
NobleProg

# Cortex-M, STM32L4, Nucleo-64 i KA-Nucleo-Multisensor

Grzegorz Mazur

NobleProg

# Plan prezentacji

- ARM Cortex-M
- Mikrokontrolery serii STM32L4
- Zestaw do ćwiczeń: płytki STM32L476 Nucleo64 i KA-Nucleo-Multisensor

# Rodziny Cortex

- Cortex-M – Microcontroller
  - Do zastosowania w mikrokontrolerach, ograniczona ochrona i mechanizmy systemowe, niewielka wydajność, mały pobór mocy
- Cortex-R – Real-time
  - Podobne do M, wyższa wydajność – zastępują procesory sygnałowe
- Cortex-A – Application
  - Kompletne mechanizmy systemowe, do zastosowania w komputerach ogólnego przeznaczenia (smartfony, tablety)

# Co to jest Cortex-M?

- Nowsza generacja procesorów/rdzeni ARM, architektura ARMv6-M, v7-M lub v8-M
- Zestaw instrukcji Thumb 2
- Instrukcje 16- i 32- bitowe – lepsza gęstość kodu niż w klasycznym ARM
- Nowatorski system wyjątków o wysokiej wydajności, eliminujący zbędne czynności procesora i łatwy dla programisty

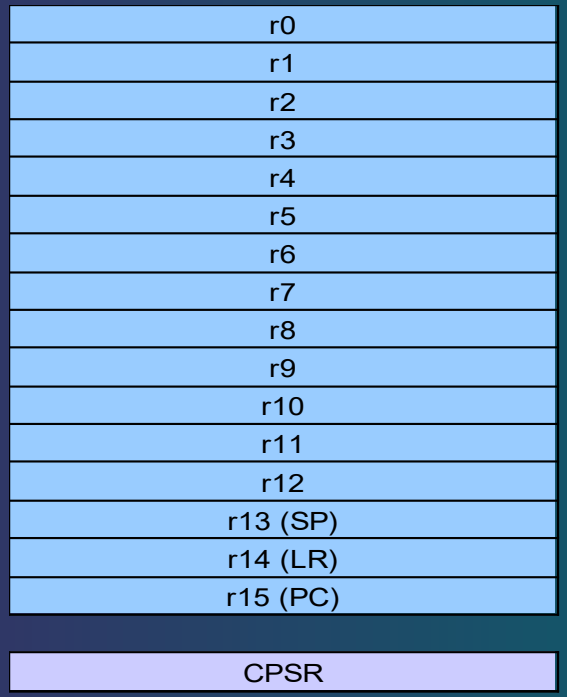
# Wersje Cortex-M

| Symbol rdzenia | Architektura | DMIPS /MHz | CoreMark/ MHz | Charakterystyka |
|---|---|---|---|---|
| M0 | ARMv6-M | 0,89 | 2,33 | Najprostszy rdzeń, zastępujący mikrokontrolery 8- i 16-bitowe, o bardzo uproszczonych mechanizmach ochrony; zastąpiony przez M0+ |
| M0+ | ARMv6-M | 0,95 | 2,46 | Nowsza odmiana podstawowego rdzenia o uproszczonej konstrukcji i obniżonym poborze mocy; umożliwia łatwą aktualizację oprogramowania przez urządzenie |
| M3 | ARMv7-M | 1,25 | 3,34 | Pierwszy, najstarszy rdzeń Cortex-M, przeznaczony do typowych zastosowań 32-bitowych; zastąpiony przez M4 |
| M4 | ARMv7-M | 1,25 | 3,42 | Najpopularniejszy rdzeń do typowych zastosowań średniej wydajności; zwykle wyposażony w jednostkę zmiennopozycyjną operującą na 32-bitowych liczbach zmiennopozycyjnych IEEE754 binary32. |
| M7 | ARMv7-M | 2,14 | 5,01 | Rdzeń superskalarny do zastosowań o wysokiej wydajności obliczeniowej, wyposażony w kieszenie kodu i danych oraz jednostkę zmiennopozycyjną (opcjonalnie z obsługą również formatu 64-bitowego) |
| M23 | ARMv8-M | 0,98 | 2,64 | Rdzeń nieco podobny do M0+, z rozbudowanymi mechanizmami ochrony. |
| M33 | ARMv8-M | 1,5 | 4,02 | Podobny do M4, z rozbudowanymi mechanizmami ochrony. |
| M35 | ARMv8-M | 1,5 | 4,02 | Wysokowydajny rdzeń z rozbudowanymi mechanizmami ochrony. |
| M55 | ARMv8.1-M | 1,6 | 4,2 | Wysokowydajny z jednostką wektorową Helium |
| M85 | ARMv8.1-M | 3,13 | 6,28 | Wysokowydajny z jednostką wektorową Helium i DSP |

# Z czego składa się Cortex-M?

- Procesor „post-RISC” o krótkim potoku (2..4 stopni)
- Timer systemowy SysTick do odmierzania stałych odcinków czasu
- Sterownik przerwań NVIC
  - Wielopoziomowy system przerwań z wywłaszczaniem
- Moduł uruchamiania i debugowania

# Procesor ARM Cortex-M




- 16 rejestrów uniwersalnych, w tym:
  - R13 – SP – wskaźnik stosu
  - R14 – LR – rejestr śladu
  - R15 – PC – licznik instrukcji
- Rejestry R0..7 dostępne dla wszystkich instrukcji, R8..R15 – tylko dla niektórych
- Rejestr stanu CPSR, w tym znaczniki NZCV
- Zestaw instrukcji Thumb, Thumb2
  - Instrukcje 16- i 32-bitowe

# Procesor ARM Cortex-M

- Architektura Load-Store
  - Operacje arytmetyczne i logiczne wyłącznie 32-bitowe, wykonywane tylko na rejestrach
  - Instrukcje ładowania i składowania danych 8-, 16- i 32-bitowych
  - Wyrównanie naturalne *
- Instrukcje 3- i 2-argumentowe
- Model operacji warunkowych ze znacznikami

# Procesor ARM Cortex-M

- Dwa poziomy zaufania – wątku i obsługi wyjątków
  - Na poziomie wątku można używać alternatywnego wskaźnika stosu
- 4..16 + 1 poziomów priorytetowych procesora
  - Wielopoziomowy system wyjątków z wywłaszczaniem
- Podstawowe mechanizmy ochrony
  - Błąd przy nielegalnych operacjach
  - Proste mechanizmy ochrony pamięci

# Adresowanie

- Jedna, 32-bitowa, liniowa przestrzeń adresowa o określonym podziale na obszary (pamięć, wejście-wyjście, zasoby procesora)
  - Wszystkie zasoby mikrokontrolera, w tym rejestry systemowe procesora, są odwzorowane w lokacje przestrzeni adresowej

# Cortex-M – programowanie

- Procesory przystosowane do programowania całkowicie w języku C
  - Łącznie z modułem startowym, który jednak w niektórych środowiskach jest pisany w asemblerze
- Procedury obsługi wyjątków nie różnią się od innych procedur
  - Zbędne rozszerzenia języka typu „interrupt”

# Cortex-M – wyjątki

- Kilka (do 14) wyjątków rdzenia, w tym przerwanie SysTick
- Przerwania peryferiali
  - Do 32 w Cortex-M0, do 240/480 w M3, M4 i silniejszych rdzeniach
- Konfigurowane priorytety wywłaszczania
  - W M0 – 4, w M4 - 16 poziomów wyjątków + poziom wątku (programu głównego)

# Cortex-M – optymalizacja obsługi wyjątków

- Późne przybycie (*late arrival*) – wyjątek o wyższym priorytecie sygnalizowany tuż po rozpoczęciu sprzętowej obsługi wyjątku o niższym priorytecie
- Łańcuchowanie (*tail chaining*) – opóźnienie obsługi wyjątku zgłoszonego w czasie obsługi wyjątku o takim samym lub wyższym priorytecie

# Późne przybycie wyjątku

- Procesor rozpoczyna składowanie kontekstu po wykryciu wyjątku o priorytecie wyższym od priorytetu procesora
- Arbitraż wyjątków następuje po składowaniu kontekstu
  - Do obsługi jest wybierany wyjątek o aktualnie najwyższym priorytecie
  - Może to być inny wyjątek, niż ten, który spowodował składowanie kontekstu

# Łańcuchowanie wyjątków

- W tradycyjnych architekturach przy zakończeniu obsługi wyjątku przy oczekującym wyjątku o takim samym lub niższym priorytecie, przejście do obsługi kolejnego wyjątku następuje po odtworzeniu kontekstu, wykonaniu jednej instrukcji wątku lub wyjątku o niższym priorytecie (*) i składowaniu kontekstu
- Cortex-M w takim przypadku nie odtwarza i nie składuje kontekstu tylko przeskakuje do obsługi kolejnego wyjątku, bez wykonania instrukcji wątku (kodu o niższym priorytecie)

# Usypianie procesora

- Instrukcja WFI usypia procesor do czasu wykrycia przerwania (również wtedy, gdy priorytet procesora uniemożliwia jego programową obsługę)
- Ustawienie bitu SleepOnExit powoduje uśpienie procesora przy obniżeniu priorytetu do poziomu wątku (przy wyjściu z obsługi wyjątku), bez konieczności użycia WFI

# Usypianie procesora

- WFI bezpośrednio po ustawieniu SleepOnExit w trybie wątku powoduje uśpienie; po obsłudze przerwania procesor pozostanie w uśpieniu
- Tryb (głębokość) uśpienia zależy od ustawień w rejestrach konfiguracyjnych – domyślnie płytkie uśpienie procesora z normalną pracą pozostałych bloków µC

# Seria STM32L4

- Mikrokontrolery z rdzeniem Cortex-M4
- Zaprojektowane jako energooszczędne (częstotliwości pracy niższe niż w F4)
- 32 KiB..1 MiB Flash, 32..320 KiB RAM
- Częstotliwość pracy do 80 MHz
- Moc obliczeniowa > 20× większa od AVR ATmega
- Większość linii portów akceptuje sygnały 5 V

# Seria STM32L4+

- Mikrokontrolery z rdzeniem Cortex-M4
- Zaprojektowane jako energooszczędne (częstotliwości pracy niższe niż w F4)
- 1..2 MiB Flash, 640 KiB RAM
- Częstotliwość pracy do 120 MHz
- Moc obliczeniowa > 30× większa od AVR ATmega
- Większość linii portów akceptuje sygnały 5 V

# STM32L476

- 80 MHz
- 128 KiB RAM (2 bloki: 96 + 32 KiB, rozdzielone w przestrzeni adresowej)
- 256..1024 KiB Flash
- Interfejsy szeregowe: UART, I2C, SPI, QSPI, SAI i inne
- Timery - łącznie > 20 kanałów
- ADC, DAC, komparatory

# STM32L496

- 80 MHz
- 320 KiB RAM (2 bloki: 256 + 64 KiB, tworzące ciągły obszar)
- 256..1024 KiB Flash
- Interfejsy szeregowe: UART, I2C, SPI, QSPI, SAI i inne
- Timery - łącznie > 20 kanałów
- ADC, DAC, komparatory

# STM32L4xx - dokumentacja

- Dostępny zestaw dokumentów i not aplikacyjnych www.st.com
- Najważniejsze dokumenty:
  - Reference Manual – opis logiczny układu, w tym peryferiali
  - Data Sheet – parametry elektryczne, obudowy, FUNKCJE WYPROWADZEŃ i sposób ich przypisania
  - Errata – uwaga na QSPI i ADC!!!

# STM32L476 – mapa pamięci

- 0x1FFF8000 – System ROM – OTP Flash z parametrami i procedurami producenta, zawiera bootloader
- 0x08000000..0x080fffff – Flash dla programu użytkownika
- 0x20000000..0x20017fff – SRAM1
- 0x10000000..0x10007fff – SRAM2

# STM32L496 – mapa pamięci

- 0x1FFF8000 – System ROM – OTP Flash z parametrami i procedurami producenta, zawiera bootloader
- 0x08000000..0x080fffff – Flash dla programu użytkownika
- 0x20000000..0x2003ffff – SRAM1
- 0x10000000..0x1000ffff oraz 0x20040000..0x2004ffff – SRAM2

# Start mikrokontrolera

- Początkowa wartość SP i PC pobierana z dwóch pierwszych słów pamięci – adres 0 i 4
- W zależności od konfiguracji startowej pod adresem 0 jest odwzorowana jedna z pamięci: Flash, SRAM1 lub system ROM (typowo Flash)
- Mikrokontrolery STM32L4 startują z wewnętrznym generatorem zegara RC – MSI, o początkowej częstotliwości 4 MHz

# Moduły peryferyjne

- Peryferia znacznie bardziej funkcjonalne i bardziej złożone niż w μC 8-bitowych
- Każdy moduł jest reprezentowany w języku C przez strukturę, której pola odpowiadają poszczególnym rejestrom sterującym

# STM32L4 - peryferia

- Transmisja szeregowa - UART, I2C, SPI, I2S
- QSPI – interfejs pamięci szeregowej lub szeregowo-równoległej SPI, odwzorowujący ją w przestrzeni adresowej
- Timery wielofunkcyjne TIM
- ADC, DAC, wzmacniacz operacyjny, komparator
- Watchdog, monitor zasilania, monitor zegara itp.

# STM32 – porty GPIO

- Porty mają logiczną szerokość 16 bitów
- Oznaczane kolejnymi literami – GPIOA, GPIOB, GPIOC
- Tryby pracy konfigurowane indywidualnie dla każdej linii
  - Wejście cyfrowe GPIO
  - Wyjście GPIO
  - wejście/wyjście modułu peryferyjnego
  - wejście/wyjście analogowe

# Rejestry ustawiania/zerowania

- W µC z rdzeniami Cortex rejestry peryferiali są zwykle wyposażone w mechanizm sprzętowego ustawiania poszczególnych bitów przez pojedynczą operację zapisu
- W STM32 każdy port jest wyposażony w 2 rejestry ustawiania/zerowania:
  - BRR – zapis 1 powoduje zerowanie bitu
  - BSRR – górna połowa zawiera BRR, zapis 1 do dolnej połowy powoduje ustawienie bitu w stan 1

# STM32 – modyfikacja stanu linii portów GPIO

```
#define LED_PORT GPIOB
#define LEDR_BIT 5
#define LEDG_BIT 6
#define LEDR_MSK (1u << LEDR_BIT)
#define LEDG_MSK (1u << LEDG_BIT)

LED_PORT->BSRR = LEDG_MSK;    // green on
LED_PORT->BRR = LEDR_MSK;     // red off
LED_PORT->BSRR = LEDG_MSK << 16 | LEDR_MSK;   // green off, red on

// toggle red
LED_PORT->BSRR = LEDR_MSK << 16 | (~LED_PORT->ODR & LEDR_MSK);
```

# BitBand

- Cecha niektórych rdzeni Cortex-M (M3, M4)
- Możliwość ustawienia stanu pojedynczego bitu w wybranych fragmentach przestrzeni adresowej poprzez zapis danej (jedna instrukcja procesora) zamiast odczytu-modyfikacji-zapisu (min. 3 instrukcje)
- Operacja wykonywana przez interfejs szyny danych procesora
- Dwa obszary o rozmiarze po 1 MiB

# BitBand

- Obszary bazowe:
  - RAM: 0x20000000
  - Peryferia: 0x40000000
- Adresy BitBand (początek obszaru + 32 MiB): 0x22000000, 0x42000000
- LSB każdego słowa 32-bitowego w obszarze BitBand odpowiada pojedynczemu bitowi w obszarze bazowym

# Taktowanie STM32L476

- MSI – wewn RC 0.1..48 MHz (domyślnie 4 MHz), może być synchronizowany do LSE
- HSI16 – wewn. RC 16 MHz
- LSI – wewn. RC 32 kHz (niedokładny)
- HSE – generator z zewn. oscylatorem lub wejście zewn. zegara, typ. 8 MHz
- LSE – generator z zewn. oscylatorem typ. 32768 Hz

# Taktowanie STM32L496, L4Px, L4Rx

- MSI – wewn RC 0.1..48 MHz (domyślnie 4 MHz), może być synchronizowany do LSE
- HSI16 – wewn. RC 16 MHz
- HSI48 – wewn. 48 MHz, może być synchronizowany z kilku źródeł
- LSI – wewn. RC 32 kHz (niedokładny)
- HSE – generator z zewn. oscylatorem lub wejście zewn. zegara, typ. 8 MHz
- LSE – generator z zewn. oscylatorem typ. 32768 Hz

# Taktowanie STM32L4

- Wewnętrzne generatory RC MSI i HSI16 mają dokładność rzędu 1%
  - Precyzja RC wystarcza do transmisji UART, ale nie nadaje się do odmierzania czasu np. w zegarku
  - Oprogramowanie może wybrać inne źródło taktowania w celu uzyskania lepszej dokładności lub włączyć PLL w celu podwyższenia częstotliwości

# Synchronizacja MSI

- Wewnętrzny generatory RC może być synchronizowany generatorem LSE z oscylatorem zegarkowym
  - Możliwe precyzyjne odmierzanie czasu i transmisja USB bez oscylatora kwarcowego wysokiej częstotliwości

# Synchronizacja HSI48

- Generator MSI48 może być używany do taktowania USB i niektórych innych peryferiów (nie do taktowania rdzenia)
- Może on być synchronizowany źródłem zewnętrznym:
  - Generatorem LSE z oscylatorem zegarkowym
  - Znacznikami czasowymi z interfejsu USB w trybie urządzenia – możliwa praca bez oscylatora kwarcowego
  - Innym sygnałem zewnętrznym

# Taktowanie STM32L4

- HCLK – zegar rdzenia
- AHBCLK, APB1CLK, APB2CLK – zegary szyn
  - Mogą być uzyskane przez dzielenie częstotliwości HCLK (w L4 nie jest to potrzebne – wszystkie zegary mogą mieć częstotliwość taką, jak HCLK)

# Start oprogramowania

- Wartość PC zapisana pod adresem 4 wskazuje procedurę startową Reset_Handler
- Moduł startowy kolejno:
  - Wywołuje procedurę użytkownika SystemInit()
  - Przygotowuje środowisko pracy dla programu w języku C
    - Inicjowanie danych statycznych zainicjowanych
    - Inicjowanie danych statycznych niezainicjowanych na wartość 0
  - Wywołuje funkcję main()

# Środowiska programowania

- STM32CubeIDE
- Keil MDK-ARM
  - darmowe w zastosowaniach niekomercyjnych do 32 KiB
- IAR EWB
- Inne bazujące na Eclipse:
  - DIY – Eclipse + GNU ARM GCC
  - Atollic TrueSTUDIO – obecnie w CubeIDE
  - Dawniejsze: SW4STM32 (AC6), CooCox

# Wybór środowiska

- Wg. uznania użytkownika
- MDK-ARM:
  - Łatwa instalacja, proste w konfiguracji i użyciu
  - częste aktualizacje
  - Wersja darmowa – tylko C, od 2020 tylko do zastosowań niekomercyjnych
    - dostępne wersje darmowe komercyjne dla µC z rdzeniami M0 i M0+
- STM32CubeIDE:
  - Eclipse, GNU GCC ARM i otoczenie
  - Wbudowany generator aplikacji
  - C++ w wersji darmowej

# Jak programować?

- Gotowe biblioteki czy operacje na rejestrach?
- Biblioteki
  - Łatwe w użyciu w prostszych przypadkach
  - Nadmiarowy, nieoptymalny kod; błędy
- Operacje na rejestrach peryferiali
  - Krótsze w zapisie i znacznie szybsze w działaniu
  - Wymagają zapoznania się z dokumentacją mikrokontrolera
    - … co i tak kiedyś musi nastąpić przy projektowaniu rzeczywistych urządzeń i ich oprogramowania

# Nucleo64

- Seria płytek uruchomieniowych z różnymi µC rodziny STM32 w obudowach LQFP64, przystosowanych do prototypowania projektów użytkownika
- Zawiera interfejs ST-Link 2.1 (SWD, VCOM, program ładujący MSD zgodny z mbed)
- Przycisk użytkownika (PC13), LED (PA5)
- Wszystkie linie portów dostępne na złączach

# Nucleo144

- Seria płytek uruchomieniowych z różnymi µC rodziny STM32 w obudowach LQFP144, przystosowanych do prototypowania projektów użytkownika
- Zawiera interfejs ST-Link 2.1 (SWD, VCOM, program ładujący MSD zgodny z mbed)
- Przycisk użytkownika (PC13)
- 3 × LED (PB14, PB7, PC7)
- Wszystkie linie portów dostępne na złączach

# Płytka Nucleo64

- Złącza Arduino Uno
- Złącza ST Morpho
  - Dostępne (prawie) wszystkie linie portów mikrokontrolera
- UART mikrokontrolera połączony z USB-VCOM ST-Link
  - Możliwa komunikacja z PC

# Nucleo64 - L476RG

- Wybrana do ćwiczeń ze względu na elastyczność mikrokontrolera (lepszą niż w F4xx)
- Wersja µC z 1 MiB Flash
- UART2 (linie PA2 i PA3) µC połączony z VCOM ST-Link
- Domyślnie brak HSE – używamy MSI z synchronizacją LSE

# ST-Link v2-1

- Interfejs do debugowania i programowania ST-Link v2-1
  - Interfejs SWD
  - VCOM połączony z UART mikrokontrolera
    - L476: PA2 i PA3 - USART2
    - L496/L4R6; PG7, PG8 - LPUART1
  - Możliwość programowania µC przez kopiowanie pliku do urządzenia pamięci masowej

# KA_NUCLEO_MULTISENSOR

- Płytka do ćwiczeń z programowania STM32
  - przyciski
  - Wyświetlanie – mpx, charlieplexing
  - ADC – fotorezystor
  - USB
  - Czujniki środowiskowe z interfejsami SPI, TWI(I2C), OneWire

# KA_NUCLEO_MULTISENSOR