Putting the Mercury into deep sleep < 24uA

Mercury V1 manual · v1.0

https://www.altimetercloud.com/support/mercuryv1/code-deep-sleep/

The altimeter can be put in to deep sleep and consume almost no power while maintaining the 3.3V supply to the device and systems.
The Status LED's and sensors can be powered down in deep sleep as well.
Due to power leak on the I2C line on hardware revisions 1 - 3 these should leave the sensors on but be told to go to sleep to obtain the most efficient deep sleep of around 24uA.
Hardware revision 4+ can have the sensors powered off and achieve around 20uA in sleep.

All unused pins should be reset using the command gpio_reset_pin(GPIO_NUM_21); and then all GPIO pins held before going in to deep sleep to obtain the absolute lowest power consumption.
At 24uA sleep power the internal 50mAh will last around 3 months in deep sleep.

This code also demonstarates how to use the POWER button which is actually a reset button as a on/off power button. This is achieved by saving a toggle of 0 or 1 to the NVS preferences. This means on each reset cycle the code will either go ON or OFF.

Using Arduino IDE? Our online programmer includes Mercury_Pins.h by default so the pin names work without issue. If you are using Arduino IDE or another programmer, copy the Mercury_Pins.h tab content and paste it into the top of your program.

```
/*
 * Mercury Deep Sleep Toggle Example
 * Each reset toggles between ON (green strobe) and OFF (deep sleep).
 * Sensors are put to sleep via I2C before entering deep sleep.
 * VACC stays HIGH to avoid current leaking through I2C pullup ESD diodes.
 */
#include "Wire.h"
#include "Preferences.h"
#include "Adafruit_NeoPixel.h"
#include "esp_sleep.h"
#include "driver/gpio.h"
#include "WiFi.h"
#include "Mercury_Pins.h"

Adafruit_NeoPixel pixels(4, LED, NEO_GRB + NEO_KHZ800);
Preferences preferences;
unsigned int onoroff = 0;

void led_clear() {
  pixels.setPixelColor(0, 0);
  pixels.setPixelColor(1, 0);
  pixels.setPixelColor(2, 0);
  pixels.setPixelColor(3, 0);
  pixels.show();
}

void i2cWrite(uint8_t addr, uint8_t reg, uint8_t val) {
  Wire.beginTransmission(addr);
  Wire.write(reg);
  Wire.write(val);
  Wire.endTransmission();
}
```

```
bool i2cPresent(uint8_t addr) {
    Wire.beginTransmission(addr);
    return (Wire.endTransmission() == 0);
}

void setup() {
    // Turn on sensor power
    pinMode(VACC, OUTPUT);
    digitalWrite(VACC, HIGH);

    // NeoPixel on, red flash to confirm reset
    pinMode(LEDPOWER, OUTPUT);
    digitalWrite(LEDPOWER, HIGH);
    delay(20);
    pixels.begin();
    pixels.setPixelColor(0, pixels.Color(80, 0, 0));
    pixels.setPixelColor(1, pixels.Color(80, 0, 0));
    pixels.setPixelColor(2, pixels.Color(80, 0, 0));
    pixels.setPixelColor(3, pixels.Color(80, 0, 0));
    pixels.show();
    delay(100);
    led_clear();

    // —— Early on/off decision before Serial or USB init ——
    preferences.begin("example", false);
    delay(2);
    onoroff = preferences.getUInt("onoroff", 0);

    if (onoroff == 0) { onoroff = 1; }
    else { onoroff = 0; }

    preferences.putUInt("onoroff", onoroff);
    delay(10);
    preferences.end();

    // ═══════════════════════════════════════════════════════════════════════════════════════════
    // SLEEP PATH — onoroff is 0, put sensors to sleep and enter deep sleep
    // ═══════════════════════════════════════════════════════════════════════════════════════════
    if (onoroff == 0) {

        pixels.setPixelColor(0, pixels.Color(80, 0, 0));
        pixels.setPixelColor(1, pixels.Color(80, 0, 0));
        pixels.setPixelColor(2, pixels.Color(80, 0, 0));
        pixels.setPixelColor(3, pixels.Color(80, 0, 0));
        pixels.show();
        delay(1200);
        led_clear();
        delay(5);

        // —— Put sensors to sleep via I2C ——
        // VACC stays HIGH — if we cut sensor power the I2C pullups
        // (connected to 3V3) would leak current through the sensor
        // ESD protection diodes. Instead we send sleep commands.
        Wire.begin(SDA, SCL);
        delay(10);

        // BMP581 (rev 3+ at 0x47): write 0x00 to ODR register = standby ~1.3uA
        if (i2cPresent(0x47)) i2cWrite(0x47, 0x37, 0x00);
```

```
        // BMP390 (rev 0-2 at 0x77): write 0x00 to PWR_CTRL = sleep ~3.4uA
        if (i2cPresent(0x77)) i2cWrite(0x77, 0x1B, 0x00);

        // LSM6DSO32 (all revisions at 0x6B): power down accel + gyro ~3uA
        if (i2cPresent(0x6B)) {
            i2cWrite(0x6B, 0x10, 0x00);  // CTRL1_XL = accel off
            i2cWrite(0x6B, 0x11, 0x00);  // CTRL2_G  = gyro off
        }

        Wire.end();

        // —— Turn off NeoPixel ——
        digitalWrite(LEDPOWER, LOW);
        pinMode(LED, OUTPUT);
        digitalWrite(LED, LOW);

        // —— Turn off output ——
        pinMode(OUT1, OUTPUT);
        digitalWrite(OUT1, LOW);

        // —— Hold pins during deep sleep ——
        // gpio_hold_en tells the RTC to maintain each pin's current
        // level while the main GPIO controller is powered down.
        // Without this, pins float and sensors could draw extra current.
        gpio_hold_en((gpio_num_t)VACC);       // Held HIGH — sensors powered in sleep
        gpio_hold_en((gpio_num_t)LEDPOWER);   // Held LOW
        gpio_hold_en((gpio_num_t)LED);        // Held LOW
        gpio_hold_en((gpio_num_t)OUT1);       // Held LOW

        esp_deep_sleep_start();
    }

    // ═══════════════════════════════════════════════════════════════════════
    // ON PATH — release holds and run
    // ═══════════════════════════════════════════════════════════════════════
    gpio_hold_dis((gpio_num_t)VACC);
    gpio_hold_dis((gpio_num_t)LEDPOWER);
    gpio_hold_dis((gpio_num_t)LED);
    gpio_hold_dis((gpio_num_t)OUT1);

    // Restore power (may have been held from sleep)
    digitalWrite(VACC, HIGH);
    digitalWrite(LEDPOWER, HIGH);
}

void loop() {
    // Flash green/blue when in on state
    for (int i = 0; i < 4; i++) {
        pixels.setPixelColor(i, pixels.Color(0, 80, 0));
    }
    pixels.show();
    delay(200);
    for (int i = 0; i < 4; i++) {
        pixels.setPixelColor(i, pixels.Color(0, 0, 80));
    }
    pixels.show();
    delay(200);
}
```

```
#pragma once
/*
 * Mercury (ESP32-C6) Pin Definitions
 * Board-specific GPIO assignments
 */

// —— Status LED (NeoPixel) ——
#define LEDPOWER      3    // NeoPixel power (drive HIGH to enable)
#define LED           2    // NeoPixel data signal

// —— I2C Bus ——
#define SDA           21   // I2C data
#define SCL           22   // I2C clock

// —— Sensor Power ——
#define VACC          20   // Sensor power rail (drive HIGH to enable)

// —— General Purpose Ports ——
#define GP06          6    // GP06 port
#define GP07          7    // GP07 port

// —— High Current Output ——
#define OUT1          5    // High current output (e.g. pyro / relay)

// —— Battery Bar LEDs ——
#define BL1           4    // Battery LED 1 (lowest)
#define BL2           14   // Battery LED 2
#define BL3           15   // Battery LED 3
#define BL4           18   // Battery LED 4
#define BL5           19   // Battery LED 5 (highest)

// —— Indicators ——
#define DISK          8    // Disk activity LED

// —— Analogue / Detection ——
#define BATIN         0    // Battery voltage (1:1 divider)
#define USBDETECT     1    // USB power detect (HIGH = USB present)
#define BUTTON        9    // BUTTON on the board, boot button but can be used
```