# Raspberry Pi HAT+ Specification

# Colophon



build-date: 2023-12-20
build-version: 5cb8d61-dirty

## 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.

# Table of contents

# Chapter 1. Introduction

Since the 2014 release of Raspberry Pi Model B+, all Raspberry Pi single board computers (SBCs) feature a 40-pin (2×20-way) 0.1-inch-pitch GPIO header.

The GPIO header provides power (5V and 3.3V), ground pins, and 28 GPIO pins.

These GPIO pins are +3.3V digital I/O, have programmable pulls, and support direct digital access from the processor. This allows you to set pins high, low or tristate; and to enable pull up, pull down, or no pull.

In addition, certain groups of GPIO pins can support alternative fixed function peripherals such as I2C, UART, SPI, PWM etc.

**NOTE**

Raspberry Pi 4 and Raspberry Pi 5 support a much larger number of alternate peripherals than Raspberry Pi 3 and earlier generation products.

## 1.1. Original HAT Specification

The original HAT specification is now deprecated. All new HATs should follow the HAT+ specification.

This document is an update to the original Hardware-Attached-on-Top (HAT) specification for add-on boards that sit on top of the Raspberry Pi and connect to the 40-pin connector to provide extra peripheral features. The original HAT specification and associated files can be found at https://github.com/raspberrypi/hats. It will eventually be archived or removed, and replaced by this document.

The main changes with the HAT+ specification are:

- HAT+ boards must be electrically compatible with the `STANDBY` power state, where the 5V power rail is powered, but the 3.3V rail is unpowered.

**NOTE**

Raspberry Pi 4 and Raspberry Pi 5 support the `STANDBY` state, while older Raspberry Pi models do not.

- The specification is less prescriptive about HAT physical dimensions.
- The HAT EEPROM content is now much simpler.
- A special class of HAT+s that can be stacked with an extra HAT+ on top is supported for a maximum stack of two HATs.

**IMPORTANT**

HAT+ boards are electrically backwards-compatible with older Raspberry Pi models, but may need up-to-date software and firmware to function correctly.

## 1.2. Why follow the HAT+ specification?

You should follow the HAT+ specification to ensure consistency and compatibility with future add-on boards, and to allow for a better end-user experience, especially for the less technically aware users.

**NOTE**

You can design an add-on board for a Raspberry Pi that does not follow this specification, but you will not be able to market it as a HAT+ board.

# Chapter 2. HAT+ Requirements

A board can only be called a HAT+ if it follows these minimum requirements:

- It plugs into the Raspberry Pi 40-way GPIO header.
- If the HAT+ board is powered from the 40-way GPIO header, it must be electrically compatible with the `STANDBY` power state, where the 5V power rail is powered, but the 3.3V rail is unpowered.
- It has a suitable ID EEPROM attached to the `ID_*` pins, using 3.9KΩ pullups to 3.3V, conforming to the HAT+ ID EEPROM specification.
- Mechanically, the board can plug into the 40-way GPIO header and can be mechanically attached to the Raspberry Pi using at least one of the mounting holes on the SBC.
- If supplying power to the Raspberry Pi, the board is a Power HAT+. It must be able to supply at least 3A at 5.1V, but it is strongly recommended that Power HAT+s should support 5A at 5.1V.

**NOTE**

13W PoE HAT+s are an allowed exception to these minimum power requirements.

# Chapter 3. Power States

## 3.1. Raspberry Pi Power States

Raspberry Pi supports the following power states:

**OFF**

No power connected to the board (the board unplugged).

**WARM-STANDBY**

The Raspberry Pi is halted/off, but all of the power rails are still enabled. This is the default mode when doing a `sudo halt` or soft power-button-off operation.

**STANDBY**

The Raspberry Pi has the 5V rail powered — so the power management chip is powered — but no other power supplies on the PMIC and board are enabled. You can configure `sudo halt` or power-button-off using the EEPROM to enter this mode instead of `WARM-STANDBY`.

**SLEEP**

Some rails are off — notably the CPU core — and Linux is in suspend-to-RAM state. Pressing the power button will cause the system to move to the `ACTIVE` state.

**ACTIVE**

All rails are up and everything is running, e.g. running desktop Linux.

**NOTE**

The `SLEEP` state is not currently supported on Raspberry Pi 5.

Historically, when a Raspberry Pi was shut down via `sudo halt` it would end up in the `WARM_STANDBY` state, with both the 5V and 3.3V rails still powered. Raspberry Pi 4 and Raspberry Pi 5 boards support the new `STANDBY` state, where the 5V power exists but everything else is turned off (so 3.3V is not powered). A HAT+ board must not assume any particular sequencing timing between the 5V and 3.3V rails, and must be electrically compatible with the `STANDBY` state.

## 3.2. GPIO Power-on State

The default power-on state for GPIO pins on the 40-way connector treats all pins as inputs with either a weak pad pull-up or pull-down. A HAT+ must tolerate weak pull-high or pull-low on any of its used GPIO pins, and must not assume that these weak pulls are applied or removed simultaneously.

Therefore, if a HAT+ needs a specific pull state at power on, it is best to provide an external pull rather than relying on the internal pad pull.

The only exceptions to the above are `GPIO2` and `GPIO3` which have on-board ~2KΩ pulls to 3.3V. GPIO pins `ID_SC` and `ID_SD` (`GPIO0` and `GPIO1`) are reserved solely for board detection and identification.

**IMPORTANT**

The only permitted connections to the `ID_` pins are an ID EEPROM and the required 3.9KΩ pull-up resistors to 3.3V. Do not connect anything else to these pins.

# Chapter 4. HAT+ ID EEPROM

At boot time, the Raspberry Pi firmware will probe the `ID_SD` and `ID_SC` pins to look for an EEPROM at one of the allowed EEPROM addresses. If one is found, the firmware will read the EEPROM, which must have data in it conforming to the HAT+ EEPROM specification minimum requirements. See Appendix A for details.

The ID EEPROM content provides:

- A product UUID (required)
- A product ID (optional – zero if not used)
- A product version (optional – zero if not used)
- A vendor name string (required), e.g. “ACME Technology Company”
- A product description string (required), e.g. “Special Sensor Board”
- A Device Tree overlay name string (required)
- Other user defined data (optional)

For a HAT+, GPIO pin and driver setup is performed by loading the Device Tree overlay which is named in the EEPROM. The overlay is not stored in the EEPROM itself, but lives in `/boot/overlays` on the filesystem.

**NOTE**

All of the pin configuration and driver information is loaded by the overlay.

The overlay name goes through the `overlay_map` translation mechanism, allowing different families of Raspberry Pi to have different implementations.

General documentation, including how to build overlays, can be found as part of our online documentation and in the overlays README file on GitHub. To add an overlay for a HAT+, please open a Pull Request on our Linux GitHub repository.

## 4.1. EEPROM Addresses

A new feature of the HAT+ specification is allowing several EEPROM addresses, which provide:

- The option to stack one standard HAT+ on top of special types of single-stackable HAT+
- The option to use the EEPROM address as one bit of information for new Power HAT+ boards

Permitted EEPROM I2C addresses take the form `7'b101_00XY`, where `XY` are:

`00`
: standard HAT+ (or legacy HAT)

`01`
: stackable HAT+ (e.g. Raspberry Pi M.2 M Key HAT)

`10`
: stackable Power HAT+ (MODE0)

`11`
: stackable Power HAT+ (MODE1)

The Power HAT addresses are reserved for HATs where the mode of the HAT is determined by its address. The firmware treats the address as a power mode variable input to the associated overlay.

## 4.2. Stackable HAT+s

We introduce the idea of a single-stackable HAT+, which only uses the `ID_*` pins and must not electrically connected to GPIOs 2-27.

A single-stackable HAT+ can only have a non-stackable (standard) HAT or HAT+ jointly stacked with it.

**NOTE**

Single-stackable HAT+s will not be backwards compatible with older firmware.

## 4.3. EEPROM Device Specification

A 24Cxx type 3.3V I2C EEPROM must be used.

**NOTE**

Some types are 5V only. Do not use these.

The EEPROM must be of the 16-bit addressable type (do not use EEPROMs with 8-bit addressing). Do not use a 'paged' type EEPROM where the I2C lower address bit(s) select the EEPROM page. The EEPROM is only required to support 100kHz I2C mode.

Devices that perform I2C clock stretching are not supported.

The write protect pin must be supported, and protect the entire device memory.

**NOTE**

Due to the restrictions above, many of the smaller I2C EEPROMs are ruled out - please check datasheets carefully when choosing a suitable EEPROM for your HAT+.

A recommended part that satisfies the above constraints is the OnSemi CAT24C32 which is a 32kbit (4kbyte) device. The minimum EEPROM size required is variable, and depends on the size of the vendor and device tree overlay name strings, as well as any other vendor-defined data in the EEPROM.

It is recommended that the EEPROM WP (write protect) pin be connected to a test point on the board and pulled up to 3.3V with a 1KΩ resistor. At board test/probe the EEPROM can be written (WP pin can be driven LOW), but this means there is no danger of a user accidentally changing the device contents once the board leaves the factory.

**NOTE**

The recommended device has an internal pull down, hence the stiff (1KΩ) pull up is required.

**NOTE**

On some devices WP does not write-protect the entire array; we have observed this behaviour in some Microchip variants, for example, so avoid using these!

# Chapter 5. Power HAT+s

If a HAT+ supplies power to the Raspberry Pi via the 5V GPIO pins, it is called a Power HAT+.

A Power HAT+ must only source current and tolerate accidental use while the USB-C power is also plugged in.

Standard AC/DC power supplies, such as the Raspberry Pi 15W and 27W power supplies, also do not sink current.

Power HAT+s (HAT+s which power the Pi via the 5V GPIO pins) must contain data in the Device Tree Overlay that describes their power capabilities (supplied voltage – which should be 5.1V - and current).

**NOTE**

For power HAT+s, the EEPROM address decodes to a powermode variable that is passed to the overlay (with value 0 or 1) at runtime.

**NOTE**

Power HAT+ can also be a regular HAT+, but this makes it non-stackable.

# Chapter 6. HAT+ Marking

It is strongly recommended that compliant HAT+ boards should display the HAT+ word mark on the board silkscreen so users know it is HAT+ compliant. See Figure 1.


*Figure 1. The HAT+ wordmark used on a silkscreen to indicate that the board is compliant.*

**NOTE**

For higher-resolution and silkscreen-compatible versions of the word mark, please get in touch with our Applications team by emailing applications@raspberrypi.com.

Alternatively, or in addition, HAT+ boards should use 'HAT+' in their name, e.g. "Raspberry Pi M.2 M Key HAT+".

While HAT+ boards that are stackable have no special extra markings, their user documentation must make it clear that they can have at most one standard HAT or HAT+ jointly stacked with them.

# Chapter 7. HAT+ Mechanical Specification

Unlike the original HAT specification, a HAT+ board only needs to connect to the 40-way GPIO header (including the `ID_*` pins), and to have at least one mechanical mounting hole align with one of the four Raspberry Pi mounting holes in order to be considered a valid HAT+ board.

The designer of the HAT+ is responsible for making sure that:

- The board does not foul the PoE headers on a Raspberry Pi 4 or a Raspberry Pi 5.

**NOTE**

The PoE headers are located in different places on Raspberry Pi 4 and Raspberry Pi 5.

- The board stack height has been considered, so board-to-board fouling of components is avoided.
- The board is packaged with a suitable stacking header, spacer and screws where required.
    - It is recommended that HAT+s use (or have the option to use) at least 15mm board-to-board spacers (16mm recommended). This allows the Raspberry Pi Active Cooler to fit between the Raspberry Pi and the HAT+, as long as there are no components on the underside of the HAT+ - if there are, larger spacers may be needed.
- The various Raspberry Pi camera, display, and PCIe flex connectors have been considered. A HAT does not have to provide cutouts for these, but it is desirable to do so if possible.

**NOTE**

HAT+s for specific models are allowed. Where a HAT+ is only compatible (electrically and/or mechanically) with a single Raspberry Pi board, or a sub-set of our boards, this must be clearly marked on the board and in the product documentation.

## 7.1. Mechanical examples

Shown below are three examples of official Raspberry Pi designs that can be used as a starting point for a HAT+ design: the Raspberry Pi Sense HAT in Figure 2, the Raspberry Pi M.2 M Key HAT+ in Figure 3, and the Raspberry PoE+ HAT in Figure 4.

*Figure 2. Mechanical diagram that can be used as a starting point for HAT+ design. Shows the layout of the Raspberry Pi Sense HAT.*

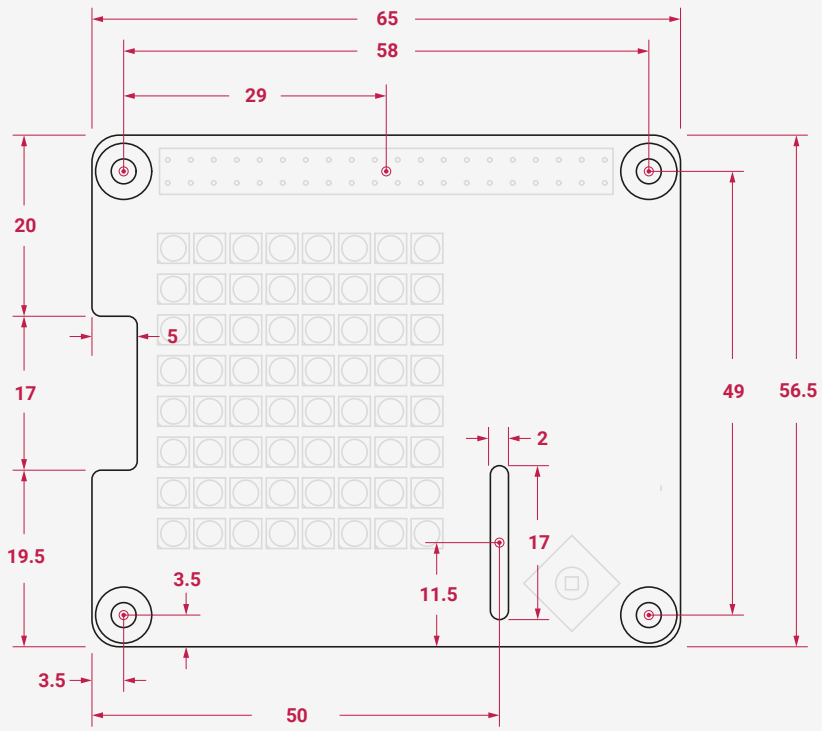



*Figure 3. Mechanical diagram that can be used as a starting point for HAT+ design. Shows the layout of the Raspberry Pi M.2 M Key HAT+.*

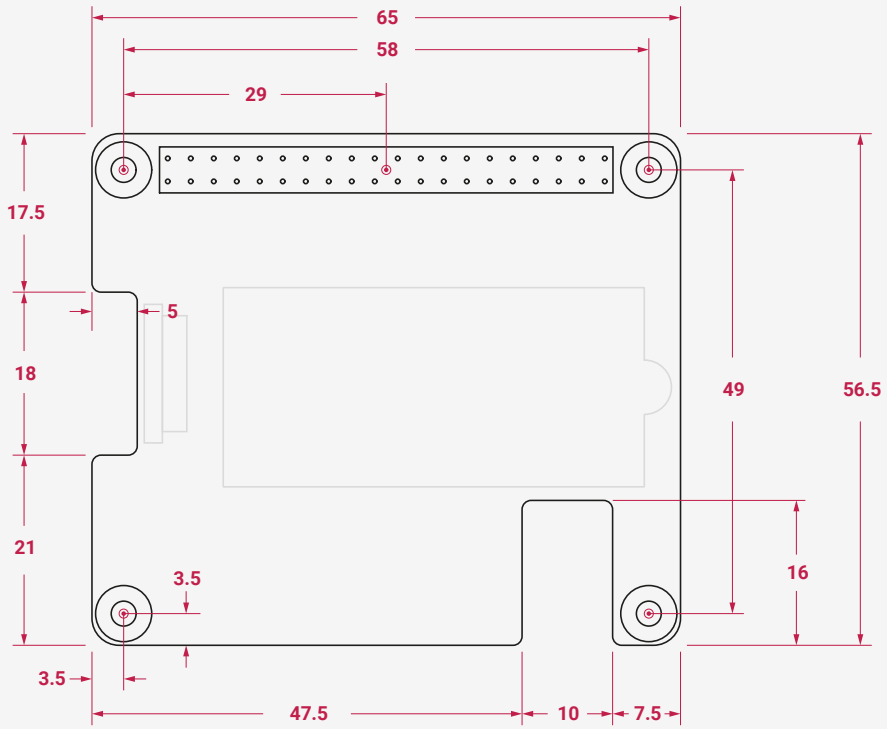
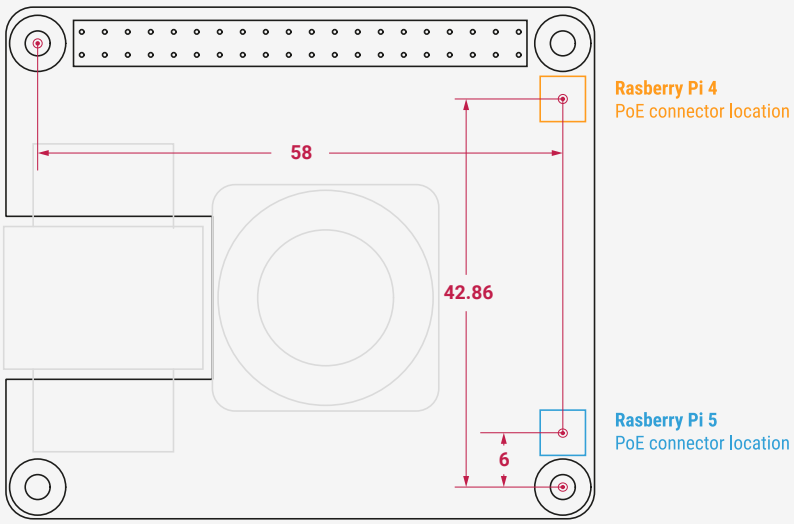


*Figure 4. Mechanical diagram that can be used as a starting point for HAT+ design. Shows the layout of the Raspberry Pi PoE+ HAT. The PoE header location for both Raspberry Pi 4 and Raspberry Pi 5 are shown.*



# Appendix A: HAT+ ID EEPROM Specification

Software tools and documentation for creating, flashing, and reading the HAT+ ID EEPROM can be found on Github. See Appendix B for more details.

## Data format specification

**NOTE**

All EEPROM strings either have an explicit length field or length can be inferred, so no strings should have a zero-terminating character.

Unlike the original HAT specification, HAT+ boards only contain the name of the device tree overlay. Raspberry Pi firmware then fetches this from `/boot/overlays` on the filesystem.

It is strongly recommended that `dt-overlay` names are of the form `manufacturername-hatplusname`, e.g. `rpi-sense-v2` or `iqaudio-dacplus`.

**IMPORTANT**

Names starting with `rpi-` are reserved for Raspberry Pi use.

The overall EEPROM structure is shown in Table 1, the structure of the HEADER block is shown in Table 2, while the structure of individual atoms is shown in Table 3.

*Table 1. EEPROM structure*

| Block | Description |
|---|---|
| HEADER | EEPROM header (required) |
| ATOM1 | Vendor info atom (required) |
| ATOM2 | Device Tree overlay name atom (required) |
| ⋮ | |
| ATOMn | n'th Device Tree overlay name atom |

*Table 2. EEPROM HEADER block*

| Bytes | Field | Description |
|---|---|---|
| 4 | signature | e.g. `0x52`, `0x2D`, `0x50`, `0x69` ("R-Pi" in ASCII) |
| 1 | version | EEPROM data format version **should be `0x02` for HAT+ specification** |
| 1 | reserved | Set to 0 |
| 2 | numatoms | Total atoms in EEPROM |
| 4 | eeplen | Total length in bytes of all eeprom data (including this header block) |

*Table 3. EEPROM ATOMx block*

| Bytes | Field | Description |
|---|---|---|
| 2 | type | atom type (see Table 4) |
| 2 | count | incrementing atom count |

| Bytes | Field | Description |
|---|---|---|
| 4 | dlen | length in bytes of data+CRC |
| N | data | N bytes, N = dlen-2 |
| 2 | crc16 | CRC-16 of entire atom (type, count, dlen, data) |

*Table 4. EEPROM ATOMx type*

| Atom type value | Description |
|---|---|
| `0x0000` | invalid |
| `0x0001` | vendor info (see Table 5) |
| `0x0002` | DO NOT USE |
| `0x0003` | device tree overlay name |
| `0x0004` | manufacturer custom data |
| `0x0005` | DO NOT USE |
| `0x0006-0xfffe` | reserved for future use |
| `0xffff` | invalid |

**NOTE**

The Device Tree overlay name atom data (type = `0x0003`) is a name string, e.g. "iqaudio-dacplus"

*Table 5. Vendor info atom data (type=0x0001):*

| Bytes | Field | Description |
|---|---|---|
| 16 | uuid | product UUID (unique product UUID) |
| 2 | pid | product ID |
| 2 | pver | product version |
| 1 | vslen | vendor string length (bytes) |
| 1 | pslen | product string length (bytes) |
| X | vstr | ASCII vendor string e.g. "ACME Technology Company" |
| Y | pstr | ASCII product string e.g. "Special Sensor Board" |

# Appendix B: HAT+ EEPROM Tools

The EEPROM tools are available as part of our utilities repository on Github.

Install the prerequisites - you will need at least version 3.10 of `cmake` and build and install the EEPROM tools as follows:

```
$ sudo apt install cmake
$ git clone https://github.com/raspberrypi/utils.git
$ cd utils/eeptools
$ cmake .
$ make
$ sudo make install
```

## Using the tools

Copy the `eeprom_settings.txt` template file to `myhat_eeprom.txt`. Then go ahead and edit the `myhat_eeprom.txt` file to suit your specific HAT, see [eprom-spec] for more details, and run

```
$ eepmake myhat_eeprom.txt myhat.eep
```

to create the `.eep` binary.

If `eepmake` has generated a product UUID for you, it is a good idea to patch your `myhat_eeprom.txt` to include and preserve it, or use `eepdump` to regenerate it.

To flash the EEPROM image you should disable EEPROM write protection. Sometimes this requires a jumper on the board, or sometimes this requires pulling a GPIO. Check your schematics to determine which.

In this specification the HAT EEPROM is connected to pins that can be driven by I2C0. However, this is the same interface as used by the camera and displays, so use of it by the ARMs is discouraged. The `eepflash.sh` script gets around this problem by instantiating a software driven I2C interface using those pins as GPIOs, calling it `i2c-9`:

```
$ sudo dtoverlay i2c-gpio i2c_gpio_sda=0 i2c_gpio_scl=1 bus=9
```

You can then check if the HAT is detected,

```
$ sudo apt install i2c-tools
$ i2cdetect -y 9 (should be at address 0x50)

   i2cdetect -y 9 0x50 0x50
       0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
   00:
   10:
   20:
   30:
   40:
   50: 50
   60:
   70:
$
```

and flash the `eep` file.

```
$ sudo ./eepflash.sh -w -t=24c32 -a=0x50 -f=eeprom.eep
```

Afterwards you should re-enable EEPROM write protection, by putting back jumper, or resetting the GPIO.

# Appendix C: Release History

*Table 6. Documentation release history*

| Release | Date | Description |
|---|---|---|
| 0.8 | 01 Dec 2023 | • Preliminary draft |
| 0.9 | 06 Dec 2023 | • Initial internal release<br>• Copy edit |
| 1.0 | 08 Dec 2023 | • Public release |
| 1.1 | 20 Dec 2023 | • Added information about EEPROM tools |