Mechatroniczne systemy wykonawcze, sensoryczne i sterujące

Mikrokontrolery ARM

dr inż. Grzegorz Góra

D1-Lab 20

ggora@agh.edu.pl

http://home.agh.edu.pl/~ggora/

Katedra Robotyki i Mechatroniki

Wydział Inżynierii Mechanicznej i Robotyki

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

PLAN WYKŁADU

Część I

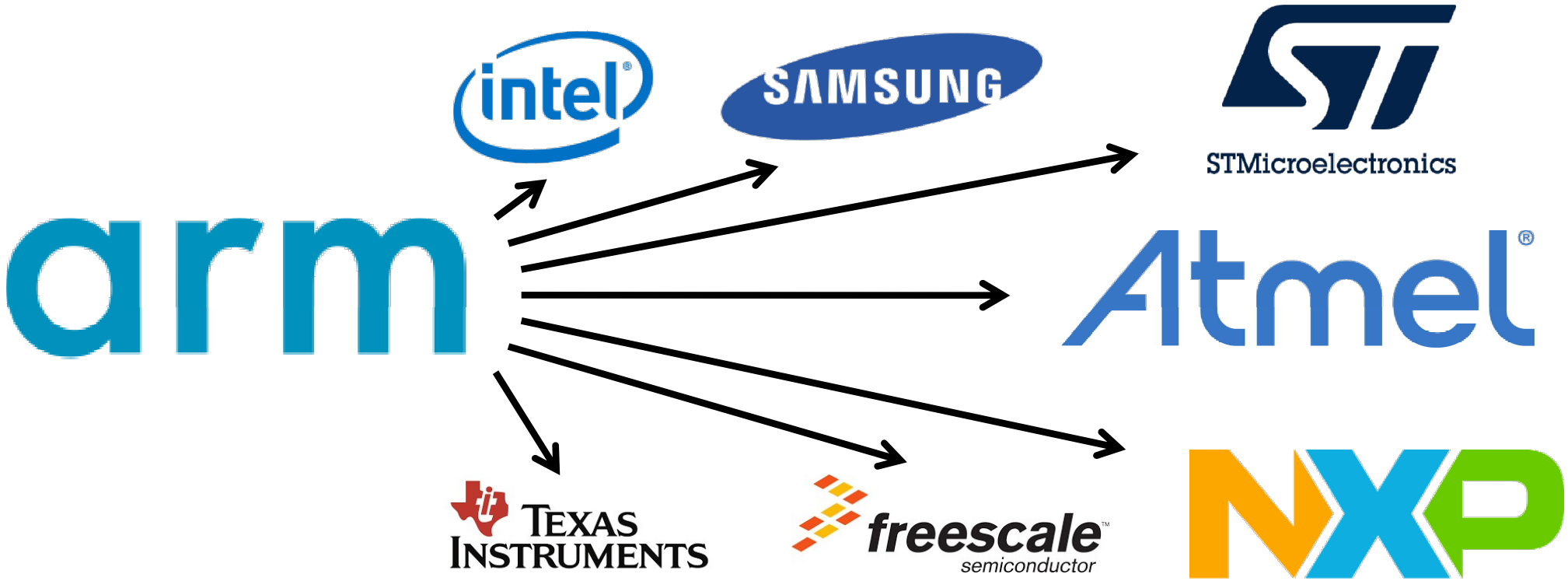
1. Mikrokontrolery z rdzeniem ARM
2. Rdzenie ARM Cortex
3. Rdzenie ARM Cortex-Mx
4. Narzędzia do programistyczne
5. Architektura mikrokontrolera na przykładzie STM32F4
6. RCC
7. GPIO
8. NVIC
9. DMA
10. SysTick
11. Timer
12. UART/USART
13. SPI
14. I2C
15. ADC
16. Pozostałe układy peryferyjne
17. Zestawy uruchomieniowe

Część II

1. Projekt w STM32 CubeIDE
2. Konfiguracja zegarów
3. GPIO – In/Out
4. UART
5. ADC
6. SysTick
7. TIM_PWM
8. TIM_ENCO (position)
9. TIM_HALL (velocity)

Mikrokontrolery ARM

ARM (ang. Advanced RISC Machines) – firma projektująca mikroprocesory. Jej zadanie kończy się na etapie zaprojektowania i przetestowania architektury układu cyfrowego (z wykorzystaniem układów reprogramowalnych FPGA/CPLD). Firma ARM zajmuje się jedynie projektowaniem i sprzedażą licencji innym firmom, tj.: STMicroelectronics (STM32), Atmel (AT91SAM), NXP (LPC), które zajmują się produkcją układów scalonych.



Rdzenie ARM Cortex

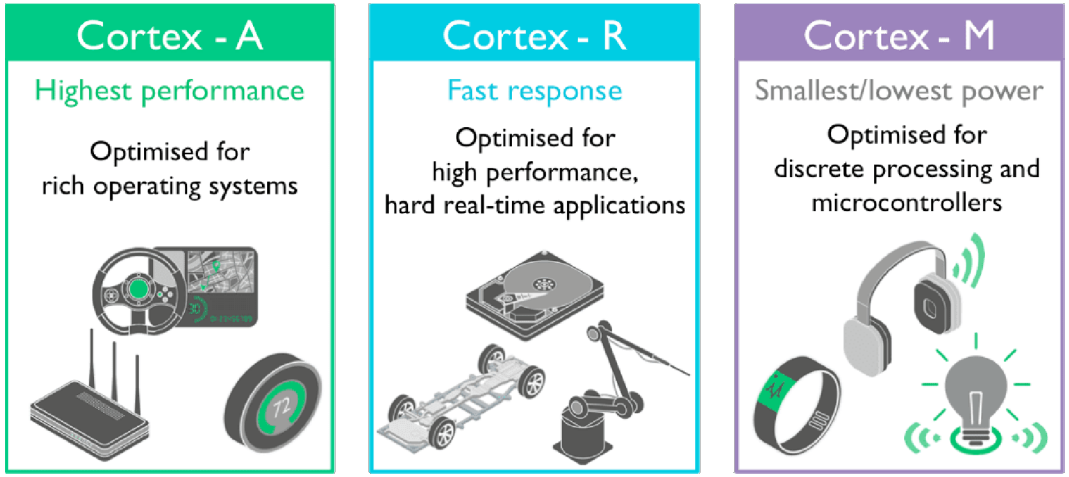
Rdzenie Cortex występują w 4 wariantach:

Cortex-A (*Application*) – rdzenie najbardziej zaawansowane, o najwyższej wydajności, przeznaczone do zastosowań, w których zwykle wykorzystuje się systemy operacyjne (np. Linux, Android, FreeRTOS), architektury 32- lub 64-bitowe;

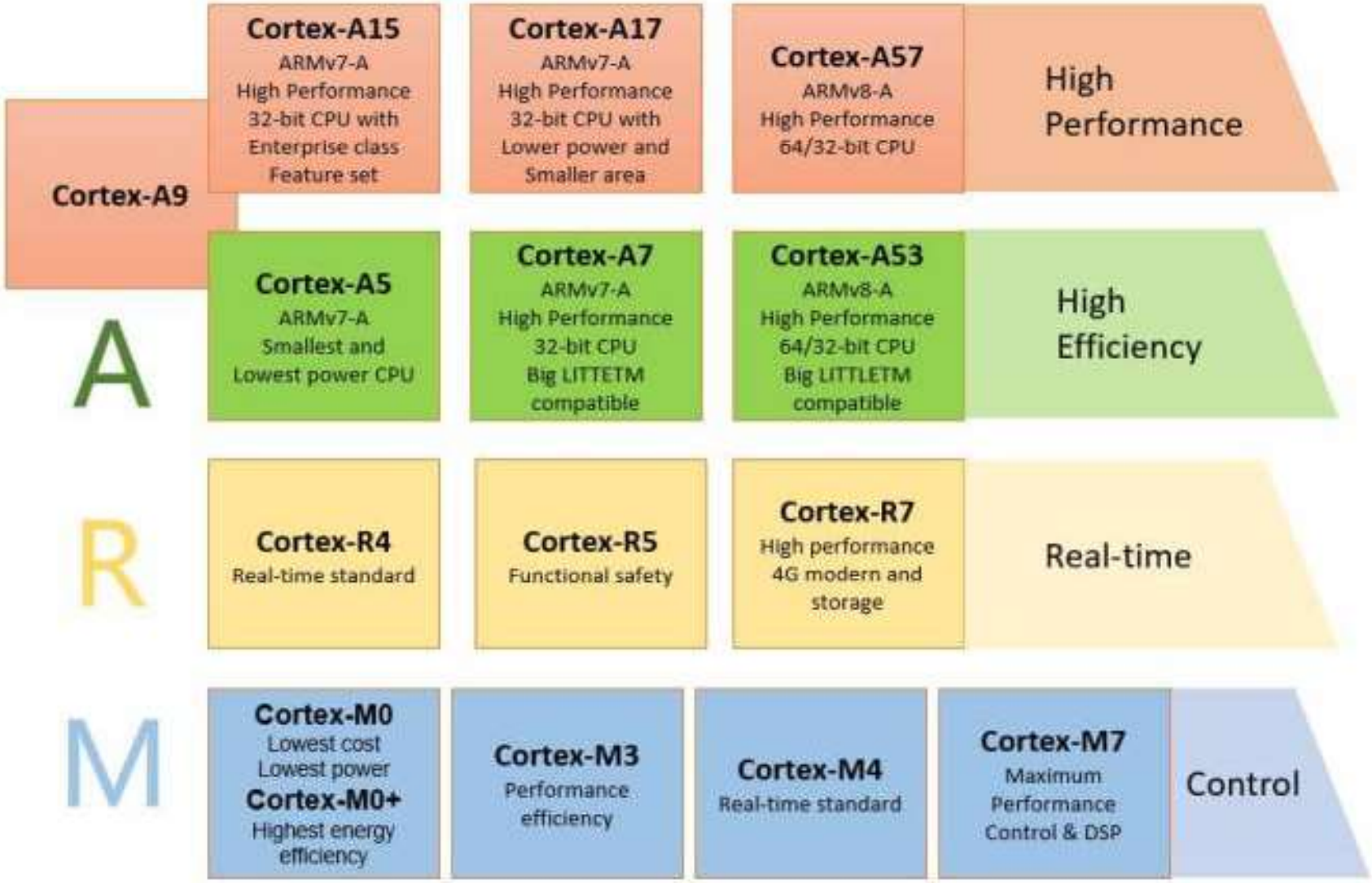
Cortex-M (*Microcontroller*) – rdzenie przeznaczone do budowy mikrokontrolerów ogólnego przeznaczenia, architektura 32-bitowa, o korzystnym stosunku jakości do ceny;

Cortex-R (*Real Time*) – rdzenie dedykowane dla systemów czasu rzeczywistego, w których krytycznym aspektem jest czas realizacji zadań, przeznaczone m.in. do zastosowań w przemyśle motoryzacyjnym;

Cortex-SC (*Secure Core*) – rdzenie bazujące na Cortex-M dedykowane dla systemów wymagających podwyższonego bezpieczeństwa (np. karty chipowe, karty SIM, dokumenty biomedyczne).



Rdzenie ARM Cortex



Rdzenie ARM Cortex-M

Rdzenie ARM Cortex-M występują w kilku wariantach:

Cortex-M0/M0+ – najmniej złożony rdzeń, tania 32-bitowa alternatywa dla mikrokontrolerów 8-bitowych, do zastosowań w mało wymagających aplikacjach; M0+ jest nowszą wersją M0 o zmniejszonym poborze energii i większej wydajności;

Cortex-M1 – rdzeń oparty na M0, zaprojektowany do zastosowania w układach FPGA;

Cortex-M3 – rdzeń zaprojektowany do wykorzystania w systemach wbudowanych (ang. Embedded);

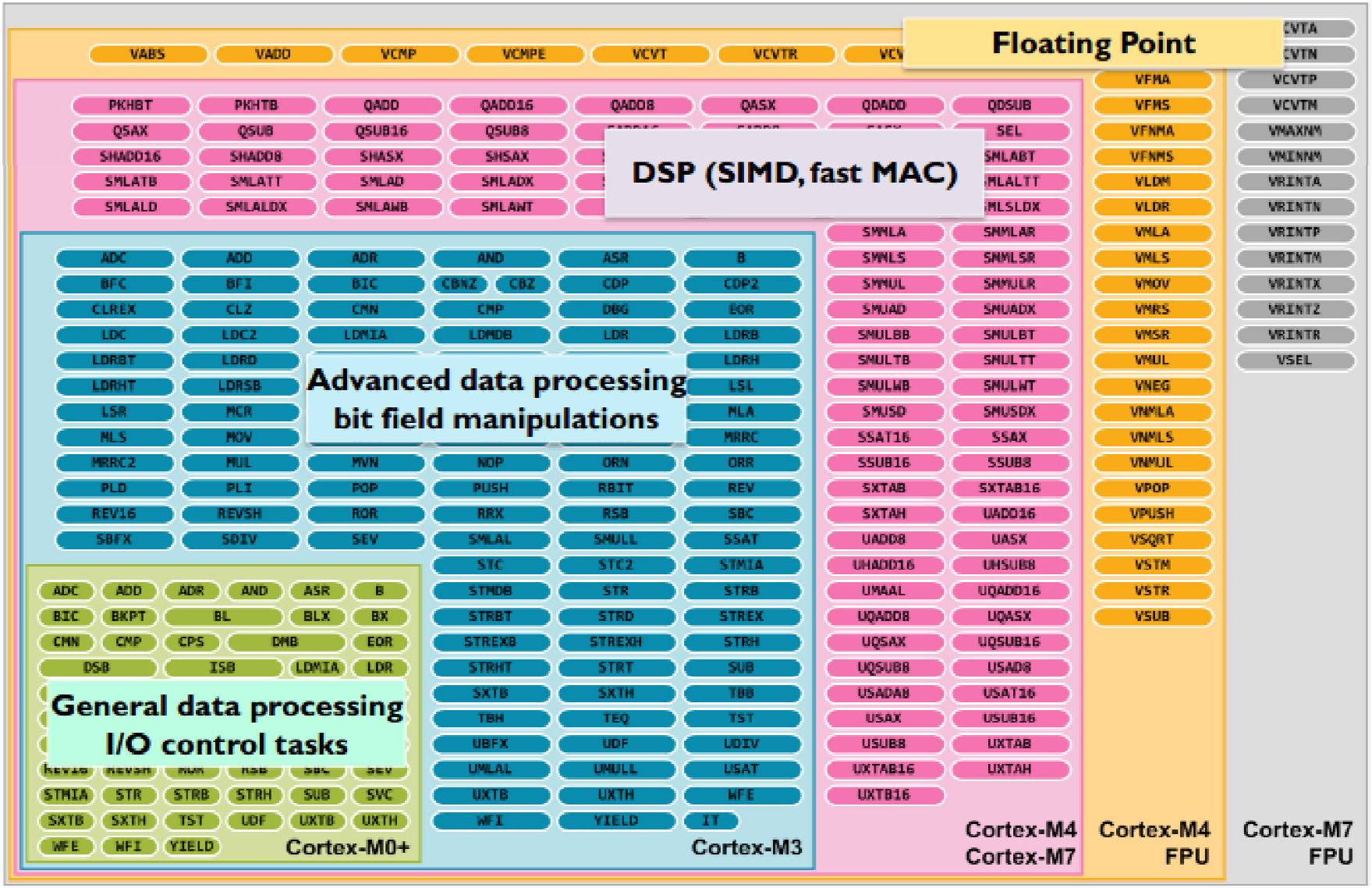
Cortex-M4 – rdzeń przewidziany do zastosowań w systemach wbudowanych, gdzie wymagane jest intensywne przetwarzanie sygnałów (m.in. Obsługuje dodatkowe instrukcje typowe dla aplikacji DSP); wersja M4F wyposażona jest w jednostkę FPU;

Cortex-M7 – rdzeń przewidziany do zastosowań w systemach wbudowanych wymagających wyższej wydajności;

Cortex-M23 – nowocześniejszy rdzeń odpowiadający zakresem zastosowań rdzeniowi M3; wyposażony w technologie podnoszące bezpieczeństwo systemu i samego procesora;

Cortex-M33/M35P – nowocześniejszy rdzeń odpowiadający zakresem zastosowań rdzeniowi M4; wyposażony w technologie podnoszące bezpieczeństwo systemu i samego procesora.

Rdzenie ARM Cortex-M



Środowiska programistyczne i narzędzia pomocnicze

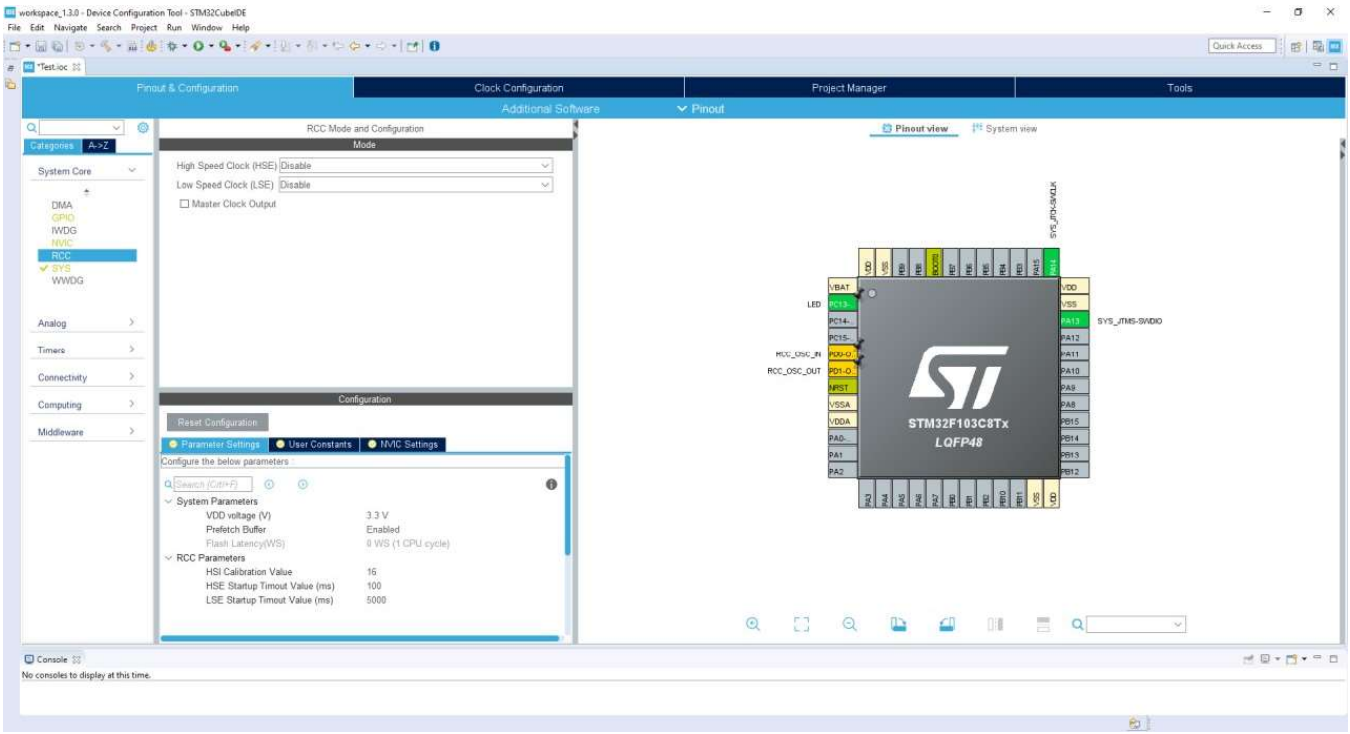
Środowisko programistyczne pozwalające na programowanie mikrokontrolerów STM32:

- STM32 Cube IDE;
- AC6 System Workbench for STM32;
- Keil uVision;
- IAR Embedded Workbench (IAR EWARM);
- Raisonance RIDE;
- Rowley CrossWorks,
- Cosmic IDE.

Środowisko STM32CubeIDE

STM32 CubeIDE jest środowiskiem wspieranym przez firmę STMicroelectronics, opartym na Eclipse. Środowisko jest dostępne na wszystkie systemy operacyjne: Windows, Linux i MacOS. STM32 CubeIDE umożliwia:

- graficzne konfigurowanie ustawień mikrokontrolera,
- kompilowanie,
- debugowanie,
- programowanie,
- zawiera zestaw bibliotek HAL.



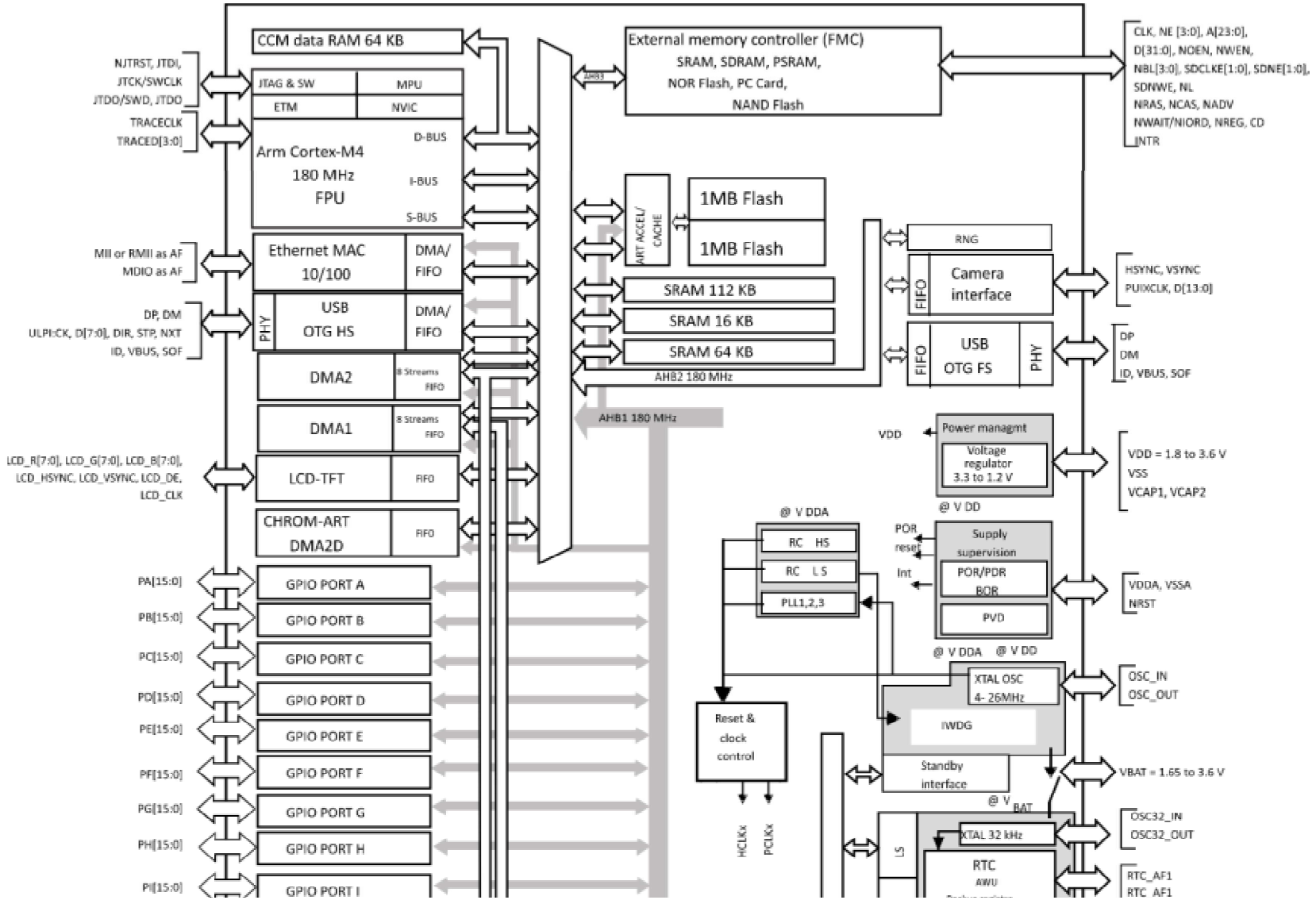
Biblioteka HAL (Hardware Abstraction Layer)

Programowanie mikrokontrolerów:

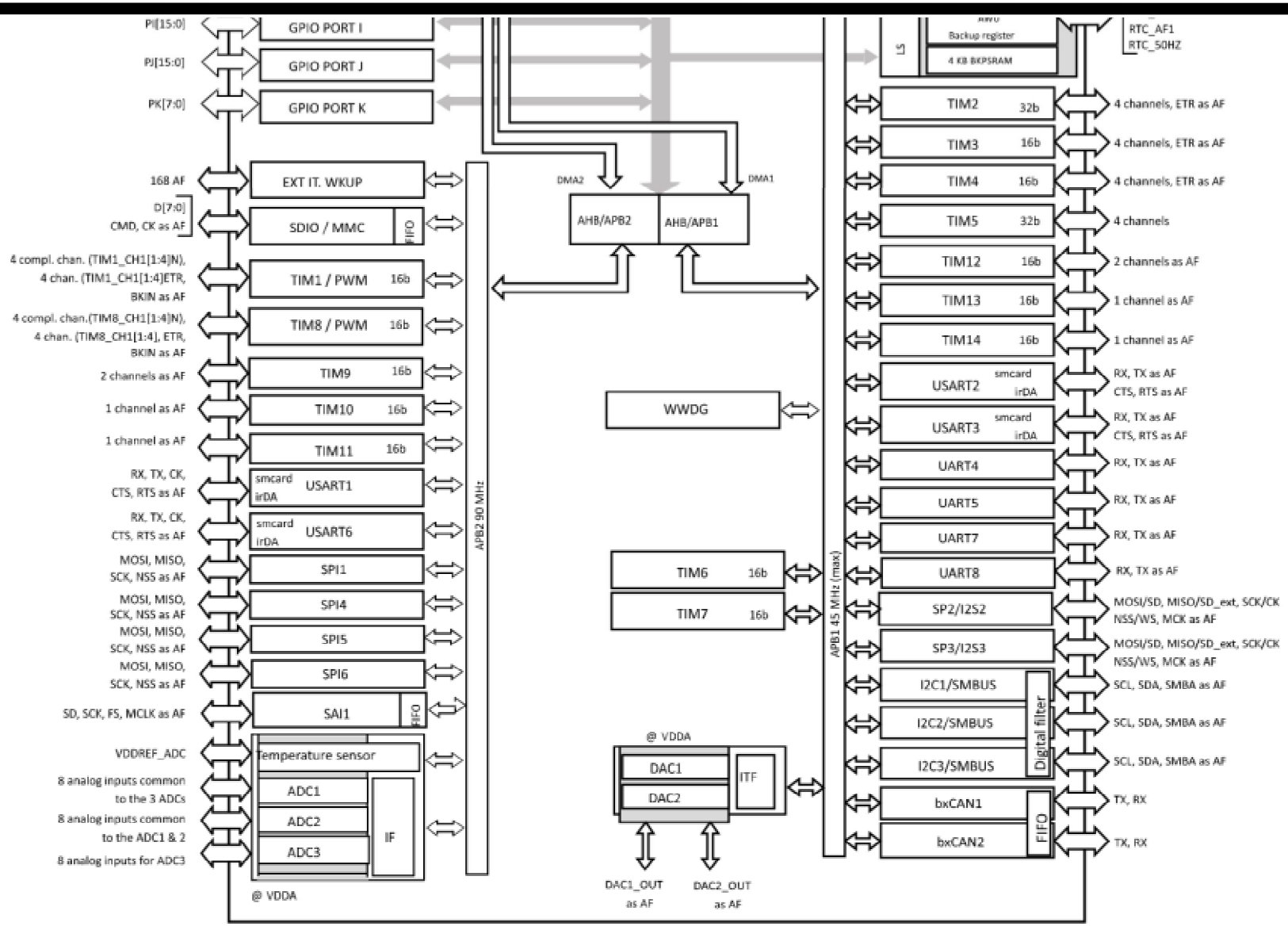
- bezpośrednia implementacja na rejestrach - podejście bardzo wydajne, ale wymaga dokładnej znajomości architektury MCU;
- wykorzystanie bibliotek HAL (ang. Hardware Abstraction Layer) .

W przypadku mikrokontrolerów STM32 producent oferuje zestaw bibliotek o nazwie HAL. Nazwa ta sugeruje, że biblioteki mają za zadanie zapewnić tzw. warstwę abstrakcji sprzętu tzn. umożliwić programiście operowanie uniwersalnymi nazwami (n. funkcji, poleceń, zmiennych, stałych) niezależnymi od konkretnego typu czy modelu mikrokontrolera. Konkretna implementacja danej funkcji czy operacji (odpowiednia dla danego mikrokontrolera) jest zdefiniowana na niższej warstwie implementacji. Wykorzystanie bibliotek HAL pozwala więc programiście uwolnić się od konieczności znajomości architektury poszczególnych mikrokontrolerów []

STM32F4 – Architektura



STM32F4 – Architektura



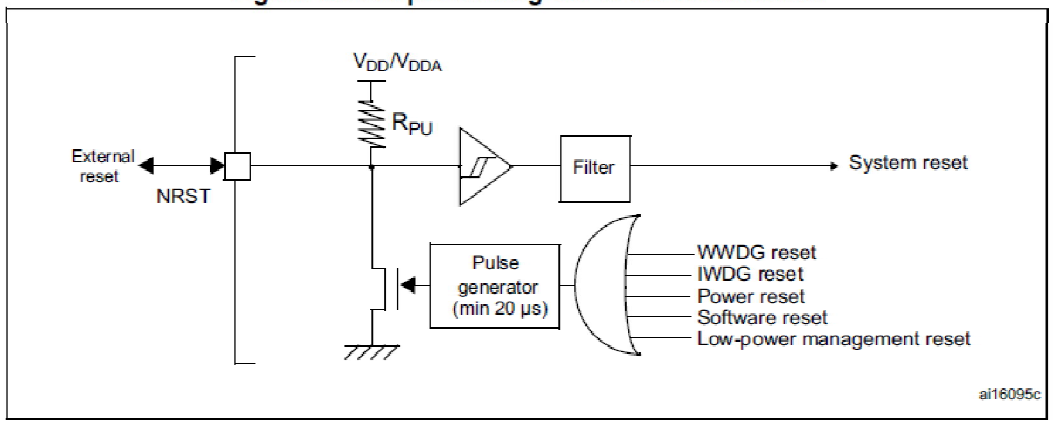
RCC (Reset and Clock Controller)

RCC – jest układem peryferyjnym przeznaczonym do zarządzania resetem systemu oraz dystrybucją sygnałów zegarowych do rdzenia i pozostałych układów peryferyjnych mikrokontrolera.

W mikrokontrolerach STM wyróżnia się 3 rodzaje resetu:

- **System reset** – zresetowanie systemu powoduje ustawienie wszystkich rejestrów do wartości resetowania (domyślne), z wyjątkiem flag resetowania w rejestrze CSR kontrolera zegara i rejestrach w domenie kopii zapasowej.
- **Power reset** – reset zasilania ustawia wszystkie rejestry do wartości resetu (domyślne), z wyjątkiem domeny kopii zapasowej.
- **Backup domain reset** – reset domeny zapasowej ustawia wszystkie rejestry RTC i rejestr RCC_BDCR do ich wartości resetu (domyślne).

Figure 15. Simplified diagram of the reset circuit

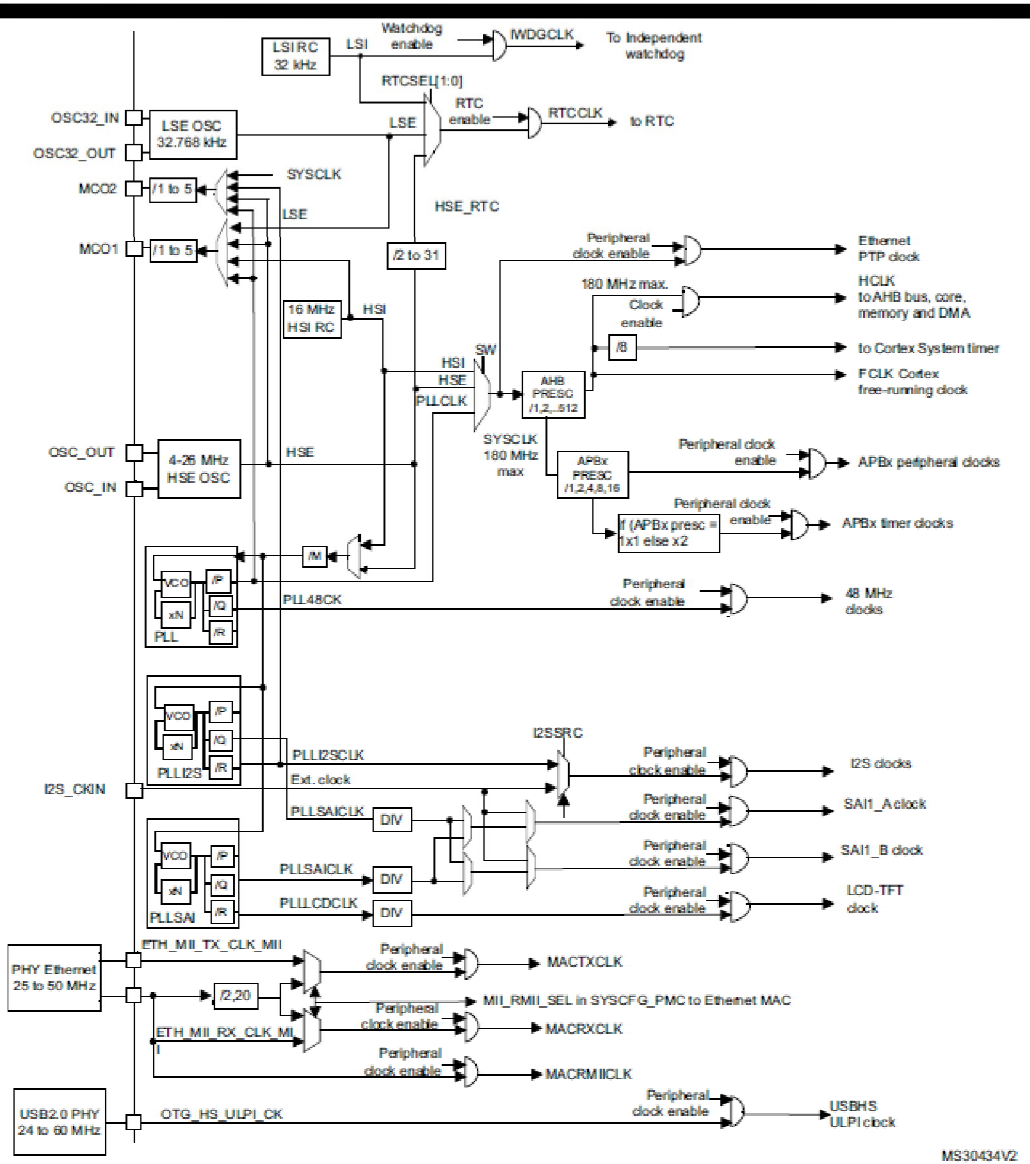


RCC (Reset and Clock Controler)

Pierwotne źródła sygnałów zegarowych:

- **HSI (High Speed Internal)** – wewnętrzny generator sygnału zegarowego do rdzenia i układów peryferyjnych o częstotliwości 16 MHz;
- **HSE OSC (High Speed External)** – zewnętrzne źródło sygnału zegarowego do rdzenia i układów peryferyjnych;
- **LSI (Low Speed Internal)** – wewnętrzny generator sygnału zegarowego do układów czasu rzeczywistego o częstotliwości 32 kHz;
- **LSE OSC (Low Speed External)** – zewnętrzne źródło sygnału zegarowego do układów czasu rzeczywistego;
- **I2S_CLKIN** – wejście zegarowe dla interfejsu audio I2S;
- **EthernetPHY_CLK** – wejście zegarowe dla inter. Ethernet;
- **USBPHY_CLK** – wejście zegarowe dla interfejsu USB.

Oprócz pierwotnych źródeł sygnałów zegarowych, w MCU ARM występują również **wtórne źródła sygnałów zegarowych**, które powstają na skutek dzielenia lub mnożenia źródeł pierwotnych. Dzięki temu istnieje możliwość dostosowania częstotliwości taktowania oddzielnie dla każdego układu peryferyjnego.



GPIO (General Purpose Inputs Outputs)

GPIO – jest podstawowym układem peryferyjnym mikrokontrolera. Ustawienie decyduje o funkcji jaką pełni pin mikrokontrolera. Możliwe funkcje pinu I/O:

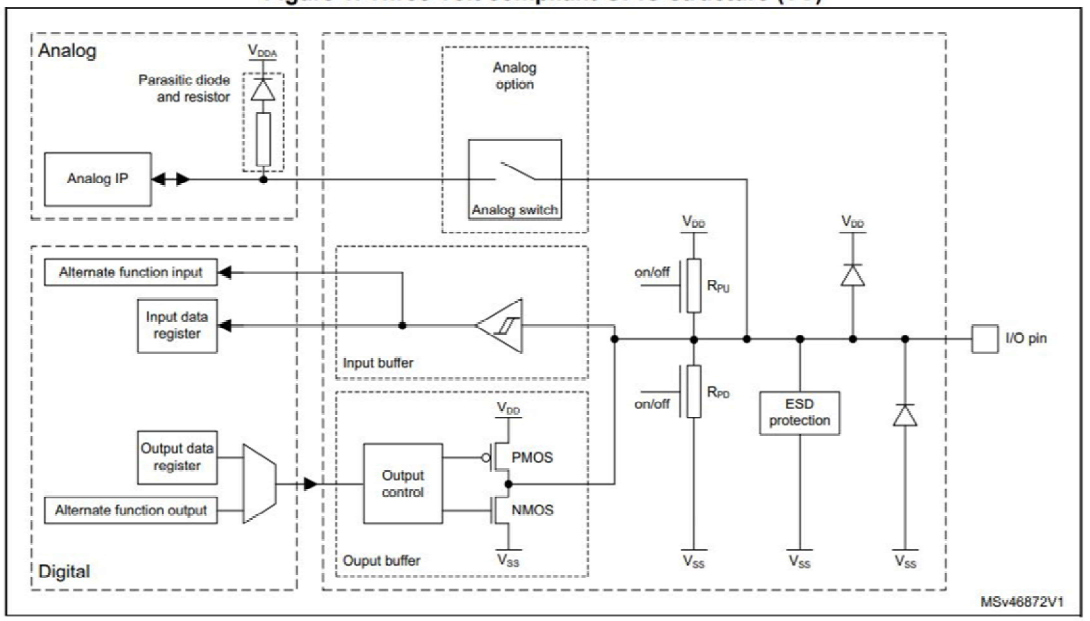
- **wejście ogólnego przeznaczenia** – stan pinu odczytywany przez użytkownika;
- **wyjście ogólnego przeznaczenia** – wyjście może być typu Push-Pull lub Open-Drain, stan pinu kontrolowany jest przez użytkownika;
- **wejście/wyjście analogowe** – pin kontrolowany przez układ analogowy (np. ADC, DAC);
- **funkcja alternatywna** – pin kontrolowany przez inny układ peryferyjny (np. UART, SPI, I2).

W przypadku wykorzystania pinu jako cyfrowego możliwe jest ustawienie rezystorów podciągających:

- **Pull-up** – włączenie rezystora pull-up;
- **Pull-down** – włączenie rezystora pull-down;
- **floating** – bez wewnętrznych rezystorów pull-up/down.

Możliwe jest też ustawienie maksymalnej częstotliwości przełączania pinu: **bardzo wolna, wolna, szybka, bardzo szybka**. Im większa szybkość przełączania tym większe zakłócenia może powodować.

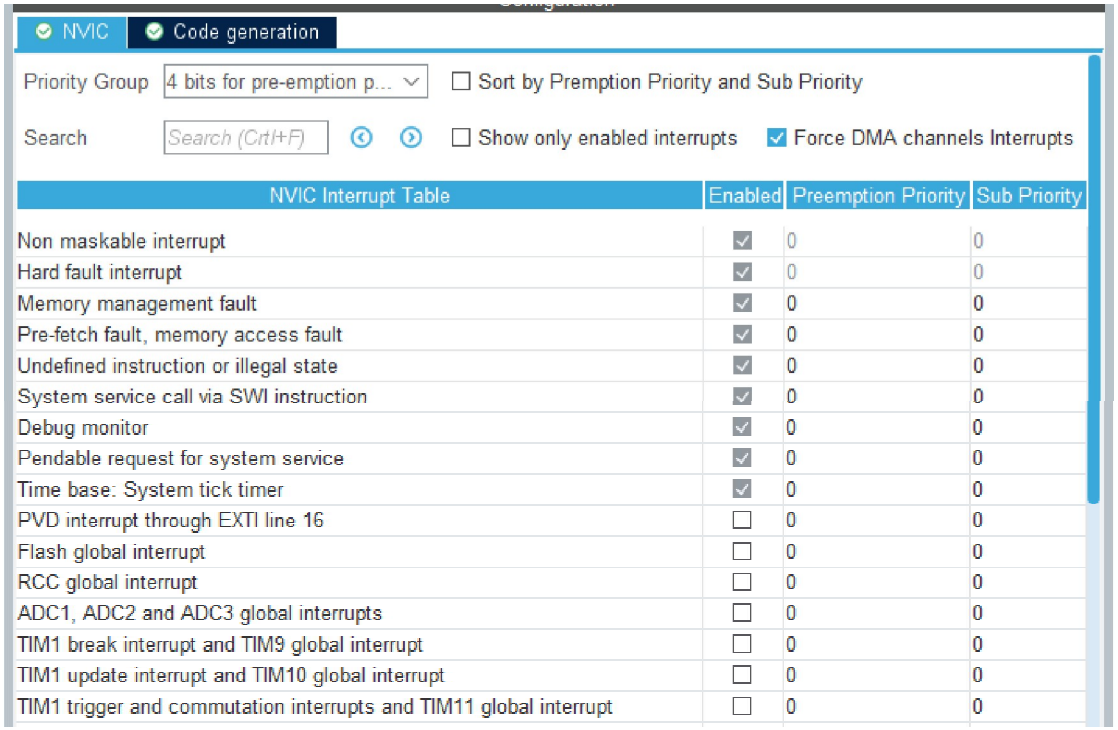
Figure 1. Three-volt compliant GPIO structure (TC)



NVIC (Nested Vectored Interrupt Controller)

Mikrokontrolery STM32 mają wbudowany mechanizm obsługi wyjątków, które obejmują zdarzenia wewnętrzne (np. otrzymanie bajtu danych po UART) , jak i zewnętrzne (pojawienie się zbocza narastającego/opadającego na jednym z wejść). Wystąpienie wyjątku powoduje przerwanie działania programu i przejście do funkcji obsługi przerwania.

Kontroler przerwań NVIC jest elementem rdzenia ARM, jego zadaniem jest zarządzanie i obsługą przerwań. NVIC oferuje priorytety przerwań. W przypadku gdy wystąpi przerwanie o wyższym priorytecie w trakcie obsługi przerwania o niższym priorytecie, w pierwszej kolejności następuje obsługa przerwania o priorytecie wyższym.



NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Pre-fetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	0	0
PVD interrupt through EXTI line 16	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
ADC1, ADC2 and ADC3 global interrupts	☐	0	0
TIM1 break interrupt and TIM9 global interrupt	☐	0	0
TIM1 update interrupt and TIM10 global interrupt	☐	0	0
TIM1 trigger and commutation interrupts and TIM11 global interrupt	☐	0	0

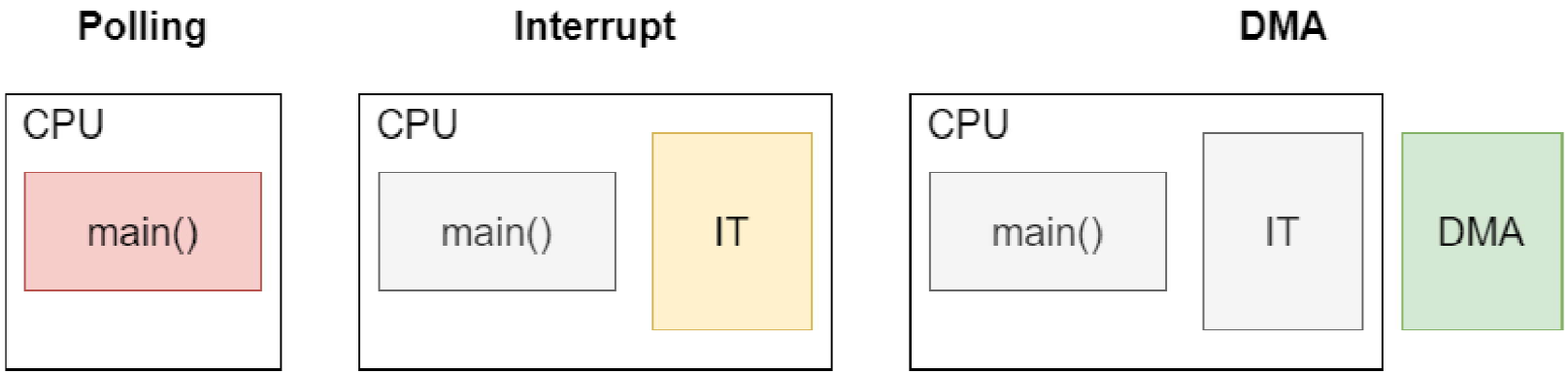
Polling, Interrrupt, DMA

W przypadku większości układów peryferyjnych w mikrokontrolerach możemy wyróżnić trzy sposoby ich obsługi.

Polling (odpytywanie) – jest najprostszą w metod polegającą na cyklicznym (w pętli głównej) odczytywaniu flag (rejestrów statusowych) i odpowiednim reagowaniu, czyli wpisywaniu danych do rejestrów lub odczytywaniu danych z rejestrów. Metoda ta najbardziej obciąża układ mikroprocesorowy oraz nadaje się do implementacji tylko w przypadku wolnych transmisji/działań.

Interrupt (przerwanie) – odpowiednio skonfigurowany interfejs może zgłosić przerwanie w określonej przez użytkownika sytuacji (np. ukończona transmisji kolejnego bajtu danych). Następnie w funkcji obsługi przerwania można zdefiniować działanie, które powinno zostać wykonane. Dzięki przerwaniom unikamy konieczności cyklicznego odczytywania rejestrów statusowych (sprawdzania flag) dotyczących interesujących nas sytuacji.

DMA (bezpośredni dostęp do pamięci) – polega na bezpośrednim przysyłaniu danych pomiędzy rejestrem układu peryferyjnego a pamięcią, z pominięciem CPU. Przesyłanie danych następuje według zdefiniowanego wzorca.

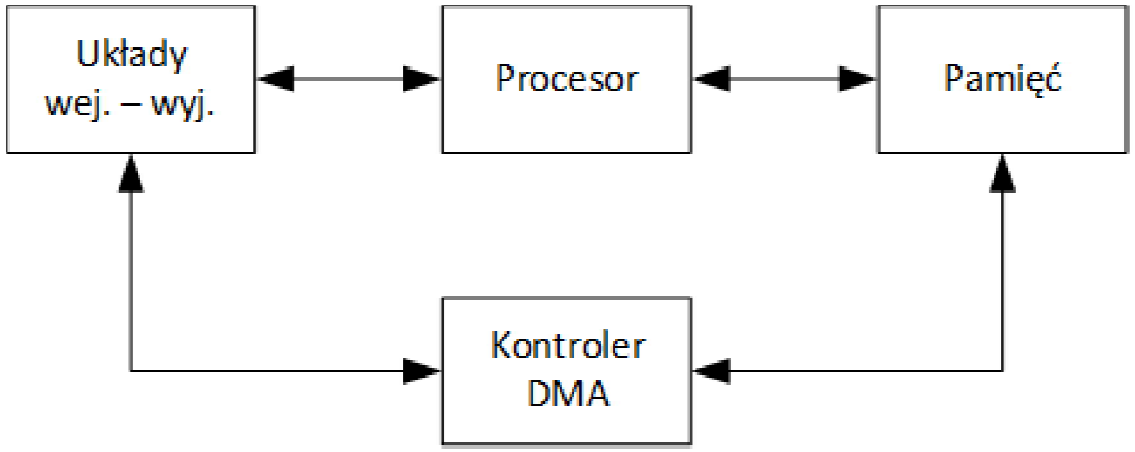


DMA (Direct Memory Access)

DMA czyli mechanizm bezpośredniego dostępu do pamięci to technika przesyłania danych z pominięciem jednostki CPU. DMA pozwala na przesyłanie danych:
- z pamięci do pamięci;
- z pamięci do układu peryferyjnego;
- z układu peryferyjnego do pamięci;
- z układu peryferyjnego do układu peryferyjnego.

Kontroler DMA ma możliwość automatycznej inkrementacji adresów po stronie układu peryferyjnego lub/i pamięci oraz możliwość cyklicznego transferu (po zakończeniu jednego rozpoczyna się kolejny od początku skonfigurowanego adresu – na zasadzie bufora kołowego). Pozwala to dobrać sposób przesyłania danych według potrzeb.

DMA generuje trzy rodzaje przerwań:
- po wykonaniu połowy transferu;
- po wykonaniu całego transferu;
- w przypadku błędu.



SysTick

SysTick jest zegarem systemowym przeznaczonym do pomiaru czasu w systemie. Jest częścią rdzenia w mikrokontrolerach ARM. Standardowo SysTick generuje przerwanie z częstotliwością 1 kHz (1 ms).

```
133 /**
134   * @brief  This function handles SysTick Handler.
135   * @param  None
136   * @retval None
137   */
138 void SysTick_Handler(void)
139 {
140   HAL_IncTick();
141 }
142
```

Timer'y

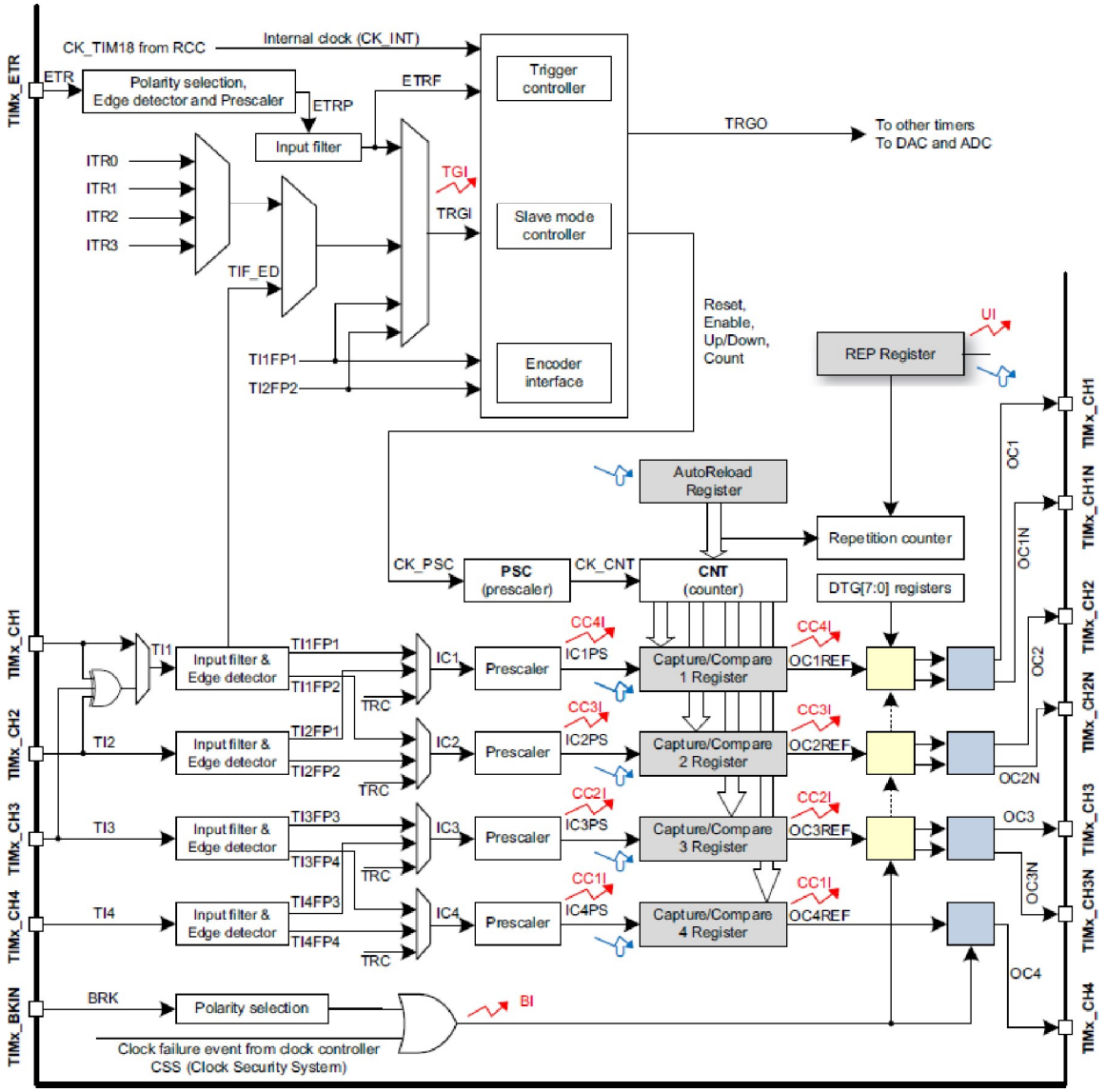
W mikrokontrolerach STM32 wyróżnia się cztery rodzaje Timer'ów, zazwyczaj są to układy 16-bitowe (niektóre 32-bitowe), zazwyczaj posiadają wbudowany (16-bitowy) preskaler:

Advanced-control timer – najbardziej zaawansowane układy, poza podstawowymi funkcjami posiadają możliwość generowania sygnału PWM, pomiaru czasu trwania impulsu, obsługi enkodera inkrementalnego, czujników Halla, itd.;

General-purpose timer – układ czasowy ogólnego przeznaczenia, zmniejszona funkcjonalność w stosunku do Advanced-control timer;

Basic timer – podstawowy układ czasowy; nie mają kanałów wejściowych ani wyjściowych, a ich podstawowym zadaniem jest funkcja odliczania czasu;

Low-power timer – układ czasowy dedykowany do funkcji o niskim poborze mocy; mogą być taktowane za pomocą oscylatorów LSE lub LSI i pracować w trybach Low Power.



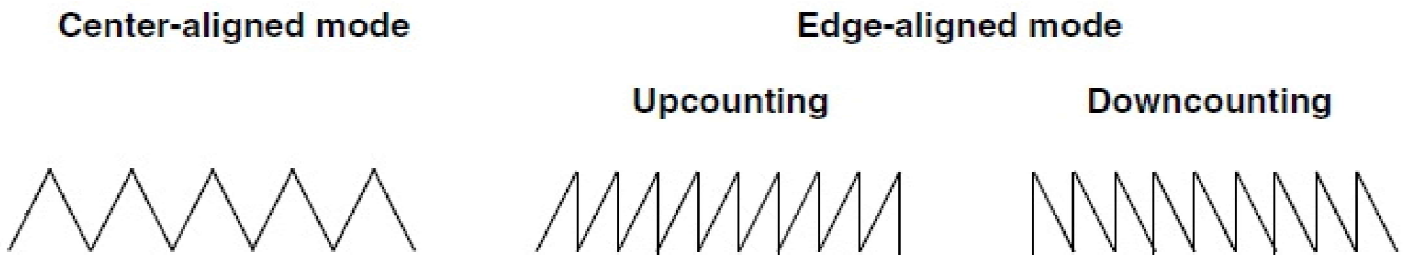
Timer'y – funkcja licznika (pomiar czasu)

Podstawową funkcją licznika jest pomiar czasu. Każde przepełnienie licznika może generować zdarzenie, które może zostać wykorzystywane m.in. jako źródło przerwania czy wyzwolenie pomiaru przetwornika ADC/DAC. Liczniki mogą pracować w jednym z trzech trybów:

tryb zliczania w górę (upcounting) – licznik zaczyna liczenie od 0 i zwiększa ją aż do osiągnięcia wartości umieszczonej w rejestrze Auto-Reload (ARR); po przepełnieniu (overflow) wartość licznika jest resetowana do 0 i ponownie następuje jej zwiększanie;

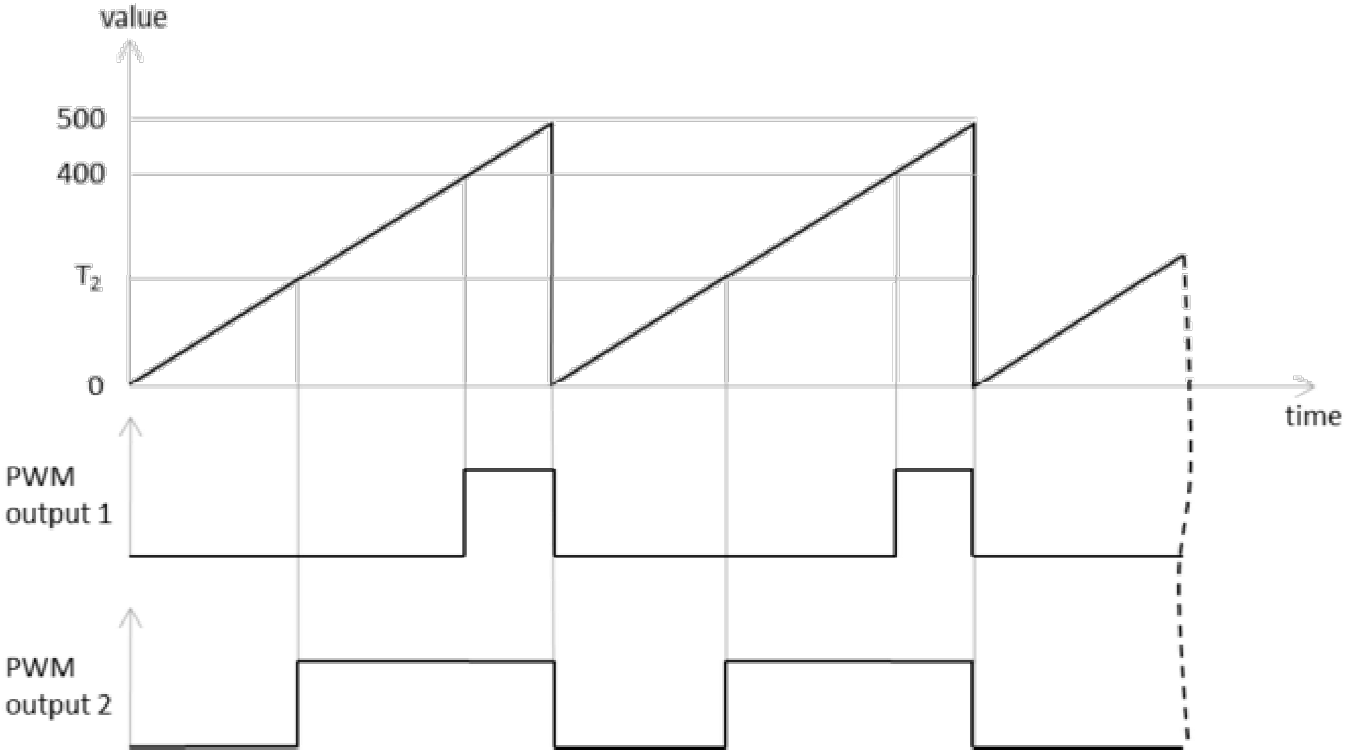
tryb zliczania w dół (downcounting) – odliczanie zaczyna się od wartości umieszczonej w rejestrze Auto-Reload (ARR) i jest zmniejszana aż do osiągnięcia 0; wówczas następuje niedomiar (underflow) i generowane jest zdarzenie; licznik po resecie przyjmuje ponownie wartość z rejestru Auto-Reload;

tryb zliczania w górę i dół (center-aligned mode) – licznik po uruchomieniu startuje od wartości 0 i zlicza w sposób analogiczny jak w trybie upcounting; po osiągnięciu wartości umieszczonej w rejestrze Auto-Reload, zaczyna zliczać w dół tak jak w trybie downcounting; następnie ponownie zlicza w górę; każde osiągnięcie przez licznik wartości 0 i tej z rejestru ARR powoduje wygenerowanie zdarzenia.



Timer - PWM

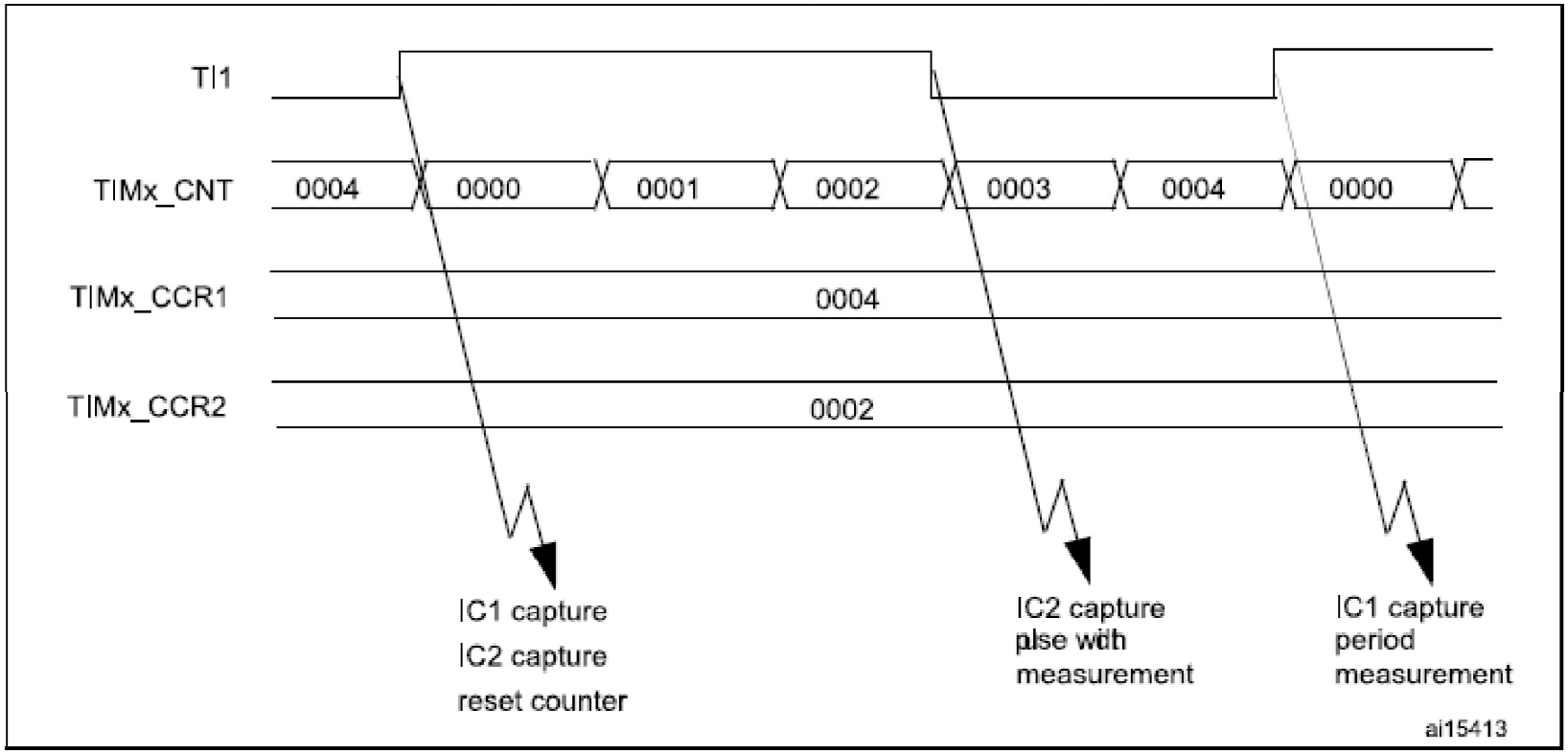
Liczniki mogą pracować w trybie PWM przeznaczonym do generowania sygnałów PWM, zazwyczaj jeden licznik może generować 3 lub 4 sygnały PWM jednocześnie. Dzięki temu możliwe jest sterowanie m.in. silnikiem BLDC/PSMS. Dodatkowo Advanced-Control Timers umożliwiają generowanie sygnałów komplementarnych (z ustawieniem tzw. „czasów martwych”) do niezależnego sterowania „górnym” i „dolnym” tranzystorem w półmostku.



Timer – PWM Input mode

Liczniki posiadają funkcję pomiaru parametrów wejściowego sygnału PWM. Możliwy jest pomiar okresu oraz współczynnika wypełnienia sygnału PWM, który stanowi wejście dla układu.

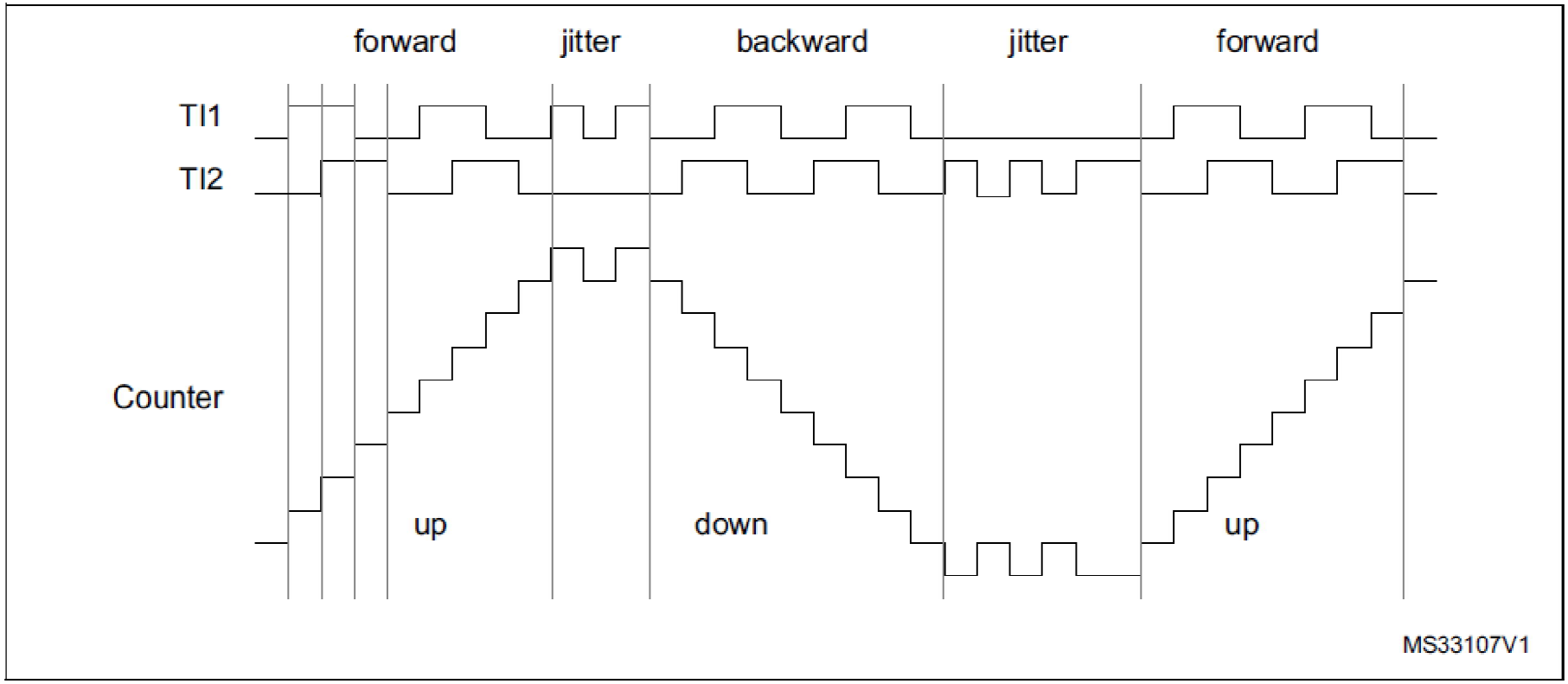
Figure 116. PWM input mode timing



Timer - Encoder interface mode

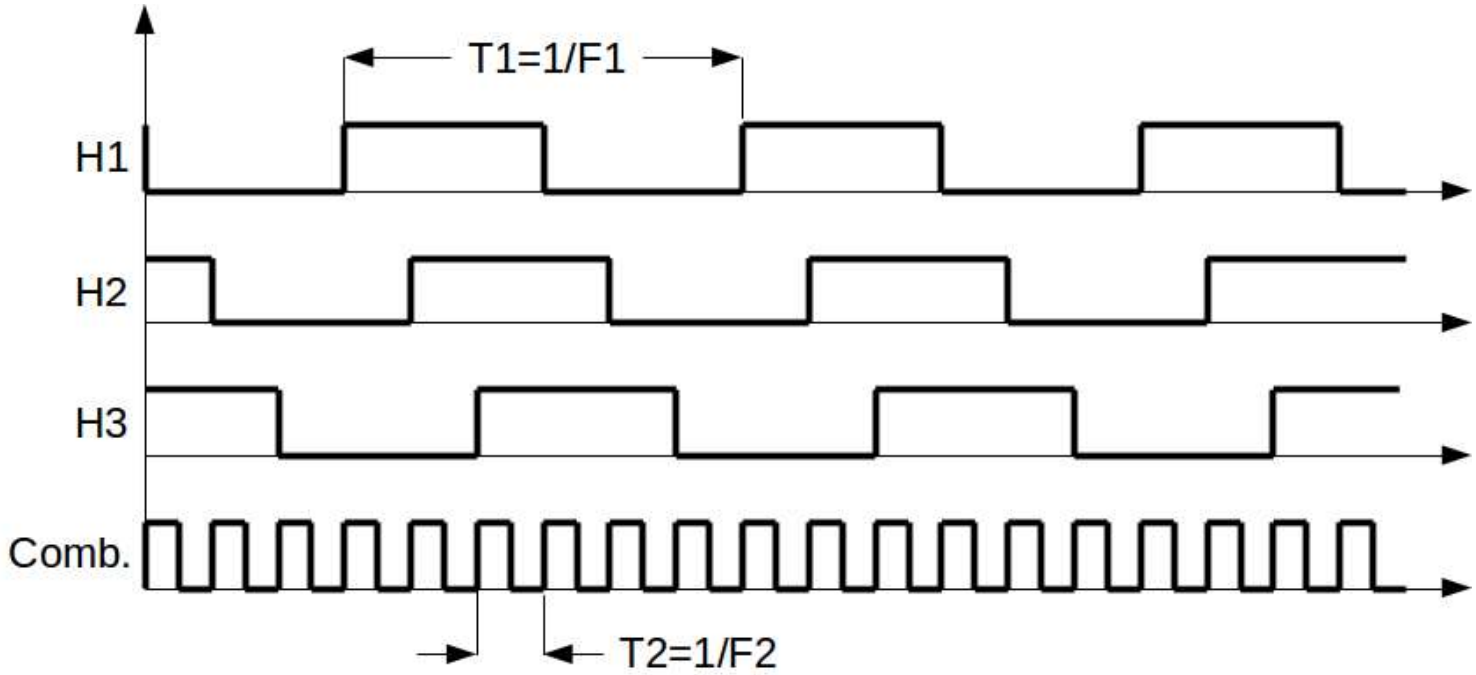
Licznik można skonfigurować do pracy w trybie umożliwiającym odczyt pozycji enkodera inkrementalnego. W tym trybie wartość licznika (CNT) określa nam pozycję wyznaczoną przy pomocy interfejsu kwadraturowego.

Figure 127. Example of counter operation in encoder interface mode.



Timer – Hall Sensor Mode

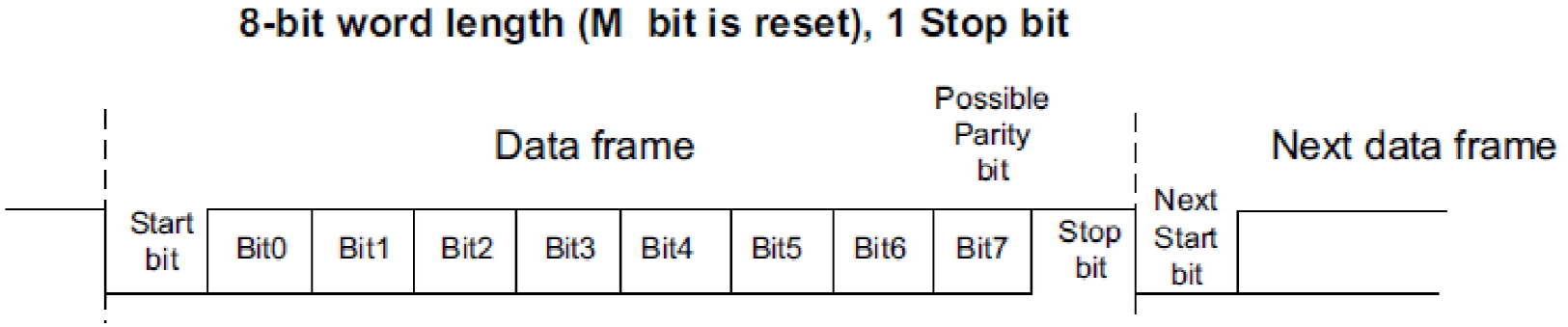
Tryb **Hall Sensor Mode** służy do pomiaru prędkości obrotowej silnika BLDC na podstawie sygnałów z czujników Halla. W pierwszym etapie generowany jest sygnał będący wynikiem operacji XOR z sygnałów H1, H2 i H3. Następnie mierzony jest czas pomiędzy poszczególnymi zboczami sygnałów. Na podstawie pomiaru tego czasu oraz ilości par biegunów napędu można wyznaczyć prędkość obrotową.



USART/UART (Universal Synchronous Asynchronous Receiver Transmitter)

Interfejs **USART/UART** jest podstawą do implementacji komunikacji poprzez RS232, RS422 oraz RS485. Interfejs ten wspiera również tryb LIN (Local Interconnection Network), Smartcard Protocol oraz IrDA (Infrared Data Association).
Ze względu na brak synchronizacji za pomocą zegara, w przypadku transmisji UART kluczowe jest, aby oba urządzenia miały takie same parametry pracy UART.

Parametry transmisji:
- prędkość transmisji (baudrate): 9600, 19200, 38400, 57600, 115200;
- ilość bitów danych: 7, 8, 9;
- bit parzystości: brak, parzysty, nieparzysty;
- Ilość bitów stopu: 1, 1.5, 2.



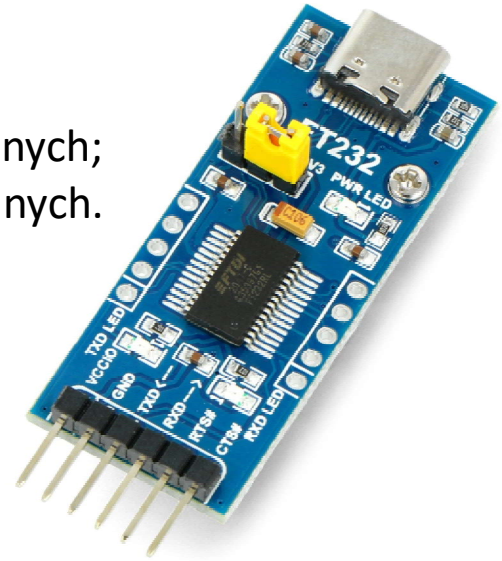
USART/UART (Universal Synchronous Asynchronous Receiver Transmitter)

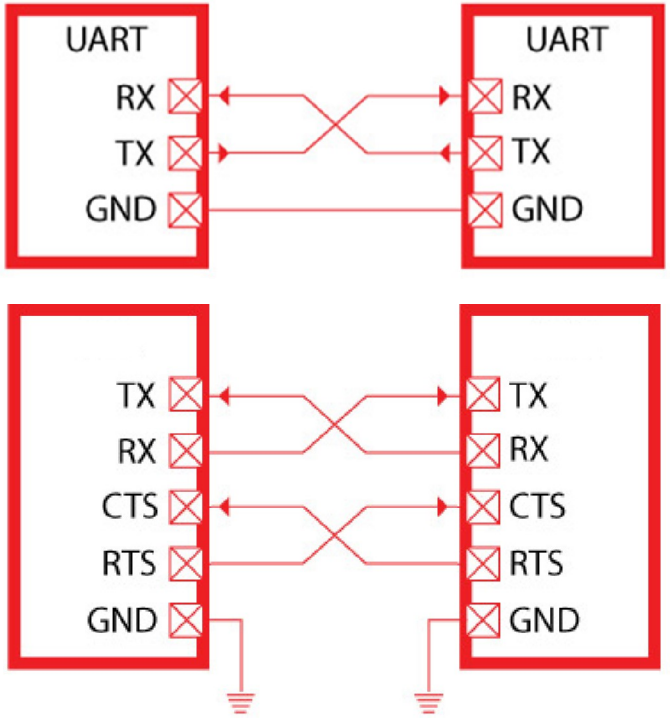
UART może pracować w trybie:

- **Simplex** – transmisja jednokierunkowa, z nadajnika do odbiornika;
- **Half Duplex** – transmisja dwukierunkowa, transmisje pomiędzy urządzeniami odbywają się w różnych chwilach czasowych;
- **Full Duplex** – transmisja dwukierunkowa, oba urządzenia mogą nadawać dane w tym samym czasie.

UART może wykorzystywać do transmisji 2 linie (bez kontroli przepływu) lub 4 linie (z kontrolą przepływu) danych:

RX – odbiór danych;
TX – transmisja danych;
RTS – żądanie transmisji danych;
CTS – gotowość odbioru danych.



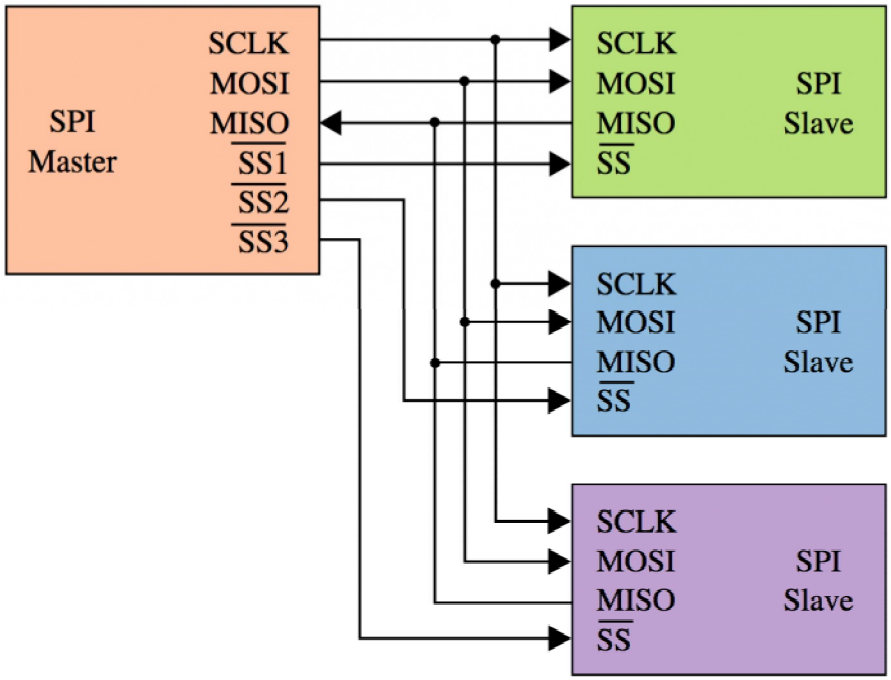


SPI (Serial Peripheral Interface)

SPI to szeregowy interfejs transmisji danych, powszechnie wykorzystywany do komunikacji z urządzeniami peryferyjnymi takimi, jak zewnętrzne przetworniki ADC i DAC, czujniki, pamięci FLASH, karty SD.

Komunikacja poprzez SPI realizowana jest z wykorzystaniem 4 sygnałów:

- **CS (SS)** – służy jako linia wyboru urządzenia, z którym następuje komunikacja;
- **SCK (SCLK)** – linia zegarowa, sygnał generowany przez urządzenie master;
- **MOSI** – linia danych, która przesyła informacje z urządzenia Master do Slave (Master Output Slave Input);
- **MISO** – linia danych, która przesyła informacje z urządzenia Slave do Master (Master Input Slave Output).



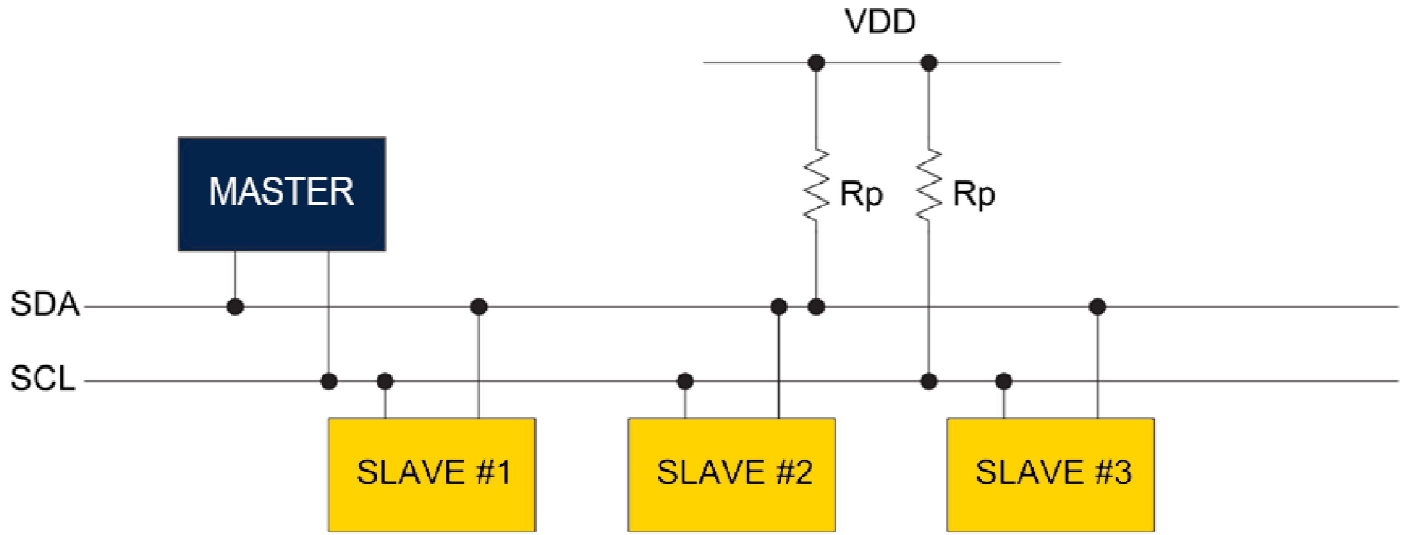
I2C (Inter-Integrated Circuit)

I2C to dwukierunkowa, pół-dupleksowa, magistrala komunikacji szeregowej, wykorzystywana do komunikacji między układami scalonymi na niewielkie odległości (najczęściej w obrębie jednego obwodu drukowanego).

Magistrala I2C składa się z dwóch sygnałów:

- SCL (Serial Clock) to linia zegara przeznaczona do synchronizacji;
- SDA (Serial Data) to linia, na której master i slave wysyłają i odbierają dane.

Linie SDA i SCL muszą być spolaryzowane rezystorami podciągającymi. Wartość tych rezystorów zależą od długości magistrali i prędkości transmisji. Typową wartością jest 4.7 kΩ.

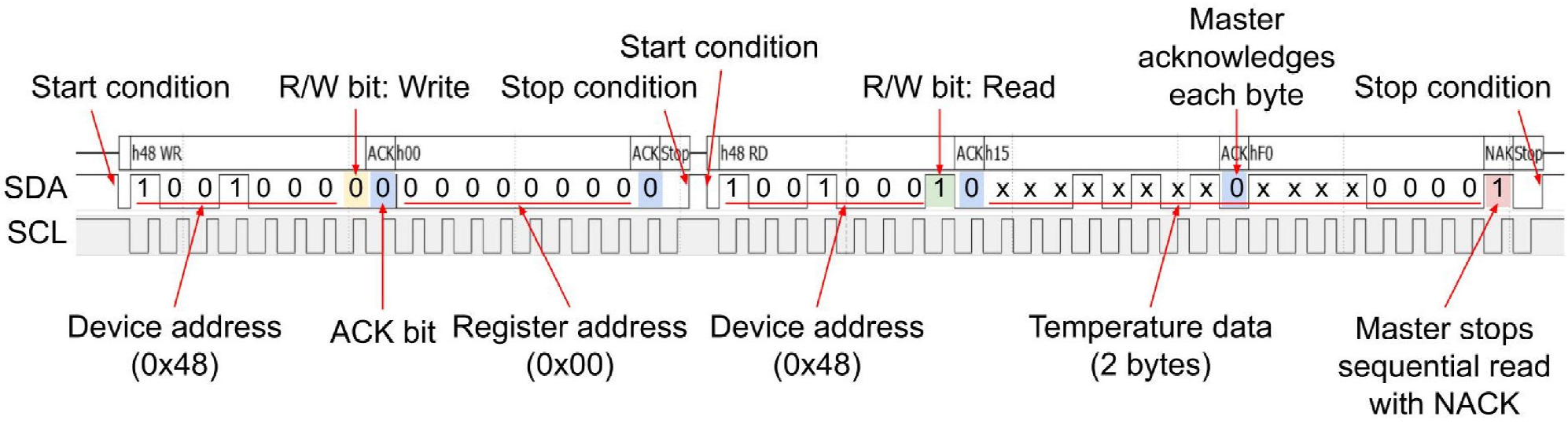


I2C (Inter-Integrated Circuit)

I2C wykorzystuje trzy prędkości transmisji danych:

- tryb standardowy (Standard-Mode) z szybkością transmisji 100 kbit/s;
- tryb szybki (Fast-Mode) z szybkością transmisji 400 kbit/s;
- tryb szybki Plus (Fast-Mode Plus) z szybkością transmisji 1 Mbit/s.

Transmisja zawsze inicjowana jest przez urządzenie Master. Wybór urządzenia Slave, z którym następuje komunikacja dokonywany jest poprzez podanie jego adresu jako pierwszego bajtu podczas transmisji. Poniżej przykładowy przebieg czasowy odczytu danych (temperatury) z czujnika.



ADC (Analog to Digital Corverter)

Mikrokontrolery STM32 posiadają od 1 do 3 wbudowanych przetworników ADC. Każdy z przetworników jest przetwornikiem wielokanałowym (zazwyczaj 16-kanałowym), co umożliwia pomiar wielu wielkości analogowych. Standardowa rozdzielczość przetworników ADC to 12-bitów (niektóre układy posiadają przetworniki 16-bitowe). Dodatkowo możliwy jest wybór rozdzielczości przetwornika od 6-bitów do 12-bitów (mniejsza rozdzielność umożliwia szybszy proces przetwarzania).

Konwerter ADC może pracować w różny sposób w zależności od wymagań aplikacji i użytkownika:

Single conversion mode – tryb pojedynczej konwersji, po otrzymaniu sygnału wyzwolenia konwersji (programowego lub z zewnętrznego wyzwalacza) wykonuje jedną sekwencję;

Continuous conversion mode – tryb ciągłej konwersji, po zakończeniu pomiarów w jednej sekwencji od razu następuje uruchomienie kolejnej;

Discontinuous conversion mode – tryb nieciągłej konwersji, jest dostępny w przypadku, gdy sekwencja zawiera więcej niż jeden kanał; wówczas pomiar na każdym kanale w sekwencji będzie wymagał oddzielnego sygnału wyzwolenia (np. jeżeli sekwencja składa się z 4 kanałów, do wykonania jednej sekwencji będą potrzebne 4 sygnały wyzwolenia).

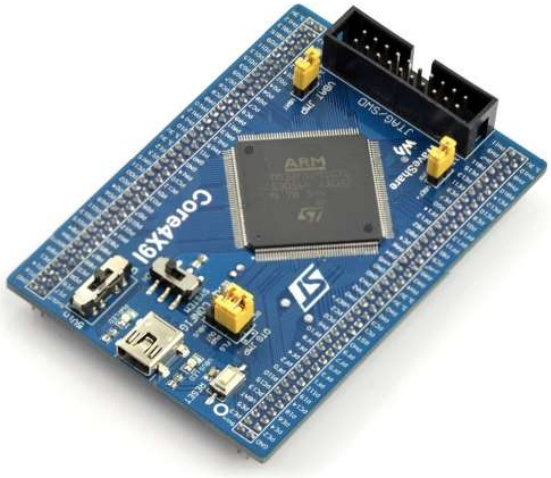
Pozostałe układy peryferyjne

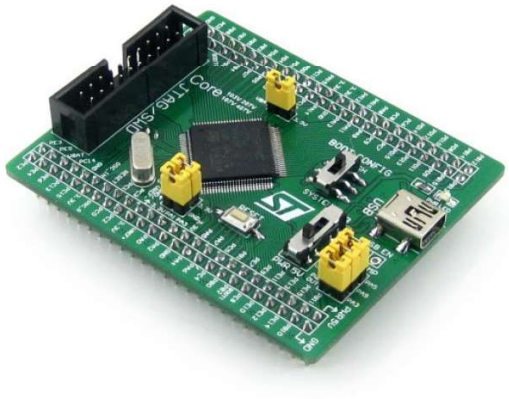
W mikrokontrolerach występują również inne układy peryferyjne:

- DAC (Digital to Analog Converter);
- CAN (Controller Area Network);
- ETH (Ethernet);
- USB (OTG_FS/OTG_HS);
- FSMC (Flexible Static Memory Controller);
- FMC (Flexible Memory Controller);
- DCMI (Digital Camera Interface);
- LTDC (LCD-TFT Controller);
- RNG (Random Number Generator);
- CRC Calculation Unit;
- RTC (Real Time Clock);
- SDIO (Secure Digital Input/Output Interface) – SD Card Interface;
- SAI (Serial Audio Interface);
- IWDG/WWDG (Watchdogs).

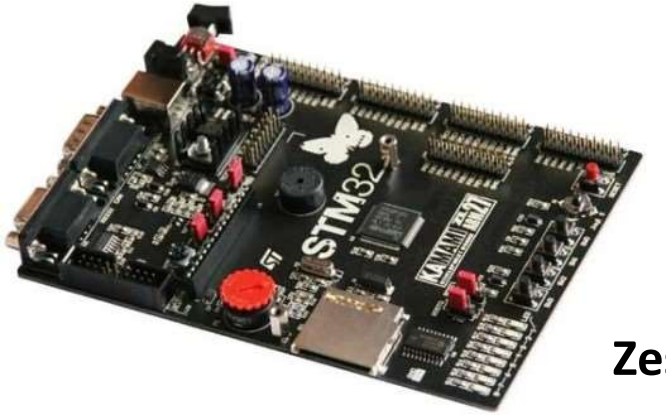
Zestawy uruchomieniowe STM32

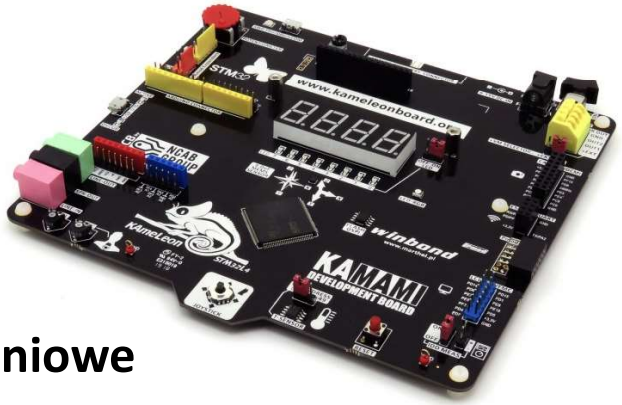
Discovery i NUCLEO





Układy Waveshare





Zestawy uruchomieniowe KAMAMI

LITERATURA

[1] ST: STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs - RM0090 Reference manual; 2019.

[2] ST: STM32F427xx STM32F429xx; 2018.

[3] Marek Galewski: STM32 - Aplikacje i ćwiczenia w języku C z biblioteką HAL.

[4] Akademia ETI – Studenckie Koło Naukowe Politechniki Gdańskiej: Prezentacja Mikrokontrolery, 2016

[5] https://www.stm32wrobotyce.pl/2022/10/06/kurs-stm32-ll-cz-1-biblioteki-low-layer-nucleo-g071rb-stm32cubeide/

[6] https://www.st.com

Rysunki

[A] https://gsasindia.com/blog/cortex-a-cortex-r-and-cortex-m/

[B] https://stackoverflow.com/questions/63690502/how-i-can-use-the-same-asm-code-for-different-cortex-m

[C] https://www.mikroe.com/blog/uart-serial-communication

[D] https://visualgdb.com/tutorials/arm/stm32/pwm/

[E] https://www.digikey.be/en/maker/projects/getting-started-with-stm32-i2c-example/ba8c2bfef2024654b5dd10012425fa23

[F] https://wiki.st.com/stm32mcu/wiki/File:i2c_bug.png