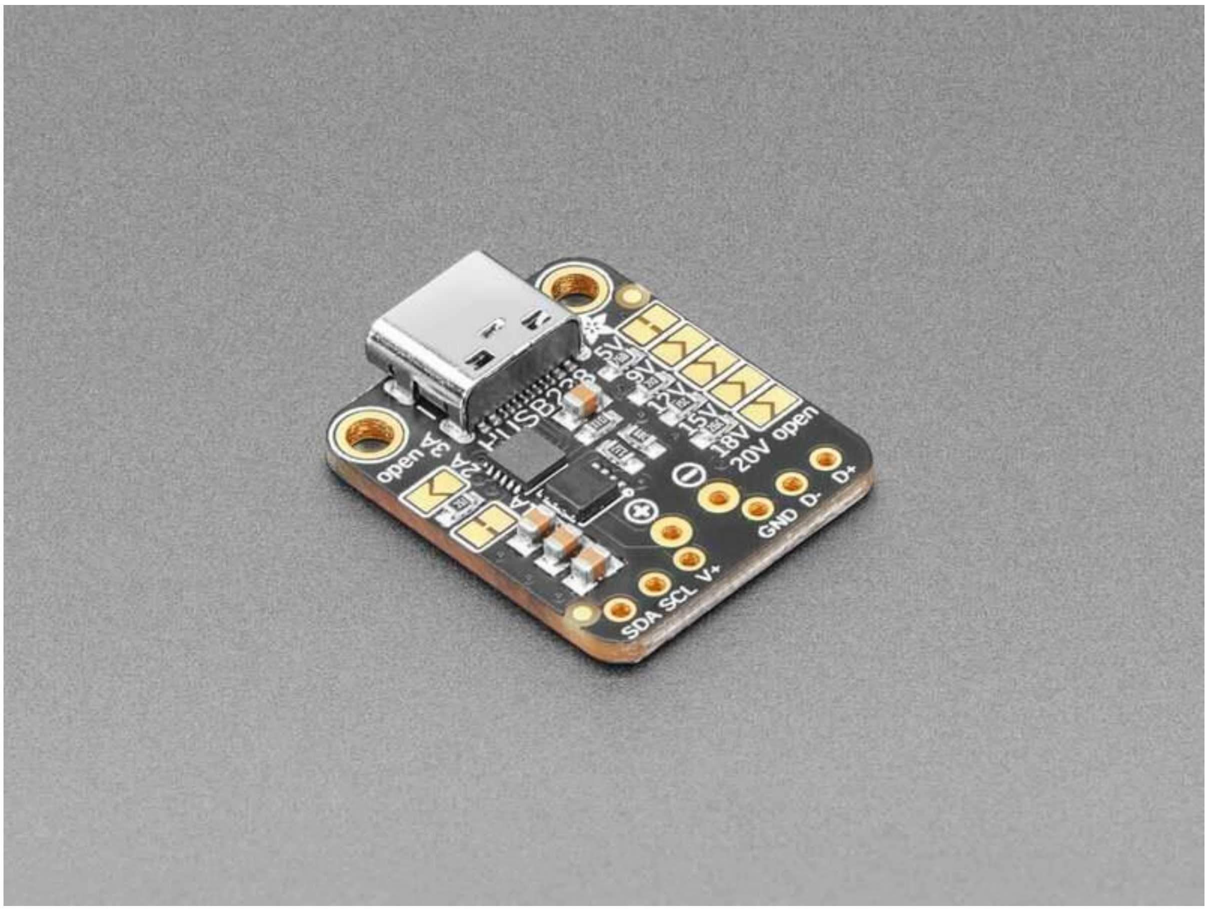
Adafruit HUSB238 USB Type C Power Delivery Breakout

Created by Liz Clark



https://learn.adafruit.com/adafruit-husb238-usb-type-c-power-delivery-breakout

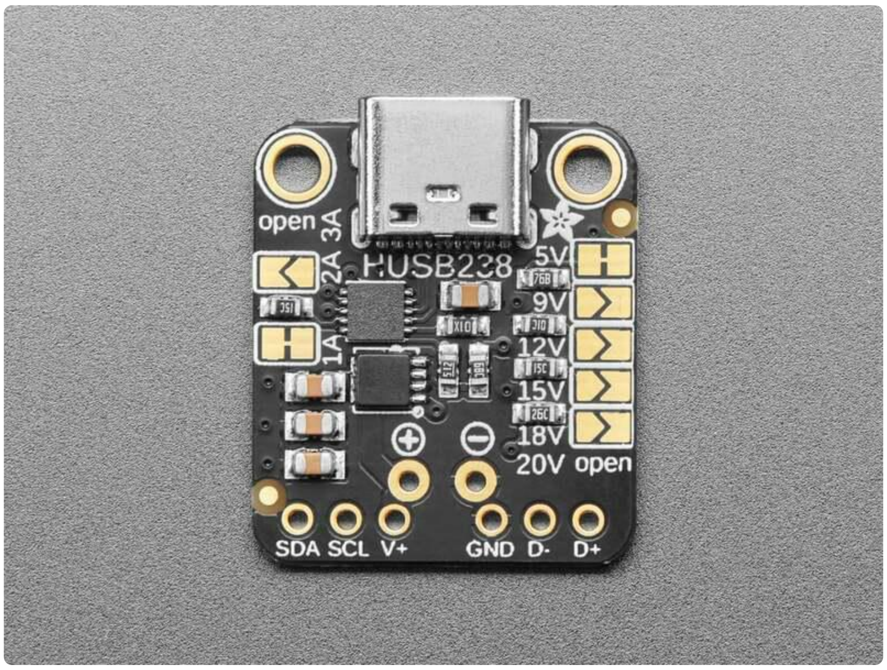
Last updated on 2023-10-24 04:18:58 PM EDT

Table of Contents

Overview

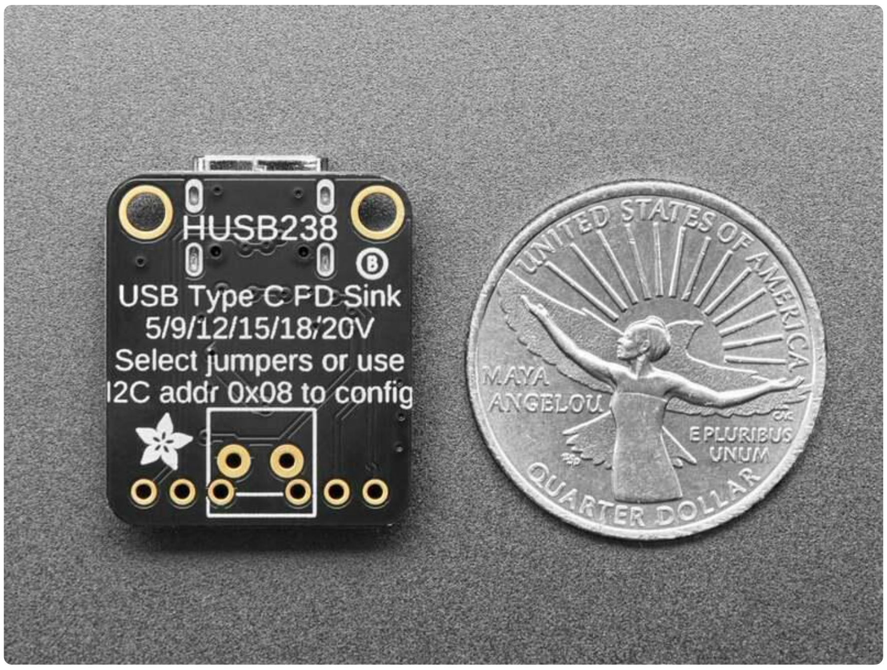
The HUSB238 USB PD sink chip is neat in that you can either use jumpers (really, resistor selection) to set the desired power delivery voltage and current or you can use I2C for dynamic querying and setting.

We've built a nice Adafruit USB Type C Power Delivery Dummy Breakout board around the HUSB238 to make it very easy to configure and integrate without having to solder any tiny resistors.

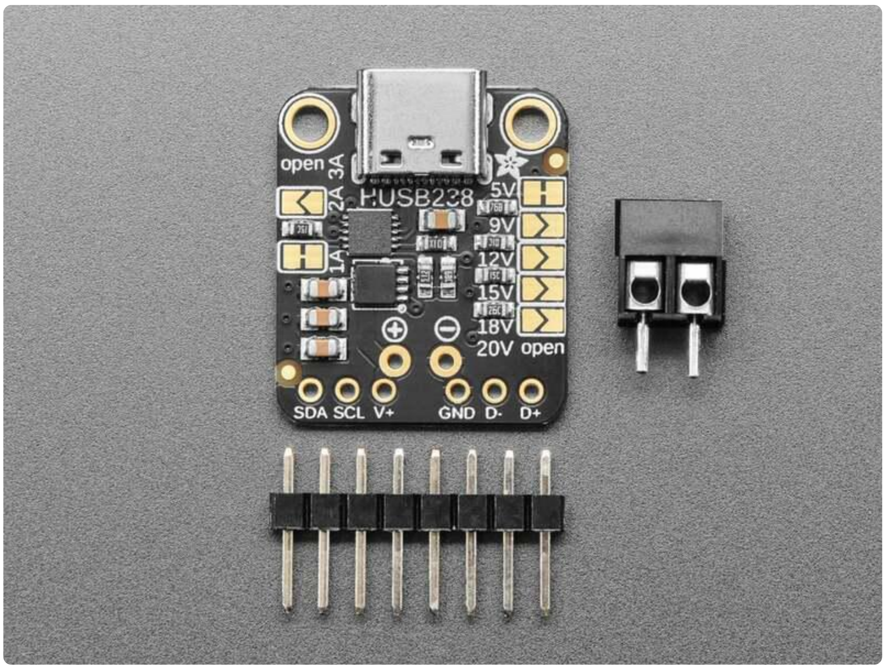


It's perfect for use with USB Type C wall adapters that can provide multiple voltages, the standard offerings are 5V, 9V, 12V, 15V, 18V and 20V. This HUSB238 breakout plugs into the USB C cable and then over the CC lines will negotiate the PD request and commands. For example we can ask what voltages are available and then pick the highest one. Or if you need a specific voltage it will specifically select that one.

This breakout will be handy for projects where you need a lot more than 5V @ 2A power: this adapter can give up to 20V at 5A - yes you can get 100W over USB C! - and you could buck that down to get a ton of current at 5V or 12V if that's needed. Or use it to convert a DC or battery-powered device into a USB C powered one!



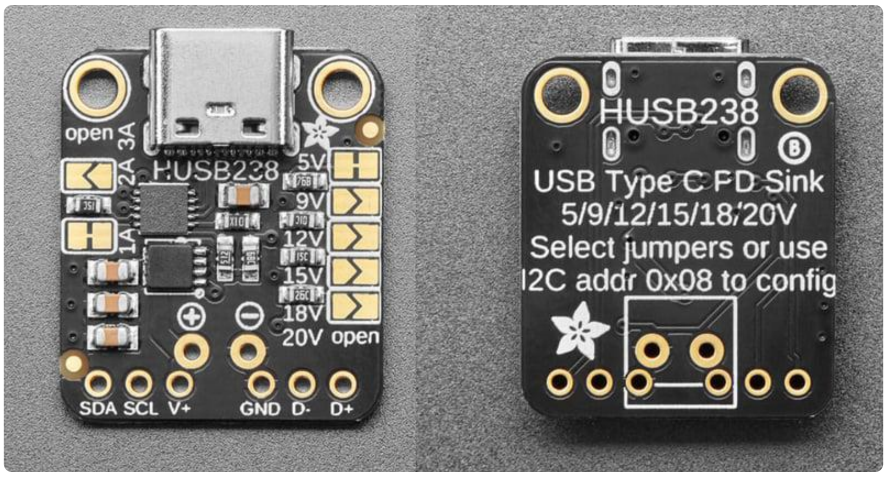
Jumper-configured usage is simple: by default its hard-wired for 5V 1A output since that's what USB C will always provide at first. Cut the 5V jumper and solder closed the 9V, 12V, 15V, 18V or 20V jumper to select the resistor that sets the voltage. You can also select the desired current from 2A to 3A, although we have found that this isn't as essential, you could always just pull as much as the adapter will provide. No microcontroller or microcomputer is required!



I2C-configured usage is a little more challenging. Since the Vout can be as high as 20V, we don't have an onboard voltage regulator or pullup resistors. Connect to a microcontroller or microcomputer board that has a separate power supply (or that can regulate from up-to-20V if you plan on selecting that high) and I2C pull-up resistors to your desired logic level. Then use the Arduino library and example code () to query the USB Type C PD source for available voltages and currents and select the desired voltage dynamically. When configuring over I2C, the jumper settings are used on startup until the I2C commands come over.

Comes with a small bit of header and a terminal block so you can decide whether you want to use it in a breadboard, or free-wired.

Pinouts



The default I2C address is 0x08.

Power Pins

- V+ - This is the voltage output (Vout) pin. This pin will output the voltage selected from the HUSB238. It is also available from the + pin from the terminal block.
- GND - Common ground for power and logic. It is also available from the - pin from the terminal block.

I2C

Since the Vout can be as high as 20V, we don't have an onboard voltage regulator or pull-up resistors on the I2C lines. Connect these pins to a microcontroller or microcomputer board that has a separate power supply (or that can regulate from up-to-20V if you plan on selecting that high) and I2C pull-up resistors to your desired logic level. When configuring over I2C, the jumper settings are used on startup until the I2C commands come over.

- SCL - I2C clock pin, connect to your microcontroller's I2C clock line.
- SDA - I2C data pin, connect to your microcontroller's I2C data line.

When configuring over I2C, the jumper settings are used on startup until the I2C commands come over.

USB Type C Port and Data Pins

At the top of the board is the USB type C port. You'll use this port to plug into a USB C PD wall adapter with a USB C cable. The USB data pins are broken out on the board: Data Plus (labeled D+ on the board silk) and Data Minus (labeled D- on the board silk).

Jumpers

A series of jumpers are used to determine the voltage and current requested from the PD adapter. By default, the 5V and 1A jumpers are closed. You can cut these jumpers to change voltages and/or amps.

Amp Jumpers

- 1A - The one amp jumper. Closed by default. Cut the jumper to change amperage.
- 2A - The two amps jumper. Open by default. Solder it closed to select.

To select 3A, leave both the 1A and 2A jumpers open.

Voltage Jumpers

- 5V - The 5V jumper. Closed by default. Cut the jumper to change voltages.
- 9V - The 9V jumper. Open by default. Solder it closed to select.
- 12V - The 12V jumper. Open by default. Solder it closed to select.
- 15V - The 15V jumper. Open by default. Solder it closed to select.
- 18V - The 18V jumper. Open by default. Solder it closed to select.

To select 20V, or the highest available voltage from your adapter, leave all of the voltage jumpers open.

Python Docs

Python Docs ()

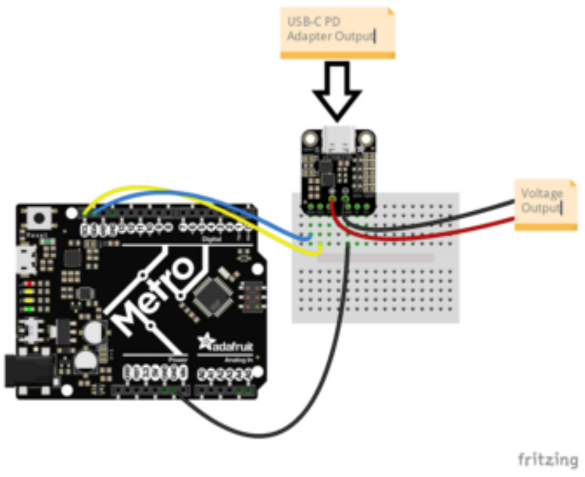
Arduino

Using the HUSB238 breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller, installing the Adafruit_HUSB238 () library, plugging in a USB C PD power supply to the breakout and running the provided example code. It's important to note that when you are controlling the breakout over I2C, the jumper settings on the board are used on startup until the I2C commands come over.

When configuring over I2C, the jumper settings are used on startup until the I2C commands come over.

Wiring

Here is an Adafruit Metro wired up to the breakout. For testing, you can connect the + and - outputs from the breakout to a multimeter with alligator clips. You'll set the multimeter to read DC voltage (labeled with a "V" and one dashed and one solid line).

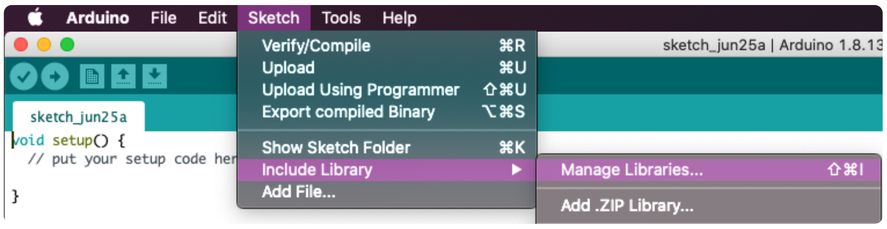


USB C PD power supply to breakout USB C port
Board GND to breakout GND (black wire)
Board SCL to breakout SCL (yellow wire)
Board SDA to breakout SDA (blue wire)
Breakout + to multimeter positive (red wire)
Breakout - to multimeter negative (black wire)

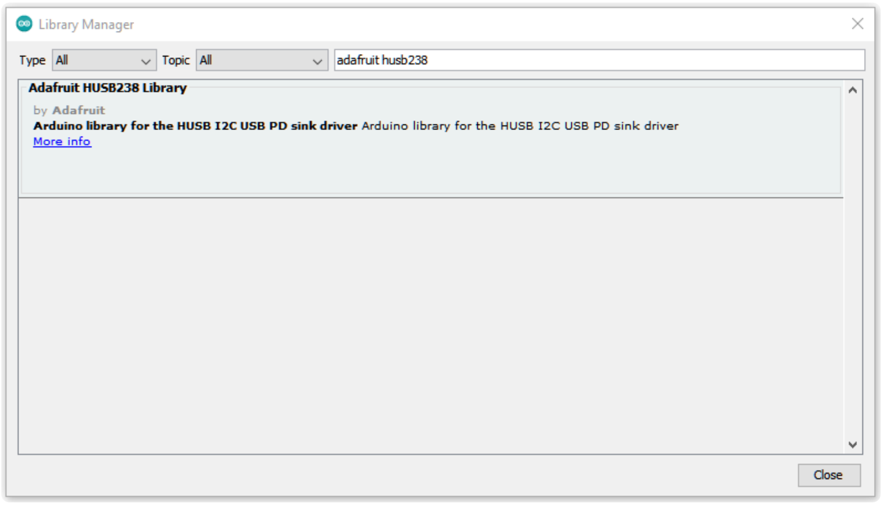
Digital Multimeter - Model 9205B+
This massive multimeter has everything but the kitchen sink included. It's a great addition to any workbench or toolbox. It's low cost, simple to use, and has a big clear...
https://www.adafruit.com/product/2034

Library Installation

You can install the Adafruit_HUSB238 library for Arduino using the Library Manager in the Arduino IDE.

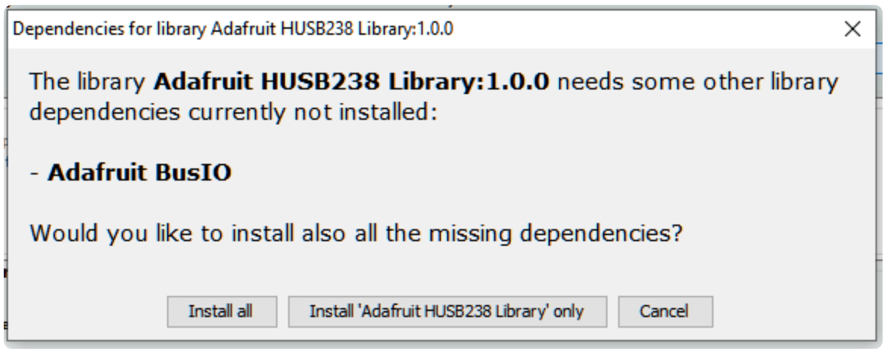


Click the Manage Libraries ... menu item, search for Adafruit_HUSB238, and select the Adafruit HUSB238 library:



Adafruit HUSB238 Library
by **Adafruit**
Arduino library for the HUSB I2C USB PD sink driver Arduino library for the HUSB I2C USB PD sink driver
More info

If asked about dependencies, click "Install all".



Dependencies for library Adafruit HUSB238 Library:1.0.0

The library **Adafruit HUSB238 Library:1.0.0** needs some other library dependencies currently not installed:

- **Adafruit BusIO**

Would you like to install also all the missing dependencies?

If the "Dependencies" window does not come up, then you already have the dependencies installed.

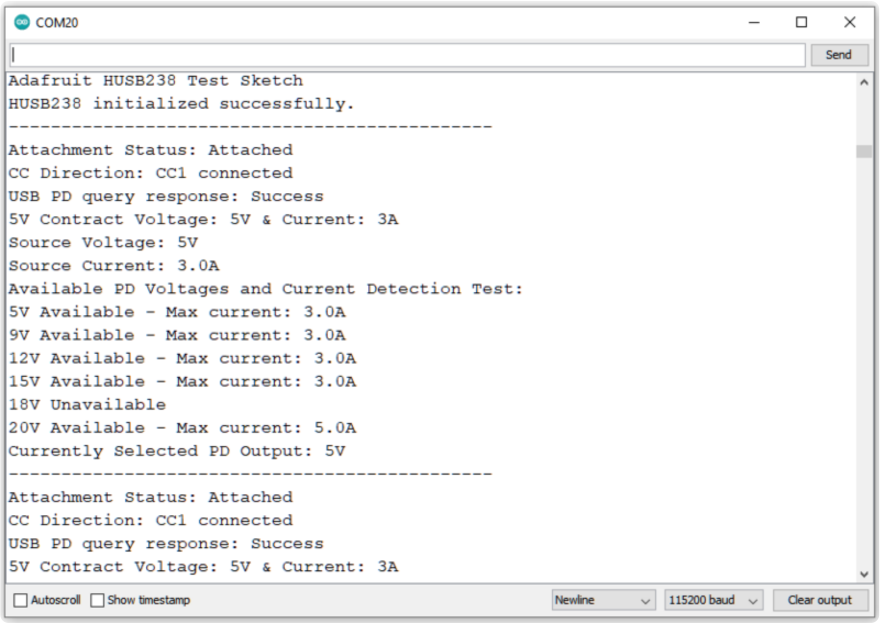
If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

Simple Test

```
#include <Wire.h>
#include "Adafruit_HUSB238.h"

Adafruit_HUSB238 husb238;
```

```
void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);
  Serial.println("Adafruit HUSB238 Test Sketch");

  // Initialize the HUSB238
  if (husb238.begin(HUSB238_I2CADDR_DEFAULT, &Wire)) {
    Serial.println("HUSB238 initialized successfully.");
  } else {
    Serial.println("Couldn't find HUSB238, check your wiring?");
    while (1);
  }
}

void loop() {
  delay(1000);  // Add a delay to prevent flooding the serial output
  Serial.println(F("-----------------------------------------------"));

  // Determine whether attached or unattached
  bool attached = husb238.isAttached();
  Serial.print("Attachment Status: ");
  Serial.println(attached ? "Attached" : "Unattached");

  if (! attached) return;

  // Test getCCStatus function
  bool ccStatus = husb238.getCCdirection();
  Serial.print("CC Direction: ");
  Serial.println(ccStatus ? "CC1 connected" : "CC2 Connected");

  // Check if we can get responses to our PD queries!
  HUSB238_ResponseCodes pdResponse = husb238.getPDResponse();
  Serial.print("USB PD query response: ");
  switch (pdResponse) {
    case NO_RESPONSE:
      Serial.println("No response");
      break;
    case SUCCESS:
      Serial.println("Success");
      break;
    case INVALID_CMD_OR_ARG:
      Serial.println("Invalid command or argument");
      break;
    case CMD_NOT_SUPPORTED:
      Serial.println("Command not supported");
      break;
    case TRANSACTION_FAIL_NO_GOOD_CRC:
      Serial.println("Transaction fail");
      break;
    default:
      Serial.println("Unknown response code");
      break;
  }

  if (pdResponse != SUCCESS)
    return;

  // Is there a default 5V 'contract' voltage available
  bool contractV = husb238.get5VContractV();
  Serial.print("5V Contract Voltage: ");
  Serial.print(contractV ? "5V" : "Other");

  // How much current can we get?
  HUSB238_5VCurrentContract contractA = husb238.get5VContractA();
  Serial.print(" & Current: ");
  switch (contractA) {
    case CURRENT5V_DEFAULT:
      Serial.println("Default current");
      break;
    case CURRENT5V_1_5_A:
      Serial.println("1.5A");
      break;
    case CURRENT5V_2_4_A:
      Serial.println("2.4A");
      break;
    case CURRENT5V_3_A:
      Serial.println("3A");
      break;
    default:
      Serial.println("Unknown current");
      break;
  }

  // What is the actual voltage being output right now?
  HUSB238_VoltageSetting srcVoltage = husb238.getPDSrcVoltage();
  Serial.print("Source Voltage: ");
  switch (srcVoltage) {
    case UNATTACHED:
      Serial.println("Unattached");
      break;
    case PD_5V:
      Serial.println("5V");
      break;
    case PD_9V:
      Serial.println("9V");
      break;
    case PD_12V:
      Serial.println("12V");
      break;
    case PD_15V:
      Serial.println("15V");
      break;
    case PD_18V:
      Serial.println("18V");
      break;
    case PD_20V:
      Serial.println("20V");
      break;
    default:
      Serial.println("Unknown voltage setting");
      break;
  }

  // What is the max current available right now?
  HUSB238_CurrentSetting srcCurrent = husb238.getPDSrcCurrent();
  Serial.print("Source Current: ");
  printCurrentSetting(srcCurrent);
  Serial.println();

  // What voltages and currents are available from this adapter?
  Serial.println("Available PD Voltages and Current Detection Test:");
  for (int i = PD_SRC_5V; i <= PD_SRC_20V; i++) {
    bool voltageDetected = husb238.isVoltageDetected((HUSB238_PDSelection)i);

    switch ((HUSB238_PDSelection)i) {
      case PD_SRC_5V:
        Serial.print("5V");
        break;
      case PD_SRC_9V:
        Serial.print("9V");
        break;
      case PD_SRC_12V:
        Serial.print("12V");
        break;
      case PD_SRC_15V:
        Serial.print("15V");
        break;
      case PD_SRC_18V:
        Serial.print("18V");
        break;
      case PD_SRC_20V:
        Serial.print("20V");
        break;
      default:
        continue;
    }
    Serial.print(voltageDetected ? " Available" : " Unavailable");

    // Loop over currents if voltage is detected
    if (voltageDetected) {
      HUSB238_CurrentSetting currentDetected = 
husb238.currentDetected((HUSB238_PDSelection)i);
      Serial.print(" - Max current: ");
      printCurrentSetting(currentDetected);
    }
    Serial.println();
  }

  // Override whatever the jumpers on the board say, and get a specific voltage!
  husb238.selectPD(PD_SRC_5V);  // Select 5V
  // Uncomment one of the following lines to select a different PD:
  // husb238.selectPD(PD_SRC_9V);  // Select 9V
  // husb238.selectPD(PD_SRC_12V);  // Select 12V
  // husb238.selectPD(PD_SRC_15V);  // Select 15V
  // husb238.selectPD(PD_SRC_18V);  // Select 18V
  // husb238.selectPD(PD_SRC_20V);  // Select 20V

  // Perform the actual PD voltage request!
  husb238.requestPD();

  // Test getSelectedPD function
  HUSB238_PDSelection selectedPD = husb238.getSelectedPD();
  Serial.print("Currently Selected PD Output: ");
  switch (selectedPD) {
    case PD_NOT_SELECTED:
      Serial.println("Not Selected");
      break;
    case PD_SRC_5V:
      Serial.println("5V");
      break;
    case PD_SRC_9V:
      Serial.println("9V");
      break;
    case PD_SRC_12V:
      Serial.println("12V");
      break;
    case PD_SRC_15V:
      Serial.println("15V");
      break;
    case PD_SRC_18V:
      Serial.println("18V");
      break;
    case PD_SRC_20V:
      Serial.println("20V");
      break;
    default:
      Serial.println("Unknown");
      break;
  }
}


void printCurrentSetting(HUSB238_CurrentSetting srcCurrent) {
  switch (srcCurrent) {
    case CURRENT_0_5_A:
      Serial.print("0.5A ");
      break;
    case CURRENT_0_7_A:
      Serial.print("0.7A ");
      break;
    case CURRENT_1_0_A:
      Serial.print("1.0A ");
      break;
    case CURRENT_1_25_A:
      Serial.print("1.25A ");
      break;
    case CURRENT_1_5_A:
      Serial.print("1.5A ");
      break;
    case CURRENT_1_75_A:
      Serial.print("1.75A ");
      break;
    case CURRENT_2_0_A:
      Serial.print("2.0A ");
      break;
    case CURRENT_2_25_A:
      Serial.print("2.25A ");
      break;
    case CURRENT_2_50_A:
      Serial.print("2.50A ");
      break;
    case CURRENT_2_75_A:
      Serial.print("2.75A ");
      break;
    case CURRENT_3_0_A:
      Serial.print("3.0A ");
      break;
    case CURRENT_3_25_A:
      Serial.print("3.25A ");
      break;
    case CURRENT_3_5_A:
      Serial.print("3.5A ");
      break;
    case CURRENT_4_0_A:
      Serial.print("4.0A ");
      break;
    case CURRENT_4_5_A:
      Serial.print("4.5A ");
      break;
    case CURRENT_5_0_A:
      Serial.print("5.0A ");
      break;
    default:
      break;
  }
}
```

Upload the sketch to your board and open up the Serial Monitor (Tools -> Serial Monitor) at 115200 baud. You'll see the HUSB238 recognized over I2C by the code. It will set the voltage to 5V over I2C. Then, it will query the attached PD adapter to see which voltage and current combinations are available and print the currently set voltage.



```
Adafruit HUSB238 Test Sketch
HUSB238 initialized successfully.
---------------------------------------------
Attachment Status: Attached
CC Direction: CC1 connected
USB PD query response: Success
5V Contract Voltage: 5V & Current: 3A
Source Voltage: 5V
Source Current: 3.0A
Available PD Voltages and Current Detection Test:
5V Available - Max current: 3.0A
9V Available - Max current: 3.0A
12V Available - Max current: 3.0A
15V Available - Max current: 3.0A
18V Unavailable
20V Available - Max current: 5.0A
Currently Selected PD Output: 5V
---------------------------------------------
Attachment Status: Attached
CC Direction: CC1 connected
USB PD query response: Success
5V Contract Voltage: 5V & Current: 3A
```

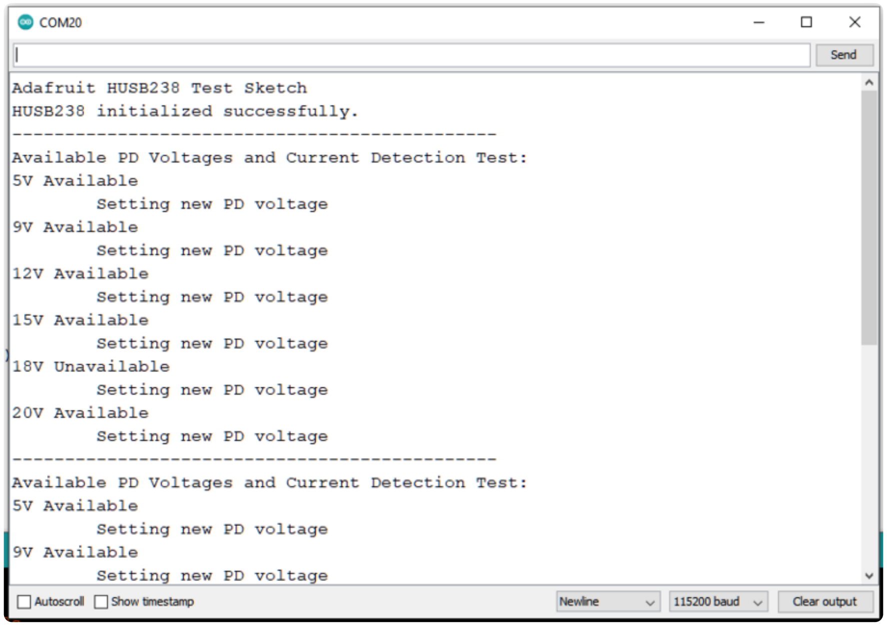
Test All Voltages Example

For this example, it's best to test with the output from the breakout connected to a multimeter in DC voltage mode. DC voltage mode is labeled with a V and two lines, one dashed and one solid. For information on using a multimeter, check out this guide. ()

Multimeters Learn Guide

```
#include <Wire.h>
#include "Adafruit_HUSB238.h"

Adafruit_HUSB238 husb238;

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);
  Serial.println("Adafruit HUSB238 Test Sketch");

  // Initialize the HUSB238
  if (husb238.begin(HUSB238_I2CADDR_DEFAULT, &Wire)) {
    Serial.println("HUSB238 initialized successfully.");
  } else {
    Serial.println("Couldn't find HUSB238, check your wiring?");
    while (1);
  }
}
```

```
void loop() {
  delay(1000);  // Add a delay to prevent flooding the serial output
  Serial.println(F("----------------------------------------------"));

  if (! husb238.isAttached())
    return;

  if (husb238.getPDResponse() != SUCCESS)
    return;

  // What voltages and currents are available from this adapter?
  Serial.println("Available PD Voltages and Current Detection Test:");
  for (int i = PD_SRC_5V; i <= PD_SRC_20V; i++) {
    bool voltageDetected = husb238.isVoltageDetected((HUSB238_PDSelection)i);

    switch ((HUSB238_PDSelection)i) {
      case PD_SRC_5V:
        Serial.print("5V");
        break;
      case PD_SRC_9V:
        Serial.print("9V");
        break;
      case PD_SRC_12V:
        Serial.print("12V");
        break;
      case PD_SRC_15V:
        Serial.print("15V");
        break;
      case PD_SRC_18V:
        Serial.print("18V");
        break;
      case PD_SRC_20V:
        Serial.print("20V");
        break;
      default:
        continue;
    }
    Serial.println(voltageDetected ? " Available" : " Unavailable");

    Serial.println("\tSetting new PD voltage");
    // Change to that voltage
    husb238.selectPD((HUSB238_PDSelection)i);
    // Perform the actual PD voltage request!
    husb238.requestPD();

    delay(2000);
  }
}
```



```
Adafruit HUSB238 Test Sketch
HUSB238 initialized successfully.
---------------------------------------------
Available PD Voltages and Current Detection Test:
5V Available
        Setting new PD voltage
9V Available
        Setting new PD voltage
12V Available
        Setting new PD voltage
15V Available
        Setting new PD voltage
18V Unavailable
        Setting new PD voltage
20V Available
        Setting new PD voltage
---------------------------------------------
Available PD Voltages and Current Detection Test:
5V Available
        Setting new PD voltage
9V Available
        Setting new PD voltage
```

Upload the sketch to your board and open up the Serial Monitor (Tools -> Serial Monitor) at 115200 baud. You'll see the HUSB238 recognized over I2C by the code. Then, it will try setting a new voltage one after the other. As the Serial Monitor updates, you should see the same voltage read by your multimeter.

Arduino Docs

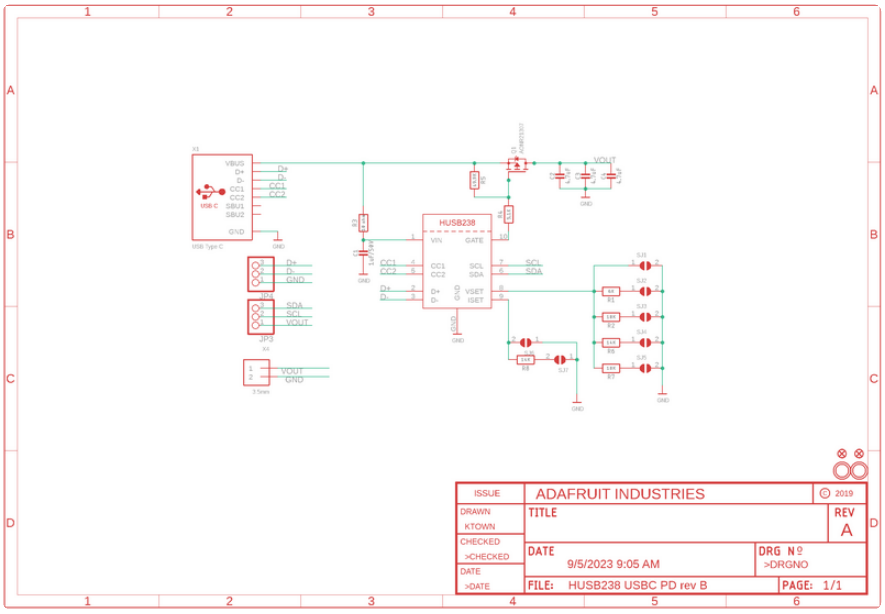
Arduino Docs ()

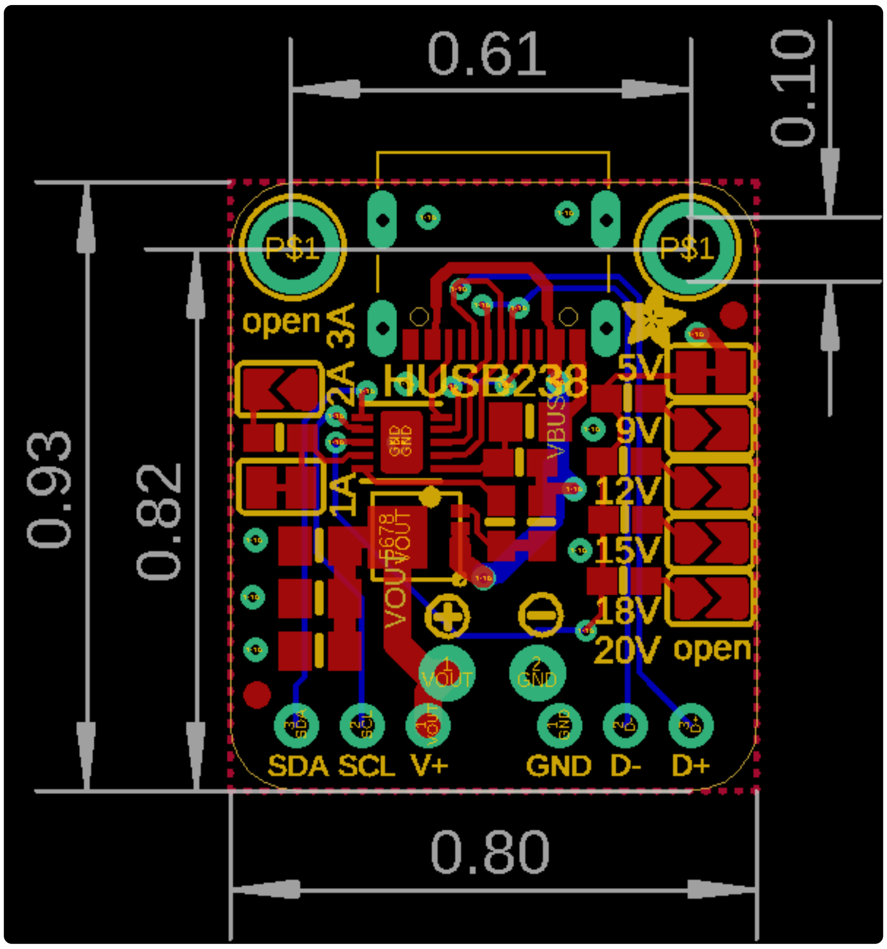
Downloads

Files

- HUSB238 Datasheet ()
- EagleCAD PCB Files on GitHub ()
- 3D models on GitHub ()
- Fritzing object in the Adafruit Fritzing Library ()

Schematic and Fab Print





0.61
0.10
0.93
0.82
0.80
P$1
P$1
open
3A
2A
1A
HUSB238
VBUS
5V
9V
12V
15V
18V
20V
open
VOUT
GND
SDA
SCL
V+
GND
D-
D+