```
import utime
import gc

from lcd_api import LcdApi
from machine import I2C

# PCF8574 pin definitions
MASK_RS = 0x01       # P0
MASK_RW = 0x02       # P1
MASK_E  = 0x04       # P2

SHIFT_BACKLIGHT = 3  # P3
SHIFT_DATA      = 4  # P4-P7

class I2cLcd(LcdApi):

    #Implements a HD44780 character LCD connected via PCF8574 on I2C

    def __init__(self, i2c, i2c_addr, num_lines, num_columns):
        self.i2c = i2c
        self.i2c_addr = i2c_addr
        self.i2c.writeto(self.i2c_addr, bytes([0]))
        utime.sleep_ms(20)   # Allow LCD time to powerup
        # Send reset 3 times
        self.hal_write_init_nibble(self.LCD_FUNCTION_RESET)
        utime.sleep_ms(5)    # Need to delay at least 4.1 msec
        self.hal_write_init_nibble(self.LCD_FUNCTION_RESET)
        utime.sleep_ms(1)
        self.hal_write_init_nibble(self.LCD_FUNCTION_RESET)
        utime.sleep_ms(1)
        # Put LCD into 4-bit mode
        self.hal_write_init_nibble(self.LCD_FUNCTION)
        utime.sleep_ms(1)
        LcdApi.__init__(self, num_lines, num_columns)
        cmd = self.LCD_FUNCTION
        if num_lines > 1:
            cmd |= self.LCD_FUNCTION_2LINES
        self.hal_write_command(cmd)
        gc.collect()

    def hal_write_init_nibble(self, nibble):
        # Writes an initialization nibble to the LCD.
        # This particular function is only used during initialization.
        byte = ((nibble >> 4) & 0x0f) << SHIFT_DATA
        self.i2c.writeto(self.i2c_addr, bytes([byte | MASK_E]))
        self.i2c.writeto(self.i2c_addr, bytes([byte]))
        gc.collect()

    def hal_backlight_on(self):
        # Allows the hal layer to turn the backlight on
        self.i2c.writeto(self.i2c_addr, bytes([1 << SHIFT_BACKLIGHT]))
        gc.collect()

    def hal_backlight_off(self):
        #Allows the hal layer to turn the backlight off
        self.i2c.writeto(self.i2c_addr, bytes([0]))
        gc.collect()
    def hal_write_command(self, cmd):
        # Write a command to the LCD. Data is latched on the falling edge
of E.
        byte = ((self.backlight << SHIFT_BACKLIGHT) |
                (((cmd >> 4) & 0x0f) << SHIFT_DATA))
        self.i2c.writeto(self.i2c_addr, bytes([byte | MASK_E]))
        self.i2c.writeto(self.i2c_addr, bytes([byte]))
        byte = ((self.backlight << SHIFT_BACKLIGHT) |
                ((cmd & 0x0f) << SHIFT_DATA))
        self.i2c.writeto(self.i2c_addr, bytes([byte | MASK_E]))
        self.i2c.writeto(self.i2c_addr, bytes([byte]))
        if cmd <= 3:
            # The home and clear commands require a worst case delay of
4.1 msec
            utime.sleep_ms(5)
        gc.collect()

    def hal_write_data(self, data):
        # Write data to the LCD. Data is latched on the falling edge of
E.
        byte = (MASK_RS |
                (self.backlight << SHIFT_BACKLIGHT) |
                (((data >> 4) & 0x0f) << SHIFT_DATA))
        self.i2c.writeto(self.i2c_addr, bytes([byte | MASK_E]))
        self.i2c.writeto(self.i2c_addr, bytes([byte]))
        byte = (MASK_RS |
                (self.backlight << SHIFT_BACKLIGHT) |
                ((data & 0x0f) << SHIFT_DATA))
        self.i2c.writeto(self.i2c_addr, bytes([byte | MASK_E]))
        self.i2c.writeto(self.i2c_addr, bytes([byte]))
        gc.collect()
```