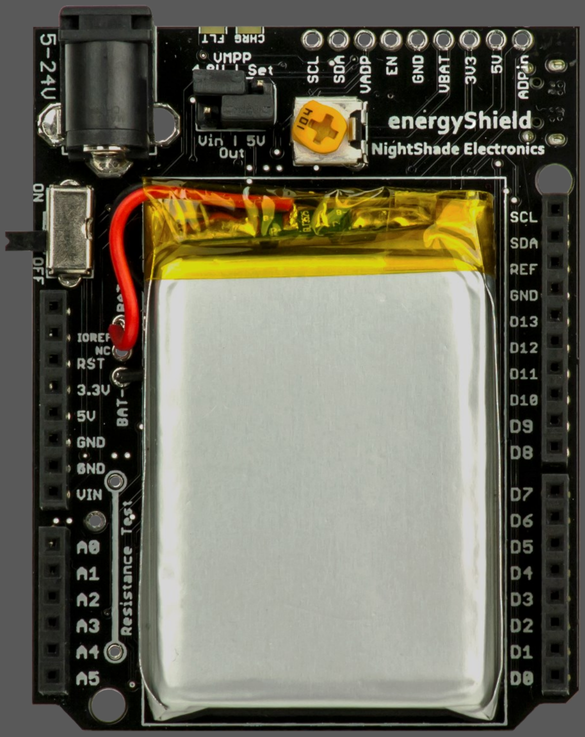
energyShield

by NightShade Electronics

ns-electric.com

Description

The energyShield is a rechargeable lithium-polymer power supply in the form of an Arduino shield or breakout board. On board, it has a lithium-polymer battery pack, an integrated charge controller, a 5V boost regulator, a 3.3V Linear LDO regulator, and an input Buck converter. The charge controller throttles the current to maintain a threshold input voltage. This allows for efficient solar charging.



Features

- 5V and 3.3V Output
- 1200mAh Capacity
- Uninterruptable Power Supply
- Multiple Charging Options
    - Solar (with adjustable MPP)
    - USB
    - Adapter
- Fuel-gauge IC
    - I2C Interface
    - Battery Charge/Discharge Current
    - Battery Voltage

Specifications

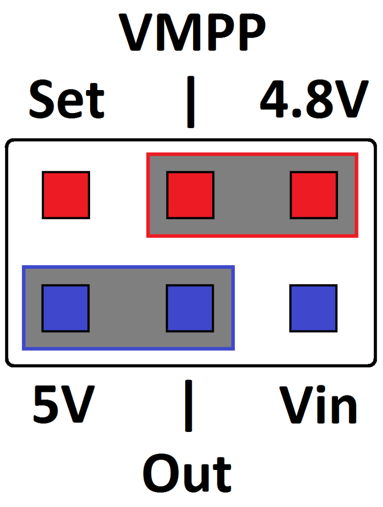
- 1200 mAh 1.5C Discharge Rate Li-Po Cell
- 5V Output: 1000 mA Max
- 3.3V Output: 300 mA Max
- Barrel Jack Input: Solar or Wall Adapter 5 - 24V
- Length: 69mm
- Width: 53.4mm
- Height: 23.4 (with pins) 19 (w/o pins)
- Mass: 52g (Shield Configuration)

Quick Start Guide

This Quick Start Guide is intended for those using an UNO R3 or Leonardo and non-solar charging methods. For solar charging or other advanced configurations, please refer to the "Advanced Configuration" section on Page 5.

energyShield

1. Follow the assembly guide on Page 4.
2. Configure the jumpers as seen in the picture to the right.
3. Charge by plugging an adapter into the barrel jack or a micro USB cable into the micro USB socket. (Charge is complete when indicator light is OFF)
4. Plug into an Arduino using the shield headers and turn the switch to ON.
5. Enjoy having no power wires!



Fuel Gauge

1. Download energyShield library from the NightShade Website.
2. Extract the contents of the ZIP file into the libraries folder located in your project folder.
3. Call the NS_energyShield.h and Wire.h libraries, and declare an object for the shield at the beginning of your sketch. For this example we named the object "eS", but you can call it whatever you would like to.

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

NS_energyShield eS;
```

4. At this point you are able to call any of the functions listed on Page 3 in your program. For example, in the program above you could call the function "eS.voltage()" and the function would return the battery voltage in mV as an integer. For a quick way to check your fuel gauge functionality, or to see an example program using this code, look at the example programs included with the library.

Communicating with the Fuel Gauge using the NS_energyShield library

NS_energyShield <object>

Function: Defines NS_energyShield object <object>

Returns: nothing

<object>.voltage()

Function: Reads the battery voltage from the fuel gauge

Returns: [int] Voltage in mV

<object>.current()

Function: Reads current charging (positive) or discharging (negative) the battery

Returns: [int] Current in mA

<object>.percent()

Function: Reads the percent of charge remaining in the battery

Returns: [int] Percent of charge in 0.5% increments (2 * Percent Charge)

<object>.temperature()

Function: Reads the temperature from the fuel gauge

Returns: [int] Temperature in 0.125 ^{o}C increments (8 * Temperature)

<object>.address()

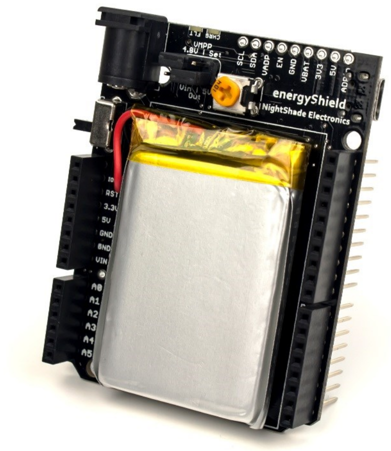
Function: Changes the target address used by the library. Only use this function if you have changed the address in the fuel gauge register. (This is an advanced operation, and most people won't need it.)

Returns: nothing

For more information and advanced I^{2}C/TWI functionality refer to the DS2786B datasheet**.**

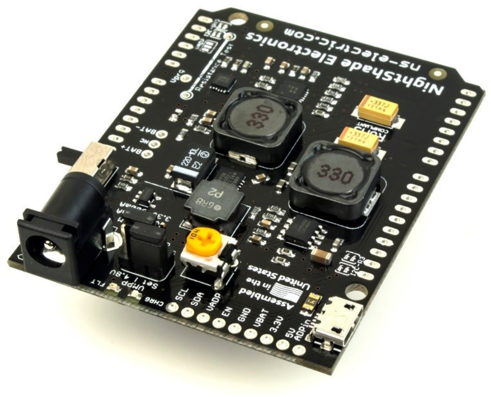
Assembly Guide

The energyShield comes with all of the surface mount components assembled and the through hole components kitted. Depending on your needs, you can solder the through hole components onto either the top or the bottom of the energyShield. Below is a list of the included parts. Before soldering the battery, please refer to the section below called "**Soldering the Battery**".



What's Included:

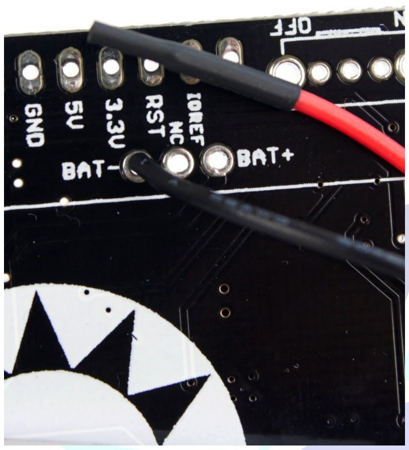
1x	energyShield
1x	1200mAh Lithium Polymer Battery
1x	Barrel Jack
1x	Potentiometer
2x	8-Pin Stacking Header
1x	6-Pin Stacking Header
1x	10-Pin Stacking Header
1x	2x3 Male Header
3x	Black Jumper
1x	Double-Sided Foam Adhesive Pad



All of the through hole components can be soldered either onto the top or bottom of the board. The two different configurations are shown in the pictures to the right. Soldering all of the components onto the top of the board (the side with the logo) allows it to be used as an Arduino shield, as shown by the top right picture. For those who do not wish to use this board as a shield, it may be preferable to solder the components into the bottom of the board, as shown in the middle right picture. If you solder the through hole components on the bottom, then you are able to solder male header pins facing down into the breakout header and plug the board into a breadboard.

Soldering the Battery

When soldering the battery, take special precaution to not let the leads short (contact each other). For this reason, in the battery connection area, you will see three holes: Bat+, Bat-, and NC. The hole labeled NC is not electrically connected to anything and, therefore, acts as a buffer space between the Bat+ and Bat- pins.



1. Before soldering the battery into place, especially for top assembly, we recommend trimming leads to roughly 25mm (1 inch).
2. Strip the wires to expose about 3mm (1/8 inch) of copper and place a piece of tape (or other insulator) over the positive lead to insulate it.
3. Solder the negative lead to the PCB. Ensure that there is no bridging to the adjacent terminals. Trim the end of the lead to a nice length.
4. Remove the tape from the positive lead and solder it into place, being careful to not touch it to anything but the positive terminal. Trim the end of the lead to a nice length.

If desired after the soldering is completed, the foam adhesive pad can be used to fasten the battery in place.

Before putting away your soldering iron, be sure to check the Page 7 for any solder junctions you want to enable.

Advanced Configuration

1. **Configure the solder junctions appropriately for your application based on the explanation on the Hardware page.**

2. **Place the two jumpers on the 6 pin header according to your needs.**

"Out" Jumper	Selects which pin the energyShield outputs the 5V net to. The options are "Vin" and "5V". The Arduino UNO (All revisions), Leonarndo, and Duemilanove all receive power on the 5V pin, and this jumper needs to be set to the "5V" option. If you put the option on the "Vin" option for these boards, you will be running 5V to the 7805 voltage regulator on the Arduino, which needs about 7V to properly operate. The result will be an unpowered Arduino. Some boards, such as the Intel Galileo require a regulated 5V input on the Vin pin. For the Galileo, and similar boards, the "Out" Jumper must be set to "Vin".
"**VMPP" Jumper**	This jumper selects the voltage threshold at which the charge regulator starts decreasing the charge current in order to keep the input voltage above the threshold. This jumper only applies to the barrel jack input. The two options for this jumper are "4.8V" and "Set". The "4.8V" option is a fixed 4.8V threshold which is intended for use with a power adapter. The "Set" option allows you to set the threshold voltage base on the max power point voltage of your solar cell (more detail below). Use the "Set" option if you are planning to use a solar cell.

3. **Setting the proper VMPP** (Only needed for solar charging.)

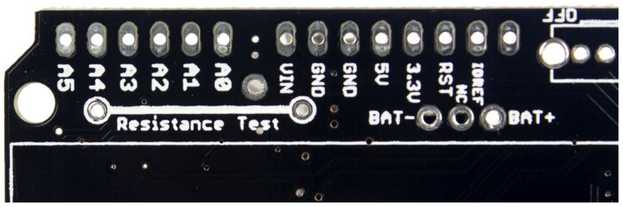
The **VMPP**, or **V**oltage at **M**ax **P**ower **P**oint, is the voltage at which your solar panel will produce the most amount of power. If you pull too much current from a solar panel, it's output voltage will drop below its VMPP, and you will not be using it at full efficiency. For this reason, the energyShield will regulate the current it draws to not drop below the threshold VMPP voltage that you set. In reality, the maximum power point varies slightly based on many factors. For this product's use we assume a constant voltage max power point. This value is typically given in the specification of your solar panel. For more information on VMPP, check out this Page on Wikipedia.

To set the VMPP you will need a multi-meter and a small screwdriver. You can calculate the proper resistance you will need to set the potentiometer to based on the equation below. The two variables you need to use are R_{pot} and V_{mpp}. The "Ω" and "V" are merely units, not variables.

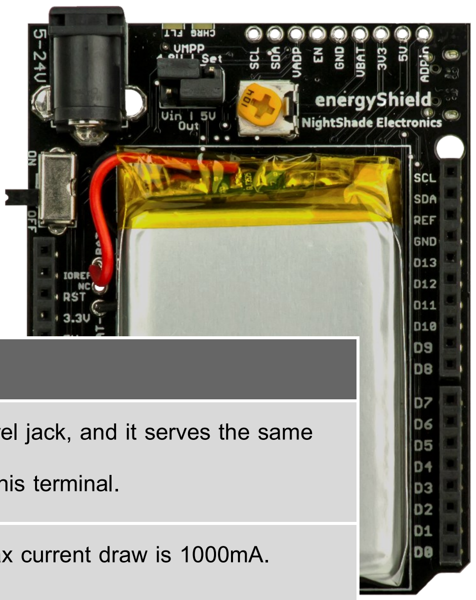
$$\boldsymbol{R_{pot}} = 4100\frac{\Omega}{V} * \boldsymbol{V_{mpp}} - 8200\Omega$$

To set the potentiometer:

1. Remove the shunt/jumper from the VMPP selection (Set/4.8V) header.
2. Set your multi-meter to 20k Omhs (Ω) and place the probes in the test points marked "Resistance Test."
3. Use a screw driver or tuner tool to adjust the potentiometer until the correct resistance is reached.
4. Remove the probes and replace the shunt on the "Set" position of the header to enable your newly set VMPP.



Interfacing with the energyShield



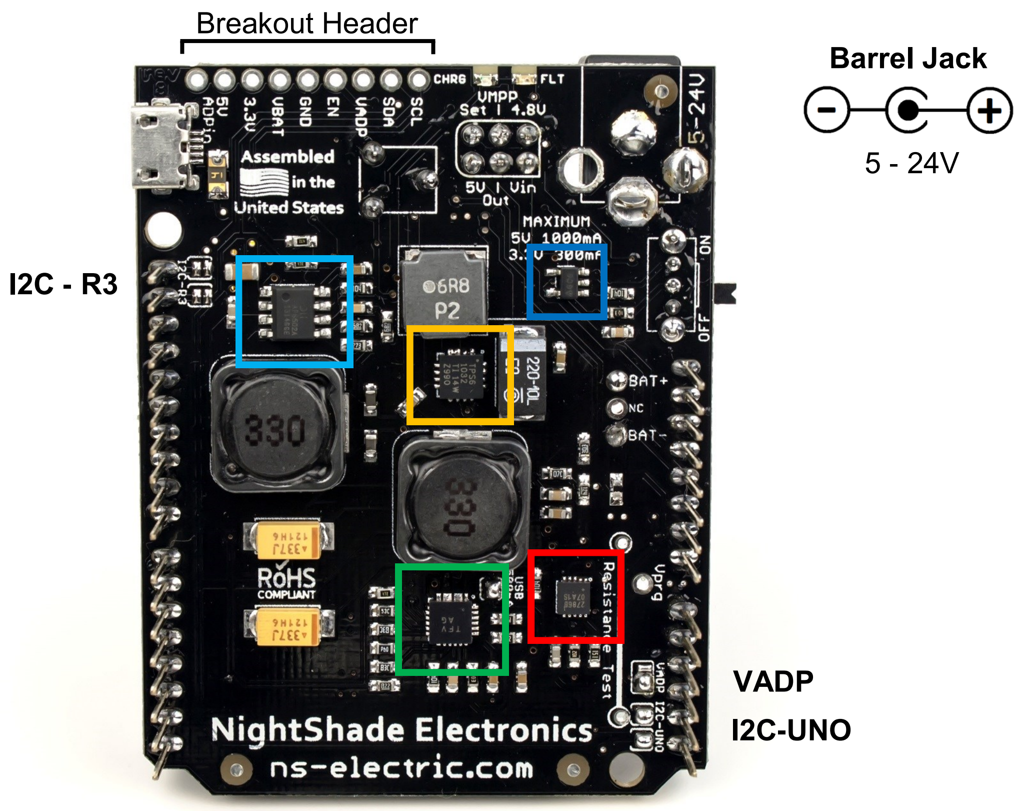
There are two ways to interface with the energyShield: the Arduino header and the Breakout. Both options have the exact same electrical connections (with the exception of the EN pin; see below), and it is up to you to decide which works best for your application. The energyShield breakout is intended for those who wish to interface the energyShield with a development board other than the standard form factor Arduino. It provides complete access to all of the features of the energyShield. Below is a list of the pins and their function. On Page 8 is a functional block diagram showing where the pins connect.

Pin	Function
ADPin	Adapter In. This pin is connected to the same net as the barrel jack, and it serves the same purpose. A solar panel or wall adapter can be connected to this terminal.
5V	5V Output. This is the main power output from the shield. Max current draw is 1000mA.
3.3V	3.3V Output. Max current draw is 300mA.
VBAT	Raw Battery Voltage. This pin can be used if you desire to draw power directly from the battery before it goes through a regulator.
GND	Ground. Use this as the ground point for the header terminals. This is the same net as the ground pins on the Arduino headers.
EN	Enable Pin. This pin allows you to externalize the function of the switch on the energyShield. Connect this pin to VBAT to enable 5V and 3.3V. Connect this pin to GND to disable the 5V and 3.3V. The current required to pull up or pull down this pin is <0.5mA. Therefore, it is a logic level input.
VADP	Adapter Voltage. This pin is a reference point to read what the ADPin voltage is. It's value is ADPin/5, which is scaled using a simple voltage divider circuit with two 1% tolerance resistors. The pin is scaled to 1/5th of the true voltage so it can be read by a standard analog pin of an Arduino. Use the "VADP" solder junction described on Page 7 to connect this net to the A2 or A3 pin on the Arduino header, or access it on the breakout header.
SDA & SCL	I2C Data Lines. Connect these to your microcontroller to communicate with the fuel gauge on the energyShield. For non-R3 Arduinos, use the solder junctions to attach SDA and SCL to A4 and A5, respectively.

Hardware

Chips

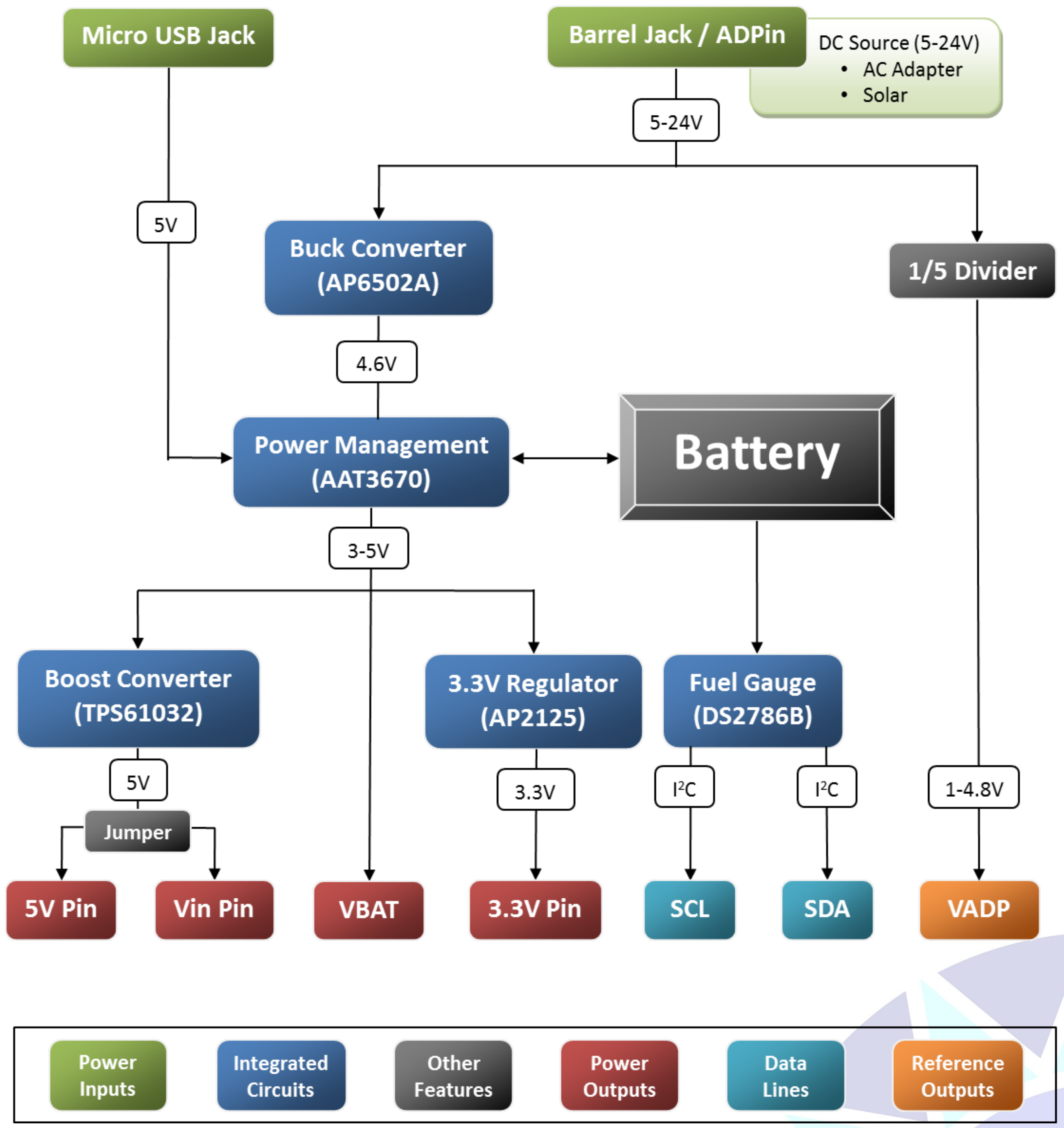
1.	Buck	Diodes AP6502A
2.	Boost	Texas Instruments TPS61032PWP
3.	Charger	Skyworks ATT3670
4.	Fuel Gauge	Maxim DS2786B
5.	3.3V Regulator	Diodes - AP2125K-3.3TRG1



Solder Junctions

I2C - R3	Two normally closed solder junctions that connect the fuel gauge I2C communication lines (SDA and SCL) to the “R3” Arduino I2C lines found on the Leonardo and the UNO R3.
VADP	A two-way solder junction to connect VADP reference voltage to either pin A2 or A3. Connect by placing a bead of solder from the middle pad to either the left or right pad, whichever is closer to the desired input (A2 or A3).
I2C-UNO	Two normally open solder junctions to connect the fuel gauge I2C lines to A4 and A5. Use this option when using the UNO R1, R2, or the Duemilanove.
USB 500mA	**Open:** (Default) USB charging is limited to 100mA, making it safe to charge with any USB port.
	Closed: Enables USB charging at 500mA. This requires a USB 2.0 or 3.0. By enabling this feature you risk overdrawing and causing damage to your USB power source. Use at your own discretion.

Functional Block Diagram


Micro USB Jack
Barrel Jack / ADPin
DC Source (5-24V)
• AC Adapter
• Solar
5-24V
5V
Buck Converter (AP6502A)
1/5 Divider
4.6V
Power Management (AAT3670)
Battery
3-5V
Boost Converter (TPS61032)
3.3V Regulator (AP2125)
Fuel Gauge (DS2786B)
5V
Jumper
3.3V
I²C
I²C
1-4.8V
5V Pin
Vin Pin
VBAT
3.3V Pin
SCL
SDA
VADP
Power Inputs
Integrated Circuits
Other Features
Power Outputs
Data Lines
Reference Outputs