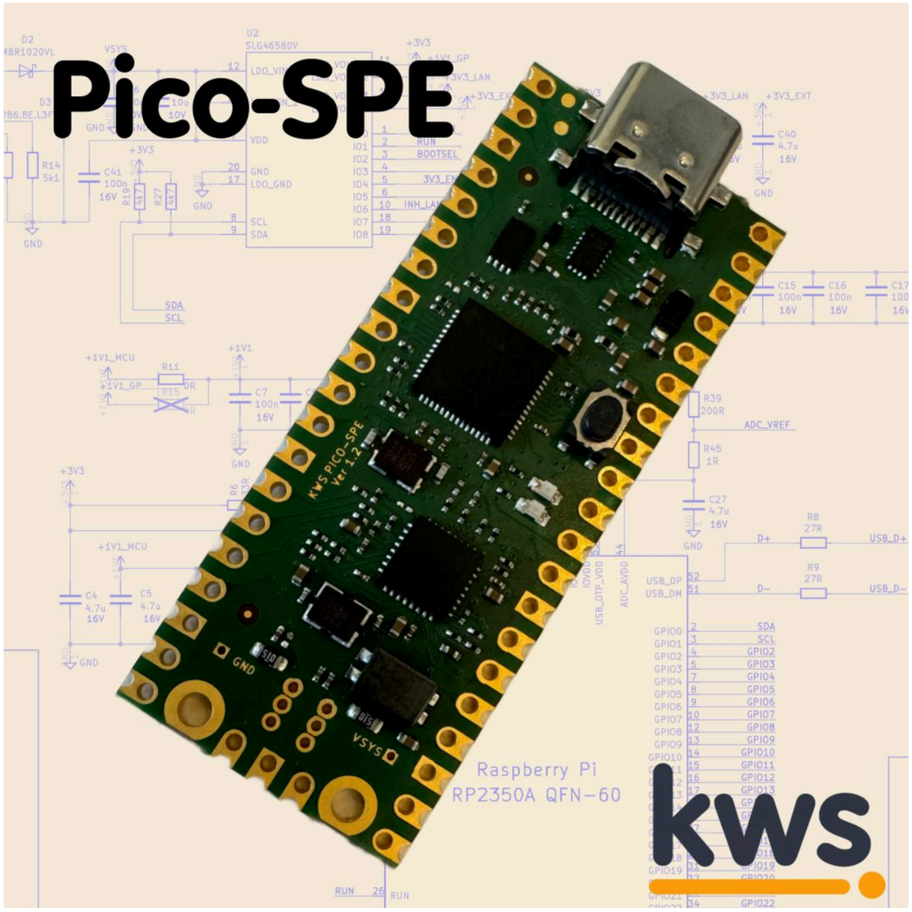
Pico-SPE Datasheet

An RP2040-based microcontroller board



Chapter 1. About Pico-SPE

Pico-SPE is a microcontroller board based on the Raspberry Pi RP2040 micocontroller chip

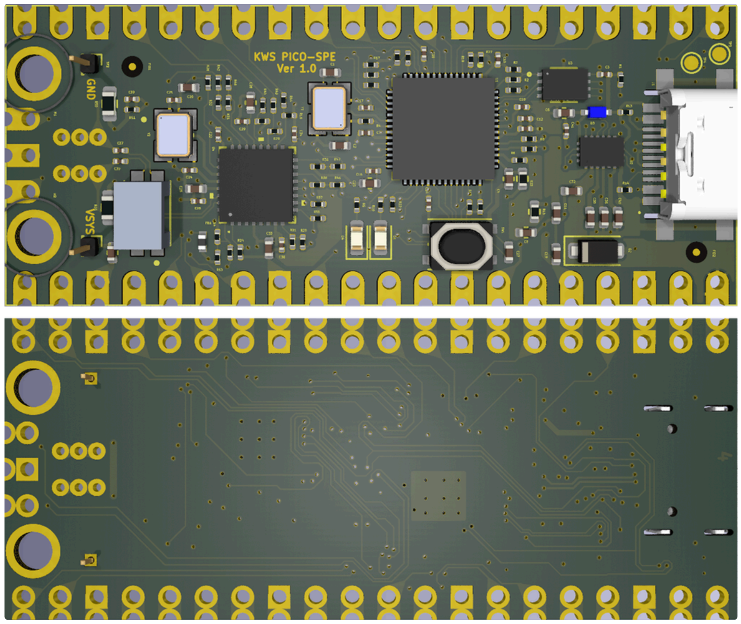


Figure 1. the Pico-SPE Rev1 board

Pico-SPE has been designed to be a low cost yet flexibale development platform for RP2040, with the following key features:

- RP2040 Micocontroller with 2MB Flash
- USB-C port for power and data (and for reprogramming the flash)
- 40 pin 21x51 ‘DIP’ style 1mm thick PCB with 0.1” through-hole pins also with edge castellations
  - Expose 19 multi-function 3.3V General Purpose I/O (GPIO), 3 can be used for ADC
  - Can be surface mounted as a module
- 3-pin ARM Serial Wire Debug (SWD) port
- Integrated Microchip LAN8651 10Base-T1S controller
  - Numerous test pads are available for LAN8651 for evaluation purposes
- Simple yet highly flexible power supply architecture
  - Various options for easily powering the unit from USB-C, external supplies or batteries
- High quality, low cost, high availability
- software examples and documentations

For full details for the RP2040 microcontroller please see the RP2040 Datasheet, however the headline features are:

- Dual-core M0+ at up to 133MHz
  - On-chip PLL allows variable core frequency
- 264kB multi-bank high performance SRAM
- External QUAD-SPI Flash with eXecute in Place (XIP) and 16kB on-chip cache
- High performance full-crossbar bus fabric
- On-board USB1.1 (device or host)
- 30 multi-function General Purpose IO (4 can be used for ADC)

    - 1.8-3.3V IO Voltage (NOTE Pico IO voltage is fixed at 3.3V)
- 12-bit 500ksps Analogue to Digital Converter (ADC)
- Various digital peripherals
    - 2 × UART, 2 × I2C, 2 × SPI, 16 × PWM channels
    - 1 × Timer with 4 alarms, 1 × Real Time Counter
- 2 × Programmable IO (PIO) blocks, 8 state machines total
    - Flexible, user-programmable high-speed IO
    - Can emulate interfaces such as SD Card and VGA

Pico provides minimal (yet flexible) external circuitry to support the RP2040 chip: flash (Winbond W25Q16JV), crystal (TXC 7V-12.000MDDE-T), power supplies and decoupling, and USB connector. The majority of the RP2040 microcontroller pins are brought to the user IO pins on the left and right edge of the board. Four RP2040 IO are used for internal functions - driving an LED, on-board Switched Mode Power Supply (SMPS) power control and sensing the system voltages.

Pico has been designed to use either soldered 0.1" pin-headers (it is one 0.1" pitch wider than a standard 40-pin DIP package) or can be used as a surface mountable 'module', as the user IO pins are also castellated. There are SMT pads underneath the USB connector and BOOTSEL button, which allow these signals to be accessed if used as a reflow soldered SMT module.

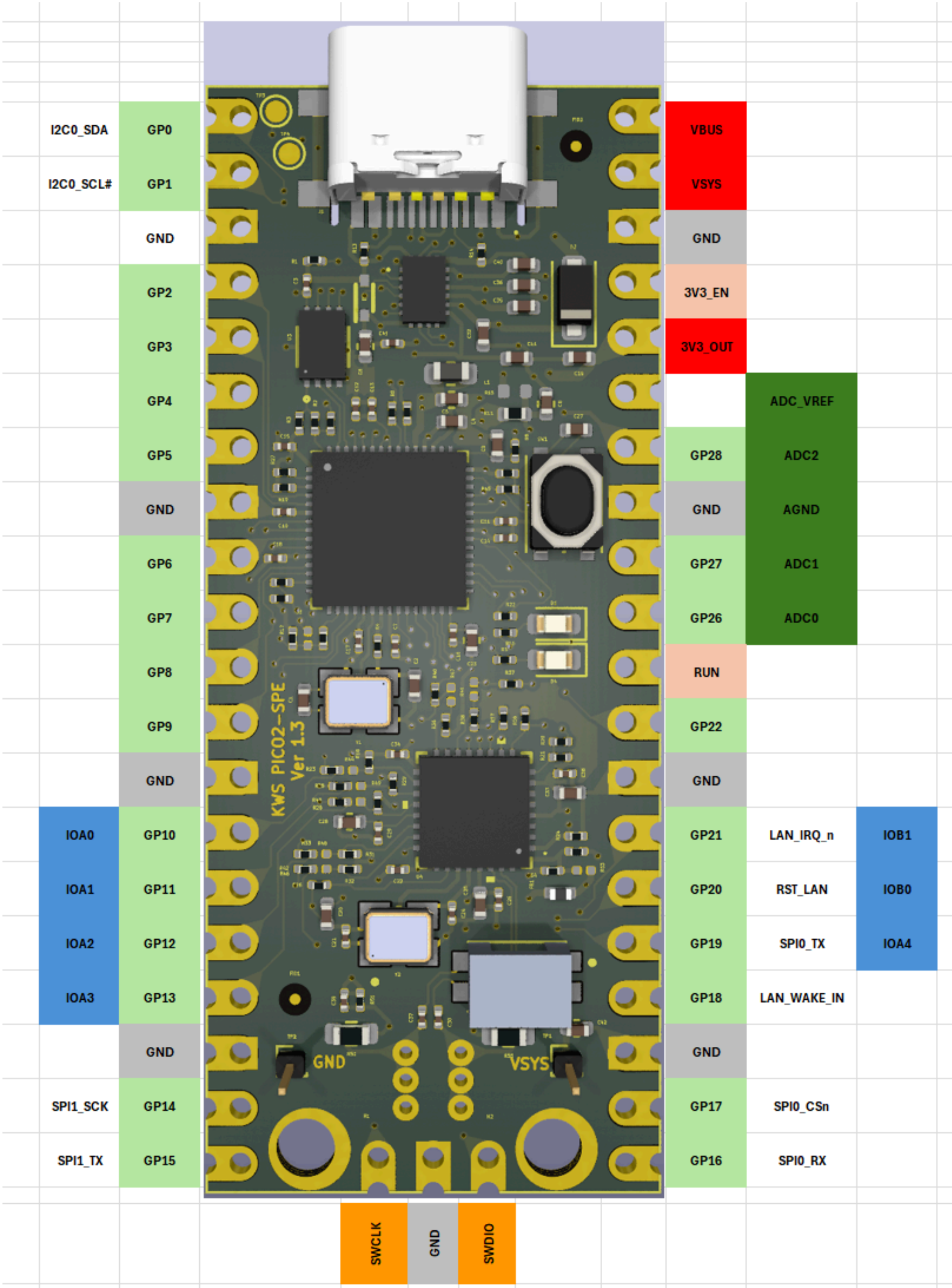


Figure 2. the pinout of the Pico-SPE Rev1 board

Pico-SPE uses an SLG46580V to generate the required 3.3V (to power RP2040 and external circuitry) from a wide range of input voltages (~1.8 to 5.5V). this allows significant flexibility in powering the unit from various sources such as a single Lithium-Ion cell, or 3 AA cells in series.

Reprogramming the Pico Flash can be done using USB (simply drag and drop a file onto the Pico which appears as a mass storage device), or the standard Serial Wire Debug (SWD) port can reset the system and load and run code without any button presses. The SWD port can also be used to interactively debug code running on the RP2040.

Getting started with Pico-SPE

The Getting started with Pico-Spe walks through loading programs onto the board, and shows how to install zephyr and build the example.

Chapter 2. Mechanical specification

The Pico-SPE is a single sided 51x21mm 1mm thick PCB with an USB-C port overhanging the top edge. The Pico-SPE is designed to be usable as a surface mount module as well as being in Dual Inline Package (DIP) type format, with the 40 main user pins on a 2.54mm (0.1") pitch grid with 1mm holes and hence compatible with veroboard and breadboard, see Figure 3.

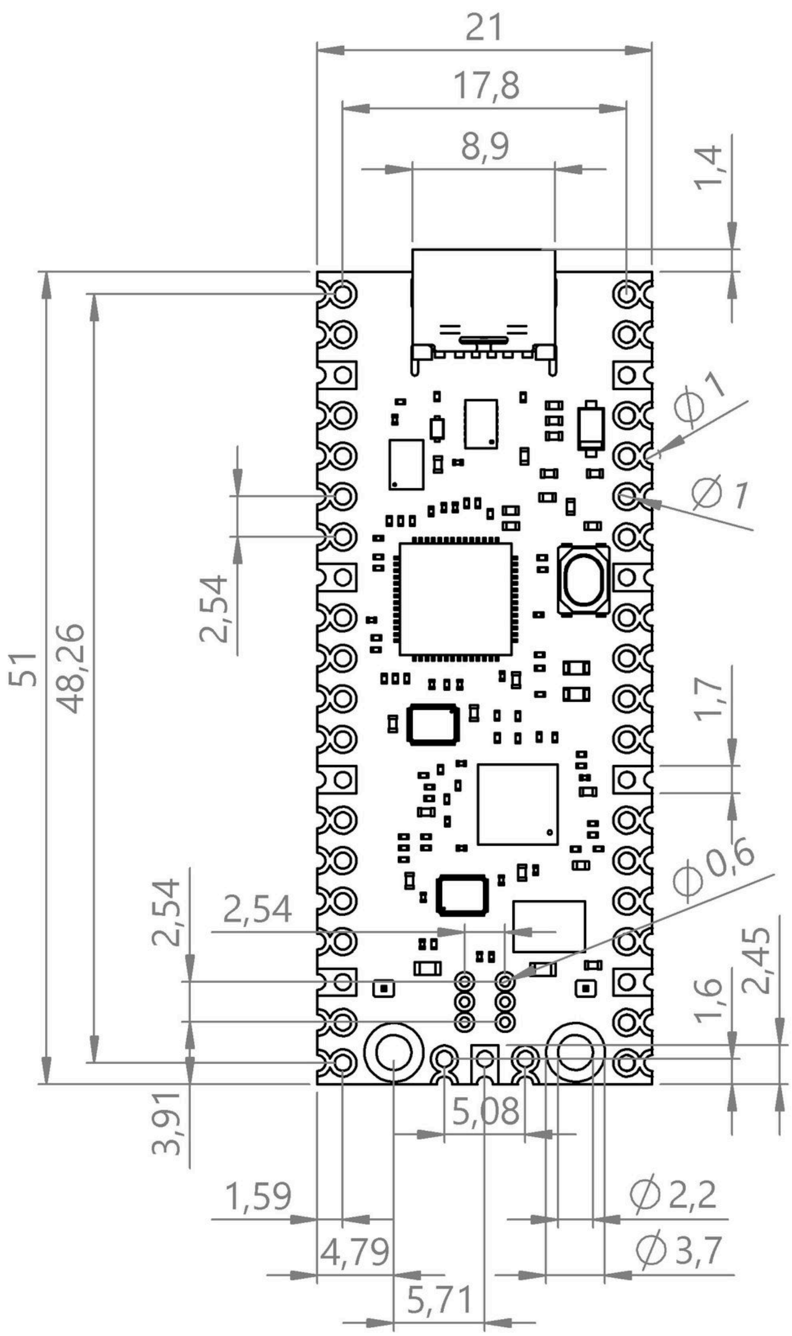


Figure 3. The dimensions of the Pico-SPE Board Rev1

2.1 Pico-SPE pinout

The Pico-SPE has been designed to directly bring out as much of the RP2040 GPIO and internal circuitry functions as possible, while also providing a suitable number of ground pins to reduce EMI (Electro Magnetic Interference) and signal crosstlak. This is important in general but especially for RP2040 which is built on a modern 40 nm silicon process and hence the digital IO edge rates are veryf ast.

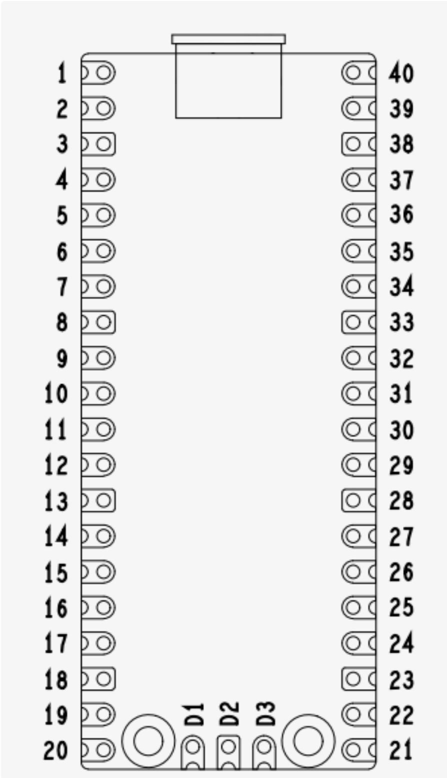


Figure 4. The pin numbering of the Pico-SPE Rev1 Board

The physical pin numbering is shown in Figure 4, for the pin allocation see Figure 2

A few RP2040 GPIO pins are used for internal board functions, these are:

GPIO25 OP Connected to user LED

Apart from GPIO and grounds pins, there are 7 other pins on the main 40-pin interface:

PIN40 VBUS

PIN39 VSYS

PIN37 3V3_EN

PIN36 3V3

PIN35 ADC_VREF

PIN33 AGND

PIN30 RUN

VBUS is the USB-C input voltage, connected to USB-C port pin B4A9 and A4B9. This nominally 5V (or 0V if the USB is not connected or not powered)

VSYS is the main system input voltage, which can vary in the allowed range 1.8V to 5.5V, and is used by the on-board SLG46580V to generate the 3.3V for the RP2040 and its GPIO.

3V3_EN connects to the SLG46580V pin.

3V3 is the main 3.3V supply to RP2040 and its I7O, generated by the SLG46580V. This pin can be used to power external circuitry (maximum output current will depend on RP2040 load and VSYS voltge, it is recommended to keep the load on this pin less than 300mA).

ADC_VREF is the ADC power supply (and reference) voltage, and is generated on Pico-SPE by filtering the 3.3V supply. This pin can be used with an external refernce if better ADC performance is required.

AGND is connected to GND.

RUN is the RP2040 enable pin, and has an internal (on-chip) pull-up resistor to 3.3V of about ~50kΩ. To reset RP2040 short this pin low.

2.2 Surface-mount foodprint

The following footprint is recommended for systems which will be reflow-soldering Pico-SPE units as modules.

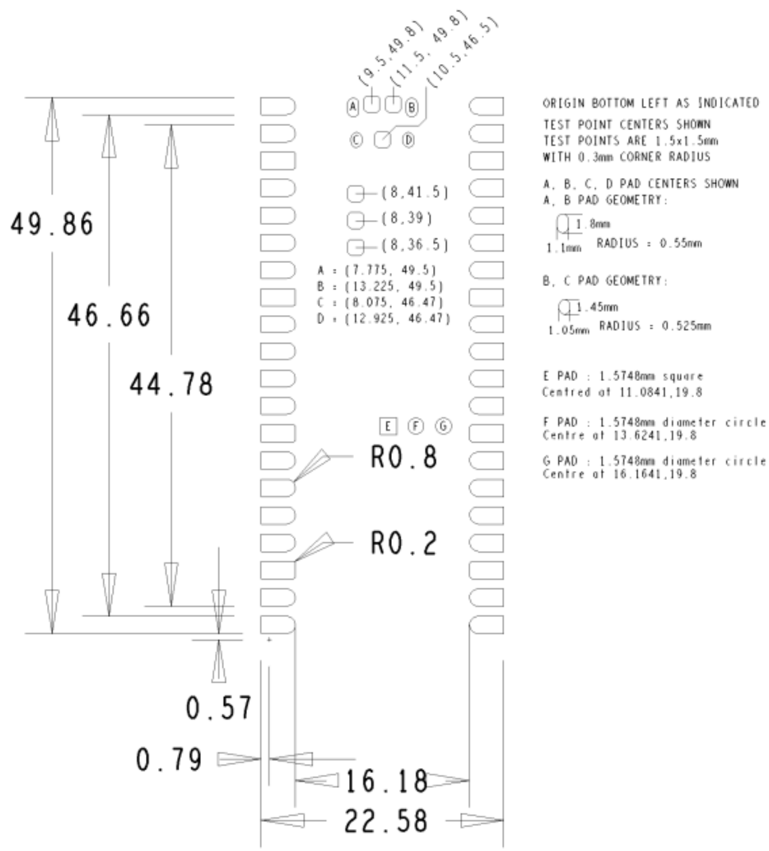


Figure 5. The SMT footprint of the Pico-SPE Rev1 board [THIS IS AN EXAMPEL WHAT WE NEED]

The footprint shows the pad size as well as the 4 USB connector shell ground pads (A,B,C,D). The USB connector on the Pico-SPE is a through-hole part, which provides it with mechanical strength. The USB socket pins do not protrude all the way through the board, however solder does pool at these pads during manufacture and can stop the module sitting completely flat. Hence we provide pads on the SMT module footprint to allow this solder to reflow in a controlled manner when Pico-SPE goes through reflow again.

The paste stencil must be bigger than the footprint. Over-pasting the pads ensures the best possible resutls when slodering. The following paste stencil (Figure 6) indicates the dimensions of solder paste zones on the Pico-SPE. We recommend paste zones 163% larger than the footprint.

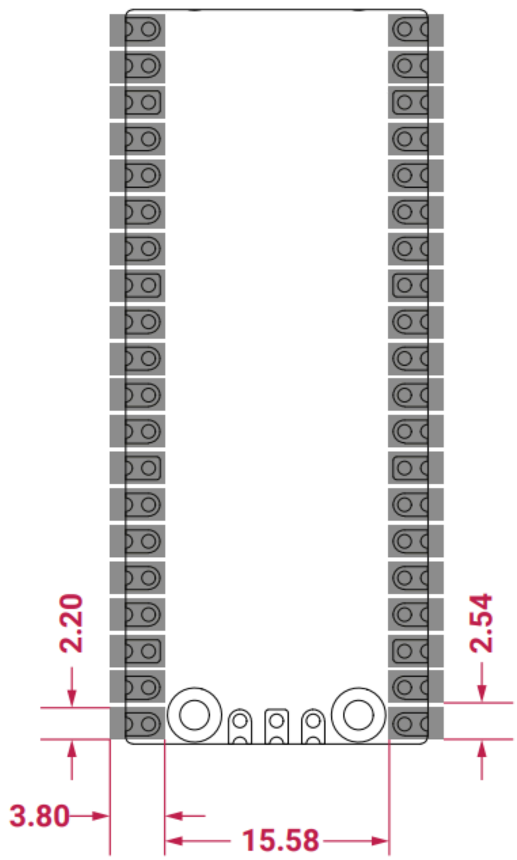


Figure 6. The paste stencil of the Pico-SPE Rev1 board

2.3 Recommended operating conditions

Operating conditions for the Pico-SPE are largely a function of the operating conditions specified by it components.

Operating Temp Max	85°C (including self-heating)
Operating Temo Min	-20 °C
VBUS	5V ± 10%
VSYS Min	1.8V
VSYS Max	5.5V

Note that VBUS and VSYS current will depend on use-case, some examples are given in the next section.

Recommended maximum ambient temperature of operation is 70°C.

Chapter 3. Electrical specification

3.1 Power consumption

The power consumption tables and graph show the VBUS current consumption on three typical Pico-SPE devices, with three different software use cases. These results are not guaranteed maximum values, they are an indication of the current consumption a user can typically expect the device to draw when used in these scenarios.

For more detailed current consumption data, please see the RP2040 Datasheet.

3.1.1 Ampel

The ‘ampel’ demo uses the spe and gpios. This sample uses the spe chip to communicate and three gpios to trigger leds.

Pico-SPE Board	**Avarage VBUS current @5V (mA)**			**Maximum VBUS current @5V (ma)**		
	Temperature (°C)			**Temperature (°C)**		
	-25	25	85	-25	25	85
#1						
#2						
#3						
Mean						

3.1.2 BOOTSEL mode

The next Table describes the BOOTSEL mode of RP2040. These measurements are made both with and without USB activity on the bus, using a Raspberry Pi 4 as a host.

Pico-SPE Board	**USB idle VBUS current @5V (mA)**			**USB active VBUS current @5V (ma)**		
	Temperature (°C)			**Temperature (°C)**		
	-25	25	85	-25	25	85
#1						
#2						
#3						
Mean						

3.1.3 SLEEP mode

The final example uses the hello_sleep binary code, which puts RP2040 into SLEEP mode, a low power state which leaves some clock infrastructure in the chip turned on.

Pico-SPE	**VBUS current @5V (mA)**

Board	Temperature (°C)		
	-25	25	85
#1			
#2			
#3			
Mean			

Chapter 4. Applications information

4.1 Programming the flash

The on-board 16Mb QSPI Flash can be (re)programmed either using the Serial Wire Debug port or by the special USB Mass Storage Device mode.

The simplest way to reprogram the Pico-SPE's Flash is to use the USB mode. To do this, power-up the board, press the BOOTSEL button for around 2s. the Pico-SPE will then appear as a USB Mass Storage Device. Dragging an special '.uf2' file onto the disk will write this file to the Flash and restart the Pico-SPE.

The USB boot code is stored in ROM on RP2040, so can not be accidentally overwritten.

To get started using the SWP port see the Getting started Pico-Spe guide.

4.2 General purpose I/O

The Pico-SPE GPIO is powered from the on-board 3.3V rail and is therefore fixed at 3.3 V.

The Pico-SPE exposes 19 of the 30 possible RP2040 GPIO pins by routing them straight out to Pico-SPE header pins. GPIO0 to GPIO22 are digital only and GPIO 26-28 are able to be used either as digital GPIO or as ADC inputs (software selectable).

One thing to note is that the ADC capable GPIO26-28 have an internal reverse diode to the VDDIO (3V3) rail and so the input voltage must not exceed VDDIO plus about 300mV. Also, if the RP2040 is unpowered, applying a voltage to these GPIO pins will 'leak' through the diode into the VDDIO rail. Normal digital GPIO pins 0-22 (and also the debug pins) do not have this restriction and therefore voltage can sfely be applied to these pins when RP2040 is unpowered.

4.3 Using the ADC

the RP2040 ADC does not have an on-board reference and therefore uses its own power supply as a reference. On Pico-SPE the ADC_AVDD pin (the ADC supply) is generated from the SLG46580V 3V3. This is a simple solution but does have the following drawbacks:

1. We are relying on the 3.3V SLG46580V output accuracy which isn't great
2. We can only do so much filtering and therefore ADC_AVDD will be somewhat noisy
3. The ADC draws current (about 150μA if the temperature sense diode is disabled, but it varies from chip to chip) and therefore there will be an inherent offset of about 150μA*200 = ~30mV. there is a small difference in current draw when the ADC is sampling (about +20μA) so that offset will also vary with sampling as well as operating temperature

Chaning the resitance between the ADC_VREF and 3V3 pin can reduce the offset at the expense of more noise - which may be OK especially if the use can support averaging over multiple samples.

The ADC offset can be reduced by tying a second channel of the ADC to ground and using this zero-measurement as an approximation to the offset.

For much improved ADC performance, an external 3.0V shunt reference, such as LM4040, can be connected from the ADC_VREF pin to ground. Note that if doing this the ADC range is limited to 0-3.0V signals (rather than 0-3.3V), and the shunt reference will draw continuous current through the 200Ω filter resistor (3.3V-3.0V)/200 = ~1.5mA

Note that the 1Ω resistor on Pico-SPE (R45) is designed to (maybe) help with shunt references that would otherwise become unstable when directly connected to 2.2μF. It also makes sure there is a little filtering even in the case that 3.3V and ADC_VREF are shorted together (which is valid thing to do if you dan't care about noise and want to reduce the inherent offset).

Finally, R39 is a physically large 603(0402) package resistor, so can be relatively removed if a user wants to isolate ADC_VREF and do their own thing with the ADC voltage, for example powering it from an entirely separate voltage (e.g. 2.5V). Note that the ADC on RP2040 has only been qualified at 3.0/3.3V but should work down to about 2V.

4.4 Powerchain

Pico-SPE has been designed with a simple yet flexible power supply architecture and can easily be powered from other sources such as batteries or external supplies. Integrating the Pico-SPE with external charging circuits is also straightforward.

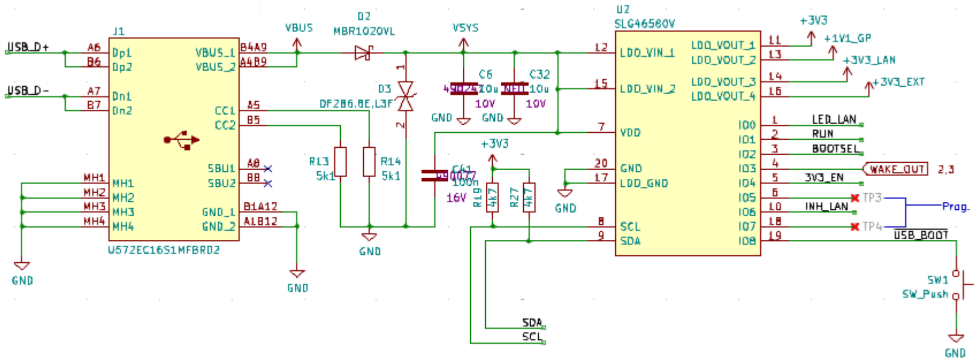


Figure 15. The powerchain of the Pico-SPE Rev1 board.

VBUS is the 5V input from the UCB-C port, which is fed through a Schottky diode to generate VSYS. The VBUS to VSYS diode (D2) adds flexibility by allowing power ORing of different supplies into VSYS.

VSYS is the main system ‘input voltage’ and feeds the SLG46560V, which generates a fixed 3.3V output for the RP2040 device and its IO (and can be used to power external circuitry).

The SLG46560V also generate from the VSYS the 3V3_Lan for the LAN8651B1T and the 3V3_EXT for the Pico-SPE Header.

4.5 Powering Pico-SPE

The simplest way to power Pico-SPE is to plug the USB-C, which will power VSYS (and therfore the ystem) from the 5V USB VBUS voltage, via D2 (so VSYS becomves VBUS minus the Schottky diode drop).

If the USB port is the **only** power source, VSYS and VBUS can be safely shorted together to eliminate the Schottky diode drop (which improves efficiency and reduces ripple on VSYS).

If the USB port is **not** going to be used, it is safe to power Pico-SPE by connecting VSYS to your preferred power source (in the range ~1.8V to 5.5V).

> If your are using Pico-SPE in USB Host mode (e.g. using one of the TinyUSB host examples) then you must power Pico-SPE by providing 5V to the VBUS pin.

The simplest way to safely add a second power source to Pico-SPE is to feed it into VSYS via another Schottky diode.his will ‘0R’ the two voltages, allowing the higher of either the external voltage or VBUS to power VSYS, with the diodes preventing either supply from back-powering the other. For example a single Lithium-Ion cell *(cell voltage ~3.0V to 4.2V) will work well, as will 3xAA series cells (~3.0V to ~4.8V) and any other fixed supply in the range ~2.3V to 5.5V. The downside of this approach is that the second power supply will suffer a diode drop in the same way as VBUS does, and this may not be desirable from an efficiency perpective or if the source is already close to the lower range of input voltage allowed for the SLG46560V.

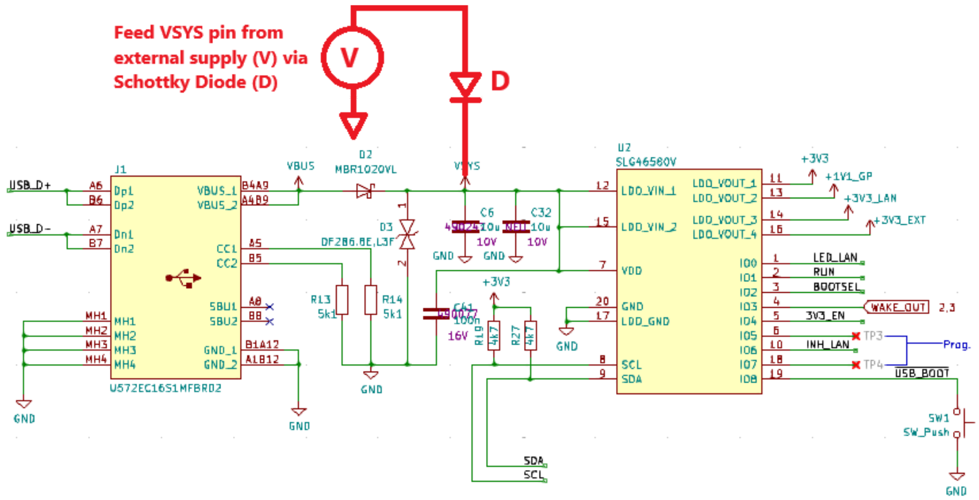


Figure 16 Pico-SPE power ORing using diodes.

An improved way to power from a secound source is using a P-channel MOSFET (P-FET) to replace the Schottky diode as a shown in Figure below. Here, the gate of the FET is controlled by VBUS and will disconnect the secondary source when VBUS is present. The P-FET should be chosen to have low on resistance and therefore overcomes the efficiency and voltage-drop issues with the diode-only solution.

Note that the Vt (threshold voltage) of the P-FET must be chosen to be well below the minimum external input voltage, to make sure the P-FET is turned on swiftly and with low resistance. When the input VBUS is removed, the P-FET will not start to turn on until VBUS drops below the P-FET's Vt meanwhile the body diode of the Pf-FET may start to conduct (depending on whether Vr is smaller than the diode drop). For inputs that have a low minimum input voltage or if the P-FET gate is expected to change slowly (e.g. if any capacitance is added to VBUS) a secondary Schottky diode across the P-FET (in the same direction as the body diode) is recommended. This will reduce the voltage drop across the P-FET's body diode.

An example of a suitable P-MOSFET for most situations is DIODES DMG2305UX which has a maximum Vt of 0.9V and Ron of 100mΩ (at 2.5V Vgs).

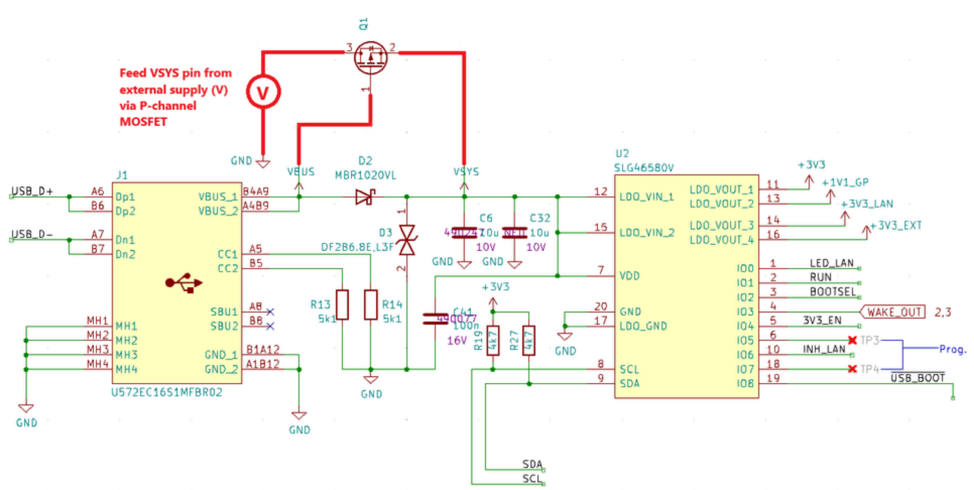


Figure 17. Pico-SPE power ORing using P channel MOSFET

⚠ If using Lithium-Ion cells they must have or be provided with adequate protection aigaint over-discharge, over charge, charging outside allowed temperature range and overcurrent. Bare, unprotected cells are dangerous and can catch fire or explode if over-discharged, over-charged or charged / discharged outside their allowed temperature and/or current range

4.6 Using a battery charger

Pico-SPE can also be used with a battery charger. Although this is a slightly more complex use case it is still straightforward. The figure below shows an example of using a ‘Power Path’ type charger (where the charger seamlessly manages swapping between powering from battery or powering from the input source and charging the battery as needed).

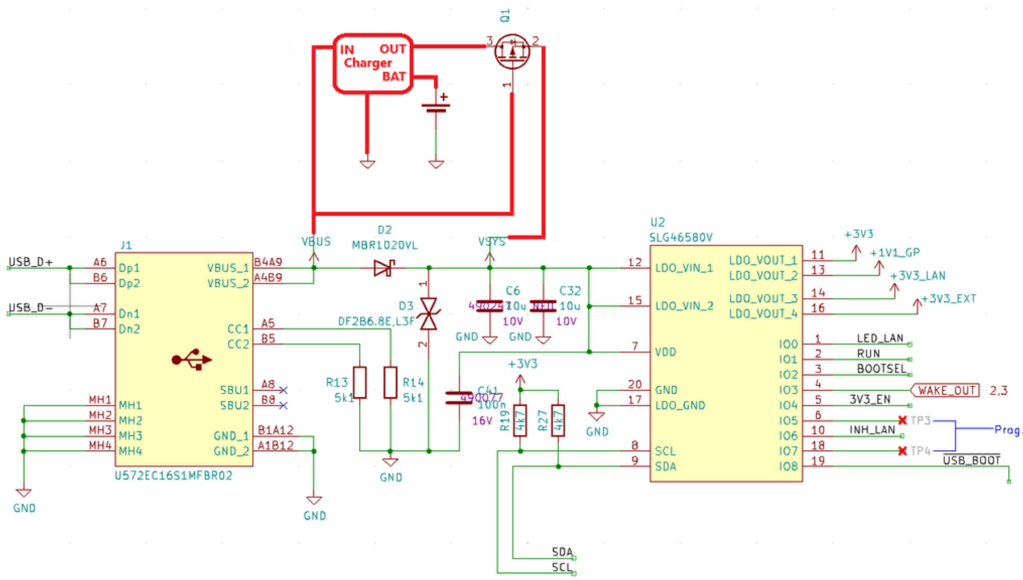


Figure 18. Using Pico-SPE with a charger

in the example we feed VBUS to the input of the charger and we feed VSYS with the output via the previously mentioned P-FET arrangement. Depending on your use case you may also want to add a Schottky diode across the P-FET as described in the previous section.

4.7 USB

RP2040 has an integrated USB1.1 PHY and controller which can be used in both Device and host mode. Pico-SPE adds the two required 27Ω external resistors and brings this interface to a standart USB-C port.

The USB port can be used to access the USB bootloader (BOOTSEL mode) stored in the RP2040 boot ROM. It can also be used by user code, to access an external USB device or host.

4.8 SPE interface

The Pico-SPE contains an on-board Single Pair Ethernet (SPE) interface using the Microchip LAN8651, which has the following feature:

- 10Base-T1S Phy designed to IEE Std. 802.3cg -2019
- Integrated Media Access Controller (MAC)

The SPE connection is on Board with 6 pins shown in Figure 19. The spe interface is connected via SPI to the RP2040.

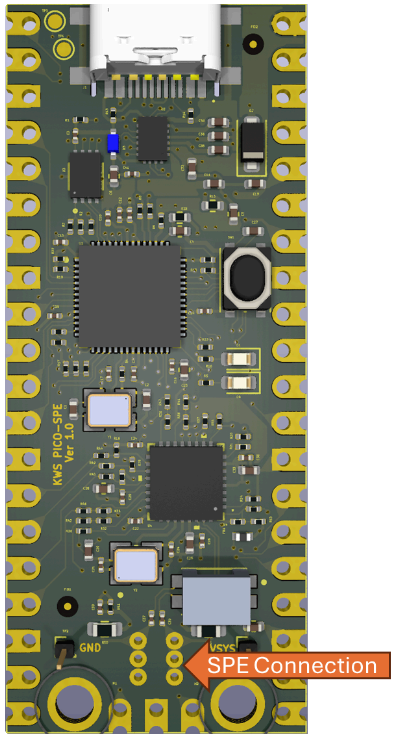


Figure 19. The SPE Connection

Full details of the Microchip LAN8651 can be found on the Microchip LAN8651 Datasheet

4.9 Debugging

Pico-SPE brings the RP2040 Serial Wire Debug (SWD) interface to a 3 pin debug header on the lower edge of the board. To get started using the debug port see the Getting started with Pico-SPE