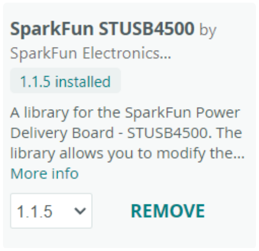
USB-PowerDelivery: Write

This project involves configuring a USB Power Delivery (PD) sink using the `STUSB4500` chip to request specific voltage and current profiles from a USB-C PD charger. Using the `SparkFun` `STUSB4500` `Arduino` library, also a modified version of the example sketch to set custom parameters for Profile 2 and Profile 3. Profile 1 remains 5V for safety. The configuration will be written to the chip's non-volatile memory (`NVM`) using `I2C` communication at address `0x2B`. The default address won't work since the address has been modified by design.

1. Install the necessary libraries from Arduino IDE.

- Open the `Arduino` IDE.
- Go to the menu: **Sketch** → **Include Library** → **Manage Libraries...**
- In the search bar, type "`SparkFun STUSB4500`".
- Select "`SparkFun STUSB4500`", choose the latest version, and click **Install**.



SparkFun STUSB4500 by SparkFun Electronics...

1.1.5 installed

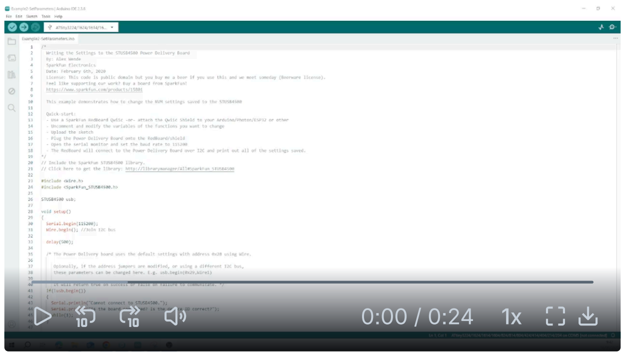
A library for the SparkFun Power Delivery Board - STUSB4500. The library allows you to modify the... More info

The library allows communication with the `STUSB4500` over `I2C` to:

- Read and write configuration settings.
- Set power profiles (voltages and currents).
- Store settings in non-volatile memory (`NVM`).
- Easily integrate `USB PD` negotiation into `Arduino` projects.

---

2. Open the example Example2-SetParameters.



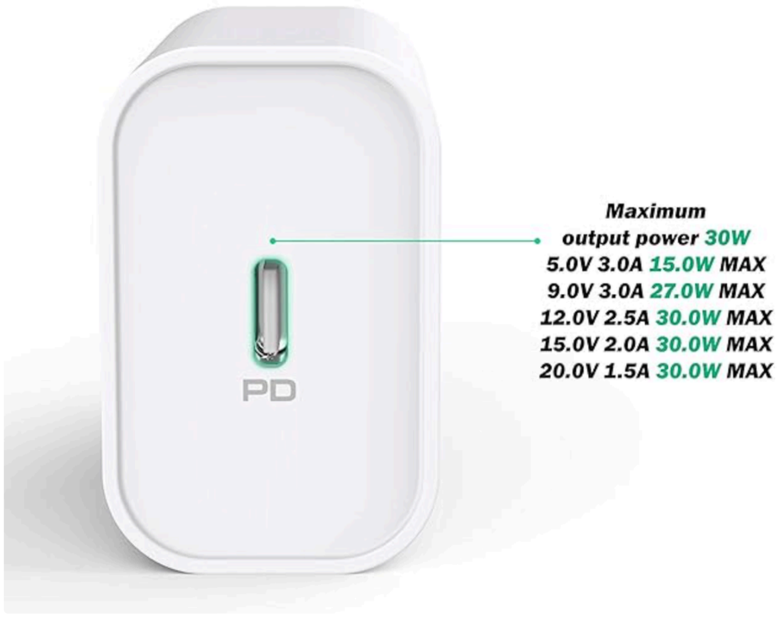
Example2-SetParameters | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ATtiny3224/1624/1614/16...
Example2-SetParameters.ino
0:00 / 0:24
1x

```
/*
 Writing the Settings to the STUSB4500 Power Delivery Board
 By: Alex Wende
 SparkFun Electronics
 Date: February 6th, 2020
 License: This code is public domain but you buy me a beer if you use this and we meet someday (Beerware license).
 Feel like supporting our work? Buy a board from SparkFun!
 https://www.sparkfun.com/products/15801

 This example demonstrates how to change the NVM settings saved to the STUSB4500

 Quick-start:
 - Use a SparkFun RedBoard Qwiic -or- attach the Qwiic Shield to your Arduino/Photon/ESP32 or other
 - Uncomment and modify the variables of the functions you want to change
 - Upload the sketch
 - Plug the Power Delivery Board onto the RedBoard/shield
 - Open the serial monitor and set the baud rate to 115200
 - The RedBoard will connect to the Power Delivery Board over I2C and print out all of the settings saved.
*/
// Include the SparkFun STUSB4500 library.
// Click here to get the library: http://librarymanager/All#SparkFun_STUSB4500

#include <Wire.h>
#include <SparkFun_STUSB4500.h>

STUSB4500 usb;

void setup()
{
  Serial.begin(115200);
  Wire.begin(); //Join I2C bus

  delay(500);

  /* The Power Delivery board uses the default settings with address 0x28 using Wire.

     Opionally, if the address jumpers are modified, or using a different I2C bus,
     these parameters can be changed here. E.g. usb.begin(0x29,Wire1)

     It will return true on success or false on failure to communicate. */
  if(!usb.begin())
  {
    Serial.println("Cannot connect to STUSB4500.");
    Serial.println("Is the board connected? Is the device ID correct?");
    while(1);
```

---

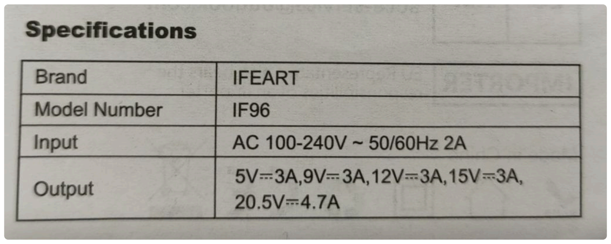
3. Take one USB PD charger with different profiles

A suitable charger was chosen and several features were taken into account, such as the reviews of the product, power delivery (PD) compatible for fast charge and the capacity to use different loading profiles (also in the charger) as it should provide with different power configurations. Something like the following screenshot:



4. Configuring Custom NVM Power Delivery Settings

⚠ It is necessary to know about the charger output values, after that, change the default values, as the following code. See the tested charger and the code adapted.



Specifications

Brand	IFEART
Model Number	IF96
Input	AC 100-240V ~ 50/60Hz 2A
Output	5V⎓3A,9V⎓3A,12V⎓3A,15V⎓3A, 20.5V⎓4.7A

Code:

```
#include <Wire.h>
#include <SparkFun_STUSB4500.h>

STUSB4500 usb;
const int VOLTAGE_LIMIT = 20; //20%

void setup()
{
  Serial.begin(115200);
  Wire.begin(); //Join I2C bus

  delay(500);

  /* The Power Delivery board uses the default settings with address 0x28 using Wire.

     Opionally, if the address jumpers are modified, or using a different I2C bus,
     these parameters can be changed here. E.g. usb.begin(0x29,Wire1)

     It will return true on success or false on failure to communicate. */
  if(!usb.begin(0x2B, Wire))
  {
    Serial.println("Cannot connect to STUSB4500.");
    Serial.println("Is the board connected? Is the device ID correct?");
    while(1);
  }

  Serial.println("Connected to STUSB4500!");
  delay(100);

  /* Set Number of Power Data Objects (PDO) 1-3 */
  usb.setPdoNumber(3);

  /* PDO1
   - Voltage fixed at 5V
   - Current value for PDO1 0-5A, if 0 used, FLEX_I value is used
   - Under Voltage Lock Out (setUnderVoltageLimit) fixed at 3.3V
   - Over Voltage Lock Out (setUpperVoltageLimit) 5-20%
  */
  usb.setCurrent(1, 3.0);
  usb.setUpperVoltageLimit(1, VOLTAGE_LIMIT);

  /* PDO2
   - Voltage 5-20V
   - Current value for PDO2 0-5A, if 0 used, FLEX_I value is used
   - Under Voltage Lock Out (setUnderVoltageLimit) 5-20%
   - Over Voltage Lock Out (setUpperVoltageLimit) 5-20%
  */
  usb.setVoltage(2, 9.0);
  usb.setCurrent(2, 3.0);
  usb.setLowerVoltageLimit(2, VOLTAGE_LIMIT);
  usb.setUpperVoltageLimit(2, VOLTAGE_LIMIT);

  /* PDO3
   - Voltage 5-20V
   - Current value for PDO3 0-5A, if 0 used, FLEX_I value is used
   - Under Voltage Lock Out (setUnderVoltageLimit) 5-20%
   - Over Voltage Lock Out (setUpperVoltageLimit) 5-20%
  */
  usb.setVoltage(3, 12.0);
  usb.setCurrent(3, 3.0);
  usb.setLowerVoltageLimit(3, VOLTAGE_LIMIT);
  usb.setUpperVoltageLimit(3, VOLTAGE_LIMIT);

  /* Write and save settings to STUSB4500 */
  usb.write();

  /* Read the NVM settings to verify the new settings are correct */
  usb.read();

  Serial.println("\New parameters:\n");

  for (int i = 1; i <= 3; i++) {
    Serial.print("PDO"); Serial.print(i); Serial.println(":");
    Serial.print("  Voltage (V): "); Serial.println(usb.getVoltage(i));
    Serial.print("  Current (A): "); Serial.println(usb.getCurrent(i));
    Serial.print("  Lower Voltage Tolerance (%): "); Serial.println(usb.getLowerVoltageLimit(i));
    Serial.print("  Upper Voltage Tolerance (%): "); Serial.println(usb.getUpperVoltageLimit(i));
    Serial.println();
  }

  Serial.print("Flex Current: "); Serial.println(usb.getFlexCurrent());
  Serial.print("External Power: "); Serial.println(usb.getExternalPower());
  Serial.print("USB Communication Capable: "); Serial.println(usb.getUsbCommCapable());
  Serial.print("Config OK GPIO: "); Serial.println(usb.getConfigOkGpio());
  Serial.print("GPIO Control: "); Serial.println(usb.getGpioCtrl());
  Serial.print("Power Only Above 5V: "); Serial.println(usb.getPowerAbove5vOnly());
  Serial.print("Request Source Current: "); Serial.println(usb.getReqSrcCurrent());
}

void loop()
{
}
```

Code explanation:

Included libraries

```
#include <Wire.h>
#include <SparkFun_STUSB4500.h>
```

- `Wire.h`: used for `I2C` communication.
- `SparkFun_STUSB4500.h`: provides functions to interface with the `STUSB4500` chip.

`STUSB4500` configuration

```
STUSB4500 usb;
const int VOLTAGE_LIMIT = 20; // 20%
```

- `usb`: creates an object to interact with the chip.
- `VOLTAGE_LIMIT`: sets voltage margin (tolerance) to ±20%.

`setup()` function

1. Initialization

```
Serial.begin(115200);
Wire.begin(); // Start I2C delay(500);
```

- `Serial.begin`: starts the serial communication between the `Arduino` and the computer at a baud rate of 115200 bits per second.
- `Wire.begin`: initializes the `I2C` communication.

2. Connect to STUSB4500

```
if(!usb.begin(0x2B, Wire))
```

- Tries to connect to the chip at `I2C` address `0x2B`.
- If it fails, prints error and halts (`while(1);`).

3. Set up PDOs (Power Data Objects)

Before profiles:

```
usb.setPdoNumber(3);
```

- `usb.setPdoNumber(3)`: configures a `PDO` setting.

Profile 1:

Profile one is always kept to 5V, so the only parameters that are changed are current and upper voltage limit.

```
usb.setCurrent(1, 3.0);
usb.setUpperVoltageLimit(1, VOLTAGE_LIMIT);
```

Profile 2:

```
usb.setVoltage(2, 9.0);
usb.setCurrent(2, 3.0);
usb.setLowerVoltageLimit(2, VOLTAGE_LIMIT);
usb.setUpperVoltageLimit(2, VOLTAGE_LIMIT);
```

Profile 3:

```
usb.setVoltage(3, 12.0);
usb.setCurrent(3, 3.0);
usb.setLowerVoltageLimit(3, VOLTAGE_LIMIT);
usb.setUpperVoltageLimit(3, VOLTAGE_LIMIT);
```

- `setVoltage(profile, voltage)`: sets the voltage for the specified profile.
- `setCurrent(profile, current)`: sets the amperage for the specified profile.
- `setLowerVoltageLimit(profile, limit)`: sets the lower voltage tolerance limit for the specified profile.
- `setUpperVoltageLimit(profile, limit)`: sets the upper voltage tolerance limit for the specified profile.

Write and verify configuration

```
usb.write();
usb.read();
```

- `write`: writes (saves) the configured settings to the non-volatile memory (`NVM`) of the `STUSB4500` chip.
- `read`: reads back the settings from the chip's non-volatile memory.

Print settings to serial

This part prints out the configured values:

```
for (int i = 1; i <= 3; i++) {   Serial.print("PDO"); ... }
```

And then reads other chip parameters:

```
Serial.print("Flex Current: "); Serial.print("External Power: "); ...
```

`loop()` function

```
void loop() {}
```

- Empty. This is a setup-only configuration sketch.

---

5. Plug USB PD charger to USB C connector in board and check voltage

To do this, the needle probes need to be placed on pins 21 (negative) and 22 (positive) of the board.