

TITLE		UART - polling i przerwania RX				VER.	v00.01		DATETIME	2026-07-20T00:00:00+02:00						
PROJECT		Freestanding C na RV32I				SERIES		Freestanding RV32I and K&R		CARD	12/13		SHEET	1/8		
SUBJ.		Inf.		PRG.	CORE	SCOPE	LEVEL	POS.	GITEA UUID		64f7c516-78b1-58ee-8619-a82d7061079f		e766c4bb-2a43-5f46-b1d0-c3383a12920c		AUTHOR N. PABISZCZAK	
GITEA		https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-uart														
HTML		https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/64f7c516-78b1-58ee-8619-a82d7061079f														

Cel karty

Uczeń odróżnia polling rejestru statusu od odbioru sterowanego zdarzeniem. Dla ścieżki przerwania wskazuje osobno: bajt w RX FIFO, stan linii external IRQ 16, selekcję przez kontroler Hazard3, wywołanie ISR i efekt konsumowany później przez main.

Zakres i zachowane przykłady

Karta korzysta z laboratoryjnego modelu UART Hazard3. Rejestr TB_UART_TEST_RX jest wyłącznie kontrolowanym bodźcem testbench; nie występuje w produkcyjnym UART ani na RP2350. Po wstrzyknięciu bajtu Task02 i Task03 przechodzą jednak przez rzeczywisty kontroler external IRQ, mtvec, dispatcher i mret modelowanego rdzenia.

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	STATUS, DATA i kontrolowany RX stimulus	kluczowe
IRQ	Task02	FIFO, pending, enable, dispatcher i mret	kluczowe
ISR	Task03	krótki ISR, jawne przepełnienie i konsumpcja w main	kluczowe

TITLE										UART - polling i przerwania RX		VER.		v00.01		DATE/TIME		2026-07-20T00:00:00+02:00									
PROJECT										Freestanding C na RV32I		SERIES		Freestanding RV32I and K&R		CARD		12/13		SHEET		2/8					
SUBJ.										Inf.		PRG.		CORE		SCOPE		LEVEL		POS.		GITEA UUID CARD UUID		64f7c516-78b1-58ee-8619-a82d7061079f		e766c4bb-2a43-5f4eeb10e-z3383a42932c NOTHUR W. PAB182232E	
GITEA https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-uart																											
HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/64f7c516-78b1-58ee-8619-a82d7061079f																											

1 Polling pyta o stan w rytmie programu

Rejestr `STATUS.RX_AVAIL` opisuje niepusty RX FIFO, a odczyt `DATA` usuwa jeden bajt. W `Task01` przerwanie RX pozostaje wyłączone: program sam wykonuje kolejne odczyty statusu. Jest to poprawne dla prostego, krótkiego oczekiwania, lecz czas CPU zależy od częstotliwości odpytywania. Zapis `TB_UART_TEST_RX='P'` jest bodźcem laboratoryjnym, nie częścią sterownika produkcyjnego.

2 RX IRQ jest ścieżką pięciu stanów

W `Task02` bajt trafia do FIFO i model podnosi external IRQ 16 tylko wtedy, gdy `CTRL.RX_IRQ_EN=1`. Kontroler Hazard3 próbkuje i udostępnia bieżący poziom pending; `mstatus.MIE`, `mie.MEIE` i enable linii decydują, czy rdzeń wejdzie do wektora. Dispatcher odczytuje `meinext`, wybiera wpis 16 z tablicy i wykonuje `jalr`. Odczyt `DATA` opróżnia FIFO, więc linia poziomowa i widoczny pending opadają; dopiero `mret` wraca do normalnego kodu.

3 ISR przekazuje minimum, main nadaje znaczenie

`Task03` nie analizuje komendy w ISR. Handler zawsze opróżnia `DATA`, a potem albo zapisuje bajt do jednej komórki, albo zwiększa licznik odrzuceń, gdy komórka jest pełna. Polityka **drop-newest** jest celowo jawna i mierzalna. To nie jest kolejka ani synchronizacja RTOS: pojedyncze 32-bitowe obiekty `volatile` i kontrolowana kolejność wystarczają wyłącznie dla tego jednego rdzenia i tego przykładu.

	TITLE UART - polling i przerwania RX		VER v00.01	DATE/TIME 2026-07-20T00:00:00+02:00		
	PROJECT Freestanding C na RV32I		SERIES Freestanding RV32I and K&R		CARD 12/13	
	SHEET 3/8					
	GITEA https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-uart		HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/64f7c516-78b1-58ee-8619-a82d7061079f			

4 Zadania — zachowane przykłady w profilu A1–A8

TASK TASK01 · Polling jednego bajtu

611a873f-8d37-5084-9ab6-1230e2505249

BLOCK A1 CONTEXT · AKTYWNE - polling w granicy modelu

Zachowany przykład: `src/tasks/task01_uart_polling.c`.

Odpowiedzialność przykładu zaczyna się od kontrolowanego wstrzyknięcia P do RX FIFO i kończy na odczycie P oraz zapisie T do TX. Nie konfiguruje kontrolera przerwań. `TEST_RX` należy wyłącznie do testbench; `DATA`, `STATUS` i `CTRL` modelują interfejs peryferium.

BLOCK A2 STRUCTURE · AKTYWNE - cztery rejestry i FIFO

Pod bazą `UART0` leżą `DATA +0`, `STATUS +4`, `CTRL +8` i testowy `TEST_RX +12`. FIFO nie jest bezpośrednio adresowalne: `RX_AVAIL` mówi tylko, czy zawiera co najmniej jeden bajt, a `DATA` zwraca i usuwa element czołowy.

BLOCK A3 DISPATCH · N/D

N/D. `RX_IRQ` pozostaje wyłączony, więc nie ma wejścia przez `mtvec`, wyboru `ISR` ani pośredniego `jalr`; wszystkie funkcje są wywołaniami bezpośrednimi normalnego kodu.

BLOCK A4 APPLICATION · AKTYWNE - przewidywanie bitów

Bez klienta TCP status po wstrzyknięciu P ma wartość `0x3`: `RX_AVAIL` i `TX_READY`. `RX_IRQ` jest zerem, bo `CTRL` nie włącza źródła. Po odczycie `DATA` FIFO jest puste, więc zostaje `0x2`. Znaki mają wartości `P=80` i `T=84`.

BLOCK A5 FLOW · AKTYWNE - inject, status, data, status, tx

Kolejność jest częścią dowodu: wyłącz `IRQ`, wstrzyknij P, odczytaj pierwszy status, odczytaj `DATA`, odczytaj drugi status, zapisz T. Zamiana `DATA` z pierwszym statusem zniszczyłaby obserwację `RX_AVAIL=1`.

BLOCK A6 STATE · AKTYWNE - EMPTY do P do EMPTY

`RX_FIFO` przechodzi `EMPTY -> ['P'] -> EMPTY`. Status jest funkcją tego stanu i nie przechowuje osobnej kopii flagi. Zapis TX jest widoczny jako T w logu testbench; bez klienta TCP model nie zachowuje tego bajtu w kolejce sieciowej. Nie zmienia to stanu RX.

BLOCK A7 RUNTIME · AKTYWNE - adresy 0xc00002xx

W listingu znajdź zapisy i odczyty pod bazą `0xc0000200`. W checkpointcie globale mają wartości 3, 80, 2 i 84. Kod 0 pochodzi z pełnego gate, a nie z samego faktu, że program dotarł do `_exit`.

BLOCK A8 PATTERNS · AKTYWNE - status-before-data

Nazwany wzorzec polling RX brzmi: sprawdź availability, dopiero potem odczytaj destructive `DATA`. Jest prosty i deterministyczny, ale zużywa czas CPU, jeśli pętla oczekiwania nie ma ograniczenia ani innej pracy.

BLOCK Ćwiczenie · Przewidywanie → wykonanie → wniosek

Przewidź bity statusu przed i po odczycie P. Wskaż, która operacja tworzy bodziec testbench, która tylko obserwuje FIFO, a która usuwa bajt. Potwierdź także zapis T do rejestru TX. **Evidence:** Tabela bitów `STATUS` przed i po `DATA`, odczyt globali `Task01`, wskazanie transakcji MMIO w listingu `RV32I` oraz kod wyjścia `Hazard3`.

Acceptance: `Hazard3`: `status_before=0x3`, `rx_byte='P'=80`, `status_after=0x2`, `tx_byte='T'=84`, osobna linia T w logu testbench, `pass=1` i kod wyjścia 0.

	TITLE				UART - polling i przerwania RX				VER.		v00.01				DATETIME				2026-07-20T00:00:00+02:00																		
	PROJECT								Freestanding C na RV32I								SERIES				Freestanding RV32I and K&R				CARD		12/13		SHEET		4/8						
	SUBJ.		PRG.		CIR		SCOPE		LEVEL		POS.		GITEA UUID								64f7c516-78b1-58ee-8619-a82d7061079f								e766c4bb-2a43-5f4eeb10a-33383a42932c								
	AUTHOR		W.		PAB182C3E																																
GITEA https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-uart																																					
HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/64f7c516-78b1-58ee-8619-a82d7061079f																																					

BLOCK TASK01 · Zapis dowodu ucznia

1. Przewidywanie przed uruchomieniem

.....

.....

2. Status i bajt w modelu UART

.....

.....

3. Transakcje MMIO RV32I i kod wyjścia

.....

.....

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

.....

.....

	TITLE UART - polling i przerwania RX				VER v00.01	DATE/TIME 2026-07-20T00:00:00+02:00			
	PROJECT Freestanding C na RV32I				SERIES Freestanding RV32I and K&R	CARD 12/13	SHEET 5/8		
	DIR: Inf.	PRG:	CORE	SCOPE	LEVEL	POS:	GITEA UUID 64f7c516-78b1-58ee-8619-a82d7061079f		e766c4bb-2a43-5f4eeb10e-p3383a42932c
	GITEA https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-uart						AUTHOR W. Pab1222322		
HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/64f7c516-78b1-58ee-8619-a82d7061079f									

TASK TASK02 · Odbiór R przez external IRQ 16

1cd0396c-3eea-5a66-92a9-48439de20b15

BLOCK A1 CONTEXT · AKTYWNE - pięć granic odpowiedzialności

Zachowany przykład: `src/tasks/task02_uart_rx_interrupt.c`.

TEST_RX tworzy bodziec, UART utrzymuje FIFO i linię 16, kontroler Hazard3 utrzymuje pending/priority/enable, dispatcher wybiera wpis, a ISR odczytuje DATA. Normalny kod tylko konfiguruje, włącza globalny bit, czeka i ocenia dowód.

BLOCK A2 STRUCTURE · AKTYWNE - stan peryferium, CSR i tabela

Stan obejmuje CTRL.RX_IRQ_EN, niepusty FIFO, pending linii 16, tablicę `_external_irq_table[16]`, priorytet, mie.MEIE i mstatus.MIE. Żaden pojedynczy bit nie wystarcza do wejścia w ISR.

BLOCK A3 DISPATCH · AKTYWNE - meinext do wpisu 16

Wektor machine external IRQ wchodzi do `isr_external_irq`. Odczyt `meinext` zwraca gotowy offset tablicy — numer linii 16 przesunięty o dwa bity. Dispatcher dodaje go do bazy `_external_irq_table` i wykonuje `jalr` do `task02_uart_handler`. To jest rzeczywisty wybór wykonawcy, nie zwykłe statyczne wywołanie.

BLOCK A4 APPLICATION · AKTYWNE - R i status 0x13

Przed globalnym enable pending ma być 1, lecz licznik handlera nadal 0. Po enable ISR ma zobaczyć `RX_AVAIL`, `TX_READY` i `RX_IRQ`, czyli 0x13, odczytać `R=82` i opublikować numer 16.

BLOCK A5 FLOW · AKTYWNE - konfiguracja przed globalnym enable

Najpierw MIE jest wyzerowane, potem ustawiane są handler, priorytet, enable linii, MEIE i CTRL. Dopiero po wstrzyknięciu R i zmierzeniu pending program ustawia MIE. ISR czyta DATA, dispatcher odtwarza kontekst i wykonuje `mret`.

BLOCK A6 STATE · AKTYWNE - PENDING, ACTIVE, DRAINED

Po bodźcu stan jest PENDING: FIFO zawiera R, linia 16 jest aktywna, lecz globalny gate blokuje wejście. Po MIE przechodzi do ACTIVE. Odczyt DATA tworzy DRAINED: FIFO pusty, linia i pending opadają, a licznik wynosi 1.

BLOCK A7 RUNTIME · AKTYWNE - mcause 0x8000000b i meinext 16

Wejście sprzętowe ma machine external interrupt cause 11, czyli `mcause=0x8000000b`; konkretną linię 16 wybiera dopiero kontroler przez `meinext`. Listing ma zawierać zapis tablicy, operacje CSR, pośredni `jalr` i `mret`.

BLOCK A8 PATTERNS · AKTYWNE - drain level-triggered source

Dla źródła poziomowego ISR musi usunąć warunek w peryferium, tutaj opróżnić DATA. Samo wejście do handlera nie jest acknowledge. Gdyby FIFO pozostał niepusty, linia byłaby nadal aktywna i przerwanie wróciłoby po `mret`.

BLOCK Ćwiczenie · Przewidywanie → wykonanie → wniosek

Rozpisz osobno bity CTRL, pending linii 16, mstatus.MIE, mie.MEIE i enable kontrolera. Przewidź stan przed globalnym enable, numer wybrany przez `meinext` oraz czynność, która usuwa źródło poziomowe.

Evidence: Ślad źródło→pending→enable→dispatcher→ISR→resume, wartości globali w checkpointcie, fragment `jalr` z `irq_dispatch.S` i kod wyjścia Hazard3.

Acceptance: Hazard3: `pending_before_enable=1`; jeden handler IRQ 16 widzi status 0x13 i `byte='R'=82`; po odczycie DATA `pending_after=0`, `pass=1` i kod 0.



TITLE				UART - polling i przerwania RX				VER.		v00.01				DATETIME				2026-07-20T00:00:00+02:00													
PROJECT								Freestanding C na RV32I								SERIES				Freestanding RV32I and K&R				CARD		12/13		SHEET		6/8	
SUBJ.		Inf.		PRG.		CORE		SCOPE		LEVEL		POS.		GITEA UUID		64f7c516-78b1-58ee-8619-a82d7061079f		e766c4bb-2a43-5f4eeb10a-33383a42932c		AUTHOR		W. Pab182232E									
GITEA																https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-uart															
HTML																https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/64f7c516-78b1-58ee-8619-a82d7061079f															



BLOCK

TASK02

·

Zapis dowodu ucznia

1. Przewidywanie przed uruchomieniem

.....

.....

2. Macierz pending i enable przed ISR

.....

.....

3. Numer IRQ, status, bajt i powrót mret

.....

.....

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

.....

.....

	TITLE		UART - polling i przerwania RX		VER	v00.01		DATE/TIME	2026-07-20T00:00:00+02:00											
	PROJECT		Freestanding C na RV32I		SERIES	Freestanding RV32I and K&R		CARD	12/13			SHEET	7/8							
	REV.	Inf.	PRG.		CORE		SCOPE		LEVEL			POS.		GITEA UUID	64f7c516-78b1-58ee-8619-a82d7061079f		e766c4bb-2a43-5f4eeb10e-p3383a42932c	AUTHOR	W. Pab122232E	
	GITEA https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-uart																			
HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/64f7c516-78b1-58ee-8619-a82d7061079f																				

TASK TASK03 · Jednokomórkowa skrzynka drop-newest

c60d485a-0e23-522a-a06a-78c86be36f76

BLOCK A1 CONTEXT · AKTYWNE - ISR transportuje, main konsumuje

Zachowany przykład: `src/tasks/task03_uart_mailbox.c`.

Handler odpowiada za szybkie opróżnienie DATA i próbę publikacji jednego bajtu. Nie interpretuje komendy. Main odpowiada za konsumpcję. Gdy skrzynka jest pełna, kontrakt jawnie odrzuca najnowszy bajt i liczy stratę.

BLOCK A2 STRUCTURE · AKTYWNE - payload, flaga i telemetria

Skrzynkę tworzą `mailbox_byte` i `mailbox_full`. Osobne liczniki `irq`, `accepted` i `dropped` nie sterują algorytmem; są telemetrią dowodu. Każdy obiekt ma szerokość 32 bitów i jest `volatile` w tym jednordzeniowym kontrakcie `bare-metal`.

BLOCK A3 DISPATCH · AKTYWNE - ten sam IRQ, inny odbiorca

Linia 16 przechodzi przez ten sam dispatcher co Task02, lecz wpis tablicy wskazuje `task03_uart_handler`. Dynamiczny wybór kończy się pośrednim `jalr`; decyzja `accepted/drop` odbywa się dopiero wewnątrz handlera na podstawie danych.

BLOCK A4 APPLICATION · AKTYWNE - A przyjęte, B odrzucone, C przyjęte

Po A skrzynka jest pełna i `accepted=1`. B wywołuje drugie IRQ, DATA zostaje opróżnione, lecz `payload A` pozostaje, a `dropped` rośnie do 1. Main pobiera A. Po C pusta skrzynka znów przyjmuje dane; main pobiera C.

BLOCK A5 FLOW · AKTYWNE - drain przed decyzją publish/drop

ISR najpierw odczytuje DATA, dzięki czemu usuwa źródło poziomowe niezależnie od stanu skrzynki. Następnie zwiększa `irq_count` i wybiera publikację albo drop. Main czeka na konkretną liczbę obsługi, a nie na sam upływ pętli.

BLOCK A6 STATE · AKTYWNE - EMPTY, FULL(A), FULL(A), EMPTY, FULL(C), EMPTY

Pełny ślad skrzynki to `EMPTY -> FULL(A) -> FULL(A) -> EMPTY -> FULL(C) -> EMPTY`. Drugie przejście nie zmienia payloadu, ale zmienia licznik `dropped`; dlatego stan domenowy i telemetria muszą być czytane razem.

BLOCK A7 RUNTIME · AKTYWNE - trzy wejścia i trzy mret

Checkpoint ma pokazać `irq=3`, `accepted=2`, `dropped=1`, 65, 67 i `full=0`. W listingu wskaż trzy klasy operacji: MMIO DATA, zapisy globali przez ISR oraz odtworzenie kontekstu i `mret` w dispatcherze.

BLOCK A8 PATTERNS · AKTYWNE - bounded mailbox z jawną utratą

Jednokomórkowa skrzynka ogranicza czas i pamięć ISR. Jej ważną częścią jest polityka przeciążenia i licznik strat. Nie należy nazywać jej kolejką; nie zachowuje serii zdarzeń i nie zastępuje synchronizacji wielordzeniowej ani RTOS.

BLOCK Ćwiczenie · Przewidywanie → wykonanie → wniosek

Prześledź A, B i C. Dla każdego bajtu zapisz stan FIFO, `mailbox_full`, decyzję ISR i późniejszy odczyt main. Wyjaśnij, dlaczego B jest odrzucone, a mimo to DATA musi zostać odczytane.

Evidence: Tabela trzech zdarzeń A/B/C, liczniki ISR, dwie wartości odebrane przez main, stan końcowy skrzynki oraz kod wyjścia Hazard3.

Acceptance: Hazard3: `irq_count=3`, `accepted=2`, `dropped=1`, `first_consumed='A'=65`, `second_consumed='C'=67`, `mailbox_full=0`, `pass=1` i kod 0.

	TITLE				UART - polling i przerwania RX				VER.		v00.01				DATETIME				2026-07-20T00:00:00+02:00																																																												
	PROJECT								SERIES								CARD				SHEET																																																										
	Freestanding C na RV32I								Freestanding RV32I and K&R								12/13				8/8																																																										
	SUBJ. Inf.								PRG.								CIR								SCOPE								LEVEL								POS.								GITEA UUID								64f7c516-78b1-58ee-8619-a82d7061079f								e766c4bb-2a43-5f4eeb10a-33383a42932c								AUTHOR W. Pab182232E						
GITEA https://zsl-gitea.mpabi.pl/edu-inf/lab-rv32i-c-uart																																																																															
HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/64f7c516-78b1-58ee-8619-a82d7061079f																																																																															

BLOCK TASK03 · Zapis dowodu ucznia

1. Przewidywanie przed uruchomieniem

.....

.....

2. Ślad ISR dla A, B i C

.....

.....

3. Konsumpcja main, liczniki i kod wyjścia

.....

.....

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

.....

.....