```python

from micropython import const
import framebuf
import time

# Register definitions
SET_CONTRAST = const(0x81)
SET_ENTIRE_ON = const(0xA4)
SET_NORM_INV = const(0xA6)
SET_DISP = const(0xAE)
SET_MEM_ADDR = const(0x20)
SET_COL_ADDR = const(0x21)
SET_PAGE_ADDR = const(0x22)
SET_DISP_START_LINE = const(0x40)
SET_SEG_REMAP = const(0xA0)
SET_MUX_RATIO = const(0xA8)
SET_IREF_SELECT = const(0xAD)
SET_COM_OUT_DIR = const(0xC0)
SET_DISP_OFFSET = const(0xD3)
SET_COM_PIN_CFG = const(0xDA)
SET_DISP_CLK_DIV = const(0xD5)
SET_PRECHARGE = const(0xD9)
SET_VCOM_DESEL = const(0xDB)
SET_CHARGE_PUMP = const(0x8D)

class SSD1306_DualSPI(framebuf.FrameBuffer):
    def __init__(self, width, height, spi, dc, res, cs1, cs2, external_vcc=False):
        self.width = width
        self.height = height
        self.external_vcc = external_vcc
        self.pages = self.height // 8
        self.buffer = bytearray(self.pages * self.width)
        super().__init__(self.buffer, self.width, self.height, framebuf.MONO_VLSB)

        self.spi = spi
        self.dc = dc
        self.res = res
        self.cs1 = cs1
        self.cs2 = cs2
        self.rate = 10 * 1024 * 1024

        for pin in (dc, res, cs1, cs2):
            pin.init(pin.OUT, value=0)

        self.reset()
        self.init_display()

    def reset(self):
        self.res(1)
        time.sleep_ms(1)
        self.res(0)
        time.sleep_ms(10)
        self.res(1)

    def write_cmd(self, cmd, cs):
        self.spi.init(baudrate=self.rate, polarity=0, phase=0)
        cs(1)
        self.dc(0)
        cs(0)
        self.spi.write(bytearray([cmd]))
        cs(1)

    def write_data(self, buf, cs):
        self.spi.init(baudrate=self.rate, polarity=0, phase=0)
        cs(1)
        self.dc(1)
        cs(0)
        self.spi.write(buf)
        cs(1)

    def init_display(self):
        for cs in (self.cs1, self.cs2):
            for cmd in (
                SET_DISP,
                SET_MEM_ADDR, 0x00,
                SET_DISP_START_LINE,
                SET_SEG_REMAP | 0x01,
                SET_MUX_RATIO, self.height - 1,
                SET_COM_OUT_DIR | 0x08,
                SET_DISP_OFFSET, 0x00,
                SET_COM_PIN_CFG, 0x02 if self.width > 2 * self.height else 0x12,
                SET_DISP_CLK_DIV, 0x80,
                SET_PRECHARGE, 0x22 if self.external_vcc else 0xF1,
                SET_VCOM_DESEL, 0x30,
                SET_CONTRAST, 0xFF,
                SET_ENTIRE_ON,
                SET_NORM_INV,
                SET_IREF_SELECT, 0x30,
                SET_CHARGE_PUMP, 0x10 if self.external_vcc else 0x14,
                SET_DISP | 0x01,
            ):
                self.write_cmd(cmd, cs)
        self.fill(0)
        self.show()

    def show(self):
        for page in range(0, self.pages):
            left_start = page * 256
            right_start = left_start + 128

            self.write_cmd(0xB0 | page, self.cs1)
            self.write_cmd(0x00, self.cs1)
            self.write_cmd(0x10, self.cs1)
            self.write_data(self.buffer[left_start:left_start + 128], self.cs1)

            self.write_cmd(0xB0 | page, self.cs2)
            self.write_cmd(0x00, self.cs2)
            self.write_cmd(0x10, self.cs2)
            self.write_data(self.buffer[right_start:right_start + 128], self.cs2)

    def poweroff(self):
        for cs in (self.cs1, self.cs2):
            self.write_cmd(SET_DISP, cs)

    def poweron(self):
        for cs in (self.cs1, self.cs2):
            self.write_cmd(SET_DISP | 0x01, cs)

    def contrast(self, contrast):
        for cs in (self.cs1, self.cs2):
            self.write_cmd(SET_CONTRAST, cs)
            self.write_cmd(contrast, cs)

    def invert(self, invert):
        for cs in (self.cs1, self.cs2):
            self.write_cmd(SET_NORM_INV | (invert & 1), cs)

    def rotate(self, rotate):
        for cs in (self.cs1, self.cs2):
            self.write_cmd(SET_COM_OUT_DIR | ((rotate & 1) << 3), cs)
            self.write_cmd(SET_SEG_REMAP | (rotate & 1), cs)

```