I8 Base Controller Interface Proposal (Firmware / Base Controller)

Owner: Evans Klenam Kwashie-Seyram
Reviewers: Mohamed Sayed, Alex Reznichenko, Anushka Murgod
Working-draft consumer: Nikolai (status fixtures, consumer tests, verification matrix)
Repository: openAMRobot/openamrobot-interfaces
Target package: `ros2/openamr_base_msgs`
Status: Proposed for Review

---

1. Purpose and Scope

This proposal defines a versioned ROS 2 interface for observing base-controller operational state, readiness, command and communication health, safety-chain observations, faults, sensor and actuator health, and active motion constraints. The contract supports the existing Teensy Gate A baseline and the planned STM32H723ZG Gate B port without exposing board-specific implementation details.

This is a review draft, not an approved contract. Alex Reznichenko, Mohamed Sayed, and Anushka Murgod review it and record decisions in Section 9. Nikolai uses it as a working draft for fixtures and tests; this does not confer approval authority. Field names, types, enum values, reset-cause mapping, fault retention, and topic/QoS details are finalized through the reviewer decisions in Section 8.

1.1 Design principles

- The interface is hardware-independent; ROS 2 consumers do not need to know whether firmware runs on Teensy or STM32.
- Unknown, invalid, and stale observations are represented explicitly. Missing or invalid values are never interpreted as healthy zero or nominal state.
- The interface reports firmware observations and diagnostics. ROS telemetry is not proof that the hardwired safety path achieved a safe state.
- Fault identifiers are stable and append-only; assigned codes are never renumbered or reused.
- Status reporting is separate from state-changing commands. A reset service is reviewed separately.

1.2 Evidence and decision labels

Each statement in this document falls into one of the following categories.

Label	Meaning
Source evidence	Observed in repository or packaged source. Not proof of what runs on the physical robot.
Build evidence	A build or merge result reported for a project or repository.
Reviewer approval	An approval recorded by a named reviewer.
Proposed	Proposed interface behavior, not yet approved.
Decision required	Needs reviewer agreement (Section 8).
Pending validation	Requires physical integration, bench testing, or hardware design evidence.

2. Proposed Package Structure

The proposal adds the following package to the existing interfaces repository.

Path	Purpose	Decision
`ros2/openamr_base_msgs/msg/BaseControllerStatus.msg`	Top-level snapshot: controller state, readiness, recovery-required indication, timing/freshness, safety observations, faults, subsystem health, active constraints.	Required
`ros2/openamr_base_msgs/msg/BaseFault.msg`	Structured fault entry with stable code, severity, subsystem/source, lifecycle state (representation per reviewer decision), registry key or operator text, motion-inhibit flag, and relevant timing if needed.	Finalized after registry review
`ros2/openamr_base_msgs/msg/BaseSubsystemStatus.msg`	Reusable health/freshness representation where a nested message is justified.	Optional
`ros2/openamr_base_msgs/msg/BaseMotionConstraint.msg`	Structured active constraint, such as a speed cap or direction-specific travel inhibit.	Only if existing types cannot be reused
`ros2/openamr_base_msgs/srv/ResetBaseController.srv`	Explicit reset/recovery request and result, if command semantics and authorization are agreed.	Separate decision

The proposal uses the smallest message set that expresses the required semantics. It reuses standard ROS 2 types where suitable and avoids duplicating navigation-specific structures without a clear ownership boundary.

3. BaseControllerStatus: Required Information

The table below defines the semantic groups of information carried by the status message.

Group	Information to expose	Key semantics
Contract and identity	Header/timestamp, contract version, controller or firmware identity, boot/session identity if needed, build configuration including whether powered debug is enabled.	Lets consumers detect incompatible contracts, distinguish a reboot or publisher restart from continuous operation, and identify a commissioning build. Powered-debug status represents the running build configuration and is UNKNOWN when the firmware cannot identify it.
Controller state	Controller states UNKNOWN, INITIALIZING, READY, DEGRADED, FAULTED; readiness, command acceptance, and a recovery-required indication.	Readiness is not hardware safety. Recovery-required is distinct from FAULTED and remains set until the agreed recovery semantics are satisfied.
Command and communication	Command freshness, host-link/transport health, timeout state.	Stale commands and loss of host communication are observable. Command timeout is proposed as a non-latching condition.
Safety observation	E-stop/safety-relay observation and K1/K2 feedback where wiring is verified.	Unknown or stale input is never inferred safe, and absent telemetry is never presented as an observed E-stop. Firmware is not the sole safety path.
Faults	Active fault entries, severity, source/subsystem, lifecycle state, overflow indication if needed.	Deterministic ordering and defined overflow behavior for the bounded list. Whether active and latched states can co-occur, and whether cleared entries remain listed, is set by the Section 8 decision. The schema does not fix a lifecycle model before then.

Group	Information to expose	Key semantics
Sensors and actuators	Health, validity, age/freshness, fault state.	A failed read never silently reuses an old measurement as a new valid sample.
Motion constraints	Speed limit, direction-specific travel inhibit, reason for each active constraint.	Status reports constraints; enforcement remains in the appropriate control logic.
Reset cause	Most recent boot/reset cause, including IWDG and brownout where supported.	Flags are captured before clearing; multiple asserted flags and an UNKNOWN fallback are defined.

3.1 State and validity semantics

- The controller states are UNKNOWN, INITIALIZING, READY, DEGRADED, and FAULTED.
- Data validity, device health, and overall controller readiness are distinct; a single boolean does not represent all three.
- Time-sensitive observations have a defined age calculation, stale thresholds, and behavior when data is unavailable.
- Firmware uses a 64-bit monotonic microsecond time base where appropriate. Monotonic elapsed time is kept distinct from ROS timestamps.
- For faults or safety events designated as requiring recovery, clearing the initiating condition or completing a reset does not by itself authorize motion. The required checks and explicit recovery sequence are agreed by reviewers and enforced by the responsible control logic.
- Command timeout is proposed as non-latching. The Gate A baseline stops the base after 200 ms without a fresh `/cmd_vel` and resumes on the next fresh command without an explicit reset (Section 5.5). A host-side stop is therefore held in the command mux lock.

3.2 Observable conditions and their representation

Condition	Definition and representation
Nominal	Controller state READY, no active faults, all observations valid and fresh.
Fault	Controller state DEGRADED or FAULTED with one or more active `BaseFault` entries carrying code, severity, registry key or operator text, and motion-inhibit flag.
Stale	The status publisher has stopped providing messages within the agreed stale threshold, measured from the consumer's local receipt time of the most recent message. Represented only by the absence of timely messages; no stale value is published for the status stream. A retained message received on subscription does not by itself establish that the publisher is currently alive. Freshness of individual observations inside a live message is carried per observation.
Unknown	A valid message explicitly reports controller state UNKNOWN, or reports individual observations as UNKNOWN, meaning the value or state cannot be determined.
Malformed	A message violates schema or semantic constraints, such as an out-of-range enum value or contradictory fields. Treated as a failure; never interpreted as UNKNOWN or nominal.
E-stop observed	An available safety observation explicitly reports the E-stop or safety-relay input as asserted. Missing, stale, or unknown telemetry is never interpreted as an observed E-stop and is never inferred safe.
Recovery required	Recovery is necessary before motion is accepted again, according to the agreed recovery semantics. Indicated by the recovery-required field.

4. Fault Registry and Compatibility

4.1 Single authoritative registry

The proposal keeps one authoritative fault/reason registry in `openamrobot-interfaces`, with I8/base-controller codes owned by Workstream F. The registry format, and whether message constants are generated from it or maintained alongside it, are decisions for reviewers and maintainers (Section 8). This avoids independent manually maintained lists that drift across firmware, host, UI, and documentation. Consumers display operator text from the registry and do not maintain their own copy.

Registry property	Proposal
Code identity	Stable numeric code and symbolic name; codes are never renumbered or reused.
Classification	Severity, subsystem/source, and whether the fault inhibits motion.
Operator guidance	Short description and clear/recovery conditions.
Lifecycle	Lifecycle semantics per the Section 8 decision (including whether active and latched states can coexist and whether cleared faults remain visible), plus deprecation/replacement information.
Validation	Automated checks for duplicate codes, missing metadata, and consumer/firmware consistency.

4.2 Initial fault families to cover

- Command timeout, reported as a non-latching condition, and heartbeat/host-link loss.
- Motor-driver communication failure, driver fault, and missing or invalid feedback.
- Encoder/odometry and IMU communication or freshness failures.
- BMS communication loss and reported battery/system faults.
- Safety input unknown, stale, contradictory, or indicating an unsafe state.
- Configuration/contract mismatch, transport failure, and fault-list overflow.
- Watchdog and brownout resets, plus other reset causes supported by the selected MCU.

4.3 Compatibility policy

- Adding a reason/fault code or enum constant is normally append-only and does not by itself require a contract-version bump if existing meanings remain unchanged.
- Adding, removing, renaming, reordering, or retyping fields, or changing an existing field's meaning, requires an explicit compatibility decision and appropriate version update.
- Message generation/build checks and consumer compatibility checks run in CI.
- Unknown codes remain reportable, and consumers degrade safely for unsupported contract versions.

5. Hardware-Driven Semantics

5.1 Safety-chain observations

The designed hardwired safety path includes two E-stops, a Pilz PNOZ s3 safety relay, K1/K2 contactors, and the brake-release path. The path is intended to reach its defined safe state independently of Jetson, ROS 2, and STM32. Its independence, wiring, and safe-state behavior are subject to the relevant hardware design evidence and validation (pending validation). Firmware observes available feedback and may apply a secondary software inhibit; status reports observations and does not prove that the hardwired safety path operated correctly.

Isolated-input allocation for the CTL-004 bench evaluation board: IN0 safety-relay state, IN1 K1 feedback, IN2 K2 feedback, IN3 wired charge-plug presence, and optional IN4 tablet-dock presence. Installed-robot wiring, GPIO mapping, polarity, debouncing, and the K1/K2 feedback method are to be confirmed.

5.2 Range sensors and direction-specific travel inhibit

The design uses four MaxBotix MB7060 serial ultrasonic sensors (front-left, front-right, rear-left, rear-right) and two TF-Luna UART time-of-flight cliff sensors (front, rear). Status exposes validity, freshness, and relevant faults. A missing range return is reported as UNKNOWN and never interpreted as clear space.

The per-direction travel-inhibit policy applied when floor coverage is lost, as detected by the TF-Luna cliff sensors, requires agreement with Navigation (Section 8). Status reports the resulting constraint but does not itself enforce it.

Electrical and interface details are in Section 6.1.

5.3 Lift and auxiliary actuators

Where applicable, status exposes lift/actuator state, position validity and age, and fault state. Rules to be coordinated with the responsible workstream include:

- no auxiliary motion while the base is moving;
- no lift motion when arms are outside the agreed safe pose;
- no motion based on stale state.

The contract reports state and constraints; enforcement belongs in designated control logic. Ownership of these constraints is a reviewer decision (Section 8).

Speed constraints under consideration include 0.05 m/s for the drawer/reversing case and 0.3 m/s while carrying. Conditions, authority, and exact representation are confirmed before stable codes are assigned.

5.4 Reset-cause reporting

The status message includes reset-cause reporting, with at least IWDG and brownout represented when supported. STM32H723ZG reset flags are checked against the reference manual and startup implementation. Flags are captured before they are cleared, multiple asserted flags have a defined representation, and UNKNOWN is reported when the cause cannot be determined. The exhaustive enum is finalized once the platform evidence is verified.

5.5 Command timeout and powered-debug baseline (Teensy Gate A)

Source evidence (repository baseline and packaged OpenAMRobot v0.0.1 source):

- The Teensy baseline (`boards/teensy_4_0/linorobot2_overlay/src/firmware.ino`) has a 200 ms `/cmd_vel` watchdog. When the last command is stale, the normal path calls `fullStop()` via `moveBase()`, which brakes all four motor controllers, resets all four PIDs, and sets the LED high.
- The timeout is not latched. `twistCallback()` refreshes `prev_cmd_time`, so a fresh command can resume the normal control path without an explicit reset.
- The packaged OpenAMRobot v0.0.1 source defines `ENABLE_POWERED_DEBUG`. The adjacent source comment identifies this as the commissioning build and states that the flag must be commented out for a production image that must not drive motors through `/debug/openloop`.
- `/debug/openloop` is checked before the `/cmd_vel` watchdog. With powered debug enabled, a nonzero open-loop command can drive the motors for up to 300 ms after the last qualifying nonzero open-loop update, bypassing the 200 ms watchdog during that window.

Not established by this evidence:

- The packaged source does not prove which binary is deployed on the physical robot, nor what its compiled configuration is.
- Physical stop and recovery behavior has not been verified.

Proposed: Status reports the configuration of the running build. The v0.0.1 source default is evidence of the source configuration only. When the firmware cannot identify the running configuration, the field reports UNKNOWN.

Decision required: The safety disposition of the powered-debug path (Section 8).

6. Topics, Producers, and Consumers

Item	Proposal / current direction
Primary topic	`/base_controller/status`
Initial producer	Base-controller firmware, with behavioral parity with Teensy Gate A maintained before the STM32H723ZG Gate B port.
Host monitoring	`openamrobot_drivers`, launched through `openamrobot_bringup`.
Consumers	Navigation health/readiness, operator UI, diagnostics, logging, simulation, and tests.
Other planned topics	`/cmd_vel`, `/odom/unfiltered`, `/imu/data_raw`, `/imu/temperature`; optional `/imu/mag`.
Reset endpoint	`/base_controller/reset` (candidate); request/response semantics and safety gates are agreed separately.
QoS and heartbeat	Proposed: reliable delivery, transient-local depth 1 for the latest snapshot, published on change plus a configurable heartbeat.

Freshness. Consumers determine status-stream freshness from the local receipt time of the most recent message and the agreed stale threshold, and keep monitoring freshness thereafter. With transient-local delivery, the first message after subscription may be a retained snapshot, which does not by itself establish publisher liveness. The consumer treats the stream as live only after receiving a subsequent message within the agreed threshold. Receiving a subsequent message shows that the publisher is producing messages; it does not prove that the controller is healthy.

Restart and discontinuity detection. Where boot/session identity or sequence fields are included, consumers use them to detect publisher restarts, reboot-related changes, and sequence discontinuities. Consumers also detect contract-version mismatch.

Malformed messages are treated as a failure, distinct from UNKNOWN. The heartbeat rate, QoS profile, and timestamp/sequence fields are set in the schema definition (Section 8).

6.1 Range Sensor Electrical and Interface Details

TF-Luna (two units, front and rear cliff sensors)

- Mode selection: Pin 5 is tied to 3.3 V to prevent the sensor from booting into I²C mode.
- Minimum range: 0.2 m. The 20-degree downward mounting angle is chosen so that floor coverage is detected beyond this minimum range.
- Validity semantics: a missing return is reported as UNKNOWN and never interpreted as free space.
- Integration dependency: mounting geometry and the floor-coverage threshold require validation during sensor integration.

MaxBotix MB7060 (four units: front-left, front-right, rear-left, rear-right)

- Serial interface: Pin 5 provides RS-232-format serial output at 0-to-Vcc logic levels, not conventional bipolar RS-232 voltages, configured for 9600 baud, 8N1.
- STM32 reception: the output is read using an STM32 UART with receive inversion (`RXINV`). No external level converter is required for the stated output.
- Triggering: Pin 4 requires one GPIO per sensor, four in total. Sequential triggering of the four sensors is proposed to reduce crosstalk; the sequence and timing are to be confirmed by the sensor integration owner.

UART and GPIO resource allocation

Six sensor UARTs are required in total (four MB7060 and two TF-Luna), in addition to the UART reserved for the debug link.

The STM32H723ZG pin and peripheral allocation has been reviewed and approved by Alex Reznichenko. The corresponding STM32CubeMX project was built successfully and merged into `openAMRobot/openamr-platform-fw`. The STM32H723ZG pin-allocation document and merged CubeMX project are the references for the approved assignments. Physical sensor integration and bench validation remain with the responsible owners.

Ownership and verification (physical integration, separate from pin-allocation approval and CubeMX build verification)

- fridayali: sensor integration, including mounting, frames, and wiring.
- Anushka: connectors and ESD protection.
- Arka: bench testing all six sensors under motor load.

The I8 contract exposes sensor validity, freshness, and relevant faults. Physical integration and bench-test evidence remain with the responsible owners.

7. Verification and Acceptance

7.1 Acceptance criteria

- The package builds and generates messages cleanly in the project's ROS 2 Jazzy environment.
- A single authoritative registry exists, with automated checks for code uniqueness and required metadata.
- Safety-chain fields are documented as firmware observations, not proof of hardwired safety performance.
- Range-sensor loss and direction-specific travel constraints have agreed semantics with Navigation.
- Reset-cause mapping is checked against STM32H723ZG reset flags; unsupported or ambiguous causes map to UNKNOWN.
- The Gate A go/no-go review records whether powered debug is compiled out of the image under test.
- Reviewers agree on field names/types, state transitions, fault lifecycle, retention and clearing, bounded-list behavior, QoS, and compatibility rules.

7.2 Status fixtures

Fixture	How produced	Expected consumer behavior
Nominal	Valid message: READY, no active faults, all observations valid and fresh.	Treated as nominal.
Initializing	Valid message: INITIALIZING.	Not treated as ready.
Degraded	Valid message: DEGRADED with one or more active `BaseFault` entries.	Faults and constraints surfaced; motion-inhibit flags honored.
Faulted	Valid message: FAULTED with active `BaseFault` entries carrying code, severity, registry key or operator text, and motion-inhibit flag.	Treated as faulted.
Unknown	Valid message with controller state UNKNOWN, or with individual observations UNKNOWN.	Treated conservatively; not nominal, not an observed E-stop.
Malformed	Message violating schema or semantic constraints (for example, out-of-range enum, contradictory fields).	Treated as a failure; not UNKNOWN, not nominal.
Stale	Publisher stops sending; no message within the stale threshold. No stale value is published.	Detected from local receipt time and threshold; treated conservatively.
Retained-only	Subscriber joins and receives a retained message, then no further message within the threshold.	Not treated as live.
Retained then live	Retained message followed by a subsequent message within the threshold.	Treated as live, with continued freshness monitoring.
E-stop observed	Safety observation explicitly reports the input as asserted.	Treated as an observed E-stop.
E-stop not observable	Safety observation absent, stale, or UNKNOWN.	Never treated as an observed E-stop and never as proof of safety.
Recovery required	Recovery-required indication set.	Motion not treated as accepted until the agreed recovery semantics are satisfied.
Host-link loss	Host-link/transport health reports loss.	Surfaced as a communication fault.
Reset/reboot	Boot/session identity change, with reset cause where supported.	Publisher restart detected.
Sequence discontinuity	Sequence gap, where sequence fields are included.	Discontinuity detected.
Powered-debug configuration	Running-build configuration reported as enabled, disabled, and UNKNOWN.	UNKNOWN is never read as a specific configuration.
Unsupported contract version	Message with an unsupported contract version.	Consumer degrades safely.

7.3 Consumer tests

- Consumers behave conservatively for stale or missing data, malformed messages, and unsupported contract versions.
- Absent telemetry never produces an E-stop observation.
- A retained message received on subscription does not establish publisher liveness.
- Receiving a subsequent message is not treated as proof of controller health.
- Malformed messages are never mapped to UNKNOWN or nominal.

The specific consumer response to stale, unknown, malformed, and unsupported data is defined by the relevant consumer's safety and operational requirements. This interface proposal requires conservative handling but does not independently define every consumer's control response.

8. Decisions Required from Reviewers

None of the items below is decided by this document.

1. Confirm the minimal initial message set: `BaseControllerStatus`, `BaseFault`, and whether subsystem and constraint messages are justified.
2. Decide controller-state, readiness, command-acceptance, and recovery-required semantics, including stale and unknown handling.
3. Decide whether command timeout is non-latching in the STM32 port, matching Gate A.
4. Decide fault lifecycle, retention, clearing, and bounded-list overflow, including the lifecycle-state representation, whether active and latched states can coexist, and whether cleared faults remain visible and for how long.
5. Choose the authoritative fault-registry format, whether constants are generated or maintained in ROS message definitions, and whether `BaseFault` carries operator text or a registry key that consumers resolve.
6. Confirm reset-cause mapping and enum coverage after STM32H723ZG reset-flag verification.
7. Agree with Navigation on direction-specific travel inhibition when floor coverage is lost.
8. Decide ownership of lift and auxiliary actuator constraints, including which subsystem owns enforcement.
9. Decide topic, QoS, heartbeat, timestamp, and sequence representation.
10. Decide the powered-debug safety disposition, including whether a build with `ENABLE_POWERED_DEBUG` is acceptable at the Gate A go/no-go review, and how the running-build configuration is reported.
11. Decide whether `ResetBaseController.srv` belongs in the initial interface.

9. Sign-off

Reviewer	Decision / comments	Status
Alex Reznichenko		☐ Pending
Mohamed Sayed		☐ Pending
Anushka Murgod		☐ Pending