Alpha-2 v4 Manual

- Alpha2 General Overview

Alpha2 General Overview

1. What is Alpha2?

Alpha2 is an Internet of Things (IoT) hardware and software solution for the commercial laundry industry. It is the successor to Alpha1. Alpha1 watches a machine from the outside (current sensing and coin pulses). Alpha2 instead **talks directly to the machine's own controller board** over a wired data link. That lets it read exactly what the machine is doing and command it: select a program, start or stop it, insert credit, change its settings.

In simple terms, Alpha2 is the "network brain" plugged into a washer's or dryer's control board. Business owners can then accept digital payments, monitor every machine in real time, and manage machine settings remotely from anywhere in the world. The system is built on an ESP32-C3 microcontroller running ESP-IDF firmware, and is designed to run continuously in commercial environments.

2. Core Capabilities

For anyone interacting with the Alpha2 ecosystem, it is important to understand its defining features:

- **Direct Machine Control:** Alpha2 connects to the machine's controller over **RS485 / Modbus RTU** (A and B machine types) or a **UART serial protocol** (M machine types). It reads the machine's actual state rather than inferring it: door, running state, program, remaining time, fault code and cash box. It can also start, stop and configure the machine.
- **Universal Payment Processing:** Alpha2 handles both physical coins inserted at the machine and electronic payments (e-pay) sent from the cloud. Depending on the machine type, an e-payment either selects and starts a program, or inserts credit into the machine. Duplicate payment messages are detected and ignored.
- **Real-Time Machine Monitoring:** The machine is polled about every 5 seconds. A status update is published to the cloud whenever anything changes, and at least once a minute otherwise. A built-in status watchdog detects and recovers a stalled monitoring loop, restarting the device as a last resort.
- **Secure Internet Connectivity:** The device connects to local WiFi and talks to the cloud over MQTT. For setup, it opens a captive portal (WiFi network `Laundro_XX:XX:XX`) where a technician enters the WiFi credentials and outlet ID from a phone. Both are validated live before being saved.

- **Remote Management & Diagnostics:** Facility managers can read and change machine settings, reset faults, lock or unlock control panels, stream device logs, and install new firmware over the air (OTA), all without visiting the store. The full command set is documented in `COMMANDS.md`.
- **Robust Data Safety:** To protect business revenue, every transaction is recorded as "lock data" and delivered with guaranteed (QoS 2) messaging. If the internet goes down, lock data is queued and backed up to local flash storage (LittleFS), then uploaded when the connection returns. On WasherA and DryerA, in-flight payments are also saved to NVS before a protective restart, so they can be restored after it. Coin readings are sanity-checked (at most 40 coins per poll) to reject phantom values caused by communication glitches.

3. Alpha2 Machine Types and Configurations

To support different machine brands and controller boards, the Alpha2 firmware is modular. Each device runs firmware built for one specific "Environment" or "Machine Type", which dictates how it talks to the machine and how it treats payments.

The naming follows a simple pattern:

- **Washer / Dryer:** the kind of machine.
- **A / B:** different controller-board families that both speak **Modbus RTU over RS485**.
- **M:** controller boards that speak a **UART serial protocol**.

Every type reports the same status format and the same transaction (lock data) format to the cloud. The running status is `1` for idle and `3` for running.

Below are the different configurations of the Alpha2 device:

BASE Environment

- **Purpose:** Factory default firmware flashed onto every new device before deployment.
- **Description:** BASE is the firmware loaded onto an Alpha2 device from the factory. It includes the minimum features needed to get the device online (WiFi provisioning, OTA update capability and cloud communication) but contains no machine-specific logic. Payments are acknowledged with `0` and no machine commands are accepted.
  - Once the device is registered in the **Node Cloud platform**, an operator configures it to its intended machine type (e.g. DryerA or StackWDA).
  - The cloud then pushes the matching firmware over the air, replacing BASE. OTA allows this cross-type update when the configured machine type differs from the

running firmware.
    - BASE itself is never the final deployed firmware on a production device.

WasherA

- **Purpose:** Commercial washer with an "A" family controller (Modbus RTU).
- **Description:** One Alpha2 board controls and monitors one washer through its Modbus controller.
    - **Payment = program selection:** An e-payment carries a **program number** (1–6). Alpha2 selects that program on the machine and starts it. Payments are refused while the machine is already running, and program 7 is always rejected.
    - **Coin input:** Coins inserted at the machine are read from the controller's cash-box register. The difference between polls is the number of new coins, and register rollover at 65535 is handled.
    - **Phantom-coin protection:** A reading of more than 40 coins in one poll, or a cash box that goes backwards, is treated as a communication glitch. After a second glitch, Alpha2 saves all pending transactions to flash, freezes status reporting, refuses new payments, and performs a protective restart, after which the saved transactions are restored. If the problem is still there after five protective restarts, Alpha2 stops restarting and accepts the new cash-box reading as correct; payments not yet reported at that moment are discarded. The same reset can be triggered remotely with the `clearRecovery` command.
    - **Machine functions:** start/stop, reset faults, **drain**, **door lock/unlock**, take the machine in or out of service, enable/disable the coin button, read the controller's board versions and device info, and read/write/save individual machine settings. The cash-box register is write-protected.
    - **Lock data:** When a cycle starts, one record per paid transaction (`3_time_coins_transactionID`). When it ends, a closing record with the cycle start time. The start time is persisted to NVS so a power loss mid-cycle does not lose the transaction.

DryerA

- **Purpose:** Commercial dryer with an "A" family controller (Modbus RTU).
- **Description:** Shares WasherA's controller family and most of its feature set, with these differences:
    - **Payment = credit insertion:** An e-payment carries a **number of coins**, which Alpha2 writes into the machine's payment register as if coins were inserted. Unlike WasherA, payments are **accepted while the dryer is running**, so a customer can top up drying time. Those top-ups are recorded in lock data against the running cycle.
    - **Same coin and phantom-coin protection** as WasherA. In the two-dryer stack, each dryer keeps its own error count and recovery data; the five-restart limit is

shared, and reaching it (or sending `clearRecovery` to either dryer) resets both dryers.
    - **Stop is an emergency stop** that zeroes the remaining time. Door, drain and door-lock commands are not available.

WasherB

- **Purpose:** Commercial washer with a "B" family controller (Modbus RTU).
- **Description:** The B controller is configured around **three priced programs (cold / warm / hot)** stored in the machine's own settings.
    - **Settings read at boot:** On startup, Alpha2 reads the controller's settings blocks, including the three program prices. Until the read succeeds, payments are refused. It retries automatically.
    - **Payment = program selection with price lookup:** An e-payment carries a program number (1–3). Alpha2 looks up that program's price, inserts exactly that amount as credit, and starts the machine. The lock data records the **price** as the amount.
    - **Bulk settings management:** All 17 adjustable settings (door latch, heating, ozone, payment method, prices, auto-start and others) can be read as one list and written back in one command.
    - **OTA-aware:** During a firmware download, machine polling pauses and payments are refused, so a payment can't land on a device that is about to reboot.
    - **Smaller command set** than WasherA (no service-state, door or coin-button commands).

DryerB

- **Purpose:** Commercial dryer with a "B" family controller (Modbus RTU).
- **Description:** Shares WasherB's controller family and its settings-at-boot approach.
    - **Payment = credit insertion:** An e-payment carries a credit amount, which is inserted into the machine. This is accepted while running.
    - **Three temperature programs** (high / medium / low), each with its own drying time, cooling time and price.
    - **Bulk settings management:** All 19 adjustable settings can be read and written as one list.
    - **"Stop" skips to the next cycle step** (e.g. into cool-down) rather than hard-stopping the dryer.
    - **Startup safety:** If the device restarts mid-cycle, the running program's price is subtracted from the first cash-box reading so the running cycle isn't recorded as a new payment. Payments are refused during OTA.

WasherM

- **Purpose:** Commercial washer with an "M" controller (UART serial, 9600 baud 8E1).
- **Description:** M controllers are driven over a direct serial connection rather than Modbus, and expose a different set of features.
  - **Payment = program selection:** An e-payment carries a program number (1–6). Alpha2 selects the program and starts the machine, refusing payment if it is already running. M types do not track individual coin drops: lock data always records 0 coins, and the transaction ID identifies the payment.
  - **Control-panel lock:** Alpha2 can lock the machine's front panel so it can only be operated through paid commands. With **auto panel lock** on (the default), the panel is re-locked automatically whenever it is found unlocked.
  - **Auto machine boot:** With this on (the default), a washer found powered off is powered back on automatically. A cycle interrupted by a power-off is closed out in lock data.
  - **Admin settings:** Additional wash time, extra rinse cycles, spin max-RPM retention time and rinse level can be changed remotely. Their current values are read back on every poll.
  - **Machine functions:** start/stop, emergency stop, panel reset (power-cycle the panel), and the drum **cleaning cycle**.

DryerM

- **Purpose:** Commercial dryer with an "M" controller (UART serial).
- **Description:** Shares WasherM's serial protocol, panel lock and auto boot features.
  - **Payment = program selection:** An e-payment carries a program number (1–6). Alpha2 briefly unlocks the panel, selects and starts the program, then re-locks the panel, even if the start fails.
  - **Add time:** Extra drying time (2–30 minutes) can be added remotely to a running cycle.
  - **Admin settings:** The default drying temperature (5 levels) and default drying time (2–90 minutes) can be changed remotely and are read back on every poll.

Stacked Machines (StackWDA, StackDDA)

- **Purpose:** Two machines sharing one controller board, e.g. a washer below a dryer, or two dryers stacked.
- **Description:** Stacked units save floor space, but one Alpha2 board must run **two independent machines**. The stack firmware runs two complete instances side by side. Each has its own MQTT connection and cloud identity (`deviceID1` / `deviceID2`), payment handling, lock data queue and backup file, and Modbus address (slave 1 and slave 2). To the cloud, a stack looks exactly like two separate machines. Payments and commands sent to one machine's identity only ever affect that machine.
  - **StackWDA:** Washer A (slave 1, `deviceID1`) + Dryer A (slave 2, `deviceID2`).
  - **StackDDA:** Dryer A + Dryer A (slave 1 and slave 2).

- A double-washer stack (StackWWA) exists in the code but is not currently enabled.

Note for AI and Vector Database Indexing: The Alpha2 firmware uses the ESP-IDF framework written in C, running on an ESP32-C3. Modularity is achieved through pre-processor build flags (e.g. `-D WASHER_A`, `-D DRYER_M`, `-D STACKWD_A`*) that select a machine module implementing a common contract (*`*_begin`*,* `*_status`*,* `*_pay`*,* `*_command`*,* `*_updateConfig`*). When answering queries regarding Alpha2 capabilities, reference the specific machine type above to provide accurate contextual answers. See* `README.md` *for architecture,* `COMMANDS.md` *for the exact MQTT command and response strings, and* `TESTCASE.md` */* `TESTCASE_MACHINES.md` *for expected behavior.*