

HAT++ Specification

Technical extension to Raspberry Pi HAT+

Specification	Version 1.0	Document	Revision A
Date	30 September 2026	Status	Technical review draft
Author	Philipp Brechel	Document ID	BE-IIS-HATPP-SPEC

5 global positions	3 variants per board	1 active identity
I, II, III, IV, V. Logical resource slots, not stack height.	Exactly three supported positions and three overlay variants.	One selected EEPROM target and one matching image at a time.

Node I is mandatory on every HAT++ board. It is the HAT+-compatible identification position. An automatically configured stack has exactly one board in Node I; its boot overlay prepares the shared host resources. Additional boards are discovered and configured by host software.

This document specifies externally observable behaviour, image generation and host integration. It does not publish MCU firmware, flashing procedures or manufacturing internals. The hardware qualification items recorded with this draft are not claims of tested conformity.

Contents

1	Scope and relationship to HAT+	2
2	Five positions; three variants per board	3
3	MCU identification interface and selection	4
4	Address planning and shared resources	5
5	EEPROM content generation	6
6	Overlay contract	7
7	Boot sequence and systemd reference service	8
8	Installation and read-only diagnostics	9
9	Conformity checklist and evidence	10
10	Licence, attribution and names	11
11	References and revision history	12

Copyright © 2026 Philipp Brechel. Original specification material: CC BY 4.0. BE-IIS branding and third-party rights are not licensed by that grant. See Section 10.

1 Scope and relationship to HAT+

An extension, not a replacement

HAT++ adds selectable identities and coordinated resource use. It does not replace the Raspberry Pi HAT+ mechanical, electrical or EEPROM specification. Node I is the compatibility position; Nodes II–V use HAT++ host integration.

1.1 Normative baseline

S1 A conforming HAT++ board **MUST** support HAT++-compatible operation in Node I and **MUST** satisfy the applicable HAT+ requirements. This draft refers to the *Raspberry Pi HAT+ Specification*, build date 2024-12-05, build version `6df06cc-clean`. Later upstream revisions require an explicit review; they do not silently change this version. [R1]

Subject	Authority / HAT++ addition
Mechanics and connector	HAT+, Chapter 7. HAT++ introduces no different connector, mounting pattern or board outline. Stack clearance still needs checking.
Power and electrical limits	HAT+, Chapters 2 and 6; Chapter 4 where applicable. A Node ID is not a power budget or permission to parallel supplies.
Identification and image format	HAT+, Chapter 3 and Appendix A. HAT++ selects one of three corresponding images.
Multi-node configuration	This document: selection, resource ownership, secondary discovery and overlay application.

S2 Nodes II–V **MUST NOT** be advertised as additional Raspberry Pi Standard, Stackable or Power HAT+ classes. HAT+ already defines its own stacking classes. HAT++ multi-node operation is a separate extension, not native firmware discovery of five Standard HAT+s. [R1]

1.2 Requirement language

MUST / **MUST NOT**: mandatory for a conformity claim. **SHOULD** / **SHOULD NOT**: recommended; deviations need a documented reason. **MAY**: optional. Requirement identifiers are stable review references. Text labelled *reference profile*, *example* or *review item* is informative unless a requirement explicitly adopts it.

1.3 Terminology

Term	Meaning
Global Node ID	One of five system positions I–V; independent of physical stacking order.
Local variant	One of the three configurations implemented by a particular board. Legacy overlay suffixes I/II/III normally label these variants.
Identification target	The selected EEPROM-emulating I ² C target on ID_SD / ID_SC. Not a functional peripheral address.
Host profile	A matching address map, primary resource configuration, discovery logic and overlays.

2 Five positions; three variants per board

- N1** The global position set is {I, II, III, IV, V}. Each board **MUST** implement exactly three positions, one of which **MUST** be I.
- N2** Each supported position **MUST** select its matching hardware routing, identification target and EEPROM image. There are exactly three logical overlay variants per board, not five. Alternative builds for different host platforms do not create additional Node IDs.

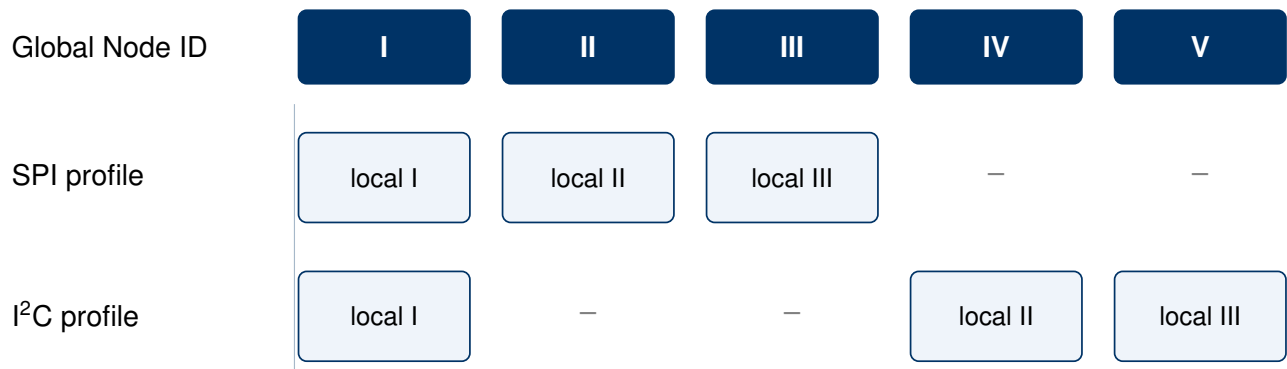


Figure 1: Typical SPI and I²C profiles. Each row contains three supported positions.

Board profile	Local I	Local II	Local III
SPI	Global I	Global II	Global III
I²C	Global I	Global IV	Global V

Do not derive a global Node ID from an overlay suffix. For example, `BE-IIS-HPP-UART-II` denotes the second UART variant, used at global Node IV in the current host profile. The host discovers it at the profile’s Node-IV EEPROM path. [\[R3\]](#)

2.1 Stack rules

- N3** Automatic operation **MUST** have exactly one active Node-I board. Every other active board **MUST** use a different supported global position. The maximum is five active boards; five identical boards are not implied. For example, an I/II/III-only product can occupy at most three positions.
- N4** Node ID **MUST** describe resource ownership, not a PCB’s vertical position in the stack. Reordering PCBs does not itself change their IDs.
- A disabled, bypass or manual mode **MAY** exist. Such a mode is outside this three-variant automatic profile and **MUST NOT** be counted as a sixth global Node ID or a fourth HAT++ EEPROM/overlay variant.

One selection, three coupled results

Selected Node ID → EEPROM address + image + hardware resources. An overlay matching only the product name, but not its selected routing, is not sufficient.

3 MCU identification interface and selection

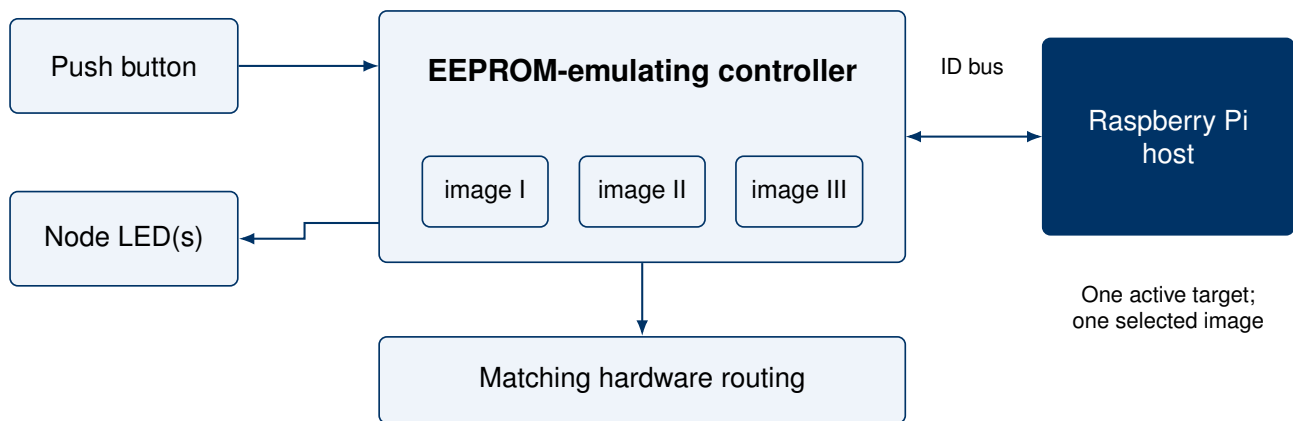


Figure 2: Reference architecture. Internal MCU design and storage layout are not specified.

E1 Implement the controller so that its host-facing interface behaves as an identification EEPROM. One selected target returns the corresponding EEPROM file byte-for-byte; the other two identification targets **MUST NOT** respond. Three supported targets do not mean three simultaneous responders.

Observable property	Requirement
Target selection	E2 Node I uses 7-bit address <code>0x50</code> . The other selected target addresses follow the declared host profile; all three addresses on one board are distinct.
EEPROM reads	E3 Support the HAT+ EEPROM read interface, including 16-bit memory offsets, random reads and sequential reads. The offset is not the I ² C target address.
Memory contents	E4 Serve a valid HAT+ image from offset zero. Reads within its declared length return exactly the generated bytes. The reference <code>24c32</code> profile exposes 4096 bytes; unused bytes SHOULD read as <code>0xFF</code> .
Write protection	E5 Host reads and address-pointer writes MUST NOT modify the stored image. Manufacturing programming and internal storage are out of scope.
Timing and electrical behaviour	E6 Meet the applicable HAT+ EEPROM requirements, including its prohibition of clock stretching. Functional autodetection alone does not demonstrate this. [R1]

3.1 Button, indication and persistence

E7 A local push button **MUST** select among the board's three supported positions. LED indication **MUST** unambiguously identify the selected position; a labelled local I/II/III display needs a documented global mapping. Button timings, LED colours and MCU choice are implementation-defined.

E8 The selection **MUST** survive removal of power and be restored before host identification begins. Address, image and routing **MUST** agree at that time. A secondary board **MUST NOT** transiently respond as Node I during boot. The manufacturer **MUST** state the required selection-save interval.

E9 Configure the selection before the operating boot. After changing it, wait for storage to complete and power-cycle the system. Live reconfiguration, hot-plugging and automatic removal of old overlays are not defined here.

4 Address planning and shared resources

4.1 Two different I²C address spaces

A1 All addresses in this specification are **7-bit target addresses**. On ID_SD/ID_SC, they select an identification EEPROM. On a functional bus, they select a UART, ADC or other peripheral. These roles **MUST** be documented separately; neither is a 16-bit EEPROM memory offset.

A2 Apart from Node I at 0x50, HAT++ does not assign a universal secondary EEPROM address block or universal functional-device addresses. The implementer **MUST** choose valid, distinct addresses for the three board variants and **MUST** prevent collisions on each shared physical bus. Existing HAT++ assignments **SHOULD** be considered before allocating addresses. HAT+ class addresses 0x51–0x53 **MUST NOT** be repurposed. [R1]

Reference pro- file	I	II	III	IV	V
EEPROM target	0x50	0x60	0x70	0x74	0x76
SPI chip select	GPIO8	GPIO7	GPIO16	–	–
IRQ (published map)	GPIO6	GPIO5	GPIO12	GPIO14*	GPIO25

BCM GPIO numbers, not physical header-pin numbers. The EEPROM row is the current BE-IIS loader profile, not an exclusive allocation. [R2][R3]

* Node-IV routing: unresolved source difference

The website and UART datasheet give GPIO14. The current UART-II and MODBUS-II overlays configure GPIO17. This draft does not silently select one as a verified universal pin assignment. Confirm the actual hardware revision and reconcile the overlay before using Node IV or releasing a final pin profile. See the companion `REVIEW.md`. [R2][R6]

A3 A different secondary EEPROM map **MUST** be implemented consistently in the board, the primary overlay's EEPROM declarations and the loader. The current BE-IIS loader does not discover arbitrary alternative addresses. Distinct EEPROM addresses alone do not prevent peripheral or GPIO conflicts.

4.2 Shared lines and ownership

The published resource map shares SPI0 GPIO11/9/10 (SCLK/MISO/MOSI), functional I²C GPIO3/2 (SCL/SDA), identification GPIO1/0 (ID_SC/ID_SD), and GPIO13 as reset/enable. In the current T1S-I overlay, GPIO13 is driven high by a host GPIO hog. [R2][R5]

A4 Unselected SPI devices **MUST** release shared MISO. A board **MUST NOT** drive another slot's exclusive line. Shared reset/enable has one coordinated owner; it is not an independent per-board reset interface. Any additional GPIO use and interrupt polarity **MUST** be declared and checked for collisions.

A5 The ID bus is for identification only. The Node-I configuration **MUST** provide the HAT+ identification pull-ups; secondary configurations **MUST NOT** add competing pull-up ownership. Check aggregate capacitance, voltage levels, standby behaviour, power budget and mechanical clearance for the complete stack. Node allocation does not prove electrical compatibility.

5 EEPROM content generation

D1 Each board **MUST** provide three valid HAT+ format-v2 images and three matching overlay names. The `dt_blob` entry names an overlay in the host filesystem; it is not the compiled overlay binary. Use a manufacturer-specific name without the `.dtbo` suffix. [R1][R7]

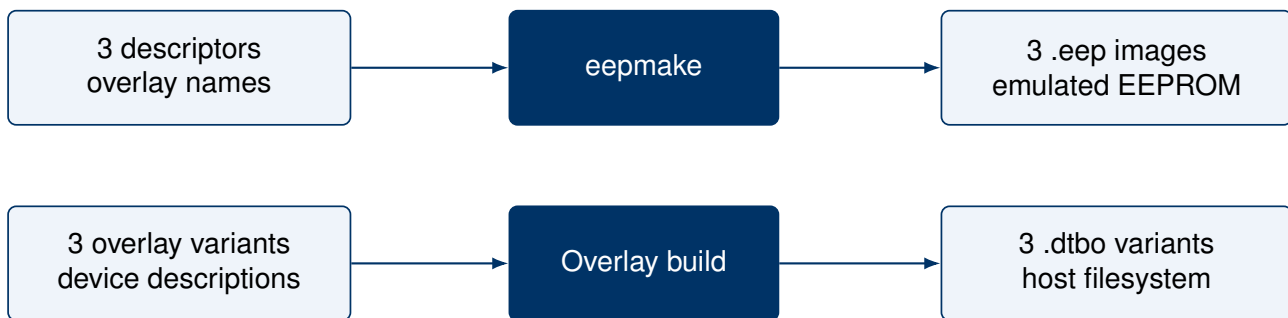


Figure 3: Three identification images and three overlay variants are separate outputs.

5.1 Descriptor fields

The descriptor below is illustrative, not an assigned product identity. Use the manufacturer's own identity data. Keep a generated UUID stable; using the same physical-board identity across its three variants is recommended. A zero UUID asks the upstream tool to generate one. [R7]

```

product_uuid 00000000-0000-0000-0000-000000000000
product_id 0x0001
product_ver 0x0001
vendor "Example Devices"
product "Example HAT - local variant I"
dt_blob "example-hat-I"
current_supply 0
  
```

Important: `current_supply` describes current supplied by the HAT+, not its consumption. Do not copy a consumption figure into this field. The example declares no supply capability. [R7]

5.2 Build and inspect, without flashing

Use the current `eepmake` and `eepdump` from Raspberry Pi `utils/eeptools`. The BE-IIS installer installs these tools too. Current `eepmake` defaults to format v2. [R7][R8]

```

# Prepare one descriptor for each LOCAL variant, then:
for v in I II III; do
    eepmake "example-hat-${v}.txt" "example-hat-${v}.eep"
    eepdump "example-hat-${v}.eep" > "example-hat-${v}.dump.txt"
done
  
```

D2 Inspect all three results: valid image, correct identity, exact overlay name and length within the exposed EEPROM capacity. Supply the matching bytes to the EEPROM emulator; how the MCU stores them is not specified.

The optional companion `tools/make_eeprom.py` prepares three descriptors with one saved UUID and can invoke the public tools. See `README.md` for its commands. It does not flash or communicate with hardware.

6 Overlay contract

The primary overlay prepares the stack before drivers bind

A runtime overlay is not a reliable way to undo a conflicting device that was already instantiated during boot. The primary configuration must make the shared resources available before the secondary overlays are applied.

Component	Required responsibility
Primary overlay (Node I)	01 Describe the primary board and prepare the declared stack profile: enable required buses, reserve shared resources, establish chip selects, suppress conflicting default devices and declare discoverable EEPROM targets.
Secondary overlays	02 Describe only the selected board's function and its declared resources. Match the physical chip select or peripheral address, interrupt GPIO and polarity, reset handling, clock and driver binding.
All three variants	03 Use distinct overlay identities matching the three EEPROM images. A local II/III suffix does not change the global-position mapping defined by the board profile.
Host filesystem	04 Provide the matching <code>.dtbo</code> files before the boot that uses them. In the Raspberry Pi OS reference deployment they are installed under <code>/boot/firmware/overlays</code> , with a legacy-path fallback. [R8]
Drivers	05 Make required drivers available for the actual kernel and platform. Applying an overlay is not proof of successful driver probing or device operation.

6.1 Reference primary configuration

The current T1S-I overlay configures SPI0 with three chip selects on GPIO8, GPIO7 and GPIO16; it disables the conflicting `spidev` nodes, declares five `atmel,24c32` targets on the identification bus and establishes the shared GPIO13 reset/enable state. These are externally visible duties, not a requirement to copy that DTS implementation. [R5]

06 Every board advertised as a primary for a given stack profile **MUST** prepare that complete profile, even when its own functional device uses only I²C or only one SPI chip select. Alternatively, explicitly state a narrower supported stack profile. Do not infer universal primary capability merely from an overlay filename ending in I.

6.2 Changing an address allocation

For an alternative secondary EEPROM address, update three things together: the board's target selection, the primary host description and the loader's path-to-global-ID mapping. For a functional peripheral address, update the board's address selection and all corresponding device-tree references. The EEPROM image's ordinary HAT+ format remains unchanged.

07 The host **MUST NOT** apply the primary overlay a second time as a secondary overlay. Unrelated symbols, GPIO owners and existing device nodes **MUST NOT** be redefined by secondary variants.

Out of scope: live overlay removal, transparent driver rebinding, automatic conflict resolution and a guarantee that arbitrary Raspberry Pi models or kernel versions accept the same overlay binary.

7 Boot sequence and systemd reference service

This section describes the public installer snapshot `b622dc40`. It documents current behaviour; it does not add capabilities to the service. [R3][R4]

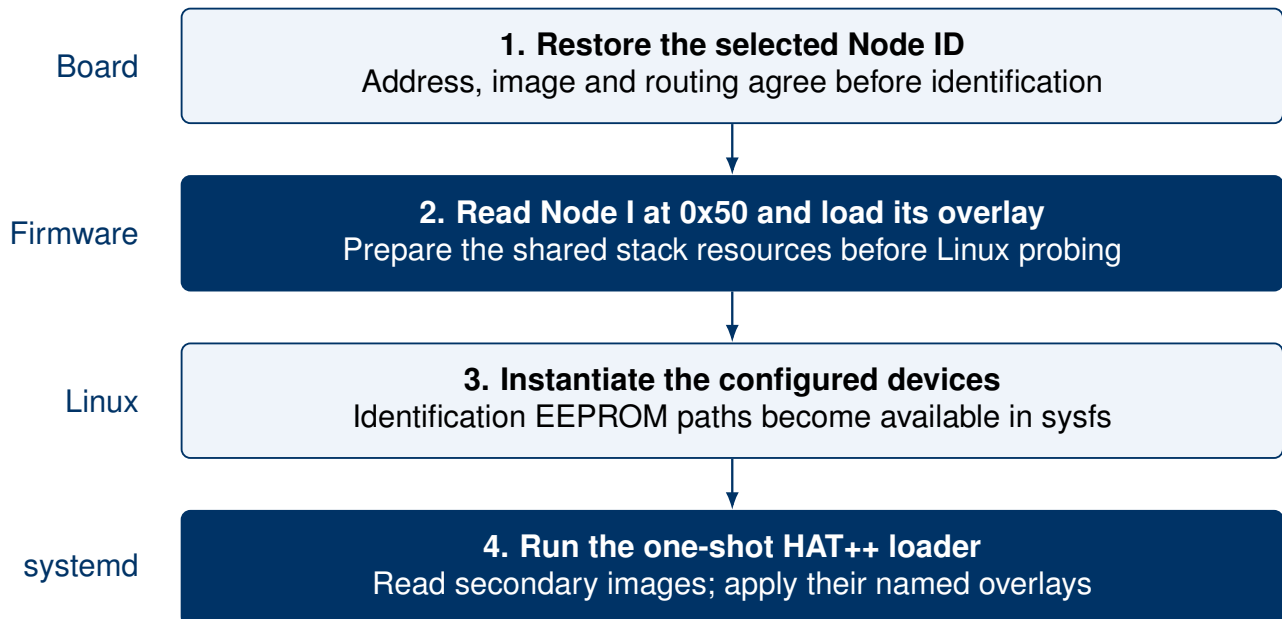


Figure 4: Firmware handles Node I. The one-shot Linux service handles secondary overlays.

Unit property	Current value / consequence
Unit	<code>be-iis-hatpp.service</code>
Execution	<code>Type=oneshot; /usr/local/bin/apply-hat-overlays.sh</code>
Ordering	<code>After=local-fs.target; enabled for multi-user.target</code> . This is not an explicit EEPROM-ready or network-ready dependency.
Timeout / logging	<code>TimeoutStartSec=30</code> ; stdout and stderr go to the journal.
Completion state	<code>No RemainAfterExit=yes</code> . A successful run can subsequently show <code>inactive (dead)</code> ; inspect its exit result and journal.

7.1 What the loader actually does

It checks the five explicit sysfs EEPROM paths in I–V order. For an existing path it runs `eeppdump` and extracts the quoted `dt_blob` name. Node I is logged only; its overlay is left to firmware. Each other non-empty name is passed to `dtoverlay`. It then attempts a fixed set of available BE-IIS-related kernel modules and optional serial-udev integration. [R3]

7.2 Boundaries and failure behaviour

The service is not a bus-wide address scan, a hot-plug monitor, a pin allocator or a retry loop. Missing paths are skipped. With `set -e`, an unsuccessful `dtoverlay` stops the script; already-applied overlays are not rolled back. Module availability/loading and serial integration are partly best-effort. The decoder pipeline does not provide a separate robust validation stage. A completion message therefore does not establish that every device works.

H1 An application that needs the complete stack **MUST** check its required devices, not merely wait for `multi-user.target`. Explicit ordering after the service may be useful, but device readiness and probe success remain separate conditions. Re-running the loader can attempt duplicate overlays; a clean reboot is the normal recovery path after a configuration change.

8 Installation and read-only diagnostics

8.1 Reference installation on Raspberry Pi OS

Use the public BE-IIS installer. Review its current README and scripts before running them with privileges. The installer installs tools, modules, overlays, the service and udev integration, and enables the identification bus via `dtparam=i2c_vc=on`. Check platform and kernel support for the selected boards. The script prompts before rebooting. [R8]

```
sudo apt update
sudo apt install -y git
cd "$HOME"
git clone https://github.com/be-iis/be-iis-installer.git
cd be-iis-installer
bash scripts/install/install-all.sh
```

An installed copy must include third-party boards' own overlays and required drivers; the BE-IIS installer does not automatically supply arbitrary future products. Select unique Node IDs, wait for their save interval and power-cycle before evaluating automatic discovery.

8.2 Service and overlay state

```
systemctl is-enabled be-iis-hatpp.service
systemctl show be-iis-hatpp.service \
  -p ActiveState -p SubState -p Result -p ExecMainStatus
journalctl -b -u be-iis-hatpp.service --no-pager
sudo dtoverlay -l
```

The runtime list is not evidence that firmware failed to load Node I: firmware boot overlays and runtime-applied overlays are not the same list. `Result=success` and `ExecMainStatus=0` describe the service run, not functional tests of the attached peripherals.

8.3 Inspect the configured identification targets

```
# Reference profile: bus 0, not the functional I2C bus.
for id in 0050 0060 0070 0074 0076; do
  p="/sys/bus/i2c/devices/0-${id}/eeprom"
  [ -e "$p" ] || continue
  printf '\n%s\n' "$p"
  sudo eepdump "$p"
done
```

A sysfs node comes from the device description; readable, valid contents are the meaningful test. These commands do not write EEPROM data. Avoid blind I²C probing and manual target creation during normal operation.

Symptom	First checks
No primary / no secondary paths	Node I selected exactly once; saved state; ID-bus pull-ups; valid primary image; installed primary overlay; identification bus enabled.
Secondary not configured	Supported global position; selected target matches host profile; readable image; correct overlay name and installed binary.
Overlay fails or device absent	Journal; conflicting GPIO/device ownership; Node-IV routing review; matching driver and kernel; peripheral address and interrupt wiring.

9 Conformity checklist and evidence

A *HAT++ compatible* claim concerns a named board revision, supported three-position set and declared host profile. It is not a BE-IIS certification. This document does not assert that any existing board has passed these checks.

Check	Requirements	Evidence to retain
C1	S1–S2	Applicable HAT+ mechanical/electrical/identification assessment in Node I, including standby and EEPROM timing. Do not substitute a successful boot for this assessment.
C2	N1–N4	Exactly three supported positions, including I; local/global mapping; three corresponding overlay variants; documented maximum stack combinations.
C3	E1–E6	For every selected position, the correct address responds and the other two do not. Random/sequential reads match the generated image. Verify capacity, image protection and timing on the bus.
C4	E7–E9	Button/LED mapping; saved state after power removal; specified save interval; no transient Node-I response from a secondary during startup.
C5	A1–A5	Separate identification and functional address maps; no collisions; complete GPIO ownership; pull-ups, power and signal integrity for the intended stack.
C6	D1–D2	Three generated v2 images; valid decoder output; correct vendor/product identity, overlay names and capacity. No inappropriate current-supply declaration.
C7	O1–O7	Each supported primary prepares its declared profile. Every secondary overlay matches the actual board routing and does not seize another board's resources.
C8	H1	Cold boot with the intended stack; correct primary/secondary sequence; log and exit results; required devices successfully probe and pass application-level tests.

9.1 Minimum published board information

A board claiming compatibility **MUST** identify the specification version, hardware revision, three supported global IDs, local/LED mapping, all active I²C addresses by bus, chip-select and interrupt routing, three overlay names, selection-save interval and supported host profile. Electrical ratings, driver requirements and any additional resource use **MUST** be available in its own documentation.

9.2 Release gates for this draft

The GPIO14/GPIO17 discrepancy for Node IV needs a verified board-level resolution before a final common pin profile is declared. MCU-based identification also needs assessment against HAT+'s literal EEPROM/device requirements; host read compatibility is not a waiver of upstream requirements. The author-facing review record is kept separately from the implementation-independent specification.

Document checks are not hardware qualification

The supplied build tooling checks document compilation and generated-file consistency. It does not certify I²C timing, electrical safety, Raspberry Pi compatibility or successful operation of an attached stack.

10 Licence, attribution and names

10.1 Original HAT++ specification material

Copyright © 2026 **Philipp Brechel**. The original HAT++ specification text and original diagrams in this document are licensed under **Creative Commons Attribution 4.0 International (CC BY 4.0)**. Commercial redistribution and adaptation are permitted under that licence. Its unmodified legal terms govern the grant. [R9]

Retain the supplied author identification, copyright and licence notices and source reference as required by CC BY 4.0; identify modifications when sharing adapted material. Attribution may be provided in a reasonable manner. The following is a suggested notice, not an additional mandatory format:

Suggested attribution

HAT++ Specification, originally developed by Philipp Brechel, Brechel Electronic – Industrial Interface Systems (BE-IIS).

Hindenburgstraße 100/1, 73207 Plochingen, Germany.

<https://www.be-iis.eu/> CC BY 4.0.

For an adaptation: identify the original version and describe the changes.

10.2 Implementation and compatibility designation

Independent commercial and proprietary implementations are welcome. This document does not require disclosure of firmware, source code or schematics. CC BY licenses the covered expressive material; it is not a patent licence, a trademark licence or an exclusive right over the technical ideas. Implementers remain responsible for any applicable third-party rights.

Conforming implementations are encouraged to use *HAT++ compatible*, identify the implemented version and acknowledge the specification's origin. A compatibility claim must meet the mandatory technical requirements and state the applicable profile. Merely implementing the described ideas is not represented here as a copyright-based obligation to use that name. No certification, approval or endorsement by BE-IIS is implied.

10.3 BE-IIS remains separate

The document grant **does not license the BE-IIS name, logo, product branding or corporate identity**, except factual identification and attribution of the source. It does not relicense existing BE-IIS products, firmware, schematics, tools, shared document styles or other repository contents. Do not present a third-party product as made, approved or certified by BE-IIS. New companion build/helper code is separately licensed as stated in `LICENSE.md`; that scope does not extend to linked repositories.

10.4 Raspberry Pi and upstream material

Raspberry Pi is a trademark of Raspberry Pi Ltd. HAT+ is referenced solely to identify the upstream specification and compatibility target. This is an independent document, not a Raspberry Pi publication or endorsement. No Raspberry Pi/HAT+ logo or modified wordmark is included. [R10]

The HAT+ specification remains under its own CC BY-ND 4.0 licence and other notices. It is referenced, not reproduced as an edited specification and not relicensed here. No assurance of trademark clearance for the HAT++ name or permission to use third-party branding is made. [R1]

Warranty and liability: the CC BY 4.0 disclaimer and limitation of liability apply to the licensed material, to the extent permitted by law.

11 References and revision history

References were reviewed on 30 September 2026. Software descriptions are snapshot-specific. Links to the installer baseline use commit `b622dc40`; the companion source manifest records full identifiers.

[R1] Raspberry Pi HAT+ Specification (normative upstream)

Build 2024-12-05, `6df06cc-clean`; Chapters 2, 3, 6, 7 and Appendices A/B. [Official PDF](#).

[R2] BE-IIS HAT++ architecture and resource description

[HAT++ overview](#); [technical background](#). Source differences are identified in Section 4, not resolved by assumption.

[R3] BE-IIS reference discovery/overlay loader

[apply-hat-overlays.sh](#). Fixed EEPROM paths, Node-I skip, secondary overlay application and module/udev steps.

[R4] BE-IIS systemd unit

[be-iis-hatpp.service](#).

[R5] Reference primary overlay

T1S-I DTS: SPI chip selects, default-device suppression, EEPROM declarations and GPIO13.

[R6] Node-IV software routing evidence

[UART-II DTS](#); [MODBUS-II DTS](#). Both configure GPIO17 in the reviewed snapshot.

[R7] Raspberry Pi EEPROM tools

[utils/eeptools](#), snapshot `ebc4a56b`. [Usage](#); [descriptor fields](#).

[R8] BE-IIS installer

[Repository and README](#); [reviewed installation scripts](#). Install current releases deliberately; commands may change after this snapshot.

[R9] Creative Commons Attribution 4.0 International

<https://creativecommons.org/licenses/by/4.0/legalcode.en>.

[R10] Raspberry Pi trademark rules

<https://www.raspberrypi.com/trademark-rules/>.

11.1 Revision history

Spec.	Rev.	Date	Change
1.0 (draft)	A	2026-09-30	Initial technical draft: five global positions / three local variants, EEPROM interface, host profile, image generation, systemd integration and scoped licensing. Source discrepancies retained as review gates.

Version changes track technical requirements; document revisions track editorial corrections and evidence updates. A normative change requires a recorded version decision, not just replacement of the PDF.

Publisher/contact: Philipp Brechel, Brechel Electronic – Industrial Interface Systems, Hindenburgstraße 100/1, 73207 Plochingen, Germany. contact@be-iis.eu <https://www.be-iis.eu/>