Breadboard Buddy

Benjamin Crall

Rev 0.1 - March 10

1 Introduction

2 Specs

2.1 Power

Parameter		Min	Typ	Max	Unit
Input Voltage	USB		5		V
	EXT	5	12	30	V
	AIN	0		3.3	V
	GPIO	0		3.3	V
Input Current	USB			0.5	A
	EXT			2	A
	GPIO			10	mA
Output Voltage	3v3		3.3		V
	3v3A		3.3		V
	GPIO	0		3.3	V
Output Current	3v3	0.5		2	A
	3v3A			300	mA
	GPIO			10	mA

2.2 System

- Microcontroller: RP2040
- Flash: 2MB

2.3 Peripherals

- 16 GPIO pins capable of input with pull up/down and constant or PWM output.
- 4 analog inputs
- 1 LED connected to GPIO16
- 1 barrel jack switch connected to GPIO17
- 1 button connected to GPIO18

3 Setup

3.1 Wiring

Connect the Breadboard Buddy to a breadboard's power rails. Ensure that the positive and negative outputs go to the red and blue sides of each rail. Connect external circuitry to the GPIO or analog pins. If you are doing a lot of analog circuitry, you may with to configure the right rail to be disconnected and manually connect 3v3A there instead. If you are using any voltage higher than 3.3V in your circuit, ensure that it can not reach any of the input pins to the microcontroller.

Connect the USB port of the Breadboard Buddy to your computer, and open and connect your favorite serial monitor.

3.2 Configuration

The right power rail can be configured to output 3v3, EXT, or USB by moving the jumper near the barrel jack. Removing this jumper disconnects that power rail.

Each pin that you want to be an output will have to be explicitly set up as such by using the 'mode' command. GPIO pins can be set up as digital inputs or outputs, or PWM outputs. Analog pins can only be set up as analog inputs.

3.3 Firmware

The Breadboard Buddy firmware runs on CircuitPython. To Install it:

1. Download CircuitPython for the Raspberry Pi Pico from https://circuitpython.org/board/raspberry_pi_pico/
2. Connect the Breadboard Buddy to your computer. Disconnect anything else from the Breadboard Buddy.
3. Press and hold the `Reset` button, then press and hold the `Boot` button
4. Release the `Reset` button then release the `Boot` button.
5. The Breadboard buddy should appear as a USB mass storage device named `RPI-RP2` . If this does not happen, try steps 3 and 4 again, waiting longer between actions. If the issue precisest, you have an issue with your hardware.
6. Copy CircuitPython file you just downloaded onto the Breadboard Buddy. The file should be `adafruit-circuitpython-raspberry_pi_pico-en_US-<version>.uf2`. The Breadboard Buddy should reboot when the upload is complete.
7. The Breadboard Buddy should now appear as a USB mass storage device named `CIRCUITPY`. If this does not happen, try again from step 3. If the issue precisest, you have an issue with your hardware.

8. Copy `code.py` from `firmware` onto the Breadboard Buddy. The Breadboard Buddy should blink its light several times. You have now installed the Breadboard Buddy firmware.

It is possible to run other software on the Breadboard Buddy either through CircuitPython or by flashing it yourself. This guide will not help you with that.

4 Usage

The Breadboard Buddy is interfaced with by sending commands over serial and reading their response.

4.1 Pins

4.1.1 Pin Fields

For commands that take a pin as a field, there are several ways the pin can be specified.

By number: Pins can be specified by number. Digital pins are 0-15. Analog pins are ain0-ain4. Example: `5`, `12`, `3`, `ain2`

By Name: Pins can also be set up to be identified by a name. Names can be used in place of numbers. Names must be unique for each pin, and each pin can only have 1 alias. Names may contain letters and numbers, but no dashes, spaces, or other characters. Names must start with a letter. Names are not case sensitive, but the given case is stored. Example: `LED`, `BitClk`, `Q`, `Addr2`

By range: Pins can be specified by a range. Ranges are specified by a start pin then a dash then the end pin. Ranges are inclusive. The range will be evaluated in the order given. If names are used, they will be converted to pin numbers before the range is calculated. Names and numbers can be mixed. You may not mix analog and digital pins in a range. Example: `0-7`, `ADDR0-ADDR4`, `ain0-ain3`

By list: Pins can be specified by a list. Lists are comma separated. The command will execute on each pin in the order they are specified. Names and numbers can be mixed. Example: `1,2,8,4`, `ADDR0,ADDR1,ADDR2`

4.1.2 Pin States

Digital: Digital pins can be either high, low, or a PWM value. When specifying the state of a pin, you can use

- `H`, `HIGH`, `1`, `ON` for high output
- `L`, `LOW`, `0`, `OFF` for low output
- Numbers from 0.000 to 1.000 (only if the pin is configured as PWM)

Note that `ON` and `OFF` should be avoided if you are using active low logic to avoid confusion since they do not reverse for active low signals.

Analog: Analog pins will return the voltage on that pin in the form `#.###`. Analog pins can not be set to values.

4.2 Commands

There are two forms of commands, questions and instructions.

Instructions: An instruction tells the Breadboard Buddy to do something. This could be to change a configuration field, or set an output value. Most instructions will take the form `<instruction> <field> <value>`. If no value is provided, the command will be assumed to be a question.

Questions: Questions are asked in the form `<question>[?] <field>`. While the question mark is not required, it is encouraged since any command given with a question mark will be interpreted as a question, even if values are given (in which case they will be ignored).

Some commands have a short form and a long form. The short form is the part of the long form shown in CAPS. Either form is acceptable. No part of a command is case sensitive. Multiple fields are separated by spaces. Commands listed with a `?` are read only.

Command	Fields	Action
Mode	Pin mode	Sets/gets the mode of a digital pin
Name	Pin name	Sets/gets the name of a pin
Digital	Digital pin states	Sets/gets value of a digital pin(s)
Analog?	none	Gets the value of an analog pin(s)
Read?	none	Gets the value on any pin(s)
Bin	‘0‘,‘1‘	Sets/gets the value of pin(s) in binary

4.2.1 Mode

The mode command sets the mode of the digital pins. If queried, returns the configuration of the pin. The options for mode are:

`OUTPUT`: sets the pin up as an output and sets the pin's value to low. Output pins can can drive their value to either `HIGH` or `LOW`.

`INPUT`: sets the pin up as an input pin without any pull. Input pins are left floating and must be driven externally for constant results.

`PULLUP`: sets the pin up as an input pin with a weak pull up. The pin will read as high unless something externally drives it low.

`PULLDOWN`: sets the pin up as an input pin with a weak pull up. The pin will read as low unless something externally drives it high.

PWM: sets the pin up as a Pulse Width Modulated output. PWM outputs can output psudo-analog values by alternately driving the output high and low and changing the ratio of high to low.

Example

Task	Sets up pin 4 as an output
Command	`mode 4 OUTPUT`
Response	`OK`

4.2.2 Name

Handles aliases for pins. The first field is the new name for that pin. Names may only contain letters and numbers, and must contain at least one letter. If the pin is specified by name, the pin will be renamed. Only a single pin may be specified when setting a name. If queried by number, the name of that pin is returned. If queried by name, the number of that pin is returned.

Example

Task	Sets the alias for pin 14 to `CLK`
Command	`name 14 CLK`
Response	`OK`

Example

Task	Gets the alias for pin 14
Command	`name?  14`
Response	`CLK`

4.2.3 Digital

The digital command interacts with digital pins. The field is the value to set. Setting a PWM pin to high or low will result in 100% or 0% duty cycle respectively. Attempting to set input pins has no effect. Querying output pins returns their set value. Querying input pins returns the value read from that pin. The result of the query will be `H`, `L`, or `0.###`

Example

Task	Get the values from pins 8-11
Command	`digital?  8-11`
Response	`HLHH`

Example

Task	Set the values from pins 8-11 to LHHL
Command	`digital 8-11 L,H,H,L`
Response	`OK`

4.2.4 Analog

Queries analog pins. The Analog command is always assumed to be a query. The result of the query will be the voltage on the analog pin in the form `#.###`

Example

Task	Get the values from pins ain2
Command	`analog? ain2`
Response	`2.851`

4.2.5 Read

Reads the value of one or more pins. You can mix analog and digital pins. The Read command is always assumed to be a query. The result of the query will be the same as would be returned from `Digital?` and `Analog?` for each pin.

Example

Task	Get the values from pins 0-5
Command	`read? 5-0`
Response	`HHLHLH`

4.2.6 Binary

Question: Gets the digital value of pins. The result will be `0` or `1`. If an analog pin is specified, the output will be `0` if the voltage is less than half of the maximum, `1` otherwise. An optional field specifies the number format, either `DEC`, `HEX`, or `BIN` (default). The number will be padded so that the number of characters won't change depending on the values of the pins.

Instruction: Sets the pin(s) based on the binary value of the field value. The pins are specified MSB first. If the field has more bits than the number of pins, the most significant bits will be truncated. If the field has less bits than the number of pins, the value will be padded on the MSB end. An optional second field specifies the number format, either `DEC`, `HEX`, or `BIN` (default).

Example

Task	Get the values from pins 0-5 as a binary number where 0 is the LSB
Command	`binary? 5-0`
Response	`110101`

Example

Task	Read the binary encoded number on pins 0-3 in hex
Command	`binary? 3-0 HEX`
Response	`C`

Example

Task	Set the binary encoded number on pins 0-3 to 0x5
Command	`binary 3-0 5 HEX`
Response	`OK`

5 Theory Of Operation

5.1 The Power System

5.1.1 Terminology

Rails are physical power distribution systems found on the sides of a breadboard.
Domains are the power nets on the PCB. Some domains can connect to the rails.

5.1.2 Power Input

USB: You can power the board using the USB connector you use to communicate with it. The USB port does not support USB-PD. The USB port is protected by a 500mA polyfuse as well as a diode allowing safe connection of both power sources simultaneously. When using USB as the power source, you should limit 3.3V to 500mA.

EXT: You can also power the board form an external power supply with a 5.5x2.1mm center positive barrel jack. You can supply any voltage from 5V to 30V. This jack is protected by a 2A polyfuse and a diode. When using the barrel jack, the full 2A output of the 3v3 regulator is available as long as the external power supply can support that much power.

5.1.3 Power Output

The left power rail is always powered from 3v3. The right power rail can be configured to be powered from 3v3, EXT, or USB, or it can be left unpowered. This allows you to provide your own power source to this rail. This rail can be set to USB even if EXT is present.

5.1.4 Power Domains

Domain	Description	Source	Voltage	Current	Curret Limiter
3v3	Digital supply	VBUS	3.3V	2A or 500mA	Reg IC
EXT	External input	Barrel jack	5-30V	Max 2A	2A Polyfuse
3v3A	Analog supply	VBUS	3.3V	300mA	Reg ID
USB	USB input	USB-C port	5V	500mA	500mA Polyfuse
VBUS	System power	EXT or USB	5-30V	2A or 500mA	

3v3 (aka 3v3D): This is the main 3.3V power domain. It is powered by a switching voltage regulator from VBUS. It is used for the digital parts of the board, as well as the 3.3V output on the left power rails, and optionally the right power rails. When powered from EXT, it can supply up to 2A, although this may be limited by the board's power source. When powered from USB, it can supply up to 500mA. This domain should be used for all external digital circuitry.

EXT: This is the external power source provided through the barrel jack. It is used to supply the voltage regulators if it is present. It can also be sent to the right power rails if desired. It has a total capacity of 2A which may be limited by the external power supply, and it is shared between VBUS and the breadboard power rails.

3v3A: This is the analog supply. It is powered by a linear voltage regulator. It is used for the analog parts of the RP2040, and is made available for external analog circuitry on the analog pin header. This domain should be used for all external analog circuitry to reduce power supply noise.

USB: This is the USB power input. It is used to supply the voltage regulators if EXT is not present. It can also be sent to the right power rails if desired. It has a total capacity of 500mA which is shared between the rails and VBUS if EXT is not present.

VBUS: This is the power domain that powers the voltage regulators. It is powered by EXT if it is present, or USB. It is not accessible externally.

6 Credits

- The center positive symbol came from Wikipedia . The image is in the public domain.
- The firmware is built on top of CircuitPython