bytes = b'\x1bM\x00'**

Font type A

escpos.constants.**TXT_FONT_B: bytes = b'\x1bM\x01'**

Font type B

escpos.constants.**CODEPAGE_CHANGE: bytes = b'\x1bt'**

Prefix to change the codepage. You need to attach a byte to indicate the codepage to use. We use escpos-printer-db as the data source.

escpos.constants.**BARCODE_TXT_OFF: bytes = b'\x1dH\x00'**

HRI barcode chars OFF

escpos.constants.**BARCODE_TXT_ABV: bytes = b'\x1dH\x01'**

HRI barcode chars above

escpos.constants.**BARCODE_TXT_BLW: bytes = b'\x1dH\x02'**

HRI barcode chars below

escpos.constants.**BARCODE_TXT_BTH: bytes = b'\x1dH\x03'**

HRI both above and below

escpos.constants.**BARCODE_FONT_A: bytes = b'\x1df\x00'**

Font type A for HRI barcode chars

escpos.constants.**BARCODE_FONT_B: bytes = b'\x1df\x01'**

Font type B for HRI barcode chars

escpos.constants.**BARCODE_HEIGHT: bytes = b'\x1dh'**

Barcode Height [1-255]

escpos.constants.**BARCODE_WIDTH: bytes = b'\x1dw'**

Barcode Width [2-6]

escpos.constants.**BARCODE_TYPE_A: Dict[str, bytes] = {'CODABAR': b'\x1dk\x06', 'CODE39': b'\x1dk\x04', 'EAN13': b'\x1dk\x02', 'EAN8': b'\x1dk\x03', 'ITF': b'\x1dk\x05', 'NW7': b'\x1dk\x06', 'UPC-A': b'\x1dk\x00', 'UPC-E': b'\x1dk\x01'}**

Barcodes for printing function type A

escpos.constants.**BARCODE_TYPE_B: Dict[str, bytes] = {'CODABAR': b'\x1dkG', 'CODE128': b'\x1dkI', 'CODE39': b'\x1dkE', 'CODE93': b'\x1dkH', 'EAN13': b'\x1dkC', 'EAN8': b'\x1dkD', 'GS1 DATABAR EXPANDED': b'\x1dkN', 'GS1 DATABAR LIMITED': b'\x1dkM', 'GS1 DATABAR OMNIDIRECTIONAL': b'\x1dkK', 'GS1 DATABAR TRUNCATED': b'\x1dkL', 'GS1-128': b'\x1dkJ', 'ITF': b'\x1dkF', 'NW7': b'\x1dkG', 'UPC-A': b'\x1dkA', 'UPC-E': b'\x1dkB'}**

Barcodes for printing function type B The first 8 are the same barcodes as type A

escpos.constants.**BARCODE_FORMATS = {'CODABAR': ([(1, 255)], '^[A-Da-d][0-9\\$\\+\\-\\.\\/\\:]+[A-Da-d]$'), 'CODE128': ([(2, 255)], '^\\{[A-C][\\x00-\\x7F]+$'), 'CODE39': ([(1, 255)], '^([0-9A-Z \\$\\%\\+\\-\\.\\/]+|*[0-9A-Z \\$\\%\\+\\-\\.\\/]+*)$'), 'CODE93': ([(1, 255)], '^[\\x00-\\x7F]+$'), 'EAN13': ([(12, 13)], '^[0-9]{12,13}$'), 'EAN8': ([(7, 8)], '^[0-9]{7,8}$'), 'GS1 DATABAR EXPANDED': ([(2, 255)], '^\\([0-9][A-Za-z0-9 \\!\\"\\%\\&\'\\(\\)*\\+\\,\\-\\.\\/\\:\\;\\<\\=\\>\\?_\\{]+$'), 'GS1 DATABAR LIMITED': ([(13, 13)], '^[01][0-9]{12}$'), 'GS1 DATABAR OMNIDIRECTIONAL': ([(13, 13)], '^[0-9]{13}$'), 'GS1 DATABAR TRUNCATED': ([(13, 13)], '^[0-9]{13}$'), 'GS1-128': ([(2, 255)], '^\\{[A-C][\\x00-\\x7F]+$'), 'ITF': ([(2, 255)], '^([0-9]{2})+$'), 'NW7': ([(1, 255)], '^[A-Da-d][0-9\\$\\+\\-\\.\\/\\:]+[A-Da-d]$'), 'UPC-A': ([(11, 12)], '^[0-9]{11,12}$'), 'UPC-E': ([(7, 8), (11, 12)], '^([0-9]{7,8}|[0-9]{11,12})$')}**

supported barcode formats

2.4.4 Exceptions

Module *escpos.exceptions* ESC/POS Exceptions classes.

Result/Exit codes:

- *0* = success
- *10* = No Barcode type defined *BarcodeTypeError*
- *20* = Barcode size values are out of range *BarcodeSizeError*
- *30* = Barcode text not supplied *BarcodeCodeError*
- *40* = Image height is too large *ImageSizeError*
- *41* = Image width is too large *ImageWidthError*
- *50* = No string supplied to be printed *TextError*
- *60* = Invalid pin to send Cash Drawer pulse *CashDrawerError*
- *70* = Invalid number of tab positions *TabPosError*
- *80* = Invalid char code *CharCodeError*
- *90* = Device not found *DeviceNotFoundError*
- *91* = USB device not found *USBNotFoundError*
- *100* = Set variable out of range *SetVariableError*
- *200* = Configuration not found *ConfigNotFoundError*
- *210* = Configuration syntax error *ConfigSyntaxError*
- *220* = Configuration section not found *ConfigSectionMissingError*

author
python-escpos developers

organization
Bashlinux and python-escpos

copyright
Copyright (c) 2012-2017 Bashlinux and python-escpos

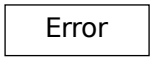
license
MIT

exception escpos.exceptions.**Error**(*msg*, *status=None*)

Bases: Exception

Base class for ESC/POS errors.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

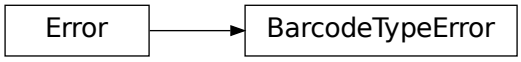
exception escpos.exceptions.**BarcodeTypeError**(*msg=''*)

Bases: *Error*

No Barcode type defined.

This exception indicates that no known barcode-type has been entered. The barcode-type has to be one of those specified in *escpos.escpos.Escpos.barcode()*. The returned error code is *10*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

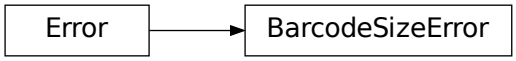
exception escpos.exceptions.**BarcodeSizeError**(*msg=''*)

Bases: *Error*

Barcode size is out of range.

This exception indicates that the values for the barcode size are out of range. The size of the barcode has to be in the range that is specified in *escpos.escpos.Escpos.barcode()*. The resulting return code is *20*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

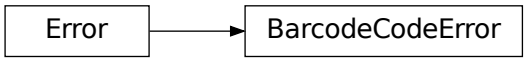
exception escpos.exceptions.**BarcodeCodeError**(*msg=''*)

Bases: *Error*

No Barcode code was supplied, or it is incorrect.

No data for the barcode has been supplied in *escpos.escpos.Escpos.barcode()* or the the *check* parameter was True and the check failed. The return code for this exception is *30*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

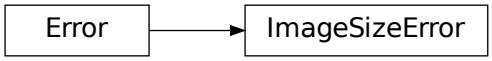
exception escpos.exceptions.**ImageSizeError**(*msg=''*)

Bases: *Error*

Image height is longer than 255px and can't be printed.

The return code for this exception is *40*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

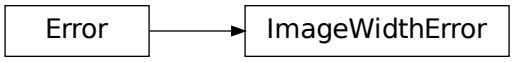
exception escpos.exceptions.**ImageWidthError**(*msg=''*)

Bases: *Error*

Image width is too large.

The return code for this exception is *41*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

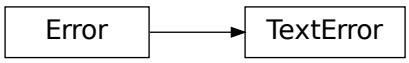
exception escpos.exceptions.**TextError**(*msg=''*)

Bases: *Error*

Text string must be supplied to the *text()* method.

This exception is raised when an empty string is passed to *escpos.escpos.Escpos.text()*. The return code for this exception is *50*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

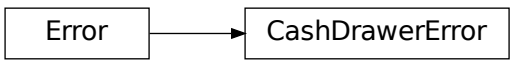
exception escpos.exceptions.**CashDrawerError**(*msg=''*)

Bases: *Error*

Valid pin must be set in order to send pulse.

A valid pin number has to be passed onto the method *escpos.escpos.Escpos.cashdraw()*. The return code for this exception is *60*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception escpos.exceptions.**TabPosError**(*msg=''*)

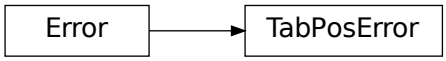
Bases: *Error*

Tab position is invalid.

Valid tab positions must be set by using from 1 to 32 tabs, and between 1 and 255 tab size values. Both values multiplied must not exceed 255, since it is the maximum tab value.

This exception is raised by *escpos.escpos.Escpos.control()*. The return code for this exception is *70*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

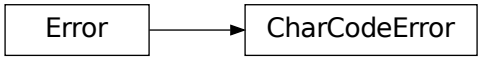
exception escpos.exceptions.**CharCodeError**(*msg=''*)

Bases: *Error*

Valid char code must be set.

The supplied charcode-name in *escpos.escpos.Escpos.charcode()* is unknown. The return code for this exception is *80*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

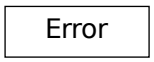
exception escpos.exceptions.**DeviceNotFoundError**(*msg=''*)

Bases: *Error*

Device was not found.

The device seems to be not accessible. The return code for this exception is *90*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

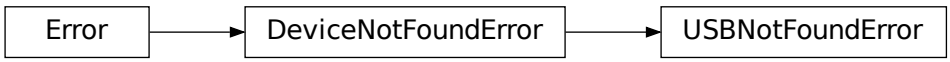
exception escpos.exceptions.**USBNotFoundError**(*msg=''*)

Bases: *DeviceNotFoundError*

USB device was not found (probably not plugged in).

The USB device seems to be not plugged in. The return code for this exception is *91*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

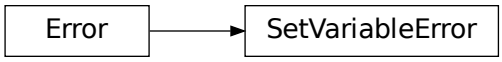
exception escpos.exceptions.**SetVariableError**(*msg=''*)

Bases: *Error*

A set method variable was out of range.

Check set variables against minimum and maximum values The return code for this exception is *100*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception escpos.exceptions.**ConfigNotFoundError**(*msg=''*)

Bases: *Error*

The configuration file was not found.

The default or passed configuration file could not be read The return code for this exception is *200*.

inheritance:

add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

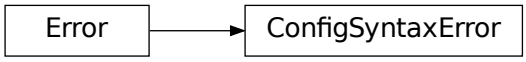
exception escpos.exceptions.**ConfigSyntaxError**(*msg=''*)

Bases: *Error*

The configuration file is invalid.

The syntax is incorrect The return code for this exception is *210*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

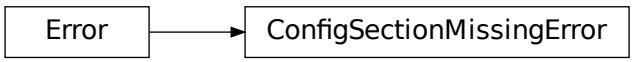
exception escpos.exceptions.**ConfigSectionMissingError**(*msg=''*)

Bases: *Error*

The configuration file is missing a section.

The part of the config asked for does not exist in the loaded configuration The return code for this exception is *220*.

inheritance:



add_note()

Exception.add_note(note) – add a note to the exception

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

2.4.5 Capabilities

Module *escpos.capabilities* Handler for capabilities data.

exception escpos.capabilities.**NotSupported**

Bases: Exception

Raised if a requested feature is not supported by the printer profile.

add_note()

Exception.add_note(note) – add a note to the exception

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class escpos.capabilities.**BaseProfile**

Bases: object

This represents a printer profile.

A printer profile knows about the number of columns, supported features, colors and more.

profile_data: Dict[str, Any] = {}

get_font(*font*)

Return the escpos index for *font*.

Makes sure that the requested *font* is valid.

Return type
int

get_columns(*font*)

Return the number of columns for the given font.

Return type
int

supports(*feature*)

Return true/false for the given feature.

Return type
bool

get_code_pages()

Return the support code pages as a {name: index} dict.

Return type
Dict[str, int]

escpos.capabilities.**get_profile**(*name=None*, ***kwargs*)

Get a profile by name.

If no name is given, return the default profile.

escpos.capabilities.**get_profile_class**(*name*)

Load a profile class.

For the given profile name, load the data from the external database, then generate dynamically a class.

Return type
Type[*BaseProfile*]

escpos.capabilities.**clean**(*s*)

Clean profile name.

Return type
str

escpos.capabilities.**ProfileBaseClass**

alias of DefaultProfile

class escpos.capabilities.**Profile**(*columns=None*, *features=None*)

Bases: DefaultProfile

Profile class for user usage.

For users, who want to provide their own profile.

get_code_pages()

Return the support code pages as a {name: index} dict.

Return type
Dict[str, int]

get_font (*font*)

Return the escpos index for *font*.

Makes sure that the requested *font* is valid.

Return type
int

```
profile_data: Dict[str, Any] = {'codePages': {'0': 'CP437', '1': 'CP932',
'11': 'CP851', '12': 'CP853', '13': 'CP857', '14': 'CP737', '15':
'ISO_8859-7', '16': 'CP1252', '17': 'CP866', '18': 'CP852', '19':
'CP858', '2': 'CP850', '20': 'Unknown', '21': 'CP874', '22': 'Unknown',
'23': 'Unknown', '24': 'Unknown', '25': 'Unknown', '254': 'Unknown',
'255': 'Unknown', '26': 'Unknown', '3': 'CP860', '30': 'TCVN-3-1', '31':
'TCVN-3-2', '32': 'CP720', '33': 'CP775', '34': 'CP855', '35': 'CP861',
'36': 'CP862', '37': 'CP864', '38': 'CP869', '39': 'ISO_8859-2', '4':
'CP863', '40': 'ISO_8859-15', '41': 'CP1098', '42': 'CP774', '43':
'CP772', '44': 'CP1125', '45': 'CP1250', '46': 'CP1251', '47': 'CP1253',
'48': 'CP1254', '49': 'CP1255', '5': 'CP865', '50': 'CP1256', '51':
'CP1257', '52': 'CP1258', '53': 'RK1048', '6': 'Unknown', '66':
'Unknown', '67': 'Unknown', '68': 'Unknown', '69': 'Unknown', '7':
'Unknown', '70': 'Unknown', '71': 'Unknown', '72': 'Unknown', '73':
'Unknown', '74': 'Unknown', '75': 'Unknown', '8': 'Unknown', '82':
'Unknown'}, 'colors': {'0': 'black'}, 'features': {'barcodeA': True,
'barcodeB': True, 'bitImageColumn': True, 'bitImageRaster': True,
'graphics': True, 'highDensity': True, 'paperFullCut': True,
'paperPartCut': True, 'pdf417Code': True, 'pulseBel': False,
'pulseStandard': True, 'qrCode': True, 'starCommands': False}, 'fonts':
{'0': {'columns': 42, 'name': 'Font A'}, '1': {'columns': 56, 'name':
'Font B'}}, 'media': {'dpi': 'Unknown', 'width': {'mm': 'Unknown',
'pixels': 'Unknown'}}, 'name': 'Default', 'notes': 'Default ESC/POS
profile, suitable for standards-compliant or Epson-branded printers. This
profile allows the use of standard ESC/POS features, and can encode a
variety of code pages.\n', 'vendor': 'Generic'}
```

supports (*feature*)

Return true/false for the given feature.

Return type
bool

get_columns (*font*)

Get column count of printer.

Return type
int

2.4.6 Config

Module *escpos.config* ESC/POS configuration manager.

This module contains the implementations of abstract base class *Config*.

class escpos.config.**Config**

Bases: object

Configuration handler class.

This class loads configuration from a default or specified directory. It can create your defined printer and return it to you.

load(*config_path=None*)

Load and parse the configuration file using pyyaml.

Parameters
config_path – An optional file path, file handle, or byte string for the configuration file.

printer()

Return a printer that was defined in the config.

Throw an exception on error.

This method loads the default config if one has not been already loaded.

2.4.7 Image helper

Module *escpos.image* Image format handling class.

This module contains the image format handler *EscposImage*.

author
Michael Billington

organization
python-escpos

copyright
Copyright (c) 2016 Michael Billington <michael.billington@gmail.com>

license
MIT

class escpos.image.**EscposImage**(*img_source*)

Bases: object

Load images in, and output ESC/POS formats.

The class is designed to efficiently delegate image processing to PIL, rather than spend CPU cycles looping over pixels.

property width: int

Return width of image in pixels.

property width_bytes: int

Return width of image if you use 8 pixels per byte and 0-pad at the end.

property height: int

Height of image in pixels.

to_column_format(*high_density_vertical=True*)

Extract slices of an image as equal-sized blobs of column-format data.

Parameters
high_density_vertical (bool) – Printed line height in dots

Return type
Iterator[bytes]

to_raster_format()

Convert image to raster-format binary.

Return type
bytes

split(*fragment_height*)

Split an image into multiple fragments after fragment_height pixels.

Parameters
fragment_height (int) – height of fragment

Returns
list of PIL objects

center(*max_width*)

Center image in place.

Param
Maximum width in order to deduce x offset for centering

Return type
None

Returns
None

2.4.8 CLI

Module *escpos.cli*

CLI.

This module acts as a command line interface for python-escpos. It mirrors closely the available ESCPOS commands while adding a couple extra ones for convenience.

It requires you to have a configuration file. See documentation for details.

escpos.cli.**str_to_bool**(*string*)

Convert string to bool.

Used as a type in argparse so that we get back a proper bool instead of always True.

Return type
bool

escpos.cli.**print_extended_information**()

Print diagnostic information for bug reports.

Return type
None

escpos.cli.**generate_parser**()

Generate an argparse parser.

Return type
ArgumentParser

escpos.cli.**main**()

Handle main entry point of CLI script.

Handles loading of configuration and creating and processing of command line arguments. Called when run from a CLI.

Return type
None

escpos.cli.**demo**(*printer*, ***kwargs*)

Print demos.

Called when CLI is passed *demo*. This function uses the DEMO_FUNCTIONS dictionary.

Parameters

- **printer** (*Escpos*) – A printer from escpos.printer
- **kwargs** – A dict with a key for each function you want to test. It's in this format since it usually comes from argparse.

Return type
None

2.4.9 Magic Encode

Module *escpos.magicencode* Magic Encode.

This module tries to convert an UTF-8 string to an encoded string for the printer. It uses trial and error in order to guess the right codepage. The code is based on the encoding-code in py-xml-escpos by @fvdsn.

author
Patrick Kanzler

organization
python-escpos

copyright
Copyright (c) 2016 Patrick Kanzler and Frédéric van der Essen

license
MIT

class escpos.magicencode.**Encoder**(*codepage_map*)

Bases: object

Take available code spaces and pick the right one for a given character.

Note: To determine the code page, it needs to do the conversion, and thus already knows what the final byte in the target encoding would be. Nevertheless, the API of this class does not return the byte.

The caller use to do the character conversion itself.

get_sequence(*encoding*)

Get a sequence.

get_encoding_name(*encoding*)

Return a canonical encoding name.

Given an encoding provided by the user, will return a canonical encoding name; and also validate that the encoding is supported.

Todo: Support encoding aliases: pc437 instead of cp437.

---

can_encode(*encoding*, *char*)

Determine if a character is encodable in the given code page.

Parameters

- **encoding** – The name of the encoding.
- **char** – The character to attempt to encode.

encode(*text*, *encoding*, *defaultchar='?'*)

Encode text under the given encoding.

Parameters

- **text** – Text to encode
- **encoding** – Encoding name to use (must be defined in capabilities)
- **defaultchar** – Fallback for non-encodable characters

find_suitable_encoding(*char*)

Search in a specific order for a suitable encoding.

It is the following order:

1. code pages that we already tried before; there is a good chance they might work again, reducing the search space, and by re-using already used encodings we might also reduce the number of codepage change instruction we have to send. Still, any performance gains will presumably be fairly minor.
2. code pages in lower ESCPOS slots first. Presumably, they are more likely to be supported, so if a printer profile is missing or incomplete, we might increase our change that the code page we pick for this character is actually supported.

escpos.magicencode.**split_writable_text**(*encoder*, *text*, *encoding*)

Split up the writable text.

Splits off as many characters from the beginning of text as are writable with "encoding". Returns a 2-tuple (writable, rest).

class escpos.magicencode.**MagicEncode**(*driver*, *encoding=None*, *disabled=False*, *defaultsymbol='?'*, *encoder=None*)

Bases: object

Help switching to the right code page.

A helper that helps us to automatically switch to the right code page to encode any given Unicode character.

This will consider the printers supported codepages, according to the printer profile, and if a character cannot be encoded with the current profile, it will attempt to find a suitable one.

If the printer does not support a suitable code page, it can insert an error character.

force_encoding(*encoding*)

Set a fixed encoding. The change is emitted right away.

From now one, this buffer will switch the code page anymore. However, it will still keep track of the current code page.

write(*text*)

Write the text, automatically switching encodings.

write_with_encoding(*encoding*, *text*)

Write the text and inject necessary codepage switches.

2.4.10 Codepages

Module *escpos.codepages* Helper module for codepage handling.

class escpos.codepages.**CodePageManager**(*data*)

Bases: object

Holds information about all the code pages.

Information as defined in escpos-printer-db.

static get_encoding_name(*encoding*)

Get encoding name.

Todo: Resolve the encoding alias.

get_encoding(*encoding*)

Return the encoding data.

2.4.11 Katakana

Module *escpos.katakana* Helpers to encode Japanese characters.

I doubt that this currently works correctly.

escpos.katakana.**encode_katakana**(*text*)

I don't think this quite works yet.

Return type
bytes

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

INDEX

D

E

F

G

H

I

W