CMMME PRIMA

Complete Manual Firmware 2.0.25

Start simple. Go deeper when you need to.

A complete original rewrite covering safe connection, first setup, everyday configuration, servo power control, DCC and button operation, diagnostics, maintenance, and the technical design of the project.

RELEASE 2.0.25 | 3 NOVEMBER 2026

32 outputs | 16 physical inputs | DCC | Wi-Fi

READ THIS FIRST

Safety and scope

Prima is an exposed low-voltage controller for adult hobby use. Correct power connection matters more than any software setting.

POWER SAFETY
Use regulated 5 V DC only. Check polarity before power-up. The external input is not polarity protected or regulated. Never use external 5 V and USB power at the same time.

BEFORE USB FIRMWARE WORK
Disconnect external 5 V and DCC before connecting USB. If the board is installed under a layout, isolate it first.

What this manual covers

This manual replaces the old firmware workflow with the current 2.0.25 interface and behavior. It is written from the maintained firmware source and the hardware information in Prima Manual V1.1.

Original hardware reference: CMMME Prima Manual V1.1 (05/01/22).

Essential limits

- 32 independently configurable outputs: Servo, Relay or LED.
- 16 physical switch/button inputs plus opto-isolated DCC input.
- Typical starting supply: regulated 5 V / 2 A; size the supply and wiring for the connected load.
- An SG90-class servo can draw roughly 600 mA at movement start and more if stalled.
- Firmware current-shaping features reduce surges but do not replace sound electrical design.

CURRENT RECOMMENDED BASELINE
Global Speed 400, Step 4, Simultaneous Servos 2. The tested two-servo installation also operates at 800 / 4 / 2.

NAVIGATION

Contents

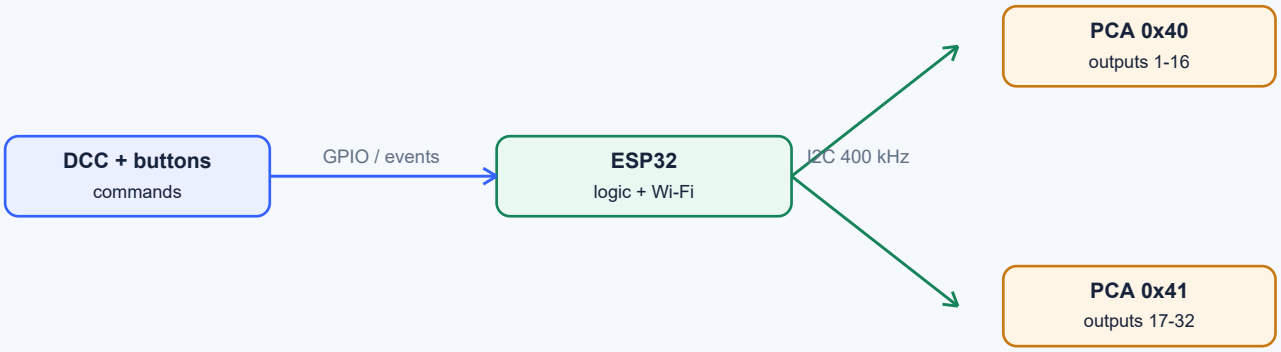
Part I gets a new unit working. Part II explains every user function. Part III is the project and developer reference.

PART I - QUICK START	**4-12**
Board at a glance	4
Connect one servo	5
Power and command inputs	6
First power-on and Wi-Fi	7
Configure the first servo	8
Add DCC control	9
Add a physical button	10
Control two servos together	11
Save, back up and commission	12
PART II - COMPLETE OPERATION	**13-25**
Interface and health	13
Node and Wi-Fi configuration	14
Output model	15
LED and relay outputs	16
Servo basics	17
Servo advanced movement	18
Global servo safety controls	19
Servo queue behavior	20
DCC commands and shared addresses	21
Buttons and handlers	22
Diagnostics and brownouts	23
Troubleshooting	24-25
PART III - TECHNICAL REFERENCE	**26-35**
System architecture	26
Real-time tasks and timing	27
Hardware interfaces	28
Configuration files and recovery	29
HTTP and WebSocket interfaces	30
Flash layout and update process	31
Source tree and build environment	32
Validation and release workflow	33
Firmware change history	34
Glossary and support	35

PART I - QUICK START

Board at a glance

Prima combines an ESP32, two 16-channel PWM controllers, DCC input and physical inputs on one carrier board.



Main functional areas

Area	Purpose
ESP32 module	Runs DCC decoding, button logic, movement control, Wi-Fi and the web interface.
PCA9685 #1	Outputs 1-16 at I2C address 0x40.
PCA9685 #2	Outputs 17-32 at I2C address 0x41.
Output headers	Each channel provides V+, PWM and GND.
Physical inputs	16 terminals for switches or buttons, referenced to GND.
DCC input	Opto-isolated connection to the layout DCC bus.
5 V input	External regulated 5 V supply for controller and connected loads.
Micro USB	Firmware service connection; never use with external 5 V connected.

OUTPUT NUMBERING

The first PCA9685 serves outputs 1-16. The second serves outputs 17-32. The web interface presents one continuous 1-32 list.

PART I - QUICK START

Connect one servo

Start with a single unloaded servo on output 1. Prove the system before adding mechanisms or more channels.



Typical three-pin output group

Servo: red to V+, yellow/orange to PWM, black/brown to GND. Confirm the actual servo lead colours.

1 **Power off**

Disconnect USB, external 5 V and DCC before changing output wiring.

2 **Identify the three conductors**

Typical SG90 colours are red = 5 V, yellow/orange = PWM, black/brown = GND. Confirm the lead order for your servo.

3 **Use output 1**

Connect the servo to the output 1 V+, PWM and GND pins. Do not offset the plug by one pin.

4 **Leave the horn/linkage free**

The first movement may use default positions. An unloaded servo prevents mechanical damage while endpoints are set.

Other load wiring

Load	V+	PWM	GND
Servo	Servo positive	Servo control	Servo ground
5 V relay module	Vcc	IN	GND
LED to V+	LED long/anode	LED short/cathode	Not used
LED to GND	Not used	LED long/anode	LED short/cathode

LED PROTECTION

Use a suitable current-limiting resistor unless the chosen LED assembly already includes one. Output polarity/mode in the UI must match the wiring.

PART I - QUICK START

Power and command inputs

Prima needs regulated 5 V power plus at least one command method: DCC or a physical input.

1

Connect command input

For DCC, connect the two DCC bus wires to the DCC terminal. For a physical switch, connect an input terminal to GND through the switch.

2

Check polarity

Confirm external 5 V positive and negative at the Prima terminal. There is no reverse-polarity protection.

3

Choose the supply

For commissioning one or two micro servos, a quality regulated 5 V / 2 A supply is a sensible baseline. Increase capacity and wire gauge for larger loads.

4

Power on

Apply external 5 V with USB disconnected. Do not connect or disconnect servos while powered.

Current planning

Example	Approximate design concern	Firmware setting
One servo	Startup current and a stalled linkage	Simultaneous 1 or 2
Two paired servos	Combined startup surge	Simultaneous 2; conservative step
Many shared-address servos	Burst can exceed supply even if only two are physically fitted later	Simultaneous 1 or 2; correct mappings
Relay/LED banks	Steady load can dominate even without servo movement	Size supply for total load

DO NOT DIAGNOSE BY SUPPLY LABEL ALONE

Measure voltage at the Prima during movement. Thin leads, poor connectors and long cable runs can cause a brownout even with a high-current supply.

PART I - QUICK START

First power-on and Wi-Fi

Current firmware keeps a board-specific recovery access point available when station Wi-Fi is not usable.

1. **Find the recovery network**
   Look for cmmme_prima_XXXXX, where XXXXX is a five-digit code unique to the ESP32.
2. **Use the matching password**
   The generated password is password_prima_XXXXX using the same five-digit code.
3. **Open the controller**
   The supplied 2.0.x configuration normally uses http://192.168.25.1/. An upgraded older configuration may still use 192.168.4.1.
4. **Ignore the no-internet warning**
   The access point is local control, not an internet connection.

If Prima is already on your home/layout network

Open the address assigned by the router. The /status page reports the current IP, SSID, mode and signal strength. A static IP may be set in Node Configuration if required.

Recovery behavior

Station reconnection uses backoff so network loss does not block DCC or output processing. The firmware retains station-mode intent and can recover it from saved credentials if a configuration key is damaged.

> **OLD MANUAL DIFFERENCE**
> Firmware 2.0.x does not use the shared CMMME_****** / MyCMMME defaults described in Manual V1.1. It uses board-specific credentials.

PART I - QUICK START

Configure the first servo

Use the channel picker, then make a small safe movement with the linkage still disconnected.

1 **Open Output Configuration**
Select channel 1 and choose output type Servo.

2 **Set two positions**
Choose conservative endpoints within the global minimum/maximum. Avoid a full mechanical sweep on first test.

3 **Set movement**
Use moderate Up/Down Speed with non-zero acceleration and deceleration. Leave special hesitation/bounce features off.

4 **Apply Live**
Test each direction. Stop if the servo buzzes, stalls, reaches a hard stop or causes the controller to restart.

5 **Attach and adjust**
Fit the horn/linkage near its neutral position, then move endpoints inward before moving them outward.

Safe starting values

Setting	Start with	Why
Global min/max	150 / 860 supplied	Wide enough for commissioning; reduce for the actual mechanism.
Global speed	400	Limits trajectory speed across all servos.
Global step	4	Caps each physical PWM change.
Simultaneous	1 during first test	Only one servo can create startup load.

SAVE AFTER TESTING
Apply Live is for controlled testing. Save the configuration only after both positions and command behavior are correct.

PART I - QUICK START

Add DCC control

A DCC command selects an address and state. Every output using that address can respond.

1

Choose command type

Use a normal DCC switch/accessory command for two-state equipment. Use DCC signal commands when the command station supports multi-aspect or multi-position control.

2

Enter the address

Assign the accessory or signal address to the configured output.

3

Send both states

Operate Throw/Close (or the relevant aspect) from the command station while watching the intended output.

4

Confirm the health indicator

The DCC status should show recent signal activity. If not, troubleshoot the input before changing output settings.

Shared addresses

It is valid to give multiple outputs the same DCC address. For example, a servo, frog-polarity relay and panel LED can respond to one turnout command.

Desired behavior	Configuration
One turnout servo	One servo output uses address 101.
Two servos move as a pair	Both servo outputs use address 101; simultaneous limit 2.
Turnout + relay + LED	All three outputs use address 101 with their own type-specific states.
Independent equipment	Use different addresses.

UNEXPECTED MASS MOVEMENT

If many outputs respond, inspect duplicate addresses and button mappings. Do not assume the controller is randomly powering all channels.

PART I - QUICK START

Add a physical button

A physical input is mapped to a logical command. Outputs respond through their configured command address.

1 **Wire one input**
Connect input 1 to GND through a momentary button or suitable switch.

2 **Configure the input**
In Physical Inputs, select input 1 and the required button behavior, thresholds and event mask.

3 **Configure one handler**
Map that button event to the same logical command/address used by the desired output.

4 **Test once**
Press the button and confirm only the intended address group responds.

The important mental model



The web interface edits configuration; real-time tasks continue to service DCC and outputs independently.

Do not create one handler per physical output unless each needs a different command. If outputs 1 and 4 both use address 101, a single handler that emits address 101 is enough to move both.

WHY BROAD BUTTON MAPPINGS CAUSED BROWNOUTS
A button configuration can fan out to many output addresses in one scan. Firmware 2.0.25 sends every servo request through the same queue and holds excess commands until a motion slot is available. The mapping should still express only the intended command.

PART I - QUICK START

Control two servos together

The scheduler allows a paired movement while preventing a larger command burst from starting every servo at once.



Limit 2: commands are accepted immediately; physical movement is staggered.

1 **Give both outputs the same address**
For example, outputs 1 and 4 both use DCC accessory address 101.

2 **Set Simultaneous Servos to 2**
Both intended servos may begin together. Any third servo waits.

3 **Use one DCC or button command**
Send address 101 once. Do not duplicate the button handler across unrelated outputs.

4 **Observe power and motion**
Confirm supply voltage remains stable and both mechanisms travel freely.

Known-good tested profile

Global speed	Global step	Simultaneous
800 counts/s	4 counts/frame	2 servos

The 2.0.17 scheduler baseline completed six paired output 1 and 4 movements from 185 to 850 and back during a 30-minute monitored hardware test. The boot, brownout and watchdog counters did not change, and both motion slots cleared after every cycle. Release 2.0.18 extended that scheduler to browser preview moves; 2.0.25 retains it.

This is an installation-specific result, not a universal maximum. The factory-safe fallback remains 400 / 4 / 2. Use lower values for weak supplies, long wiring, stiff mechanisms or high-current servos.

ZERO MEANS UNRESTRICTED
Setting speed, step or simultaneous to 0 disables that limit. It is not a low or off-motion value.

PART I - QUICK START

Save, back up and commission

A working configuration is valuable. Save it, export it, then test the worst expected operating case.

1. **Save**
   Save the tested output, button and network settings. Allow a requested restart to complete.
2. **Export a full backup**
   Keep the backup off the controller. It should include node, Green Hat, both switch files, button file and all handler files.
3. **Test every command source**
   Operate each DCC address and each physical input in both states.
4. **Test the largest burst**
   Trigger the command group most likely to produce peak current and verify orderly queuing.
5. **Check status**
   Confirm firmware version, DCC/PCA health, reset reason and the 800 / 4 / 2 or chosen limits.

Commissioning checklist

- No servo reaches a mechanical stop.
- No relay chatters and no LED exceeds its rating.
- DCC commands reach only the intended address groups.
- Each physical input creates only its intended logical command.
- No restart occurs during the largest expected burst.
- A dated configuration backup exists and can be opened as JSON.

BROWSER CACHE
After a firmware/filesystem update, press Ctrl+F5 once if new controls do not appear. Static assets may be cached for several minutes.

PART II - COMPLETE OPERATION

Interface and health

The web interface is a configuration client. The controller continues real-time DCC and output work without an open browser.

Area	Use
Node Configuration	Wi-Fi mode, static IP, device identity, NTP and diagnostics option.
Output Configuration	All 32 LED, relay and servo channels plus global servo settings.
Physical Inputs	16 button/switch definitions.
Button Handlers	Maps input events to logical commands and optional enabling conditions.
Diagnostics	Opt-in live maintenance information.
Health strip	Lightweight DCC and PCA state shown throughout the interface.

Health meanings

Indicator	Healthy	Unhealthy action
PCA	Both 0x40 and 0x41 respond; outputs may enable.	Check board seating, I2C, power and PCA addresses.
DCC	Recent input edges/packets detected.	Check DCC bus wiring and command station output.
Outputs	Configuration valid, PCA verified and startup delay complete.	Read /status; malformed config or missing PCA locks output.
Wi-Fi	Station connected or recovery AP active.	Use recovery AP; repair station credentials.

FAIL-SAFE OUTPUT LOCK

The firmware does not enable outputs merely because configuration files exist. Configuration must validate and both PCA9685 devices must respond.

PART II - COMPLETE OPERATION

Node and Wi-Fi configuration

Choose local AP operation, station operation, or the supported combined behavior without making layout control depend on Wi-Fi.

Setting	Meaning	Guidance
Wi-Fi mode	AP, station or combined mode.	Use station for normal network access; retain recovery access.
Device name	Friendly Prima identity.	Include MAC if several units share a layout.
Static IP	Fixed station address.	Use only outside the router DHCP pool or reserve it.
AP gateway	Recovery AP address.	Supplied config: 192.168.25.1.
AP credentials	Generated from the ESP32 identity.	SSID/password share the same five-digit suffix.
Wi-Fi timeout	Optional shutdown after inactivity.	Leave Wi-Fi always on for frequent browser access.
NTP	Network time service.	Optional; not required for DCC/output operation.
Diagnostics	Shows maintenance page/API.	Enable only while investigating or monitoring.

Recovery logic

- The selected station/AP intent is retained in NVS separately from LittleFS.
- Last-known-good credentials and the ESP32 Wi-Fi profile can recover station intent.
- Reconnect uses backoff; DCC and servo tasks do not wait for network operations.
- Hostnames are sanitized before being applied.
- Wi-Fi restarts are deferred outside HTTP/WebSocket handlers.

PASSWORD HANDLING

Passwords are masked in logs and UI status. Treat exported node configuration and Wi-Fi credentials as sensitive layout information.

PART II - COMPLETE OPERATION

The output model

Each of 32 channels combines a command source, address conditions, device type, states and a movement profile.

Field	Purpose
CmdSource	Selects switch/accessory, signal/aspect or other supported command semantics.
Addr	One or more logical addresses watched by the output.
CondData	Command-state data used to select output behavior.
DevType	Servo, Relay or LED.
Positions	Target PWM values and movement configuration for each state/aspect.
UpSpeed / DownSpeed	Requested directional travel/transition speed.
AccelRate / DecelRate	Change of speed near start/end.
PowerOff	Optionally stops PWM holding after completion.
Momentary / PulseMs	Relay or LED pulse mode and duration, 20-60000 ms (default 250 ms).
MomentaryIdle	Resting relay level or LED brightness, 0-4095.
Frequency / Lambda	Bounce/oscillation frequency and damping.
UseHesi / HesPoint / HesSpeed	Optional hesitation point and reduced speed.

Files and channel ranges

The 32 supplied outputs are split across switches.cfg (23 drivers) and switches1.cfg (9 drivers). The browser assembles them into one list and does not declare Ready until all 32 are present and validated.

RANGE ENFORCEMENT
Servo targets are constrained by global Servo Min/Max. Relay and LED PWM values may use the full 0-4095 PCA range.

PART II - COMPLETE OPERATION

LED and relay outputs

These output types share the PCA hardware but use different connection and state models.

LED outputs

- Wire the LED to V+/PWM or PWM/GND and select the matching mode; always use suitable current limiting.
- The level slider controls brightness; wiring polarity can make a lower numeric value appear brighter.
- Maintained mode follows each configured event level. Momentary mode applies the selected event brightness for 20-60000 ms, then returns to the configured resting brightness (0-4095).

Relay outputs

- Use a compatible 5 V relay module: Vcc to V+, IN to PWM and GND to GND.
- Maintained mode follows the configured binary state. For a pulse, enable Momentary, choose resting Low or High, and set a 20-60000 ms duration (250 ms default). Low/High is the electrical IN level, not necessarily relay Off/On; choose the level that leaves the module unenergised.
- Every accepted DCC state or button-generated toggle produces the same pulse opposite rest: Low rest pulses High; High rest pulses Low. Another accepted state change during a pulse restarts the timer; repeated packets from the same DCC burst are suppressed.
- Startup and configuration reload place the output at rest without replaying an old activation. Older files without MomentaryIdle use their first event as the fallback rest value.

QUICK RELAY SETUP
Select Relay > enable Momentary > choose Resting output > set Pulse duration > confirm both command events > Apply Live. Test from DCC and the physical button before connecting the final load.

Mixed command group example

Output	Type	Address 101 response
1	Servo	Moves turnout blades.
2	Relay	Changes frog polarity.
3	LED	Changes panel indication.

SUPPLY BUDGET
Relay coils and LED banks add steady load. The simultaneous-servo limit controls only active servo movements; it does not cap total 5 V current.

PART II - COMPLETE OPERATION

Servo basics

Servo PWM positions are 12-bit PCA counts, not degrees. Mechanical travel must be commissioned on the actual installation.

Control	Effect	Practical advice
Position	Target PCA count for a command state.	Start near centre and expand travel gradually.
Up/Down Speed	Requested rate by movement direction.	The global speed ceiling may reduce it.
Acceleration	How quickly requested speed is reached.	Avoid zero; gentler starts reduce shock.
Deceleration	How movement slows near target.	Increase for a softer arrival.
Power Off	Stops holding PWM after completion.	Reduces idle current but may allow drift.
Initial state	Target selected after configuration/startup.	Ensure startup cannot drive into a stop.

Endpoint procedure

- Disconnect the linkage or place it at a safe neutral point.
- Move to state A; adjust only until the mechanism reaches the required position.
- Move to state B; repeat without forcing the servo.
- Reduce the global min/max if the hardware should never travel farther.
- Cycle at least ten times and check heat, noise and repeatability.

PWM IS NOT ANGLE

The same count may produce a different angle on another servo. Do not copy endpoint values blindly between servo models or mechanisms.

PART II - COMPLETE OPERATION

Advanced servo movement

Hesitation and bounce effects can model semaphore and mechanical motion, but should be added only after safe endpoints are proven.

Feature	Meaning	Use
Soft start/stop	Uses acceleration/deceleration around the move.	Turnouts and linkages needing reduced shock.
Hard movement	More direct transition.	Only where the mechanism tolerates it.
Bounce	Overshoots/oscillates around the target.	Semaphore arm simulation.
Frequency	Oscillation rate.	Sets how quickly bounce cycles.
Lambda	Damping factor.	Controls how quickly oscillation dies away.
Hesitation point	Position where motion changes or pauses.	Mechanical pinch-point simulation.
Hesitation speed	Reduced speed near the hesitation point.	Fine movement through the selected region.

Commissioning sequence

- Prove both endpoints with plain soft movement.
- Enable one special behavior at a time.
- Use a small bounce amplitude before attaching the signal mechanism.
- Check the global step limit: a low step can dominate the apparent frequency.
- Retest after changing supply, servo model or linkage geometry.

LATEST-COMMAND COALESCING

If a new command arrives while moving, the output retains the latest requested state. Intermediate reversals may be skipped to avoid unnecessary rapid direction changes.

PART II - COMPLETE OPERATION

Global servo safety controls

Open the collapsed Global servo settings header only when changing limits that affect every servo.

Setting	Range	Default	What and why
Servo Min Pos	0-4095	150 supplied	Lowest permitted servo target; protects against travel below the mechanism's safe range.
Servo Max Pos	0-4095	860 supplied	Highest permitted target; prevents excess travel.
Global Speed	0-65535 counts/s	400	Caps requested movement rate; lower values reduce abrupt load.
Global Step	0-4095 counts/frame	4	Caps each published PWM change; smooths current spikes.
Simultaneous	0-32 servos	2	Caps active servo movements; extra commands wait.

How the ceilings combine

Per-output speed requests the motion. The global speed limit caps that request. The step limit caps the actual PWM delta published in one PCA frame. The simultaneous limit decides whether the trajectory may start now or must wait.

Tuning order

- Set safe min/max first.
- Commission with simultaneous 1.
- Raise step gradually until motion is acceptably responsive.
- Raise speed only if the per-output request is being capped.
- Set simultaneous 2 for paired servos, then test the largest command burst.

UNRESTRICTED VALUES

Zero disables speed, step or simultaneous protection. Use only when the electrical and mechanical implications are understood.

PART II - COMPLETE OPERATION

Servo queue behavior

The scheduler accepts command bursts immediately but limits how many servo trajectories are physically active.



Limit 2: commands are accepted immediately; physical movement is staggered.

Rules

- A servo consumes one slot while its PWM trajectory is moving.
- LED and relay transitions do not consume servo slots.
- A servo already at target does not hold a sustained movement slot.
- Waiting work starts when an active servo finishes.
- The most recent requested state is retained for a moving or waiting output.
- Lowering the limit does not abruptly stop a servo that is already moving.

Examples with limit 2

Requested group	Result
Two servos	Both start together.
Three servos	Two start; the third waits.
Six servos	Move in batches as slots become free.
Two servos + relay + LED	Both servos can move; relay and LED respond independently.

PROTECTION AND INTENT ARE SEPARATE

The queue protects the supply from a servo-start burst. It does not correct a mistaken address or button-handler mapping.

PART II - COMPLETE OPERATION

DCC commands and shared addresses

DCC accessory commands are continuously decoded. Signal/aspect commands extend the same address model to multiple states.

Mode	States	Typical use
Switch/accessory	Throw / Close	Turnout, two-position servo, relay, two-state indicator.
Signal/aspect	Multiple aspect values	Multi-aspect signal or multi-position servo.

Real-time behavior

- The DCC waveform interrupt is attached before Wi-Fi startup on GPIO2.
- A dedicated task drains/decodes packets every 1 ms.
- Accessory and signal command processing is not throttled by browser activity.
- The old live DCC Viewer was removed from production firmware to avoid runtime load.
- A standalone PrimaDccViewer firmware exists for bench diagnostics when required.

Address fan-out

One command is evaluated against all configured outputs. Shared addresses are therefore a feature, not an error. Use them intentionally for groups; use unique addresses for independent hardware.

DCC INACTIVE
Check the track bus at the input terminals, command-station output and /status DCC counters before changing servo settings.

PART II - COMPLETE OPERATION

Buttons and event handlers

The physical input layer detects events; the handler layer converts an event into a control command.

Layer	Important fields	Purpose
Button definition	PortNr, PinNr, ButtonType, EventMask, ButtonAddr	Defines 16 physical inputs and which press/release/click events exist.
Timing	HoldThreshold, DblClickThreshold, RefreshInterval, Sensitivity	Shapes switch behavior and filtering.
Handler	ButtonNr, CtrlCmd	Selects which detected event emits which command.
Condition	EnableSource, EnableAddr, EnableState	Optionally permits a handler only when another state is true.

Configuration file capacity

The supplied configuration contains 16 button definitions and 64 handlers split across btnevt.cfg, btnevt1.cfg and btnevt2.cfg. This split is normal and is preserved by full backup/restore.

Debugging unexpected action

- Identify the last button event in Diagnostics.
- Find every handler using that ButtonNr.
- Inspect each CtrlCmd and address.
- Find every output that watches those addresses.
- Remove duplicates or narrow enable conditions.
- Retest with simultaneous servo limit 1 or 2.

ONE HANDLER CAN CONTROL A GROUP

If multiple outputs intentionally share the same address, one handler command controls the entire group. Do not multiply handlers by output count.

PART II - COMPLETE OPERATION

Diagnostics and brownouts

Diagnostics are opt-in and designed to preserve evidence from intermittent resets without burdening normal service.

Evidence	Interpretation
Reset reason	Brownout, watchdog, software restart or another ESP32 reset source.
Boot/brownout/watchdog counters	Persistent history stored in Preferences/NVS.
Last event	Most recent button or DCC accessory/signal event.
Motion breadcrumb	Output number, current/target position, requested/applied PWM and moving state.
Active / limit	Current active servo moves and configured concurrency ceiling.
Timing counters	Output/DCC service cycles, maximum runtime/gap, overruns and stack margin.
PCA write errors	I2C output communication failures and dirty retries.

Brownout breadcrumb design

An alternating, checksummed record is kept in ESP32 RTC memory. It avoids a flash write during motion and can survive a brownout reset long enough to report the immediately preceding event and output state.

Recommended investigation

- Record /status before reproducing the fault.
- Trigger once, then read reset reason and preReset immediately.
- Repeat with simultaneous limit 1.
- Measure 5 V at the board during movement.
- Separate mapping faults from electrical resets.
- Disable maintenance diagnostics when no longer required.

PART II - COMPLETE OPERATION

Troubleshooting - operation

Start with the symptom. Avoid changing several unrelated settings at once.

Symptom	Likely area	Action
Reset at servo start	Peak current / wiring drop	Limit 1, lower speed/step, measure 5 V, inspect stall and connectors.
DCC works; button triggers many outputs	Handler/address fan-out	Trace ButtonNr -> CtrlCmd -> every matching output.
Relay pulses every other command	Old firmware/web pair	Install 2.0.25 application and matching LittleFS, then Ctrl+F5.
Momentary output rests wrong	Resting output setting	Choose Relay Low/High or LED resting brightness, then Apply Live.
Some commands act later	Expected queueing	Compare requested servo count with simultaneous limit.
Motion jumps	High step or delayed service	Lower step; inspect Diagnostics timing and PCA errors.
Servo buzzes at endpoint	Mechanical stop / holding PWM	Move endpoint inward; consider PowerOff after move.
Wrong output group moves	Duplicate address	Audit addresses and signal/aspect conditions.
PCA not ready	I2C/output hardware	Check main-board/PCA power, seating, SDA/SCL and 0x40/0x41; after USB-only ESP reset, power-cycle the complete board if needed.
DCC inactive	Input path	Check DCC terminal, optocoupler path and command station.

A BROWNOUT IS EVIDENCE, NOT A DIAGNOSIS

Reset reason 9 confirms the ESP32 saw low supply voltage. The cause can still be a stalled servo, wiring drop, poor connector, weak supply or an unintended command burst.

PART II - COMPLETE OPERATION

Troubleshooting - web and updates

Firmware and web assets are separate flash regions and must remain matched.

Symptom	Action
New fields missing	Press Ctrl+F5. Confirm LittleFS was updated as well as the application.
Page reports wrong output count	Do not operate outputs. Reload; Apply Live/restore must provide exactly 32 valid drivers, including unused channels.
Cannot reach station IP	Join cmmme_prima_XXXXX recovery AP and open its gateway.
Recovery AP details unknown	Use the board-specific five-digit suffix for SSID/password; inspect serial/status if available.
Diagnostics page is 404	Enable Maintenance diagnostics in Node Configuration, save, then reload.
DCC Viewer is 404	Expected in production firmware; use Diagnostics or standalone viewer firmware.
Configuration restore rejected	File is malformed, unsupported or exceeds safety limits; validate JSON and backup contents.
Upload stops during filesystem erase/write	Disconnect external 5 V/DCC, reconnect USB only, use a lower baud or verified chunked tool.
COM port visible but no bootloader	Power-cycle USB only; use BOOT/EN sequence or port-recovery procedure.

After any update

- Verify firmware version in /status.
- Confirm web controls match the release.
- Restore/verify configuration before powering mechanisms.
- Check PCA and DCC health.
- Test one servo, then the intended burst.
- Keep the previous verified application and filesystem images together for rollback.

PART III - TECHNICAL REFERENCE

System architecture

The project isolates time-sensitive control from web, Wi-Fi and storage work.



The web interface edits configuration; real-time tasks continue to service DCC and outputs independently.

Data flow

Stage	Implementation
Input capture	NmraDcc ISR for GPIO2 waveform plus IoTT button scanner.
Decode/event	Dedicated DCC task and button/event handler logic produce logical commands.
Model	IoTT_SwitchList holds 32 configured device models and target states.
Scheduling	Global limits, per-output trajectories, latest-command coalescing and concurrency queue.
Publish	Dirty PCA values flush about once per hardware PWM frame.
Observability	/status, opt-in diagnostics, RTC breadcrumb and persistent counters.
Configuration	LittleFS JSON plus browser HTTP/WebSocket client.

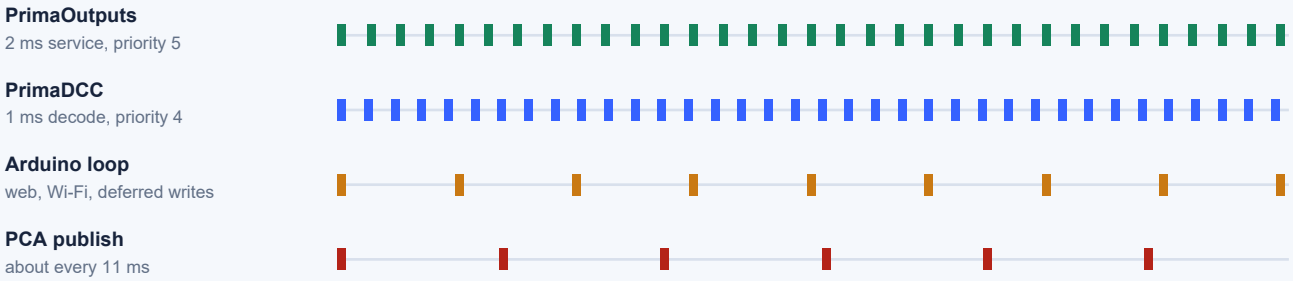
Failure boundaries

- Wi-Fi failure does not stop DCC processing.
- Browser traffic does not own output timing.
- Malformed output configuration locks output enable.
- Missing PCA hardware locks output enable but input/service loops continue.
- Flash writes are deferred away from motion processing.

PART III - TECHNICAL REFERENCE

Real-time tasks and timing

Both dedicated tasks are pinned to the Arduino running core; motion has higher priority than decode while the DCC ISR continues capturing.



Task	Cadence	Priority / stack	Responsibility
PrimaOutputs	2 ms delay	5 / 8192	Process motion, buttons, output enable, brownout snapshot.
PrimaDCC	1 ms delay	4 / 4096	Drain and decode DCC packets.
Arduino loop	Cooperative	watchdog 10 s	Wi-Fi housekeeping, WebSocket cleanup, deferred state writes.
PCA publish	about 11 ms	92 Hz hardware	Flush pending channel PWM values once per hardware frame.

Non-catch-up pacing

The output task uses vTaskDelay after each pass rather than vTaskDelayUntil. A long pass therefore does not cause hundreds of back-to-back catch-up calls that could starve DCC and web work.

I2C behavior

The bus runs at 400 kHz with a 5 ms timeout. Failed PCA writes remain dirty and are retried rather than freezing the motion service.

PART III - TECHNICAL REFERENCE

Hardware interfaces

These are the firmware-visible connections. Board-level electrical details remain tied to the installed Prima PCB revision.

Interface	Assignment	Notes
DCC receive	ESP32 GPIO2	Strapping/ADC2-capable pin; not reused as status LED.
I2C SDA	GPIO21	Two PCA9685 devices and associated board functions.
I2C SCL	GPIO22	400 kHz Fast-mode.
PCA #1	I2C 0x40	Outputs 1-16.
PCA #2	I2C 0x41	Outputs 17-32.
PWM frequency	about 92 Hz	Chosen for analog servo range and near-100 Hz measured behavior.
Physical inputs	16 board terminals	Processed by IoTT_Mux64Buttons definitions.
DCC electrical	Opto-isolated PCB input	Separates track-bus voltage from ESP32 logic.
Power	Regulated 5 V DC	External input not regulated or reverse-polarity protected.

Startup ordering

Firmware mounts LittleFS and reads node configuration, initializes I2C, then attaches the DCC receiver before starting Wi-Fi. This order matches the production board behavior required for reliable GPIO2 DCC interrupts.

GPIO2 CAUTION

GPIO2 is both DCC input and an ESP32 strapping pin. External circuitry and service procedures must not force an invalid boot level.

PART III - TECHNICAL REFERENCE

Configuration files and recovery

Configuration is JSON in LittleFS. Numbered files extend collections beyond one file without changing the browser's logical model.

Path	Contents
/configdata/node.cfg	Wi-Fi, device identity, interface selection, NTP, diagnostics.
/configdata/greenhat.cfg	Module list and Green Hat configuration file references.
/configdata/gh/0/switches.cfg	Outputs 1-23 plus global servo fields.
/configdata/gh/0/switches1.cfg	Outputs 24-32.
/configdata/gh/0/btn.cfg	16 physical input definitions and thresholds.
/configdata/gh/0/btnevt.cfg	First 25 button handlers.
.../btnevt1.cfg	Next 26 button handlers.
.../btnevt2.cfg	Final 13 button handlers.
/www/*	HTML, CSS, JavaScript and images served by the controller.

Atomic writes

A new file is written as .tmp, flushed, then the previous generation is renamed .bak before the new file is promoted. If parsing fails at boot, the invalid file is quarantined as .bad and the previous .bak generation is restored when possible.

SCHEMA AND FIRMWARE VERSIONS DIFFER

Configuration Version values describe JSON schema. PRIMA_FIRMWARE_VERSION identifies the application release. Do not assume they should match.

PART III - TECHNICAL REFERENCE

HTTP and WebSocket interfaces

The web app uses HTTP for pages/config reads and a serialized WebSocket for live status, tests, saves and backup transfer.

Route	Method	Purpose
/	GET	Main interface.
/status	GET	Lightweight firmware, network, reset, health and servo scheduler JSON.
/api/diagnostics	GET	Compact diagnostics when enabled.
/api/diagnostics?verbose=1	GET	Engineering channel detail when enabled.
/api/config?name=...	GET	Reads an allowed configuration file.
/wifi/setup	GET	Wi-Fi setup page.
/wifi/scan	GET	Available network scan.
/wifi/connect	POST	Submits station credentials.
/wifi/status	GET	Current Wi-Fi state.
/wifi/restart	request	Deferred Wi-Fi restart.
/healthz	GET	Simple service health response.
/ws	WebSocket	Live status, test commands, saves and configuration transfer.

Transport safeguards

- WebSocket configuration payloads may be up to 64 KiB.
- All sends are serialized and checked for client readiness.
- Backup transfer is exclusive so live events cannot interleave with file frames.
- Busy backup frames are retried.
- HTTP concurrency is bounded; rejected request count appears in status.
- Static assets may be cached, while status/diagnostics responses are no-store.

PART III - TECHNICAL REFERENCE

Flash layout and update process

The 4 MB image contains boot data, NVS, two application slots and a LittleFS partition.



app0/app1: 0x140000 each LittleFS: 0x290000, size 0x170000 4 MB flash layout

Region	Offset	Size	Use
NVS	0x9000	0x5000	Preferences, Wi-Fi intent and counters.
OTA data	0xE000	0x2000	Boot/OTA selection metadata.
app0	0x10000	0x140000	Primary application slot.
app1	0x150000	0x140000	Second application slot.
LittleFS	0x290000	0x170000	Configuration and web interface.

Update choices

Method	Changes	Risk/control
Application only	Firmware binary.	Preserves LittleFS but new UI fields may be missing.
LittleFS only	Web/config defaults image.	Replaces live configuration; back up and restore.
Full upload	Bootloader, partition table, app and LittleFS.	Most complete; take a full backup first.
Chunked flash	Writes verified smaller blocks at low baud.	Useful for marginal serial links/recovery.

2.0.25 CUSTOMER UPDATER IS A COMPLETE FACTORY FLASH

Use the Windows updater for original-firmware upgrades. It erases NVS, settings and Wi-Fi, then writes bootloader, partition table, application and clean factory LittleFS. Back up first, use USB as the only power source, and reconfigure all 32 outputs afterward. It never changes the PC Wi-Fi connection.

PART III - TECHNICAL REFERENCE

Source tree and build environment

The repository pins the ESP32 core and libraries so a verified build can be reproduced.

Path	Role
Sketchbook/LC2021/	Production sketch, configuration defaults and web assets.
Sketchbook/libraries/	Project-pinned Arduino dependencies.
Sketchbook/releases/	Verified application-only release images and extracted content.
Sketchbook/PrimaBasicRecovery/	Minimal recovery firmware; does not drive outputs.
Sketchbook/PrimaDccViewer/	Standalone DCC diagnostic firmware without Wi-Fi/output load.
validate_project.ps1	Validates JSON, exactly 32 outputs, ranges and JavaScript syntax.
upload_all.ps1	Compiles and flashes firmware/partition/LittleFS.
flash_full_chunked.ps1	Low-baud verified chunked recovery flash.
device_config_roundtrip.mjs	Exercises API backup and exact no-op save round trip.

Pinned toolchain

Component	Version / target
Arduino-ESP32 core	2.0.17
ESPAsyncWebServer	3.10.3
AsyncTCP	3.4.10
Board FQBN	esp32:esp32:esp32
Filesystem	LittleFS built with mklittlefs
Flash utility	esptool.py 4.5.1 in the installed core

PART III - TECHNICAL REFERENCE

Validation and release workflow

A release is not complete until configuration, browser code, firmware and rendered user documentation all agree.

1. **Validate source data**
   Run validate_project.ps1. It checks every JSON file, 32 outputs, servo bounds, versioned browser assets and JavaScript syntax. Run both Node regression suites.

2. **Compile with pinned dependencies**
   Build for esp32:esp32:esp32 using Arduino-ESP32 2.0.17 and the project library directory.

3. **Back up the live device**
   Export all configuration before replacing LittleFS or the partition table.

4. **Flash and verify hashes**
   Use esptool verified writes. Reduce baud or use chunked writes if the serial link is unstable.

5. **Restore and compare**
   Restore device-specific configuration, with node.cfg last, then read files back.

6. **Functional test**
   Verify /status, DCC/PCA health, one output, shared-address group, physical button and maximum burst.

Current 2.0.25 build result

Metric	Result
Application image	1,079,072 bytes; SHA-256 bae2235d48191bdda5bf7b2c035e7e4462b571fca69cc134075becbabf348a71
LittleFS image	1,507,328 bytes; SHA-256 ae21ff6edc8627268267023af54e1efc589d35476f478de148aabde1508f4629
Configuration validation	Exactly 32 factory outputs and matching 2.0.25 browser assets passed
Hardware baseline	Six 185-850-185 cycles at 800 / 4 / 2 over 30 minutes on 2.0.17; no new reset

RELEASE NUMBERING
Increment patch for each testable firmware change; minor for compatible feature sets; major for incompatible configuration/hardware changes.

PART III - TECHNICAL REFERENCE

Firmware change history

The 2.0 series progressively removed timing, transfer and observability failure modes from the original firmware.

Release	Main change
2.0.25	Every accepted Relay DCC/button state produces the same pulse opposite rest; repeated packet bursts are coalesced.
2.0.24	Relay Low/High and LED 0-4095 resting levels are configurable and applied at startup, reload and pulse expiry.
2.0.23	Added momentary Relay/LED mode with configurable 20-60000 ms pulse duration.
2.0.22	Corrected PCA relay full-on/full-off output handling and relay previews.
2.0.21	Apply Live accepts chunked frames, retries acknowledgements and preserves valid files while PCA hardware recovers.
2.0.20	Apply Live inventories actual fragments; cleanup failures are reported and the browser waits up to 15 seconds.
2.0.19	Apply Live/restore write first, remove only obsolete tail fragments and stop on transfer errors.
2.0.18	Diagnostics, desktop jump and relay fixes; queued previews; strict config and multi-tab receive safety.
2.0.17	Button bursts wait for global servo capacity; delayed hand-off prevents same-pass release bursts.
2.0.16	Collapsed and explained Global servo settings.
2.0.15	Configurable simultaneous-servo limit and queued starts.
2.0.14	Editable global speed/step limits with safe fallbacks.
2.0.13-2.0.0	Motion pacing, brownout diagnostics, Wi-Fi recovery, transfer reliability and lightweight health foundation.

2.0.13 was a short experiment that removed the hard 400/4 clamps; 2.0.14 replaced it with explicit editable global limits.

REFERENCE

Glossary and further support

Key project terms plus a direct route to current CMMME support information.

Term	Meaning
PCA count	12-bit 0-4095 PWM position/value.
Frame	One PCA hardware update period, about 10.9-11 ms.
Brownout	ESP32 reset caused by supply voltage falling below its safe threshold.
DCC accessory	Two-state switch/turnout command.
DCC signal/aspect	Multi-state command associated with one address.
Handler	Rule converting a physical input event into a command.
LittleFS	Flash filesystem containing configuration and the web UI.
NVS	ESP32 non-volatile key/value store for preferences and recovery state.
RTC memory	Small retained memory used for pre-reset breadcrumbs.

FURTHER SUPPORT

For current contact details, product assistance and further support, visit the CMMME contact page.

https://cmmme.co.uk/contact/

Scan the QR code with a phone or tablet to open the same page.

BEFORE REQUESTING SUPPORT

Record the firmware version, reset reason, power-supply rating, global servo settings, affected output/address and the steps that reproduce the problem.

End of CMMME Prima Complete Manual 2.0.25

Retain this manual with the matching application, LittleFS image and configuration backup.