Raspberry Pi HAT+ Specification

A guide to designing Hardware-Attached-on-Top of a Raspberry Pi.

Raspberry Pi Ltd

Colophon

build-date: 2024-10-10
build-version: fb2e5d6-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.

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. Additionally, you can 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, and more.

NOTE

Raspberry Pi 4 and later support more alternate peripherals than earlier products.

1.1. Changes to the original HAT specification

The original Hardware-Attached-on-Top (HAT) specification is now deprecated. All new HATs should follow the HAT+ specification.

This document is an update to the original 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 github.com/raspberrypi/hats. It will eventually be archived or removed, and replaced by this document.

The HAT+ specification makes the following major changes to the HAT+ specification:

- HAT+ boards must be electrically compatible with the `STANDBY` power state, where the 5V power rail is powered, but the 3.3V rail is not powered. Raspberry Pi 4 and later models support the `STANDBY` state.
- The HAT+ specification is less prescriptive about HAT physical dimensions.
- HAT+ boards have simpler EEPROM content.
- The HAT+ specification introduces a special class of **stackable** HAT+s that can be combined with an extra HAT+.

HAT+ boards are electrically backwards-compatible with older Raspberry Pi models. However, you may need to update software and firmware on older Raspberry Pi models for full functionality.

1.2. Why follow the HAT+ specification?

You should follow the HAT+ specification to:

- ensure consistency and compatibility with future add-on boards
- allow for a better end-user experience, especially for less technically-aware users

Only boards that follow the HAT+ specification can be marketed as HAT+ boards. Add-on boards that do not follow the specification can still be used, but cannot use HAT or HAT+ markings.

Chapter 2. Power States

2.1. Raspberry Pi Power States

Raspberry Pi devices support the following power states:

`OFF`

No power connected to the board (unplugged).

`WARM_STANDBY`

The Raspberry Pi is off, but all of the power rails are still enabled. Usually a result of 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 causes the system to transition to the `ACTIVE` state.

`ACTIVE`

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

2.2. `STANDBY` mode support

Historically, Raspberry Pi devices shut down via `sudo halt` transition to the `WARM_STANDBY` state with both the 5V and 3.3V rails still powered. Raspberry Pi 4 and later support a new `STANDBY` state that only powers the 5V rails.

HAT+ boards must not assume any particular sequencing timing between the 5V and 3.3V rails, and must be electrically compatible with the `STANDBY` state.

2.3. 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 the GPIO pins at any time. HAT+s must not assume simultaneous application or removal of these weak pulls.

If a HAT+ needs a specific pull state at power-on, provide an external pull rather than relying on the internal pad pull.

`GPIO2` and `GPIO3` are exceptions to the above rules. These pins have on-board ~2KΩ pulls to 3.3V.

IMPORTANT

The only permitted connections to the `ID_SC` and `ID_SD` (`GPIO0` and `GPIO1`) pins are an ID EEPROM and the required 3.9KΩ pull-up resistors to 3.3V. These pins are reserved solely for board detection and identification. **Do not connect anything else to these pins.**

Chapter 3. HAT+ ID EEPROM

At boot time, Raspberry Pi firmware probes the `ID_SD` and `ID_SC` pins to look for an EEPROM at one of the allowed EEPROM addresses. When found, firmware reads the EEPROM. The EEPROM must have data in it conforming to the HAT+ ID EEPROM specification minimum requirements.

ID EEPROM content provides the following information:

- 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 (recommended)
- Other user defined data (optional)

For more information, see EEPROM specification.

The firmware loads the Device Tree overlay named in the EEPROM. This overlay determines all pin configuration and driver information for a HAT+. Raspberry Pi OS doesn't store the overlay in the EEPROM itself; instead, it lives in `/boot/firmware/overlays` on the filesystem.

The overlay name goes through the `overlay_map` translation mechanism, allowing different Raspberry Pi models to support different implementations.

For more information, including how to build overlays, see the Raspberry Pi overlay documentation and the overlays README on GitHub.

To add an overlay for a HAT+, open a pull request on the Raspberry Pi Linux GitHub repository.

3.1. HAT+ classes

The HAT+ specification includes several possible EEPROM addresses. These addresses provide support for stackable HAT+s and Power HAT+ power modes.

Permitted EEPROM I2C addresses use the form `101_00XY`, where the `XY` suffix defines the **class** of the HAT+. The following table defines the class defined by each valid suffix:

Table 1. EEPROM I2C address suffixes

Suffix	Class	Description
`00`	**Standard**	Any HAT+ that uses the typical HAT+ pins also used by legacy HATs, *including HAT+s that also provide power*. This class includes legacy HATs.
`01`	**Stackable**	Any HAT+ that only uses the `ID_*` pins and does not electrically connect to GPIOs 2-27. These HAT+s are considered *Stackable* because they can be *stacked* on top of a host Raspberry Pi simultaneously with a Standard HAT+ or legacy HAT. Stackable HAT+s are not backwards-compatible with older firmware.
`10`	**Power (`MODE0`)**	A HAT+ that provides sufficient power for a Raspberry Pi but not peripherals (~3A), that does not electrically connect to GPIOs 2-27. Power HAT+s are not backwards-compatible with older firmware.

Suffix	Class	Description
11	**Power** (`MODE1`)	A HAT+ that provides sufficient power for a Raspberry Pi *and* peripherals (~5A), that does not electrically connect to GPIOs 2-27. Power HAT+s are not backwards-compatible with older firmware.

Power HAT+ addresses determine how much power the host Raspberry Pi attempts to draw from the HAT+. Firmware treats the address as variable input to the associated overlay.

3.2. Stackability

You can connect multiple HAT+s to a single host Raspberry Pi.

You can connect one HAT+ from each class to a host Raspberry Pi at a time. This means that you can simultaneously use up to three HAT+s with a single Raspberry Pi, assuming your configuration can power all peripherals:

- one Standard HAT+ or legacy HAT
- one Stackable HAT+
- one Power HAT+

Add-on boards that do not include an EEPROM, including the Raspberry Pi PoE and Raspberry Pi PoE+ HATs, do not follow the HAT+ standard. As a result, you may or may not be able to combine them with other HAT+s. Check the documentation for your HAT for more information.

3.3. EEPROM Device Specification

When selecting EEPROM for your HAT+, adhere to the following requirements:

- Use a 24Cxx 3.3V I2C EEPROM. Do *not* use 5V-only EEPROM.
- Use 16-bit addressable EEPROM. Do *not* use EEPROMs with 8-bit addressing.
- Use EEPROM with a write protect (WP) pin that protects the *entire* device memory.
- Do *not* use EEPROM that performs I2C clock stretching.
- Do *not* use a paged EEPROM where the I2C lower address bits select the EEPROM page.
- Your EEPROM only needs to support 100kHz I2C mode.
- EEPROM size depends on the size of vendor-defined data, including vendor and device tree overlay name strings.

To make testing as easy as possible, connect the EEPROM write protect pin to a test point on the board pulled-up to 3.3V with a 1KΩ resistor. During board test and probe, you can write to the EEPROM by driving the write protect pin low. But once the HAT+ leaves the factory, there is little to no risk of consumers accidentally changing EEPROM contents.

The restrictions above exclude many smaller I2C EEPROMs. Check datasheets carefully when choosing EEPROMs for your HAT+.

3.3.1. Recommended Part

A recommended part that satisfies the above constraints is the OnSemi CAT24C32 which is a 32Kb (4KB) device. This device has an internal pull-down, hence the stiff (1KΩ) pull-up.

WARNING

On some devices, the write protect pin does not write-protect the entire array. We have observed this behaviour in some Microchip variants, for example; avoid using these.

Chapter 4. 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 connected to USB-C power. Standard AC/DC power supplies, such as the Raspberry Pi 15W and 27W power supplies, also do not sink current.

4.1. Power modes

For Power HAT+s, the EEPROM I2C address decodes to a power mode variable that is passed to the overlay (with value 0 or 1) at runtime. This is intended to be a communication channel between the hardware (e.g. a PoE HAT+) which can detect conditions at boot, kernel drivers, and user space code.

This has the following impact on the Device Tree applied:

- If the firmware detects a `MODE0` Power HAT+, it applies the "mode0" DT parameter.
- If the firmware detects a `MODE1` Power HAT+, it applies the "mode1" DT parameter.

If the firmware sees a `MODE1` Power HAT+, it automatically sets `usb_max_current_enable=1`.

For more information about power modes, see Table 1.

4.2. Define an overlay for a Power HAT+

Power HAT+s can describe their supplied power capabilities in their Device Tree overlay. To define this data in an overlay:

1. Define a Device Tree node, `/chosen/power`.
2. In that node, create a property (e.g. `hat_current_supply`) that holds a value in mA. Simple HAT+s can hard-code this into the overlay.

For more details about `/chosen/power`, see the Raspberry Pi documentation.

4.2.1. Example overlay

The following example demonstrates what part of a Power HAT+ overlay might look like:

```
fragment@2 {
    target-path = "/chosen";
    __overlay__ {
        power: power {
            hat_current_supply = <2500>; // mA
        };
    };
};
...
__overrides__ {
    ...
    mode0 =  <&power>, "hat_current_supply:0=3000";
    mode1 =  <&power>, "hat_current_supply:0=5000";
};
```

The default value of 2500 demonstrates how to encode a fixed value.

Chapter 5. HAT+ marking

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.

For higher-resolution and silkscreen-compatible versions of the wordmark, see the following resources:

- usage guidelines
- CAD symbol
- SVG, black
- SVG, white
- EPS, black
- EPS, white

For more information, contact applications@raspberrypi.com.

5.1. Product name conventions

HAT+ boards should use 'HAT+' in their name. For example, "Raspberry Pi M.2 HAT+ M Key".

While HAT+ boards that are stackable have no special extra markings, their user documentation must make it clear that they can be used simultaneously with a standard HAT or HAT+.

You can make a HAT+ that is only compatible (electrically or mechanically) with a specific Raspberry Pi model or subset of models. Board-specific HAT+s must clearly indicate the supported models both on the board and in product documentation.

Chapter 6. HAT+ electrical specification

A board can only be called a HAT+ if it adheres to the following minimum electrical 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 pull-ups 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 the board supplies power to the Raspberry Pi, the board is a Power HAT+. Power HAT+s must be able to supply at least 3A at 5.1V, but we strongly recommend 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 7. HAT+ mechanical specification

Valid HAT+s must meet the following minimal mechanical requirements:

- The HAT+ must connect to the 40-way GPIO header, including the `ID_*` pins.
- At least one mechanical mounting hole on the HAT+ must align with one of the four Raspberry Pi mounting holes.

Most HAT+s should also meet the following design recommendations:

- The HAT+ should not foul the board PoE headers found on Raspberry Pi 4 and later boards. For more information about the (distinct) locations of these headers on each board, see the mechanical drawings.
- The HAT+ should include a suitable stacking header, spacer and screws to avoid contact between HAT+ and board components (or board-mounted peripherals such as the Active Cooler). To allow space for the Active Cooler between the Raspberry Pi and the HAT+, provide *at least* 15mm board-to-board spacers; 16mm spacers are ideal. If the underside of your HAT+ has components on it, use even larger spacers. For more information about board heights, see the mechanical drawings.
- The HAT+ should not interfere with access to camera, display, and PCIe flex connectors.

7.1. Mechanical examples

The figures below show examples of official Raspberry Pi designs. You can use these examples as a starting point for your own HAT+ designs.

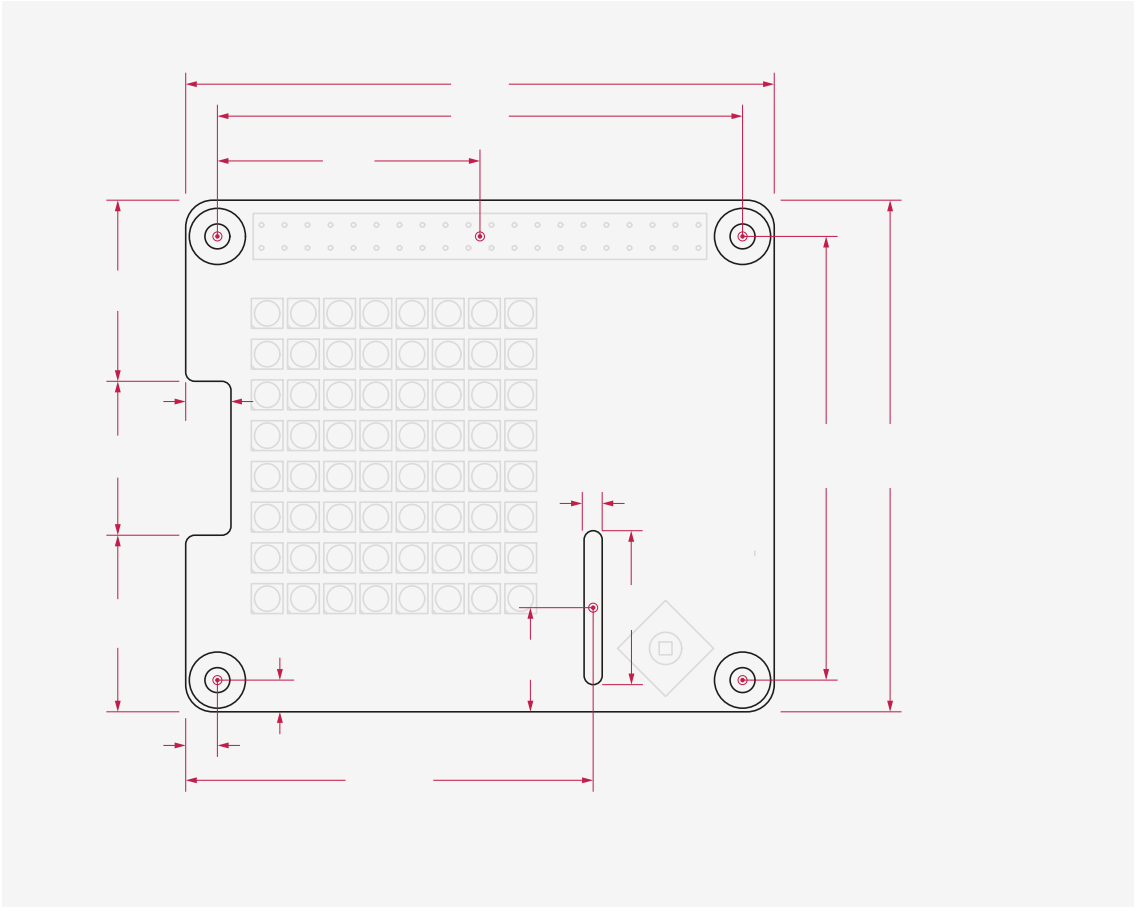


Figure 2. Mechanical diagram showing the layout of the Raspberry Pi Sense HAT.

Figure 3. Mechanical diagram showing the layout of the Raspberry Pi M.2 HAT+ M Key.

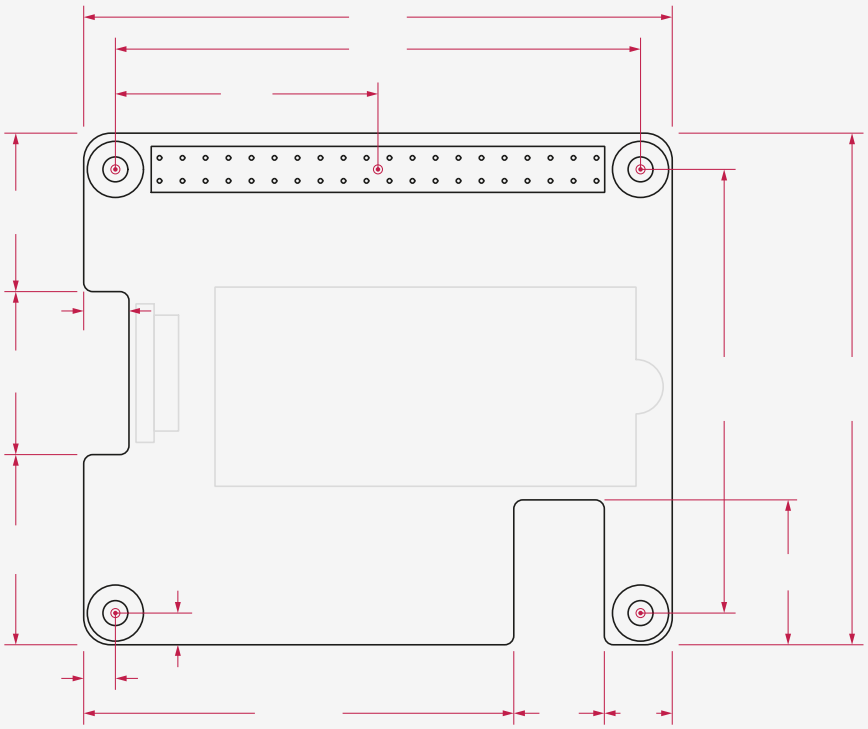
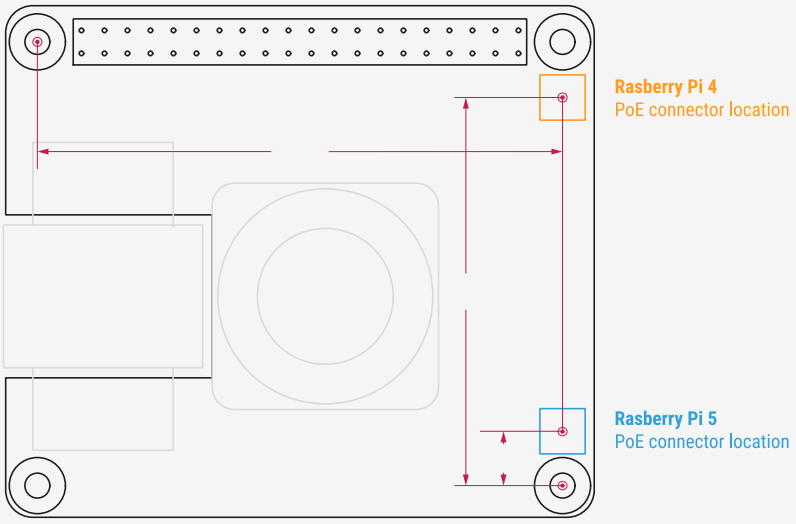


Figure 4. Mechanical diagram showing the layout of the Raspberry Pi PoE+ HAT. Indicates PoE header location for both Raspberry Pi 4 and Raspberry Pi 5.



Appendix A: HAT+ ID EEPROM specification

Software tools and documentation for creating, flashing, and reading the HAT+ ID EEPROM can be found in the Raspberry Pi utils GitHub repository. For usage information, see EEPROM tools.

Data format specification

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

We strongly recommend that `dt-overlay` names use the form `manufacturername-hatplusname`, e.g. `iqaudio-dacplus` or `rpi-sense-v2`. Names starting with `rpi-` are reserved for Raspberry Pi use. Because EEPROM strings either have an explicit length field or length can be inferred, strings don't need a zero-terminating character.

Table 2 shows the overall structure of the EEPROM:

Table 2. EEPROM structure

Block	Description
HEADER	EEPROM header (required)
ATOM1	Vendor info atom (required)
ATOM2	Device Tree overlay name atom (required)
⋮	
ATOMn	nth Device Tree overlay name atom

Table 3 shows the structure of the HEADER block:

Table 3. EEPROM HEADER structure

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 number of atoms in EEPROM
4	`eeplen`	Total length in bytes of all EEPROM data (including this header block)

Table 4 shows the structure of individual atoms:

Table 4. EEPROM ATOMx structure

Bytes	Field	Description
2	`type`	atom type (see Table 5)
2	`count`	incrementing atom count
4	`dlen`	length in bytes of `data`+`crc16`
N	`data`	`N` bytes, `N` = `dlen` - 2

Bytes	Field	Description
2	`crc16`	CRC-16 of entire atom (`type` + `count` + `dlen` + `data`)

Table 5. EEPROM ATOMx type

Atom type value	Description
`0x0000`	invalid
`0x0001`	vendor info (see Table 6)
`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 6. Vendor information atom data (type=`0x0001`)

Bytes	Field	Description
16	`uuid`	product UUID (unique)
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. "Super Special Sensor Board"

Appendix B: HAT+ EEPROM tools

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

Before you can build the tools, you'll need `cmake`. Run the following command to install it:

```
$ sudo apt install cmake
```

Next, clone the utilities repository:

```
$ git clone https://github.com/raspberrypi/utils.git
```

Navigate into the `eeptools` subdirectory of the utilities repository:

```
$ cd utils/eeptools
```

Run `cmake` to configure the utilities build:

```
$ cmake .
```

Run `make` to build the tools:

```
$ make
```

Finally, run `make install` to make the tools available on your command line:

```
$ sudo make install
```

Use the tools

1. Copy the `eeprom_settings.txt` template file to `myhat_eeprom.txt`.
2. Edit `myhat_eeprom.txt` to suit your specific HAT+. For more information, see [eeprom-spec].
3. Run the following command to create the `.eep` binary:

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

4. If `eepmake` generates a product UUID for you, patch `myhat_eeprom.txt` to include and preserve it. Alternatively, use `eepdump` to regenerate the UUID.
5. To flash the EEPROM image, disable EEPROM write protection. Sometimes this requires a jumper on the board or pulling a GPIO. Check your schematics to determine which.

6. In this specification, the HAT+ EEPROM is connected to pins that can be driven by `I2C0`. Because cameras and displays also use this interface, HAT+ use 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
```

7. Install `i2ctools`:

```
$ sudo apt install i2c-tools
```

8. Check if the HAT+ has been detected:

```
$ i2cdetect -y 9

    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:
$
```

9. Flash the `eep` file:

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

10. Finally, re-enable EEPROM write protection by putting back the jumper or resetting the GPIO, depending on your configuration.

Appendix C: Release History

10 October 2024

- Copy edits and structural improvements.
- Removed draft watermark.

20 December 2023

- Added information about EEPROM tools.

08 December 2023

- Initial release.