Power switching RP2040 for low standby current applications

Raspberry Pi Ltd

2023-02-10: githash: c65fe9c-clean

Colophon

build-date: 2023-02-10
build-version: githash: c65fe9c-clean

Legal Disclaimer Notice

TECHNICAL AND RELIABILITY DATA FOR RASPBERRY PI PRODUCTS (INCLUDING DATASHEETS) AS MODIFIED FROM TIME TO TIME (“RESOURCES”) ARE PROVIDED BY RASPBERRY PI LTD (“RPL”) "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN NO EVENT SHALL RPL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE RESOURCES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

RPL reserves the right to make any enhancements, improvements, corrections or any other modifications to the RESOURCES or any products described in them at any time and without further notice.

The RESOURCES are intended for skilled users with suitable levels of design knowledge. Users are solely responsible for their selection and use of the RESOURCES and any application of the products described in them. User agrees to indemnify and hold RPL harmless against all liabilities, costs, damages or other losses arising out of their use of the RESOURCES.

RPL grants users permission to use the RESOURCES solely in conjunction with the Raspberry Pi products. All other use of the RESOURCES is prohibited. No licence is granted to any other RPL or other third party intellectual property right.

HIGH RISK ACTIVITIES. Raspberry Pi products are not designed, manufactured or intended for use in hazardous environments requiring fail safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, weapons systems or safety-critical applications (including life support systems and other medical devices), in which the failure of the products could lead directly to death, personal injury or severe physical or environmental damage (“High Risk Activities”). RPL specifically disclaims any express or implied warranty of fitness for High Risk Activities and accepts no liability for use or inclusions of Raspberry Pi products in High Risk Activities.

Raspberry Pi products are provided subject to RPL’s Standard Terms. RPL’s provision of the RESOURCES does not expand or otherwise modify RPL’s Standard Terms including but not limited to the disclaimers and warranties expressed in them.

Document version history

Release	Date	Description
1.0	3 Nov 2022	• Initial release

Scope of document

This document applies to the following Raspberry Pi products:

RP2040

Introduction

Even in deep sleep RP2040 draws a typical current of ~180µA, and sleep current is very dependent on PVT: process (current varies from chip to chip), voltage (current varies linearly with voltage), and temperature (current varies nonlinearly with temperature).

For many use cases where minimal current draw is required, the best option is to power off the system (or the RP2040 part of the system) completely if possible. This application note gives a couple of options for how this can be done, and these circuits are simple enough that a designer can adjust them for their own use case.

NOTE

The circuits in this document are provided as-is. The user should ensure they are suitable for their own use case and make sure they build and test them to confirm they work as expected in the user's own application.

Background

GPIO behaviour

RP2040 digital GPIOs are 3V3 failsafe, which means that when the chip is powered off, the pads are high impedance and can survive up to 3.63V (3.3V+10%) being applied to them indefinitely.

NOTE

When the chip is powered off there are no pulls on any pins and digital GPIO pins are high impedance.

NOTE

Pins that share ADC function are NOT failsafe as they have diodes to the VDD/VSS rails, so these must be isolated when powered off, if any signal is trying to drive into them when in this state.

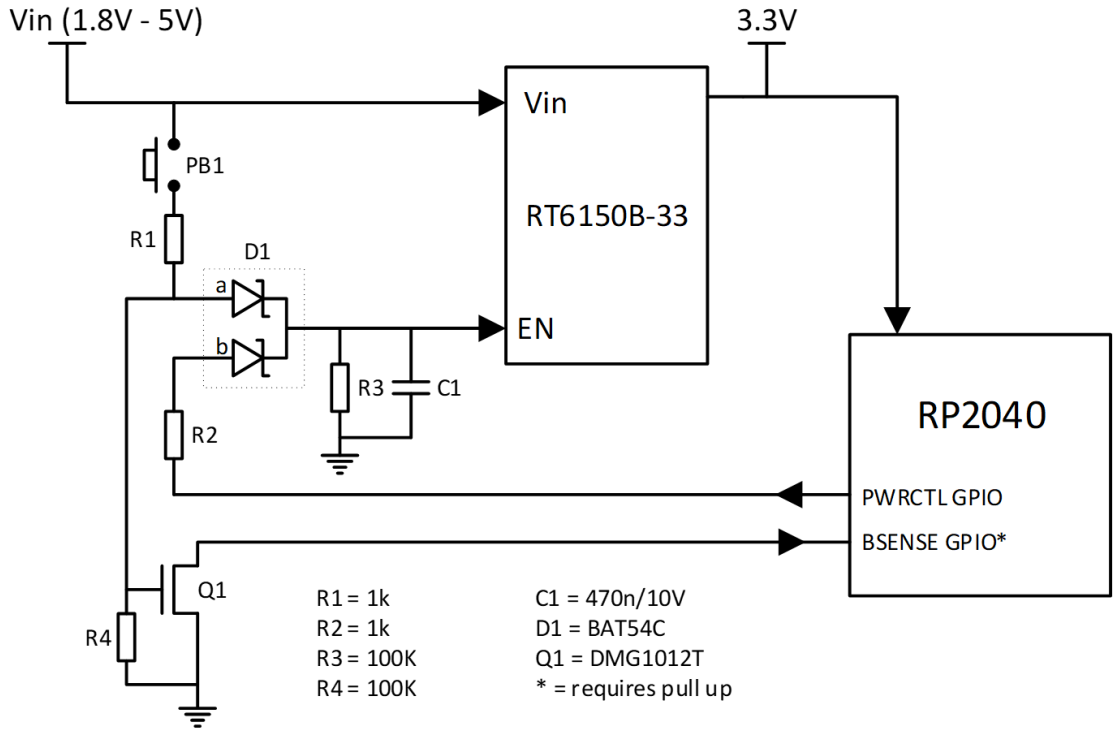
Basic principle

The basic principle of the following circuits is to have an "enable" signal that is pulled high by a push button, and debounced/held stable by a capacitor for long enough that RP2040 can boot. RP2040, once booted, can drive a GPIO to "latch" this enable signal.

The following circuits also have the option of RP2040 reading the button (via a second GPIO) and therefore offer the option of using this button for user input, or as a dual power on/power off button (for example, hold the button for three seconds to power off the system). Note that if button sensing is not required in a user's application, the extra circuitry for this can be omitted.

Example 1: power conversion

This example circuit is suitable when the input voltage can be lower or higher than the RP2040 IO voltage and some kind of converter is used — for example, an SMPS or LDO with suitable enable pin.



In this example we use the Richtek RT6150B-33 buck-boost SMPS as found on Raspberry Pi Pico, outputting a fixed 3.3V to RP2040.

Circuit operation

The above circuit will cope with a power input from 1.8V up to the maximum input voltage of the RT6150B (5.5V max). Operation is as follows:

- When PB1 is pressed, C1 is charged up to (almost) Vin quickly via R1 (while R3 discharges C1 slowly).
    - This should give time for RP2040 to boot and drive PWRCTL high to keep the voltage on C1 higher than the EN pin threshold and therefore latch the power on.
- The D1 diodes isolate the RP2040 IO and Vin so that neither drives the other if and when they are at a higher voltage.
    - D1a also allows R4 to pull the gate of Q1 low when PB1 is open and therefore sense the button press (because when pressed R1 pulls Q1 gate high).
- Q1 will pull BSENSE GPIO low when the button is pressed (note that the circuit relies on the internal BSENSE GPIO pull-up being activated).
- R2 will limit GPIO current into C1 if C1 is large and has discharged significantly (but still above EN threshold) when PWRCTL drives high. If C1 is small this resistor is probably not needed.
- R1 limits current through PB1 and D1a Schottky but also serves as ESD current limiting for Q1.
- We use a MOSFET for Q1 with a low Vt and integral ESD clamp on the gate (Diodes DMG1012T).

NOTE

If power button sensing is not needed, then Q1, R4, and D1a can be omitted.

Powering off

To power off the system, the RP2040 simply floats the PWRCTL GPIO (or drives it low), and after C1 discharge time and assuming PB1 is not pressed, the power will latch off. For example, with button sensing we can use PB1 as a power on button, as a user button, and as a power off button:

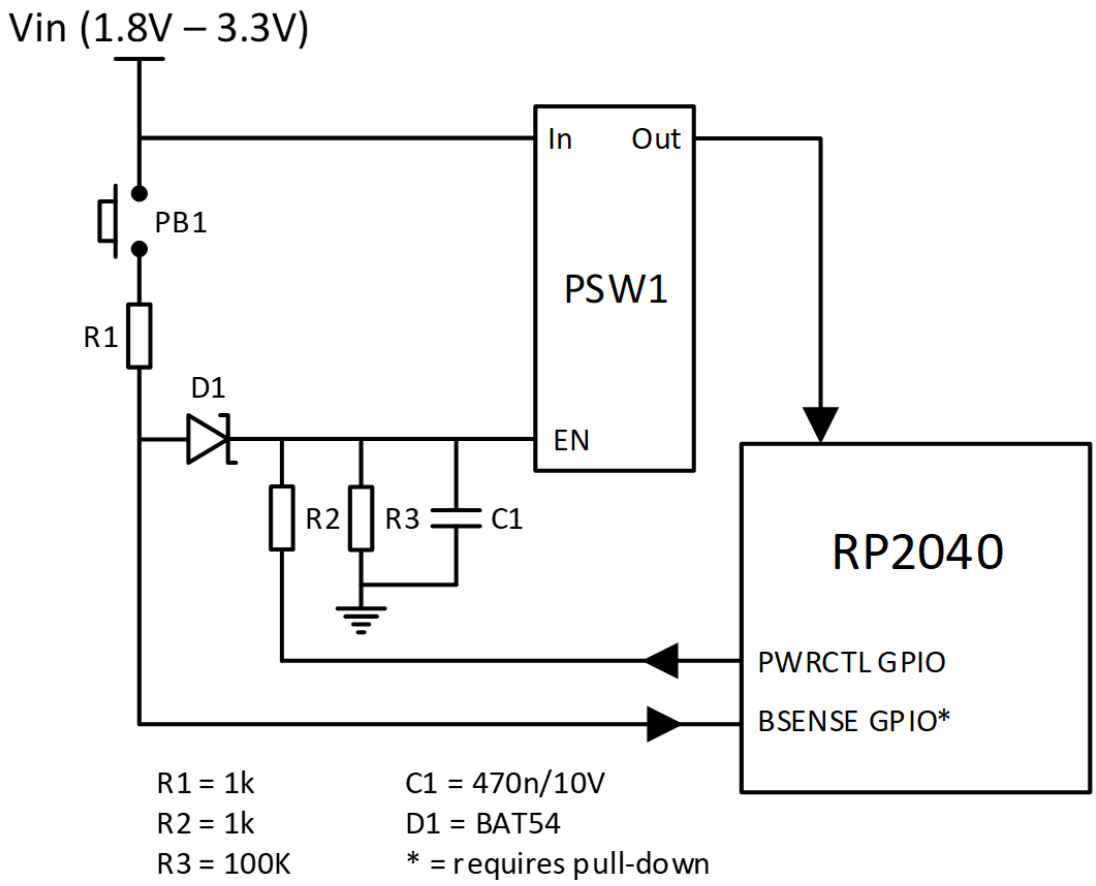
- User presses PB1 to power system on
- System can detect short user button presses
- System can detect (using suitable software) when button is pressed for e.g. >3 seconds, and can then perform system shutdown operations before finally driving PWRCTL low.

NOTE

If the user is holding the button to perform a power down, the power will only physically be removed when the user releases the button. This is likely to be acceptable as the user will still see the system appear to power down (LEDs off etc) and will then release the button.

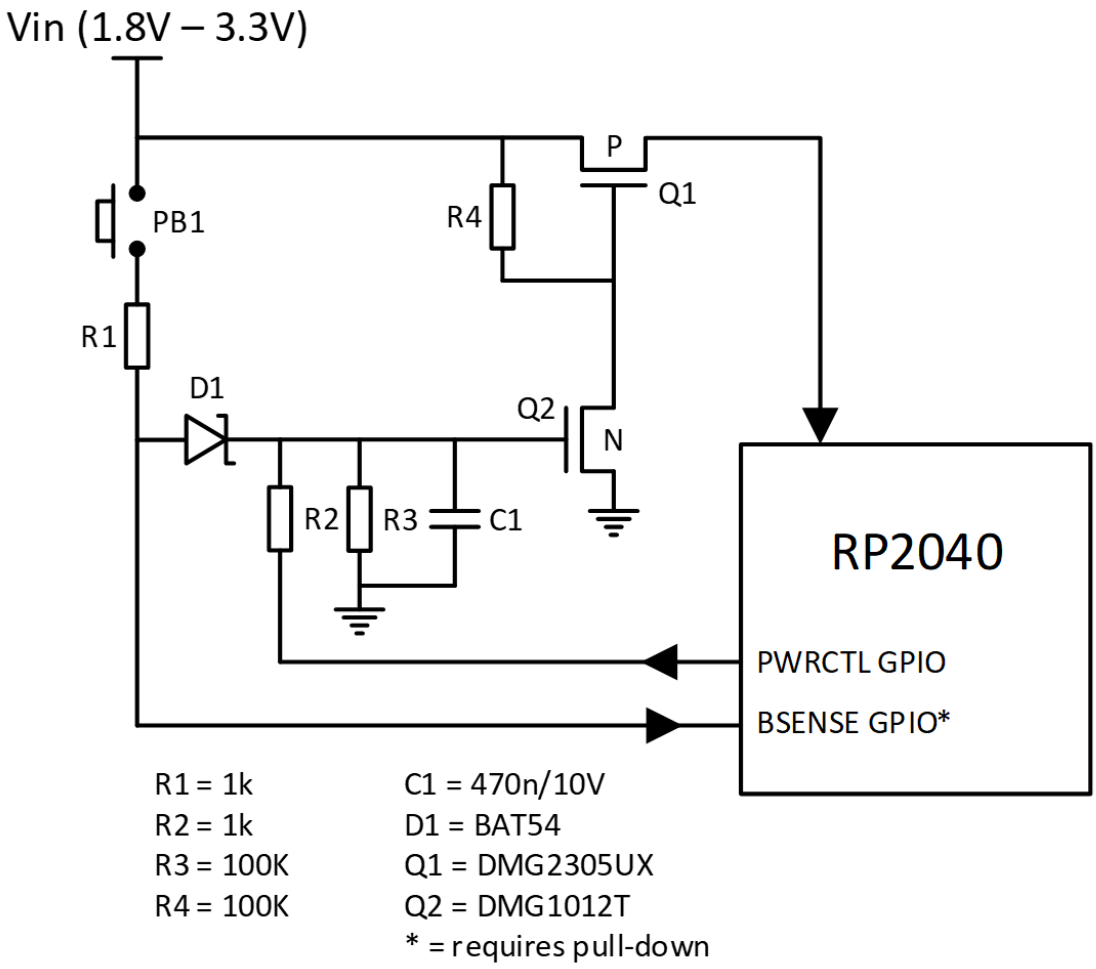
Example 2: power switching

The first example circuit can be adapted to drive a high-side power switch (with suitable enable input). For example, a user may have an always-on 3.3V rail available, or the system may be operating from batteries (2 × alkaline AA cells give ~2V-3.6V). In this case we know the voltage used is within safe limits to drive the RP2040 directly. As we know the RP2040 IO voltage and input voltage are the same, we can simplify the original circuit:



Circuit operation is largely as before, though the BSENSE GPIO now needs a pull-down rather than a pull-up and the GPIO sense is inverted. Note that the PWRCTL GPIO should be floated to turn off rather than pulled low. PSW1 needs to have a suitable low-threshold EN pin; for example, something like Ti TPS22917 should work with Vin down to 1.8V.

PSW could also be a simple P-channel MOSFET with an N-channel MOSFET pulling its gate:



The P and N MOSFETs must have suitably low threshold voltages (the example devices chosen have Vt of 1 volt maximum).

Note also that using a PFET like this has no current limiting. MOSFET on resistance, downstream load current, and especially inrush current and load capacitance need to be taken into account.

Conclusion

The circuits described here give some examples and can be adapted to a user's needs, based on the basic working principles and circuit design ideas presented.

For example, rather than a push button, wakeup could be initiated by an RTC chip once every minute, for low-power sensing applications. Or perhaps more than one ON signal is required (in which case diode ORing them onto the EN pin node may be a good strategy).

Raspberry Pi