TIMER / generation 1 — User manual

Library: `src/lib2/timer/1/`
Edition: 2026-11-18, source/document review (documentation only)
Qualification: Versioned provider used in donor composition; independent timing and physical target qualification not established.

What this member does

Offer stable TM sampling and a count-based wait for timer transitions.

Public interfaces

Entry point or group	Inputs, outputs and behavior
`TMR_SAMPLE`	Sample TM until two consecutive values agree; returns `$8`, clobbers `$9`.
`TMR_WAIT`	Wait `$10` low-byte TM transitions; zero returns immediately; clobbers `$8/$9/$22/$23`, calls OFFCHK.

How to integrate

Include `timer.s` once after defining OFFCHK. Set `$10` to desired transition count and `CAL TMR_WAIT`, or `CAL TMR_SAMPLE` to read a stable sample. Keep wait short enough for the consumer’s power/BRK policy.

Composition sketch (not an independently assembled client):

```
        LD      $10,4
        CAL     TMR_WAIT
```

Environment, dependencies and memory

Requires OFFCHK and `$SY=1`, valid context and TM processor register. No private RAM. It is not wall-clock calibrated or reentrant/interrupt-proof.

Limits and evidence

TM transitions are not milliseconds; do not present this as a real-time scheduler.

This documentation does not introduce a new build or runtime PASS. Source equivalence, native assembler parity, emulator execution and physical-machine evidence remain distinct.

Actual source and further reference

- This exact member folder in Google Drive
- Source modules: `timer.s`.
- Further contracts/source notes: timer.s. These may include dated donor paths or older acceptance statements.

- For a new consumer, pin exact source files and hashes, verify full include closure, inspect register/bank/resource conflicts and test the resulting application on each claimed target.

Praeteritum ad futurum servamus.