

O ile nie zaznaczono inaczej, zagadnienia dotyczą mikrokontrolerów ARM Cortex-Mx na przykładach wykorzystywanych w trakcie zajęć

1. Magistrale wewnętrzne w mikrokontrolerach ARM Cortex-M (AHB, APB), elementy konfiguracji modułów peryferyjnych (taktowanie).

Adres bazowy

- adres: 0x40000000

AHB1 offset

- offset: 0x00020000

AHB2 offset

- offset: 0x08000000
- Rejestr RCC powiązany z magistralą AHB2
- Porty GPIO powiązane z magistralą AHB1

2. Rejestry wewnętrzne mikrokontrolerów ARM Cortex-M.

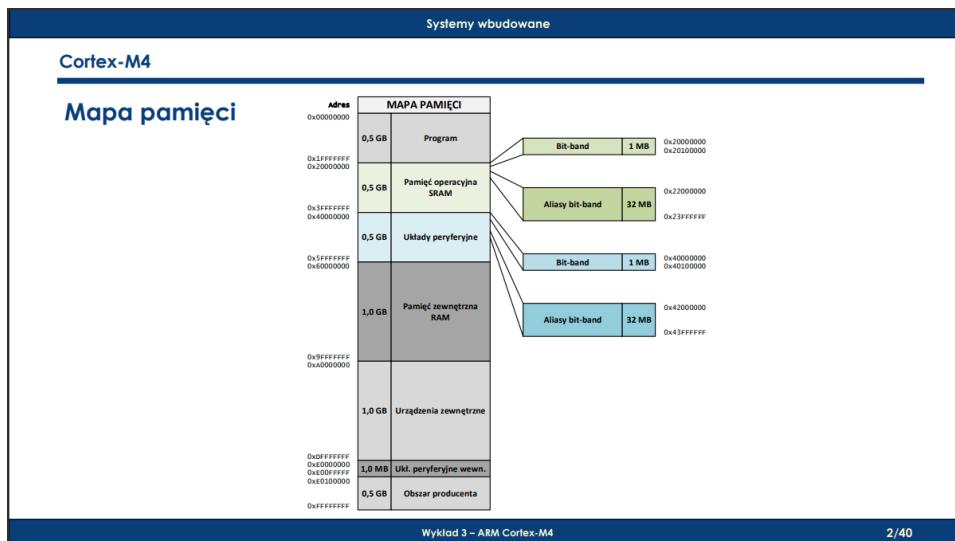
Systemy wbudowane												
Dokumentacja DS1307												
Rejestry wewnętrzne DS1307												
ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE		
00h	CH	10 Seconds			Seconds			Seconds	00–59			
01h	0	10 Minutes			Minutes			Minutes	00–59			
02h	0	12	10 Hour	10 Hour	Hours			Hours	1–12 +AM/PM 00–23			
		24	PM/ AM									
03h	0	0	0	0	0	DAY		Day	01–07			
04h	0	0	10 Date		Date			Date	01–31			
05h	0	0	0	10 Month	Month			Month	01–12			
06h	10 Year			Year			Year	00–99				
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—		
08h–3Fh								RAM 56 x 8	00h–FFh			
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0					
OUT	0	0	SQWE	0	0	RS1	RS0					

Rejestry wewnętrzne DS1307

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

RS1	RS0	SQW/OUT OUTPUT	SQWE	OUT
0	0	1Hz	1	X
0	1	4.096kHz	1	X
1	0	8.192kHz	1	X
1	1	32.768kHz	1	X
X	X	0	0	0
X	X	1	0	1

- SQWE – aktywacja wyjścia SQW/OUT. Domyślnie 0 (niekatywne). Jeśli „1” → wyjście z częstotliwością RS1-RS0, domyślnie „11” czyli 32,768 kHz

3. Ogólna struktura mapy pamięci mikrokontrolerów ARM Cortex-M na wybranych przykładach.**4. Rejestry konfiguracji i obsługi portów GPIO.**

Rejestry **GPIO**

- MODER
 - offset: 0x00
- OTYPER
 - offset: 0x04
- OSPEEDR
 - offset: 0x08
- PUPDR
 - offset: 0x0C
- ODR
 - offset: 0x14

5. Ogólna charakterystyka podstawowego zestawu instrukcji procesorów ARM.

Systemy wbudowane

Cortex M-4 **Instrukcje**

Wybrane **instrukcje** asemblera i objaśnienia [PM0214]

LDR	- Load
STR	- Store
ADD	- Add
SUB	- Subtract
CMP	- Compare
ORR	- logiczny OR
AND	- logiczny AND
CBZ	- Compare and branch if zero
BNE	- branch if not equal
B	- branch (to label)

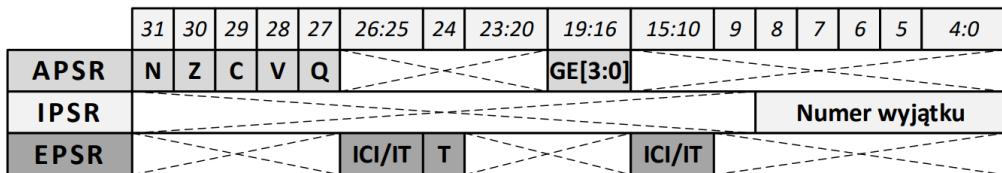
Wykład 1 – ARM Cortex-M4

31/35

6. Rejestr znaczników.

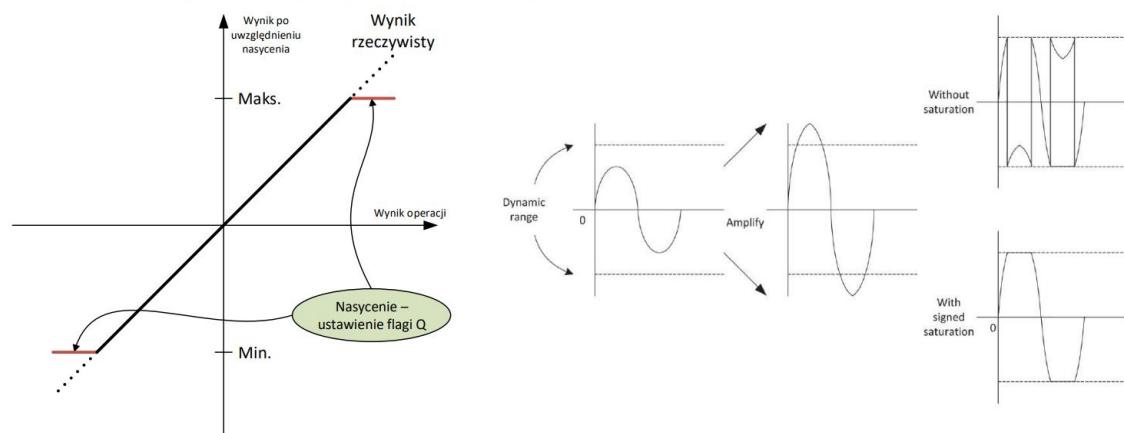
Znaczenie flag

- **N**(egative) – liczba ujemna
- **Z**(ero)
- **C**(arry) – przeniesienie lub BRAK pożyczki
- **(o)V**(erflow) – nadmiar



Znaczenie flag

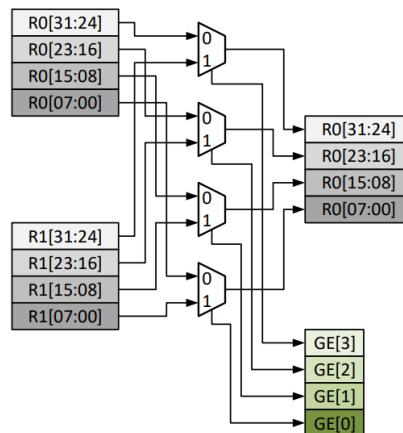
- **Q** – nasycenie (ang. saturation)



Cortex-M4

Znaczenie flag

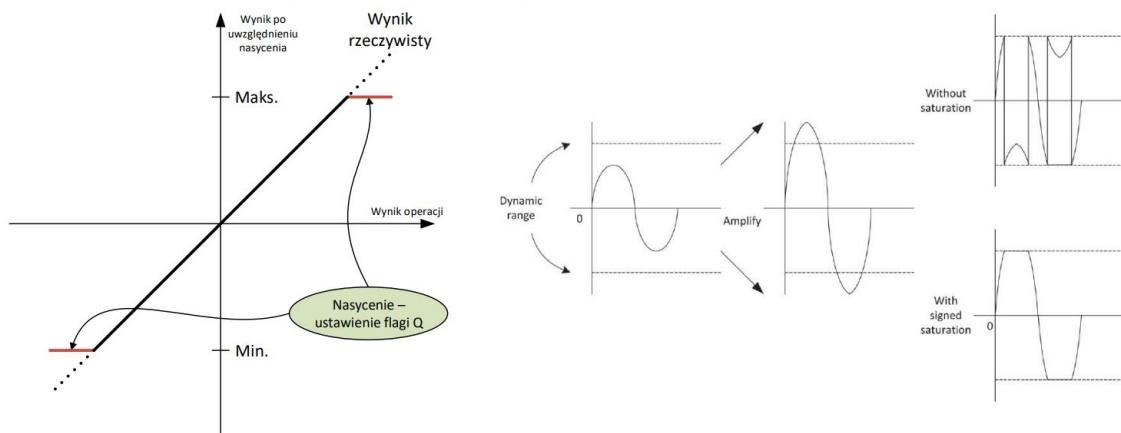
- **G(reater)E(qual)** – oddzielnie dla każdego bajta



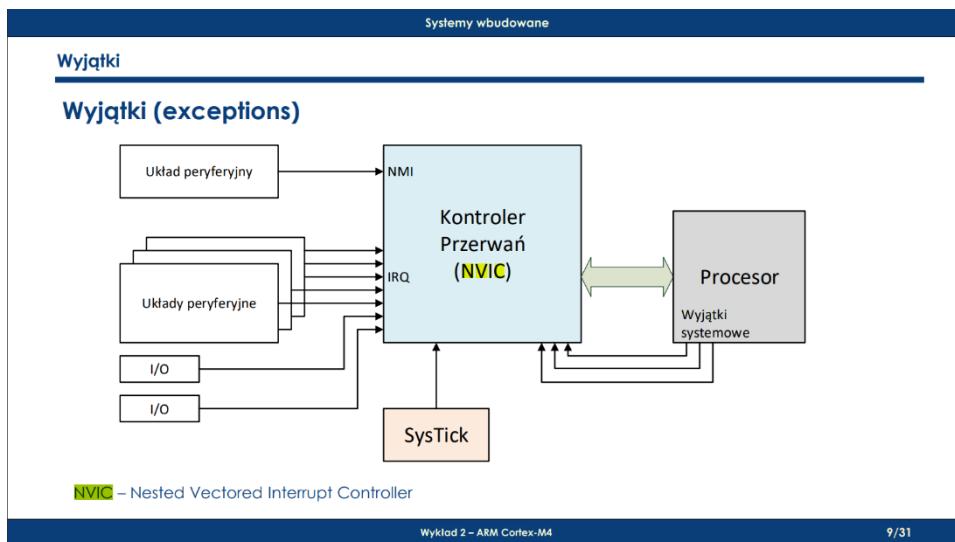
7. Arytmetyka saturacyjna.

Znaczenie flag

- **Q** – nasycenie (ang. saturation)



8. NVIC, rodzaje i typy wyjątków.



Wyjątki**Typy wyjątków**

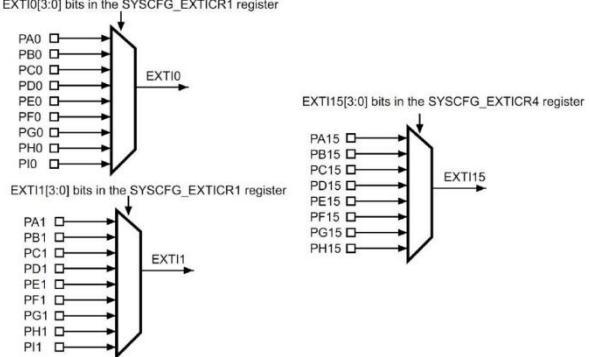
Numer wyjątku	Nr przerwania CMSIS	Typ wyjątku	Priorytet	Funkcja
1	-	Reset	-3	Reset
2	-14	NMI	-2	Przerwanie niemaskowalne
3	-13	HardFault	-1	Wszystkie klasy błędów w przypadku braku możliwości aktywacji procedury obsługi z powodu wyłączenia bądź maskowania
4	-12	MemManage	Ustawiany	Błąd zarządzania pamięcią; naruszenie zasad Memory Protection Unit (MPU), np. pobranie instrukcji z niewykonwalnego obszaru pamięci
5	-11	BusFault	Ustawiany	Odpowiedź błędu z magistrali systemowej
6	-10	Usage Fault	Ustawiany	Błędna instrukcja lub błędna próba zmiany stanu (np. przełączenie do stan ARM w Cortex-M3)
7-10	-	-	-	Zarezerwowane
11	-5	SVC	Ustawiany	Supervisor Call za pomocą instrukcji SVC
12	-4	Debug monitor	Ustawiany	Programowa obsługa uruchamiania (rzadko wykorzystywana)
13	-	-	-	Zarezerwowane
14	-2	PendSV	Ustawiany	Obecne żądanie System Service
15	-1	SYSTICK	Ustawiany	Układ czasowo-licznikowy SysTick
16-255	0-239	IRQ	Ustawiany	Wejścia przerwań (IRQ)

9. Ogólna charakterystyka priorytetów źródeł przerwań.**Wyjątki****Typy wyjątków**

Numer wyjątku	Nr przerwania CMSIS	Typ wyjątku	Priorytet	Funkcja
1	-	Reset	-3	Reset
2	-14	NMI	-2	Przerwanie niemaskowalne
3	-13	HardFault	-1	Wszystkie klasy błędów w przypadku braku możliwości aktywacji procedury obsługi z powodu wyłączenia bądź maskowania
4	-12	MemManage	Ustawiany	Błąd zarządzania pamięcią; naruszenie zasad Memory Protection Unit (MPU), np. pobranie instrukcji z niewykonwalnego obszaru pamięci
5	-11	BusFault	Ustawiany	Odpowiedź błędu z magistrali systemowej
6	-10	Usage Fault	Ustawiany	Błędna instrukcja lub błędna próba zmiany stanu (np. przełączenie do stan ARM w Cortex-M3)
7-10	-	-	-	Zarezerwowane
11	-5	SVC	Ustawiany	Supervisor Call za pomocą instrukcji SVC
12	-4	Debug monitor	Ustawiany	Programowa obsługa uruchamiania (rzadko wykorzystywana)
13	-	-	-	Zarezerwowane
14	-2	PendSV	Ustawiany	Obecne żądanie System Service
15	-1	SYSTICK	Ustawiany	Układ czasowo-licznikowy SysTick
16-255	0-239	IRQ	Ustawiany	Wejścia przerwań (IRQ)

10.Organizacja przerwań zewnętrznych (EXTI).**Przerwania****Przerwania zewnętrzne**

- możliwe przypadki grupowania większej liczby linii przerwań, np. EXTI10_15 dla portów A, B, ... X



????

11. Charakterystyka timera SysTick.

- **SysTick** → rdzeń ARM
 - licznik 24 bitowy zliczający w dół
 - możliwość automatycznego przeładowania
 - generowanie przerwania maskowalnego przy osiągnięciu 0
 - programowalne źródło sygnału zegarowego

12. Dostęp bitowy do pamięci.

Systemy wbudowane

Cortex-M4

Bit-banding – dostęp bitowy do pamięci

- Bajtowa organizacja pamięci
 - każdy adres wskazuje pojedynczą komórkę pamięci, ale np. odczyt pamięci do rejestru może pobrać całe 32 bitowe słowo, czyli 4 komórki pamięci
 - modyfikacja pojedynczego bitu wymaga odczytu bajtu (lub słowa), modyfikacji konkretnego bitu i ew. ponowny zapis bajtu
- Wyróżniony obszar pamięci → mapowanie w postaci aliasów o dostępie bitowym
 - każdy bit dostępny pod własnym adresem
- Obszar bit-band dostępny także w konwencjonalny sposób

Wykład 3 – ARM Cortex-M4 3/40

Systemy wbudowane

Cortex-M4

Bit-banding – dostęp bitowy do pamięci

- Adresy poszczególnych bitów:
$$Adr_{bit} = BitBandAlias_{area} + 32 \cdot Byte_{offset} + 4 \cdot Bit_{nr}$$

Na przykład dla rejestru ODR PORTD:

$$Adr_5 = 0x42000000 + 32 \cdot 0x0020414 + 4 \cdot 5 = 0x4264CBFC$$

0x40020414
PORTD:ODR[7:0] 7 6 5 4 3 2 1 0
0x4264CC4
0x4264CC00
0x4264CBFC
0x4264CBF8
0x4264CBF4
0x4264CBF0
0x4264CBEC
0x4264CBE8
Adres bitu nr 5

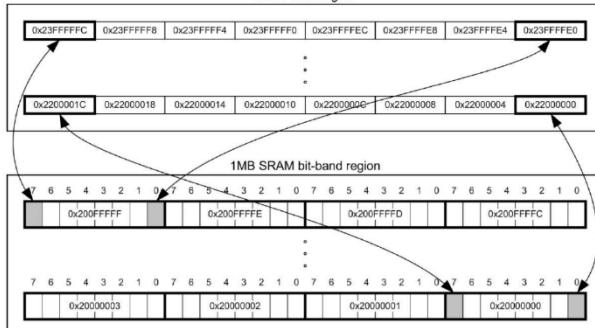
Wykład 3 – ARM Cortex-M4 4/40

Cortex-M4

Bit-banding – dostęp bitowy do pamięci

- pełna mapa adresów bit-band dla pamięci SRAM

32MB alias region



Cortex-M4

Bit-banding – dostęp bitowy do pamięci

- Zapis słowa do regionu aliasów
 - aktualizacja tylko 1 bitu
- Zapis liczby z wartością najmniej znaczącego bitu wynoszącą 1 zapisuje „1” do określonego bitu
- Zapis liczby z wartością najmniej znaczącego bitu wynoszącą 0 zapisuje „0” do określonego bitu
- Bitы z zakresu [31:1] w słowie aliasu są ignorowane
 - zapis 0xFF ma taki sam skutek jak zapis 0x01
 - zapis 0x0E ma taki sam skutek jak zapis 0x00

13. Zegar czasu rzeczywistego (RTC) – charakterystyka i możliwości.

Niezależny układ czasowo-licznikowy zliczający w kodzie BCD (Binary Coded Decimal). Rzeczywisty czas, kalendarz, dwa programowalne przerwania alarmu, programowalne wybudzenie, automatyczne wybudzanie w celu niskiego poboru energii.

14.Układ czasowo-licznikowy na przykładzie TIM1.

Cortex-M4 STM32F407

TIM1 (TIM8)

- Zaawansowany układ sterowania
- **16-bitowy** licznik automatycznie przeładowywany
- programowalny preskaler.
- Zastosowanie
 - pomiar długości impulsów sygnałów wejściowych
 - generowanie przebiegów wyjściowych
 - porównywanie wyjść,
 - PWM,
 - uzupełniające PWM z wprowadzaniem czasu martwego (ang. dead time).
- Długości impulsów i okresy przebiegu mogą być modulowane od kilku mikrosekund do kilku milisekund za pomocą preskalera timera i preskalera kontrolera zegara RCC.
- Niezależność zegarów sterujących (TIM1 i TIM8) oraz zegarów ogólnego przeznaczenia (TIMx) – brak współdzielenia zasobów – możliwość synchronizacji

15. Kontroler bezpośredniego dostępu do pamięci (DMA) – możliwości.

Przenoszenie danych między różnymi obszarami pamięci (włącznie z urządzeniami zewnętrznymi). Dwa kanały o tym samym priorytecie, kanał o niższym numerze ma pierwszeństwo.

W połączeniu z Bus Matrix brak ryzyka blokady dostępu magistrali, cztery priorytety na kanał

16. Karta SD – magistrale, sterowanie, systemy plików, możliwości biblioteki FatFs.

Magistrale???

Karty SD

Sterowanie

- Wybrane instrukcje

Polecenie	Opis
CMD0	Zerowanie karty, włączenie trybu SPI
CMD12	Zakończenie transmisji wielu bloków
CMD16	Konfiguracja długości bloku odczytu/zapisu
CMD17	Odczyt bloku określonej wielkości
CMD24	Zapis bloku określonej wielkości
CMD32	Przesłanie adresu pierwszego bloku do skasowania
CMD33	Przesłanie adresu ostatniego bloku do skasowania
CMD38	Kasowanie zakresu bloków wyznaczonego poleceniami CMD32, CMD33

Karty SD**System plików**

- podział na sektory
- rozmiar sektora zależy od wielkości karty
- sektor zawiera co najwyżej dane jednego pliku
 - w przypadku, gdy plik jest mniejszy od wielkości sektora zajęty jest cały sektor
 - pliki większe od sektora zajmują wiele sektorów

Systemy plików kart SD

- FAT12
- FAT16
- FAT32
- exFAT

Karty SD**Obsługa programowa**

- biblioteka FatFs
 - http://elm-chan.org/fsw/ff/00index_e.html
- obsługa systemów FAT12, FAT16, FAT32, exFAT
- obecna wersja 0.15 (z 6.11.2022 r., stan na dzień 25.03.2023 r.)
- bezpośrednie wsparcie z poziomu STM32CubeMX (wersja 0.12c)

**17. Sieci przemysłowe (CAN, LIN, MOST, CAN-FD) – charakterystyka i zastosowania.****CAN****CAN – Controller Area Network**

- Najbardziej rozpowszechniona sieć automotive
- Zaprojektowana przez firmę Bosch w latach 80'
- 8 bajtów na ramkę, prędkość do 1 Mb/s
- działania na zasadzie rozgłaszenia
- Brak determinizmu przy dużym obciążeniu

LIN

LIN – Local Interconnect Network

- Pierwsza specyfikacja – 2002
- Niski koszt
- Prędkość do 20 kb/s
- Do 8 bajtów na ramkę (rozszerzalny przez LIN-TP)
- Komplementarny do CAN, budowa sieci hierarchicznych
- Zwykle używana w aplikacjach zarządzających komfortem (drzwi, fotele, oświetlenie)
- Definicja w pliku LDF

MOST

MOST – Media Oriented Systems Transport

- Dostępna od 2001
- Kosztowana
- Do 150 Mb/s (MOST150)
- Różne warstwy fizyczne (światłowód, UTP, kabel koncentryczny)
- Transmisja sygnałów multimedialnych między ECU (audio, video, głos, dane)

CAN-FD

CAN-FD (Flexible Datarate)

- Zaproponowana przez Bosch w 2010
- Długość: do 64 bajtów
- Prędkość: do 6 Mb/s

18. CAN – rodzaje i ogólna struktura ramek.

CAN**Ramka CAN**

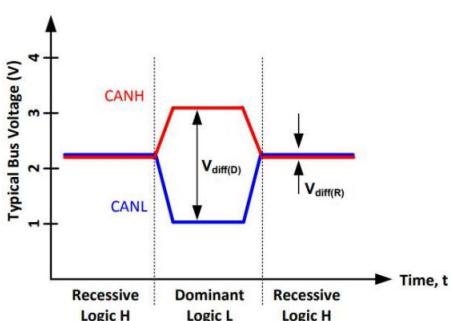
- Identyfikator komunikatu lub identyfikator arbitrażu – definicja priorytetu komunikatu.
 - niższa wartość – wyższy priorytet
- węzły transmisujące komunikaty prowadzą jednocześnie nastąpienie magistrali w tym samym czasie w celu rozstrzygnięcia priorytetu komunikatu
 - możliwość prowadzenia nieprzerwanej transmisji w przypadku wystąpienia kolizji
 - Ethernet – przerwanie transmisji w przypadku kolizji
- ID rozróżnia różne typy komunikatów, które ECU może wysyłać bądź reagować (mechanizm filtracji po stronie kontrolera)
 - np. moduł główny może oczekiwany na stan klawiatury w celu włączenia lub wyłączenia danego systemu, ale nie jest zainteresowany statusem modułu Air Bag.
- CAN 2.0A – identyfikatory 11 bitowe
- CAN 2.0B – identyfikatory 11 lub 29 bitowe.

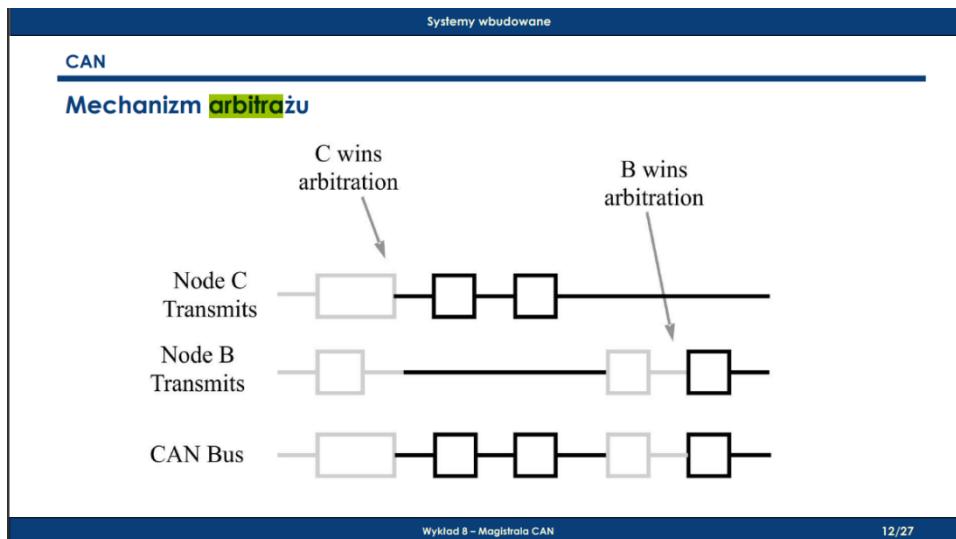
**CAN****Ramka CAN**

- **Pole kontrolne (Control field)**
 - 2 zarezerwowane bity reserved bits (do wykorzystania w przyszłych wersjach jak Extended CAN) oraz DLC (Data Length Code) – definicja liczby bajtów przesyłanych w ramach pola danych (od 0 do 8 bajtów).
- **Pole danych (Data field)** – standard CAN – do 8 bajtów danych (CAN FD do 64 bajtów danych).
 - Właściwa informacja przekazywana do pozostałych węzłów w sieci.

**19. CAN – warstwa fizyczna, stany recesywne i dominujące, mechanizm arbitrażu.****CAN****Warstwa fizyczna**

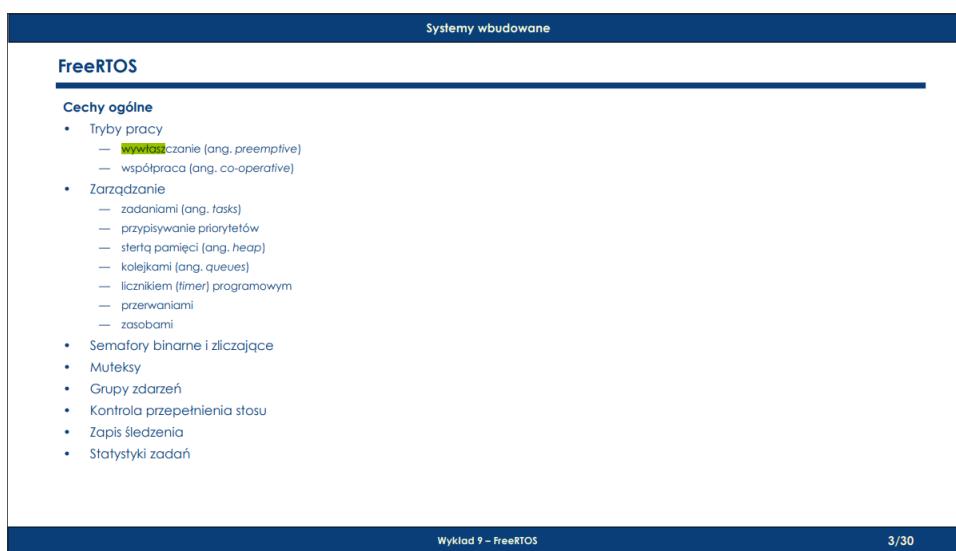
Jeśli wartość V_{diff} jest mniejsza niż 0,5V magistrala znajduje się w stanie recesywnym (logiczna „1”). Jeśli V_{diff} jest większe niż 0,9V magistrala znajduje się w stanie dominującym (logiczne „0”). Wartość między ww. wartościami oznacza stan niezdefiniowany. Zasada różnicowego przesyłu sygnałów.





- Systemy wbudowane
- ## CAN
- ### Mechanizm arbitrażu
- Każdy węzeł stale monitoruje własną transmisję
 - Bit recesywny węzła B jest zastępowany przez bit dominujący węzła C o wyższym priorytecie
 - B wykrywa, że stan magistrali nie odpowiada bitowi, który został przesłany.
 - Węzeł B zatrzymuje transmisję, podczas gdy węzeł C kontynuuje przesyłanie komunikatu.
 - Kolejna próba przesłania wiadomości jest podejmowana przez węzeł B po zwolnieniu magistrali przez węzeł C.
 - Część fizycznej warstwy sygnalizacyjnej ISO 11898
 - składowa kontrolera CAN
 - całkowicie przezroczysta dla użytkownika (programisty)
- Wykład 8 – Magistrala CAN
- 14/27

20. FreeRTOS – tryby pracy, główne komponenty.



21. FreeRTOS – stany zadań.

Uruchomiony, gotowości, zawieszony, zablokowany

Systemy wbudowane

FreeRTOS

Możliwe stany zadań

- Uruchomiony (**Running**)
 - w trakcie działania
 - brak konieczności oczekiwania na zasoby
 - brak oczekujących zadań o wyższym priorytecie
- Gotowość (**Ready**)
 - brak konieczności oczekiwania na zasoby
 - może zostać uruchomione, gdy np. zadanie o wyższym priorytecie oczekuje na zasoby
- Zawieszony (**Suspended**)
 - nie jest dostępne dla Planisty i nie może być uruchomione z powodu wykonania funkcji vTaskSuspend()
 - może zostać wznowione za pomocą funkcji vTaskResume() lub xTaskResumeFromISR()
- Zablokowany (**Blocked**)
 - oczekiwanie na zasoby
 - zdarzenie powiązane z timerem
 - synchronizacja – odbiór danych z kolejki
 - inne źródła: kolejki, semafory, mutexy, grupy zdarzeń, bezpośrednie powiadomienia

Wykład 9 – FreeRTOS 21/30

22. FreeRTOS – stos i sterta, modele stert.

Sterty: heap_1, heap_2, heap_3, heap_4, heap_5

Systemy wbudowane

FreeRTOS – sterowanie zadaniami

Rodzaje stert FreeRTOS – zarządzanie pamięcią

- heap_1
 - Najprostsza, brak możliwości zwolnienia pamięci.
- heap_2
 - Zwolnienie pamięci, brak możliwości łączenia sąsiednich wolnych bloków.
- heap_3
 - Wykorzystanie standardowych funkcji `malloc()` i `free()` z zawieszeniem pracy planisty w celu zapewnienia bezpieczeństwa wątków.
- heap_4
 - łączenie sąsiednich wolnych bloków, w celu uniknięcia fragmentacji. Zawiera opcję bezwzględnego adresowania.
- heap_5
 - Zgodna z heap_4, z możliwością rozciągnięcia sterty na niesącujące ze sobą obszary pamięci.

Wykład 9 – FreeRTOS 30/30

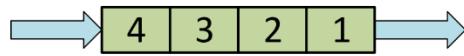
23. FreeRTOS i ARM (STM32) – ogólne sposoby programowania (FreeRTOS API, CMSIS).

??????

24. FreeRTOS – kolejki, semafory i mutexy.

FreeRTOS**Kolejka (ang. queue)**

- komunikacja między zadaniami w środowisku wielozadaniowym
- przesyłanie komunikatów pomiędzy zadaniami oraz między zadaniami i przerwaniem
- realizacja na zasadzie bufora FIFO (ang. First In First Out)
 - dane zapisywane od strony lewej
 - odczyt od strony prawej



- FreeRTOS – możliwość zapisu także od czoła kolejki

FreeRTOS**Kolejka**

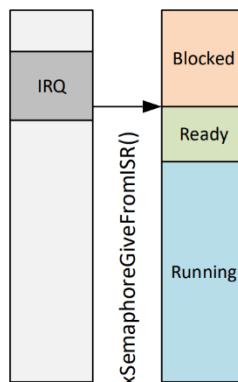
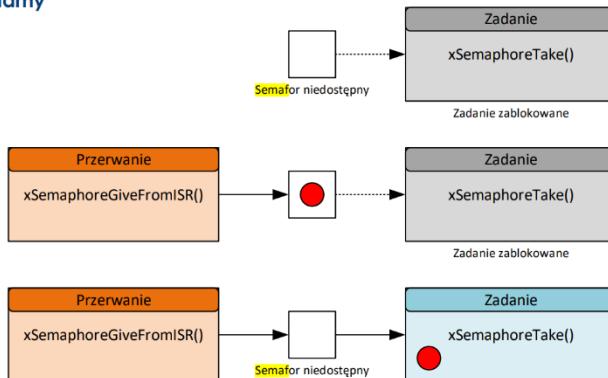
- Zapis kopii danych do kolejki (a nie referencji)
 - Możliwość ponownego wykorzystania danych oryginalnych
- Alokacja pamięci dla kolejki – zadanie jądra systemu
- Niewielkie komunikaty i dane – możliwość bezpośredniego przekazywania do kolejki
- Większe struktury danych – definiowanie kolejki wskaźników i posługiwianie się kopią wskaźników
- Pojedyncza kolejka – możliwość odczytu różnych typów danych z różnych źródeł
 - możliwość zapisu danych do kolejki przez różne zadania

FreeRTOS – kolejki**Zmiana stanu zadania**

- Automatyczne blokowanie zadania – przeniesienie do stanu **Blocked**:
 - odczyt z pustej kolejki
 - próba zapisu do pełnej kolejki
- Przejście do stanu **Ready**:
 - odczyt – umieszczenie nowych danych w pustej kolejce
 - zapis – zwolnienie miejsca w pełnej kolejce
- Odblokowanie wielu zadań w kolejności priorytetów
- Przy równych priorytetach odblokowanie najdłużej oczekującego zadania
- Automatyczne odblokowanie w sytuacji wyczerpania czasu blokady

FreeRTOS – semafory**Semafor binarny**

- sterowanie wykonywaniem działań
- synchronizacja zadań lub zadań i przerwań

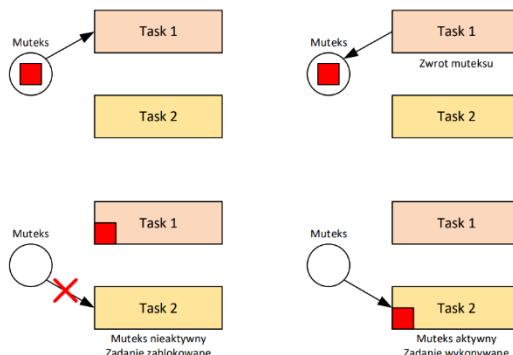
**FreeRTOS – semafory****Semafor binarny****FreeRTOS – semafory****Semafor zliczający**

- W przypadku przerwań nadchodzących z większą niż możliwa do obsługi prędkością → utrata zdarzeń
- Semafor zliczający** → kolejka o długości większej niż jeden
- Oddanie **semafora** liczącego → kolejny element w kolejce
- Wykorzystanie **semaforów zliczających**:
 - Liczenie zdarzeń: wartość licznika to różnica między liczbą zdarzeń, które miały miejsce, a liczbą zdarzeń przetworzonych
 - Zarządzanie zasobami: wartość licznika wskazuje liczbę dostępnych zasobów; zadanie musi najpierw uzyskać **semafor**, zanim uzyska kontrolę nad zasobem; zadanie zwraca **semafor** po zakończeniu pracy z zasobem

FreeRTOS – muteksy**Muteks – Mutual exclusion**

- Specjalny typ binarnego **semajora**, służący do kontrolowania dostępu do zasobu współdzielonego między co najmniej dwoma zadaniami
- Muteks → rodzaj tokena powiązanego z udostępnianym zasobem
- Dostęp do zasobu wymaga pobrania tokenu
- Po zakończeniu pracy z zasobem → konieczność zwrócenia tokena
- Wykorzystanie funkcji: xSemaphoreTake(), xSemaphoreGive()
- Możliwość oczekiwania przez zadanie o wyższym priorytecie na zadanie o niższym priorytecie, posiadające muteks

 - Opóźnienie zadania o wyższym priorytecie zadanie o niższym priorytecie
 - „odwrócenie priorytetu” (ang. *priority inversion*)

FreeRTOS – muteksy**Muteks****25. FreeRTOS – zakleszczenie zadań.****FreeRTOS – muteksy****Zakleszczenie (ang. deadlock)**

- Dwa zadania oczekują na zasób, utrzymywany przez drugie z nich, np.
 - Zadanie A wykonuje się i pomyślnie pobiera muteks X.
 - Zadanie A jest wywolane przez zadanie B.
 - Zadanie B pomyślnie pobiera muteks Y przed próbą pobrania muteksu X - ale muteks X jest zatrzymany przez zadanie A, więc nie jest dostępny dla zadania B.
 - Zadanie B wchodzi w stan Zablokowany, aby czekać na zwolnienie muteksu X
 - Zadanie A jest kontynuowane. Próbuje przejąć muteks Y - ale muteks Y jest trzymany przez Zadanie B, więc nie jest dostępny dla Zadania A.
 - Zadanie A wchodzi w stan Zablokowany, aby czekać na zwolnienie muteksu Y.

26. FreeRTOS – funkcje typu „hook”.

FreeRTOS**Funkcje typu Hook**

- wspierane przez jądro systemu FreeRTOS
- pomoc w obsłudze błędów
- Rodzaje:
 - Idle **Hook**
 - Tick **Hook**
 - Malloc Failed **Hook**
 - Stack Overflow **Hook**
- STM32CubeMX → freertos.c.

FreeRTOS**Idle task i „idle task hook”**

- Tworzona automatycznie przez planistę (osKernelStart())
- Najniższy priorytet
- uruchamiana w przypadku braku innych zadań (w stanie **Ready**)
- możliwość współdzielenia tego samego priorytetu z innymi zadaniami,
- Funkcja **Hook**
- `void vApplicationIdleHook (void); // [weak] version in freertos.c file,`
- nie powinna być wstrzymywana lub blokowana (np. `while(1). osDelay()` itp.)
- odpowiedzialna za czyszczenie zasobów po usunięciu innych zadań
- wykonywana w każdej iteracji IdleTask

27. Ogólna architektura platformy Zynq.

28. Możliwe sposoby współdziałania części PS i PL na platformie Zynq.

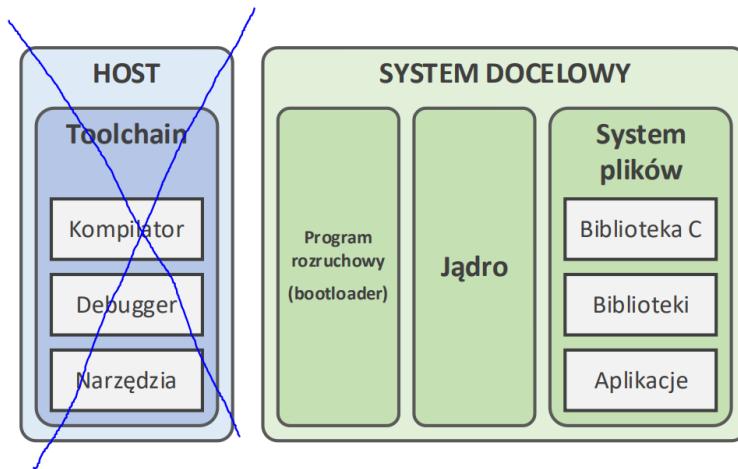
29. Wyróżniki syntezu na poziomie systemu (narzędzia/języki, przebieg procesu projektowania).

30. Zalety i ograniczenia różnych modeli architektury systemów z wykorzystaniem układów reprogramowalnych.

31. Rekonfigurowalność dynamiczna.

32. Kryteria wyboru platform sprzętowych przy projektowaniu systemów wbudowanych.

33. Podstawowe składniki systemu Linux (embedded)

Budowa Embedded Linux**34. Toolchain – zawartość i funkcje.**

Systemy wbudowane

Embedded Linux

Narzędzia i oprogramowanie do rozwoju oprogramowania na platformie docelowej

The diagram shows the components of a Toolchain within the **HOST** environment:

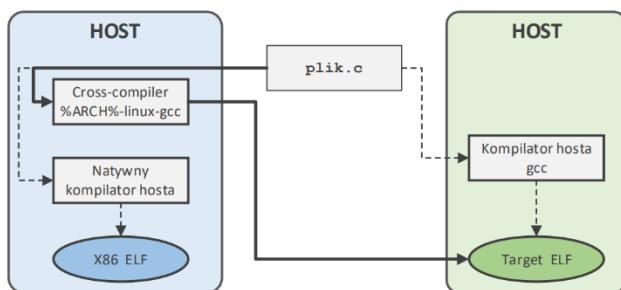
- Toolchain:** Contains:
 - Komplikator
 - Biblioteka C
 - Debugger
- Separate components:
 - Binutils
 - Nagłówki jądra
 - Pozostałe

Wykład 14 – Linux i IoT

6/46

35. Kompilacja i debugowanie skrośne.

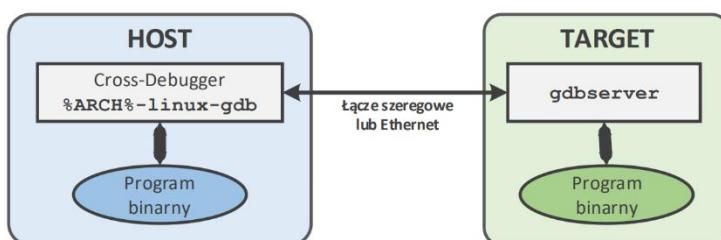
Embedded Linux Toolchain

Kompilacja skrośna

Embedded Linux Toolchain

Debugger i powiązane narzędzia

- Praktyczne rozwiązywanie – debugowanie skrośne z poziomu hosta

**36. JTAG – funkcje, przeznaczenie, możliwości i zasoby.**

Systemy wbudowane

Embedded Linux Toolchain

JTAG (ang. Joint Test Action Group)

- standard IEEE 1149.1
- protokół do testowania połączeń na płytach drukowanych, do uruchamiania i programowania układów programowalnych i systemów mikroprocesorowych
- wewnętrzna warstwa sprzętową interfejsu
- kontroler TAP oparty na maszynie stanów dedykowanej debugowaniu i niezależnej od podstawowych zadań mikrokontrolera,
- możliwość programowania układu w gotowym urządzeniu, bez konieczności odłączania (ang. In-System Programming, ISP)

37. Programy rozruchowe i ich funkcje.

Systemy wbudowane

Embedded Linux Kernel

Program rozruchowy (bootloader)

- Ładowanie i uruchamiania jądra Linuxa po załączeniu zasilania

```

graph TD
    Bootloader[Bootloader] --> Image[Obraz jądra]
    Image --> Kernel[Jądro systemu]
  
```

Wykład 14 – Linux i IoT 18/46

38. Linux embedded jako system czasu rzeczywistego – możliwości i ograniczenia.

Możliwości:

- mniejsze ograniczenia
- wielu dostawców soft/development/support
- prostota tworzenia rozwiązań/frameworki

Ograniczenia:

- NOT RTOS
- wolniejszy (nie tak blisko „bare metal”)
- relatywnie duży pobór pamięci

39. Protokół MQTT – cechy i zastosowanie.

Systemy wbudowane

MQTT (materiały prof. P. Dziurzańskiego, WI ZUT)

MQTT

- hist. MQ Telemetry Transport
- Protokół przesyłania wiadomości dla Internetu rzeczy (IoT)
- Powstał w 1999 r.; ustandaryzowany przez OASIS w 2014 r.; najnowsza wersja 5.0 w 2019 r.
- Zaprojektowany do lekkiego transportu wiadomości typu publikuj/subskrybuj
- Przydatny do podłączania zdalnych urządzeń z niewielką pamięcią i minimalną przepustowością sieci
- MQTT** jest stosowany w wielu różnych gałęziach przemysłu

Wykład 14 – Linux i IoT 32/46

MQTT**Model publish & subscribe (pub/sub)**

- Alternatywa dla tradycyjnej architektury klient-serwer
- Oddzielenie klienta, wysyłającego wiadomość (wydawcę) od klienta lub klientów odbierających wiadomości (subskrybentów)
- brak bezpośredniego kontaktu między wydawcą i subskrybentami
- Połączenie między wydawcami i subskrybentami obsługiwane jest przez komponent pośredni – broker
 - filtrowanie wszystkich przychodzących wiadomości i ich poprawna dystrybucja do subskrybentów

40. Funkcja brokerka MQTT**MQTT****Broker MQTT**

- Centralny węzeł, przez który musi przejść każda wiadomość
- Odpowiedzialny za odbieranie wszystkich wiadomości, ich filtrowanie, określanie subskrybenta każdej wiadomości i wysyłanie wiadomości do tych subskrybentów
- Przechowuje również dane sesji wszystkich klientów, którzy mają trwałe sesje (ang. *persistent*, przeciwieństwo *clean*), w tym subskrypcje i nieodebrane wiadomości
- Przeprowadzanie uwierzytelniania i autoryzacji klientów
- Wysoko skalowalny, możliwy do integracji z back-endem, łatwy do monitorowania i odporny na awarie
- W niektórych implementacjach obsługa milionów jednocześnie połączonych klientów MQTT