

Software timer i timer daemon task

one-shot, periodic, command queue i typed callback context

Karta	K14 / 16	Czas	30 minut
Platforma	Hazard3 / RV32I	Kernel	FreeRTOS V11.3.0
Timery	static one-shot + periodic	Heap delta	0 B
Wersja	v00.01	UUID	78db37df-...
		karty	

Polityka jest częścią typu

```
OneShotTimer<Probe> one(storage1, "one", 5, probe1);
PeriodicTimer<Probe> cycle(storage2, "cycle", 2, probe2);
auto accepted = cycle.start(); // command queue
// callback jeszcze nie musi byc wykonany
```

Kontrakt K14. Wrapper nie jest copy/move. **start/reset/change/stop** zwracają przyjęcie polecenia przez kolejkę daemona. Callback działa później na współdzielonym timer service tasku, a context i storage muszą nadal istnieć.

Plan 30 minut

Czas	Tryb	Dowód
0–5	contract	non-copy, typed policy, explicit start
5–10	queue	2 start accepted; callback snapshot 0
10–15	periodic	callback w tickach 2 i 4
15–20	commands	reset@1, change 2–3@4
20–25	daemon	handle/name, osobny stos
25–30	stop/lifetime	tick 7, inactive, bez 4. callbacku

Predykcja

Czy pdPASS ze start oznacza wykonany callback? _____. Gdzie callback się wykona: ISR, caller czy daemon?
_____.

1 Kolejka poleceń a wykonanie

```
main: start(one), start(periodic) - accepted, callbacks=0
      ↓ scheduler + Tmr Svc processes commands
      ↓ expiry - daemon invokes typed trampoline
```

Pomiar	Predykcja	Odczyt
start commands accepted	2	_____
callbacks przy return ze start	0	_____
verifier entry tick	0	_____
periodic first expiry	2	_____
periodic second expiry	4	_____

2 Reset one-shot i zmiana okresu

```
// tick 1
one.reset();           // new expiry: 1 + 5 = 6
// tick 4, after periodic callbacks 2 and 4
cycle.change_period(3); // next expiry: 4 + 3 = 7
```

Pomiar	Predykcja	Odczyt
reset command tick / accepted	1 / true	_____
one-shot callback tick	6	_____
one-shot active po expiry	false	_____
change command tick / accepted	4 / true	_____
periodic next callback	7	_____

Wyjaśnij różnicę między reset i change-period:

3 Dowód kontekstu timer daemon

Callback nie działa w przerwaniu ani na stosie taska, który wywołał **start**. Wszystkie software timers współdzielą jeden daemon.

Pomiar	Predykcja	Odczyt
current == timer daemon handle	true	_____
task name zaczyna się od Tmr S	true	_____
callback SP != verifier SP	true	_____
callback może blokować	nie	_____

Dlaczego callback ma być krótki?

Jeden długi callback opóźnia obsługę poleceń i expiries wszystkich timerów. Ciężką pracę przekaz taskowi przez notification, semaphore albo queue.

Zaproponuj granicę callback/task dla sterowania LED:

4 Storage i lifetime

```
StaticTimer_t storage;          // control block, caller-owned
Probe context;                 // timer ID points here
PeriodicTimer<Probe> timer(storage, "led", 2, context);
```

- oba control blocks i verifier storage są w .bss;
- heap przed/po xTimerCreateStatic: bez zmiany;
- context, storage i wrapper żyją dłużej niż pending command/callback;
- wrapper nie udaje, że destruktor synchronicznie opróżnił daemon queue.

Stop

stop() accepted: _____. Active po przetworzeniu: _____. Callback count po dalszych 4 tickach: _____. Dlaczego samo accepted nie wystarcza jako dowód? _____.

5 Hazard3/GDB i zaliczenie

```
make check
riscv64-unknown-elf-gdb build/task01_sw_timer/prog.elf
b software_timer_debug_checkpoint
```

```
p g_start_commands_accepted
p g_callbacks_when_start_returned
p g_verifier_entry_tick
p g_reset_tick
p g_change_tick
p g_one_shot_tick
p g_periodic_ticks
p g_one_shot_active_after_expiry
p g_periodic_active_after_stop
p g_daemon_handle_match
p g_daemon_name_match
p g_heap_before_timer_create
p g_heap_after_timer_create
p g_timer_pass
```

Zaliczenie

- ☐ rozróżniam accepted command od wykonanego callbacku;
- ☐ pokazuję periodic ticks 2, 4, potem po zmianie 7;
- ☐ reset@1 przesuwa one-shot na tick 6 i timer staje inactive;
- ☐ callback działa na Tmr Svc i osobnym stosie;
- ☐ stop pozostawia inactive i brak czwartego callbacku;
- ☐ static storage daje heap delta 0 i wrapper bez runtime C++.

Wyjście

Dlaczego nie wolno przechwycić wskaźnika do krótkotrwałego obiektu jako timer ID?

Co długi callback robi z innymi timerami?

Następna karta K15: jawne API FromISR, kontrakt higherPriorityTaskWoken, UART RX i GPIO edge wrappers.