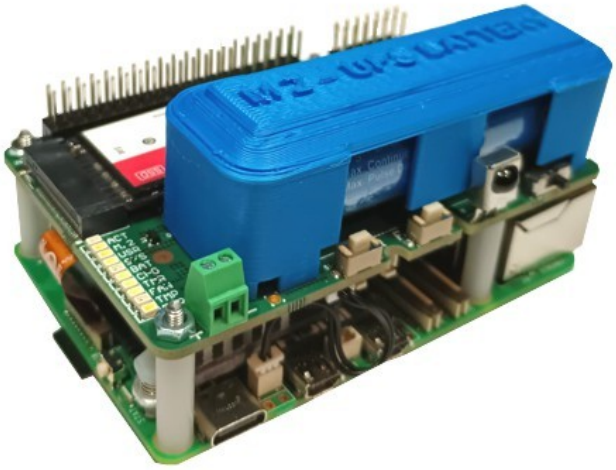
Pi Modules Technologies

M.2 UPS PMH

Standard and Advanced models

M.2 UPS and Power Management HAT

User Guide



Advanced model with 18650 battery and protective cap

Especially designed for Raspberry Pi 5

M2_UPS_PMH_Manual_V8F | 22 October 2026
Application V88 (0x0088) | Bootloader V88 at 38400 bit/s

Hardware installation and operating instructions

www.pimodules.com | support@pimodules.com

Raspberry Pi is a trademark of Raspberry Pi Technologies L.P. Product illustrations show optional equipment.

About this manual

This guide explains installation, configuration and operation of the **M.2 UPS PMH**. Read the power and battery instructions before connecting the board. Follow the hardware setup, Linux installation and commissioning chapters in that order.

Document revisions

Revision	Date	Changes
V06	20 Oct 2025	Initial consolidated installation guide and register reference.
V07	21 Oct 2025	Advanced assembly photographs and supported backup-source codes.
V08	21 Oct 2026	V88 serial updates, CRC verification and interrupted-transfer recovery.
V09	22 Oct 2026	Product naming, I²C, IR interface, enclosure and Read Only labels.
V12	22 Oct 2026	Automatic Linux installer V01: setup, checks, backups and removal.
V13	22 Oct 2026	Raspberry Pi RTC backup from the M.2 main battery; no separate RTC cell.
V14	22 Oct 2026	Release records, accessories, assembly, backup-source and supply guides.
V17	22 Oct 2026	Optional security chapter removed; existing operating instructions retained.
V8E / V8F	22 Oct 2026	Tool and migration guides; V8F unifies fonts and adds 4000 F energy estimates.

Software supplied with this manual

This edition applies to the V88 application (firmware identifier 0x0088) and the matching V88 protected bootloader at 38400 bit/s. Both Standard and Advanced models are covered. Installing the V88 application alone does not replace that loader.

Component	Status in this edition
Application / bootloader	V88 (0x0088), combined ICSP image plus matching protected bootloader at 38400 bit/s.
Daemon	v1.1: 250 ms heartbeat, safe-shutdown request handling, CPU temperature and NTP-gated hourly RTC synchronization.
Status monitor	v2.1, package 06: V88 map; Linux RTC driver/sysfs; read-only inspection. Configurator 01: confirmed edits and readback (see added guide).
Linux installer	V01: fresh-card installation and post-reboot checks passed on the reported Pi 5 / Debian 13 setup; shutdown and cable-return boot tested.

Package contents

This package contains the manual, automatic Linux installer V01, shutdown daemon, read-only monitor, configuration tool, register reference and Python firmware updater, together with compiled V88 firmware and application/bootloader sources. The installer configures Linux; it does not program MCU firmware.

Hardware identification

Identify Standard or Advanced from the assembly and startup banner. Use the matching connectors, battery profile and power-input ratings throughout installation.

Contents

Unlocked firmware features	4
Firmware bug fixes and additions	5
Daemon and host software changes	6
Schematics and PCB updates	7
System overview and introduction	8
Standard and Advanced models	9
Core features	10
Embedded peripherals and interfaces	11
System protection and PCB construction	12
Add on equipment and package contents	13
Application specific add on extensions	14
Advanced assembly configurations	15
Battery and supercapacitor installation	17
Power supply unit recommendations	20
Assembly of the Magic ON/OFF Switch	21
Assembly of the THT 40 Pins Connector	22
Assembly of the Gold-Plated POGO Pin	23
Installation step by step	24
Automatic Linux installer	35
UPS operation and File Safe Shutdown	41
Intelligent mobile power bank and slide switch	42
System LEDs and user controls	43
RTC operation and time synchronization	44
Cooling and enclosure integration	45
Application integration	46
Programming interface and I²C address groups	47
Module status registers and monitoring	48
Hardware RTC register interface	49
Module command registers	50
Bootloader entry and recovery	51
Local and remote firmware updates	53
Troubleshooting	55
Commissioning checks	56
Frequently asked questions	57
Technical specifications	62
A Complete Description of the **M.2 UPS PMH** Programming Registers	69
Tools and SD to SSD migration	78
Supercapacitor cycle life and temperature	86

Unlocked firmware features

This feature matrix describes the supplied **M.2 UPS PMH** application and protected bootloader V88. It separates MCU functions from Linux software and optional hardware. Presence of a register in the reference table does not mean that every possible application function is enabled or configured by the supplied daemon.

Feature	Availability in this package
UPS and charging	Application V88: cable/battery recognition, charging control, timed FSSD and low-battery protection, with the matching backup-source profile.
Shutdown and restart	Application V88 with the Linux daemon: controlled shutdown, LPR standby and power-return startup. Pi-button control wiring is required.
Optional ON/OFF switch	V88 supports the fitted switch in slide-switch mode. Normal mode and switch mode have different runtime behavior.
M.2 hardware RTC	Read/write access through the DS1307-compatible bank at 0x68. Daemon v1.1 performs NTP-confirmed UTC synchronization.
Raspberry Pi RTC backup	Hardware function through the dedicated supply cable and the main M.2 battery; separate from the M.2 RTC driver.
Firmware updates	V88 protected UART bootloader, 38400 bit/s. Button or software entry, Flash CRC32 verification and interrupted-write recovery.
Cold recovery startup	V88 protected loader starts Raspberry Pi once during the specified cold/brownout recovery of an incomplete application write.
Monitoring	Optional monitor v2.0 reads the V88 status/command layout. A monitoring read counts as host activity.
Linux setup	Installer V01 installs/configures the existing daemon and tools. Fifteen simulated tests passed; clean-card hardware validation is pending.
Application interfaces	Advanced-model interfaces require matching fitted hardware and application software. The daemon does not implement email delivery, IR decoding or a user scheduler.

Identify the running application

```
sudo i2cget -y 1 0x69 0x38 b
sudo i2cget -y 1 0x69 0x39 b
```

At the default address, V88 returns low byte 0x88 and high byte 0x00. Keep the installed-loader record separately: the application version and PCB/BL identity byte do not identify which protected loader is installed. Hardware and firmware must match the procedures in this manual.

Firmware bug fixes and additions

The following are documented development milestones relevant to this package. They are not a complete list of every intermediate build and are not instructions to install an older image. Application and protected-loader changes must be tracked separately.

Milestone	Documented change	Relevance to V88
V4C	RTC hardware writes enabled with the revised application stack arrangement.	Successful RTC write/read and continuing heartbeat were observed in the reported tests.
V53	Repeated normal power cycles and optional switch-mode behavior were tested.	Established the shutdown and switch workflow used for later validation.
V58	Hardware FSSD/factory-recovery work; normal UPS operation provided the later power-control reference.	Historical factory-button behavior is superseded; do not follow that earlier entry sequence.
V73 baseline	Repeated FSSD/LPR/power-return cycles and cable return during shutdown were validated in the power-control development sequence.	V88 retains the validated power behavior while using the later bootloader entry scheme.
V80	RESET+FSSD selects direct bootloader entry; keyboard-selected factory defaults removed.	No HCM selection menu and no KEY jumper are required.
V84	Protected bootloader moved to 38400 bit/s.	Specify -b 38400 in the Python updater; changing Python alone does not change the loader UART.
V87	Bootloader walking-LED progression reduced to one step per two HEX records.	The indication does not change transfer baud or CRC verification.
V88	Cold/brownout recovery of an incomplete write starts Raspberry Pi once.	Allows Linux/SSH to restart so the complete application can be reloaded.

Current update acceptance record

Reported bench tests confirmed complete serial transfers at 38400 bit/s, Flash readback/CRC32 verification, and recovery after interrupted uploads, MCU reset and complete power loss. Successful completion is established by the updater's final CRC-verified message, not by LED activity alone.

The retained V88 build reports application compilation with 0 errors and 35 warnings, and protected-loader compilation with 0 errors and 0 warnings. This is a build record; runtime acceptance still requires the commissioning checks for the actual assembled system.

Daemon and host software changes

The Linux tools have independent version numbers. Updating a manual does not automatically change the daemon, monitor, updater or installed MCU image. Keep the extracted package together so that scripts, service files and instructions remain matched.

Component	Version and purpose	Change or validation record
Shutdown daemon	M2_PMH_i2c.py v1.1	250 ms heartbeat, shutdown handshake, CPU-temperature reporting and NTP-gated hourly UTC synchronization. Runs as M2_PMH_i2c.service.
Status monitor	Read-only monitor v2.0	Status/command layout checked against V88. Optional; does not provide shutdown protection.
Firmware updater	M2_PM_UPS_BL01D.py	Supports the supplied loader workflow and application Flash readback/CRC32 verification. Explicit -b 38400 is required; its default remains 9600.
Linux installer	install_M2_UPS_PMH.py V01	Dependencies, boot configuration, daemon/service, monitor/updater copies, checks, backups and restoration. Fifteen simulated tests passed; first clean-card hardware trial remains pending.

Service changes and existing overrides

Daemon v1.1 does not require a GPIO interrupt connection. It polls the command bank over I²C and discovers the M.2 RTC by kernel name and bus/address. Automatic RTC writes wait for confirmed NTP; loss of network synchronization does not stop the heartbeat.

Review service drop-ins before changing an existing installation. An override containing --no-rtc-sync keeps synchronization disabled. Installer V01 stops for custom units or drop-ins instead of silently deleting site-specific settings. During maintenance, keep cable power present and avoid duplicate daemons.

```
sudo systemctl cat M2_PMH_i2c.service
sudo journalctl -u M2_PMH_i2c.service -b -n 15 --no-pager
```

What these tools do not configure

The supplied daemon does not implement email notifications, scheduled application wake events or Raspberry Pi fan cooling profiles. The installer does not flash MCU firmware, modify Raspberry Pi EEPROM or select battery chemistry/current. The monitor is not a substitute for the shutdown service.

Schematics and PCB updates

Identify the actual Standard or Advanced assembly and its PCB markings before fitting accessories. Firmware version, assembly model, PCB revision and protected-loader revision are separate records. The startup PCB/BL value 0x40 is an identification byte and should not be interpreted as a complete board or bootloader change history.

Hardware record	Meaning and action
Standard and Advanced	Use the connector set and power-input ratings specified for the fitted model. Advanced application headers must not be assumed present on Standard.
Pi-button control connection	The M.2 control circuit operates the Raspberry Pi 5 power button through its transistor/contact arrangement. It is not the older Raspberry Pi RUN/PEN connection.
Prototype control-line bias correction	A reported prototype correction removed R4 after reset-time bias of the control transistor caused unintended Pi-button activity. This is a development record, not a general user rework instruction. Follow the schematic for the shipped PCB revision.
RTC backup cable	The optional/specified assembly cable connects the M.2 main-battery RTC backup supply to the dedicated Raspberry Pi 5 RTC power connector.
Backup-source assembly	Cable battery, one 18650 cell and the designated 4000 F option require the matching source arrangement. Do not populate alternative battery connections simultaneously.

Keep a hardware record with the system

Record the PCB marking, assembly model, MCU application version, installed-loader image/hash, backup-source part, switch arrangement and fitted cables. Keep the order-specific schematic/assembly drawing with that record. A firmware banner alone cannot prove that a connector or accessory is fitted.

The assembly and inspection chapters below use external board connections rather than MCU package pin names. Rework should follow the drawing for the exact board; photographs are useful for orientation but do not replace polarity or continuity checks.

System overview and introduction

The M.2 UPS and Power Management HAT combines:

Raspberry Pi 5 Powering

External 5-32 VDC

Passive PoE up to 32 V

Battery backup

Battery Back up for the Raspberry Pi RTC

Power supervision

PCIe SUSCLK 32786 Khz supply

and an M-key M.2 interface in an 85 × 56 mm board designed for Raspberry Pi 5. It supports SSD lengths 2230, 2242 and 2280 and provides backup power to the Raspberry Pi and attached SSD.

During normal operation the Raspberry Pi uses external cable power and the UPS manages battery charging. When external power disappears, the battery supports the system while the controller measures the outage and, when required, requests File Safe Shutdown (FSSD). A Linux service must process that request so the operating system can shut down before power is removed.

System components

Component	Role
Raspberry Pi 5	Runs Linux, applications and the shutdown service.
UPS microcontroller	Supervises power, battery, buttons, timing and power control.
Battery source	Provides temporary power during an outage or mobile operation.
M.2 interface	Connects a compatible PCIe M-key SSD through the Pi FPC interface.
Linux service	Maintains I^2C communication, handles shutdown and synchronizes the UPS RTC.

Two hardware variants

Two models are available. The **Standard Model** provides the core M.2 and UPS functions. The **Advanced Model** adds an external DC input, passive PoE and additional application interfaces. Software support, fitted connectors , Switches, Raspberry Pi 5 RTC Cable, Battery Cover are included in the packega.

Power control principle

The M.2 controller can operate the Raspberry Pi 5 power button through a transistor interface. A shutdown state is different from complete electrical isolation: a cable connected directly to the Raspberry Pi USB-C connector still supplies the Pi power circuitry. Switch-controlled standby therefore also depends on Raspberry Pi EEPROM settings described later.

Standard and Advanced models

The two models share the Raspberry Pi 5 M.2 and UPS functions. The table below summarizes their electrical and mechanical differences; the full comparison is in Technical specifications.

Parameter	Standard	Advanced
Host	Raspberry Pi 5	Raspberry Pi 5
Board dimensions	85 × 56 mm	85 × 56 mm
M.2 SSD lengths	2230 / 2242 / 2280	2230 / 2242 / 2280
M.2 supply capability	5 A	5 A
M.2 reference signal	SUSCLK 32.768 kHz	SUSCLK 32.768 kHz
Battery backup output	5.25 V, up to 5 A continuous	5.25 V, up to 5 A continuous
External source	Pi 5 V GPIO power path	Pi 5 V GPIO and EPR DC input
EPR input	Not fitted	5–32 V DC
Passive PoE	Not fitted	5–32 V DC, variant dependent
Maximum charger current	580 mA	1.1 A
Adjustable charge current	100–580 mA	100 mA–1.1 A
Battery connections	Cable battery or one 18650	Cable battery or one 18650
Supported Battery Chemistry	LiPO, Li-ion, LiFePO4, Super Capacitor 400F	LiPO, Li-ion, LiFePO4, Super Capacitor 400F
Auxiliary backed supplies	Not fitted	5 V / 200 mA and 3.3 V / 200 mA
HAT identification EEPROM	Not fitted	Not fitted

Passive PoE is a dedicated passive DC arrangement. It is NOT compatible with an IEEE 802.3af/at/bt switch or injector. Injector wiring, polarity and the fitted board variant must be confirmed before connection. The maximum DC voltage for passive PoE is 32V DC.

Core features

The **M.2 UPS PMH** combines power backup and a PCIe M-key storage interface on one Raspberry Pi 5 HAT. The controller monitors the source and battery independently of Linux, while the host service turns an FSSD request into an orderly operating-system shutdown.

General

The 85 × 56 mm board uses I²C for routine host interaction. It provides system indicators, a shutdown control, a hardware RTC and support for an optional slide ON/OFF switch as Power Bank. The M.2 interface supports 2230, 2242 and 2280 SSD lengths through a 16-contact Raspberry Pi PCIe FPC connection.

Powering options

In Standard configuration, normal power is supplied through the Raspberry Pi 5 V power path. The Advanced assembly adds the designated (**E**xternal **P**owe**r**ing input) EPR DC input and passive PoE option. Battery backup supports both the Raspberry Pi and M.2 supply. One backup source is used at a time: a compatible cable battery or a cell in the 18650 holder. Battery input is protected against revert polarity insertion.

Battery charging and mobile operation

The charger adapts its demand to source conditions. A source that cannot support both the running system and the requested charging current may produce repeated source transitions. The optional switch provides a controlled mobile-power mode and requests shutdown when moved OFF.

RTC support and system scheduler

The **M.2 UPS PMH** hardware RTC is separate from the Raspberry Pi RTC. The host can read and synchronize it through the DS1307 compatible I²C interface. The hardware specification also provides scheduler functions; application-specific event programming requires the corresponding firmware/API release.

Monitoring and host integration

The Linux service supplies CPU temperature, handles shutdown requests and synchronizes the UPS RTC. Power measurements and status registers are available through the I²C interface.

User and programmer interface

The API is divided into I²C banks. Default address examples in this guide use 0x68 for the RTC, 0x69 for status and 0x6B for commands. User LED, KEY and additional interfaces depend on the selected assembly and firmware support.

Embedded peripherals and interfaces

Application interfaces on the Advanced model

The Advanced specification includes an external application header, auxiliary backed supplies, a user KEY input and user LED connection, analogue input functions, serial connectivity, an IR receiver interface and a 1-wire interface. Check each fitted connector against the board drawing before connecting equipment.

Interface	Installation considerations
Auxiliary 3.3 V and 5 V	Specified at 200 mA each. Include the load in the total power budget and check behavior while the Pi is OFF.
KEY input	Active-HIGH application input on M.2 where fitted. Use the designated 3.3 V contact and common ground. Not used for bootloader selection.
IR interface	Interface for a compatible infrared receiver on the Advanced model. Follow the board connector labels and receiver pinout; receiver decoding depends on the application software.
Serial interface	Use the documented 3.3 V logic-level connection. A true ± voltage RS-232 interface requires a compatible level converter.
Analogue input	The specification describes pre-scaled ranges. Confirm channel, connector, range selection and maximum voltage before use.
1-wire	The interface can use Raspberry Pi GPIO4 when enabled. It is an optional use of GPIO, separate from normal I²C interaction.

Embedded sensors

Power monitoring uses the fitted PAC1720 device and voltage/current/power measurement circuitry. The available monitoring points differ between Standard and Advanced. Do not apply an external measurement voltage to an unverified application pin.

Serial diagnostics and user applications

The diagnostic terminal is useful for startup and power-transition traces. A user application that also opens the same serial device can interfere with diagnostics or firmware transfers. Give one process ownership of the selected port during each task.

Serial signals are 3.3 V logic level. Connect a compatible adapter and common ground; a true positive/negative-voltage RS-232 connection requires a level converter.

System protection and PCB construction

System protection

The **M.2 UPS PMH** combines MCU supervision, battery cut-off functions and power-path protection. The technical specification lists PPTC, ESD and reverse-polarity measures on the appropriate power inputs. These features do not make every connector tolerant of arbitrary voltage or incorrect wiring.

The Linux service remains essential to File Safe Shutdown. Hardware can sustain the supply and request shutdown, but Linux must close applications and storage before the controller completes its power sequence. Allow enough grace time for the actual workload.

Watchdog and still alive monitoring

A still-alive mechanism can monitor host communication. Select watchdog actions and intervals only from the matching firmware specification. Do not experiment with an undocumented watchdog command on a system with unsaved data.

Power button control

The board operates the Raspberry Pi 5 button through a transistor interface where fitted. The MCU control signal is not an independent hard-reset input on the Raspberry Pi processor. Direct USB-C power remains electrically present when the Pi is in standby.

PCB construction

The supplied **M.2 UPS PMH** specification describes a four-layer PCB with 2 oz copper, immersion-gold finishing, cooling planes and thermal vias. The central opening supports air circulation from the Raspberry Pi official cooler and helps keep the SSD area ventilated.

Enclosure and accessory clearance

A dedicated enclosure specially designed for the **M.2 UPS PMH** and Raspberry Pi 5 assembly is available. Select the enclosure version suited to the fitted SSD, battery arrangement and optional accessories.

Verify the assembled height, battery access, USB-C and Ethernet access, SSD fixing point, cooler clearance and FPC bend radius. Case compatibility cannot be inferred from the Raspberry Pi footprint alone. Keep conductive case parts and loose screws away from the board underside.

Add on equipment and package contents

What is in the package

Check the order-specific packing list before assembly. Models and kits may differ in battery, holder, switch, spacers, FPC cable and accessory connections. A mechanical illustration is not a packing list.

Item to check	Purpose
M.2 UPS PMH HAT and assembly identification	Confirms Standard or Advanced and PCB revision.
Compatible PCIe FPC cable	Connects the Pi PCIe port to the HAT.
Spacers and fixing hardware	Supports the boards and the selected SSD length.
One approved backup source	Cable battery, 18650 or the approved supercapacitor option.
Power-button connection (POGO PIN)	Required for the intended Pi power-control arrangement.
Optional accessories	Slide switch, 18650 Battery Plastic Cover, RTC supply cable and external application connections as ordered.

Batteries and holders

The cable socket and 18650 holder are alternative backup connections. Select cell chemistry, protection, charge current and discharge capability as a complete configuration. Do not parallel two sources through the available connectors.

Supercapacitor option

The 4000 F supercapacitor option requires its designated assembly and charge profile. Install the selected backup source according to its connector polarity and mechanical mounting arrangement.

Through hole and external parts

For a kit requiring soldering, follow the assembly drawing for its PCB revision and inspect every joint before applying power. Keep fitted components in their designated positions.

Application specific add on extensions

Select accessories for the actual **M.2 UPS PMH** model and PCB revision. The base package and optional items depend on the order. A header location or a reserved register does not by itself provide an installed peripheral or a complete application driver.

Accessory or extension	Selection and use
RTC backup supply cable	Dedicated cable offered by Pi Modules Technologies, from the M.2 main-battery backup supply to the Raspberry Pi 5 RTC connector. No separate Pi RTC battery is needed while this supply is available.
Slide ON/OFF switch	Use the specified onboard part or a compatible external switch wired to the designated switch connections. Enable slide-switch mode for its intended behavior.
18650 protective cover	Use the cover matching the single-cell assembly. Confirm contact insulation, battery access and cooler/SSD clearance.
Dedicated system enclosure	Select the enclosure for the fitted Standard/Advanced board, battery arrangement, SSD, cooler and cable exits.
Advanced application connections	Compatible IR receiver, KEY input, analogue/1-wire connections and serial accessories where fitted. Check interface level, pinout and software support separately.
Serial level converter	The documented serial connection is 3.3 V logic. A true RS-232 connection requires a suitable converter; never connect ± RS-232 levels directly.

PICTURE IS MISSING

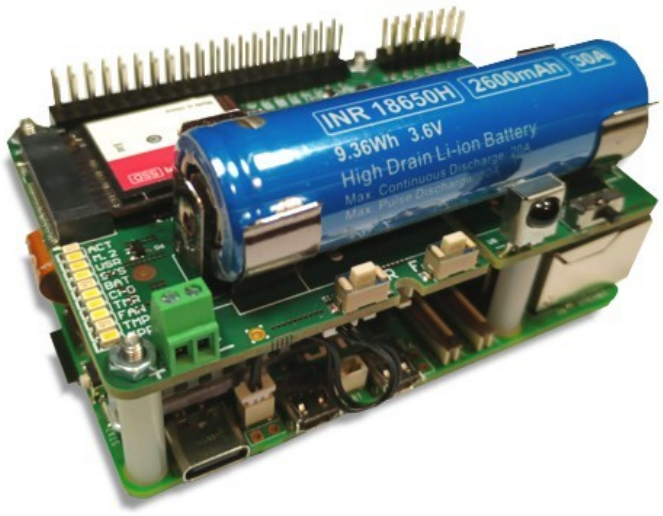
Image 1 M.2 accessory set with the dedicated RTC cable, slide switch, battery cover and matching connectors identified.

Prepare an application extension

Disconnect cable and battery power before attaching hardware. Identify supply, ground, signal direction and maximum signal voltage from the board-specific connection drawing. Check the accessory's pinout independently of its wire colors. Route and secure wires away from the fan, battery contacts, SSD and underside components.

Commission one extension at a time, first under cable power and then with the intended power transitions. Keep the normal shutdown daemon active. Application code must not compete with its shutdown/RTC ownership or repeatedly write reserved registers.

Advanced assembly configurations



Advanced model with a single 18650 battery

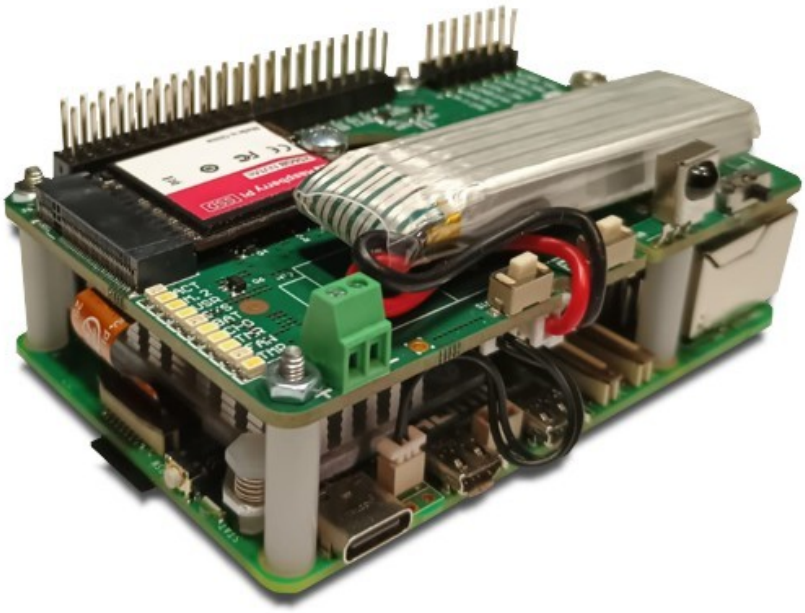
This assembly uses the single-cell 18650 holder. Select the battery profile matching the cell chemistry; 18650 identifies a physical cell size, not its chemistry. The protective-cap configuration shown on the front cover uses the same holder.

Supported backup sources

Only the following four backup sources are supported. Use one backup source at a time and select the corresponding profile before enabling charging.

Backup source	Code	ASCII value
LiPo battery	L	0x4C
Li-ion battery	I	0x49
LiFePO4 battery	F	0x46
Supercapacitor 4000 F	C	0x43

Ultra high current battery configuration



Advanced model with a 650 mAh ultra high current battery

The cable-connected 650 mAh ultra high current battery is an alternative to the cell in the 18650 holder. Capacity is expressed in mAh; it is separate from charging current, expressed in mA. Use the profile specified for the fitted battery and observe the connector polarity.

Do not connect this battery and a cell in the 18650 holder at the same time. Battery capacity and discharge capability must suit the Raspberry Pi, SSD and attached peripherals. The maximum charging-current specifications remain 580 mA for Standard and 1.1 A for Advanced. Battery Charger is starting and stopping automatically.

Battery and supercapacitor installation

Battery selection and capacity

Choose the backup source by chemistry, usable energy, continuous discharge capability, protection circuit, connector and physical fit. Capacity in mAh alone does not establish that a battery can supply Raspberry Pi 5 and an SSD. Use one approved source and its matching firmware profile.

Source	Specification for selection	Profile
Standard cable battery	Protected 3.7 V, 650 mAh, 15C LiPo specified for the system. Leads and protection must support the complete load.	L / 0x4C
Single 18650 Li-ion	One protected compatible cell. Capacity and permitted current are those on the selected cell's data sheet; confirm actual length and holder fit.	I / 0x49
Compatible LiPo assembly	Use the approved cable/holder arrangement and its labelled voltage, capacity and discharge rating. A cylindrical package does not define chemistry.	L / 0x4C
Compatible LiFePO4 assembly	Use a cell and holder/cable approved for the board, with the correct chemistry-specific charging profile.	F / 0x46
Designated supercapacitor	Use the matched 4000 F assembly and factory profile, not an arbitrary capacitor with the same capacitance.	C / 0x43

The 18650 name identifies a cell format, not a guaranteed protected-cell length or current rating. Check the actual cell dimensions and the assembled clearance. The listed 650 mAh pack is the defined standard pack; other capacities are not implied to be supplied or approved by this manual.

PICTURE IS MISSING

Image 2 Approved battery variants beside their labels, with connector polarity, actual dimensions and continuous-discharge ratings visible.

Charging-current choice

Standard charging is limited to 580 mA; Advanced charging is limited to 1.1 A. These are board limits, not a requirement to charge every cell at that current. Select a source whose permitted charge current is compatible, and use the corresponding approved configuration. The controller may reduce charging when input power is limited.

Fit the cable battery or single cell

1. Shut Linux down normally. Disconnect external power and the backup source before handling connectors or holders. Remove conductive tools, jewellery and loose hardware from the battery area.

2. For the cable battery, inspect the XH 2.5 mm mating connector, lead condition and polarity against the marked board connection. Use the correct prewired connector; do not infer polarity solely from a third-party plug shape.

3. For the 18650 arrangement, inspect the contacts and insulation and confirm the cell's chemistry, polarity and physical fit. Insert only one compatible cell, in the marked direction, without forcing it or damaging its wrapper.

4. Use either the cable battery connection or the 18650 holder. Never connect a cable battery and a cell in the holder together. Keep unused contacts insulated and fit the appropriate cover.

PICTURE IS MISSING

Image 3 Close-up of the cable battery socket and single-cell holder, with positive/negative markings and the two alternative connection routes.

5. Secure the pack or holder so the leads cannot reach the cooler, sharp PCB edges or SSD hardware. Confirm that the holder/cover cannot touch exposed conductive parts when the enclosure is closed.

6. Verify the source profile against the fitted chemistry before permitting charging. Do not experiment with profile changes while an unknown or incompatible source is connected. Use the configuration intended for the supplied assembly.

First charge and acceptance

Apply cable power and observe normal startup. Confirm cable-mode recognition, charging permission and actual charging behavior; permission alone is not proof of charging current. The reported current-limit value is also not a direct current measurement. Inspect supply and battery temperature during the first charge.

A complete charge time cannot be predicted from capacity divided by the maximum board current alone. Charge taper, starting state of charge, load sharing, source limits and temperature all affect it. Charge adequately before the timed FSSD and repeated restart tests, and record the battery used with those results.

Install and test the 4000 F option

Use the designated M.2 supercapacitor assembly and its matched C / 0x43 profile. Verify the module's rated voltage, polarity, connector, mounting and factory configuration from the supplied assembly label/drawing. Do not substitute an unapproved part or combine it with an alternative backup source.

A supercapacitor may remain charged after external power is removed. Disconnect it before fitting other parts and follow the assembly's discharge instructions before handling exposed contacts. Never discharge it by shorting its terminals with a tool.

PICTURE IS MISSING

Image 4 Designated 4000 F assembly fitted to the M.2 system, showing polarity, insulation, fixing points and the approved connection.

Charging and startup

Start under stable cable power. An initially discharged source needs charging before it can provide a useful outage reserve. Allow charging to reach the expected ready condition for the matched assembly; do not judge readiness from elapsed time alone. Repeat tests only after restoring a comparable charge level.

Supercapacitor energy and battery equivalence

For a nominal effective capacitance of 4000 F charged to 4.2 V, the ideal stored energy is $E = ½ \times C \times V^2 = 35{,}280$ J = 9.80 Wh. Dividing by a 3.7 V nominal Li-ion battery voltage gives an energy-equivalent capacity of 2.65 Ah (about 2650 mAh). This total includes energy below the UPS operating range; it is not the usable backup capacity or a recommendation to discharge the module to 0 V.

The V88 C profile uses a nominal 2.8 V low-source threshold to request FSSD. Between 4.2 V and 2.8 V, $E = ½ \times 4000 \times (4.2^2 - 2.8^2) = 19{,}600$ J = 5.44 Wh, equivalent in energy to about 1470 mAh at 3.7 V before conversion losses. With an illustrative 85% conversion efficiency, the regulated output receives about 4.63 Wh. Runtime t = 60 × 4.63 / load in watts gives the estimates below; 85% is an assumption, not a measured board efficiency.

Constant output load	Estimated time, 4.2 to 2.8 V at 85%
5 W	About 55.5 minutes
10 W	About 27.8 minutes
15 W	About 18.5 minutes

These are constant-capacitance energy estimates, not measured runtime guarantees. Confirm the effective capacitance over this voltage range for the supplied assembly; voltage-dependent capacitance, ESR, temperature, ageing and converter losses affect the result. A configured runtime timer can request shutdown earlier. The 2.8 V threshold starts FSSD, not an instantaneous power cut: retain reserve for Linux shutdown and the final power sequence. Validate the complete Pi/SSD assembly under its actual load.

Power supply unit recommendations

The external supply must power Raspberry Pi 5, SSD, attached peripherals and battery charging at the same time. Choose a suitable Raspberry Pi 5 USB-C supply and cable; the system guidance calls for at least 3 A and preferably 5 A where the workload requires it. Confirm the supply's actual capability and connector rating, not only the marking on an extension cable.

Estimate the complete power budget

As an illustration, a 5 V, 5 A source has a nominal 25 W budget. An assumed 10 W Pi load, 3 W SSD, 2 W peripherals and 5 W allocated to charging/conversion total 20 W, leaving 5 W nominal margin. These are example assumptions, not measured loads or guaranteed board output ratings; measure the real assembly and allow for startup peaks and cable losses.

Standard and Advanced have different maximum charging currents. Charging power and converter loss must be included in the input budget. If the source is marginal, the controller may reduce charging or report repeated battery/cable transitions even while the cable remains attached.

Choose the correct power input

For ordinary USB-C operation, connect the source to Raspberry Pi 5 as documented. On Advanced, use 5–32 V only at the designated EPR or approved passive-PoE input arrangement, observing its polarity and order-specific connection. Never apply that higher voltage to the Raspberry Pi 5 V header, RTC input or a signal connector.

PICTURE IS MISSING

Image 5 USB-C and Advanced external-power connections, with connector polarity, permitted input locations and cable routing identified.

Check voltage drop under load

Use short, suitable power leads and sound connectors. Test with the SSD active, peripherals connected and charging enabled. Capture cable/battery mode changes, reduced charge current, Linux undervoltage indications and resets. A lightly loaded idle test cannot establish that the supply is adequate during charging or storage activity.

```
vcgencmd get_throttled
sudo journalctl -k -b --no-pager | grep -Ei "voltage|under.?voltage|power"
sudo journalctl -u M2_PMH_i2c.service -b -n 15 --no-pager
```

Resolve the source/cable problem before repeating outage tests. A successful backup transition is not evidence that the cable supply has enough continuous power for the intended workload.

Assembly of the Magic ON/OFF Switch

Fit the onboard ON/OFF switch

These instructions apply to a kit or board intentionally supplied for user assembly. If the switch and connectors are factory fitted, inspect them rather than removing or reheating them. Work with USB-C, all other external supplies and the backup source disconnected.

1. Identify the dedicated slide-switch footprint and the specified switch. Check orientation, mounting tabs, travel and enclosure access before soldering. The optional switch is an application control, not a contact for switching the full Pi supply current.

2. Seat the part squarely and mechanically support the board. Tack one joint, recheck alignment and free movement, then solder the remaining required joints without bridging adjacent pads.

PICTURE IS MISSING

Image 6 Onboard slide switch positioned in its M.2 footprint before soldering, showing orientation and OFF/ON travel.

3. Inspect both PCB sides for bridges, splashes and excessive protruding leads. Trim only where the assembly requires it and remove loose cuttings. Operate the switch through its full travel and confirm that it cannot contact the case or battery.

4. Before enabling switch mode, place the switch ON so that selecting the mode does not immediately request a deliberate OFF shutdown. Use the Intelligent mobile power bank and slide switch chapter to configure and read back the operating mode.

Use an external switch

Use the specified switch connections and the approved contact arrangement. Transfer every required contact according to the M.2 assembly drawing; do not assume that two arbitrary pads are a simple power switch. Check the contact mapping with power removed and keep the control wiring away from supply and fan wiring.

PICTURE IS MISSING

Image 7 External switch wiring diagram showing each board contact, the matching switch terminal and both switch positions.

After assembly, test ON/OFF with cable present and absent. OFF must produce the intended orderly shutdown and inhibit an unwanted restart in switch mode. Return to normal timed UPS mode if the switch is not being used. This test assumes a working Linux shutdown daemon.

Assembly of the THT 40 Pins Connector

Use the correct 2 × 20 connector supplied for the board and its intended stacking height. Confirm pin-1 orientation, gender, lead length and mechanical clearance against the fitted M.2 assembly. Do not substitute a header whose long ends can touch the battery, cooler or Raspberry Pi components.

1. Disconnect all sources. Inspect the PCB connector footprint, pads and surrounding components. Verify that a factory-fitted socket/header is not being mistaken for an unpopulated assembly location.

2. Align all 40 pins before insertion and support the connector body evenly. For an assembly designed to pass through an existing socket, keep the pins straight and follow that socket's guide alignment. Do not force a bent pin or press on an unsupported PCB.

PICTURE IS MISSING

Image 8 Correct 40-pin connector orientation and pin-1 alignment, shown from the top and mating side of the M.2 board.

3. Seat the connector to the intended height. Tack two diagonally opposed joints where the design calls for soldering, check that the connector is square and parallel, then complete the remaining joints.

4. Inspect every joint for wetting, bridges and loose solder. Check clearance on both sides. Trim leads only if the specified connector/assembly calls for it; do not shorten the functional mating section of a stack-through header.

PICTURE IS MISSING

Image 9 Finished 40-pin joints and connector seating height, with an example of the required underside clearance.

Inspect before powering

With sources disconnected, check for solder bridges and unintended continuity between adjacent supply and ground contacts. Mate the assembly carefully without a one-row or one-column offset. Confirm spacers support the boards and that no connector force is carried by the Pi-button spring contact. Apply cable power first and inspect normal startup before connecting extra loads.

Assembly of the Gold-Plated POGO Pin

The Pi-button contact must meet the designated Raspberry Pi 5 power-button connection used by this M.2 assembly. It is not an instruction to contact an older Raspberry Pi RUN or PEN pad. Identify the correct point from the Raspberry Pi 5/M.2 drawing before fitting the pin.

1. Disconnect external power and the backup source. Fit the correct spacers and temporarily assemble the boards without soldering the contact. Check the available gap, contact centre and clearance from neighbouring copper and components.

PICTURE IS MISSING

Image 10 Raspberry Pi 5 power-button contact location and the matching POGO position on M.2, with the contact centre marked.

2. Fit the specified spring contact in its intended orientation. The spring tip must touch the correct point when the boards are fully supported by their spacers. It must have working travel and must not be rigidly bottomed out, tilted or used as a structural spacer.

3. Where soldering is required, tack the board-side joint while maintaining alignment. Remove power-free trial assemblies as necessary to inspect seating. Complete the specified joint without allowing solder to enter the spring mechanism or contact tip.

PICTURE IS MISSING

Image 11 Side view of the fitted spring contact, showing the supported board spacing, contact alignment and remaining spring travel.

Final mechanical check

Reassemble with the intended SSD, cooler, battery cover and case. Verify contact engagement without PCB bending and confirm that tightening the case does not create a permanent button press. Do not copy spring compression dimensions from another Pi model; use the specified M.2 contact and spacer stack.

Check the power-control function

Start under cable power with the normal Linux daemon running. Verify normal heartbeat, then perform the documented FSSD and power-return cycle. Repeat with the final enclosure fitted and, where used, switch OFF/ON. A Pi that stays in standby or stops unexpectedly can indicate a misaligned or continuously pressed contact; disconnect power and inspect it before altering firmware settings.

A static voltage check alone does not establish correct pulse timing or mechanical contact. When diagnosing a failure, record the actual Raspberry Pi button signal and the serial log around the event. The POGO tip and an internal transistor control point are different measurement locations.

Installation step by step

Complete the hardware preparation and assembly under stable cable power. For Linux setup, choose either the Automatic Linux installer chapter or manual Steps 3–6 below. Steps 7–10 verify operation after either route. Do not run both software setup routes on the same card without reviewing the existing configuration.

Step 1 — Prepare the hardware and operating system

1. Identify the HAT model, PCB revision and firmware supplied with it.

2. Confirm the battery type, polarity and intended backup connector.

3. Check the SSD is a supported PCIe M-key device and select its fixing point.

4. Identify the 16-contact FPC, its contact orientation and both latches.

5. Check spacer heights, cooler clearance and the Pi power-button connection.

6. Keep RESET, FSSD and the optional slide switch accessible for testing.

Software preparation

Boot Raspberry Pi OS from a working microSD or NVMe installation. Confirm network or console access before relying on UPS-controlled shutdown.

```
cat /etc/os-release
uname -a
lsblk -o NAME,SIZE,FSTYPE,MOUNTPOINTS
lspci
```

Back up the operating system and application data before changing storage or power configuration. Verify the selected NVMe device and boot order separately from UPS installation.

Install the hardware, enable I²C, configure the UPS RTC and start the service before performing a cable-loss test.

The latest image of an SD card is available in the download section in ***www.pimodules.com***.

Step 2 — Assemble and inspect the boards

Disconnect cable and battery power before assembly. Use the sequence below, then compare the finished assembly with the hardware figures in this chapter.

Connect a suitable external source before relying on battery startup. The battery isolation and start conditions depend on the assembly. Confirm the MCU startup banner, Linux boot and correct power mode; do not interpret a momentary startup transition as a completed test.

Mechanical assembly

1. Shut down the Raspberry Pi. Remove the USB-C supply, all other power sources and the UPS battery.

2. Fit the Raspberry Pi official cooler and the correct spacers. Keep the cooler air path and the HAT ventilation opening clear.

3. Install a compatible M-key PCIe SSD. Use the fixing point for its length and secure it without bending the board.

4. Fit the HAT onto the 40 pin connector without a one-row or one-column offset. Secure the assembly with the appropriate spacers and screws.

5. Connect the PCIe FPC with the correct orientation at both ends. Open and close the connector latches carefully; do not force the cable. For easy installation fit it first to the Raspberry Pi and then after assembly to the **M.2 UPS PMH** socket**.**

6. Verify the Raspberry Pi power-button contact (POGO PIN) or cable arrangement where fitted. Check clearance around the battery holder, SSD and case.

7. Connect one approved battery with correct polarity. Recheck the selected chemistry before permitting charging.

First power-up checks

Apply external cable power first. Observe the startup indication and confirm that the Raspberry Pi boots. Install or check the Linux service before testing removal of external power. Keep a local serial capture available during initial commissioning.

```
uptime
sudo systemctl status M2_PMH_i2c.service --no-pager
sudo journalctl -u M2_PMH_i2c.service -b -n 15 --no-pager
```

Verify that the service reports successful I^2C reads and that new errors are not accumulating. Confirm the battery is charging under stable cable power. Do not begin repeated power-cycle tests while the battery state or supply quality is uncertain.

SSD boot configuration is a separate Raspberry Pi task. Confirm the intended SD or NVMe boot path before testing UPS shutdown and restart.

Hardware connections top view

Top view of the Advanced assembly, showing the SSD area, controls, power connections and application interfaces.

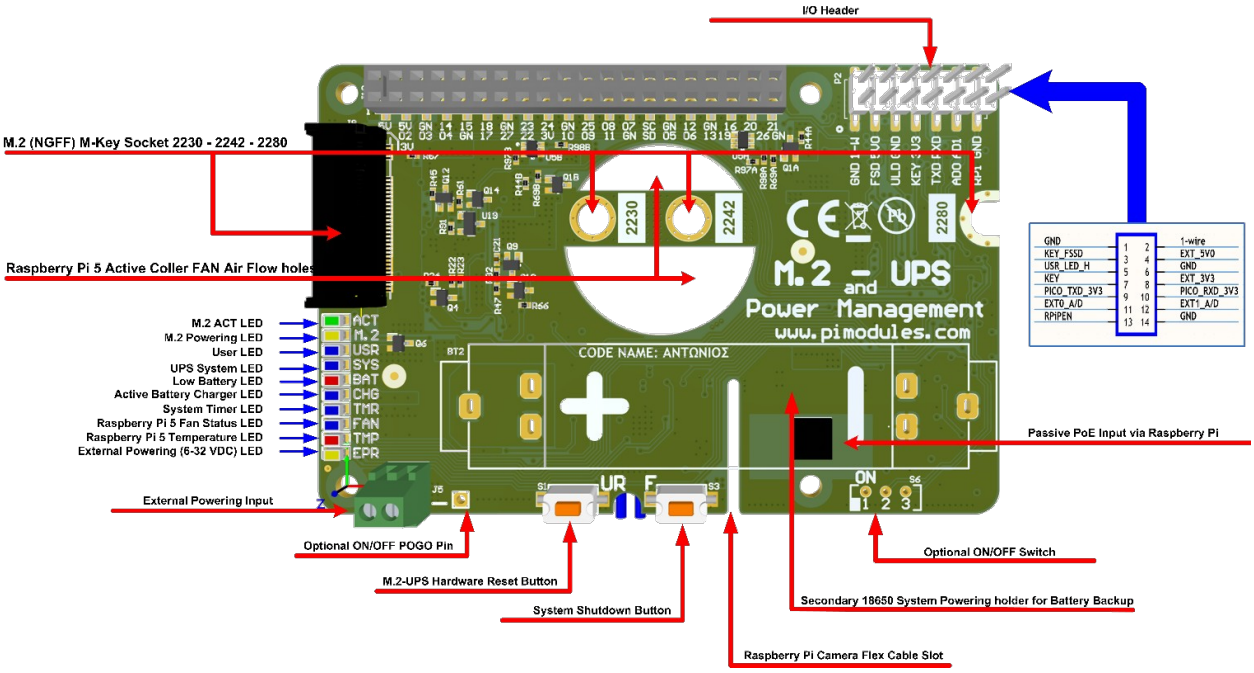


Figure 2 Advanced board top view with original reference annotations

How to use this view

Locate the socket and fixing point for the selected SSD length before fitting the SSD. Keep the central cooler opening clear. Locate RESET and FSSD before commissioning, and make sure a case does not hold either button down.

Use the board labels to identify each connection. KEY is an application input on the Advanced model. Bootloader entry uses only M.2 RESET and FSSD and is available on both Standard and Advanced.

The battery footprint shows polarity markings. Check the markings on the actual fitted holder and battery connector. The external-power terminal shown is an Advanced feature, not a Standard-model connection.

Hardware connections underside view

This underside view shows the controller and power circuitry, PCIe FPC connector, battery socket and the external interface locations. Check that spacers, screws and the Raspberry Pi cooler cannot contact exposed components.

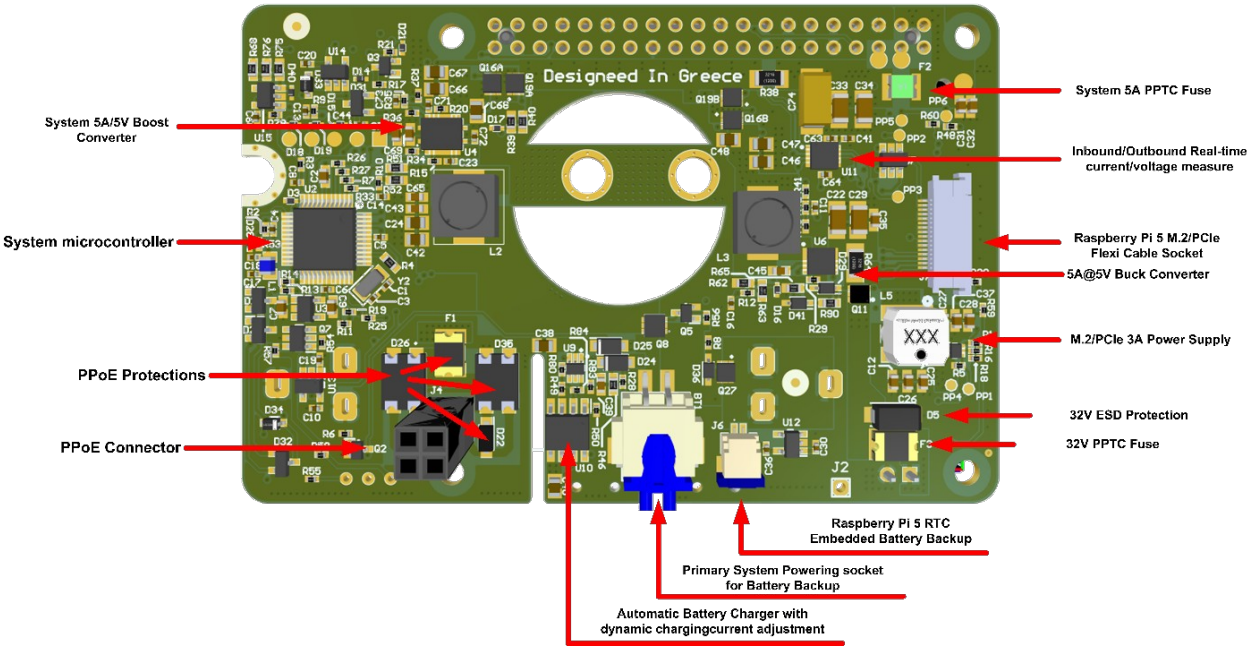


Figure 3 Advanced board underside with original reference annotations

Connections to inspect before assembly

Verify the PCIe cable orientation while the connector is accessible. Confirm the intended battery connection and the Raspberry Pi RTC backup cable route. If the power-button contact is fitted, verify its alignment and insulation before applying power.

The underside view identifies the battery-holder and board-to-board assembly area. Keep the battery contacts insulated from adjacent boards and metal enclosure parts.

SSD and mechanical assembly illustrations

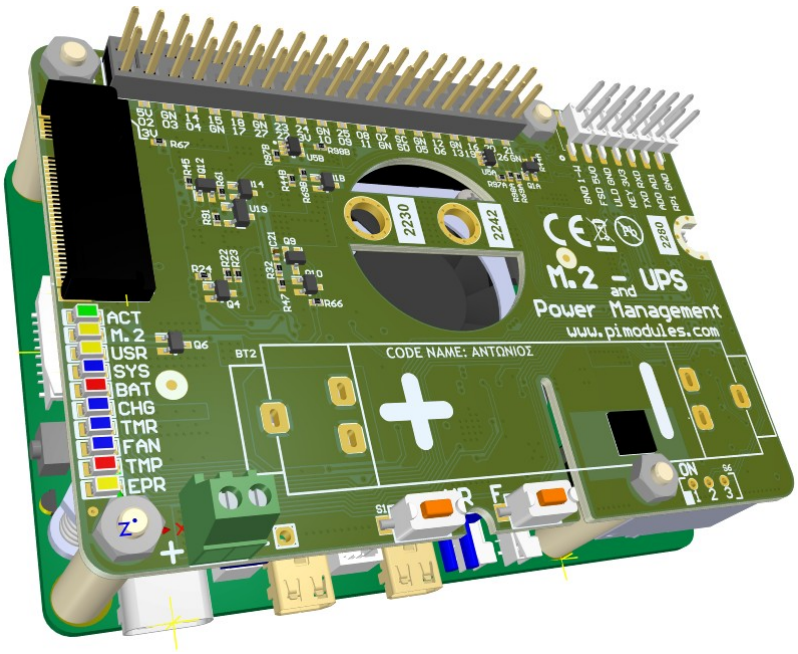


Figure 4 HAT fitted above the Raspberry Pi without battery or SSD

The empty assembly view exposes the SSD mounting positions, central air opening, indicator labels and front controls. It is useful for checking access before fitting the battery holder and storage.

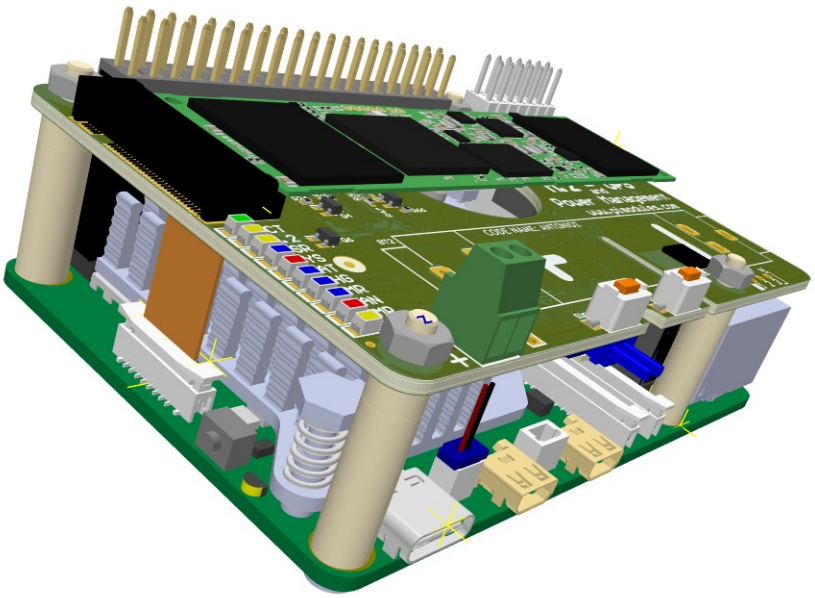


Figure 5 SSD installed with cooler and inter-board spacing visible

Keep the SSD fixing screw, cable bend and cooler opening clear of the battery and enclosure. Secure the assembled boards without bending either PCB.

Power supply and battery preparation

Prepare the external power supply and one compatible backup source before first use. The following steps cover the battery connection, charging profile, safety checks and supply capacity needed by the Raspberry Pi 5, SSD and attached peripherals.

Choose the correct backup source

Use either the cable battery connector or the single-cell 18650 holder. Do not populate both at once. The specified cable connector is XH 2.5 mm. Check polarity at the board and battery; a connector that physically fits is not sufficient evidence of electrical compatibility.

Only these backup sources are supported: LiPo (L, 0x4C), Li-ion (I, 0x49), LiFePO4 (F, 0x46), and the designated 4000 F supercapacitor (C, 0x43). Select the profile matching the fitted source before charging. Other battery chemistries and backup-source codes are not supported.

The standard battery specification is a protected 3.7 V, 650 mAh, 15C LiPo with suitable high-current leads. Use the charging profile matching the fitted backup source.

Handle batteries safely

Disconnect external power and the battery before mechanical work. Prevent short circuits and reverse polarity. Do not use damaged, swollen or overheated cells. Keep the cell clear of sharp component leads, SSD heat and moving fan parts. Stop operation if abnormal temperature, smell or swelling occurs.

Provide enough supply margin

The external supply must support the Raspberry Pi, SSD, peripherals and charging demand. A marginal supply can cause repeated cable/battery transitions or reduce charging current. The charging algorithm adjusts demand to available power; maximum charge current is not maintained under every condition.

Use an appropriate Raspberry Pi 5 USB-C supply and suitable cables minimum 3A, preferable 5A. Apply 5–32 V only to the designated Advanced EPR or approved passive PoE connection. Never apply that voltage range to the Raspberry Pi 5 V header or an application input.

Runtime planning

Battery runtime depends on usable battery energy, discharge limits, converter efficiency and the complete load. A programmed runtime limit specifies when shutdown begins; it is not a promise that the battery can sustain that interval. Low-battery protection can shorten operation.

Step 3 — Install Linux dependencies and enable I²C

Extract this package into ~/M2_UPS_PMH_Manual_V8F. Commands below assume Raspberry Pi OS, apt, systemd and the default I²C address group.

```
cd ~/M2_UPS_PMH_Manual_V8F
sudo apt update
sudo apt install -y python3-smbus i2c-tools util-linux python3-serial
sudo raspi-config nonint do_i2c 0
```

The daemon imports the distribution smbus module. A separate pip environment is not required for this installation. The service runs with the privileges needed for I²C, RTC access and system shutdown.

Enable the I²C interface

```
sudo raspi-config
```

Select Interface Options, then I²C and enable it. On current Raspberry Pi OS, configuration is normally stored in /boot/firmware/config.txt. If using another image, confirm its active boot configuration path before editing.

```
ls -l /dev/i2c-*
sudo i2cdetect -l
```

After applying the interface change, reboot when requested by the configuration tool. Keep stable cable power connected. A raw I²C scan should not be used as a continuous health monitor; the daemon logs provide the routine communication check.

Confirm the configured bus

The daemon uses bus 1 on GPIO2 SDA and GPIO3 SCL. Other overlays or a changed address group require corresponding software changes. Do not move the wiring or change the API address merely because the scan shows several adjacent addresses.

Preserve existing boot settings

```
sudo cp /boot/firmware/config.txt \
  /boot/firmware/config.txt.before-m2-pmh
```

Step 4 — Configure and identify the UPS RTC

Add the RTC overlay

At the default address group, the UPS exposes a DS1307-compatible RTC at 0x68. Add the following line under an applicable [all] section of the active boot configuration, without duplicating an existing entry.

```
sudo nano /boot/firmware/config.txt
```

```
[all]
dtoverlay=i2c-rtc,ds1307
```

If the same section already contains an I²C setting, preserve it. The explicit interface setting, when needed, is dtparam=i2c_arm=on. Do not put the RTC overlay under a section that applies only to another Pi model.

Reboot and identify the RTC

```
sudo reboot
```

After reconnecting over SSH, inspect the RTC names. The Raspberry Pi 5 already has its own RTC, so the UPS is often rtc1, but enumeration must be checked rather than assumed.

```
ls -l /dev/rtc*
for d in /sys/class/rtc/rtc*; do
  echo "$d"
  cat "$d/name"
done
```

The expected UPS name includes rtc-ds1307 and 1-0068. A driver claiming address 0x68 may appear as UU in an I²C scan. This is normal kernel ownership, not evidence of a missing device.

If no UPS RTC appears

```
sudo journalctl -k -b --no-pager | \
  grep -Ei "rtc|i2c|SDA|probe"
```

Confirm the overlay, I²C bus and bank selection. Save any probe failure. For the specific case of an existing, unbound 1-0068 device, the recovery binding command is documented in the RTC chapter. Do not create a duplicate I²C device or force raw access to a kernel-owned address.

Step 5 — Install the shutdown daemon

Copy the matching script

From the extracted package root, use the files in installation/. On an existing installation, back up the current service and deliberate overrides before replacing them.

```
cd ~/M2_UPS_PMH_Manual_V8F
sudo install -m 0755 installation/M2_PMH_i2c.py \
  /usr/local/sbin/M2_PMH_i2c.py
```

```
python3 -c "from smbus import SMBus; print('smbus OK')"
python3 - <<'PY'
import ast
from pathlib import Path
ast.parse(Path("installation/M2_PMH_i2c.py").read_text())
print("syntax OK")
PY
```

A syntax check does not validate the firmware protocol or power-control wiring. Use only the script intended for this hardware and application release.

Do not run duplicate daemons

If the service is already installed and healthy, retain its existing unit and review it with systemctl cat. Reinstalling or restarting is unnecessary merely to check the system. A second daemon or a foreground check can compete with the active service for bus access.

```
sudo systemctl cat M2_PMH_i2c.service
pgrep -af M2_PMH_i2c.py
```

Optional read-only check

On a new installation with no daemon running, the supplied script provides a ten-second check mode. It reads the heartbeat without acknowledging shutdown, writing temperature data or synchronizing the RTC. It does not provide operating-system shutdown protection.

```
sudo /usr/bin/python3 /usr/local/sbin/M2_PMH_i2c.py --check
```

Keep external power connected during this check. After it completes, install and enable the normal service before testing an outage.

Step 6 — Install and enable the systemd service

Install the supplied unit. For a service already running, replace its files under cable power and restart it promptly after installation.

```
sudo install -m 0644 installation/M2_PMH_i2c.service.txt \
  /etc/systemd/system/M2_PMH_i2c.service
```

```
[Unit]
Description=M2 PMH I²C heartbeat and shutdown daemon
After=systemd-modules-load.service

[Service]
Type=simple
User=root
ExecStart=/usr/bin/python3 /usr/local/sbin/M2_PMH_i2c.py
Restart=on-failure
RestartSec=2

[Install]
WantedBy=multi-user.target
```

Restart=on-failure allows recovery from a program failure without automatically restarting the daemon after its normal shutdown path. The service does not need to wait for network synchronization before starting I²C polling; its RTC task waits separately for NTP.

Start the service

```
sudo systemctl daemon-reload
sudo systemctl enable --now M2_PMH_i2c.service
sudo systemctl status M2_PMH_i2c.service --no-pager
sudo journalctl -u M2_PMH_i2c.service -b -n 15 --no-pager
```

Expected startup messages identify v1.1, the 250 ms heartbeat, bus 1 and address 0x6B. When synchronization is enabled but NTP is not ready, a waiting-for-confirmed-NTP message is expected.

Existing overrides

Use systemctl cat to review drop-in files. An override that supplies --no-rtc-sync prevents automatic RTC updates even if the main unit omits that option. Preserve deliberate site settings and change only the relevant option.

Step 7 — Verify heartbeat, time and charging

M2_PMH_i2c.service polls bus 1, address 0x6B, register 0x00 every 250 ms. Keep the service enabled whenever UPS-controlled shutdown is required.

Check the running service

With the installation complete, confirm that systemd reports the service active and that successful-read totals increase by approximately 240 each minute.

```
sudo systemctl enable M2_PMH_i2c.service
sudo systemctl start M2_PMH_i2c.service
sudo journalctl -u M2_PMH_i2c.service -b -n 15 --no-pager
```

The installed script is /usr/local/sbin/M2_PMH_i2c.py. A service restart interrupts heartbeat traffic briefly; perform maintenance while cable power is present.

Check the integration settings

```
sudo systemctl cat M2_PMH_i2c.service
timedatectl status
ls -l /dev/rtc*
```

RTC synchronization waits for confirmed NTP synchronization, then writes Linux UTC time to the UPS RTC and repeats hourly. The heartbeat continues while synchronization waits.

Step 8 — Select normal UPS or slide-switch operation

```
sudo i2cget -y 1 0x6b 0x13 b
sudo i2cget -y 1 0x6b 0x01 b
```

Normal UPS operation reads 0x00 and 0x01. Slide-switch operation reads 0xAA and 0xFF. Use the switch configuration chapter to change mode.

Step 9 — Test battery takeover and shutdown

With the battery charged and the service active, remove cable power. Raspberry Pi and the SSD remain running. In normal mode the controller requests FSSD after about 70 seconds, then allows about 30 seconds for shutdown. Keep the battery connected.

Step 10 — Test power return and the optional switch

Reconnect cable power after shutdown and check normal startup. Repeat with cable return during FSSD. If switch mode is enabled, verify OFF prevents startup with cable connected and ON permits it. Complete at least five successive cycles, then inspect the service journal for accumulating errors.

Automatic Linux installer

The **M.2 UPS PMH** installer V01 prepares Raspberry Pi OS for the existing shutdown daemon, RTC synchronization and serial firmware-update connection. It is an alternative to manual software Steps 3–6. Hardware assembly, backup-source selection and the later power-loss tests remain separate tasks.

Supported system and validation status

Use a Raspberry Pi 5 Model B with Raspberry Pi OS Bookworm or Trixie, Python 3, apt and systemd. The installer accepts the Debian/raspbian operating-system identifiers with version 12 or newer. This edition is supplied for use with the validated M.2 application and bootloader V88; it does not install them.

Installer V01 has passed 15 host-side tests using a temporary Linux filesystem and simulated system commands. A clean-card test on Raspberry Pi 5 hardware has not yet been completed for this manual edition. The procedures below describe the implemented installer; a successful simulated test is not a hardware validation result.

Prepare the card and hardware

1. Write Raspberry Pi OS to the microSD card using Raspberry Pi Imager. Set the user name, password, SSH access, network and Wi-Fi country where applicable. Boot Linux and confirm SSH access before starting this procedure.

2. With all power disconnected, assemble the Raspberry Pi and M.2 board, including the 40-pin connection, Pi-button control contact or cable, cooler, spacers and SSD/FPC where used. Connect one approved backup source with correct polarity and matching chemistry.

3. Reconnect a stable cable supply and keep it connected through installation, reboot and verification. Internet access is needed for the initial apt package installation. Do not begin a cable-loss test until the final check and charging verification have passed.

Files supplied in this package

The installer/ directory contains install_M2_UPS_PMH.py, payload/ with four unchanged software files from the V88 package, a payload checksum manifest, instructions and the simulated tests. It also contains the standalone M2_UPS_PMH_Installer_V01.zip for copying the installer independently of the manual. Keep the script and payload/ together.

The installer has no graphical interface. Run it from SSH with sudo and the distribution Python. A separate pip installation, virtual environment or desktop session is not required.

Run the first installation

Copy M2_UPS_PMH_Manual_V8F.zip to the Raspberry Pi user's home directory. These commands use the complete manual package; no separate installer archive needs to be extracted.

```
cd ~
unzip M2_UPS_PMH_Manual_V8F.zip
cd ~/M2_UPS_PMH_Manual_V8F/installer
sudo python3 install_M2_UPS_PMH.py
```

If using only the standalone installer archive instead, extract M2_UPS_PMH_Installer_V01.zip in the home directory and run the same Python command from ~/M2_UPS_PMH_Installer_V01. Choose one extraction route and retain that folder for subsequent checks and removal.

What happens during installation

The installer first verifies the payload checksums, Raspberry Pi model, operating-system version, systemd and the actual boot configuration path. It examines existing service units, overrides, known older UPS services and running UPS scripts, and checks RTC/UART settings in config.txt and included files. Conflicts stop installation rather than being silently overwritten.

It runs apt-get update, then installs python3-smbus, python3-serial, i2c-tools and util-linux. This is not a full operating-system upgrade. With --no-apt, those dependencies must already be present. Python imports are checked before changing the boot configuration or daemon files.

Where the I²C device and tools are already available, it reads the supply-mode status before continuing. A reported battery mode stops installation and asks for cable power. Absence of this early read on a clean card does not prove that the power supply is suitable; stable cable power is still required.

Next, it saves original files and service states, writes the managed boot/module settings, and installs the daemon, unit, optional monitor and updater. An active daemon is stopped only immediately around necessary daemon/unit replacement, then restarted promptly. An unchanged daemon is not restarted merely because the installer is run again.

Finally, it enables and starts M2_PMH_i2c.service, reserves the selected serial port from its Linux console/getty when UART setup is enabled, and attempts to load i2c-dev. It records installation state and whether boot changes require a reboot.

Restart only after installation succeeds

A successful install command means that the Linux files were installed and the service started. It is not the final hardware check. If REBOOT REQUIRED is printed, reboot while cable power remains connected. The installer does not reboot automatically.

```
sudo reboot
```

SSH disconnects during reboot. Reconnect after Linux has started, then use the check procedure below. If installation prints ERROR, retain the output and resolve that error before treating the installation as complete.

Boot configuration and installed files

The actual boot file is selected as /boot/firmware/config.txt where present, otherwise /boot/config.txt. The installer appends one marked block under [all], enabling I²C and adding the DS1307-compatible RTC overlay and uart0-pi5 overlay where equivalent supported settings are not already present. Other configuration is preserved.

```
dtparam=i2c_arm=on
dtoverlay=i2c-rtc,ds1307
dtoverlay=uart0-pi5
```

The last line is omitted with --no-uart. A recognized existing global uart0 setting may also make it unnecessary. These lines take effect on the next boot; the installer does not apply a live Device Tree overlay.

Installed path	Purpose
/usr/local/sbin/M2_PMH_i2c.py	Existing daemon v1.1; heartbeat, shutdown requests, CPU temperature and RTC synchronization.
/etc/systemd/system/M2_PMH_i2c.service	Starts the daemon automatically at boot; enabled by the installer.
/opt/M2_UPS_PMH/M2_PMH_status.py	Optional read-only status monitor; not a replacement for the daemon.
/opt/M2_UPS_PMH/M2_PM_UPS_BL01D.py	Existing updater, installed for separate use; never run by the installer.
/etc/modules-load.d/M2_UPS_PMH.conf	Loads i2c-dev during Linux startup.
/var/lib/M2_UPS_PMH/installer.json	Original service/getty states, file hashes, backup references and reboot marker.
/var/backups/M2_UPS_PMH/<UTC timestamp>/	Original versions of files changed by this installation.

UART and RTC behavior

On Raspberry Pi 5, uart0-pi5 enables the 40-pin header UART on GPIO14/15 as /dev/ttyAMA0, without RTS/CTS. A console targeting ttyAMA0 is removed from cmdline.txt and its serial getty is masked. An alias is treated the same way only if it resolves to ttyAMA0. The separate ttyAMA10 debug UART is retained.

The M.2 RTC is identified by rtc-ds1307 and bus/address 1-0068 rather than a fixed rtc1 number. The unchanged daemon waits for confirmed NTP, writes Linux UTC to this RTC and repeats hourly. The installer itself does not issue a forced clock write or serial transfer.

Verify after reboot

From the same extracted package, run:

```
cd ~/M2_UPS_PMH_Manual_V8F/installer
sudo python3 install_M2_UPS_PMH.py --check
```

The check does not edit configuration. It tests for /dev/i2c-1, performs three command-bank reads at address 0x6B, register 0x00, and reads the firmware identifier at status address 0x69, offsets 0x38–0x39. It does not scan every bus address, acknowledge a shutdown request or transmit data through the UART.

It also finds the registered M.2 RTC by its kernel name, checks that /dev/ttyAMA0 exists unless UART setup was excluded, verifies that the shutdown service is active, and looks for a recent successful heartbeat from its current process within the last 90 seconds. RTC registration is checked without setting its time. UART existence does not prove the complete serial wiring or bootloader transfer.

NTP synchronization is reported separately. A pending NTP state is informational and does not by itself fail the check: the daemon continues its heartbeat and waits before writing the RTC. A missing interface, absent RTC, failed I²C read, pending shutdown request, inactive service or missing recent successful heartbeat produces NOT READY and a failed result.

Result / exit code	Meaning
CHECK PASSED / 0	All implemented checks passed. Continue with charging checks and manual commissioning tests.
CHECK NOT PASSED / 2	One or more readiness checks failed. Save the output and resolve the reported cause.
ERROR / 1	Installer precondition, command or file operation failed. Inspect the error and backups.
Interrupted / 130	The command was interrupted. Check state before retrying.

Save the first card test result

```
sudo python3 install_M2_UPS_PMH.py --check 2>&1 | tee M2_install_check.txt
sudo journalctl -u M2_PMH_i2c.service -b -n 25 --no-pager
timedatectl status
```

Retain M2_install_check.txt with the firmware version and test observations. The file captures the text result; a shell pipeline ending in tee does not preserve the Python exit code by default. For an automated pass/fail decision, run --check directly or use shell pipefail.

Before testing a power outage, confirm increasing successful-read totals, adequate battery charge and the intended normal/slide-switch settings. CHECK PASSED does not perform or certify a shutdown, restart, charging-current or battery-capacity test. Continue with installation Steps 7–10 and the Commissioning checks chapter.

Options and repeated installation

Option	Effect
No option	Install/configure Linux, dependencies and service; keep firmware and battery settings unchanged.
--dry-run	Inspect prerequisites and print the plan; no apt downloads, configuration writes or service changes. Still requires sudo and the supported Pi/OS.
--no-apt	Skip apt commands; all dependencies must already be installed.
--no-uart	Install Linux UPS/RTC support while retaining serial configuration and consoles. Reuse the same choice on subsequent installs.
--check	Check the running installation after reboot; cannot be combined with installation options.
--uninstall	Restore backed-up files and original service/getty states; cannot be combined with installation options.
--help	Show command usage and exit.

Installation options may be combined where appropriate, for example --no-apt --no-uart. To change the saved UART choice, first uninstall under cable power, then install with the new choice. The installer does not merge an existing custom service or remove site-specific overrides automatically.

Rerun without duplicating configuration

```
sudo python3 install_M2_UPS_PMH.py
```

A normal rerun of the same package retains the first backup set, preserves one managed boot block and skips replacement of unchanged files. If a managed file was edited outside the installer, it stops and reports that file rather than overwriting the edit. Back up that change and review the original installation before continuing.

Uninstall and restore previous files

Keep cable power connected: uninstalling can remove the Linux shutdown protection. The following command restores the original files and service/getty state recorded for this installation. Dependencies and backup copies are retained.

```
sudo python3 install_M2_UPS_PMH.py --uninstall
sudo reboot
```

Before changing anything, uninstall checks file hashes and backup availability. A changed managed file causes a stop so that later user edits are not lost. After removal, reboot to apply the restored boot configuration. If a previous daemon was active/enabled, its recorded state is restored; otherwise this service is stopped/disabled.

Conflicts and installation failure

The installer stops for an unsupported Raspberry Pi or OS, missing/inconsistent payload files, unexpected service unit location, custom service content, drop-in overrides, another detected UPS daemon, known enabled older UPS services, or conflicting RTC/UART configuration. Included boot files are inspected; ambiguous, missing or recursive includes are rejected. Keep the error output and review the existing configuration instead of deleting it blindly.

If an ordinary error or Ctrl+C is caught during file installation, the installer restores files changed in that invocation and attempts to restore the previous service state. Apt package changes are retained. Original files remain in the backup directory for inspection.

A power cut or forced process termination cannot execute that rollback. The recorded state and backups support recovery, but an interrupted write may require manual review because file hashes no longer match the expected installed version. Keep cable power present, inspect the files, and use --uninstall before starting a fresh installation when restoration is unambiguous. Never treat an interrupted install as a verified installation.

Diagnose a first boot that is not ready

Missing I²C, RTC or UART before the requested reboot is expected; reboot and check again. If the M.2 is waiting in bootloader mode, its application banks are unavailable: complete the separate firmware recovery procedure before expecting the Linux check to pass. Failed apt commands require network/repository repair before retrying.

```
sudo journalctl -u M2_PMH_i2c.service -b -n 30 --no-pager
sudo journalctl -k -b --no-pager | grep -Ei "rtc|i2c|SDA|probe"
ls -l /dev/i2c-1 /dev/rtc* /dev/ttyAMA0
sudo systemctl cat M2_PMH_i2c.service
```

Installer questions

Does it update the MCU? No. It copies the Python updater for later use but never runs it, sends a bootloader request or programs a HEX file. Firmware update and interrupted-transfer recovery remain in their own chapters.

Does it choose the battery type or ON/OFF mode? No. Chemistry, charging current, runtime and switch mode remain unchanged. Verify them separately against the fitted hardware. Raspberry Pi EEPROM, boot order, PCIe/SSD configuration, Wi-Fi, network and user accounts are not changed.

What does the simulated validation cover? Fifteen tests cover installation/reinstallation, checks, restoration, configuration conflicts, payload integrity, service/apt failure, changed-file protection, UART aliases and I²C errors/timeouts. They also check that firmware/EEPROM programming and automatic reboot commands are not executed. Hardware acceptance requires the clean-card trial and subsequent commissioning cycles.

UPS operation and File Safe Shutdown

Cable powered operation

With a valid external source present, the controller recognizes cable mode and manages battery charging. Charging may start after source qualification and may be reduced when available input power is limited. A charger-enabled flag does not by itself prove that a particular charging current is flowing.

Loss of external power

When external power is removed, the UPS changes to battery operation. With runtime value 0x01, timed FSSD begins after approximately 70 seconds. The normal shutdown grace period is approximately 30 seconds. These are separate intervals.

The Linux service must receive the shutdown request and perform an orderly operating-system shutdown. The controller then completes its power-control sequence. Do not unplug the battery during this sequence.

Event	Expected sequence
Cable removed	Battery supports the running Raspberry Pi and SSD.
Runtime limit reached	Controller initiates FSSD; Linux begins shutdown.
Grace interval completed	Controller finalizes shutdown and power control.
Cable returns before FSSD	Controller qualifies the source and returns to cable operation.
Cable returns during FSSD	Shutdown completes; automatic restart requires the normal-mode return sequence.
Cable returns after shutdown	Normal UPS mode permits automatic restart when the source is accepted.

Cable return during the last seconds of FSSD is a specific commissioning case. Test it together with repeated cycles; a single successful restart does not establish reliable behavior over multiple transitions.

Other shutdown causes

Low battery, a configured switch action or a manual FSSD request can initiate shutdown before the timed limit. An unlimited runtime setting removes the elapsed-time limit only; it does not disable battery protection or manual shutdown.

Intelligent mobile power bank and slide switch

The optional slide switch provides controlled mobile operation and an OFF state with File Safe Shutdown. Enable it only on an assembly with the supported switch wiring and Raspberry Pi button-control connection.

Enable switch mode

```
sudo i2cset -y 1 0x6b 0x13 0xaa b
sleep 3
sudo i2cget -y 1 0x6b 0x13 b
sudo i2cget -y 1 0x6b 0x01 b
```

Expected values are 0xAA for switch mode and 0xFF for the runtime register. The firmware couples switch mode to unlimited elapsed battery runtime. Low-battery protection remains active.

Return to normal timed operation

```
sudo i2cset -y 1 0x6b 0x13 0x00 b
sleep 3
sudo i2cget -y 1 0x6b 0x13 b
sudo i2cget -y 1 0x6b 0x01 b
```

Expected values are 0x00 and 0x01. Read back both settings after changing mode.

Raspberry Pi EEPROM prerequisite

For OFF to remain effective when USB-C power is connected directly to the Pi, configure the Pi EEPROM with the following settings. Preserve the existing BOOT_ORDER and unrelated settings. This is a Raspberry Pi configuration operation; the M.2 controller does not write its EEPROM.

```
sudo rpi-eeprom-config --edit

POWER_OFF_ON_HALT=1
WAIT_FOR_POWER_BUTTON=1
```

Apply the EEPROM change using the Raspberry Pi tool instructions and the required reboot before testing. Direct USB-C power remains physically present in standby, and the Raspberry Pi power button can independently affect the Pi.

While ON, cable insertion should change the power source without an unnecessary restart. Switch OFF initiates shutdown. After a manual FSSD with the slide still ON, move the slide OFF for at least two seconds, then ON to request a new start.

System LEDs and user controls

Indicators are listed from top to bottom in the supplied board view: ACT nearest the SSD end, EPR nearest the front controls. EPR is fitted on the Advanced model.

Indicator	Meaning in normal operation
ACT	M.2 SSD activity indication.
M.2	M.2 supply indication.
USR	User-programmable indication or mapped system event.
SYS	System power-mode heartbeat; cadence described below.
BAT	Battery-level indication; interpret with battery measurements.
CHG	Battery charging indication; verify actual current when diagnosing charging.
TMR	Scheduler or timer indication when configured.
FAN	Raspberry Pi official cooler activity indication when fan-status information is supplied by software.
TMP	Raspberry Pi core temperature indication when temperature information is supplied by software.
EPR	Advanced external power status, where fitted.

SYS heartbeat cadence

The normal SYS indication changes approximately every 0.5 seconds on cable power and every 1.5 seconds on battery. Startup and reset sequences temporarily own the LEDs. Bootloader programming uses a separate walking indication, advancing once per two accepted HEX records; its rate follows transfer activity.

Buttons

FSSD is the user shutdown control during normal operation. M.2 RESET resets the UPS MCU and is not a substitute for orderly Linux shutdown. The Raspberry Pi 5 button is a separate physical control, also operated by the UPS transistor interface.

Serial diagnostics

The default local diagnostic speed is 115200 baud, 8 data bits, no parity and 1 stop bit. Use a 3.3 V logic-level adapter and common ground.

Useful messages include the firmware startup banner, Cable Powered or Backup Powered transitions, charger events and FSSD start/finalization. I²C counters continue increasing while communication is active. Counter wraparound can occur and does not alone indicate a reset.

A silent terminal does not prove that the MCU has stopped. Verify baud rate, cable and terminal settings, then compare LED activity and I²C responsiveness. Save the last complete output before resetting hardware.

RTC operation and time synchronization

The UPS contains its own hardware real-time clock, separate from the Raspberry Pi 5 RTC. The board can also provide backup supply to the Pi RTC through a dedicated cable. These are distinct functions; the auxiliary RTC supply cable does not replace software configuration of the UPS clock.

Raspberry Pi RTC backup from the main battery

The **M.2 UPS PMH** can supply the Raspberry Pi 5 RTC backup input from its main backup battery through the designated RTC supply cable. This uses the same larger battery that provides UPS backup; no separate battery is needed on the Raspberry Pi RTC connector when this connection is fitted.

When external cable power is lost, the M.2 battery continues to support the Raspberry Pi RTC backup supply. The clock can therefore retain its time after Linux has shut down and the Raspberry Pi main operating supply is off, provided the battery remains connected, has sufficient charge and the RTC backup path remains powered. This connection powers the clock backup domain; it does not keep Linux running.

A dedicated RTC backup cable is available from Pi Modules Technologies. It connects the M.2 main-battery backup supply to the dedicated RTC power connector on Raspberry Pi 5. Observe the specified connector orientation and polarity. If the cable is absent or the main battery is removed or depleted, battery-backed time retention cannot be assumed. This is separate from the UPS controller's own RTC and its DS1307-compatible I²C interface.

Identify the RTC device

```
ls -l /dev/rtc*
for d in /sys/class/rtc/rtc*; do
  echo "$d"
  cat "$d/name"
done
```

The UPS DS1307-compatible interface uses I²C address 0x68 and is commonly /dev/rtc1. Device numbering is not guaranteed; identify the rtc-ds1307 1-0068 device before selecting it.

Read and set the UPS clock

```
timedatectl status
sudo hwclock --show --rtc=/dev/rtc1
```

Once Linux time is correct and synchronized, a manual write can be tested with the following command. Replace rtc1 if the verified device has another name. Capture UART output during the test.

```
sudo hwclock --systohc --utc --rtc=/dev/rtc1
sleep 2
sudo hwclock --show --rtc=/dev/rtc1
```

Linux may display local time while firmware diagnostics display UTC. Compare timezone-aware values before diagnosing an apparent one-hour offset.

Retention and recovery

Ordinary Raspberry Pi restart and slide OFF/ON retain the UPS RTC while the controller remains powered. Factory defaults restore a reference date; confirmed NTP synchronization then restores current UTC time.

If the driver failed to bind during boot, inspect the kernel log first. A manual bind is appropriate only when the configured I²C device 1-0068 exists and has no bound driver:

```
sudo journalctl -k -b --no-pager | tail -n 60
echo 1-0068 | sudo tee /sys/bus/i2c/drivers/rtc-ds1307/bind
```

Cooling and enclosure integration

Raspberry Pi official cooler

The HAT is designed around the Raspberry Pi 5 official cooler. The central board opening provides an air path above the cooler and helps cool the assembled SSD. The fan remains connected to and controlled by the Raspberry Pi.

Fit the cooler using its Raspberry Pi instructions before placing the HAT above it. Route the fan cable without trapping it between boards. Check that neither the SSD nor a battery lead obstructs the fan or ventilation opening.

Temperature and fan indications

The daemon supplies Raspberry Pi CPU temperature as a packed BCD byte every five seconds. FAN and TMP are separate indicators. The official cooler remains controlled by Raspberry Pi; this package does not configure fan cooling profiles.

```
cat /sys/class/thermal/thermal_zone0/temp
vcgencmd measure_temp
vcgencmd get_throttled
```

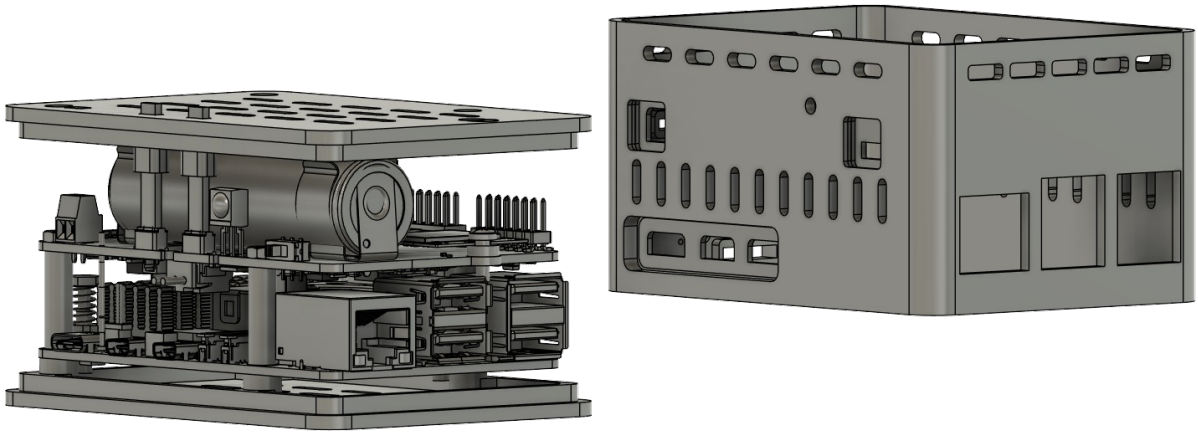
The sysfs temperature value is normally expressed in thousandths of a degree Celsius. Record temperature under the intended enclosure and workload, with battery charging enabled, rather than testing only at idle.

Case compatibility

A dedicated enclosure is available for the Raspberry Pi 5 and **M.2 UPS PMH** assembly, allowing both boards to be housed together as one complete system. Choose the enclosure configuration to match the Standard or Advanced model and the fitted SSD, battery and cooler. Check connector and button access when assembling the enclosure.

Confirm the complete mechanical assembly, not just the HAT outline. The enclosure must accommodate the chosen SSD, battery holder, access to FSSD and RESET, optional switch travel and cable entry. Keep the board underside insulated from metal panels.

An enclosure can change both thermal behavior and button access. Repeat the charging, cable-loss and restart tests after the final enclosure is fitted.



Dedicated enclosure for Raspberry Pi 5 and M.2 UPS PMH — V601 with buttons extending 2 mm above the top cover.

Application integration

Time and event data

Keep the UPS RTC in UTC. Applications can read power status, voltage, temperature and system information through the register reference at the end of this manual.

The bundled daemon performs heartbeat, shutdown handling, temperature reporting and RTC synchronization. It does not configure scheduled wake events or deliver email notifications.

Interactions with switch and UPS shutdown

A deliberate switch OFF state and battery protection take priority in the power-control design. Configure one shutdown owner and avoid competing applications changing command-state or power-mode settings.

User applications and notifications

Applications may use monitoring data to record power events or send notifications. Email broadcasting described in the prospectus requires a host application and network configuration; the reference heartbeat daemon v1.1 does not implement email delivery.

Safe application design

Let the system service own the shutdown handshake and RTC synchronization. Avoid multiple programs writing the command-state register. Keep application reads bounded and handle I^2C errors without retry loops that monopolize the bus.

If application data must be flushed during FSSD, integrate that work with normal systemd shutdown ordering and test its worst-case completion time. A file-safe power sequence depends on the actual operating system and storage behavior.

Commission the intended workload

Test event logging, storage writes and the expected network loss together with UPS operation. A passing idle test does not establish that a high-write workload completes within the configured grace interval.

Programming interface and I²C address groups

Examples use Linux bus 1 and the default address group. Addresses and offsets are hexadecimal. The appendix lists the V88 field layout, access types, encoding and reserved locations.

Address group	Addresses
Default	0x68–0x6F
Alternate 1	0x58–0x5F
Alternate 2	0x48–0x4F
Reduced no RTC group	0x69 and 0x6B

A scan showing several consecutive addresses can represent register banks of the same controller. It does not imply that eight independent physical devices are fitted. A device claimed by a Linux driver may appear as UU.

Default access	Purpose	Access
0x68	DS1307-compatible RTC	Linux RTC driver
0x69	Status bank	Read; host temperature write at 0x1A
0x6A	Internal RTC structure	Reserved; use 0x68
0x6B	Command bank	Read / described writes
0x6C–0x6E	Reserved banks	Leave unchanged
0x6F	Diagnostic snapshot	Read-only, 16 bytes

Non-destructive configuration checks

```
sudo i2cget -y 1 0x6b 0x13 b
sudo i2cget -y 1 0x6b 0x01 b
sudo journalctl -u M2_PMH_i2c.service -b -n 10 --no-pager
```

Use the released API definitions for all other writes. Arbitrary register writes can change power behavior, initiate shutdown or alter persistent settings. Do not force raw I²C access to an RTC address already owned by the kernel driver.

The complete byte layout is in the Register and variable reference at the end of this manual and in reference/REGISTER_MAP_V88.csv. Use the described fields and leave reserved locations unchanged.

Module status registers and monitoring

Status information uses the default bank at 0x69. Multi-byte values are little-endian. Apart from the host-supplied CPU temperature byte at 0x1A, status fields are read-only to user applications.

Monitoring functions

Information	How it should be interpreted
Power mode	Current accepted source state; correlate with cable/battery transition messages.
Battery voltage	A measurement, not a universal percentage. Chemistry, current and temperature affect interpretation.
Supply voltage and current	Use the correct channel and sign convention. Advanced inputs add monitoring points.
Charger state	Separate commanded enable, charging activity and measured current.
CPU temperature	Host-provided telemetry in the reference daemon, rather than an independent Pi CPU sensor inside the MCU.
System and restart reasons	Read with the matching firmware definitions; preserve the raw value and timestamp.

Heartbeat diagnostics

```
sudo journalctl -u M2_PMH_i2c.service -b -n 15 --no-pager
systemctl show M2_PMH_i2c.service \
  -p ActiveState -p NRestarts -p ExecMainStartTimestamp
```

Successful-read totals should increase while the host is running. Error totals are cumulative for that daemon process. NRestarts counts systemd automatic restarts, not every possible manual stop/start; compare the process start timestamp and Linux uptime as well.

Counter wraparound

MCU diagnostic counters can wrap at their declared width. A count falling from a high value to a small value is not by itself proof of a reset. A new startup banner, reset-cause report and reinitialized state provide stronger evidence.

Hardware RTC register interface

At the default address group, Linux accesses the UPS clock through the DS1307-compatible bank at 0x68. The date/time bytes use binary-coded decimal (BCD). Prefer the kernel RTC interface over direct register writes.

Offset	Date or time field	Encoding
0x00	Seconds	BCD seconds; use the supported clock control behavior.
0x01	Minutes	BCD minutes.
0x02	Hours	BCD hours; the reference system uses 24 hour time.
0x03	Day of week	Driver and firmware convention must agree.
0x04	Day of month	BCD calendar day.
0x05	Month	BCD month.
0x06	Year	Two-digit BCD year in the firmware-supported range.

BCD example

A minute byte of 0x27 represents decimal 27 minutes, not decimal 39. A complete date write must contain a valid combination of fields. Use hwclock or the supplied daemon so the Linux driver applies the intended format.

Commit behavior

The MCU captures incoming date/time bytes and commits a completed request outside the transfer handler. Serial diagnostics show QUEUED and COMMIT_END. Confirm the new time by reading it back.

Distinguish the two RTCs

The Raspberry Pi 5 RTC and the UPS RTC have separate Linux device entries and separate functions. Verify /sys/class/rtc names before setting a clock. A generic hwclock command without --rtc can select the wrong one.

Module command registers

The default command bank is 0x6B. Runtime, serial, switch and supported configuration fields are listed in the final register reference. The shutdown handshake at 0x00 is owned by the Linux daemon.

Offset	Function	Established behavior
0x00	Host state and shutdown handshake	Read by the daemon. Shutdown requests 0xCC and 0x77 are recognized by v1.1; an acknowledgement is followed by Linux poweroff.
0x01	Battery runtime control	0x01 gives the default of about 70 seconds before timed FSSD. 0xFF disables the elapsed-time limit.
0x13	Slide switch enable	0xAA enables switch mode and couples runtime to 0xFF; 0x00 returns to normal mode and runtime 0x01.

Read back after configuration

```
sudo i2cget -y 1 0x6b 0x01 b
sudo i2cget -y 1 0x6b 0x13 b
```

Do not treat 0x01 as a literal one-second runtime, or 0xFF as a battery-protection bypass. The timed limit, FSSD grace interval, switch behavior and low-battery decisions are separate parts of the operating sequence.

Persistent and runtime state

Some settings are retained by the controller while others are runtime handshakes or counters. Do not assume every write is persistent or that reading a value back proves it has been saved. Use the documented mode-setting procedure and verify behavior after a controlled restart.

Reserved fields

Reserved locations occupy bytes in the bank but do not authorize a hardware action. Leave them unchanged. A field name alone does not establish that a peripheral is fitted or active.

Bootloader entry and factory defaults change program flow or persistent settings. Use the dedicated firmware-update procedure. RESET plus FSSD selects only the bootloader; it does not erase configuration or select factory defaults.

Bootloader entry and recovery

The V88 bootloader loads and verifies the UPS application through the Raspberry Pi serial port. Standard and Advanced use the same procedure. Keep stable cable power connected. The KEY header is not required. RESET plus FSSD opens the bootloader directly; there is no Hardware Command Mode menu and no button-selected Factory Defaults.

Two ways to enter the bootloader

Use either the two M.2 buttons or the software command while the application is running. Before starting, have the matching HEX file and upload command ready. Close other programs using the loader UART and stop the optional status monitor. Keep the normal shutdown service installed and enabled.

Entry using RESET and FSSD

1. Connect a suitable cable supply and keep Raspberry Pi running with SSH available.

2. Press and hold M.2 RESET. While holding RESET, press and hold FSSD.

3. Release RESET while continuing to hold FSSD. After approximately one second, release FSSD. Do not keep it held for several seconds; a key not released within the startup release window cancels the request.

4. The controller checks cable power and enters the bootloader. Tested V83 and later entry was approximately three seconds from the application entry message; this is an observation, not a guaranteed startup time.

5. Start the prepared upload within 15 seconds of actual bootloader entry. Do not make a second FSSD press. This entry path does not deliberately reset Raspberry Pi.

Entry using an SSH command

The simplest software procedure is the combined entry-and-update command in the next chapter. To enter the bootloader alone, write 0xBB to command address 0x6B, register 0x00:

```
sudo i2cset -y 1 0x6b 0x00 0xbb b
```

The application must be running and must have accepted cable power. The write restarts the M.2 controller into its protected loader. It does not intentionally press the Raspberry Pi power button. After this command use the manual-entry upload command with -j, within the 15-second window.

The 15 second entry window

The limit applies before the first accepted application Flash write. If no write starts, the bootloader returns to the intact application through a resume path. Serial probes, noise, malformed records and incomplete lines do not extend this window. Application initialization follows the timeout, so the visible return can take longer than 15 seconds.

Once an actual write begins, the bootloader saves a persistent incomplete-update marker before modifying the application. The 15-second exit is then disabled. The loader remains in recovery until a complete transfer, successful Flash verification and end-of-file command finish the update. It will not start a partly written application.

Bootloader LED indication

USR, SYS, BAT, TMR, FAN and TMP form a walking sequence during programming. V88 advance this sequence once per two HEX records, including an all-off step between cycles. It indicates transfer activity, not a calibrated percentage. Use the Python progress and final CRC result to determine completion. ACT, M.2, CHG and EPR are not part of this programming animation.

Interrupted transfer and power loss

If a transfer is interrupted after writing begins, keep or restore cable power and send the entire firmware again from the beginning. No buttons or new I²C loader request are needed while recovery is active. The same recovery state survives an MCU reset and loss of power. A new valid image can replace the interrupted image; verification applies to the image selected for the new attempt.

If interruption happens before the first application write, the previous image remains intact. In that case the normal 15-second timeout still applies; enter the bootloader again if necessary. A correct update clears the recovery condition and resumes the application.

Automatic Raspberry Pi startup during recovery

In V88, an incomplete update combined with an MCU power-up or brownout reset invokes one Pi power-button sequence after the supply paths are enabled: 0.5 seconds released, 6 seconds pressed, 2.25 seconds released, 0.5 seconds pressed, then released. The initial long press establishes an OFF state before the final wake pulse; a Pi already booting during this recovery can therefore restart.

This recovery sequence takes about 9.35 seconds before the loader wait. Allow another 60–90 seconds for Linux and SSH to become available. Recovery has no 15-second upload deadline after writing has already begun, so there is time for Raspberry Pi to boot. Run the recovery upload command once SSH returns.

The sequence is not used for ordinary button or I²C loader entry, MCU RESET alone, watchdog or software reset, a completed-update resume, or startup with an intact application. If only the MCU is reset while Pi stays running, recovery leaves its power button released.

Factory defaults

RESET plus FSSD no longer selects factory defaults. The software maintenance request at command register 0x00 remains available as code 0xDD, subject to the documented application conditions. Use it only when restoring configuration is intended. It is separate from bootloader recovery and can restore the RTC reference date. Routine firmware programming does not require factory defaults.

Local and remote firmware updates

Application and protected bootloader

The application provides UPS operation. The protected bootloader receives application updates and recovery transfers. The Python updater excludes the loader, configuration and activation-marker regions from application writes. Consequently, uploading a combined HEX through Python does not replace its protected loader. A newer application banner does not prove that a newer loader is installed.

Initial installation or loader replacement

Program firmware/M2_PMH.hex through ICSP using the board programming connection and a compatible programmer. This combined V88 image contains the application and matching 38400-bit/s bootloader. The separate bootloader directory is for maintenance; do not program its standalone HEX as the complete UPS application. Follow the programmer procedure for the fitted board and power arrangement.

The supplied V88 package contains matching application and protected-loader images. Installing an application image through the serial updater does not replace the protected loader; use the documented programming procedure when replacing that loader.

Serial connection and preparation

The supplied V88 loader uses 38400 bit/s, 8 data bits, no parity and 1 stop bit. The updater default is still 9600 for compatibility, so always supply -b 38400. Application diagnostic UART speed, normally 115200, is a different setting and does not change loader speed. Select the UART actually wired to the board; the tested assembly uses /dev/ttyAMA0.

```
sudo apt install -y python3-serial python3-smbus
cd ~/M2_UPS_PMH_Manual_V8F/firmware
python3 ../updater/M2_PM_UPS_BL01D.py -G -f M2_PMH.hex
```

The -G command validates Intel HEX locally without entering BL or writing hardware. Retain a known complete copy of the HEX and use SHA256SUMS.txt to identify the supplied files. Close serial terminals on the loader UART. The read-only monitor should be stopped; it is not needed for programming.

Automatic entry and upload from SSH

Use this command while the UPS application is running under accepted cable power. Omitting -j allows Python to send the I²C bootloader request automatically:

```
sudo python3 ../updater/M2_PM_UPS_BL01D.py -v -i -k \
  -p /dev/ttyAMA0 -b 38400 -f M2_PMH.hex
```

Upload after buttons or during recovery

Use this command after RESET plus FSSD, after a separate 0xBB I²C command, or when an interrupted update has already left the board waiting in BL. It does not depend on the UPS application responding to I²C:

```
sudo python3 ../updater/M2_PM_UPS_BL01D.py -v -i -j -k \
  -p /dev/ttyAMA0 -b 38400 -f M2_PMH.hex
```

Meaning of the upload options

-v enables progress information; -i skips the application version read; -j skips I²C bootloader entry; -k skips the extra I²C reset after programming. The serial loader completes its own verified exit. None of these options disables Flash readback or CRC32 verification. Keep -j for a board already in BL, and omit it only when requesting entry from a running application.

Progress and completion

Tested transfers of 6356 records and 76245 application bytes took about two minutes at 38400, compared with about six minutes at 9600. Duration depends on image size, Flash operations and host timing. Progress is reported roughly every 5 percent; it is not printed for every record. Do not interrupt a valid transfer merely because output is temporarily quiet.

The updater verifies 26 Flash ranges for the tested image and sends the final end-of-file command only after all ranges match. The record count and range count can differ with another image. Successful completion is explicitly reported as:

```
Transfer completed; programmed application Flash CRC32 verified
```

An early ACK timeout or NAK may be recovered during connection negotiation. Flash readback available confirms a valid protocol reply; it is not proof of completed programming. Treat the update as successful only after the final CRC-verified completion message. A timeout or CRC mismatch during data transfer requires investigation and a complete retry.

Repeat after an interruption

Restore stable cable power, allow Pi to start, and use the command with -i -j -k at 38400. The transfer starts at record one; it does not continue at the interrupted percentage. Do not issue factory defaults to repair an incomplete image. If the protected bootloader itself is damaged, use ICSP; application recovery cannot reconstruct it.

Checks after an update

1. Confirm the final CRC-verified completion message and the expected application version in the startup banner.

2. Confirm the application resumes and successful heartbeat reads increase. I²C is unavailable while BL runs, so the daemon may record errors during programming; check that new errors stop after application return.

3. Read back switch mode and runtime. Check RTC synchronization, charging and one orderly shutdown/return cycle before further cable-loss testing.

```
sudo systemctl start M2_PMH_i2c.service
sudo journalctl -u M2_PMH_i2c.service -b -n 15 --no-pager
uptime
```

If you stopped the service manually, restart it only after the application has returned. Do not force raw access to a kernel-owned RTC device and do not repeatedly trigger I²C entry while the board is already in BL.

Troubleshooting

Symptom	Checks and next action
SSH unavailable; Pi LED red	Check whether the Pi has shut down or is waiting for its power button. Inspect UART and supply state before assuming a crash.
Repeated MCU startup banners	Save the log from before the restart. Record firmware and reset-cause fields. Check power and escalate with the complete capture.
No serial messages	Check baud rate, logic levels, adapter and common ground. Compare LEDs and I²C activity.
RTC device missing	Inspect /dev/rtc*, RTC names and kernel probe errors. Confirm address selection and driver binding.
RTC write times out	Stop repeated writes. Capture kernel and daemon logs plus UART. Check whether heartbeat still responds.
Battery does not charge	Confirm cable mode, battery chemistry, protection state and available supply margin. Check measured current as well as CHG indication.
Cable/battery mode oscillates	Check supply, USB-C lead, connectors and load. Capture transitions with charging enabled.
No timed FSSD	Read switch mode and runtime. Switch mode 0xAA normally sets runtime 0xFF.
No restart after cable return	Check mode, physical switch, completion of FSSD and Pi EEPROM standby settings. Test return during FSSD separately.
One I²C error at startup	Observe later totals. A single historical error differs from an increasing error count; preserve both for diagnosis.

Collect a useful report

```
uptime
sudo journalctl -u M2_PMH_i2c.service -b -n 30 --no-pager
sudo journalctl -k -b --no-pager | tail -n 60
sudo i2cget -y 1 0x6b 0x01 b
sudo i2cget -y 1 0x6b 0x13 b
```

Include a complete serial capture, not only the new startup banner. Note whether a reset was manual, the cable-return timing and the exact battery and supply used.

Commissioning checks

Expected results

Run the following checks after installation and after changing battery, supply, storage or enclosure. Keep the serial capture and service journal for diagnosis.

Test	Expected result
Cable operation	Stable source recognition; charging starts when conditions permit.
Cable removed	Pi and SSD continue on battery; no unintended reboot.
Timed FSSD	About 70 s with runtime 0x01, followed by about 30 s grace.
Cable return during FSSD	Shutdown completes, then normal-mode startup follows.
Cable return after shutdown	Startup after the cable source is accepted.
Repeated cycles	At least five consistent shutdown / return cycles.
Slide OFF / ON	OFF requests shutdown; ON permits startup.
RTC and service	Correct time after NTP; successful reads increase without growing error totals.
BL idle exit and recovery	No write: return after 15 s. Interrupted write: remain in BL across reset/power loss and accept a full verified retry.

Frequently asked questions

Answers to common installation, operation, recovery and firmware-update questions for Standard and Advanced assemblies.

What does the M.2 UPS and Power Management HAT do?

It combines an M-key PCIe interface with battery backup, power monitoring and File Safe Shutdown for Raspberry Pi 5. The UPS supports the Pi and SSD while Linux finishes its shutdown.

Which models are available?

Standard provides the core M.2 and UPS functions. Advanced adds the designated 5–32 V external input, passive PoE and additional application interfaces. The final appendix compares both assemblies.

Is it compatible with other Raspberry Pi models?

This product is designed for Raspberry Pi 5. Do not assume compatibility with another Raspberry Pi model because its GPIO connector fits.

Is installation completely plug and play?

Mechanical connection alone is not sufficient for File Safe Shutdown. Enable I²C, configure the RTC interface and install the supplied Python daemon and systemd service as described in the installation chapters.

Does the HAT need another USB-C cable?

In normal Standard operation, keep the supply connected to Raspberry Pi 5 USB-C. The HAT receives power through the 5 V GPIO power path and supplies backup power through that path during an outage.

Can higher external input voltages be applied to the Raspberry Pi header?

No. The Raspberry Pi header is a 5 V power path. The 5–32 V range applies only to the designated Advanced EPR input or approved passive PoE input.

Can I use another HAT above this board?

Only when the connector arrangement, mechanical clearance, I²C addresses and combined power budget permit it. Check the fitted header and SSD, cooler and battery clearance before stacking.

Will it fit an existing case?

A dedicated enclosure specially designed for the **M.2 UPS PMH** and Raspberry Pi 5 assembly is available. Choose the version matching the board and fitted battery, SSD and cooler. For other cases, verify the full assembly and connector access; the board footprint alone does not establish compatibility.

Which SSDs can be fitted?

The board accepts compatible PCIe M-key devices in 2230, 2242 and 2280 lengths. Mechanical fit does not prove driver or boot support. Validate the intended NVMe or other PCIe device separately.

What is the M.2 reference clock?

The interface provides a 32.768 kHz SUSCLK reference. This signal is separate from the DS1307-compatible RTC interface used by Linux.

What backup current is available?

The specification gives a total backup supply capability of 5.25 V at up to 5 A continuous, with backup also covering the SSD. Battery discharge capability, connectors, wiring and thermal conditions must support the actual load.

Can I connect the 18650 battery and cable battery together?

No. Use one backup source at a time. These are alternative connections; connecting cells in parallel without a suitable management circuit can cause uncontrolled current between them.

Which batteries are supported?

Only LiPo (L, 0x4C), Li-ion (I, 0x49), LiFePO4 (F, 0x46), and the designated 4000 F supercapacitor (C, 0x43) are supported. Use one source at a time with the matching charge profile and adequate discharge capability. The standard battery specification is a protected 3.7 V, 450 mAh, 15C LiPo.

Can I use a supercapacitor?

Use only the designated 4.2 V 4000 F assembly with its approved profile. The cycle-life comparison uses the FAQ reference of more than 20000 cycles. Published component temperature limits differ (+65°C or +70°C upper limit); see the Supercapacitor cycle life and temperature chapter. These are not ratings for the complete Raspberry Pi assembly.

Is the metallic battery holder insulated?

The supplied FAQ describes a silicone conformal coating on the holder. Inspect the coating and maintain clearance from exposed circuitry. Do not rely on the coating to tolerate damaged insulation, loose metal or assembly errors.

What is the maximum charging current?

Standard: 580 mA. Advanced: 1.1 A. Charging demand is adjusted dynamically to available cable power, starting at a low current. The maximum is not a guaranteed current for every battery or supply condition.

Why does charging start slowly or reduce its current?

The supply must power the Pi, SSD, peripherals and charger together. The controller reduces charging demand when source margin is limited and increases it as conditions permit. Check measured battery current as well as CHG indication.

Why can battery mode appear while the cable is connected?

An insufficient source or excessive voltage drop can make the cable path unacceptable under load. Check supply rating, USB-C lead, connectors, peripheral load and charging demand.

How quickly does backup take over?

The line-interactive transfer-time specification is a maximum of 50 microseconds. Advanced online power paths are specified at zero transfer delay.

What protects the battery and power path?

The supplied documentation describes battery monitoring and cut-off protection, a 5 A PPTC and controlled power switching, with additional input protections on the applicable model. Use a compatible protected battery and the correct polarity.

When does timed shutdown begin?

In normal mode, command register 0x01 defaults to 0x01 and timed FSSD begins after approximately 70 seconds on battery. A separate grace interval of approximately 30 seconds follows.

Can I extend battery runtime?

A runtime value of 0xFF removes the elapsed-time limit. For values 0x00–0xFE, the timed threshold is approximately 10 + 60 × value seconds. Low-battery protection remains active.

What happens if cable power returns during FSSD?

The shutdown sequence completes before the normal-mode restart sequence begins. Test return near the end of the grace interval and several successive cycles. In switch mode, the physical switch must permit ON.

How does the optional slide switch work?

Enable switch mode with 0xAA at 0x6B register 0x13; the runtime becomes 0xFF. OFF requests safe shutdown and holds the Pi off even when USB-C is connected, with the documented Pi EEPROM configuration and power-button connection. ON permits startup and cable/battery operation.

Which GPIO pins are required by the daemon?

Normal communication uses I²C and the shared ground; the daemon uses no extra Raspberry Pi GPIO. The optional 1-wire interface uses Raspberry Pi GPIO4 where implemented. Power control requires the documented Pi button connection.

What do the LEDs show?

The PCB row is ACT, M.2, USR, SYS, BAT, CHG, TMR, FAN, TMP and EPR. FAN and TMP are separate indicators. SYS normally flashes about every 0.5 s on cable and 1.5 s on battery. EPR is an Advanced indicator.

Can I monitor or control the UPS with scripts?

Yes. Use the released I²C API and the supplied daemon. The default banks include RTC at 0x68, status at 0x69 and commands at 0x6B. Do not write undocumented offsets or run a second shutdown daemon.

Are scheduler and event-triggered applications available?

The bundled daemon manages heartbeat, shutdown, temperature and RTC synchronization. It does not configure scheduler events or send email. Reserved scheduler fields are identified in the register appendix.

Does the UPS contain a separate DS1307 chip?

No. The UPS controller exposes its RTC through a DS1307-compatible I²C interface, with a 32.768 kHz clock source. Linux uses the rtc-ds1307 driver. This clock is separate from the Raspberry Pi 5 RTC.

Can the HAT back up the Raspberry Pi RTC too?

Yes. The **M.2 UPS PMH** main battery backs up the Raspberry Pi 5 RTC through the dedicated cable, including after power loss and Linux shutdown. No separate RTC battery is needed while this backup supply is available.

Why does the RTC display a different hour?

The daemon writes UTC after Linux confirms NTP synchronization and then hourly. Linux tools may display local time. Compare timezone-aware values before adjusting the clock; do not copy an incorrect Linux time into the UPS RTC.

Does a normal restart reset the UPS RTC?

Ordinary Raspberry Pi restart and switch OFF/ON should retain it while the UPS controller remains powered. Factory defaults deliberately restore a reference date; a later confirmed NTP synchronization restores current time.

Why does i2cdetect show UU at 0x68?

UU means a Linux kernel driver owns the address. It is expected when the RTC driver is bound. Read the clock through /dev/rtcN after verifying its name rather than forcing direct access to the claimed address.

What is the bootloader used for?

It accepts a complete application HEX through the Python updater, verifies actual Flash using CRC32, and protects recovery after interruption. The V88 loader uses 38400 bit/s. Python does not replace the protected loader; use ICSP when installing a new loader.

How do I enter the bootloader?

With cable power connected, hold M.2 RESET and FSSD, release RESET, then release FSSD after about one second. No second press or KEY jumper is required. Alternatively, use the automatic SSH update command without -j, or write 0xBB to 0x6B register 0x00. RESET plus FSSD does not select factory defaults.

Which external voltages can the Advanced model accept?

The designated EPR and passive PoE inputs are specified for 5–32 V DC. Check polarity, wiring, input current and source rating. The allowed input range alone does not make every solar panel suitable.

Can EPR and passive PoE be connected together?

The supplied FAQ specifies one of these external sources at a time. Use EPR or passive PoE, following the input wiring for the fitted Advanced assembly.

Can EPR and Raspberry Pi USB-C be connected together?

The supplied specification permits GPIO 5 V and EPR to be present together through the designed power paths, with source selection by the controller. Do not parallel supplies through arbitrary wiring.

Is passive PoE compatible with standard active PoE?

Do not assume it is. This is a dedicated passive DC arrangement, with no claim here of IEEE 802.3af/at/bt negotiation. Use a compatible injector and verify its voltage, pin assignment and polarity.

What should I capture when a test fails?

Save uptime, the daemon journal, kernel errors and the complete serial trace preceding the event. Record firmware, supply, battery, switch position and exact cable sequence. Note whether any reset was manual.

Which software files belong with this manual?

The ZIP contains this manual in PDF and Word, the daemon and systemd unit, a read-only monitor, the V88 register CSV, installation and update instructions, the 01D Python updater, compiled V88 combined firmware, and separate bootloader maintenance files. README.txt identifies the directories and supplied versions.

Does Standard need the Advanced KEY header for recovery?

No. Standard and Advanced both use M.2 RESET and FSSD. The Advanced KEY input is not used for bootloader entry.

Why must I release FSSD after RESET?

Release FSSD after about one second so startup can debounce the release and proceed into BL. There is no command-selection menu. Keeping FSSD held beyond the startup release deadline cancels the request.

Does entering BL stop Raspberry Pi 5?

Ordinary button or I²C bootloader entry does not intentionally press the Pi power button. During BL the UPS application and I²C are unavailable. V88 cold recovery after an incomplete write is different: it deliberately establishes Pi OFF, then wakes Pi so SSH can be used to retry the update.

Can I leave the loader after an interrupted Flash write?

Before the first application write, the 15-second idle exit can return to the intact image. After a write starts, the persistent marker keeps recovery active across MCU reset and complete power loss. Send the whole image again and wait for verified completion. The incomplete application is never intentionally executed.

Why do I get repeated ACK timeouts?

Check -b 38400, the selected UART, cable power and whether BL is active. A command using -j cannot enter BL by itself. If no write began and the 15-second window expired, request entry again. Close other programs using the same UART.

Can I change baud rate only in Python?

No. The loader and Python must use the same speed. V88 requires -b 38400. Replacing a 9600-bit/s loader requires ICSP; a routine application upload leaves the old loader in place.

Why does an interrupted transfer start from zero?

Recovery accepts a complete new image and rewrites it from the beginning. This avoids relying on partly programmed data. The final CRC check verifies the newly selected image.

Will Raspberry Pi start after a power failure during programming?

With the V88 protected loader installed, an incomplete-write marker plus a power-up or brownout reset triggers one Pi startup sequence. Allow roughly 9.35 seconds for that sequence and then Linux boot time. An MCU RESET alone does not invoke this sequence.

How do I know which bootloader is installed?

Keep the ICSP installation record and HEX hash. The application version banner and the PCB/BL identification byte 0x40 do not by themselves distinguish all loader revisions or serial speeds.

Does the 15 second limit interrupt a long upload?

No. It applies only before the first accepted application Flash write. Once writing starts, recovery waits for a complete verified transfer, including across MCU reset or power loss.

Technical specifications

Electrical and mechanical comparison of Standard and Advanced assemblies. Standard maximum charging current is 580 mA; Advanced maximum is 1.1 A. Ratings apply to the designated input, output and fitted assembly.

Feature	Standard	Advanced / Passive PoE
Raspberry Pi® 5 System Compatibility		
Compatible Raspberry Pi Models	Especially Designed for Raspberry Pi® 5	Especially Designed for Raspberry Pi® 5
HAT EEPROM	Not Exist	Not Exist
HAT Dimensions	85mm x 56mm	85mm x 56mm
Raspberry Pi® 5 System Interfaces		
Use I²C API for Raspberry Pi® 5 interaction User selectable addresses	GND, SDA0, SCL0 I²C Addresses 1: 68 69 6a 6b 6c 6d 6e 6f I²C Addresses 2: 58 59 5a 5b 5c 5d 5e 5f I²C Addresses 3: 48 49 4a 4b 4c 4d 4e 4f I²C Addresses 4: 69 6b	GND, SDA0, SCL0 I²C Addresses 1: 68 69 6a 6b 6c 6d 6e 6f I²C Addresses 2: 58 59 5a 5b 5c 5d 5e 5f I²C Addresses 3: 48 49 4a 4b 4c 4d 4e 4f I²C Addresses 4: 69 6b
Selectable use of Raspberry Pi® 5 RS232 serial0	GND, TXD0, RXD0 OFF(HiZ)	GND, TXD0, RXD0 OFF(HiZ)
Raspberry Pi® 5 RTC Powering	Offered via dedicated cable supplying Raspberry Pi RTC via UPS Battery	Offered via dedicated cable supplying Raspberry Pi RTC via UPS Battery
Selectable use of Raspberry Pi® GPIO	I²C only for interaction with	I²C only for interaction with GPIO_GEN4 (only if 1-wire is used)
Passive Cooling	4 layers PCB 2oz copper with enhanced ground and cooling planes, covered with huge number of thermal vias Cooling hole over Official Cooling FAN for air circulation	4 layers PCB 2oz copper with enhanced ground and cooling planes, covered with huge number of thermal vias Cooling hole over Official Cooling FAN for air circulation
Active Cooling	Hole Placed exactly above the Official Raspberry FAN Offers improved Cooling for the Raspberry Pi 5 as also SSD above	Hole Placed exactly above the Official Raspberry FAN Offers improved Cooling for the Raspberry Pi 5 as also SSD above
Dedicated Passive PoE on PCB	none	5V-32V DC with reverse polarity, 2 level ESD and PPTC fuse protection Continuously current measure over Passive PoE supply on 12V-32V

Feature	Standard	Advanced / Passive PoE
M2. Interface		
Supported M.2 SSD Interfaces	2230 – 2242 – 2280	2230 – 2242 – 2280
Supported M.2 SSD Height	3.2mm	3.2mm
Supported M.2 SSD Cooling	Via Official Raspberry Pi 5 Cooler Air Pass Hole	Via Official Raspberry Pi 5 Cooler Air Pass Hole
M.2 SSD Indicators	ACT, M.2 (PWR)	ACT, M.2 (PWR)
Supported Power Supply Current	3A	3A
Supported M.2 32.768 kHz Clock	Yes	Yes
UPS Functionality on M2. SSD	Yes	Yes
Power Monitoring		
UPS Type	Line Interactive on Raspberry Pi® 5V GPIO	Line Interactive on Raspberry Pi® 5V GPIO On-line on EPR On-line on PPoE
UPS Response time	Line Interactive Maximum 50 uS	Line Interactive Maximum 50 uS On-Line 0 uS
Automatic Restart on Cable Power Return	YES	YES
Raspberry Pi Battery Backup	5.25V@5A current continuous supply to Raspberry Pi via 5V GPIO Pins Power Backup cover also the M.2	5.25V@5A current continuous supply to Raspberry Pi via 5V GPIO Pins Power Backup cover also the M.2
Cable Power Input	Raspberry Pi® GPIO 5V	Raspberry Pi® GPIO 5V UPS External Power Input 5V-32V DC supplying Pi with 5V@5A Can be used both (GPIO 5V and EPR) at the same time (isolated with ZVD)
Cable Power Monitoring Point	Raspberry Pi® GPIO 5V	Raspberry Pi® GPIO 5V/EPR/PPoE
UPS Activation Powering Triggers/Thresholds	Proprietary Algorithm of Falling Power Peak Analysis Programmable by user Self-learning system	Proprietary Algorithm of Falling Power Peak Analysis Programmable by user Self-learning system

Feature	Standard	Advanced / Passive PoE
Cable Powering Reactivation	After 10s of continuously cable powering (without power spikes)	After 10s of continuously cable powering (without power spikes) on any cable power source (GPIO or External or Passive PoE)
Auxiliary 5V and 3V3 Battery Backed Supply on I/O Pins	none	5V@200 mA current and 3V3@200 mA continuous supplies on I/O Pin battery backed, with possibility to continuous supply auxiliary devices with Raspberry Pi disconnected
Power Back-up		
Total Back-up Power	5.25V 5A continuously Supply	5.25V 5A continuously Supply
Number Power Back-up Sources	Two (One at the time can be used)	Two (One at the time can be used)
Power Back-up Source Types	Cable Battery or 18650 Socket (One at the time can be used)	Cable Battery or 18650 Socket (One at the time can be used)
Supported Batteries Power Back-up		
Supported Batteries Chemistry	LiPO, LiFePO4, Li-Ion, Super Capacitor 4000F	LiPO (standard Supported with system), LiFePO4, Li-Ion, Super Capacitor 4000F
On bard XH-2.5mm Battery Socket (with mechanical reverse polarity protection)		
Default (standard)	PCM Protected LiPO 3.7V, 450 mAh 15C with silicone high current cables	PCM Protected LiPO 3.7V, 450 mAh 15C with silicone high current cables
Optional bigger Capacities Batteries	LiPo, Li-ion and LiFePO4 only; use the matching source profile.	LiPo, Li-ion and LiFePO4 only; use the matching source profile.
On bard 18650 Battery Holder (with electronic reverse polarity protection)		
18650 Support	Single 18650 cell: Li-ion, LiFePO4 or compatible LiPo only; use the matching source profile.	Single 18650 cell: Li-ion, LiFePO4 or compatible LiPo only; use the matching source profile.
Charging process		
LiPO/LiFePO4/Li-Ion/Super Capacitor	Automatic Selected: Full Charging Cycle Trickle Charging	Automatic Selected: Full Charging Cycle Trickle Charging
Maximum Charging Current	580mA	1.1A

Feature	Standard	Advanced / Passive PoE
Charging Current Setting	Yes, from 100mA to 580 mA Dynamically Decrease/Increase the charging current to appropriate level according to cable power available	Yes, from 100mA to 1.1 A Dynamically Decrease/Increase the charging current to appropriate level according to cable power available
Batteries Protection		
Standard LiPO 450 mAh	PCM and on-board cut-off protection system overcharge or over discharge, over voltage and under voltage PCB temperature monitoring (if battery is placed on PCB)	PCM and on-board cut-off protection system overcharge or over discharge, over voltage and under voltage PCB temperature monitoring (if battery is placed on PCB)
High-Capacity Li-Ion, LiPO, LiFePO4	PCM and on-board cut-off protection system overcharge or over discharge, over voltage and under voltage PCB temperature monitoring (if battery is placed on PCB)	PCM and on-board cut-off protection system overcharge or over discharge, over voltage and under voltage PCB temperature monitoring (if battery is placed on PCB)
Battery Electrical Isolation System	Battery is Electrically Isolated (however cable connected) until system start up for the first time and receive 5V from GPIO	Battery is Electrically Isolated (however cable connected) until system start up for the first time and receive 5V from GPIO (or Passive PoE or 5V0-32V from EXT
Optional	Slide ON/OFF switch (external or internal), OFF always with File Save shutdown functionality	Slide ON/OFF switch (external or internal), OFF always with File Save shutdown functionality
Independent of the Cable Battery Powering		
Direct Battery Powering (Intelligent Power Bank) with Internal/External ON/OFF Slide Switch	Optional ON/OFF Slide Switch with File Safe Shutdown functionality when switching to OFF (keep battery powering ON until system shutdown)	Optional ON/OFF Slide Switch with File Safe Shutdown functionality when switching to OFF (keep battery powering ON until system shutdown)
Additional Features		
Power Consumption Measure	Dedicated IC measuring Inbound/Outbound Current/Voltage/Power	Dedicated IC measuring Inbound/Outbound Current/Voltage/Power
ESD Protected True 5V0/3V3 1-wire interface	Not implemented	Directly connected to Raspberry Pi ® (if used only) GPIO04

Feature	Standard	Advanced / Passive PoE
Independent from Raspberry Pi ® 3V3 battery backed supply @200 mA With battery Back-up (Raspberry Pi® 5 can be OFF when the power Auxiliary 3V3 source is available)	Not implemented	Yes On separated pins
Independent from Raspberry Pi ® 5V0 supply @200 mA With battery Back-up (Raspberry Pi® 5 can be OFF when power Auxiliary 5V0 source is available)	Not implemented	Yes On separated pins
1 x 12 Bit A/D converters ESD protected, pre-scaled to 5V or 25V with Sampling rate 200K SPS, DMA buffered, Low Pass Software filtered/ Nonfiltered Both A/D with Voltage Follower buffer	Not implemented	Yes
3V3 RS232 Port that can be programmed as: primary Raspberry Pi® Port Secondary (independent from the existing on Raspberry Pi®)	Not implemented	Yes
System LEDs Indicators	ACT – SSD activity M.2 – SSD power USR – user indication SYS – system status BAT – battery level CHG – charger status TMR – timer FAN – official cooler activity TMP – Raspberry Pi core temperature	ACT – SSD activity M.2 – SSD power USR – user indication SYS – system status BAT – battery level CHG – charger status TMR – timer FAN – official cooler activity TMP – Raspberry Pi core temperature EPR – external power status
User LED Indicator	White With capability to connected external LEDs (soldering) Possibility of Mapping of System Events to User LED	White With capability to connected external LEDs (plug in) Possibility of Mapping of System Events to User LED
System Keys	UPSR, FSSD	UPSR, FSSD
User Programmable Key	AKEY (soldered cable only)	AKEY
External Connectivity to Keys	FSSD, AKEY With capability to connected external KEYs) Optional ON/OFF slide Switch	FSSD, AKEY With capability to connected external KEYs) Optional ON/OFF slide Switch

Feature	Standard	Advanced / Passive PoE
Independent to Raspberry Pi® Watchdog (Still Alive)	Yes	Yes
Battery Backed Hardware Real Time Clock and Calendar	Yes/Selectable When UPS (power cycling is used)	Yes/Selectable When UPS (power cycling is used)
System Switch ON/OFF	Optional Supported with Embedded or external ON/OFF slide switch or external one (even is system is powered by USB C) Selectable/Combined In addition to Raspberry Pi 5 Push button	Optional Supported with Embedded or external ON/OFF slide switch or external one (even is system is powered by USB C) Selectable/Combined In addition to Raspberry Pi 5 Push button
Solar Panel Supply Input	on 5V0 USB-C battery charging current adopted to existing solar conditions	on 5V0 USB-C on EPR (5V-32VDC) battery charging current adopted to existing solar conditions
System Cooling		
Advanced Automatic Active Cooling System (FAN)	Raspberry Pi official cooler controlled by the Pi; CPU temperature supplied through the Linux daemon.	Raspberry Pi official cooler controlled by the Pi; CPU temperature supplied through the Linux daemon.
Passive Cooling	Thought extended hole system supporting air circulation, extended cooling copper planes	Thought extended hole system supporting air circulation, extended cooling copper planes
System Control		
Vital System Information	Thought Registers Set accessed via I²C API	Thought Registers Set accessed via I²C API
Selected Vital System Information	Local serial diagnostic output.	Local serial diagnostic output.
Programmable/Accessible all the system parameters via I²C API	Supported settings and status fields are listed in the register appendix; reserved fields remain unchanged.	Supported settings and status fields are listed in the register appendix; reserved fields remain unchanged.
SysInfo Register	Yes, Provide core information about the system: System FSSD Reason - System Wakeup Reason - Power Management Restart Reason -	Yes, Provide core information about the system: System FSSD Reason - System Wakeup Reason - Power Management Restart Reason -
M.2 - UPS & Power Management HAT Running Register	YES, provide information to remote user if system is running properly	YES, provide information to remote user if system is running properly

Feature	Standard	Advanced / Passive PoE
Remote and local Bootloader for Live Firmware Update	YES	YES
e-mail sending on event	Host application required; not supplied by the daemon.	Host application required; not supplied by the daemon.
Measuring and Monitoring System		
Real Time Raspberry Pi® System current measure	Dual High-side bi-directional Hardware Current Sensing Monitor with power calculation (5V0 path)	Dual High-side bi-directional Hardware Current Sensing Monitor with power calculation (5V0 path) (5V0-32V DC path)
Powering Mode Status	YES	YES
Battery Level	YES	YES
Raspberry Pi® 5V level	YES	YES
External Powering Level	NO	YES
Passive PoE Level	NO	YES
Raspberry Pi® Core Temperature	YES	YES
Raspberry Pi® Active cooler FAN status	Host fan telemetry required; not supplied by daemon v1.1.	Host fan telemetry required; not supplied by daemon v1.1.
A/D(s) Level	Not fitted	Analogue input on the fitted Advanced assembly.
Charger Status	YES	YES
Scheduler		
Time/Calendar Scheduler	Basic scheduler register interface.	Basic scheduler register interface.
Shutdown/Wake-up on	Timed battery limit; low battery; FSSD button; optional slide switch; accepted cable return.	Timed battery limit; low battery; FSSD button; optional slide switch; accepted cable return.
Manufacturing		
PCB Manufacturing	4 Layers, 2 OZ Copper, 6mils/6mils Immersion Gold Plated Lead-free alloy assembly	4 Layers, 2 OZ Copper, 6mils/6mils Immersion Gold Plated Lead-free alloy assembly

A Complete Description of the M.2 UPS PMH Programming Registers

Applies to firmware V88 (0x0088) with default I²C addresses. B = one byte; W = two bytes. Multi-byte values are little-endian. Signed values use two's complement. BCD values contain decimal digits in each nibble. Padding bytes are reserved and must not be written. A read-only monitor and machine-readable CSV accompany this appendix.

Saved settings are written by the normal foreground handler after a changed value is accepted. Allow at least three seconds and read back the value. Avoid continuously rewriting settings. Runtime and volatile fields are not a promise of retention after MCU reset. Reserved fields are listed to make the bank layout complete; leave them unchanged.

Status bank — 0x69

Offset	Variable	Type / access	Description
00	mode	U8 Read Only	Accepted power mode: 0xF0 battery; 0xF1 cable.
01	backing_mode	U8 Read Only	Backup-source mode code; 0x01 indicates battery backup.
02	rtime	U16 Read Only	Reserved runtime / energy field; not a battery percentage.
04	Foreground loop counter	U16 Read Only	Foreground-loop counter; wraps modulo 65536.
06	scalevel	U16 Read Only	Supercapacitor level field; raw profile-dependent value.
08	batlevel	U16 Read Only	Battery voltage, packed BCD in 0.01 V units; 0x0390 = 3.90 V.
0A	rpilevel	U16 Read Only	Raspberry Pi supply voltage, packed BCD in 0.01 V units.
0C	eprlevel	U16 Read Only	External source voltage, packed BCD in 0.01 V units; zero when inactive.
0E	curilevel	S16 Read Only	PAC1720 channel 1 signed current, mA.
10	pwrlevel	S16 Read Only	PAC1720 channel 1 signed power field, mW; 16-bit storage.
12	vltlevel	U16 Read Only	PAC1720 channel 1 voltage, binary mV.
14	aEXT0level	U16 Read Only	Auxiliary analogue channel 0, packed BCD; fitted Advanced input only.
16	aEXT1level	U16 Read Only	Reserved analogue channel 1 field; do not infer a live measurement.
18	key_pressed	U8 Read Only	Application KEY status code, where fitted: 0 no event, 1 or 2 input event.
19	PCB_temp	U8 Read Only	PCB temperature slot; raw, no active temperature conversion guaranteed.
1A	RPI_temp	U8 R/W host	Host CPU temperature, packed BCD degrees Celsius, 0x00–0x99; daemon writes every 5 s.

Offset	Variable	Type / access	Description
1B	_padding_1B	U8 Reserved	Alignment byte; leave unchanged.
1C	optlevel	U16 Read Only	Reserved input field; leave unchanged.
1E	BAT_chg_stat	U8 Read Only	Charger current-limit code; multiply the binary value by 10 for mA.
1F	BAT_chg_real	U8 Read Only	Reserved charger-feedback slot; not a measured-current value.
20	SCA_pwm_stat	U8 Read Only	Supercapacitor control status slot; raw.
21	FAN_status	U8 Read Only	Fan-status slot; daemon v1.1 does not populate fan-speed telemetry.
22	sysinfo	U16 Read Only	System information word: bits 3:0 FSSD reason; 7:4 wake reason; 11:8 restart reason.
24	RS232_status	U8 Read Only	Serial status byte; raw.
25	RS232_data	16 bytes Read Only	16-byte serial receive buffer; raw bytes.
35	pcb_version	U8 Read Only	PCB / bootloader identification byte, hexadecimal.
36	Model identifier	U8 Read Only	Assembly identifier: ASCII A (0x41) or S (0x53).
37	default_battery	U8 Read Only	Factory backup-source identification, ASCII chemistry code.
38	firmware_version	U16 Read Only	Application firmware identifier; V88 reads 0x0088.
3A	dummy	U16 Reserved	Reserved trailing word; leave unchanged.

Command bank — 0x6B

Offset	Variable	Type / access	Description
00	M2_PMH_state	U8 R/W volatile	Linux heartbeat / shutdown handshake. 0xCC and 0x77 are daemon shutdown requests; daemon acknowledges with 0x00. Maintenance commands are listed below.
01	bat_run_time	U8 R/W saved	0x00–0xFE: timed FSSD after about 10 + 60 × value seconds. 0x01 default (70 s); 0xFF no elapsed-time limit. Low-battery protection remains active.
02	RS232_rate	U8 R/W saved	Packed baud nibbles: high local UART, low Pi UART. 0 off, 1=4800, 2=9600, 3=19200, 4=38400, 5=57600, F=115200. Default 0xF0.
03	rpi_hold_on	U8 R/W saved	Pi startup hold setting, seconds; default 0. A changed zero is clamped to 1 by the runtime handler.

Offset	Variable	Type / access	Description
04	Reserved serial route	U8 Reserved	Serial routing slot; do not modify.
05	sta_timer	U8 R/W volatile	Still-alive countdown byte; 0xFF disables it. Expiry is a controller action; leave disabled for the supplied daemon.
06	enable5V	U8 Reserved	Auxiliary supply-control slot; leave unchanged.
07	battype	U8 R/W saved	Backup profile: L=0x4C LiPo, I=0x49 Li-ion, F=0x46 LiFePO4, C=0x43 supercapacitor (4000 F). Only these four profiles are supported. Change only for the matching fitted source.
08	setA_D	U8 Reserved	Analogue-range configuration slot; leave unchanged.
09	led_green	U8 Reserved	LED command slot; system animations own physical LED outputs.
0A	led_blue	U8 Reserved	LED command slot; leave unchanged.
0B	relay	U8 Reserved	No user relay control is specified for this assembly.
0C	bmode	U8 Reserved	Sound-control slot; leave unchanged.
0D	_padding_0D	U8 Reserved	Alignment byte; leave unchanged.
0E	bfreq	U16 Reserved	Sound-frequency word; leave unchanged.
10	bdur	U8 Reserved	Sound-duration slot; leave unchanged.
11	fttemp	U8 Reserved	Temperature-threshold slot; not a Raspberry Pi fan profile.
12	LED_OFF	U8 R/W saved	LED-output gate. V88 physical LED writes require value ON (1); default 1. Zero inhibits normal writes. Bootloader/reset indication owns outputs temporarily.
13	on_the_go	U8 R/W saved	0x00 normal UPS, runtime becomes 0x01. 0xAA optional slide-switch mode, runtime becomes 0xFF. Default 0x00.
14	map_led_green	U8 Reserved	LED mapping mask; leave unchanged.
15	map_led_blue	U8 Reserved	LED mapping mask; leave unchanged.
16	chg_ctrl	U8 R/W runtime	Charger permission: 0 disables; 1 enables charging when power and battery conditions permit. Default 1; permission is not measured current.
17	fssd_duration	U8 R/W saved	Linux shutdown grace interval in seconds; default 30 (0x1E).

Offset	Variable	Type / access	Description
18	rpi_OFF_time_reg	U8 R/W runtime	No-host-activity timeout in minutes; default 5. Foreground uses 60 × value + startup hold seconds.
19	ad_corectionf	S8 Reserved	Signed calibration byte; normal startup resets correction to zero.
1A	Scheduler_Sel	U8 Reserved	Scheduler selection slot; not configured by the supplied daemon.
1B	BS_action_time	U8 R/W saved	Basic scheduler action-duration byte; raw. Do not alter during UPS commissioning.
1C	BS_inaction_time	U8 R/W saved	Basic scheduler inactive-duration byte; raw.
1D	_padding_1D	U8 Reserved	Alignment byte; leave unchanged.
1E	BS_Start_time	U16 R/W saved	Basic scheduler time word: low byte minute 0–59, high byte hour 0–23, binary; 0xFFFF special value.
20	BS_multipier	U8 R/W saved	Basic scheduler duration multiplier byte; raw.
21	BS_RUN	U8 R/W saved	0x00 stops basic scheduler and sets runtime 0x01; 0xFF starts action and sets runtime 0xFF. Leave 0x00 for ordinary UPS use.
22	SAS_number	U8 Reserved	Scheduler event-number slot; leave unchanged.
23	SAS_run	U8 Reserved	Scheduler event-run slot; leave unchanged.
24	ps_checking_time	U8 R/W saved	Cable-check interval: value × 250 microseconds; minimum 10, default 30 (7.5 ms).
25	BAT_level_report	U8 R/W saved	Battery report scheduling interval byte, seconds; default 5. 0xFF suppresses regular level messages.
26	SC_level_start	U16 R/W saved	Supercapacitor startup threshold word; profile-specific raw ADC scale. Use the matched factory profile.
28	debug_mode	U8 R/W volatile	Diagnostic output routing: 00 off, F0 local, 0F Pi UART, FF both. Default F0.
29	RS232_data	8 bytes Reserved	Eight-byte serial transmit slot; leave unchanged.
31	_padding_31	U8 Reserved	Alignment byte; leave unchanged.

System information reason codes

FSSD reason (low nibble): 1 button, 2 low battery, 3 low supercapacitor, 4 timed battery limit, 5 simple scheduler, 6 event/time scheduler, 7 event. Wake reason (next nibble): 1 button, 2 running Pi, 3 Pi cable power, 4 external power, 5 simple scheduler, 6 event/time scheduler, 7 event. Restart reason (bits 11:8): 1 power-up, 2 hardware reset, 3 watchdog, 4 other. Zero indicates no published reason in that field.

Command-state maintenance codes — 0x6B / 0x00

Normal applications leave this byte to the daemon. The following codes change routing, configuration or program flow; use the dedicated recovery procedures for factory defaults and firmware updates.

Code	Action
0x00	Idle / daemon acknowledgement.
0x11	Save Pi UART diagnostic routing at 115200 and reset MCU.
0x22	Save local UART diagnostic routing at 115200 and reset MCU.
0x33	Save diagnostic output OFF and reset MCU.
0x99	Publish 0x77 shutdown request to the Linux daemon.
0xAA	Unconditional FSSD request.
0xDD	Factory defaults maintenance request.
0xEE	M.2 controller reset request.
0xFF	Local bootloader request.
0xBB	Pi-side bootloader request.
0xCC	Outgoing Linux shutdown request; no user maintenance action.
0x60	Select default I²C group and reset.
0x68	Select reduced no-RTC address group and reset.
0x50	Select alternate group 0x58–0x5F and reset.
0x40	Select alternate group 0x48–0x4F and reset.

RTC bank — 0x68

Offset	Field	Encoding / range
00	Seconds	BCD 00–59; driver manages control bits.
01	Minutes	BCD 00–59.
02	Hours	BCD 00–23 in the supplied 24-hour configuration.
03	Weekday	Driver calendar weekday code.
04	Day	BCD 01–31, valid for the selected month.
05	Month	BCD 01–12.
06	Year	BCD 00–99, interpreted as 2000–2099.
07 and above	Reserved compatibility area	Use the Linux RTC driver for clock access.

Address 0x6A exposes an internal RTC structure and is reserved for this package. Addresses 0x6C–0x6E are reserved banks. The supported Linux clock interface is 0x68.

Diagnostic snapshot — 0x6F

Offset	Variable	Description
00	Format identifier	0x58; identifies the snapshot format, not the running firmware.
01	Foreground stage	Raw stage identifier.
02	Flags	Bit 0: Pi running; bit 1: FSSD status.
03	Charger wiper	Raw potentiometer position.
04–05	Tick counter	U16, seconds, modulo 65536.
06–07	Stage age	U16 seconds since the checkpoint, modulo 65536.
08–09	Loop counter	U16 foreground-loop counter.
0A	Local bus operation	0 none; 1 write8; 2 read8; 3 read16; 4 potentiometer write.
0B	Local bus address	Seven-bit address for the recorded operation.
0C	Local bus register	Register or wiper payload, depending on operation.
0D	Power mode	0xF0 battery; 0xF1 cable.
0E	Target stage	Raw checkpoint target.
0F	Serial diagnostic failures	Low byte of the diagnostic transmit-failure counter.

Read-only inspection examples

```
cd ~/M2_UPS_PMH_Manual_V8F
sudo python3 optional_monitor/M2_PMH_status_06.py
sudo i2cget -y 1 0x69 0x38 w
sudo i2cget -y 1 0x69 0x08 w
sudo i2cget -y 1 0x6b 0x13 b
```

Firmware identification is hexadecimal. A battery voltage word 0x0390 means 3.90 V, not binary 912 mV. Status updates can be suspended in low-power or bootloader operation; zero or stale reserved fields are not proof of an attached sensor.

Reading and decoding register values

The preceding tables are the complete address and field reference for the supplied V88 layout. The following explanations show how to use those fields from Linux. Register offsets and device addresses are hexadecimal; each bank has its own offset zero. Use the default addresses below only while the controller is configured for the default address group.

Byte and word transfers

U8 is one unsigned byte. U16 is a two-byte unsigned word with its low byte at the listed offset and its high byte at the next offset. S8 denotes a signed byte. The b and w suffixes in i2cget select byte and SMBus word reads respectively. A word read beginning at an unaligned or incorrect offset combines unrelated fields.

```
sudo i2cget -y 1 0x69 0x38 w
sudo i2cget -y 1 0x69 0x08 w
sudo i2cget -y 1 0x6b 0x01 b
```

The first command reads the application version: 0x0088 means V88. The second reads the battery-voltage field. In this field, 0x0390 represents decimal digits 0390 and therefore 3.90 V. The third reads the binary runtime setting: 0x01 means one minute plus the controller's approximately ten-second allowance, not one second.

BCD and binary are different encodings

BCD stores two decimal digits per byte. For example 0x42 represents 42 in a BCD minutes field, but 66 in an ordinary binary field. Convert voltage/temperature fields according to the encoding stated in the table; do not apply the BCD conversion to counters, masks, current-limit codes or firmware identifiers.

A charger current-limit code is binary and is multiplied by 10 to obtain mA. For example 0x3A is decimal 58 and represents a 580 mA limit. This is a requested or reported limit, not a measured battery current. A charge-permission value of 1 also does not establish that current is flowing.

Access and update rules

Read Only fields are controller-published status. R/W saved fields are settings processed by the firmware and retained when its settings handler saves them. R/W volatile fields are working state and can change immediately. R/W runtime fields are not a promise of persistence. Reserved fields must be left unchanged, even if a read returns zero or an apparently meaningful value.

Read one field or a defined group at a time. Status is updated asynchronously, so separate reads need not describe the same instant. A diagnostic snapshot is useful for troubleshooting but does not make unrelated status-bank reads atomic. Counters can wrap; use modulo arithmetic when comparing them.

The supplied daemon owns the heartbeat/shutdown handshake. Keep it running during normal monitoring. A successful read contributes host activity, so continuous polling can affect tests intended to exercise absence of the host. Do not use broad write loops or force access through an attached kernel driver.

Configuration examples and interactions

Run these examples only on the intended unit, under stable cable power. First read and record the existing values. Change one setting at a time and allow the firmware to process it before reading it back. The examples use bus 1 and the default command bank 0x6B.

Battery runtime at offset 0x01

The binary values 0x00 through 0xFE request approximately 10 + 60 × value seconds before timed FSSD. Thus 0x00 gives about 10 seconds, 0x01 about 70 seconds, and 0x02 about 130 seconds. The Linux shutdown grace interval follows the FSSD request and is a separate setting. 0xFF disables the elapsed-time limit; low-source protection remains active.

```
sudo i2cget -y 1 0x6b 0x01 b
sudo i2cset -y 1 0x6b 0x01 0x02 b
sleep 3
sudo i2cget -y 1 0x6b 0x01 b
```

Normal UPS and switch mode at offset 0x13

Writing 0xAA enables the optional ON/OFF switch mode and sets runtime to 0xFF. Place the fitted switch ON before enabling this mode. Writing 0x00 selects normal UPS operation and restores runtime 0x01. Select the mode first; set a different normal-mode runtime afterwards if required. A later mode change can overwrite that runtime choice.

```
# Return to normal UPS operation and its default 70-second limit
sudo i2cset -y 1 0x6b 0x13 0x00 b
sleep 3
sudo i2cget -y 1 0x6b 0x13 b
sudo i2cget -y 1 0x6b 0x01 b
```

Charging and shutdown interval

Offset 0x16 is charging permission: 0 inhibits charging and 1 permits it when the controller's supply and battery checks allow. It does not choose a battery chemistry or force a charging current. Offset 0x07 selects the backup-source profile and must match the actual fitted assembly; changing it is not a charge-performance adjustment.

Offset 0x17 is the Linux shutdown grace interval in seconds, default 30 (0x1E). Keep enough time for the actual workload to stop safely. Offset 0x18 is the separate no-host-activity timeout in minutes. Neither field is the ordinary battery-runtime limit. Leave the still-alive timer and scheduler at their supplied settings during normal UPS commissioning.

Output and serial settings

The LED gate at 0x12 permits normal physical LED updates when set to 1. System/reset/bootloader indications may temporarily own the outputs. The UART-rate byte at 0x02 contains two independent nibbles; it does not change the protected bootloader's 38400 bit/s transfer rate. Diagnostic routing at 0x28 controls the application's log destination.

After a settings change, repeat the relevant functional check: normal cable operation, battery FSSD, LPR, cable return during FSSD and after shutdown, and switch behavior where fitted. A matching readback establishes the stored value, not the complete power-control result.

Maintenance commands and RTC access

Command byte 0x6B / 0x00

This byte is also the live Linux handshake. The maintenance-code table describes actions, not persistent user settings. A command may be consumed and cleared before it can be read back. Verify the resulting log or operating state instead of repeatedly writing it until the byte remains visible.

The 0xCC and 0x77 values are shutdown requests observed by the daemon; it acknowledges through 0x00 before requesting Linux poweroff. Do not overwrite this handshake during FSSD. A user request such as 0x99 or 0xAA deliberately starts a shutdown action and can terminate SSH.

Enter the Pi-side protected bootloader

```
sudo i2cset -y 1 0x6b 0x00 0xbb b
```

Run the updater promptly within the clean-application loader entry window. Use -j when the board is already in the bootloader, so the updater does not try another application-side entry command. The ordinary application register banks can be unavailable during loader operation.

```
sudo python3 M2_PM_UPS_BL01D.py -v -i -j -k \
  -p /dev/ttyAMA0 -b 38400 -f M2_PMH.hex
```

The 15-second entry window ends after the first actual Flash write. An incomplete write remains in recovery until a complete verified application is supplied. Use the detailed firmware-update and recovery chapters for the full procedure. The local-loader code 0xFF is a different routing request; it is not the normal Pi-side upload command.

Factory defaults and controller reset

The 0xDD maintenance request selects factory defaults; 0xEE requests a controller reset. These actions can change settings or disturb the running Pi through the normal power sequence. Save the required configuration and stop important work before using them. Factory defaults also restore the RTC reference date; confirmed NTP synchronization is needed to restore current time afterwards. The former button-selected factory-default menu is not present in V88.

Address-group commands also reset the controller. Once the group changes, the old addresses no longer describe the active interface, and the supplied daemon/RTC configuration may need corresponding changes. Do not use those commands as a connectivity test.

Use the Linux RTC driver

Bank 0x68 provides the DS1307-compatible clock interface. Identify the registered device by its driver/name rather than assuming that it will always be rtc1. When rtc-ds1307 owns address 0x68, use hwclock; do not force raw transfers past that driver. Confirm the Linux clock and NTP state before writing system time to the RTC.

```
timedatectl status
sudo hwclock --show --rtc=/dev/rtc1
# Only after confirming the Linux system time is correct:
sudo hwclock --systohc --utc --rtc=/dev/rtc1
sudo hwclock --show --rtc=/dev/rtc1
```

The RTC year range is 2000–2099 in this interface. Linux displays a local-time conversion when configured to do so while the RTC is kept in UTC. An apparent timezone difference does not by itself indicate an incorrect register write. The separate Raspberry Pi RTC power cable supplies the Pi clock domain; it does not merge the two clock interfaces.

Technical support: support@pimodules.com | www.pimodules.com

Tools and SD to SSD migration

Manual V8E adds the status monitor 06, configuration tool 01 and a Raspberry Pi 5 SD-to-NVMe migration procedure. Firmware V88 and the tested Linux installer V01 are unchanged. The following chapters cover running the tools, interpreting their output and safely applying configuration changes.

Files and launch commands

Extract the V8E ZIP in your home directory. Run the tools from the package directory; the paths below do not depend on a copy in /opt. Python 3 and python3-smbus are required; installer V01 already installs this dependency.

```
cd ~/M2_UPS_PMH_Manual_V8F
sudo python3 optional_monitor/M2_PMH_status_06.py
sudo python3 configuration/M2_PMH_config_01.py
```

The status tool only reads. Opening the configuration menu, refreshing it and exporting settings also only read. A configuration change requires an explicit YES confirmation. Keep cable power connected and the heartbeat/shutdown service running.

Optional fixed installation paths

Installer V01 installs its bundled older monitor as /opt/M2_UPS_PMH/M2_PMH_status.py. It does not automatically install these two new versioned filenames. To place the new tools at fixed paths, run:

```
cd ~/M2_UPS_PMH_Manual_V8F
sudo install -d /opt/M2_UPS_PMH
sudo install -m 0755 optional_monitor/M2_PMH_status_06.py \
  /opt/M2_UPS_PMH/M2_PMH_status_06.py
sudo install -m 0755 configuration/M2_PMH_config_01.py \
  /opt/M2_UPS_PMH/M2_PMH_config_01.py
```

Then use sudo python3 /opt/M2_UPS_PMH/M2_PMH_status_06.py or sudo python3 /opt/M2_UPS_PMH/M2_PMH_config_01.py. “No such file or directory” means the requested file path is wrong; it does not indicate an I²C fault.

Version naming and chapter guide

Configuration filenames advance in hexadecimal: M2_PMH_config_01.py, …_09.py, …_0A.py, …_0F.py, …_10.py. This sequence is independent of manual V8E, status package 06 and application firmware V88.

The next pages describe the status monitor, the configuration menu and its settings, followed by SD-to-SSD preparation, cloning, first boot and recovery.

Status monitor 06

M2_PMH_status_06.py version 2.1 uses the firmware V88 register map. It displays clocks, board identity, power and battery readings, timers, UARTs, LEDs and scheduler fields. Each line includes the raw value and register where applicable.

```
cd ~/M2_UPS_PMH_Manual_V8F
sudo python3 optional_monitor/M2_PMH_status_06.py
sudo python3 optional_monitor/M2_PMH_status_06.py --watch 5
python3 optional_monitor/M2_PMH_status_06.py --help
```

A single invocation prints one report and exits. --watch 5 repeats every five seconds; press Ctrl+C to stop. --show-map displays the map, --degrees F selects Fahrenheit, and --no-rtc skips RTC reporting. Use the default bus 1 and address group unless the installation has deliberately been changed.

Reading the report

Firmware should be 0x0088. Factory storage type and active storage type can differ: the tested Advanced assembly reports factory LiPo (L) and active Li-ion (I). The active profile must match the installed source. The PCB/bootloader identity byte is not proof of the protected loader binary version.

Linux local time and RTC UTC can differ by the configured timezone. The monitor uses the Linux RTC interface, with sysfs fallback when hwclock is unavailable. It does not force I²C access past the kernel RTC driver.

Charging enabled means permission to charge; the current limit is not a measured charge current. Battery voltage does not establish charge percentage. PAC1720 calibration and the current/power relationship remain to be validated; do not use these fields as charging acceptance criteria.

```
M.2 PMH Status | v2.1 | Package 06 | Firmware V88

Firmware               0x0088
Model                  Advanced
Default / active source LiPo (L) / Li-ion (I)
Power source           Cable power
Battery / Pi rail      4.20 V / 5.10 V
Charger permission     Enabled
Charger current limit  570 mA (not measured)
Runtime / FSSD grace   Approx. 70 s / 30 s
UPS mode               Normal UPS
Main-loop counter      44580 -> 44644
```

Terminal excerpt from the reported 22 October 2026 hardware test, reformatted for readability. Values are examples, not required defaults.

Reads count as host activity. Stop repeated monitor/configurator reads before testing no-host timeout behavior. Leave the shutdown daemon running during ordinary UPS power-loss tests. Values are sampled sequentially, not as one simultaneous snapshot.

Configuration tool 01

M2_PMH_config_01.py provides a numbered M.2-specific menu for firmware V88. It excludes unrelated peripherals, firmware flashing, factory reset, raw register writes and scheduler programming. It does not calibrate PAC1720 or write the RTC.

```
cd ~/M2_UPS_PMH_Manual_V8F
sudo python3 configuration/M2_PMH_config_01.py
sudo python3 configuration/M2_PMH_config_01.py --show
python3 configuration/M2_PMH_config_01.py --guide
```

--show prints the current configuration and exits without writes. --guide explains operation without opening the hardware. The menu shows firmware, model, power source and each setting's raw value and interpretation.

Change one setting

1. Select the setting number and press Enter. Read the current value and the displayed limits or choices. Enter the new decimal value or 0x-prefixed hexadecimal value; chemistry also accepts L, I, F or C. An empty new-value entry cancels.

2. Review the proposed change and any side effects. Type YES exactly to apply. Any other response cancels. Choosing the existing value sends no write.

3. Wait for the settling delay and readback result. "Readback verified" confirms the register value at that moment; it does not prove retention through a later reset. Refresh with R and recheck after a planned restart when persistence matters.

Example for the tested Li-ion assembly: select 1, enter I, verify that the installed battery is Li-ion, and type YES only if a change is required. Do not change chemistry merely to match a screenshot.

Menu action	Operation
R	Refresh current values; no settings are written.
E	Export a new JSON reference snapshot. Enter a filename; existing files are not overwritten. There is no automatic restore/import.
H	Show help and operating notes.
Q or Enter	Exit the menu.

Write guards and recovery

Writes require firmware 0x0088, a recognized Standard or Advanced model, cable power and an idle controller command byte. Mode changes are blocked while the basic scheduler is running; runtime changes also require normal UPS mode. Resolve the cause of a refusal before trying again.

If a write or readback is uncertain, the tool does not automatically repeat the write. Refresh, inspect the current state and decide whether another action is needed. UART-rate changes reset the MCU: save work first and expect a brief interruption. Keep the power cable connected throughout.

The menu was reported working on the target Pi. This does not validate every setting under every hardware combination; retain the known working configuration unless a change is needed.

Configuration settings reference

The following menu numbers refer to configuration package 01. All offsets are in command bank 0x6B. Use the on-screen description before confirming a change.

No and setting	Allowed value and effect
1 Backup chemistry 0x07	L / 0x4C LiPo; I / 0x49 Li-ion; F / 0x46 LiFePO4; C / 0x43 supercapacitor 4000 F. Match the installed source.
2 UPS mode 0x13	0 = normal UPS; 0xAA = Magic Switch. Fit and turn ON the switch first. A mode change sets runtime to 1 (normal) or 255 (Magic Switch).
3 Battery runtime 0x01	0–254: approximately 10 + 60 × value seconds plus startup hold before FSSD. 255 removes the elapsed-time limit; low-battery protection remains.
4 Shutdown grace 0x17	1–255 seconds; default 30. Allow enough time for Linux and applications to stop.
5 Charging permission 0x16	0 disabled / 1 enabled. Runtime setting; not a charge-current measurement.
6 LED output gate 0x12	0 inhibits normal LED writes / 1 enables them. Reset and bootloader indications can override it.
7 UART rates 0x02	Wizard asks local UART2 then Pi UART1. Codes: 0 off, 1 4800, 2 9600, 3 19200, 4 38400, 5 57600, F 115200. Changing this resets the MCU.
8 Diagnostic routing 0x28	0 off; 0xF0 local UART; 0x0F Pi UART; 0xFF both. Runtime setting; the chosen UART must be enabled.
9 Startup hold 0x03	1–255 seconds when edited. An existing factory value of 0 can be displayed. Contributes to shutdown timing.
10 No-host timeout 0x18	1–255 minutes; default 5. Runtime setting. Host reads/heartbeat keep activity present.
11 Cable interval 0x24	10–255 units of 250 microseconds. Default 30 = 7.5 ms.
12 Battery reports 0x25	1–255 seconds. 255 suppresses regular battery-level diagnostic messages.

Settings marked runtime are not guaranteed to persist. Other entries use firmware-managed retention; the configurator verifies immediate readback only. UART settings affect application UARTs, not the protected bootloader baud rate of 38400 bit/s.

Configuration menu example

This illustration reproduces the menu layout of configuration tool 01 with example values. It is an illustrative terminal view, not a photograph or a measurement from a connected board. Use the values read from your own assembly.

```
M.2 UPS PMH Configuration V01
Firmware map V88 | ENTER or Q to exit

No  Setting                        Example value
 1  Backup-source chemistry        Li-ion
 2  UPS / Magic Switch mode        Normal UPS
 3  Battery runtime before FSSD    ~70 s + hold
 4  Linux shutdown grace period    30 s
 5  Charging permission            Enabled
 6  Normal LED output gate         Enabled
 7  Application UART baud rates    UART2=115200
                                   UART1=OFF
 8  Diagnostic UART routing        Off
 9  Pi startup hold                0 s
10  No-host-activity timeout       5 min
11  Cable-check interval           7.50 ms
12  Battery diagnostic interval    5 s

R Refresh | E Export | H Help | Q Exit
Select a number to inspect or edit a setting.
```

Illustrative configuration menu. The full program additionally shows raw values, offsets, timestamps and board identification.

Example interaction

To inspect battery chemistry, enter 1. Press Enter at the new-value prompt to leave it unchanged. If the physically installed battery requires another profile, enter the matching code, review the displayed change and type YES to apply it.

An export made with E is a reference snapshot. It can help compare settings after migration or a planned restart, but is not an automatic recovery or restore command.

SD to NVMe SSD preparation

This procedure transfers a conventional Raspberry Pi OS installation on a Raspberry Pi 5 from microSD to an NVMe SSD attached through PCIe. It copies Linux, installed packages, user files and UPS service configuration. It does not copy the Pi EEPROM or the M.2 controller firmware/settings, which reside outside the SD card.

Scope: a standard FAT boot partition and ext4 root filesystem. The Debian login banner alone does not establish that the image is Raspberry Pi OS. For generic Debian, encryption, LVM, RAID, unusual partitions or external application data, validate a suitable migration method before using these commands. This procedure has not yet been reported tested on the target assembly.

1 Prepare power and identify both disks

Back up important data separately. Use a compatible NVMe SSD with sufficient capacity; an SSD at least as large as the SD card simplifies sizing. Installing an M.2 SATA drive is not equivalent to NVMe. Shut down and disconnect both cable power and backup battery before fitting the SSD or PCIe ribbon; removing USB power alone does not make a UPS assembly unpowered.

Start the existing SD system with stable cable power. Keep the shutdown daemon running. Record the following output and identify disks by name, size, model and mountpoints:

```
cat /etc/os-release
findmnt -no SOURCE /
findmnt -no SOURCE /boot/firmware
lsblk -o NAME,SIZE,MODEL,FSTYPE,MOUNTPOINTS,PARTUUID
df -h / /boot/firmware
```

The examples below require source /dev/mmcblk0 (root typically mmcblk0p2) and target /dev/nvme0n1. If root is already on NVMe, stop: these commands are not an SD-to-SSD migration. Never select the SD disk as the destination.

2 Enable PCIe only if needed

If the NVMe disk is absent, check the powered-off hardware assembly first. For an adapter without automatic HAT+ configuration, edit /boot/firmware/config.txt and enable dtparam=pciex1 in an applicable [all] section. Preserve existing I²C, RTC and UART settings. Reboot from SD, then repeat lsblk. Keep the default PCIe speed for initial validation.

```
sudo nano /boot/firmware/config.txt
# Under [all], if not already enabled:
# dtparam=pciex1
sudo reboot
```

If an SSD already contains a bootable system, ensure SD remains the source during preparation. EEPROM BOOT_ORDER=0xf461 selects SD, then NVMe, then USB. Change it only if needed, using sudo rpi-eeprom-config --edit; reboot and verify the root device again. Keep a copy of the original EEPROM configuration.

Clone the SD system to SSD

3 Prepare the cloning utility

The rpi-clone project supports NVMe destinations and is intended/tested for Raspberry Pi OS. Download its source, inspect the instructions and record the revision used. The commands below assume a new ~/rpi-clone-m2 directory; if it exists, inspect it rather than deleting it blindly.

```
sudo apt update
sudo apt install -y git rsync
git clone https://github.com/geerlingguy/rpi-clone.git \
  ~/rpi-clone-m2
cd ~/rpi-clone-m2
git rev-parse HEAD
sudo install -m 0755 rpi-clone rpi-clone-setup /usr/local/sbin/
```

4 Quiesce applications and confirm the target

Stop your databases, containers and other applications that write important data, using their normal shutdown procedures. Keep SSH, networking and M2_PMH_i2c.service running. A live filesystem copy is not an atomic database backup. Do not run updates or edit files while cloning.

```
findmnt -no SOURCE /
lsblk -o NAME,SIZE,MODEL,FSTYPE,MOUNTPOINTS
systemctl is-active M2_PMH_i2c.service
```

Confirm root is on mmcblk0 and the destination is nvme0n1. Unmount any mounted destination partitions individually after checking that they are not used by the running system. Close file-manager windows that automatically mount them. Never unmount the source root or source boot partition.

5 Clone and wait for completion

DESTRUCTIVE STEP: the following command initializes /dev/nvme0n1 and erases its existing data. Proceed only after positively identifying this disk as the intended SSD and backing up its contents.

```
sudo rpi-clone nvme0n1 -f
```

Read the source/destination summary before accepting the prompts. The target is the entire disk name nvme0n1, not nvme0n1p1 or nvme0n1p2. Wait for a successful completion and follow the prompt to unmount the clone. If an error occurs, remain on SD and resolve it before changing boot order.

The cloned root and boot configuration must refer to the destination partitions. Do not manually replace the generated SSD fstab or cmdline.txt with the original SD versions. Record the final lsblk output so the SSD partition IDs can be compared if boot troubleshooting is needed.

Keep the SD card unchanged after cloning. Changes made to the SD system after the copy will not automatically appear on the SSD. Restart application services only on the system you intend to keep using.

Boot from SSD and verify the UPS

6 Select NVMe boot

Save the current EEPROM configuration, then edit it. Set BOOT_ORDER=0xf416 (NVMe, SD, USB, repeat; digits are processed right to left). For a non-HAT+ PCIe adapter, also set PCIE_PROBE=1. Preserve unrelated settings.

```
sudo rpi-eeprom-config > ~/m2-eeprom-before-ssd.txt
sudo rpi-eeprom-config --edit
# BOOT_ORDER=0xf416
# PCIE_PROBE=1  (for a non-HAT+ adapter)
sudo reboot
```

Leave the SD installed for this reboot so the pending EEPROM update can apply. After reconnecting, inspect sudo rpi-eeprom-config and findmnt -no SOURCE /. This reboot may already select SSD. If EEPROM editing/updating is blocked or unavailable, resolve that first; the M.2 configurator does not manage Pi EEPROM.

7 Prove an SSD-only boot

Run sudo poweroff and wait for Linux to finish shutting down. Fully remove cable and backup power before removing the SD card. Keep the card safe, restore the normal battery and cable connections, and boot with only the SSD present.

```
findmnt -no SOURCE /
findmnt -no SOURCE /boot/firmware
lsblk -o NAME,SIZE,FSTYPE,MOUNTPOINTS
df -h /
systemctl is-active M2_PMH_i2c.service
cd ~/M2_UPS_PMH_Manual_V8F/installer
sudo python3 install_M2_UPS_PMH.py --check
sudo python3 ../optional_monitor/M2_PMH_status_06.py
sudo python3 ../configuration/M2_PMH_config_01.py --show
```

Root must resolve to the SSD root partition (normally nvme0n1p2) and boot to its boot partition (normally nvme0n1p1). Confirm usable free space, network access, application data, active UPS service, firmware 0x0088 and the correct battery profile. If this package was not present when cloned, copy/extract it first or use the installer directory actually copied.

Repeat the controlled UPS test with important work saved: a short cable interruption, then a planned longer interruption with enough time for graceful shutdown, followed by cable-return startup. Monitor/configurator reads count as host activity; close them before a no-host timeout test. SSD load can change shutdown duration, so confirm the configured grace period is sufficient.

8 Roll back when needed

Power off and fully isolate cable and battery before changing hardware. If SSD boot fails, disconnect the SSD and reinstall the retained SD to boot the original system. Boot-order fallback does not guarantee recovery from a kernel failure on an otherwise readable SSD. Reconnect the SSD only while fully unpowered; check power, ribbon orientation, boot order and partition IDs before another attempt. Never run the clone command in the reverse direction by accident.

References checked 22 October 2026: Raspberry Pi documentation, “PCIe connector” and “Boot from PCIe”: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#boot-from-pcie ; rpi-clone project instructions: https://github.com/geerlingguy/rpi-clone .

Supercapacitor cycle life and temperature

The designated supercapacitor offers potential advantages for frequent power interruptions and wider ambient temperatures. The comparisons below distinguish arithmetic reference ratios from verified service-life guarantees. They do not qualify an alternative battery for use with this UPS.

Cycle count comparison

Using 20,000 cycles as the reference from the existing M.2 FAQ, the cycle-count ratio is 40× against 500 cycles, 20× against 1000 cycles, or 10× against 2000 cycles.

For context, ASOL specifies at least 500 cycles for its 502540 LiPo example at 100% depth of discharge, 0.2C and 25°C. EEMB advertises more than 2000 cycles for its standard LiFePO4 pouch family. The 1000-cycle value is an additional comparison scenario, not a rating assigned to every Li-ion battery. These examples are not recommended replacement cells.

Ratios compare the stated reference counts only. Different capacity-retention endpoints, discharge depths, currents and temperatures prevent a guaranteed 40× or 10× service-life claim. Partial top-ups are not necessarily full cycles; calendar ageing also matters. The PM_097 page mentions up to 100,000 cycles without a matching detailed test protocol. Retain the 20,000-cycle comparison basis until the exact supplied part and test conditions are confirmed.

Calendar life independent of cycle count

Calendar life is the time until ageing makes the source unable to meet its capacity or backup-time requirement, even if it is rarely cycled. Cycle count cannot be converted into a reliable service life in years. In standby UPS use, temperature and sustained charge voltage/state of charge are particularly important.

Published examples illustrate the range: Tektronix gives a general 2–3 year expectation for its conventional Li-ion batteries even without full cycling. Eaton describes selected supercapacitor products with calendar lives up to 20 years, dependent on voltage and temperature. However, Schneider Electric reports estimated service lives of 10 years or more for some engineered Li-ion UPS batteries. Therefore, neither 2–3 years for every battery nor 20 years for every supercapacitor is valid.

For the supplied 4.2 V / 4000 F assembly, a verified calendar-life specification is not available in the cited product information. A precise year rating or a multiple of battery calendar life cannot yet be assigned. Obtain the manufacturer's endurance/ageing data for the exact part at the intended temperature and standby voltage; Eaton's ratings cannot be transferred to PM_097.

Practical assessment: repeat the same backup-load test periodically and compare supported time, voltage sag and temperature with the initial result. Replace the storage element when it no longer provides the required Linux shutdown reserve. This applies even if its nominal cycle count has not been reached.

Supercapacitor temperature and energy ratings

Temperature range comparison

PM_097 lists −40 to +65°C in its technical table, but −40 to +70°C in its feature description. Using the narrower published range gives a 105°C span. The LiPo example above and the Panasonic/Sanyo NCR18650B reference sheet both specify −20 to +60°C for discharge, an 80°C span.

Against that −20 to +60°C discharge range, −40 to +65°C extends the cold limit by 20°C and the hot limit by 5°C: the span is 25°C wider, or 31.25% wider.

If −40 to +70°C is confirmed for the supplied part, the span becomes 110°C: 30°C or 37.5% wider, with a 20°C lower cold limit and a 10°C higher hot limit. EEMB's cited LiFePO4 family lists −10 to +60°C; compared with that 70°C span, −40 to +65°C is 35°C or 50% wider. These percentages compare interval widths, not absolute Celsius temperatures.

Charging is a separate limit: the cited LiPo and NCR18650B references specify 0 to +45°C for charging. Do not infer that the supercapacitor can be charged throughout its advertised operating range; confirm separate charge/discharge limits for the exact part. The Raspberry Pi, SSD, charger and enclosure may impose tighter system limits.

Energy rating and source notes

The energy chapter uses the ideal constant-capacitance model: 9.80 Wh total and 5.44 Wh between 4.2 and 2.8 V. PM_097 separately lists 14 Wh; this is not consistent with 4000 F constant capacitance at 4.2 V. A hybrid device or a different capacitance definition requires its discharge curve or measured energy integral. Do not present the calculated battery equivalence as a certified module capacity.

Sources checked 22 October 2026: Pi Modules PM_097, pimodules.com/product/4000f-super-capacitor; ASOL 502540, asolbattery.com/product/asol-502540-3-7v-500mah-lithium-polymer-battery/; EEMB LiFePO4 pouch family, eemb.com/products-50; Panasonic/Sanyo NCR18650B reference datasheet, version 13.11 R1 (2012). Calendar-life references: Tektronix, Lithium-Ion Battery Maintenance Guidelines (tek.com); Eaton, Supercapacitor module resource center (eaton.com); Schneider Electric, Life Expectancy of APC Lithium Ion Batteries, FAQ000265882 (se.com).