Network Working Group
Request for Comments: XXXX
Category: Standards Track

A. Wietting
Smarter Clock Standard

April 2026

Low-Frequency Radio Time Synchronization Protocol for Distributed Embedded Clock Networks (LFRTS/1.0)

Status of This Memo

This document specifies a protocol for the Smarter Clock Standard project and requests discussion and suggestions for improvement. Distribution of this memo is unlimited.

Abstract

This document defines the Low-Frequency Radio Time Synchronization Protocol (LFRTS), a lightweight protocol for maintaining time synchronization to within 50 milliseconds across a network of embedded battery-backed real-time clock (RTC) devices in GPS-denied environments such as indoor conference spaces. The protocol defines device roles, wake-cycle state machines, drift tolerance rules, authenticated broadcast behavior, and the I2C interface exposed to host (parent) boards.

Table of Contents

1. Introduction
2. Terminology
3. Device Roles
4. Device State Machine
    4.1. NAClock WITH Parent Board
    4.2. NAClock WITHOUT Parent Board
    4.3. Step-Grandfather Clock
    4.4. Grandfather Clock
    4.5. Dead State
5. Drift Tolerance and Trust Rules
6. Radio Broadcast Authentication
7. I2C Interface
    7.1. TICK Pin Output
8. LED Synchronization Behavior
9. Design Constraints and Non-Goals
10. Security Considerations
11. IANA Considerations
12. Factory Provisioning and Birth Process
    12.1. RTC Birth Pads
    12.2. Birth Jig
    12.3. MCU Write Prohibition
    12.4. RTC Write Protection
    12.5. Debug Interface Protection

    12.6. Einstein Alarm
13. Bonus Round: Synchronized PWM LED Duty Cycle

1. Introduction

GPS-based time synchronization achieves millisecond accuracy across large outdoor deployments but is unreliable in dense indoor environments due to signal attenuation and multipath latency. This protocol addresses that gap by establishing a low-frequency radio beacon hierarchy in which one or more factory-authorized Grandfather Clocks broadcast authenticated time signals, and Non-Authoritative Clocks (NAClocks) discipline their RTCs against those signals.

The primary design goals are:

- Passive, zero-configuration time synchronization to within 50 ms (imperceptible to the human eye) across up to 500 simultaneous devices.
- Graceful degradation: when no Grandfather Clock is reachable, a connected NAClock SHOULD promote itself to relay time to peers.
- Trust-bounded re-synchronization: a device that has been offline for an extended period MUST validate its drift is within acceptable bounds before accepting or relaying a time signal.
- Minimal footprint: the device is 2 cm x 2 cm with a backup battery. It MUST NOT perform any function beyond timekeeping and exposing time data over I2C.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Grandfather Clock (GC)
: A factory-provisioned, authoritative time source. Broadcasts time signals signed with a private key known only to the manufacturer. Always connected to a parent board. Always considered to have a parent board for the purposes of this spec.

Non-Authoritative Clock (NAClock)
: Any device that is not a Grandfather Clock. Ships from the factory with a correctly set RTC and disciplines its time against received broadcasts. May be connected or unconnected to a parent board.

Step-Grandfather Clock

A NAClock that has temporarily promoted itself to broadcast time on behalf of absent Grandfather Clocks. This promotion is ONLY available to NAClocks connected to a parent board.

Parent Board
: A host microcontroller board that physically connects to the clock module. I2C MUST always be connected between the clock and the parent board. The parent board is responsible for reading the current light show program and BPM downbeat from the clock over I2C in order to drive its own output behavior. The clock detects parent board presence by the presence of 3.3 V on the parent board power pin.

TICK Pin
: A dedicated digital output pin on the clock module. The clock drives this pin HIGH for 1 ms at each RTC second boundary, producing a heartbeat pulse for the parent board to consume. The parent board connects this pin to a GPIO interrupt input. The TICK pin is OPTIONAL. When connected, it allows the parent board to trigger an I2C read at the exact second boundary rather than relying on periodic polling.

Drift Offset
: A signed correction value, in milliseconds, stored non-volatilely on the device and applied to RTC readings to compensate for accumulated crystal drift. The current Drift Offset value is reported to the parent board over I2C (Section 7, Field 7).

Beacon Pulse
: A single authenticated radio broadcast transmitted by a Grandfather Clock or Step-Grandfather Clock. A beacon pulse contains: a signed Unix timestamp, a light show slot (0-255), a BPM value (beats per minute, 8-bit unsigned integer), and a Beat Modulus (a prime number used to derive the beat phase).

Light Show Slot
: An 8-bit unsigned integer in the range [0, 255] identifying one of 256 pre-defined synchronized light patterns. Slot 0 is the default (no active show). Slots 1-255 are user-defined patterns distributed out-of-band by the operator.

BPM (Beats Per Minute)
: An 8-bit unsigned integer in the range [1, 255] specifying the tempo of the active light show. A value of 0 is reserved and MUST be treated as "no beat active" by the receiving device.

Dead State
: A permanent, non-recoverable operational state entered by any device whose observed drift delta exceeds its trust window (Section 5). A device in Dead State MUST cease all radio activity, MUST NOT update its Drift Offset, and MUST set the

dead flag in all subsequent I2C responses (Section 7, Field 7, bit 1). Dead State persists across power cycles and MUST be stored non-volatilely. Recovery requires factory re-provisioning via the Birth Jig (Section 12).

RTC Birth Pads
: A set of PCB solder pads that expose the RTC chip's serial interface directly, bypassing the MCU entirely. The Birth Jig connects to these pads to write the initial time to the RTC at the factory. These pads are used exclusively during factory provisioning and MUST NOT be routed through the MCU.

Birth Jig
: A factory-side computer, separate from the clock module's MCU, that connects to the RTC Birth Pads and writes the initial Unix timestamp directly to the RTC chip. The Birth Jig MUST have a traceable time source. The MCU is NOT in the signal path during this operation. See Section 12.

Beat Modulus
: A 16-bit unsigned integer that MUST be a prime number in the range [257, 65521]. The lower bound of 257 ensures the modulus is strictly greater than the maximum supported BPM value (256), preventing phase collisions between tempo values and modulus values. The receiving device computes its beat phase as: beat_phase = (unix_time_seconds mod Beat_Modulus). This allows operators to create polyrhythmic or offset patterns across groups of devices by assigning different prime moduli to different zones without requiring additional coordination.

3. Device Roles

A device operates in exactly one role at any given time:

Role	Broadcasts	Listens
Grandfather Clock	Yes (always)	Yes (peer GCs only)
Step-Grandfather Clock	Yes (while active	Yes (for GC)
NAClock w/ parent	No	Yes
NAClock w/o parent	No	Yes

Role transitions:

- NAClock w/ parent --> Step-Grandfather Clock: when no GC or Step-GC is heard after exhausting the retry window.

- Step-Grandfather Clock --> NAClock w/ parent: immediately upon receiving a valid beacon pulse from a Grandfather Clock.

- Grandfather Clock (peer sync): when a GC receives a valid beacon from a peer GC, it MUST defer its own broadcast for that cycle and SHOULD accept the peer's time if within drift bounds.

- A Grandfather Clock MUST NOT demote itself under any condition.

- A NAClock WITHOUT a parent board MUST NOT promote itself.

4. Device State Machine

4.1. NAClock WITH Parent Board

Because it serves a parent board, a NAClock with a parent SHOULD use a shorter wake interval than a standalone NAClock in order to provide timely I2C responses.

On each wake cycle:

Step 1. Listen for a beacon pulse from a Grandfather Clock.

Step 2a. If a GC beacon is received:
- i. Validate the beacon signature (Section 6).
- ii. Compute delta = |received_time - local_rtc_time|.
- iii. Apply drift trust rules (Section 5).
- iv. If trusted, update Drift Offset with the computed delta. The RTC register MUST NOT be modified.
- v. Return to sleep.

Step 2b. If no GC beacon is received:
- i. Sleep for a random back-off interval.
- ii. Listen again.
  - If a GC beacon is now received: go to Step 2a.
  - If still no beacon received: promote self to Step-Grandfather Clock (Section 4.3).

4.2. NAClock WITHOUT Parent Board

A NAClock without a parent board has no obligation to relay time to any consumer. It MUST NOT promote itself to Step-Grandfather Clock regardless of how long it has been without a sync signal.

On each wake cycle:

Step 1. Listen for a beacon pulse from any authoritative source (GC or Step-GC).

Step 2a. If a beacon is received:
- i. Verify the beacon HMAC signature (Section 6). If verification fails, discard beacon silently.
- ii. Compute delta = |received_time - local_rtc_time|.

iii. Apply drift trust rules (Section 5).
iv. If trusted, update Drift Offset with the computed delta. The RTC register MUST NOT be modified.
v. Return to sleep.

Step 2b. If no beacon is received:
i. Sleep for a random back-off interval.
ii. Listen again.
- If a beacon is received: go to Step 2a.
- If still no beacon: enter DEEP SLEEP. No broadcast. No I2C. Conserve power.

4.3. Step-Grandfather Clock

A Step-Grandfather Clock MUST check for a true Grandfather Clock BEFORE performing any broadcast on each wake cycle. Listening for a GC takes priority over all other actions.

On each wake cycle:

Step 1. Listen for a beacon pulse from a Grandfather Clock.

Step 2a. If a GC beacon is received:
i. Verify the beacon HMAC signature (Section 6). If verification fails, discard beacon silently and go to Step 2b.
ii. IMMEDIATELY cease all broadcasting.
iii. Apply drift trust rules (Section 5). If trusted, update Drift Offset. The RTC register MUST NOT be modified.
iv. Demote self to NAClock WITH parent board.
v. Return to sleep.

Step 2b. If no GC beacon is received:
i. Broadcast current RTC time as a Step-GC beacon.
ii. Return to sleep.
iii. Check again on the next wake cycle.

A Step-Grandfather Clock MUST NOT take any action that would prevent it from hearing a Grandfather Clock. It MUST NOT hold the radio channel in a way that blocks inbound GC signals.

4.4. Grandfather Clock

A Grandfather Clock is the sole, persistent source of authority. In deployments with multiple Grandfather Clocks, each GC MUST listen for peer GC beacons before broadcasting. This prevents simultaneous dual beaconing, which would cause destructive signal collisions, and allows GCs to fail forward from a peer's time if their own RTC has drifted.

On each wake cycle:

Step 1. Listen for a beacon pulse from another Grandfather Clock.

Step 2a. If a peer GC beacon is received:
- i. Validate the peer beacon signature (Section 6).
- ii. Compute delta = |received_time - local_rtc_time|.
- iii. Apply drift trust rules (Section 5).
- iv. If trusted, update Drift Offset with the computed delta. The RTC register MUST NOT be modified.
- v. DEFER broadcasting this cycle. The peer GC has already covered this interval. Two GCs MUST NOT broadcast simultaneously.
- vi. Return to sleep.

Step 2b. If no peer GC beacon is received:
- i. Broadcast current time, signed with the manufacturer private key, as a GC beacon pulse.
- ii. Return to sleep.

A Grandfather Clock MUST NOT accept time signals from Step-Grandfather Clocks or NAClocks. Peer GC beacons are the only external source a Grandfather Clock is permitted to trust for time correction. A Grandfather Clock MUST NOT demote itself.

4.5. Dead State

Dead State is a terminal condition. Any device -- regardless of role -- MUST enter Dead State immediately upon a trust evaluation failure (Section 5). There is no grace period and no operator override in the field.

A device in Dead State MUST:

- Cease all radio transmission immediately and permanently.
- Cease listening for beacon pulses.
- Write the dead flag to non-volatile storage so that Dead State survives power loss and power-on reset.
- Continue driving the TICK pin at each RTC second boundary so the parent board can detect that the module is still powered.
- Continue responding to I2C reads, reporting its last-known RTC time (plus Drift Offset), sync flag = 0, and dead flag = 1 in Field 7 (Section 7).

A device in Dead State MUST NOT:

- Accept or act on any beacon pulse.
- Update its Drift Offset.
- Promote itself to any role.
- Clear the dead flag without a factory re-provisioning event.

The parent board SHOULD monitor Field 7 bit 1 on every I2C read and treat a dead flag of 1 as a hardware fault requiring physical device replacement.

5. Drift Tolerance and Trust Rules

Crystal oscillators in the selected RTC component drift at a bounded rate. For this specification, the maximum drift rate is defined as:

```
MAX_DRIFT_RATE = 3 seconds per year (Read the DS for exact drift tolerances)
```

When a device receives a beacon pulse, it MUST evaluate the received time against this bound before accepting it.

Let:

```
delta          = abs(received_unix_time - local_rtc_unix_time)
years_offline  = estimated years since last confirmed sync
max_allowed    = years_offline * MAX_DRIFT_RATE
```

Trust evaluation:

- If delta <= max_allowed:
  The signal is within the expected drift envelope. The device MUST accept the correction and update its stored Drift Offset. The RTC register MUST NOT be modified at any point post-birth. All time correction is applied exclusively through Drift Offset.

- If delta > max_allowed:
  The signal exceeds the expected drift envelope. The device MUST immediately enter Dead State (Section 4.5). There is no retry, no waiting, no recovery path in the field. A device that cannot be trusted is a device that must not operate.

Note: drift is bidirectional. A clock may run fast (+delta) or slow (-delta). Both cases are evaluated against the same bound.

Note: the RTC register set at the factory birth process MUST NOT be modified at any time after birth -- not at power-on, not on sync, not ever. The birth-set RTC is the immutable baseline. All correction is accumulated in the Drift Offset field only. The mechanism by which the birth time is set is specified in Section 12; the MCU has no role in that process.

6. Radio Broadcast Authentication

Grandfather Clock beacon pulses MUST be authenticated using at minimum HMAC-SHA-256 with a shared factory key. Implementations MAY substitute a stronger scheme (e.g., HMAC-SHA-3, or ECDSA

with a manufacturer-held private key and a per-device burned public key) provided the chosen scheme meets or exceeds the security properties of HMAC-SHA-256. HMAC-SHA-256 is the floor, not the ceiling. The shared factory key MUST be burned into all devices at provisioning time and MUST NOT be extractable post-provisioning. Regardless of scheme, the following requirements apply:

- A receiving device MUST verify the beacon authentication tag before acting on any field in the beacon payload. A beacon that fails verification MUST be silently discarded.

- A receiving device MUST be able to verify the authentication tag using only data stored at factory provisioning time.

- A Step-Grandfather Clock beacon MUST be distinguishable from a GC beacon. Receivers MUST NOT treat a Step-GC beacon as authoritative for the purposes of Section 5 trust evaluation when a GC beacon is simultaneously available.

- Secure Boot MUST be enabled on all devices to prevent extraction or replacement of key material stored in firmware.

7. I2C Interface

I2C MUST always be connected between the clock module and the parent board. The parent board requires the light show slot and BPM downbeat from the clock in order to drive its output behavior; there is no substitute for this connection. The parent board is the I2C master. The clock is the I2C slave. The clock MUST NOT initiate I2C transactions.

On each I2C read request, the clock MUST respond with the following fields in order:

```
Field 1: Unix timestamp    (32-bit unsigned integer, seconds)
Field 2: Light show slot   (8-bit unsigned integer, 0-255)
Field 3: BPM               (8-bit unsigned integer, 0-255)
Field 4: Beat Modulus      (16-bit unsigned integer, prime, 257-65521)
Field 5: Beat phase        (8-bit unsigned integer, computed
                            as unix_time_seconds mod Beat_Modulus)
Field 6: Drift Offset      (16-bit signed integer, milliseconds)
Field 7: Status byte       (8-bit flags field)
                            bit 0: sync  (0 = unsynced, 1 = synced)
                            bit 1: dead  (0 = alive,   1 = dead)
                            bits 2-7: reserved, MUST be 0
```

Total response size: 13 bytes.

Light show slot rules:

- 255 slots are available for operator-defined light patterns (slots 1-255). Slot 0 is the default state (no active show) and MUST be set at factory provisioning.

- If a received slot value is outside the range [0, 255], the device MUST discard it and retain the last valid value.

BPM rules:

- A BPM value of 0 indicates no beat is active. The parent board SHOULD treat this as a cue to suspend beat-driven behavior.

- If a received BPM value is 0 and a prior non-zero value was active, the device MUST update to 0. Zero is a valid state, not an error.

Beat Modulus rules:

- The Beat Modulus MUST be a prime number in the range [257, 65521]. The lower bound of 257 is chosen because it is the smallest prime greater than 256, the maximum supported BPM value. This guarantees no collision between any BPM value and any valid modulus value.

- If a received Beat Modulus value is not prime, is less than 257, or exceeds 65521, the device MUST discard it and retain the last valid value.

- The device MUST recompute Beat phase on every I2C read using the current unix_time_seconds and the stored Beat Modulus. Beat phase MUST NOT be stored; it is always derived.

- At factory provisioning, Beat Modulus MUST default to 2.

Drift Offset rules:

- Drift Offset is a 16-bit signed integer representing the cumulative correction applied to the RTC, in milliseconds. Positive values indicate the RTC was running fast; negative values indicate it was running slow.

- At factory provisioning, Drift Offset MUST be initialized to 0.

- The parent board MAY use the reported Drift Offset to model clock behavior or detect anomalous drift rates over time.

General rules:

- If the device has never received a sync signal, it MUST still respond with its locally-ticking RTC time and MUST set the sync status flag to 0 (unsynced). All other fields report

their last-known or default values.

7.1. TICK Pin Output

I2C polling introduces latency proportional to the parent board's polling interval. For applications requiring tighter timestamp accuracy, the clock module drives a TICK pin that pulses HIGH for 1 ms at each RTC second boundary. The parent board consumes this pulse via a GPIO rising-edge interrupt and immediately reads the fresh timestamp over I2C.

TICK pin electrical specification:

- The TICK pin is a push-pull digital output driven by the clock module. It is normally LOW and pulses HIGH for exactly 1 ms at each second boundary.

- Logic HIGH level: 3.3 V. The parent board GPIO input MUST be compatible with 3.3 V logic.

- No pull-up or pull-down resistor is required on the parent board side.

Parent board behavior on TICK pulse:

Step 1. Detect rising edge on the TICK pin via GPIO interrupt.
Step 2. Immediately initiate an I2C read of all fields (Section 7). The timestamp captured at this moment reflects the start of the new second with minimal software-path latency.
Step 3. Use the fetched Unix timestamp as the authoritative second boundary for all downstream timing operations.

Clock module behavior:

- The clock MUST drive the TICK pin regardless of sync status. An unsynced clock MUST still pulse TICK at each RTC second boundary so the parent board can detect tick presence.

- The clock MUST NOT suppress TICK pulses during I2C transactions. If an I2C read is in progress when a second boundary occurs, the TICK pin MUST still pulse. The parent board is responsible for handling the race condition.

- On re-synchronization, the TICK pulse phase will shift immediately to reflect the corrected second boundary. The parent board SHOULD treat the first TICK pulse after a re-sync as authoritative and discard any buffered prior value.

Relationship to TICK pin:

- I2C is always present. The TICK pin is an OPTIONAL enhancement that allows the parent board to trigger an I2C read at the precise second boundary rather than polling on its own timer.

- A parent board that does not connect the TICK pin MUST poll I2C on its own interval to retrieve updated values. The light show slot and BPM do not change at second granularity, so lower polling rates are acceptable for those fields. The Unix timestamp and Beat phase SHOULD be polled at least once per second to maintain accuracy.

8. LED Synchronization Behavior

Each device has one onboard LED. The LED fade cycle is:

- One complete fade-up and fade-down per 60-second period.
- The cycle phase is derived from (unix_time mod 60). The cycle resets at the top of each minute.
- On re-synchronization, the LED phase MUST update immediately to reflect the corrected time. Gradual interpolation is explicitly NOT used. The snap is hard and instantaneous.

The LED is driven by the local RTC tick. The parent board's light show behavior (driven by the light show slot via I2C) is a separate concern and is outside the scope of this document.

9. Design Constraints and Non-Goals

This protocol is intentionally minimal. The following are explicitly OUT OF SCOPE for version 1.0 base compliance:

- PWM duty-cycle synchronization across devices. (An OPTIONAL extension is described in Section 13 for implementations that wish to go beyond base compliance.)
- Sub-50-millisecond accuracy (target is <= 50 ms, not below).
- Over-the-air firmware updates.
- Mesh routing or multi-hop time relay beyond one Step-GC hop.
- Any function other than timekeeping and I2C reporting.

10. Security Considerations

Secure Boot MUST be enabled (Section 6) to prevent code extraction and cloning of authentication key material.

Devices MUST NOT accept unauthenticated time signals. An attacker broadcasting an unsigned or incorrectly-signed beacon MUST be silently ignored.

The drift trust window (Section 5) provides an implicit bound on

time-injection attacks: an attacker cannot shift a device's time by more than its accumulated drift envelope without being rejected.

Hardware write protection on the RTC time registers (Section 12.4) ensures that physical possession of a device does not grant the ability to forge or overwrite a birth timestamp. Only the issuer, via the Birth Jig and manufacturer-held credential, can modify the immutable time baseline.

The MCU debug and programming interfaces (JTAG/SWD) MUST be credential-locked after factory provisioning (Section 12.5). An unauthenticated attacker with physical access MUST NOT be able to read flash contents, inject firmware, or extract key material via these interfaces.

11. IANA Considerations

This document has no IANA actions. The low-frequency radio spectrum usage is subject to applicable regional regulations (FCC Part 15 or equivalent) and is outside the scope of this specification.

12. Factory Provisioning and Birth Process

The RTC time registers are written once, at the factory, and MAY be reprovisioned by an external jig that connects directly to the RTC chip via dedicated PCB pads. The MCU MUST NOT contain any firmware routine that writes to the RTC time registers.

12.1. RTC Birth Pads

The clock PCB MUST expose a set of solder pads (the RTC Birth Pads) that provide direct access to the RTC chip's serial interface. These pads MUST be electrically connected to the RTC | MARKUP FOR FIXIN: We need to prevent turning on the MCU chip and MUST NOT be routed through or gated by the MCU.

The selected RTC chip communicates over I2C. The RTC Birth Pads MUST expose at minimum:

- The RTC I2C data line (SDA), connected directly to the RTC chip. This pad is on the same I2C bus as the MCU.
- The RTC I2C clock line (SCL), connected directly to the RTC chip. This pad is on the same I2C bus as the MCU.
- A ground reference pad.
- An MCU reset pad (active-low NRST or equivalent), so the Birth Jig can hold the MCU in reset during provisioning.

The MCU reset pad is REQUIRED. Because the Birth Jig and the MCU share the same I2C bus to the RTC, the MCU MUST be held in reset

for the entire duration of the Birth Jig's I2C transaction to prevent bus contention. The Birth Jig MUST assert MCU reset before initiating any I2C transaction and MUST NOT release reset until all I2C transactions are complete and the bus is idle.

The pads MUST be clearly labeled on the PCB silkscreen. They MUST NOT be used for any purpose other than factory provisioning.

12.2. Birth Jig

The Birth Jig is a standalone computer, physically separate from the clock module, that connects to the RTC Birth Pads and writes the initial Unix timestamp directly to the RTC chip. The MCU is NOT in the signal path during this operation.

The Birth Jig MUST:

- Have a traceable time source: a GPS-disciplined oscillator, an NTP-synchronized host with known offset, or a calibrated reference clock of equivalent or better accuracy.

- Write the current Unix timestamp directly to the RTC time registers via the Birth Pads, using the RTC chip's native serial protocol.

- Read back the written value immediately after writing and verify it matches the intended time within an acceptable tolerance (RECOMMENDED: <= 50 ms).

- Write the HMAC factory key and any other provisioning data to the MCU flash via a separate programming interface (e.g., JTAG or SWD). This is a distinct operation from RTC time setting and MUST be performed by the Birth Jig or a co-located programming fixture.

- Mark the device as birthed in a provisioning record that associates the device serial number with the birth timestamp and jig calibration certificate.

- Lock the MCU debug and programming interfaces (Section 12.5) after all firmware, key material, and provisioning data have been written and verified. This step MUST precede the RTC write-protection step below.

- As the final provisioning step, after all time and key data have been written and verified, engage the RTC write-protection mechanism (Section 12.4). The Birth Jig MUST NOT release MCU reset until both debug interface lock and RTC write protection are confirmed active.

12.3. MCU Write Prohibition

The MCU firmware MUST NOT contain any code path that writes to the RTC time registers (seconds, minutes, hours, day, date, month, year, or any equivalent time-of-day register set in the selected RTC component). This prohibition applies unconditionally: at boot, during normal operation, on sync reception, and on Dead State entry.

The MCU MAY write to non-time RTC registers (e.g., alarm registers or control registers) if required for wake-cycle management, provided those writes cannot alter the stored time-of-day value. Where the RTC supports alarms, the Birth Jig SHOULD set the Einstein Alarm at provisioning time (Section 12.6).

A firmware audit MUST confirm the absence of any RTC time-register write routine before a device is cleared for production. Presence of such a routine is a production stop condition.

12.4. RTC Write Protection

The RTC chip selected for this specification MUST support a hardware write-protection mechanism that locks the time-of-day registers against modification after factory provisioning. Write protection MUST be engaged by the Birth Jig as its final provisioning act, before MCU reset is released.

- The write-protection mechanism MUST require a manufacturer-held credential (e.g., a protection password or chip-specific key sequence defined by the RTC chip vendor) to disengage. This credential MUST be stored exclusively on authorized Birth Jig systems and MUST NOT be stored in any form on the clock module, including in MCU flash, firmware, or any non-volatile storage accessible to the MCU.

- The MCU firmware MUST NOT store, reference, derive, or transmit the write-protection credential under any circumstances.

- If the selected RTC chip does not natively support credential-gated write protection, the PCB design MUST implement an equivalent hardware mechanism (e.g., a dedicated write-protect pin permanently asserted after birth via a one-time-programmable fuse or equivalent, with no MCU-accessible signal path to deassert it).

- Re-provisioning a previously birthed device MUST require physical access to the RTC Birth Pads and possession of the manufacturer write-protection credential. Re-provisioning MUST be logged in the provisioning record with the reason, operator identity, new birth timestamp, and jig calibration certificate. Each re-provisioning event supersedes the prior birth record but MUST NOT erase it.

12.5. Debug Interface Protection

The MCU's hardware debug and programming interfaces (e.g., JTAG, SWD, or equivalent) expose flash read/write access and register inspection that could allow extraction of the HMAC factory key, the RTC write-protection credential, or any other key material. These interfaces MUST be locked after factory provisioning.

- The Birth Jig MUST engage the MCU's debug access protection (e.g., RDP Level 2 on STM32, or the equivalent read-out protection on the selected MCU family) as part of the provisioning sequence, before write protection is engaged and before MCU reset is released.

- The protection level selected MUST prevent flash read-back, MUST prevent debugger attachment, and MUST prevent code execution from RAM or external sources. The highest available irreversible protection level MUST be used.

- Where the MCU family supports a credential (password or key) to re-enable debug access, that credential MUST be treated identically to the RTC write-protection credential: stored exclusively on authorized Birth Jig systems, never on the device, and never derivable from any data stored on the device.

- A debug interface lock that is reversible (e.g., a protection level that can be cleared by a mass erase without a credential) is NOT sufficient. If the selected MCU does not support a credential-gated or fully irreversible debug lock, an alternative MCU MUST be selected.

- A firmware audit MUST verify that the debug protection level is set correctly on every production device before shipment. A device found to have debug interfaces unlocked or set to a reversible protection level is a production stop condition.

12.6. Einstein Alarm

If the selected RTC chip supports a recurring calendar alarm, the Birth Jig SHOULD configure it to fire annually on March 14 (the birthday of Albert Einstein, and International Pi Day) at 01:59:26 UTC -- chosen because, to nine significant figures, pi = 3.141592653, making 3/14 at 1:59:26 the most precise calendar representation of pi achievable at one-second granularity.

- This alarm has no defined protocol behavior. It is purely celebratory. The MCU MAY handle the alarm interrupt in any way it sees fit, or ignore it entirely.

- The alarm register is a non-time register for the purposes of Section 12.3. Setting or clearing it does not violate the RTC write prohibition.

- If the RTC does not support a calendar alarm, this section does not apply. The absence of this alarm is not a production stop condition.

Authors' Address

Smarter Clock Standard
https://github.com/your-org/smarter-clock-standard
```

13. Bonus Round: Synchronized PWM LED Duty Cycle

Non-normative. Devices that implement this section are not required to do so for base protocol compliance. Nothing in this section modifies any MUST or MUST NOT requirement elsewhere in this document.

Because every device in the network converges on the same 50-millisecond-accurate effective time (unix timestamp plus Drift Offset), they share a common sub-second clock reference without any additional communication. This makes network-wide PWM duty cycle synchronization a purely local computation: each device independently derives the same duty cycle value from its own corrected time, and the network appears visually coherent.

Duty cycle derivation:

Let t_eff_ms be the effective millisecond time:

```
t_eff_ms = (unix_time_seconds * 1000) + drift_offset_ms
```

Let phase_ms be the position within the 60-second LED cycle:

```
phase_ms = t_eff_ms mod 60000
```

The normalized phase is:

```
phi = phase_ms / 60000        (range: [0.0, 1.0))
```

The recommended duty cycle waveform is a raised sine, which produces a smooth, perceptually even fade-up and fade-down:

```
duty = 0.5 * (1 - cos(2 * pi * phi))
```

This yields duty = 0.0 at the start of the minute, rising to duty = 1.0 at the 30-second midpoint, and returning to 0.0 at
```

the 60-second boundary. All devices on the network execute identical arithmetic on the same t_eff_ms and therefore produce identical duty values at each moment.

Implementation notes:

- Implementations SHOULD NOT use integer-only approximations that accumulate per-cycle error. Recompute duty from t_eff_ms on each PWM update rather than incrementing a counter.

- On re-synchronization, t_eff_ms shifts immediately (consistent with Section 8). The duty cycle will snap to the corrected value on the next PWM update without interpolation.

- The PWM carrier frequency and resolution are hardware-dependent and are intentionally unspecified here. Any carrier frequency and bit depth suitable for visible LED dimming are acceptable.

- The raised-sine waveform is a recommendation, not a requirement. Implementations MAY substitute any waveform derived exclusively from phi, provided all devices in the deployment use the same waveform formula.