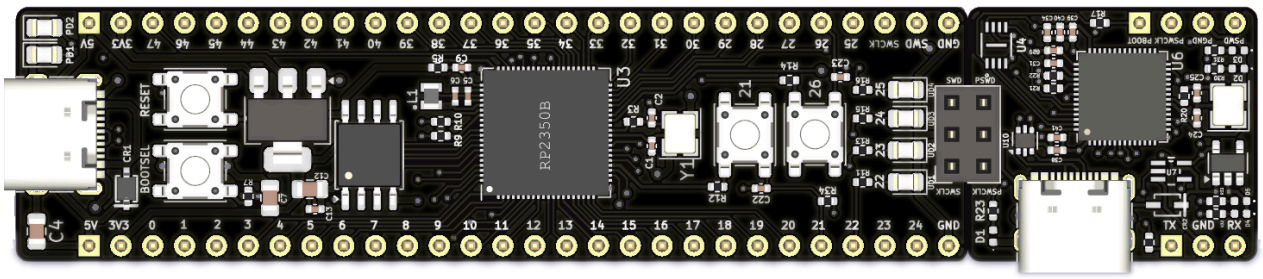
Purdue Proton

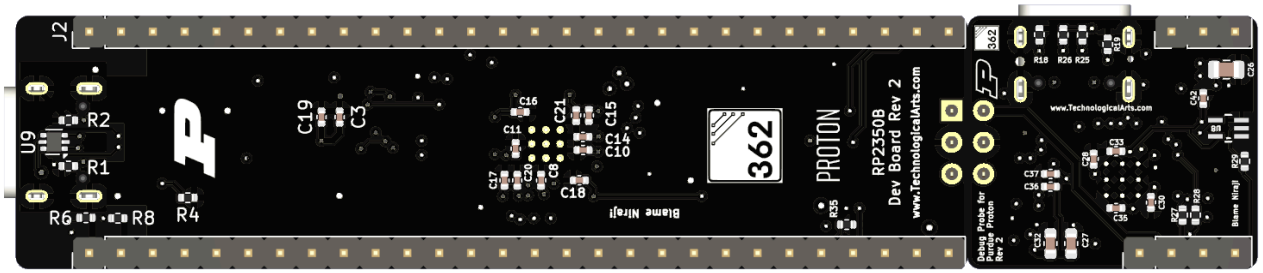
A RP2350B Development Board with Debugger

Features

- RP2350B microcontroller
    - 520 kB SRAM
    - QFN-80 package
- 16 MB quad SPI flash memory
- Debugger based on Raspberry Pi Debug Probe design
- USB-C connector on Proton board and debugger with ESD protection diodes
- External 12 MHz crystal produces 150 MHz with onboard PLL
- External 5V, 3.3V, and GND pins
- Power indicator LEDs
    - 5V is red
    - 3V3 is green
- 48 GPIO pins, SWCLK/SWD pins
- Push buttons for bootloader and reset
- Two hardware-debounced push buttons connected to GP21 and GP26
- Four user LEDs connected to GP22 (red), GP23 (yellow), GP24 (green), GP25 (blue)



Front of board



Back of board

Quick Start

1. Download and install VSCode.
2. Install the PlatformIO extension.
3. Create a new folder, and in it, create a file called “platformio.ini” with the following contents:

```
[env:proton]
platform =
https://github.com/norandomtechie/platform-raspberrypi#feature/proton
-picosdk-support
board = proton
framework = picosdk
build_src_flags = -O0
debug_tool = picoprobe
upload_protocol = picoprobe
monitor_speed = 115200
```

4. Create a folder “src” in the newly created folder, and in it, create a file “main.c”:

```
#include <stdio.h>
#include "pico/stdlib.h"

int main() {
    stdio_init_all();
    gpio_init_mask(0xf << 22);  // LEDs
    gpio_set_dir_masked(0xf << 22, 0xf << 22);  // Set LEDs to output
    gpio_put_masked(0xf << 22, 0);  // Turn off all LEDs
    for (int i = 0; i < 4; i++) {
        gpio_put_masked(0xf << 22, 1 << (i + 22));  // Turn on LED i
        sleep_ms(500);  // Wait for 500 ms
        gpio_put_masked(0xf << 22, 0);  // Turn off all LEDs
        sleep_ms(500);  // Wait for 500 ms
    }
}
```

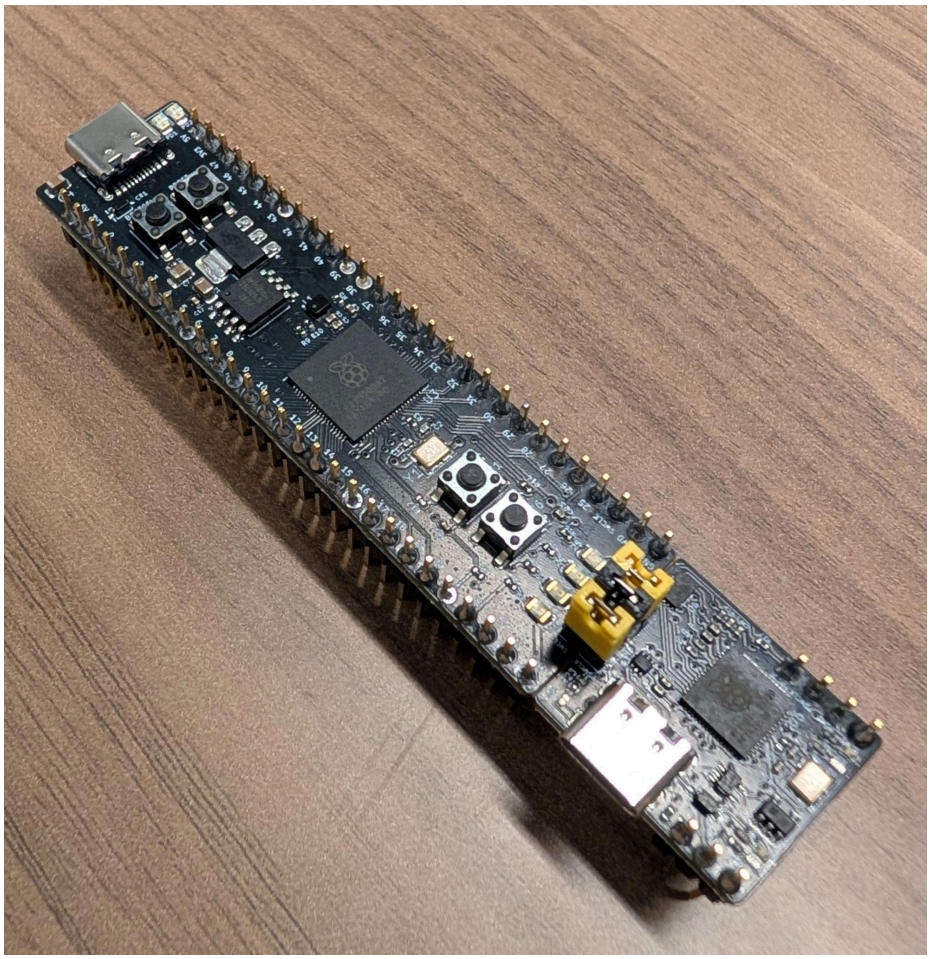
5. Connect your Proton board and debugger to your computer with data capable USB-C cables, and click PlatformIO icon on the left > proton > Upload. You should see all the LEDs turn on one by one, turn off, and repeat!

Soldering Guide

A soldering guide from start to finish is available here:
https://github.com/ece362-purdue/proton-labs/blob/main/lab0-intro/soldering.md

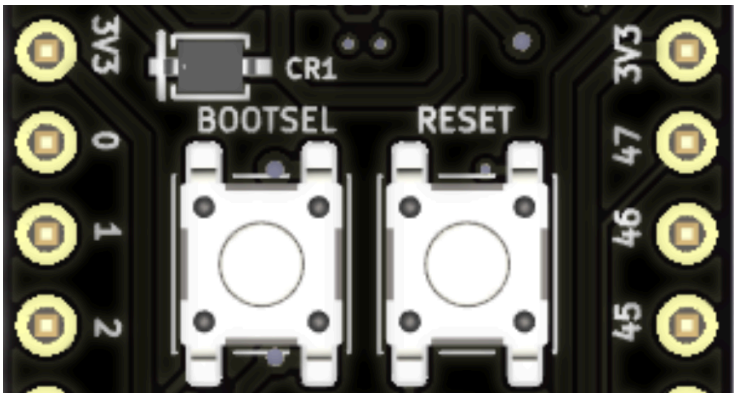
The guide will explain how to add the middle single-sided headers that connect your debugger to your development board, and the double-sided headers on the sides of the board.

The advantage of double-sided headers is to “expose” a third row of pins that you can use on your breadboard with Dupont connectors, since the board only leaves 2 rows once it is installed in the breadboard.



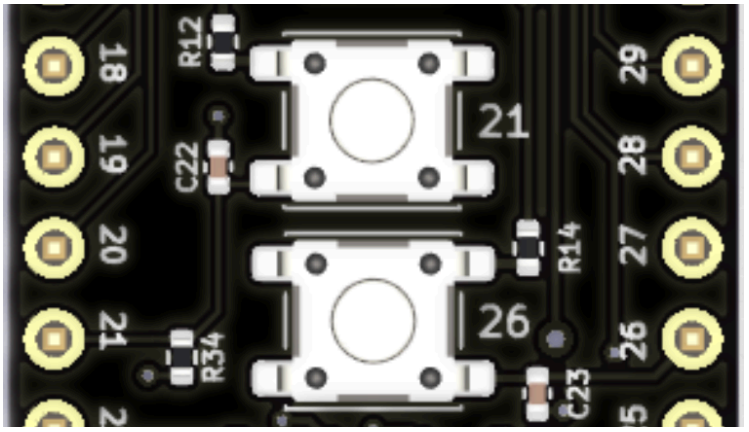
Push Buttons

The **BOOTSEL** button (connected to the eponymous pin on the RP2350B chip), when held while pressing RESET or connecting the USB-C cable, puts the device into USB boot mode, allowing a user to drag-n-drop a UF2 file into the mass storage that appears.



The **RESET** button is connected to the RUN input pin on the RP2350B chip, allowing a user to easily restart the program saved in flash.

The two user buttons connected to GP21 (upper) and GP26 (lower) can be configured as inputs. The GP21/26 have weak 10k ohm pull-down resistors to mitigate the RP2350-E9 bug which occurs when using GPIO pins as inputs, removing the need to add external pull-downs for the pushbuttons.



However, any other GPIO pin used as an input may need this fix.

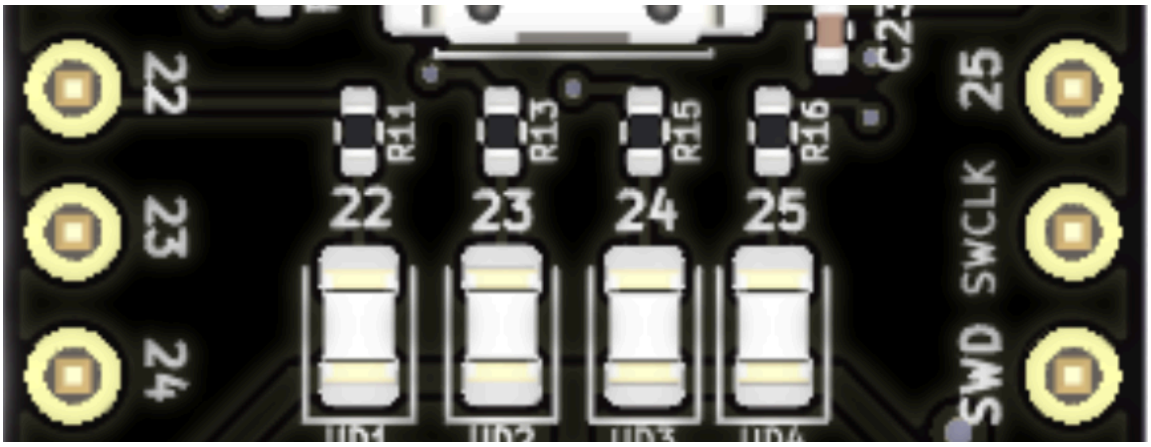
LEDs

PD1 is the 5V indicator LED that turns red. The 5V output comes from the USB connector, which is passed through a flyback diode

PD2 is the 3.3V indicator LED that turns green. The 5V is used by the low dropout (LDO) voltage regulator to produce the 3.3V output.

UD1-UD4 are the four user LEDs connected to GP22-GP25, in series with current limiting resistors (varying to equalize brightness across colors). GP22 is red, GP23 is yellow, GP24 is green, GP25 is blue.

To turn them on, the associated GPIO pin must be configured as an output and driven high.

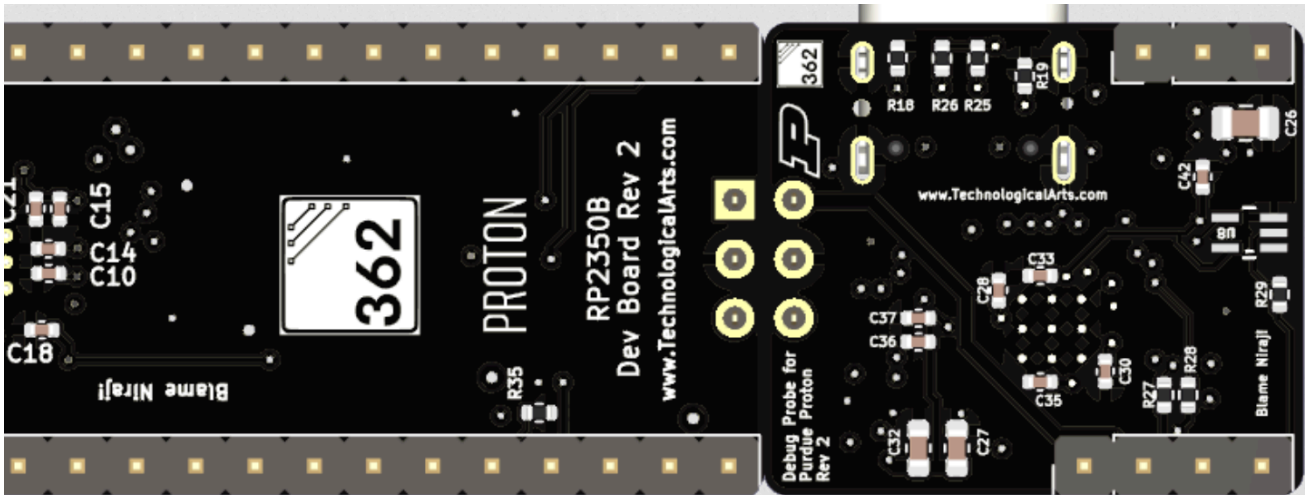


Separated Debugger/Development Board

The Proton board consists of two electrically independent designs:

- The RP2350B development board that functions as the breakout for the microcontroller, and;
- The Debug Probe that allows a user to debug the code running on the development board.

If needed, it is possible to "snap off" the debugger and move it to another part of your breadboard, or remove it entirely. The headers soldered on the 2x3 pads can be used with the official Raspberry Pi Debug Probe as well should you wish.



Debugger UART Interface

As advised by Raspberry Pi, GP0 and GP1 are used as UART TX and RX pins on the development board. To use them with the debugger, connect GP0 to RX on the debugger side, and GP1 to TX, so that they are **cross-connected**, allowing you to use the Serial Monitor function in PlatformIO to exchange serial data with your RP2350B over UART. The debugger will automatically present a serial port for this purpose on the computer connected via USB.

Design Decisions

The primary document that heavily informed our design choices for this board was the Hardware Design for RP2350 guide. Beyond that:

- The power indicator LEDs are helpful in determining if there are issues with the voltage regulator, and if the USB is properly delivering power in the first place.

- A reset button was added to make it easier for users to re-run their program without having to wire an external push button.

- Adding two push buttons and four user LEDs on the board itself provides sufficient external functionality for a user to evaluate the board before they have to put on pin headers and use it on a breadboard.

- The Debug Probe from Raspberry Pi was redesigned, with the help of their public schematics, and incorporated as part of the development board, with breakout pins so that it is supported on a breadboard.
    - Our debug probe retains the same pinout - SWCLK, GND, SWD - that would keep it compatible with the official RPi Pico 2 board.
    - The decision was made to make it part of the board for students as debugging is a critical requirement in any intro course to embedded systems. Incorporating it in a more visible way to students tends to make them (at least) aware of that aspect of embedded systems programming.

Current Issues

- Due to the RP2350-E9 erratum, GPIO pins used as inputs need a weak pull-down resistor to be added to ensure that the inputs can return to a logic 0 after being pulled high.
    - GP21 and GP26, connected to the onboard pushbuttons, already have 10k pull-down resistors added. This may limit its use for high-speed data transmissions, like SPI at >20 MHz.