Client Bitmask Reference (ESP32-WROOM)

Version: v01
Context: Filament-Silicagel-Dryer – Client / ClientComm
Scope: Explains how `outputsMask` bits are interpreted and which physical outputs are switched.

---

1. Overview

This document describes how the **CLIENT (ESP32-WROOM)** interprets bitmasks received from the HOST via the UART protocol.

- The HOST sends commands (`SET` , `UPD` , `TOG`) containing a **16-bit mask**
- The CLIENT:
- Stores the mask internally (`_outputsMask`)
- Applies bits **0–7** to physical GPIOs
- Ignores bits **8–15** (currently unused)
- Actual GPIO switching is implemented in `Client.cpp`
- Protocol handling is implemented in `ClientComm.cpp`

---

2. Bitmask Basics

- Mask size: **16 bit**
- Representation in protocol: **4-digit HEX**
- Example:
- `0x0015` = binary `0000 0000 0001 0101`
- Active bits: **0, 2, 4**

Only bits **0–7** are mapped to outputs.

---

3. Bit → GPIO → Powerboard Mapping

Bit	Logical Name	GPIO	Powerboard Pin	Function	Special Logic
0	OVEN_FAN12V	GPIO32	P5	Fan 12V (Powerboard cooling)	Always direct
1	OVEN_FAN230V	GPIO33	P7	Fan 230V (FAST)	Direct
2	OVEN_LAMP	GPIO25	P8	Lamp 230V	Door-independent
3	OVEN_SILICAT_MOTOR	GPIO26	P9	SilicaGel Motor	**Only if Door CLOSED**
4	OVEN_FAN230V_SLOW	GPIO27	P10	Fan 230V (SLOW)	Direct
5	OVEN_DOOR_SENSOR	GPIO14	P12	Door input	**Input only**
6	OVEN_HEATER	GPIO12	P6	Heater enable	**PWM 4 kHz / 50%, Door-gated**
7	(reserved)	GPIO13	—	Reserved	Direct (currently unused)

---

4. Door Logic (Safety)

- Door signal:
- **CLOSED = LOW (GND)**
- **OPEN = HIGH (+5 V)**
- Logic:
- `Door OPEN` → Heater **OFF**, Motor **OFF**
- `Door CLOSED` → Heater/Motor allowed

This safety gating is implemented **only in the CLIENT**.

---

5. Heater PWM Logic

The heater relay **must not be driven with DC**.

Measured requirement (T9):

- Frequency: **4 kHz**
- Duty cycle: **50 %**
- Period: $_{252}$ µs

Implementation details:

- PWM generated via ESP32 LEDC
- PWM runs **only while heater bit is set AND door is closed**
- On stop/fault:
- PWM detached
- GPIO driven LOW (safe state)

---

6. Command Semantics (Client Perspective)

SET

```
H;SET;MMMM
```

- Replaces entire `outputsMask`
- Immediately applied to GPIOs
- Client responds with:

```
C;ACK;SET;MMMM
```

UPD

```
H;UPD;SSSS;CCCC
```

- `SSSS` : bits to set
- `CCCC` : bits to clear
- New mask = `(old | SSSS) & ~CCCC`
- Applied immediately
- Client responds with:

```
C;ACK;UPD;MMMM
```

TOG

```
H;TOG;TTTT
```

- Toggles bits in `TTTT`
- New mask acknowledged via:

```
C;ACK;TOG;MMMM
```

⚠️ **Known issue (T9):**

- In the current implementation, `TOG` updates `_outputsMask`
- but **does not trigger the OutputsChangedCallback**
- Result: ACK/log looks correct, but GPIO state may not change

---

7. STATUS Reporting

Command:

```
H;GET;STATUS
```

Response:

```
C;STATUS;<mask>;<adc0>;<adc1>;<adc2>;<adc3>;<temp>
```

Important:

- `<mask>` = **requested outputsMask**
- Not necessarily identical to physical state
- Example: Heater bit set while door open → mask=1, heater physically OFF

---

8. Log Interpretation Example

Log line:

```
[outputsChangedCallback] outputsMask=0x            21
```

Explanation:

- `21` is **decimal**
- Decimal 21 = Hex `0x0015`
- Active bits: **0, 2, 4**
- FAN12V
- LAMP
- FAN230V_SLOW

---

9. Summary

- Bits **0–7** control real hardware
- Client enforces **safety rules** (Door-gating, Heater PWM)
- Logs show **logical mask**, not always physical reality
- This document is the authoritative reference for interpreting client bitmasks

---

End of document

Appendix A – Bitmask Effects Matrix (Client Perspective)

This appendix provides a **tabular, deterministic interpretation** of all **individual bit effects** as implemented in the CLIENT.
It is intended as a **log-reading and debugging aid**.

> Scope:
>
> - Bits 0–7
> - Physical effect on GPIO / Powerboard
> - Influence of **Door state**
> - Logical mask vs. physical output

A.1 Legend

- **Mask Bit**: Bit position in `outputsMask`
- **Mask = 1**: Bit is set in `outputsMask`
- **Mask = 0**: Bit is cleared in `outputsMask`
- **Door CLOSED**: Door input LOW (GND)
- **Door OPEN**: Door input HIGH (+5 V)

A.2 Bit Effect Table

Bit	Logical Name	Mask	Door State	Physical Output	Notes
0	OVEN_FAN12V	0	any	OFF	Direct output
0	OVEN_FAN12V	1	any	ON	Powerboard cooling fan
1	OVEN_FAN230V	0	any	OFF	Direct output
1	OVEN_FAN230V	1	any	ON	230V fan (FAST)
2	OVEN_LAMP	0	any	OFF	Door-independent
2	OVEN_LAMP	1	any	ON	Lamp 230V
3	OVEN_SILICAT_MOTOR	0	any	OFF	Direct logic
3	OVEN_SILICAT_MOTOR	1	CLOSED	ON	Allowed only if Door CLOSED
3	OVEN_SILICAT_MOTOR	1	OPEN	OFF	**Door safety override**
4	OVEN_FAN230V_SLOW	0	any	OFF	Direct output
4	OVEN_FAN230V_SLOW	1	any	ON	230V fan (SLOW)
5	OVEN_DOOR_SENSOR	0	CLOSED	LOW	Input only
5	OVEN_DOOR_SENSOR	1	OPEN	HIGH	Reflected in STATUS only
6	OVEN_HEATER	0	any	OFF	PWM disabled
6	OVEN_HEATER	1	CLOSED	**PWM ON**	4 kHz / 50 % duty
6	OVEN_HEATER	1	OPEN	OFF	**Door safety override**
7	Reserved	0	any	OFF	Currently unused
7	Reserved	1	any	ON	Drives GPIO13 (no consumer)

A.3 Important Observations

1. **Door safety is enforced only in the CLIENT**

- HOST mask may show bit=1
- Physical output may still be OFF

1. **STATUS reports the logical mask**

- Not the effective physical state
- This is intentional and by design

1. **TOG command caveat (T9)**

- Mask toggles correctly
- Physical output may not change due to missing callback
- See Chapter 6 (Known Issue)

A.4 Example Interpretation

Mask:

```
0x0015
```

Binary:

```
0000 0000 0001 0101
```

Active bits:

- Bit 0 → FAN12V = ON
- Bit 2 → LAMP = ON
- Bit 4 → FAN230V_SLOW = ON

All other outputs remain OFF.

End of Appendix A

Appendix B – Bitmask Matrix View (Logical / Human Readable)

This matrix provides a **single-glance overview** of how the 16-bit `outputsMask` maps to logical actuators.

- Columns represent **bit positions (15 → 0)**
- Rows represent **logical actuators**
- Cell value:
- `1` → actuator requested **ON**
- `0` → actuator requested **OFF**

> Note:
>
> - Bits **8–15** are currently unused by the Client
> - Physical safety overrides (Door gating, Heater PWM) are **not reflected** here

---

B.1 Bit Position Reference

```
Bit index:  15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
Hex weight: 8000 ....                  0080 0040 0020 0010 0008 0004 0002 0001
```

B.2 Logical Actuator Matrix

Actuator / Meaning (0=OFF, 1=ON)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAN12V (cooling fan)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
FAN230V FAST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
LAMP	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
SILICAGEL MOTOR	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
FAN230V SLOW	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
DOOR SENSOR (input only)	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
HEATER (PWM enable)	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
RESERVED	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

B.3 How to Read This Matrix

Example:

```
outputsMask = 0x0015
```

Binary:

```
0000 0000 0001 0101
```

Matrix interpretation:

- FAN12V = 1 → ON
- LAMP = 1 → ON
- FAN230V SLOW = 1 → ON
- All other actuators = OFF

B.4 Important Notes

- This matrix shows **logical intent**
- Real hardware state may differ due to:
- Door safety gating
- Heater PWM requirements
- Always cross-check with Appendix A for physical behavior

End of Appendix B