Code example: BMP390 & BMP581 pressure sensor with auto detect

Mercury V1 manual · v1.0

https://d21c22pth15itr.cloudfront.net/support/mercuryv1/code-pressure-sensor/

The Mercury has either a Bosch BMP390 or Bosch BMP581 pressure sensor onboard.
This script shows you how to detect which one you have and read it's temperature and pressure and convert it to an altitude.

The standard conversion used in this program assumes 1013.25 Hpa (average) pressure and a 15 degrees celcius ambient temperature.

If you track down your local weather forecast for your location and time and enter that pressure instead you should be shown roughly your height above sea level with this script.
If using our online compiler and uploader then don't forget to connect the Serial monitor after resetting (power button press) your Mercury to see the information coming through.

> **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.

```
// Pressure sensor test example for Mercury altimeters (altimetercloud.com)
#include "Wire.h"
#include "Mercury_Pins.h"

// There are quite a few options for libraries we've chosen these for our example
#include "Adafruit_BMP3XX.h"
#include "Adafruit_BMP5xx.h"

Adafruit_BMP3XX bmp3;
Adafruit_BMP5xx bmp5;

bool hasBMP581 = false;
bool hasBMP390 = false;

float seaLevelHpa = 1013.25;   // The forecast sea level weather pressure for your location.
float ambientTemp = 15.0;      // The ambient temperature in degrees celcius

void setup() {
  Serial.begin(115200);

  // Turn on the power to the sensors on the Mercury altimeter
  pinMode(VACC, OUTPUT);
  digitalWrite(VACC, HIGH);
  delay(100);

  // Start I2C on the Mercurys SDA and SCL pins
  Wire.begin(SDA, SCL);

  // Detect which pressure sensor is present on your Mercury, earlier revisions had a Bosch BMP390 and later revisions (3+) have the Bosch BMP581
  if (bmp5.begin(0x47, &Wire)) {
    hasBMP581 = true;
    Serial.println("BMP581 found");
  } else if (bmp3.begin_I2C(0x77, &Wire)) {
    hasBMP390 = true;
    Serial.println("BMP390 found");
  } else {
    Serial.println("No sensor found!");
  }
}

void loop() {
  float temp = 0, pressure = 0, alt = 0;
```

```
    // Read the detected sensor for it's pressure and temperature
    if (hasBMP581 && bmp5.performReading()) {
        temp = bmp5.temperature;
        pressure = bmp5.pressure;
    } else if (hasBMP390 && bmp3.performReading()) {
        temp = bmp3.temperature;
        pressure = bmp3.pressure / 100.0;
    } else {
        delay(500);
        return;
    }

    // Convert the pressure to altitude using the standard international standard using sea level forecast pressure and ambient temperature.
    alt = ((ambientTemp + 273.15) / 0.0065) * (1.0 - pow(pressure / seaLevelHpa, 0.190266669));
    Serial.printf("Temp: %.1fC  Pressure: %.2f hPa  Alt: %.1f m\n", temp, pressure, alt);
    delay(500);
}
```

```
#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
```