Examples

Usage examples of this `micropython-i2c-lcd` library

General

An example of all implemented functionalities can be found at the MicroPython I2C LCD examples folder

Setup Display

```
from lcd_i2c import LCD
from machine import I2C, Pin

# PCF8574 on 0x27
I2C_ADDR = 0x27
NUM_ROWS = 2
NUM_COLS = 16

# define custom I2C interface, default is 'I2C(0)'
# check the docs of your device for further details and pin infos
# this are the pins for the Raspberry Pi Pico adapter board
i2c = I2C(0, scl=Pin(13), sda=Pin(12), freq=800000)
lcd = LCD(addr=I2C_ADDR, cols=NUM_COLS, rows=NUM_ROWS, i2c=i2c)

# get LCD infos/properties
print("LCD is on I2C address {}".format(lcd.addr))
print("LCD has {} columns and {} rows".format(lcd.cols, lcd.rows))
print("LCD is used with a charsize of {}".format(lcd.charsize))
print("Cursor position is {}".format(lcd.cursor_position))

# start LCD, not automatically called during init to be Arduino compatible
lcd.begin()
```

Text

Show Text

```
# LCD has already been setup, see section "Setup Display"

lcd.print("Hello World")
```

Clear Text

This command clears the text on the screen and sets the cursor position back to its home position at `(0, 0)`

```
# LCD has already been setup, see section "Setup Display"

lcd.clear()
```

Scroll Text

```
# LCD has already been setup, see section "Setup Display"
from time import sleep

text = "Hello World"

# show text on LCD
lcd.print(text)

# scroll text to the left
for _ in text:
    lcd.scroll_display_left()
    sleep(0.5)

# scroll text to the right
for _ in text:
    lcd.scroll_display_right()
    sleep(0.5)
```

Text Flow

```
# LCD has already been setup, see section "Setup Display"

# set text flow right to left
lcd.set_cursor(col=12, row=0)
lcd.right_to_left()
lcd.print("Right to left")

# set text flow left to right
lcd.set_cursor(col=0, row=0)
lcd.left_to_right()
lcd.print("Left to right")
```

Autoscroll

```
# LCD has already been setup, see section "Setup Display"

# activate autoscroll
lcd.autoscroll()

# disable autoscroll
lcd.no_autoscroll()
```

Custom Characters

Custom characters can be defined for 8 CGRAM locations. The character has to be defined as binary of HEX list. In case you can't see the matrix, simply use the LCD Character Creator page of Max Promer

The following example defines a upright happy smiley `:-)` at the first (0) location in the displays CGRAM using 5x10 pixels. Maybe you can see it ...

```
00000
00000
10001
00100
00100
10001
01110
00000
```

```
# LCD has already been setup, see section "Setup Display"

# custom char can be set for location 0 ... 7
lcd.create_char(
    location=0,
    charmap=[0x00, 0x00, 0x11, 0x04, 0x04, 0x11, 0x0E, 0x00]
)

# show custom char stored at location 0
lcd.print(chr(0))
```

Backlight

The following functions can be used to control the LCD backlight

```
# LCD has already been setup, see section "Setup Display"

# turn LCD off
lcd.no_backlight()

# turn LCD on
lcd.backlight()

# turn LCD off
lcd.set_backlight(False)

# turn LCD on
lcd.set_backlight(True)

# get current backlight value
print("Backlight value: {}".format(lcd.get_backlight()))

# get current backlight value via property
print("Backlight value: {}".format(lcd.backlightval))
```

Cursor

The following functions can be used to control the cursor

```
# LCD has already been setup, see section "Setup Display"

# turn cursor on (show)
lcd.cursor()

# turn cursor off (hide)
lcd.no_cursor()

# turn cursor on (show)
lcd.cursor_on()

# turn cursor off (hide)
lcd.cursor_off()

# blink cursor
lcd.blink()

# stop blinking cursor
lcd.no_blink()

# set cursor to home position (0, 0)
lcd.home()

# set cursor position to first line, third column
lcd.set_cursor(col=3, row=0)

# set cursor position to second line, seventh column
lcd.cursor_position = (7, 1)

# get current cursor position via property
print("Cursor position: {}".format(lcd.cursor_position))
```

Display

```
# LCD has already been setup, see section "Setup Display"

# turn display off
lcd.no_display()

# turn display on
lcd.display()
```