



TITLEtimer: od mtime do okresowego deadline'u

VERv00.01

DATE/TIME2026-07-20T00:00:00+02:00

PROJECTFreestanding C na RV32I

SERIESFreestanding RV32I and K&R

CARD11/13

SHEET1/8

OBJ.

Inf.

PRG.

CORE

SCOPE

LEVEL

POS.

GITEA URLID

CARD UUID

f37170be-e988-5f5d-b3cf-6ea179a9cd54

45f574de-1c7b-5f41-a3af-f1632024ff0f

AUTHOR W. Pab12233E

GITEAhttps://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-machine-timer

HTMLhttps://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/f37170be-e988-5f5d-b3cf-6ea179a9cd54



Cel karty

Uczeń rozdziela źródło czasu, warunek pending, dwa poziomy enable, wybór handlera i jego efekt. Potrafi spólnie odczytać 64-bitowy licznik na RV32, uruchomić jednorazowe przerwanie oraz utrzymać fazę okresowego zegara przez $\text{deadline} += \text{period}$.

Zakres i zachowane przykłady

Trzy przykłady wykonują się na rzeczywistym modelu RTL Hazard3. Task01 wymusza rollover dolnego słowa mtime. Task02 dowodzi wejścia przez slot 7 wektora, skasowania poziomowego źródła przez przesunięcie mtimecmp i wznowienia po mret. Task03 zapisuje cztery planowane i obserwowane czasy oraz spóźnienia. Host nie jest używany jako substytut urządzenia MMIO i przerwań.

Układ każdego przykładu: pełny profil A1–A8; widok N/D ma jawny powód, a diagram nie jest wymagany, gdy tekst daje lepszy dowód.

Typ	Task	Idea	Waga
MMIO	Task01	RV32 split-register MMIO i high–low–high	kluczowe
IRQ	Task02	mtimecmp, pending, MTIE/MIE, mtvec, mret i rearm	kluczowe
TICK	Task03	deadline += period, tick sequence i lateness	kluczowe

TITLE timer: od mtime do okresowego deadline'u		VER v00.01	DATE/TIME 2026-07-20T00:00:00+02:00	
PROJECT Freestanding C na RV32I		SERIES Freestanding RV32I and K&R	CARD 11/13	SHEET 2/8
REV. Inf.	PRG.	CORE	SCOPE	LEVEL
GITEA https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-machine-timer		GITEA UUID f37170be-e988-5f5d-b3cf-6ea179a9cd54		
HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/f37170be-e988-5f5d-b3cf-6ea179a9cd54		AUTHOR W. Pab122333		

1 1 — Jeden timer, pięć różnych ról

00 WE 01

`mtime` jest źródłem rosnącego czasu, a relacja `mtime >= mtimecmp` wytwarza poziomy stan pending. `mie.MTIE` dopuszcza tę klasę przerw lokalnie, `mstatus.MIE` dopuszcza przerwy globalnie, a `mtvec` wybiera kod według przyczyny. Dopiero handler zmienia stan programu i przesuwa komparator. Żaden z bitów enable nie kasuje źródła.

W testbenchu Hazard3 `mtime` rośnie raz na krok, a linia timera jest aktywna tak długo, jak licznik nie jest mniejszy od komparatora. Dlatego powrót bez zapisu przyszłego `mtimecmp` wywołałby następną pułapkę niemal natychmiast.

2 2 — 64 bity urządzenia na 32-bitowym rdzeniu

00 WE 01

Dwa odczyty 32-bitowe nie tworzą automatycznie spójnego odczytu 64-bitowego MMIO. Sekwencja high–low–high akceptuje parę dopiero wtedy, gdy oba odczyty high są równe; zmiana high oznacza rollover między transakcjami i wymusza powtórzenie. Przy zapisie komparatora kolejność `high=UINT32_MAX`, `low`, docelowe high usuwa niebezpieczne okno z przejściowo zbyt małym `mtimecmp`.

Task01 zapisuje `mtime` tylko po to, aby fixture testbenchu deterministycznie doprowadził do rolloveru. Nie jest to obietnica, że licznik czasu będzie zapisywalny w innym SoC.

3 3 — Okres to oś czasu, nie odstęp od spóźnionego now

00 WE 01

Dla periodycznego zegara handler wykonuje `deadline += period`. Wtedy planowana sekwencja zachowuje fazę nawet wtedy, gdy wejście do ISR następuje później niż `deadline`. Reguła `now + period` użyta jako jedyny model przenosi aktualne spóźnienie do następnego terminu i tworzy dryf.

Karta mierzy `lateness = observed - scheduled`; nie obiecuje stałej wartości opóźnienia. Biezący RTL i build dają 40 taktów, lecz wynik zależy od magistrali, prologu ISR i punktu próbkowania. Przykład nie definiuje jeszcze polityki nadrabiania wielu całkowicie pominiętych okresów.

	TITLE timer: od mtime do okresowego deadline'u		VER v00.01	DATE/TIME 2026-07-20T00:00:00+02:00		
	PROJECT Freestanding C na RV32I		SERIES Freestanding RV32I and K&R		CARD 11/13	
	SHEET 3/8					
	GITEA https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-machine-timer		GITEA UUID f37170be-e988-5f5d-b3cf-6ea179a9cd54		45f574de-1c7b-5f41-a3af-f1532024ff0f	
HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/f37170be-e988-5f5d-b3cf-6ea179a9cd54						

4 Zadania — zachowane przykłady w profilu A1–A8

TASK TASK01 · Spójny odczyt mtime przez rollover	
c33548e4-07c8-599c-9457-8ba457701659	
BLOCK A1 CONTEXT · AKTYWNE - atomowość C nie jest atomowością MMIO	
Zachowany przykład: src/tasks/task01_mtime_consistent_read.c.	
Urządzenie ma 64-bitowy licznik, lecz RV32 wykonuje trzy jawne odczyty 32-bitowych rejestrów. Typ uint64_t służy do złożenia zaakceptowanych połówek; nie zamienia dwóch transakcji magistrali w jeden atomowy odczyt.	
BLOCK A2 STRUCTURE · AKTYWNE - dwa słowa jednego licznika	
timer_hw_t mapuje kolejno mtime, mtimeh, mtimecmp i mtimecmph od adresu 0xc0000100. volatile wymusza obserwowalne dostępy C, ale regułę spójności dostarcza dopiero algorytm.	
BLOCK A3 DISPATCH · N/D	
N/D. Task01 ma wyłączone przerwania i wykonuje bezpośredni odczyt MMIO; nie ma ISR, callbacku, tablicy funkcji ani wyboru odbiorcy.	
BLOCK A4 APPLICATION · AKTYWNE - kontrolowane przejście przez 0xffffffff	
Fixture ustawia high=0 i low blisko UINT32_MAX, a następnie wyszukuje małe okno, w którym high zmieni się między pierwszym i drugim odczytem. Zaakceptowany wynik musi pochodzić już z epoki high=1; kolejny odczyt nie może być mniejszy.	
BLOCK A5 FLOW · AKTYWNE - read, validate, retry	
Jedna próba czyta high_before, low, high_after. Równość high kończy pętlę i pozwala złożyć wynik. Nierówność odrzuca low należące do niejednoznacznej granicy epok i powtarza wszystkie trzy transakcje.	
BLOCK A6 STATE · AKTYWNE - epoka 0, rollover, epoka 1	
Stan urządzenia przechodzi z high=0 i low blisko maksimum przez rollover do high=1 i małego low. Pierwsza para high opisuje zmianę stanu, więc nie wolno do niej przypisać jednego snapshotu; druga próba stabilizuje się w epoce 1.	
BLOCK A7 RUNTIME · AKTYWNE - dwa dostępy high są widocznym dowodem	
Oracle raportuje offset, liczbę prób, retry, high wyniku i monotoniczność. Bieżący RTL daje offset 0x13 i attempts=2. Listing ma pokazać dwa osobne lw z rejestru high, rozdzielone odczytem low, oraz gałąź powrotną przy nierówności.	
BLOCK A8 PATTERNS · AKTYWNE - stabilny snapshot z licznika	
split-register	
Wzorzec high-low-high stosuj, gdy dokumentacja urządzenia gwarantuje monotoniczny licznik i zgodne zachowanie połówek. Fixture zapisujący czas pozostaje poza produkcyjnym API odczytu.	

	TITLE timer: od mtime do okresowego deadline'u				VER v00.01	DATE/TIME 2026-07-20T00:00:00+02:00			
	PROJECT Freestanding C na RV32I				SERIES Freestanding RV32I and K&R	CARD 11/13	SHEET 4/8		
	DIR.	PRG.	CORE	SCOPE	LEVEL	POS.	GITEA UUID f37170be-e988-5f5d-b3cf-6ea179a9cd54 45f574de-1c7b-5f41-a3af-f1632024ff0e		
	AUTHOR W. PAB12233E								
GITEA https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-machine-timer									
HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/f37170be-e988-5f5d-b3cf-6ea179a9cd54									

BLOCK Ćwiczenie · Przewidywanie → wykonanie → wniosek

Wyjaśnij, jaki błędny 64-bitowy wynik może utworzyć pojedyncze low-high podczas rolloveru. Następnie wskaż w raporcie próbę odrzuconą przez high-low-high i rozdziel kod produkcyjnego odczytu od zapisywalnego fixture'u testbenchu.

Evidence: Przewidywany błąd na granicy rolloveru, log z RTL Hazard3 z kodem wyjścia i limitem cykli, fragment listingu z dwoma odczytami high oraz wniosek o spójności zaakceptowanego snapshotu.

Acceptance: Oracle Hazard3: retry=1, attempts > 1, value_high=1, monotonic=1, pass=1 i kod wyjścia 0. W bieżącym buildzie okno znaleziono dla offsetu 0x13 i po dwóch próbach; sam offset nie jest kontraktem architektury.

BLOCK TASK01 · Zapis dowodu ucznia

1. Przewidywanie przed uruchomieniem

.....

2. Log wykonania RTL Hazard3 i kod wyjścia

.....

3. Dowód w listingu: high-low-high i retry

.....

4. Wniosek: reguła języka lub kontrakt targetu potwierdzony przez pomiar

.....

	TITLE timer: od mtime do okresowego deadline'u		VER v00.01	DATE/TIME 2026-07-20T00:00:00+02:00			
	PROJECT Freestanding C na RV32I		SERIES Freestanding RV32I and K&R	CARD 11/13		SHEET 5/8	
	URL: Inf.	PROG.	CORE	SCOPE		LEVEL	POS.
	GITEA https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-machine-timer		GITEA UUID 137170be-e988-5f5d-b3cf-6ea179a9cd54			45f574de-1c7b-5f41-a3af-f1632024f02e	
HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/f37170be-e988-5f5d-b3cf-6ea179a9cd54							

TASK TASK02 · Jednorazowe przerwanie machine timer

dc7f8696-19db-55b6-8fd9-cb676b399ec7

BLOCK A1 CONTEXT · AKTYWNE - enable nie jest źródłem

Zachowany przykład: `src/tasks/task02_oneshot_timer_irq.c`.

Źródłem jest relacja `mtime >= mtimecmp`; `mip.MTIP` reprezentuje pending. `mie.MTIE` i `mstatus.MIE` są dwiema bramkami. Program najpierw ustawia przyszły komparator, a dopiero potem otwiera obie bramki.

BLOCK A2 STRUCTURE · AKTYWNE - wspólny driver i stan dowodowy

Wspólny nagłówek zawiera dostęp MMIO, bezpieczny zapis komparatora i operacje CSR. Task przechowuje osobno `deadline`, czas zaobserwowany, `mcause`, liczbę ISR, pending przed i po oraz znacznik wznowienia.

BLOCK A3 DISPATCH · AKTYWNE - przyczyna 7 wybiera handler timera

Startup zapisuje do `mtvec` adres tablicy z bitem trybu vectored. Machine timer ma `cause=7`, więc rdzeń wybiera slot 7, którego skok wiąże się z silnym symbolem `isr_machine_timer` zamiast słabego handlera domyślnego. Atrybut GCC kończy funkcję instrukcją `mret`.

BLOCK A4 APPLICATION · AKTYWNE - one-shot z jawnym kryterium zakończenia

Main planuje `deadline` 1200 taktów po spójnym odczycie. Sukces wymaga dokładnie jednego wejścia, poprawnego `mcause`, obserwacji nie wcześniejszej niż `deadline`, skasowanego pending po rearmie i wykonania instrukcji po `wfi`.

BLOCK A5 FLOW · AKTYWNE - source → pending → enable → handler → effect

Rosnące `mtime` osiąga komparator i ustawia pending. Otwarty `MTIE` oraz `MIE` pozwalają wejść do wektora. Handler zapisuje obserwację i `mcause`, przesuwając komparator na `UINT64_MAX`, publikuje `count`, a `mret` przywraca przerwany przepływ. Main ustawia `resumed` i zamyka maski.

BLOCK A6 STATE · AKTYWNE - DISARMED, ARMED, PENDING, HANDLED, RESUMED

Komparator maksimum reprezentuje `DISARMED`. Przyszły `deadline` tworzy `ARMED`. Po osiągnięciu terminu urządzenie jest `PENDING`, handler ponownie zapisuje maksimum i publikuje `HANDLED`, a kod po pętli `wfi` potwierdza `RESUMED`.

BLOCK A7 RUNTIME · AKTYWNE - mcause, mip i mret

Raport z RTL ma: `count=1`, `mcause=0x80000007`, `pending` 0/0, `resumed=1` i `deadline_reached=1`. Objdump ma pokazać zapis trzech połówek `mtimecmp` w handlerze oraz końcowe `mret`; zwykle `ret` nie byłoby poprawnym powrotem z pułapki.

BLOCK A8 PATTERNS · AKTYWNE - arm before unmask, clear source before publish

Najpierw ustaw bezpieczny stan urządzenia, potem włącz źródło w `mie`, na końcu globalne `MIE`. W ISR usuń poziomowe źródło przed opublikowaniem zakończenia; inaczej powrót może natychmiast wejść ponownie.

BLOCK Ćwiczenie · Przewidywanie → wykonanie → wniosek

Ułóż w poprawnej kolejności: wyłącz maski, ustaw przyszły `mtimecmp`, sprawdź pending, włącz `MTIE` i `MIE`, czekaj w `wfi`, przesun komparator w ISR, wróć przez `mret`. Dla każdego kroku nazwij `source`, `pending`, `enable`, decyzję dispatchera albo efekt.

Evidence: Przewidywana sekwencja `source-pending-enable-dispatch-effect`, log RTL Hazard3 z `mcause` i kodem wyjścia, fragment listingu z wektorem oraz `mret` i wniosek o skasowaniu poziomowego źródła.

Acceptance: Oracle Hazard3: `count=1`, `mcause=0x80000007`, `pending_before=0`, `pending_after=0`, `resumed=1`, `deadline_reached=1`, `pass=1` i kod wyjścia 0. Listing ISR kończy się `mret`.

	TITLES timer: od mtime do okresowego deadline'u				VER v00.01	DATE/TIME 2026-07-20T00:00:00+02:00		
	PROJECT Freestanding C na RV32I				SERIES Freestanding RV32I and K&R	CARD 11/13		SHEET 7/8
	DIR: Inf.	PRG:	CORE	SCOPE	LEVEL	POS:		GITEA UUID f37170be-e988-5f5d-b3cf-6e179a9cd54
	GITEA URL: https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-machine-timer							GITEA CARD UUID 45f574de-1c7b-5f41-a3af-f1632024f102
HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/f37170be-e988-5f5d-b3cf-6e179a9cd54								

TASK TASK03 · Okresowe deadline'y bez dryfu fazy
6c330959-1a78-5068-8c59-6b18f712122c
BLOCK A1 CONTEXT · AKTYWNE - harmonogram i wykonanie to dwie osie Zachowany przykład: src/tasks/task03_periodic_absolute_deadline.c. Scheduled opisuje idealną oś czasu usługi, observed chwilę rzeczywistego wejścia do pomiaru w ISR, a lateness ich różnicę. Okres ma aktualizować harmonogram, nie kopiować opóźnienia wykonania do przyszłej fazy.
BLOCK A2 STRUCTURE · AKTYWNE - cztery rekordy ticków Trzy tablice po cztery elementy zapisują deadline, obserwację i lateness. g_task03_next_deadline jest stanem planisty, a count publikuje liczbę kompletnych rekordów. Stała period wynosi 1600 taktów modelu.
BLOCK A3 DISPATCH · AKTYWNE - każdy MTIP trafia przez slot 7 Dla każdego z czterech terminów rdzeń koduje interrupt bit i cause=7 w mcause, a tryb vectored wybiera ten sam isr_machine_timer. Handler sprawdza mcause przy każdym wejściu; licznik causes=4 dowodzi czterech decyzji dispatchera.
BLOCK A4 APPLICATION · AKTYWNE - stała faza i zmierzone spóźnienie Dla i>0 musi zachodzić scheduled[i]-scheduled[i-1]=1600. Dla każdego rekordu observed nie może być wcześniejsze od scheduled, a lateness musi być dokładnie ich różnicą. Po czwartym ticku comparator wraca do maksimum.
BLOCK A5 FLOW · AKTYWNE - sample, record, advance, rearm, publish ISR najpierw pobiera indeks i aktualny deadline, potem czyta czas oraz zapisuje rekord. Następnie oblicza deadline+period, uzbraja ten bezwzględny termin albo rozbraja po ostatnim ticku, a count publikuje dopiero na końcu. Main czeka w wfi do czterech rekordów.
BLOCK A6 STATE · AKTYWNE - TICK0 → TICK4 i niezmienna faza Stan count przechodzi 0,1,2,3,4; razem z każdym przejściem next_deadline rośnie dokładnie o period. Po count=4 urządzenie jest DISARMED. Tablice pozostają historią czterech zakończonych przejść i są analizowane dopiero po zamknięciu masek.
BLOCK A7 RUNTIME · AKTYWNE - cztery terminy i 40 taktów bieżącego RTL Bieżący raport podaje deadline low 0x96b, 0xfab, 0x15eb, 0x1c2b; obserwacje są o 0x28 późniejsze. Ważne są relacje step_ok, ordered i late_math, ponieważ absolutne wartości oraz 40 taktów mogą się zmienić po zmianie RTL, optymalizacji lub prologu ISR.
BLOCK A8 PATTERNS · AKTYWNE - absolute periodic deadline deadline += period zachowuje fazę. Gdyby następny termin powstał wyłącznie jako observed + period, jego przesunięcie względem osi absolutnej byłoby równe bieżącemu lateness; powtarzanie tej reguły kumuluje dryf. Osobna polityka musi zdecydować, czy po dużym opóźnieniu nadrabiać, pomijać, czy zgłaszać błąd.
BLOCK Ćwiczenie · Przewidywanie → wykonanie → wniosek Dla czterech ticków porównaj scheduled, observed i lateness. Udowodnij stały krok 1600 oraz wyprowadź, o ile przesunąłby następną fazę model now+period. Wyjaśnij granicę: przykład nie nadrabia dowolnej liczby całkowicie pominiętych okresów. Evidence: Przewidywana sekwencja czterech deadline'ów, log RTL Hazard3 z planowanymi i obserwowanymi czasami, kontrola różnic i kodu wyjścia oraz wniosek porównujący deadline+period z now+period. Acceptance: Oracle Hazard3: count=causes=4, period=1600, step_ok=ordered=late_math=naive_shift_ok=pass=1. Bieżący pomiar daje cztery lateness po 40 taktów, ale test nie uznaje liczby 40 za kontrakt sprzętowy.

	TITLE timer: od mtime do okresowego deadline'u						VER. v00.01	DATETIME 2026-07-20T00:00:00+02:00									
	PROJECT Freestanding C na RV32I						SERIES Freestanding RV32I and K&R			CARD 11/13	SHEET 8/8						
	DIR.	Inf.	PRG.	CORE	SCOPE	LEVEL	POS.	GITEA UUID f37170be-e988-5f5d-b3cf-6ea179a9cd54									
	GITEA https://zal-gitea.mpabi.pl/edu-inf/lab-rv32i-c-machine-timer						CARD UUID 45f574de-1c7b-5f41-a3af-f1632024ff0f										
	HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/f37170be-e988-5f5d-b3cf-6ea179a9cd54																

BLOCK TASK03 · Zapis dowodu ucznia

1. Przewidywanie przed uruchomieniem

2. Log RTL: deadline, observed i lateness

3. Dowód stałego kroku i braku dryfu fazy

4. Wniosek: reguła języka lub kontrakt targetu potwierdzony przez pomiar