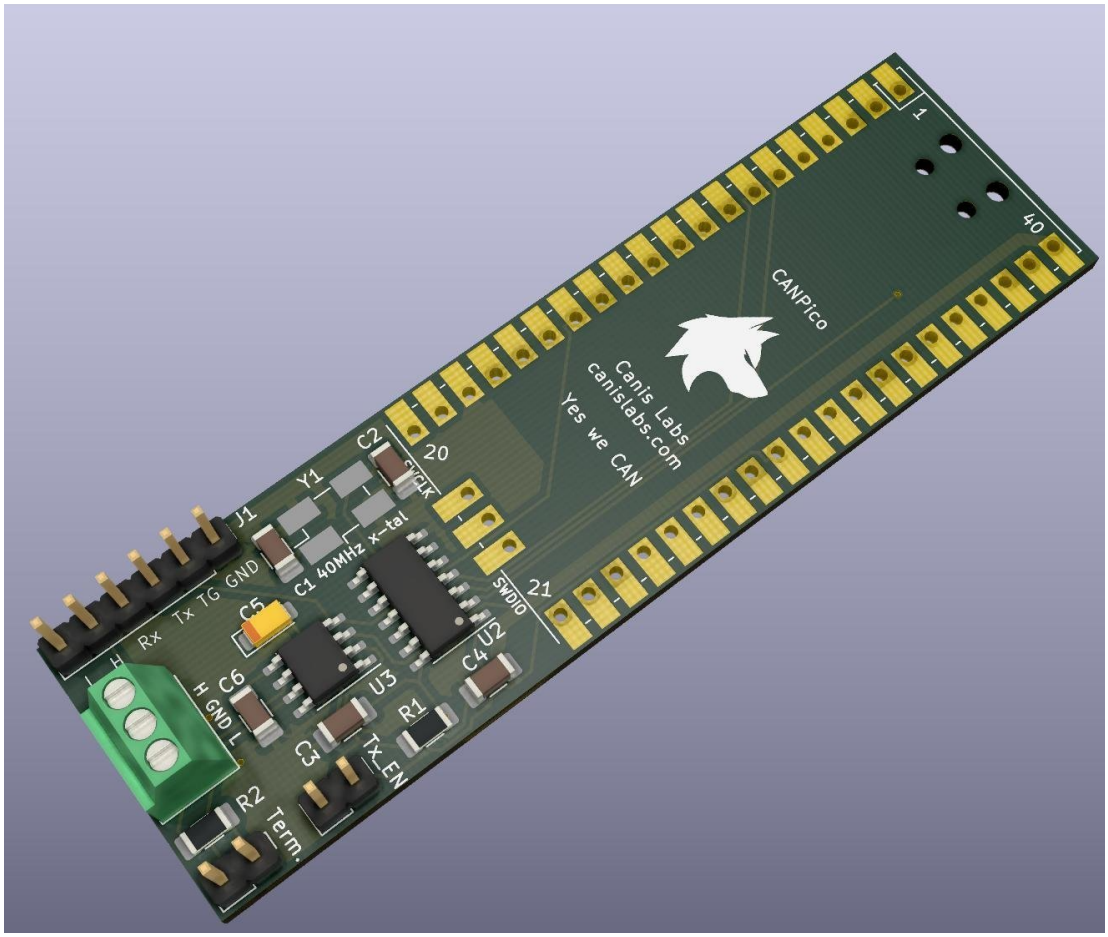
CANPico
Hardware Reference Manual

Document number	2103
Version	1
Issue date	2021-04-29

1 Introduction

The CANPico is a 'sock' board designed to connect a Raspberry Pi Pico board to a CAN bus. A Pico board is soldered down on to the upper area of the board and the lower area of the board contains the CAN connectors.



CAN bus is a protocol used in many applications, from trucks to buses to aircraft to agricultural equipment to medical devices to construction equipment and even spacecraft but its most common use is in cars.

Copyright © 2021 Canis Automotive Labs Ltd.

WARNING. Connecting the CANPico directly to a vehicle CAN bus comes with risk and should not be undertaken without understanding this risk. Grounding a vehicle chassis through the USB port of a Pico connected to a laptop, PC or USB hub powered from the mains may cause permanent damage to the vehicle's electronics and/or the Raspberry Pi Pico and devices it is connected to. The CAN transceiver will tolerate a ground potential difference ("ground offset") of between -2V/+7V. Connecting pin 2 of the screw terminal to the target system's ground will establish a common ground reference. The CAN bus must be properly terminated (with 120Ω resistors at either end). If the CAN bus is already terminated then the termination jumper on the CANPico should not be closed. In addition, causing disruption to a vehicle's CAN bus traffic may be interpreted by the vehicle fault management systems as a hardware fault and cause the vehicle to permanently shut down CAN communications with consequent loss of functionality.

The CANPico has the following features:

- Microchip MCP2517/18FD CAN controller
- Microchip MCP2562FD CAN transceiver
- Jumper header to enable a 120Ω CAN bus termination resistor
- Jumper header to enable the CAN TX line to the transceiver[1]
- Screw terminal connector to CAN bus
- CAN controller interrupt and Start-of-Frame inputs to GPIO pins
- Direct access via GPIO pins to the CAN transceiver TX and RX pins
- Test instrument header (for an oscilloscope or logic analyzer)

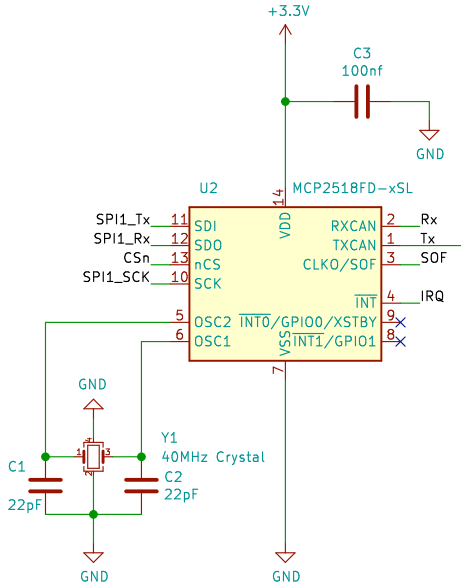
The features are described in more detail in the following sections. Appendix A contains the CANPico schematics.

[1] Normally connected, removed to prevent software from transmitting on the CAN bus

Copyright © 2021 Canis Automotive Labs Ltd.

2 CAN controller

The CAN controller used in the CANPico is the Microchip MCP2518FD (or in some cases the older MCP2517FD). The MCP2517/18FD is driven via SPI:



The CAN controller is clocked by a 40MHz crystal and the SPI clock from the RP2040 on the Pico can go up to 18.5MHz. More details on the SPI interface can be found in section 4 of the CAN controller datasheet (available from Microchip).

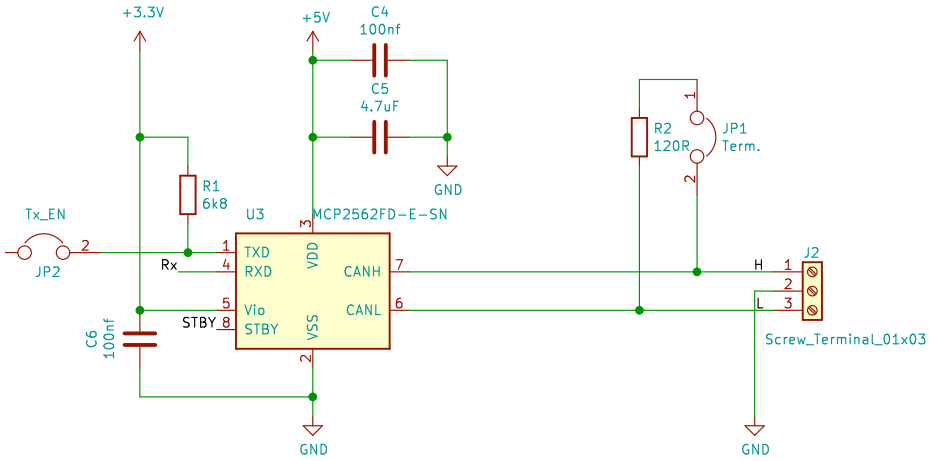
The controller can be programmed to generate an interrupt (level sensitive, active low) and this is routed to a GPIO pin on the Pico. In addition, a start-of-frame (SOF) signal is routed to another GPIO pin on the Pico, to allow timestamps in the controller for SOF to be correlated with a timer on the Pico (by latching a free-running counter at the rising edge of the SOF signal).

The SOF signal is multiplexed with a clock signal and on power-on-reset the line is clocked at 4MHz until software re-purposes it to SOF.

Copyright © 2021 Canis Automotive Labs Ltd.

3 CAN physical layer

The connection to the CAN bus driven by the MCP2562FD transceiver and is connected to the bus as shown below:



The TXD input to the transceiver comes from either the CAN controller or a GPIO pin from the Pico (the latter is to allow software to emulate aspects of the CAN protocol). The signal goes through jumper JP2 ("Tx_EN"): if this is not closed then a pull-up resistor holds TXD high and no software can send signals on the CAN bus (this is useful if software on the Pico is to monitor a CAN bus without the possibility of disturbing it).

The STBY input is driven by a GPIO line from the Pico. For normal operation this should be driven low; to put the transceiver into low-power standby mode it should be driven high.

A standard 120Ω bus termination resistor (required at each end of a CAN bus) can be included by closing the jumper JP1 ("Term.").

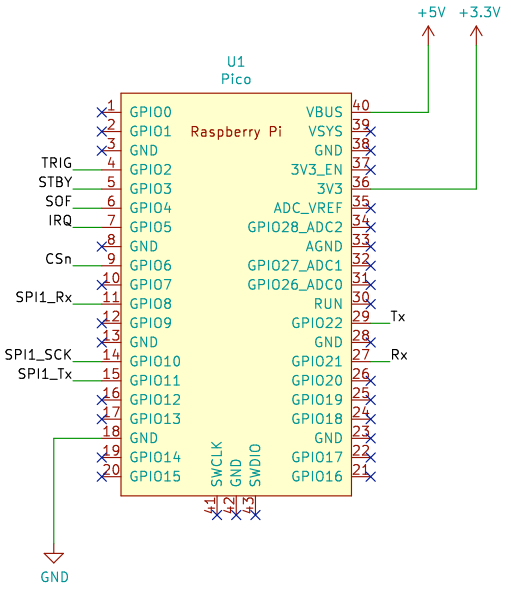
The connection to CAN is via a screw terminal, which includes GND on Pin 2.

Great care should be taken when connecting the CANPico to a vehicle CAN bus. The transceiver will tolerate a ground potential different ("ground offset") of between -2V / +7V. Any offset present might affect achievable bit rates. It is advisable to always connect Pin 2 to the target system's ground to establish a common ground reference and avoid problems.

Copyright © 2021 Canis Automotive Labs Ltd.

4 Raspberry Pi Pico connections

The interface between the CANPico and the Raspberry Pi Pico is shown below:



The Pico should be soldered to the CANPico at the 13 pins shown in the schematic above. The Pico has castellated ("half moon") edges to support this, allowing headers to be soldered into the holes if an additional Pico-compatible board is to be connected. The Pico SPI1 interface[2] is used to connect to the CANPico, leaving SPI0 free to connect to other boards.

The TRIG pin (GPIO2) connects to the test instrument header. The intention is that software can be used to evaluate conditions (e.g. receiving a CAN frame with a specific ID or payload) that will trigger a logic analyzer or oscilloscope to show what was happening on the CAN line around the event.

The STBY pin (GPIO3) is used to switch the CAN transceiver between standby and normal mode.

The SOF pin (GPIO4) is wired to the CLK0/SOF pin of the CAN transceiver. On power-up this is a 4MHz clock until software re-purposes the input. If an edge-sensitive SOF interrupt is required it will be important to initialize the CAN controller over SPI **before** enabling interrupts on GPIO4.

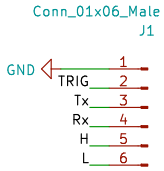
The Tx and Rx pins (GPIO22 and GPIO21) are wired to the CAN transceiver to allow direct software control (e.g. bit-banging the CAN protocol). This pin assignment is compatible with the CANHack pin assignment. If the Tx pin is driven by software then the CAN controller must be put into open drain mode output mode before the GPIO22 is enabled as an output (this prevents both the CAN controller and software directly driving the same line).

[2] Note that the chip select pin CSn is to be driven by software (the hardware chip select included in the RP2040 SPI interface is not capable of driving the MCP2517/18FD SPI interface)

Copyright © 2021 Canis Automotive Labs Ltd.

5 Test instrument header

A test instrument header is included to allow a logic analyzer and/or oscilloscope to monitor the CAN bus. The schematic for this header is shown below:



The TRIG pin is controlled from GPIO2 on the Pico. This can be driven by software to trigger a test instrument when certain conditions are observed. It can also be used as a general-purpose debugging pin (e.g. for measuring the execution time of sections code code).

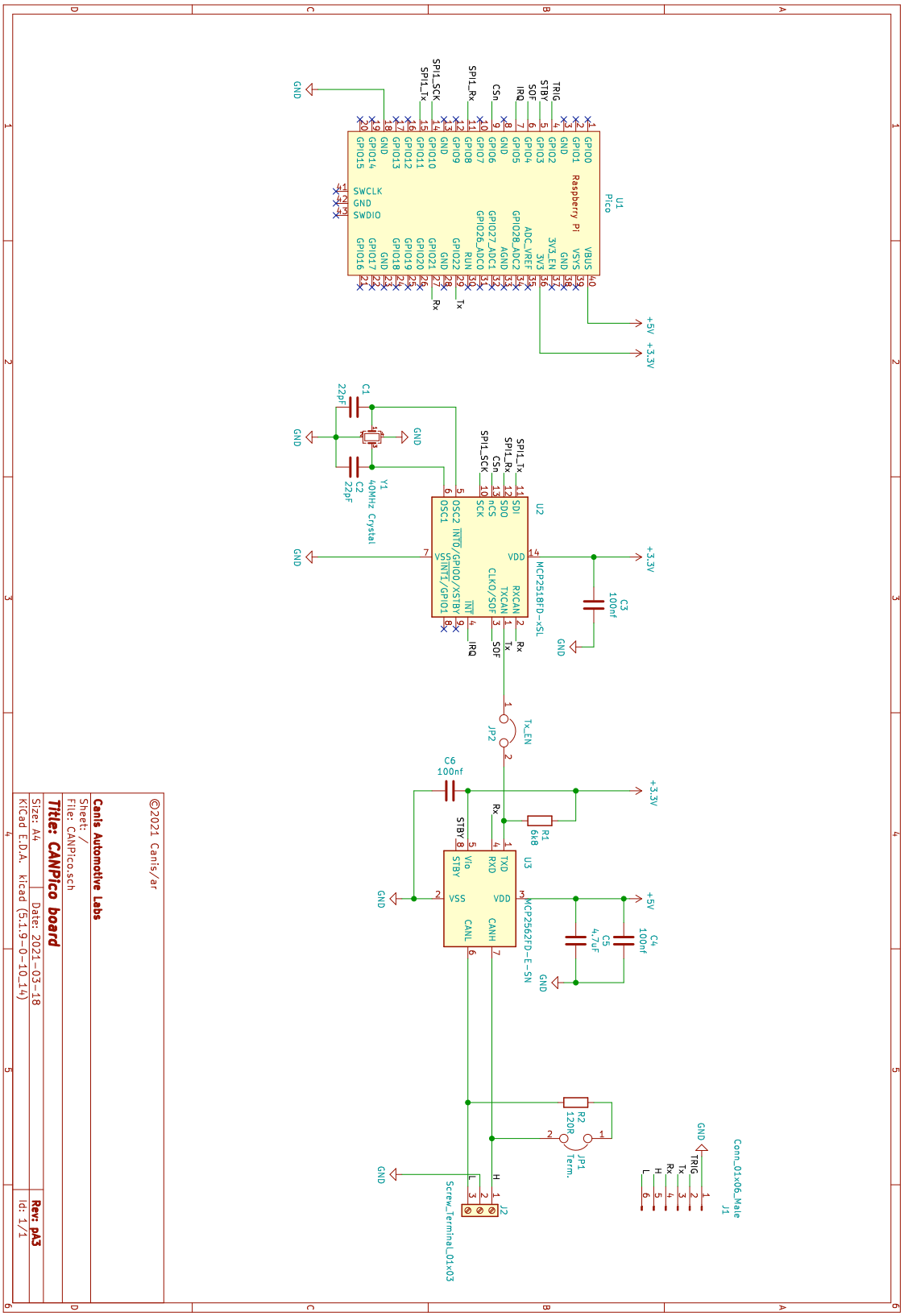
The Rx pin is connected to the CAN transceiver's RXD pin. Typically this would be an input to a logic analyzer monitoring the CAN bus.

The Tx pin is the output of the CAN controller and GPIO22.

The H and L pins are directly connected to CAN H and CAN L bus lines. Typically these would be inputs to an oscilloscope monitoring the CAN bus.

Copyright © 2021 Canis Automotive Labs Ltd.

6 Appendix A: Schematics



Copyright © 2021 Canis Automotive Labs Ltd.