Embedded Systems

Microcontrollers

Carlo Brandolese, William Fornaciari

AA 2020-2021

Microprocessors

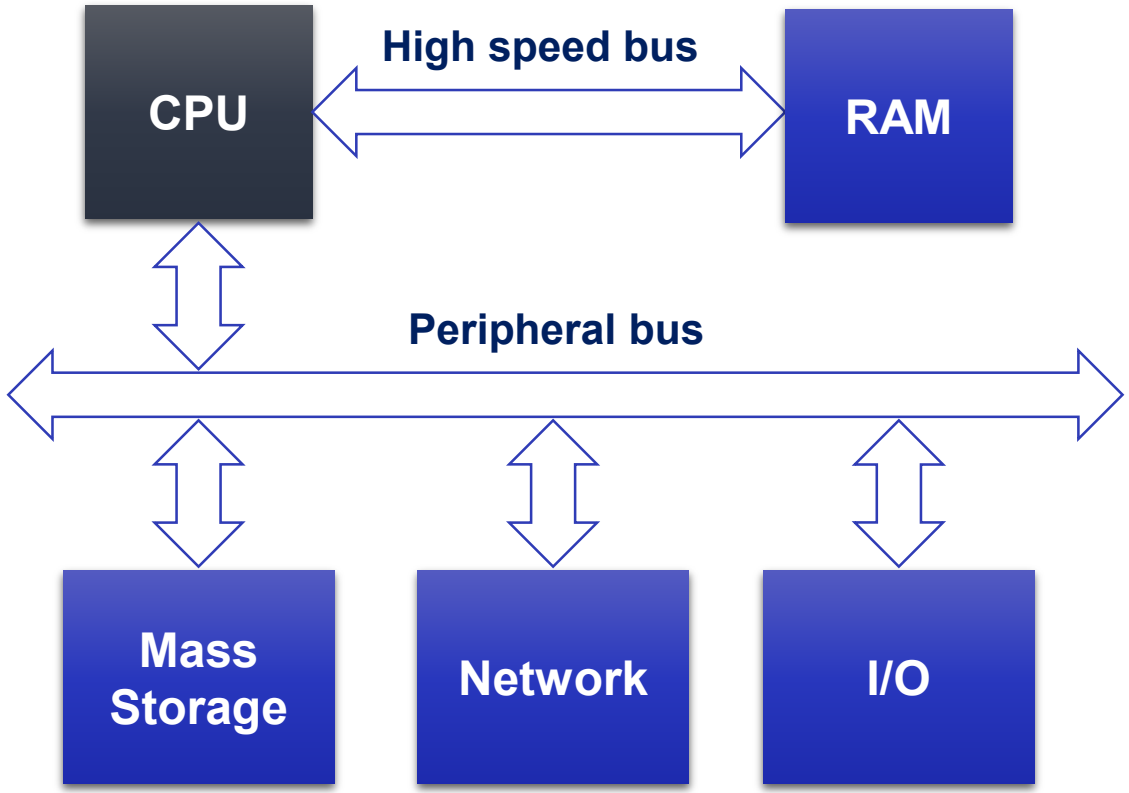
Constitute the core of general purpose computer

- Suitable for computation
- From desktop, to server, to supercomputers
- Deliver high to very-high computational power
- Require high-performance, large external memories

Main characteristics

- Complex multi-core architectures
- Very high clock speed (2 – 5 GHz)
- High cost (100 – 5000 EUR)
- Power hungry (50 – 1500 W)
- Large form factors
- Limited interaction with the environment

Microprocessors



Microcontroller

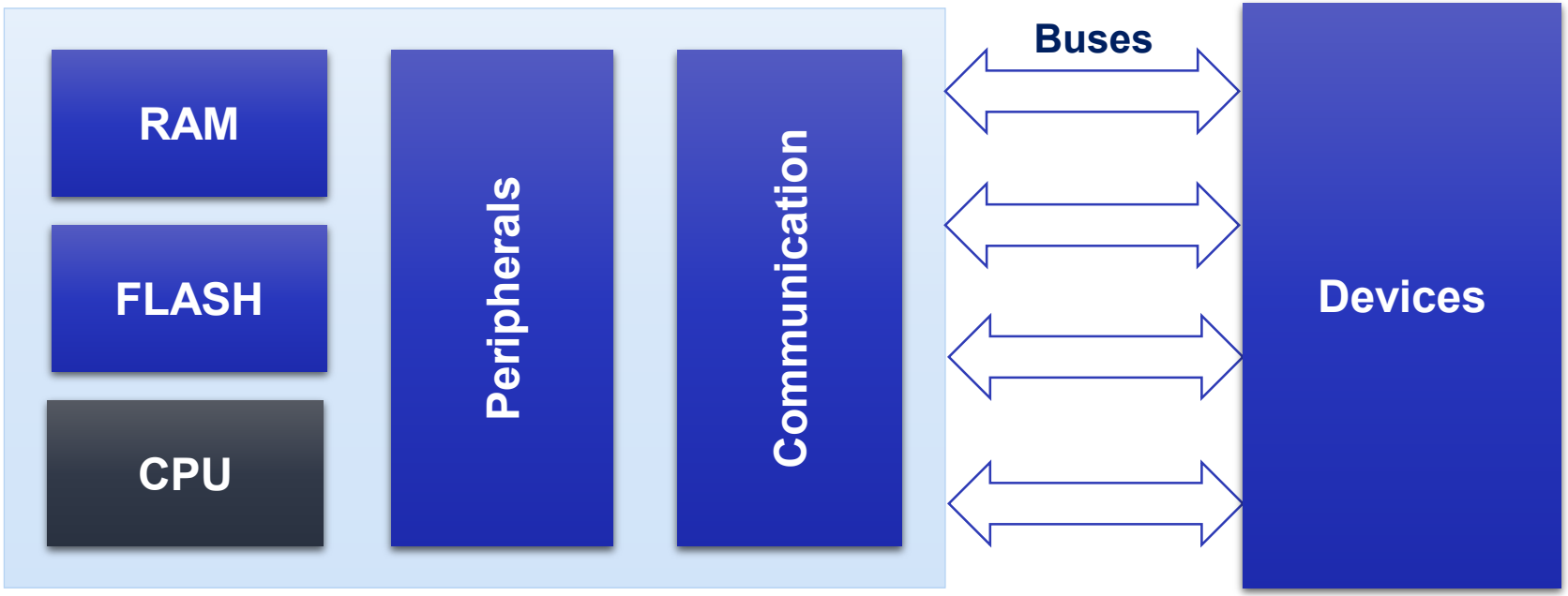
Used in a variety of dedicated (embedded) applications

- Limited computational power
- Use small, integrated memories

Main characteristics

- Simple architectures (8, 16, 32 bit, small pipelines)
- Low clock speed (8 – 160 MHz)
- Low cost (0.5 – 10 EUR)
- Low to very-low power consumption (0.1 – 300 mW)
- Small footprint
- Designed to strongly interact with the environment

Microcontrollers



Example of common microcontrollers

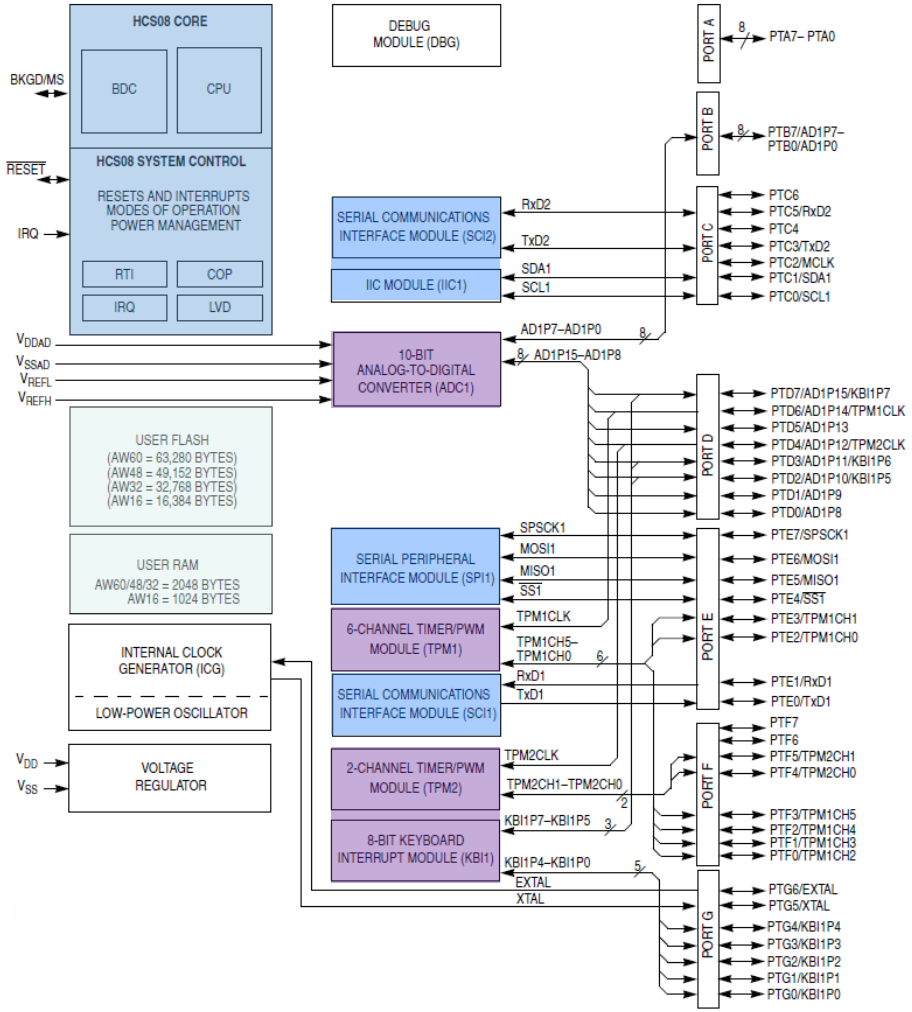
Example: NXP HCS08

CPU

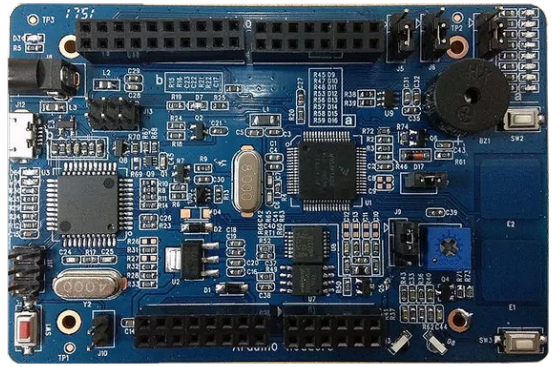
Memories

Interfaces

Peripherals

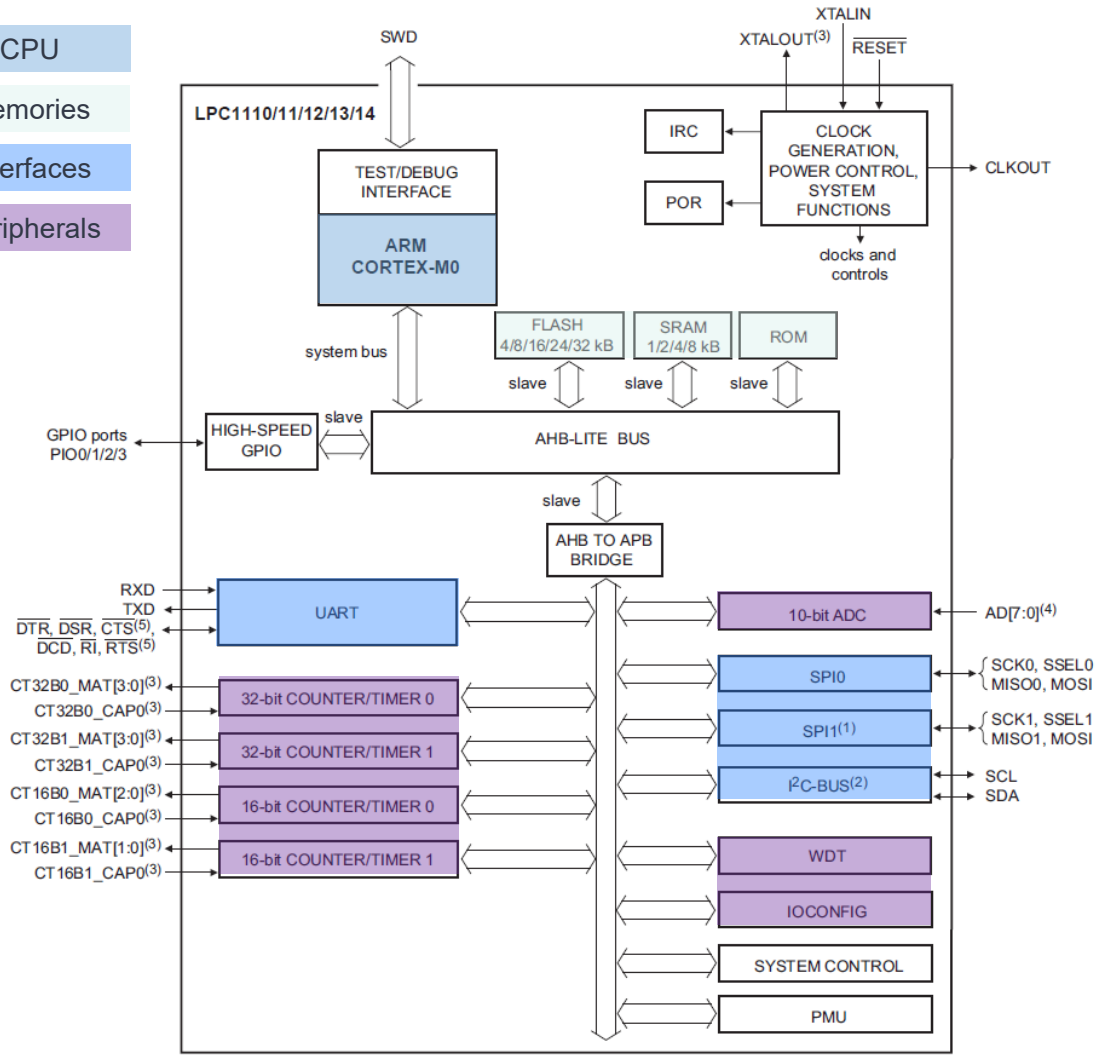


- The 8-bit S08 MCU portfolio includes more than 30 product families, providing a wide range of features and price options for product differentiation, to cover all kinds of automotive, industrial, and consumer applications



Example: NXP LPC111x – Cortex M0

CPU
Memories
Interfaces
Peripherals



- The LPC1110/11/12/13/14/15 are an ARM Cortex-M0 based, low-cost 32-bit MCU family, designed for 8/16-bit microcontroller applications, offering performance, low power, simple instruction set and memory addressing together with reduced code size compared to existing 8/16-bit architectures
- The LPC1110/11/12/13/14/15 operate at CPU frequencies of up to 50 MHz
- The peripheral complement of the LPC1110/11/12/13/14/15 includes up to 64 kB of flash memory, up to 8 kB of data memory, one Fast-mode Plus I2C-bus interface, one RS-485/EIA-485 UART, up to two SPI interfaces with SSP features, four general purpose counter/timers, a 10-bit ADC, and up to 42 general purpose I/O pins.

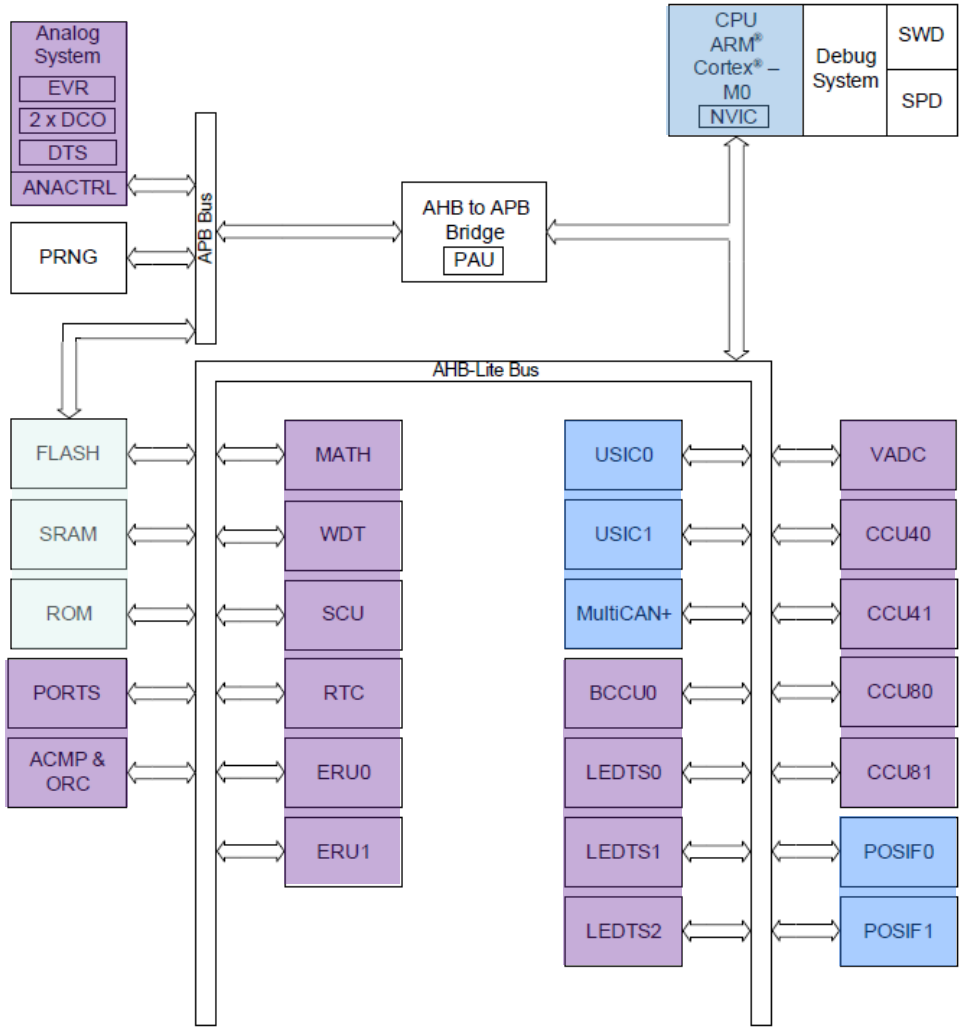
Example: Infineon XMC1000 – Cortex M0

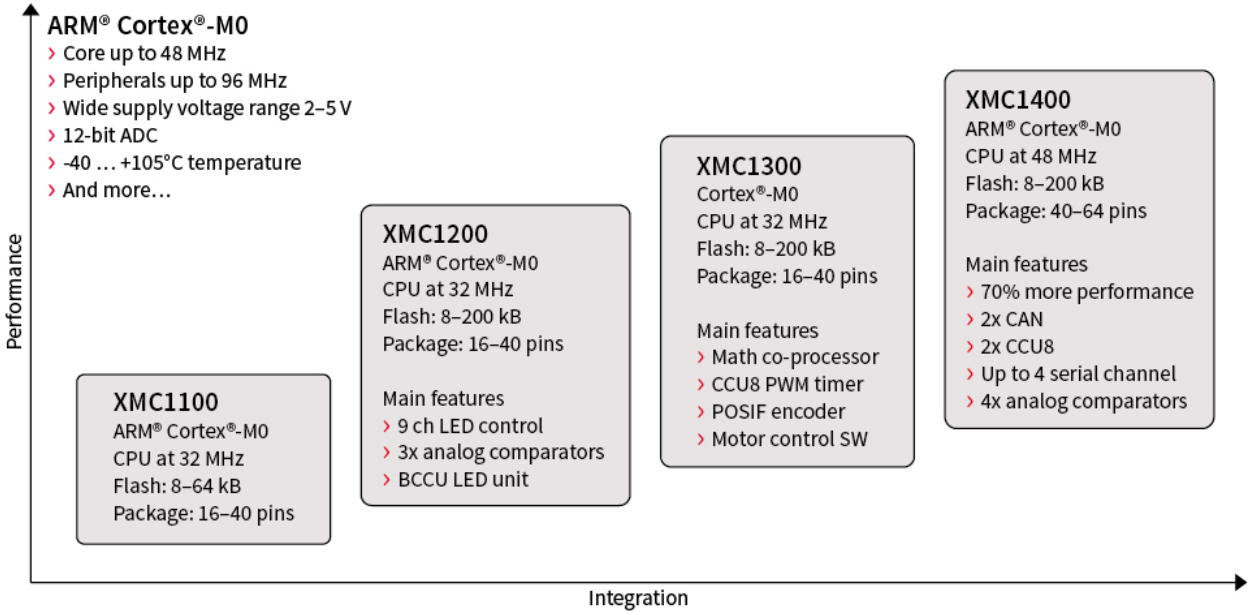
CPU

Memories

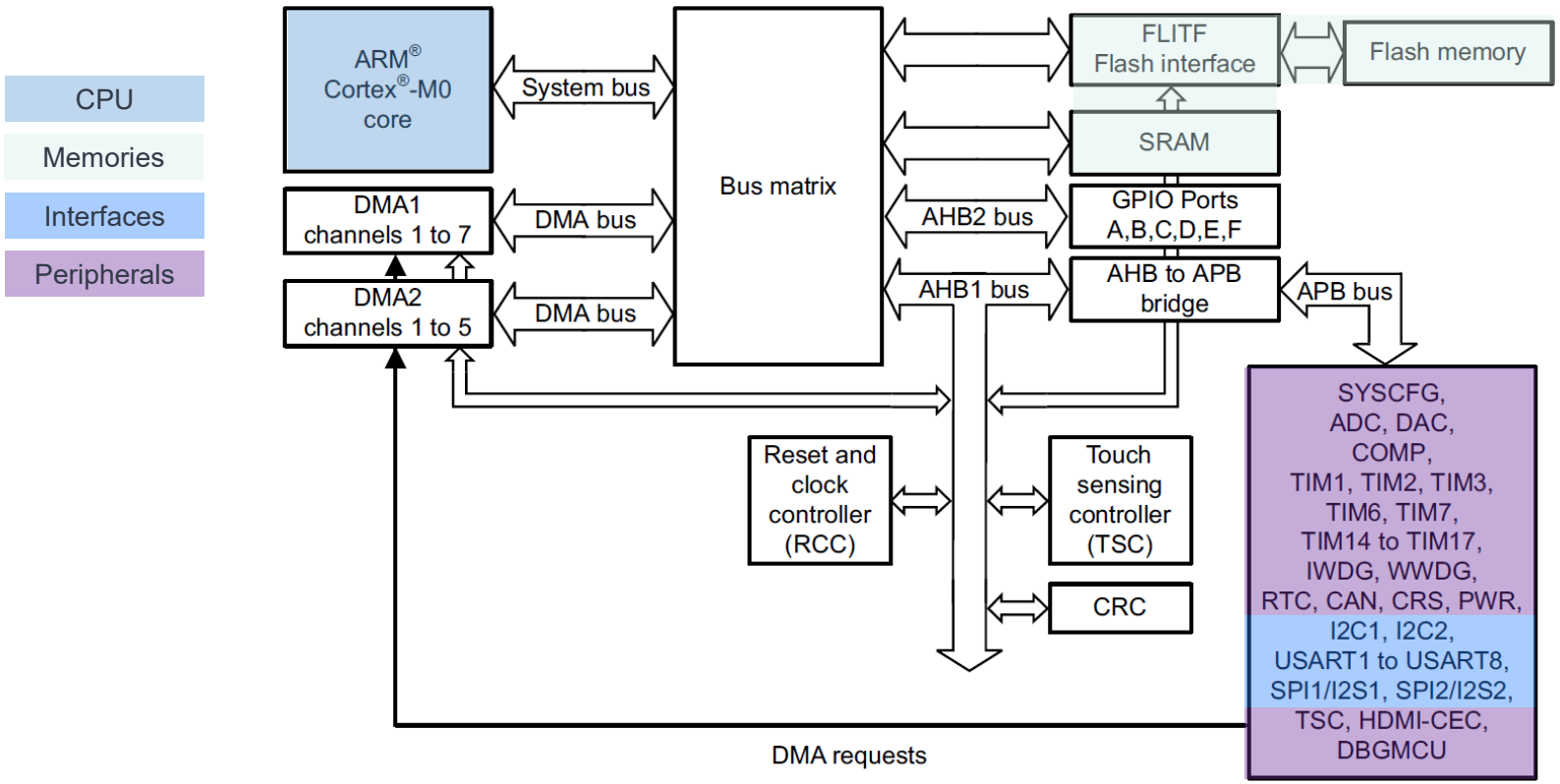
Interfaces

Peripherals



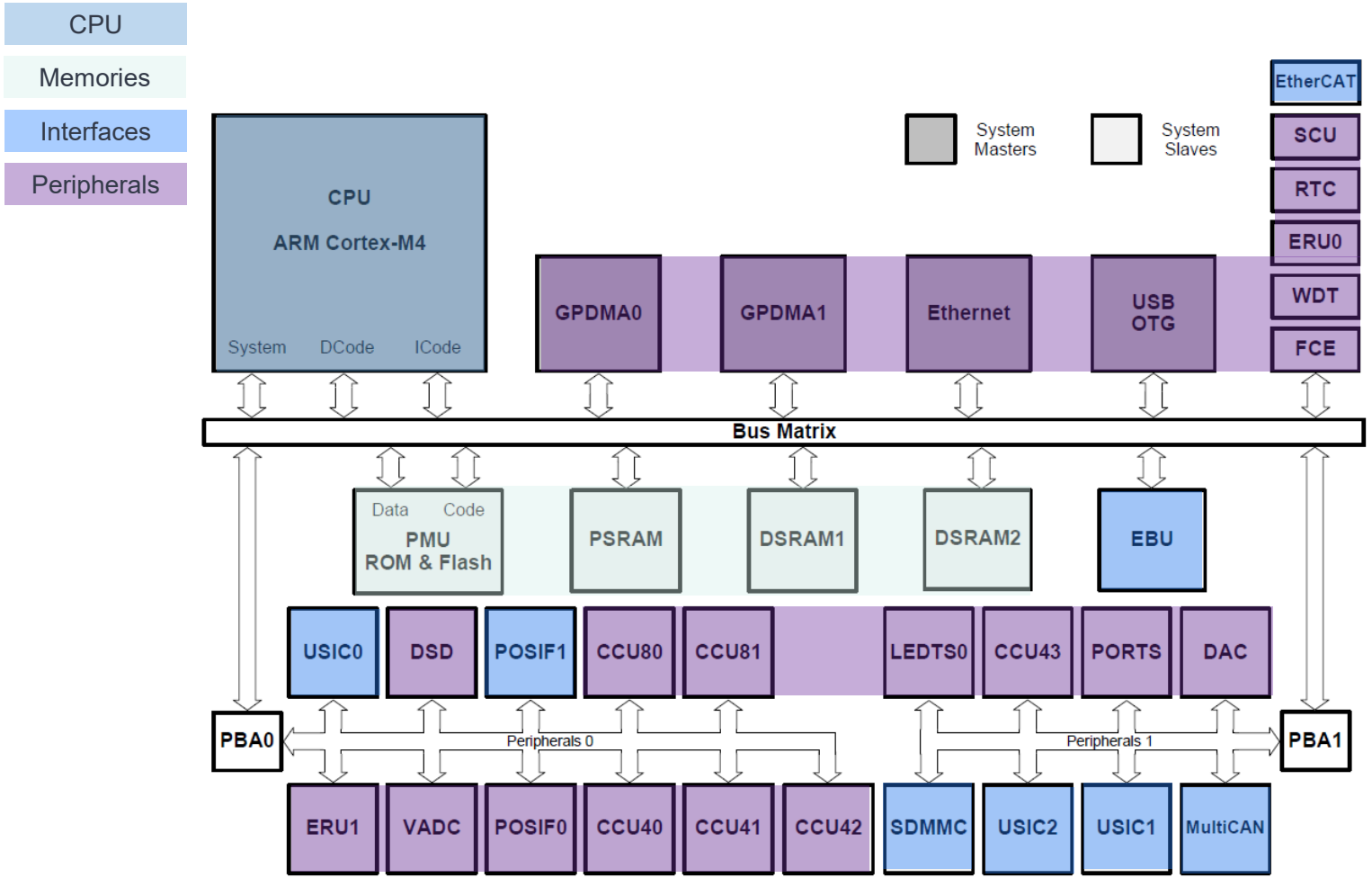


Example : STM STM32F0 – Cortex M0



- Devices in ST's Arm® Cortex®-M0-based STM32F0 Series deliver 32-bit performance while featuring the essentials of the STM32 family and are particularly suited for cost-sensitive applications. STM32F0 MCUs combine real-time performance, low-power operation, and the advanced architecture and peripherals of the STM32 platform.

Example: Infineon XMC4700 – Cortex M4



Microcontroller subsystems

Clock

Clocks

The clock is used to synchronize all subsystems, namely

- Core
- Memories
- Peripherals

Typically two clocks

- Main clock — Feed the computing; most common
  - Used for core, memories and peripherals
  - Typically 2 MHz to 144 MHz
- RTC clock — Real Time Clock, used to "count the time"
  - Used mainly for real time clock (timekeeping)
  - In some system used for low-power modes
  - Almost always 32,768 Hz

Clocks

The clock source

- Where the clock signal comes from

Internal oscillator

- Imprecise, drifts with temperature, significant process variability
- Frequency is multiplies/divided with a DPLL
- Often used with external crystal to improve stability

External oscillator

- More accurate than internal oscillators
  - Independent from microcontroller
  - Frequency is multiplied/divided with a DPLL
  - Often used with external crystal

Clocks

Several internal secondary sources

- The main clock is multiplied and dived by a dedicated clock generation and clock distribution logic
- Different clock feed different “domains”
- The clock of each domain can be controlled (regulated, gated) individually

Clock domain

- A set of elements fed by the same clock
  - Core domain: CPU, RAM, FLASH, Timers
  - Analog domain: ADC, DAC, Comparators
  - Bus domain: SPI, UART, I2C
  - Specific domains: Ethernet, USB
- Configurable interconnects

Clocks

Flexible clock distribution is crucial

- Different application require different clock management schemes

Low drift over time and temperature

- Specific peripherals such as watchdog and timers
- Typical values are in the range of 1%

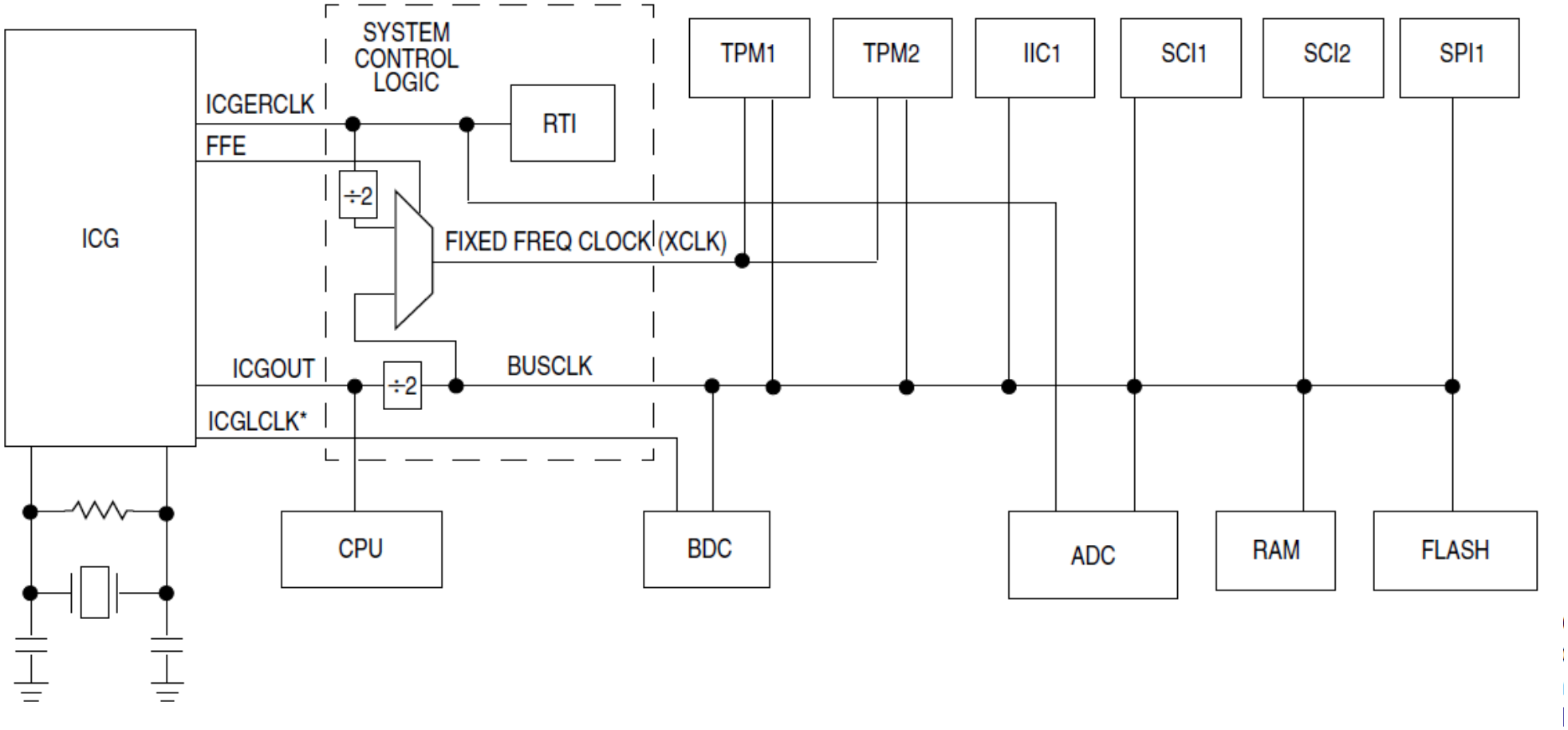
Accuracy, for time measurement

- Must be in the range of 5 to 50 ppm
- Higher accuracies achieved with specific devices or periodic resynchronization

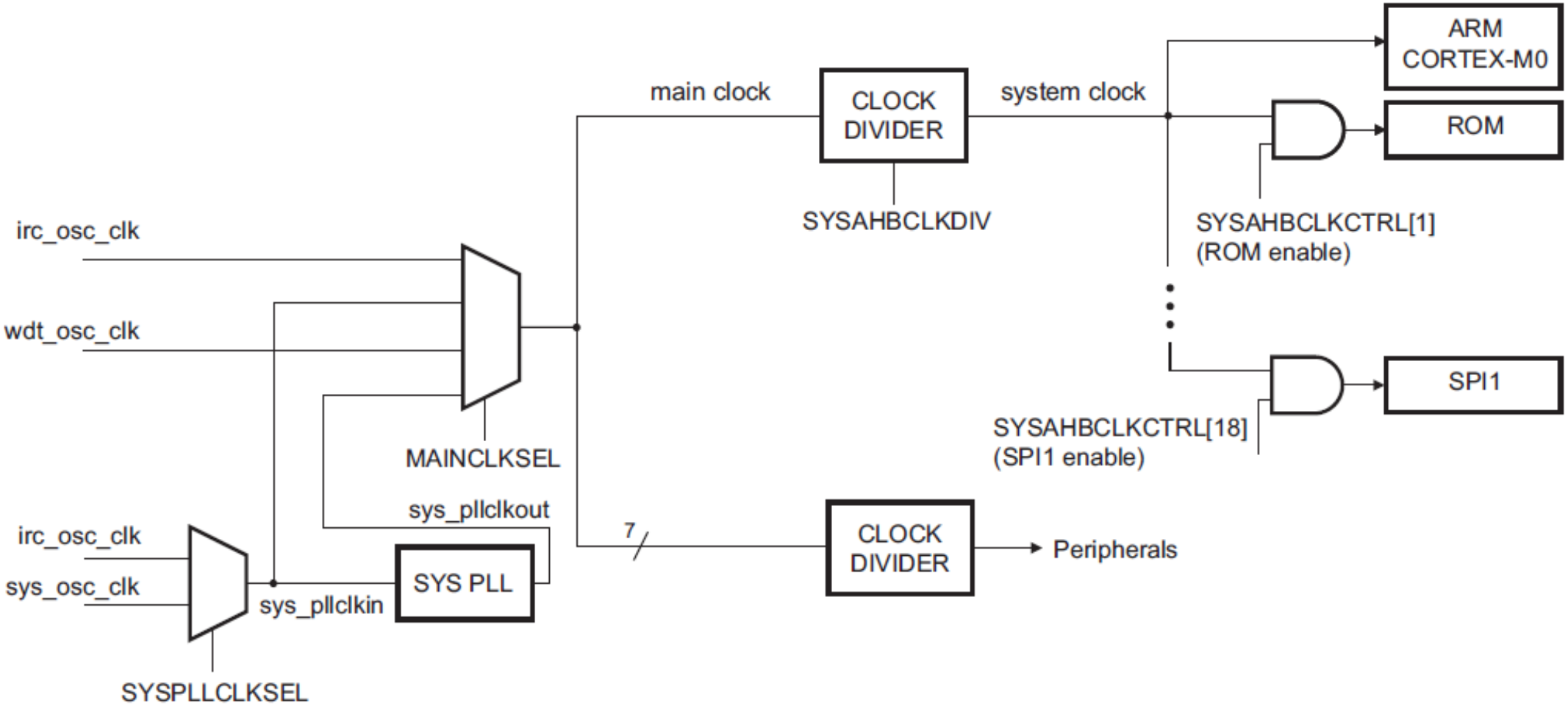
Low-power

- Frequency scaling and clock gating are used to save power during idle times

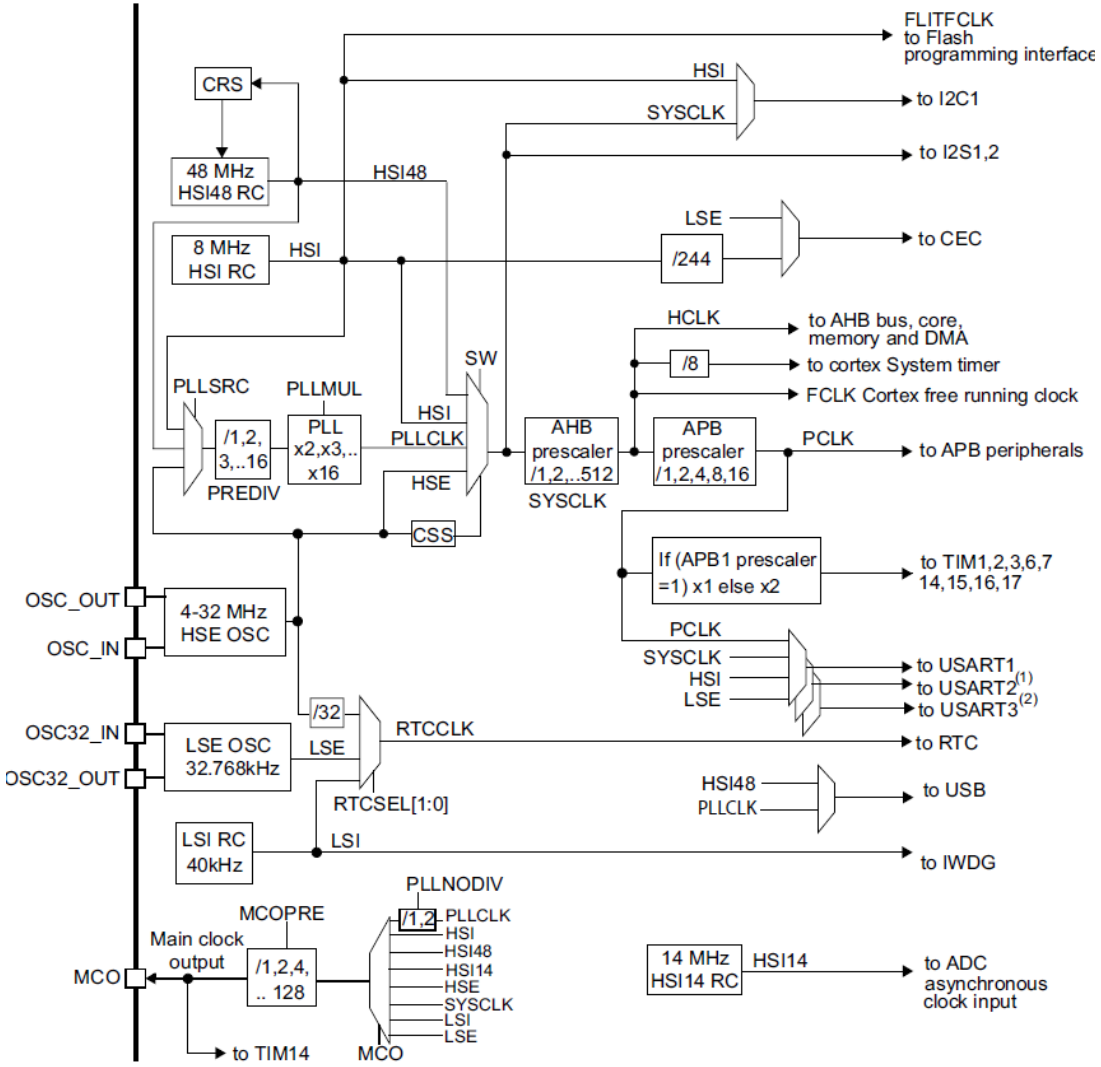
Clock distribution: NXP HCS08



Clock distribution: NXP LPC111x



Clock distribution: STM STM32F0



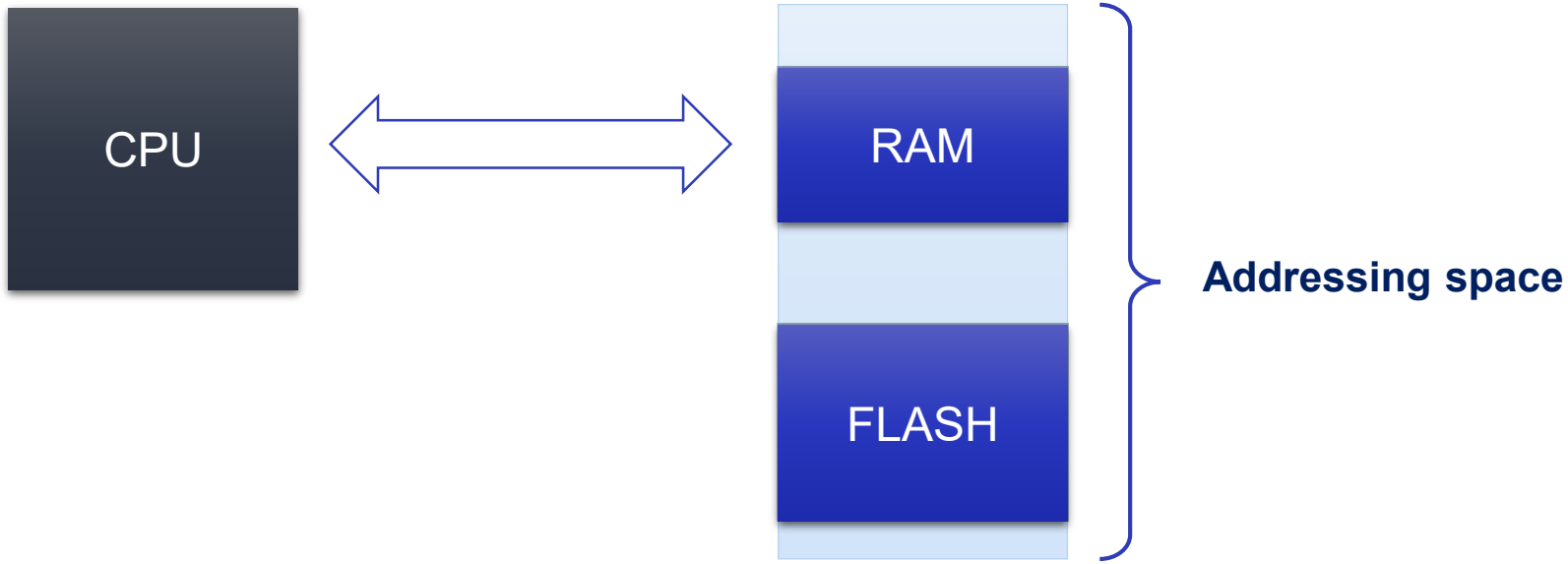
Microcontroller subsystems

Memory

Memory

Simple memory architecture

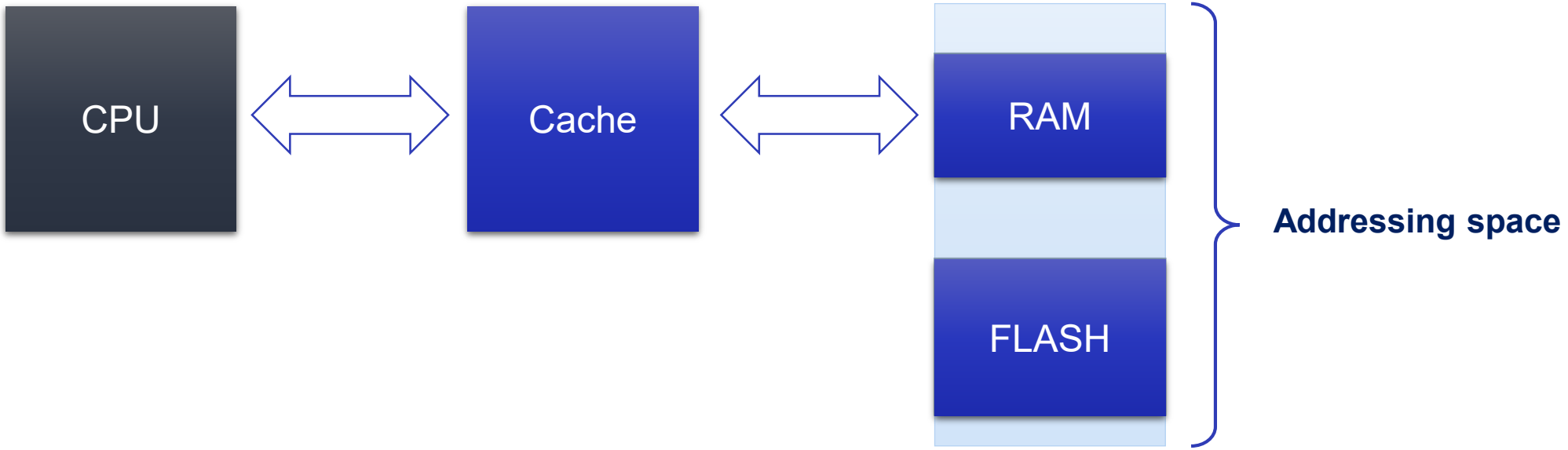
- Single addressing space
- In most cases 16 or 32 bit addressing
- Different memories (RAM, Flash) maps to different areas of same addressing space
- No need for cache



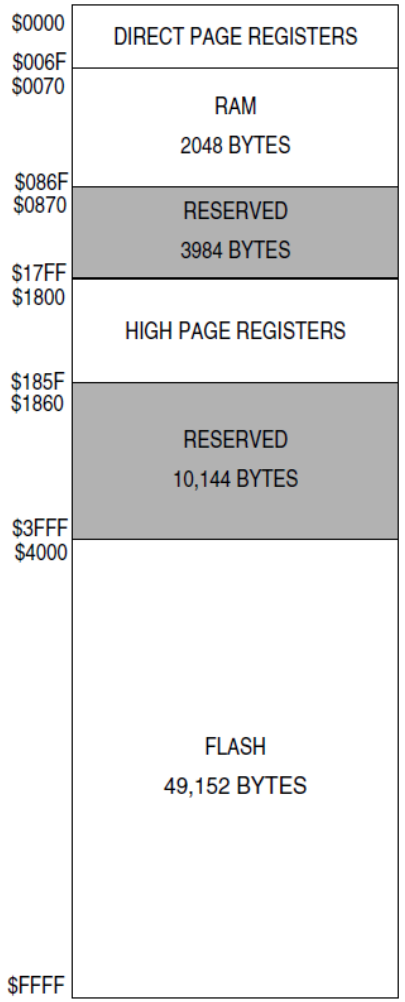
Memory

Simple memory architecture

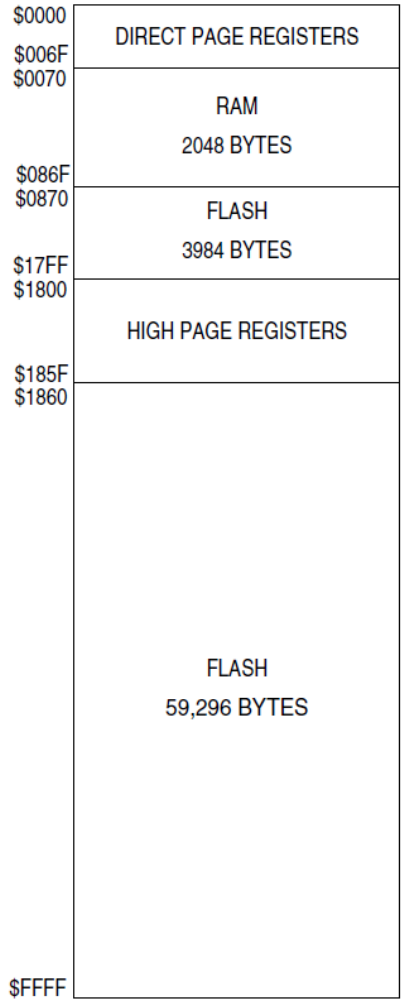
- Single addressing space
- In most cases 32 bit addressing
- Different memories (RAM, Flash) maps to different areas of same addressing space
- A small cache unified cache or a split-cache architecture may be used



Memory map: NXP HCS08

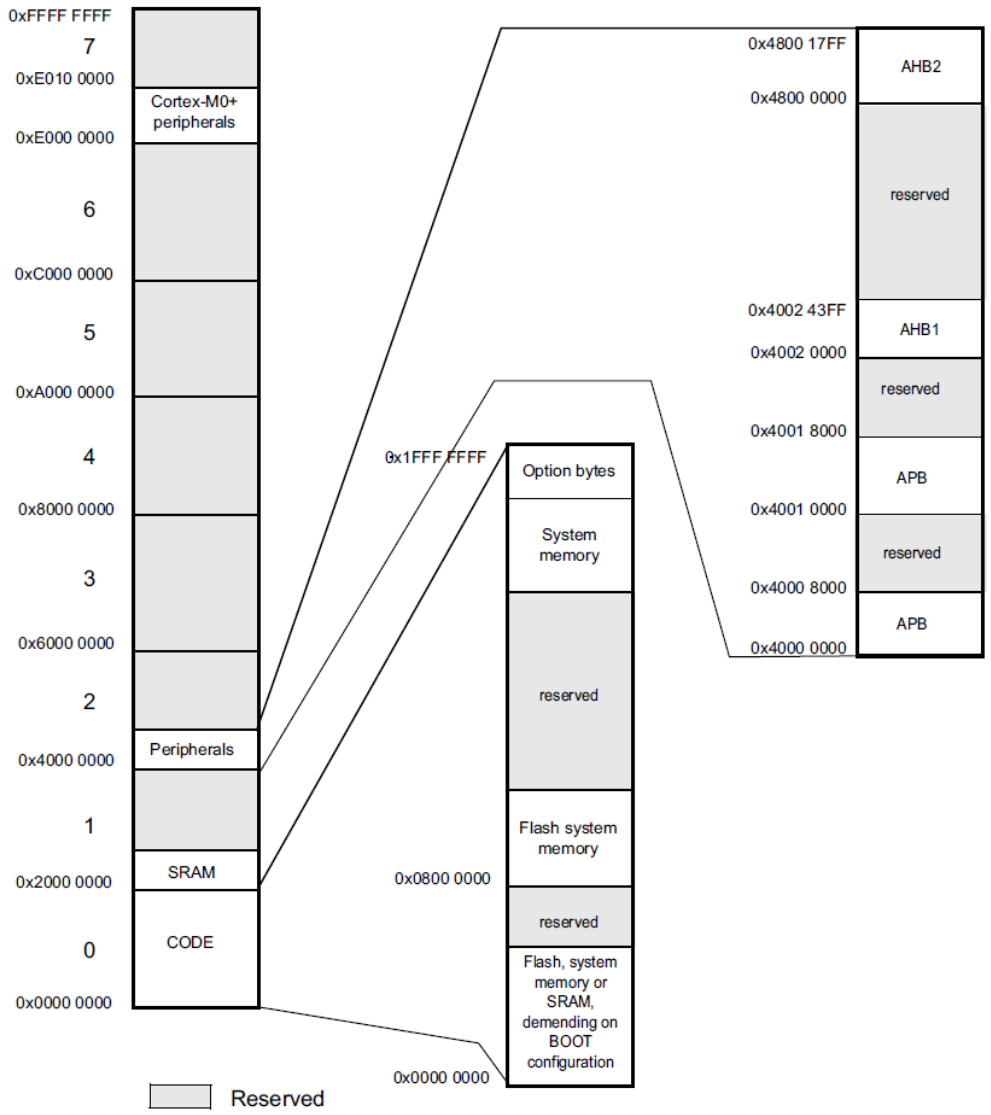


MC9S08AW48



MC9S08AW60

Memory map: STM STM32F0



Banked memory

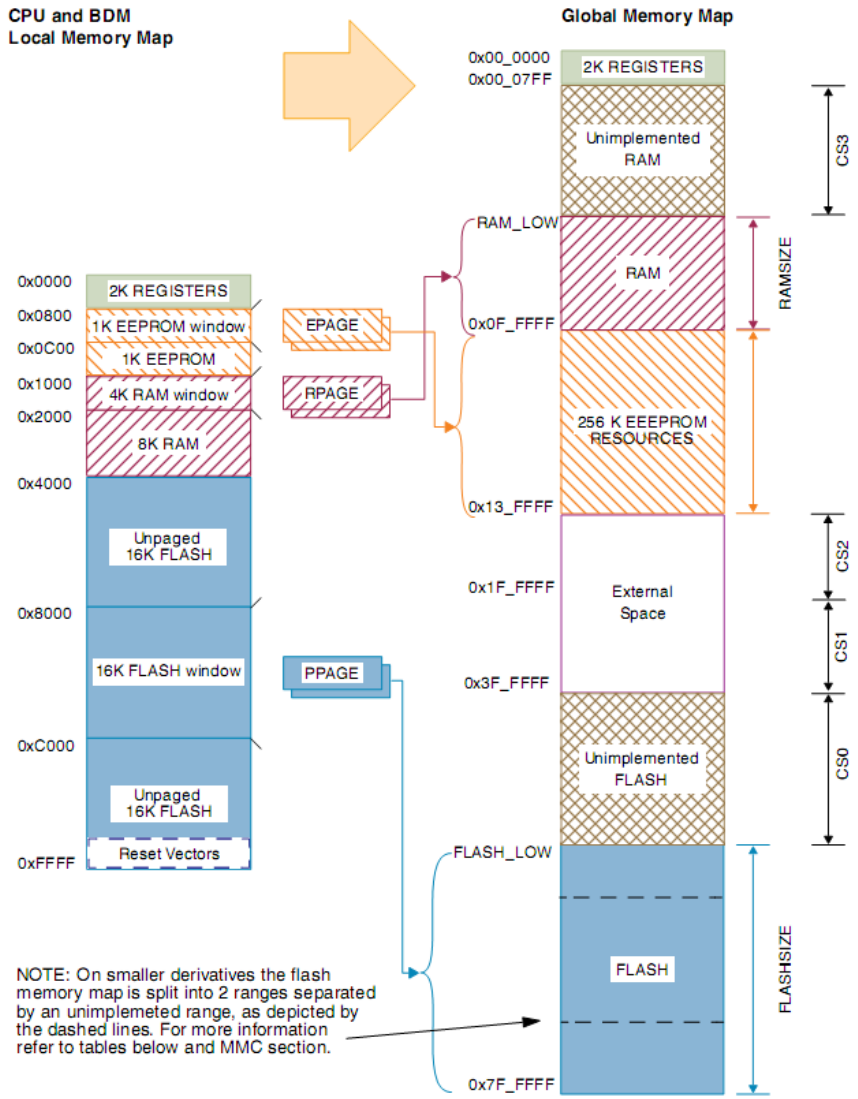
Required when the processor architecture limits the addressing space

- Examples
  - 8 bit processor with 8KB of memory
  - 16 bit processor with 1 MB of memory

Banked memory

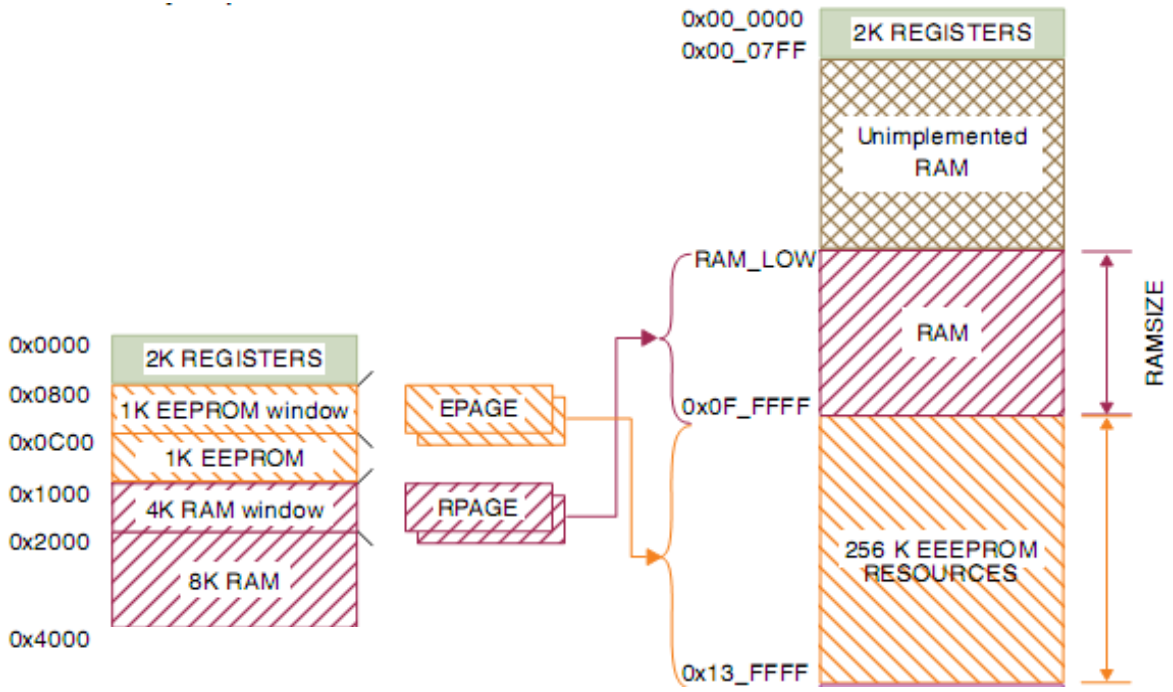
- Instruction set operates on 8 or 16 bits operands
- Distinction between local and global memory addresses
  - Local address target a small subset of the memory
  - Global address target the entire available memory
- Global addresses require address longer than the core registers
  - Specialized “segment” registers are used to extend the address bit-width

Banked memory: HCS12X



Banked memory: HCS12X

Direct access to “unpaged” memory segmants

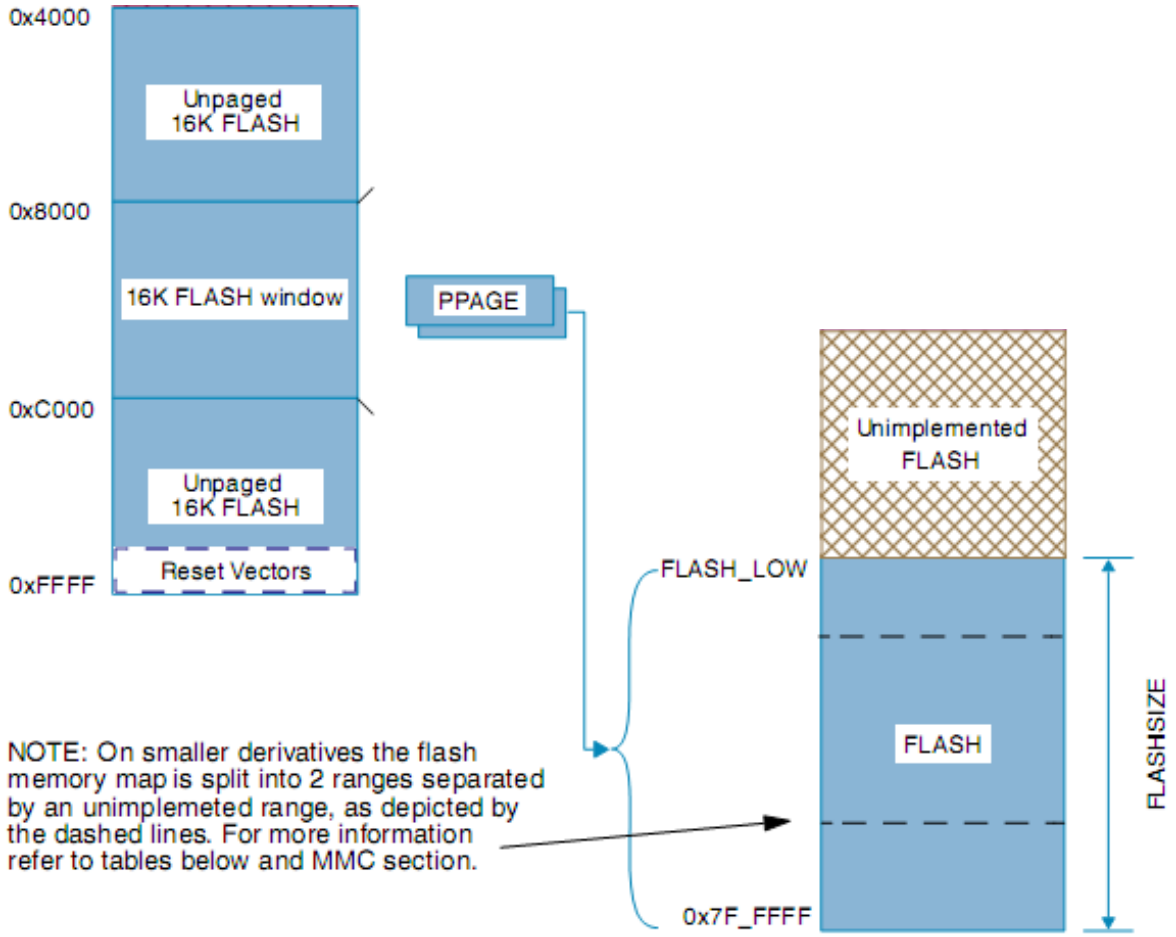


Local Memory Map

Global Memory Map

Banked memory: HCS12X

Accessing paged memory require using segmented register (PPAGE)



Microcontroller subsystems

GPIO

GPIO

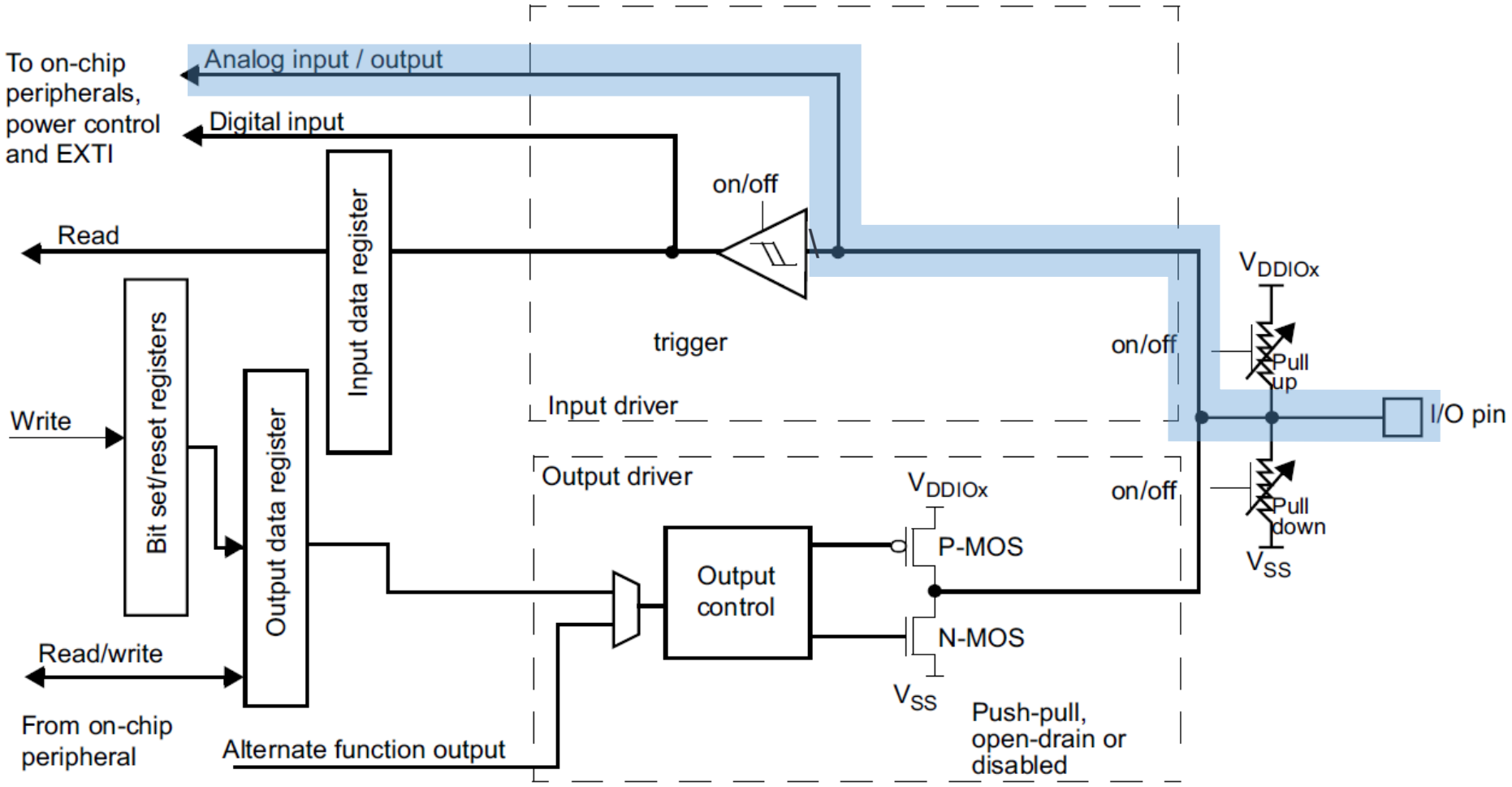
General Pupose I/O

- A GPIO correspond to a “pin”
- Each pin can serve different “functions”, not only GPIO
  - E.g. Communication interfaces
- GPIOs are usually organized in groups, called “ports”
  - All pins of a port are sampled at the same time

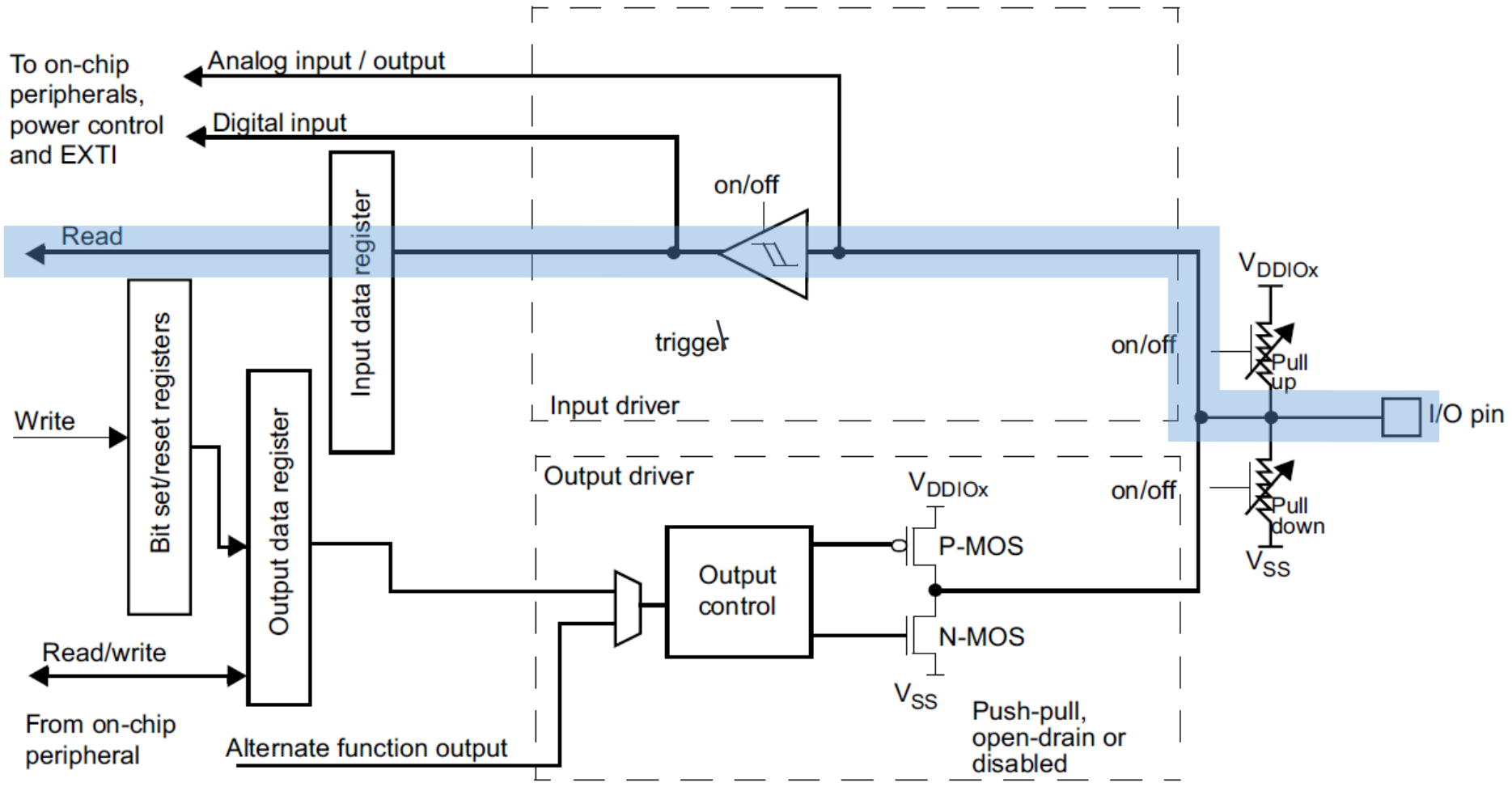
Four main functions

- Digital input
- Digital output
- Analog input
- Analog output

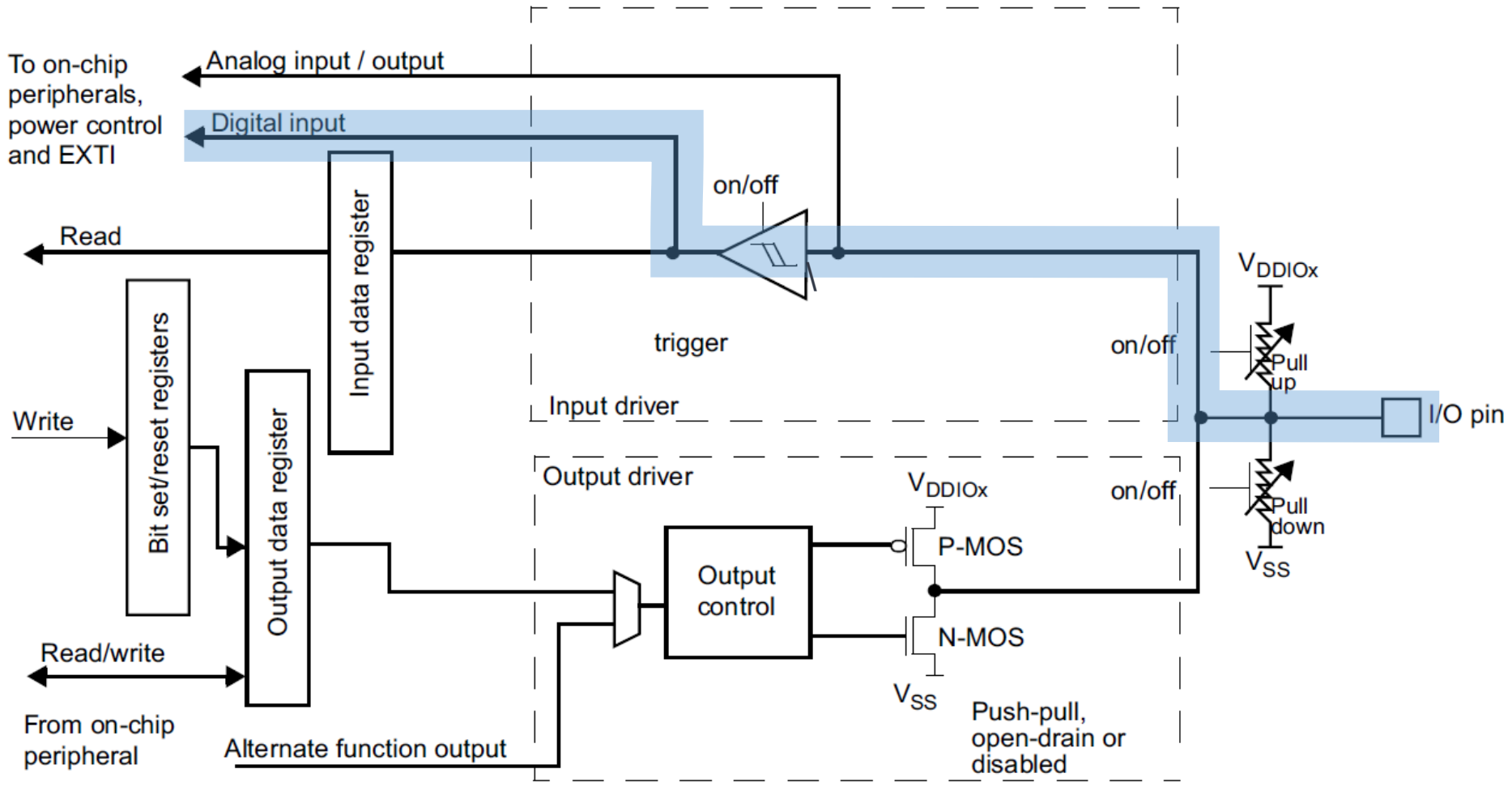
GPIO: Analog Input



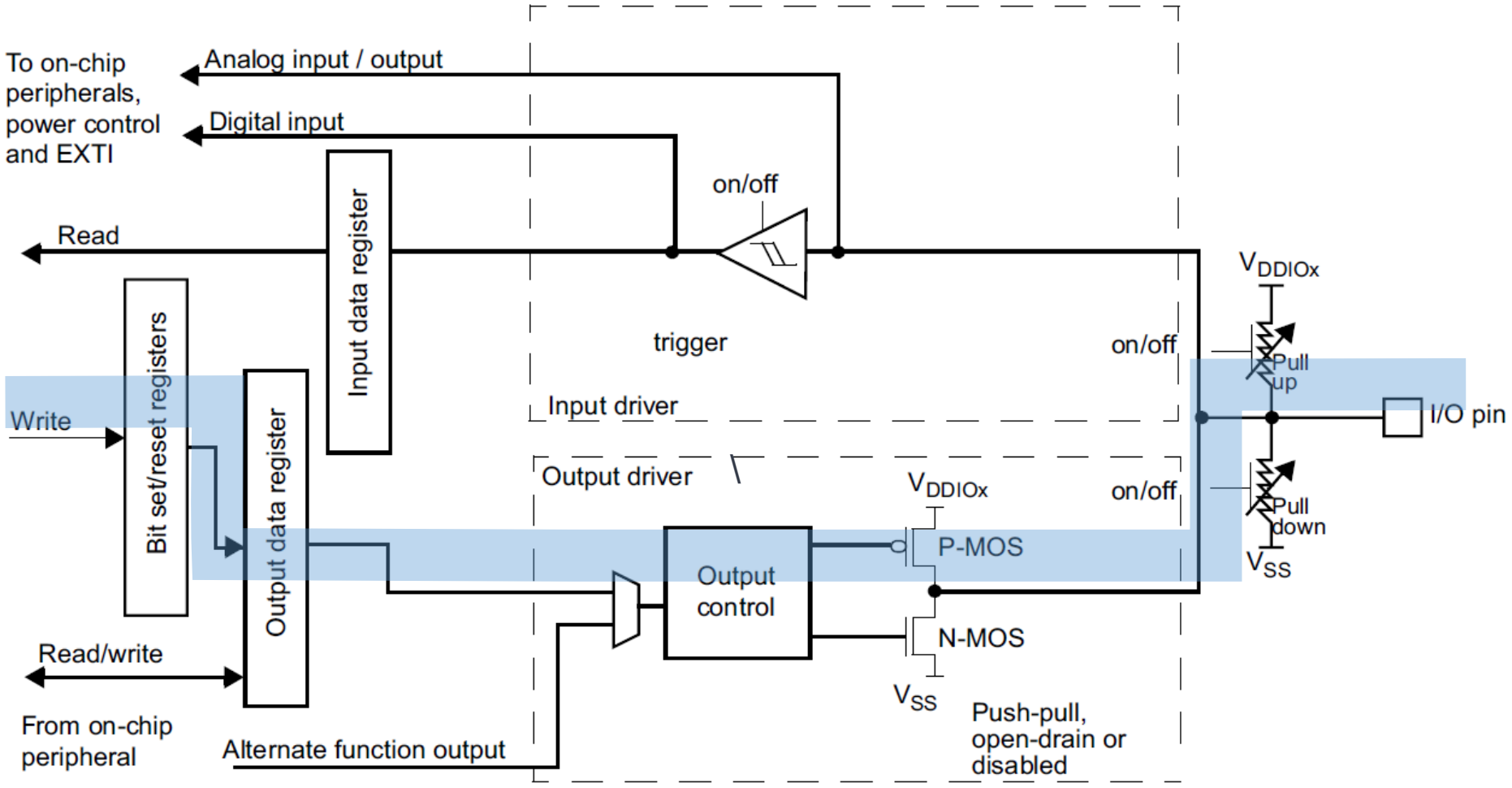
GPIO: Digital Input



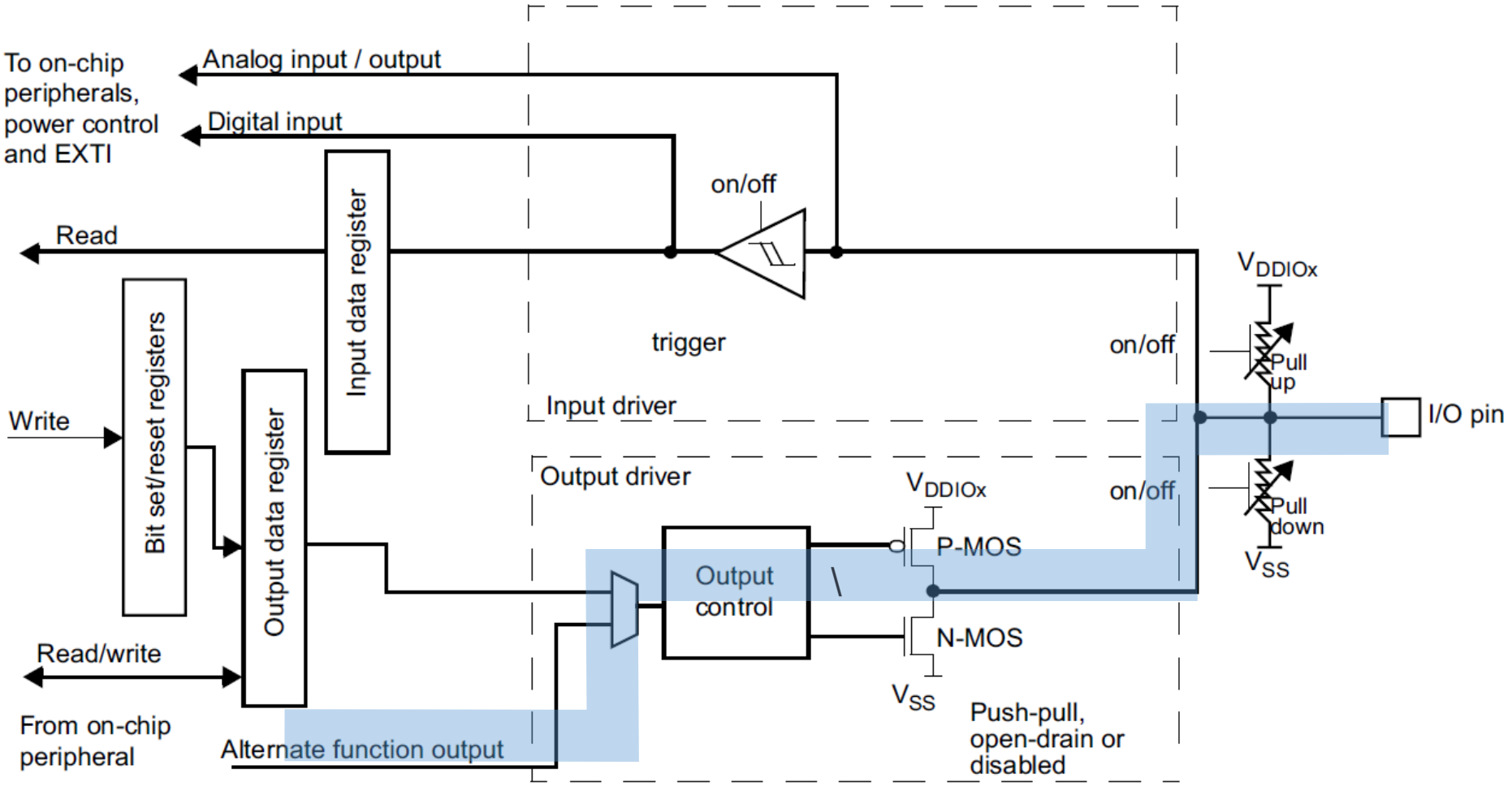
GPIO: Digital input to alternate function



GPIO: Digital Output



GPIO: Digital output from alternate function



GPIO: Functionalities overview

DDR: Data Direction Register

- Defines the pin direction

PDR: Port Data Register

- Data on the pin, if used as digital

PPSR: Port Pin State Register

- Reflects the current state of the port pins

EPCR: Edge Port Control Registers

- To configure pins for level detection and rising and/or falling edge detection
- To enable/disable interrupt related to specific pin
- To enable/disable internal pin pull up/down.

EPSR: Edge Port Status Register

- Holds the status flags related to interrupt enabled pins

Microcontroller subsystems

Analog comparators

Comparators

Analog comparators

- Differential amplifiers
- Configured to saturates the output to Vss or to Vdd

Interface

- Two analog inputs A and B
- A digital output Y

Behaviour

- When the voltage at A is higher than the voltage at B, then the output Y is a logic ‘1’ otherwise it is a logic ‘0’
- Through polarity selection, the behavior can be inverted

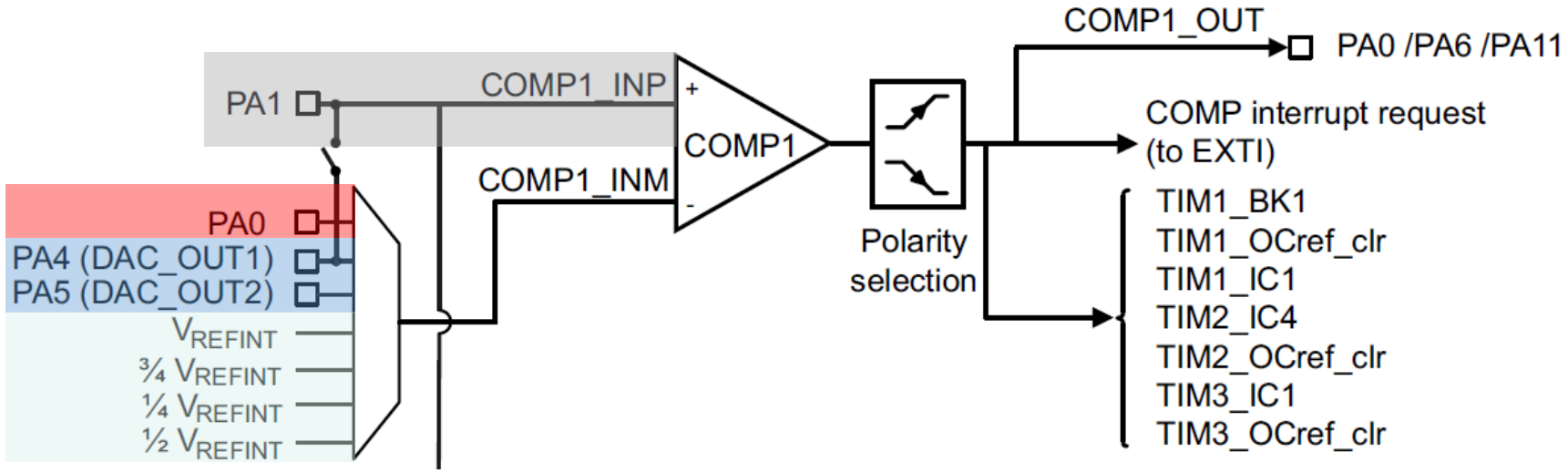
Comparators

First input

- Always an external signal (gray)

Second input

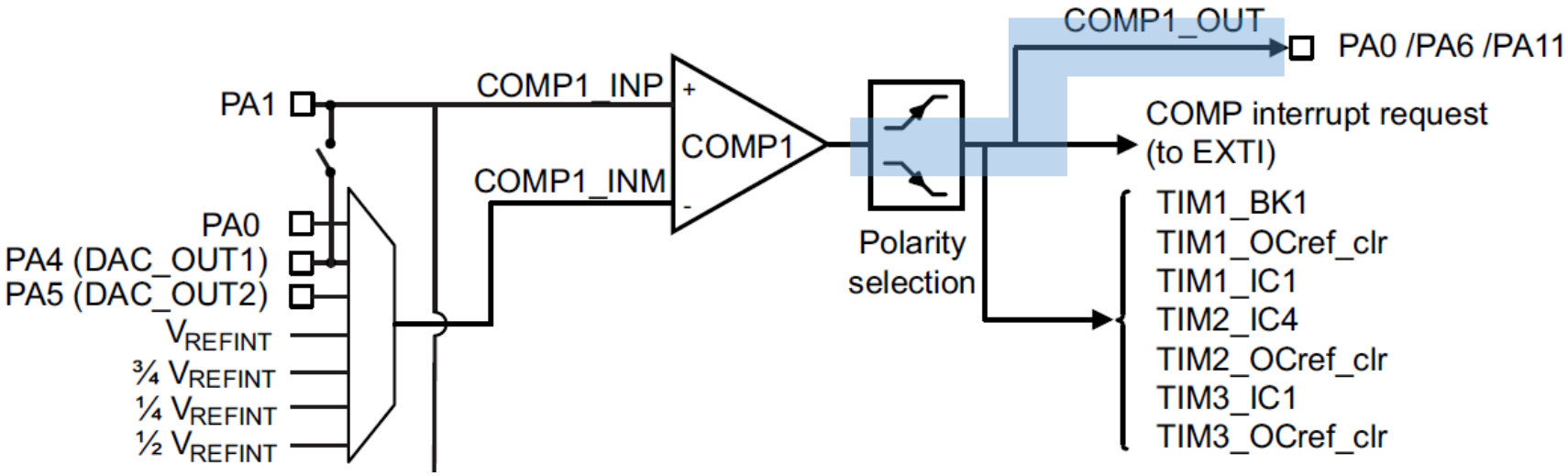
- External signal (red)
- Fixed internal voltage reference (green)
- Internal voltage generated by software via a DAC (blue)



Comparators

Output: Polling

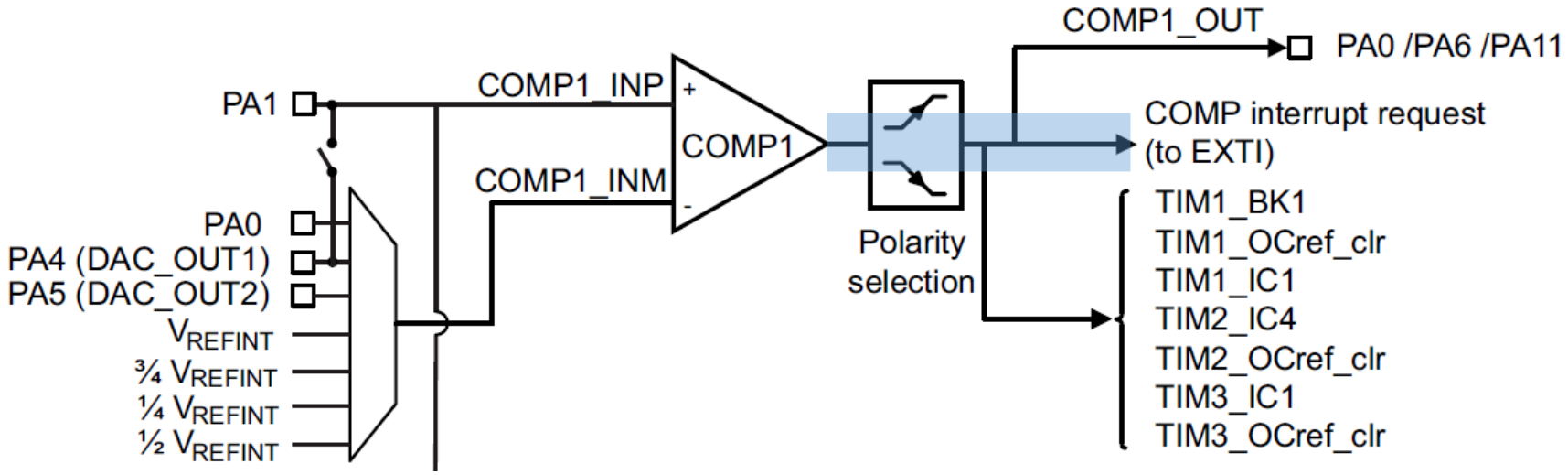
- Software reads the result of the comparison periodically
- Software performs “slow” operations based on the value



Comparators

Output: Interrupt

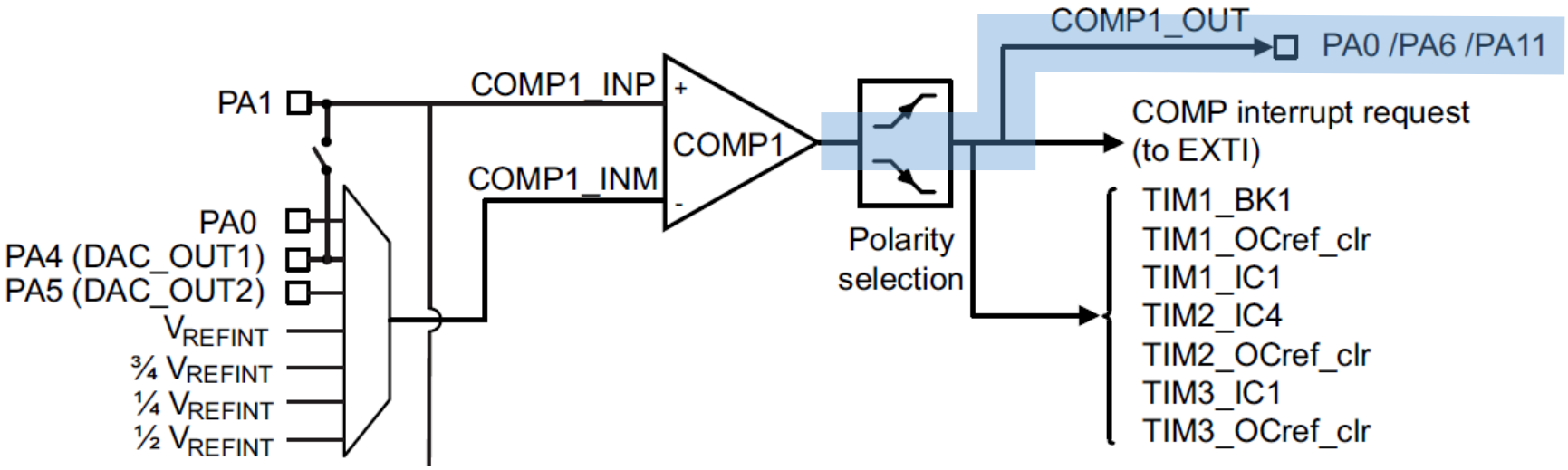
- The output of the comparator is to the interrupt controller
- An interrupt service routine is executed
- This configuration is used to react rapidly to “rare” events



Comparators

Output: Hardware

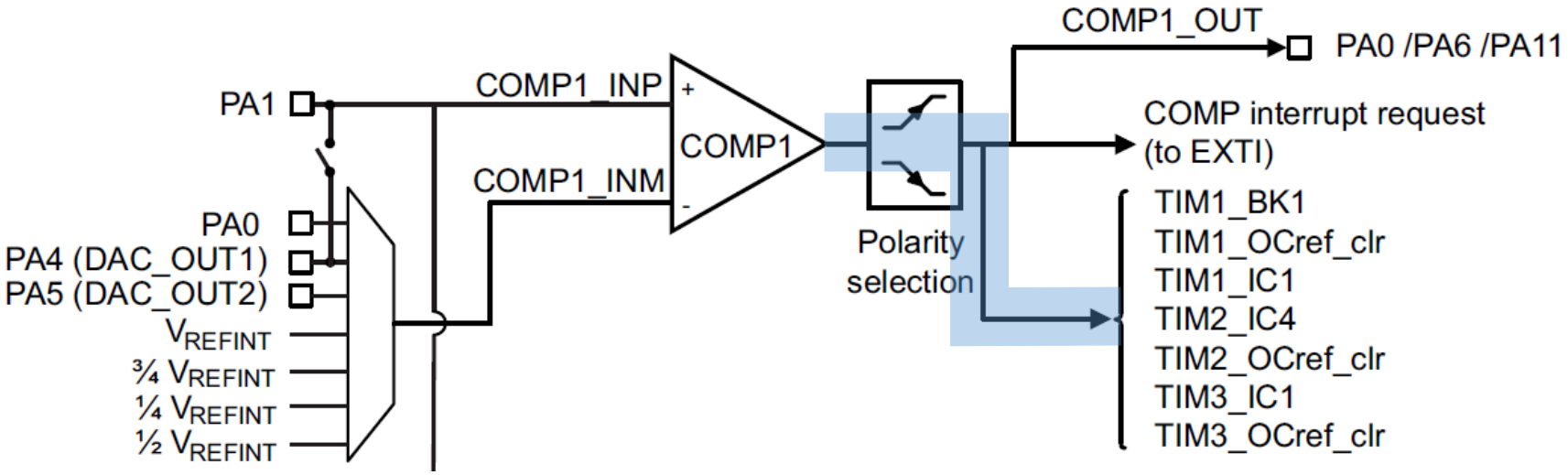
- The output of the comparator is fed directly to an output pin
- The software is not involved
- This configurations “squares” the input waveform



Comparators

Output: Hardware count / capture

- The output is fed to a control input of a timer
- Count: each time the output change a front is counted
- Capture: measure time between two fronts



Microcontroller subsystems

Analog to Digital Converter

ADC

Analog to digital converter

- Encodes the input voltage signal into a numeric value

Sampling

- The input signal varies continuously over time
- The ADC “samples”, i.e. observes, the input signal at specific moments, often based on a periodic time-base

Quantization

- The input signal is a real-valued physical quantity
- The ADC rounds the real value to fixed discrete intervals depending on the power supply voltage

ADC

Accurate timing

- Achieved by means of an external crystal on the main clock source

Stable power supply

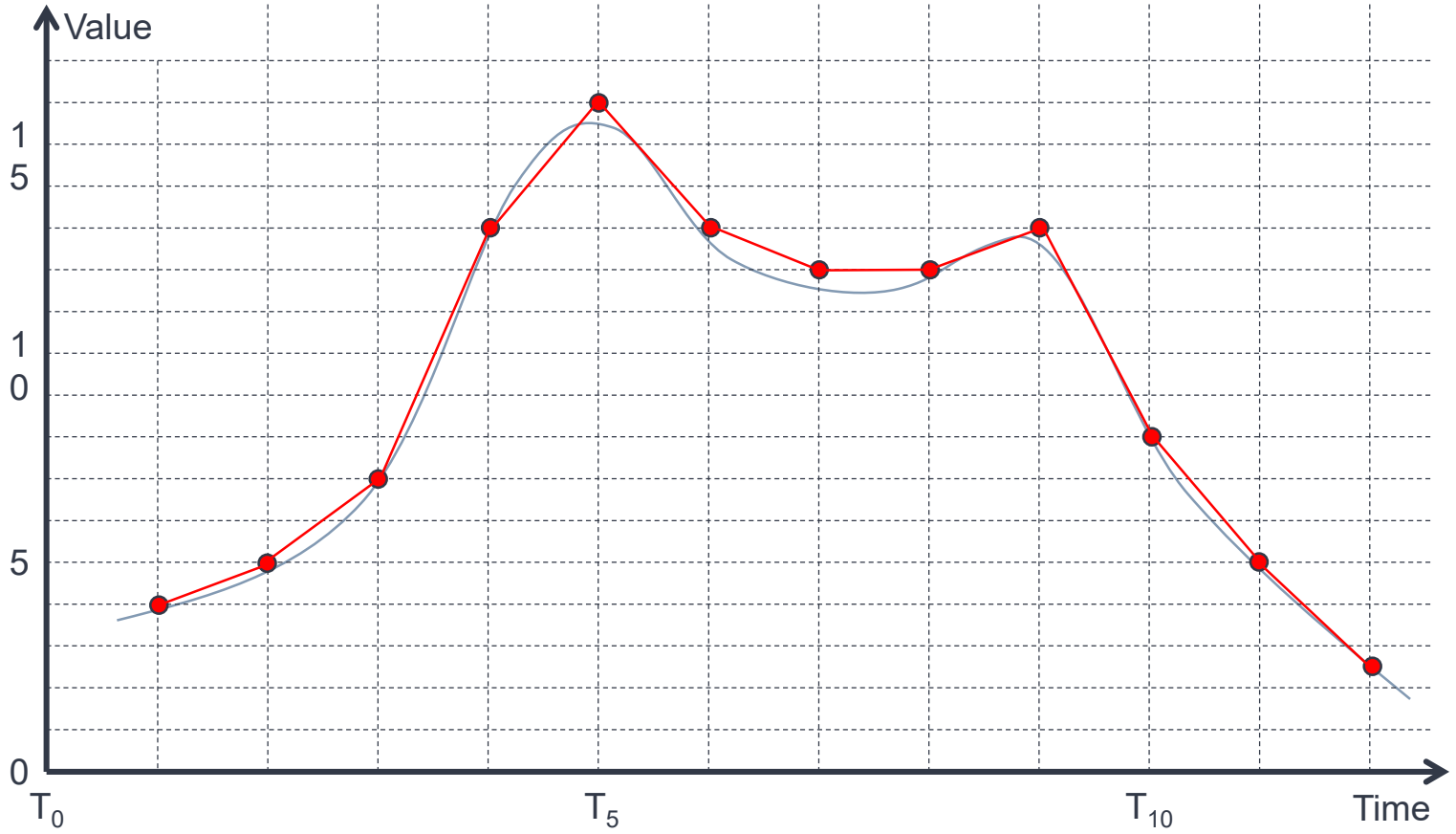
- If the power supply of the microcontroller, and thus of the ADC changes, while the input voltage is stable the effect is “seen” as change in the input voltage
- The solution is ratiometric measurement and requires a fixed absolute voltage reference, either internal or external

Input multiplexing

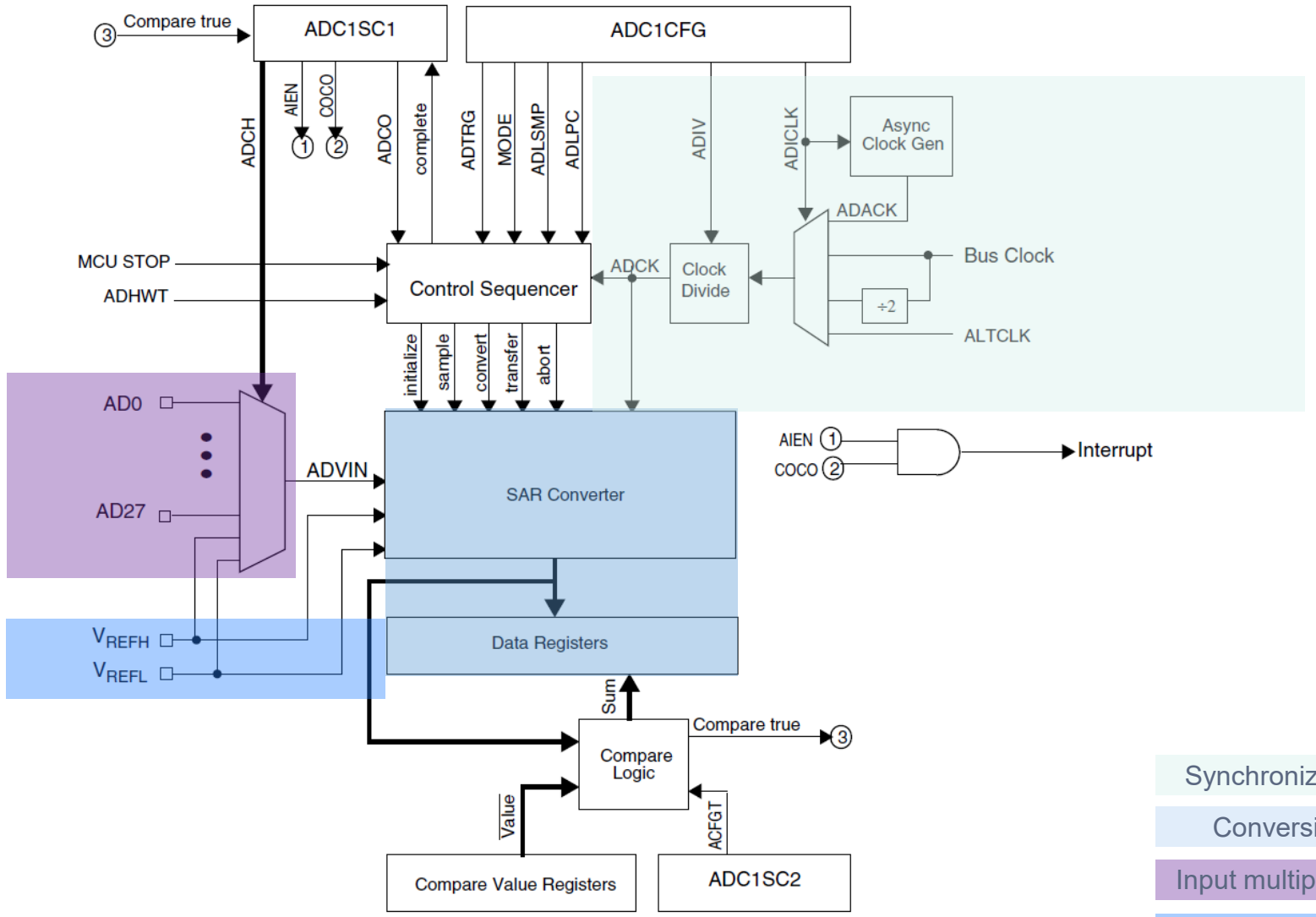
- Allows having multiple analog inputs that are cyclically converted thanks to time multiplexing

ADC

Quantization and sampling



ADC: NXP HCS08



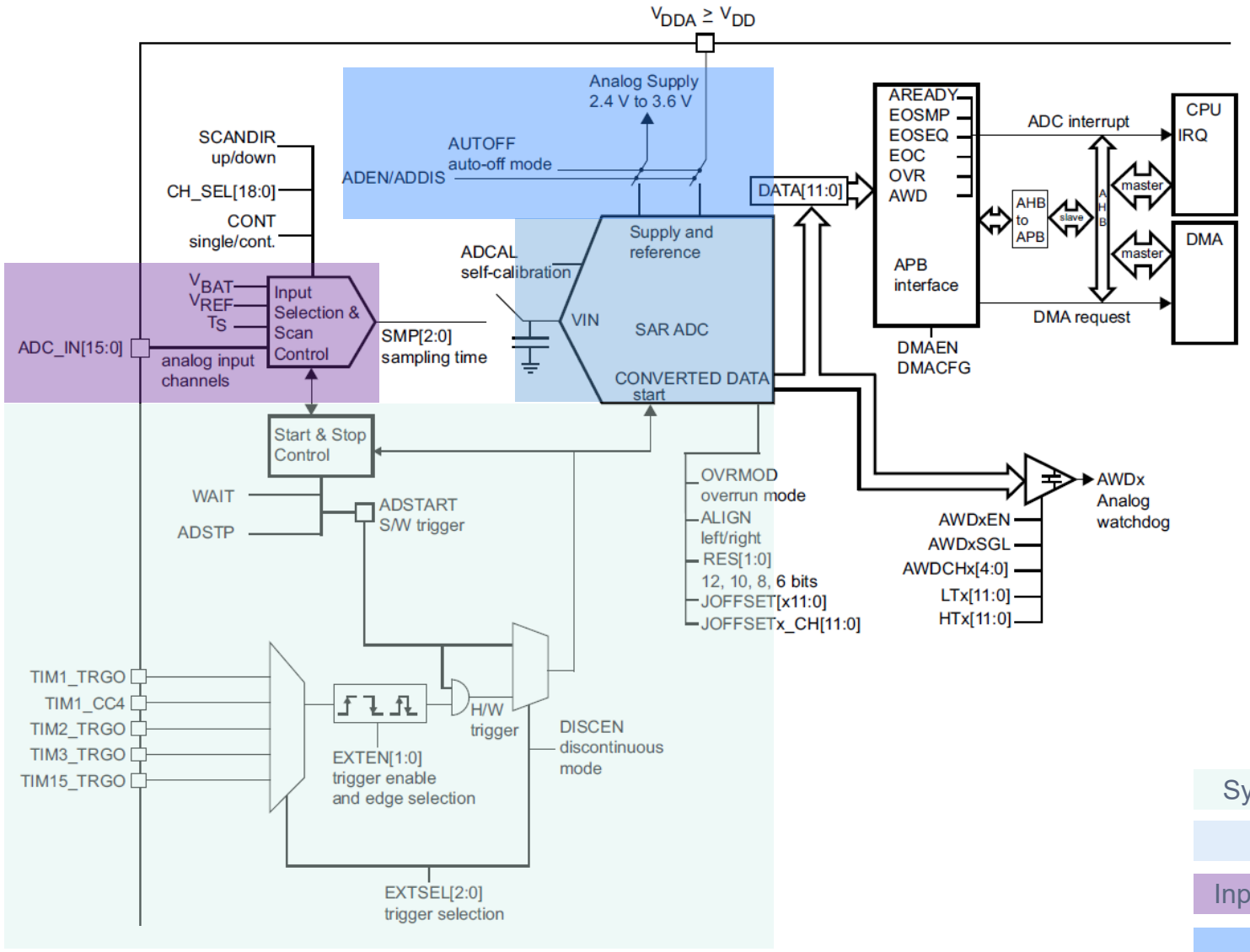
Synchronization

Conversion

Input multiplexing

Reference

ADC: STM STM32F0



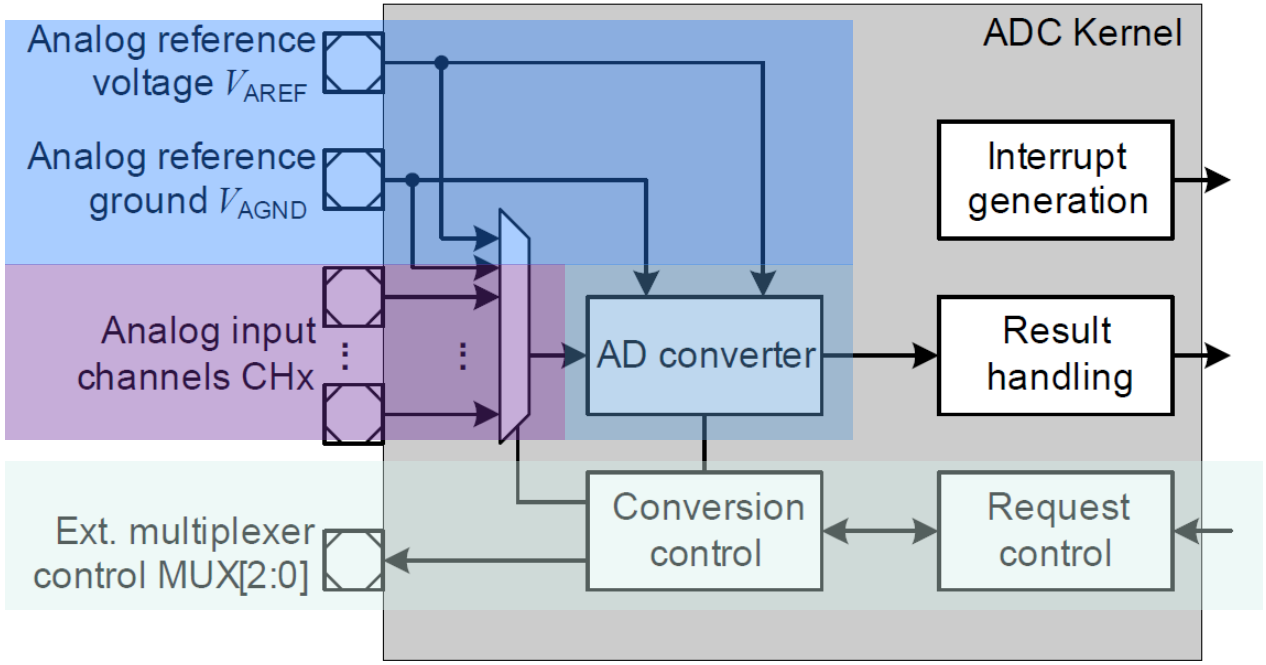
Synchronization

Conversion

Input multiplexing

Reference

ADC: Infineon XMC4700



Synchronization

Conversion

Input multiplexing

Reference

Microcontroller subsystems

Real Time Clock

RTC

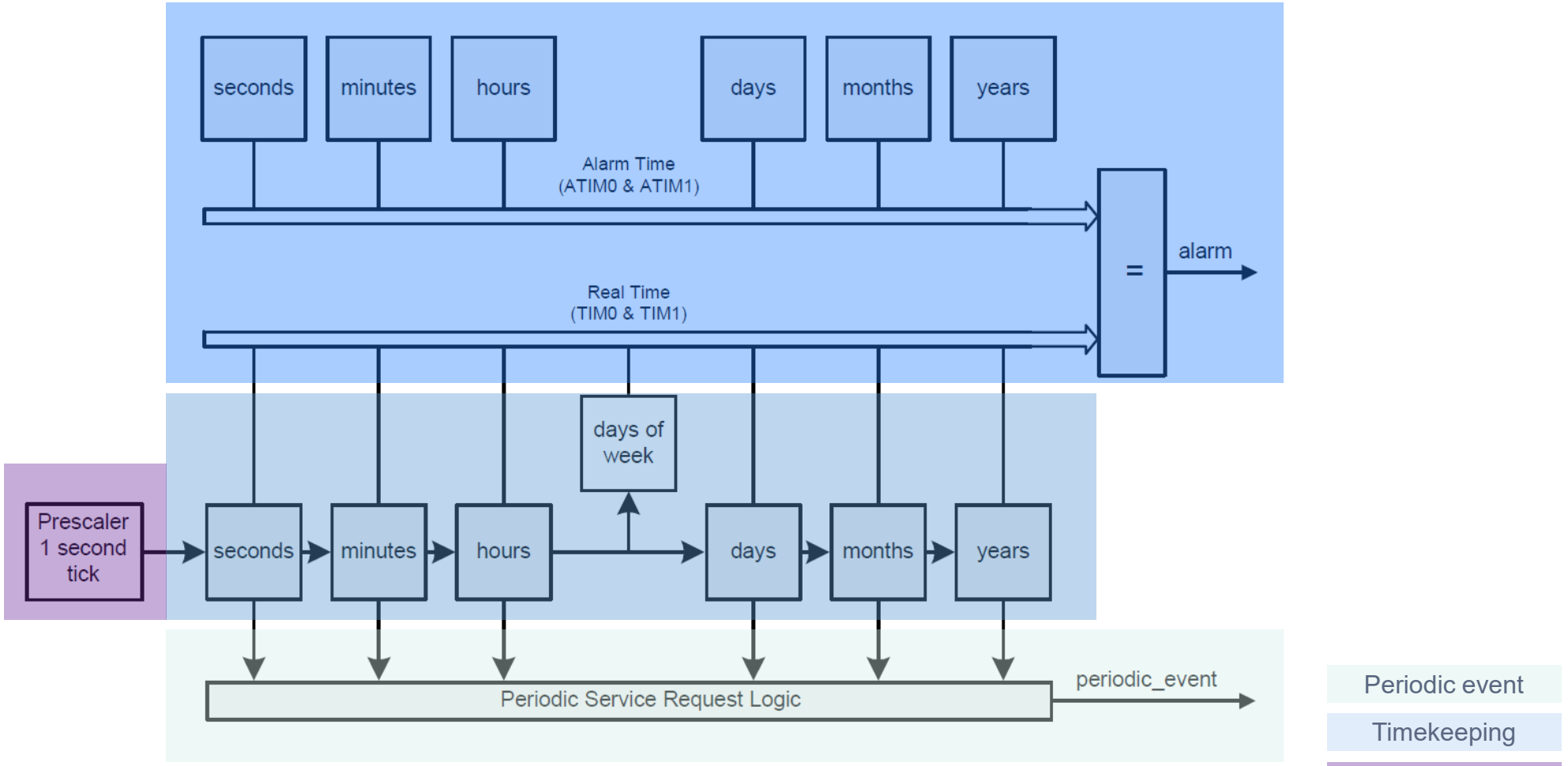
The real time clock has two main functions

- Calendar timekeeping
  - High accuracy
  - Over long periods of time
- Periodic alarm generators
  - Every second, minute, day, …
- Aperiodic event generations
  - At a specific moment in the future

The most critical aspect is drift due to clock inaccuracy

- A clock with a 5ppm crystal accumulates an error of 157s every year
- More accurate timing requires periodic resync with external sources e.g. GPS time

RTC: Infineon XMC4700



Periodic event

Timekeeping

Time base

Alarm

Timer

Very common and useful tool for embedded applications

Two main operating modes

- Counter
- Timer

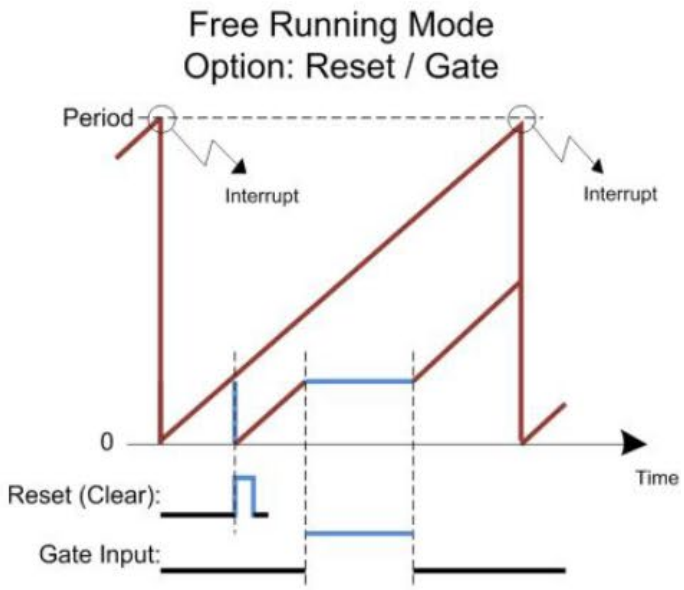
Counting the time to handle both "routine actions" and more complex opeations (examples: measuring time intervals, geenrating pwm signals...)

Many functionality

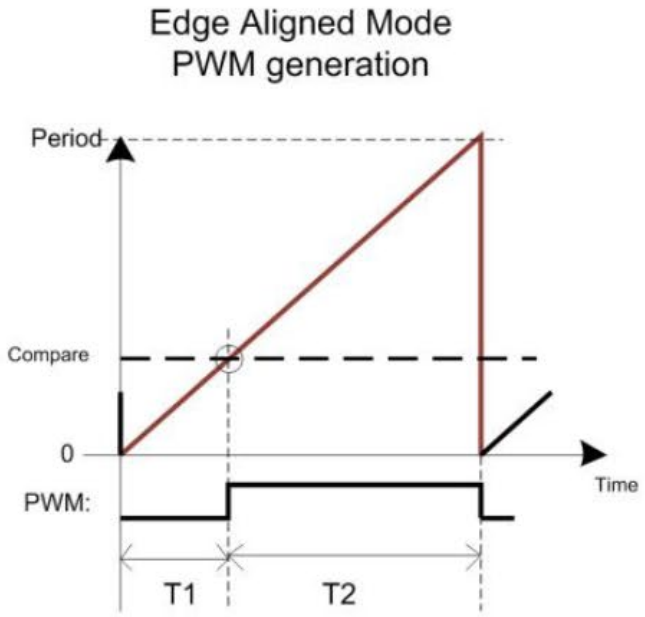
- Timer: Free running, for periodic interrupt generation
- One-shot: Generate delayed interrupt
- Compare: Generate PWM signals
- Capture: Measure time intervals
- Counter: Count external events

Timer

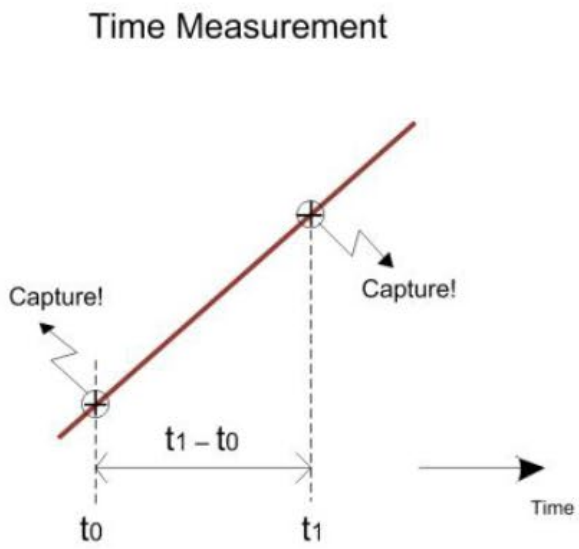
Timer



Compare

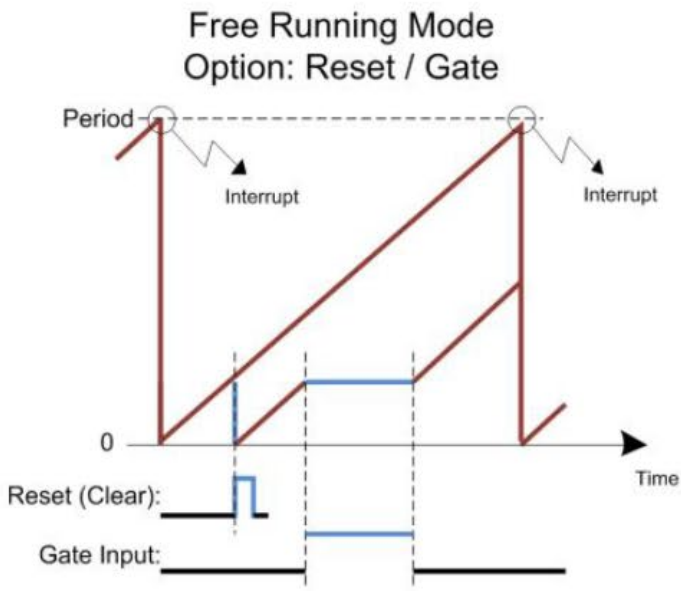


Capture

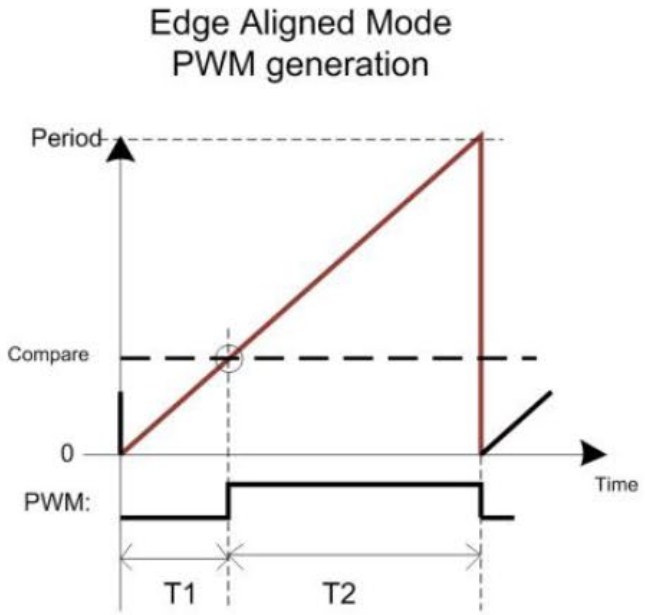


Timer

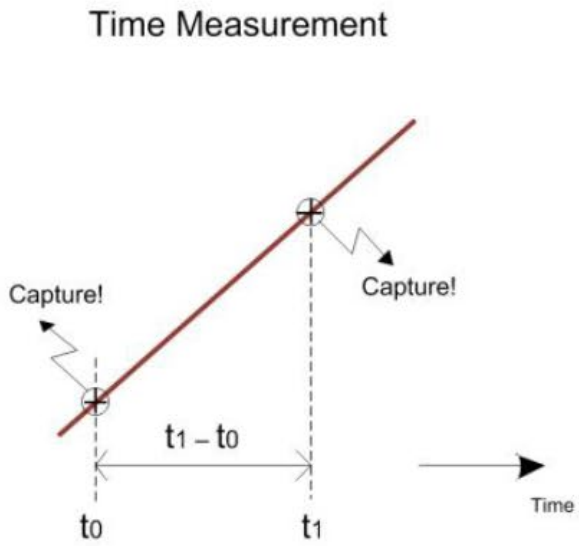
Counter



Compare



Single shot



Watchdog

Periodic timer

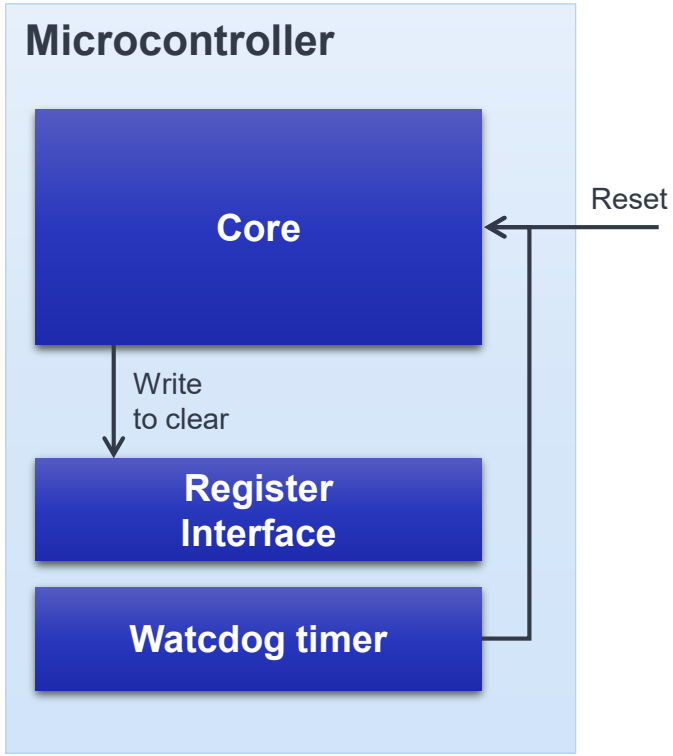
- Used to monitor the application execution evolution
- Configured once at boot to trigger an interrupt/reset at a fixed time interval
- Must be cleared before it expires to prevent triggering

Two types of operation

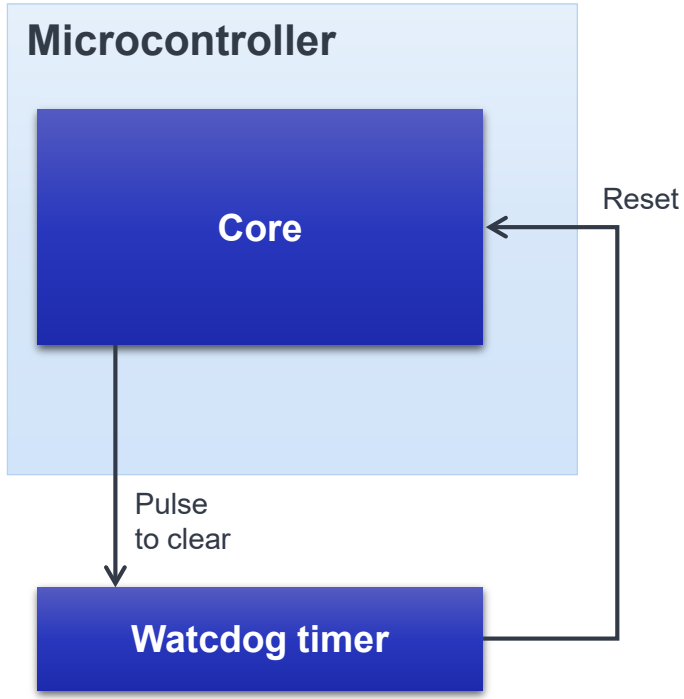
- Non-windowed
  - The watchdog must be cleared before a certain time
- Windowed
  - The watchdog must be cleared before a certain time and after another fixed time interval
- Clearing the watchdog too soon will trigger the interrupt/reset

Watchdog

Internal watchdog (COP)

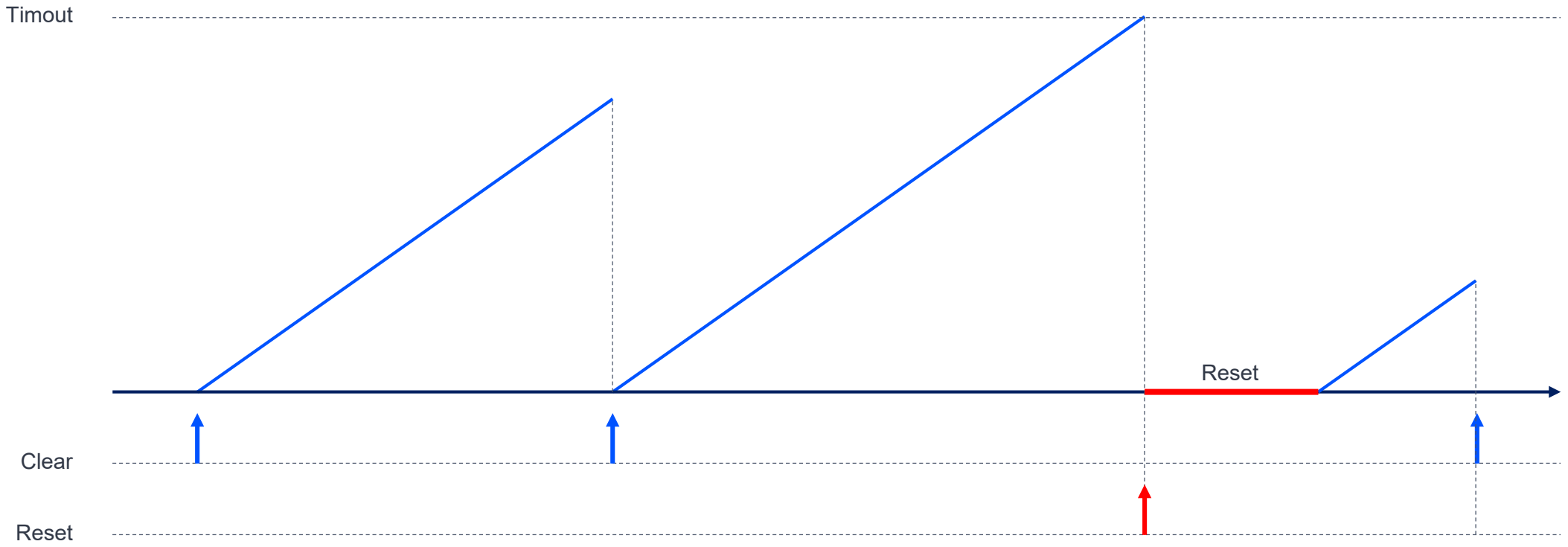


External watchdog



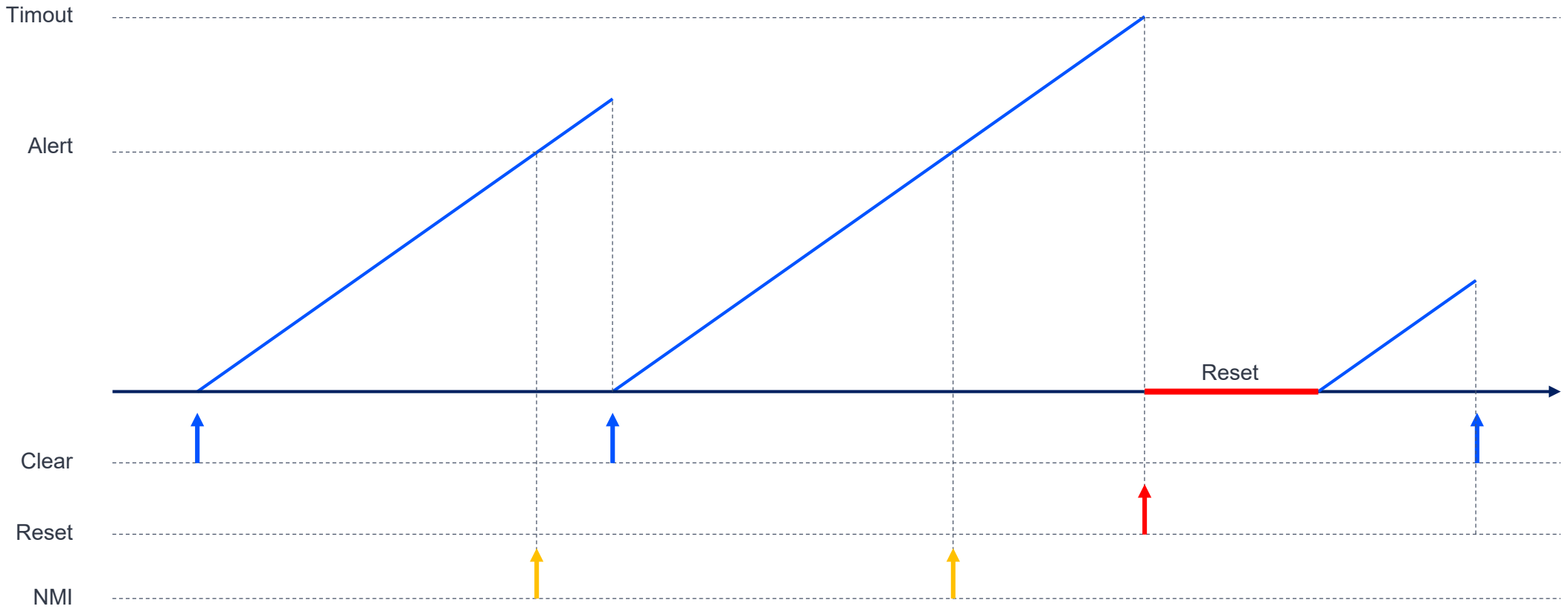
Watchdog

Non-windowed operation



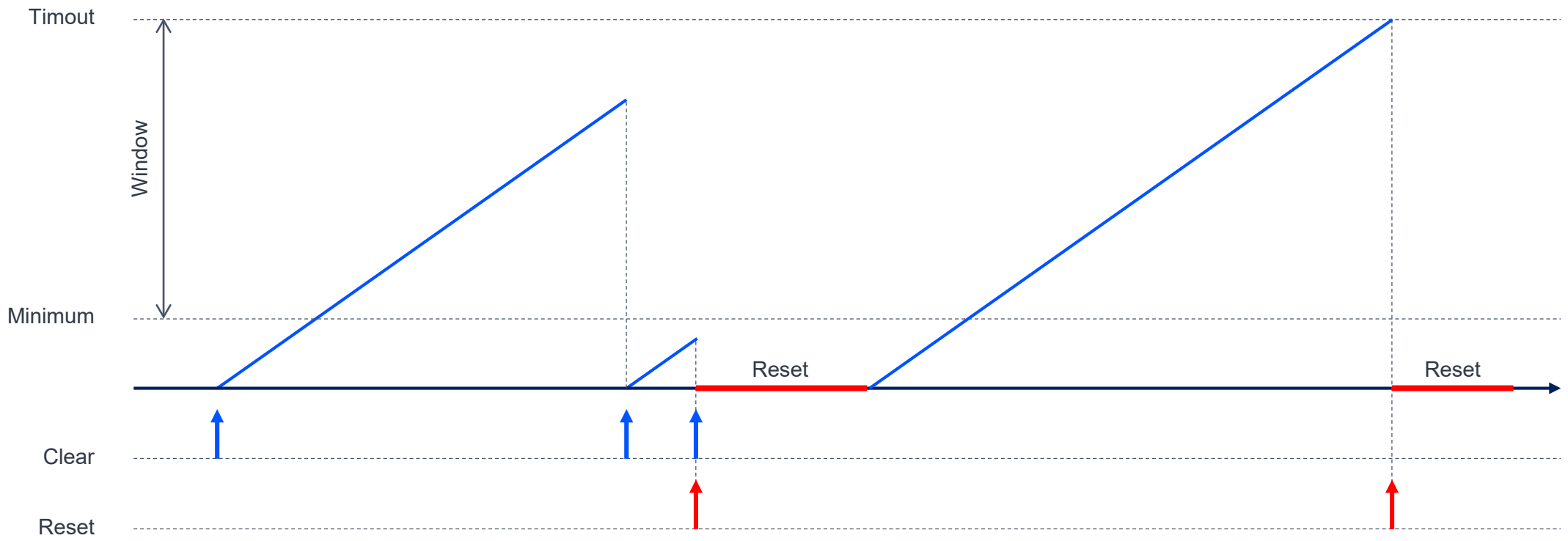
Watchdog

Non-windowed operation, with alert



Watchdog

Windowed operation



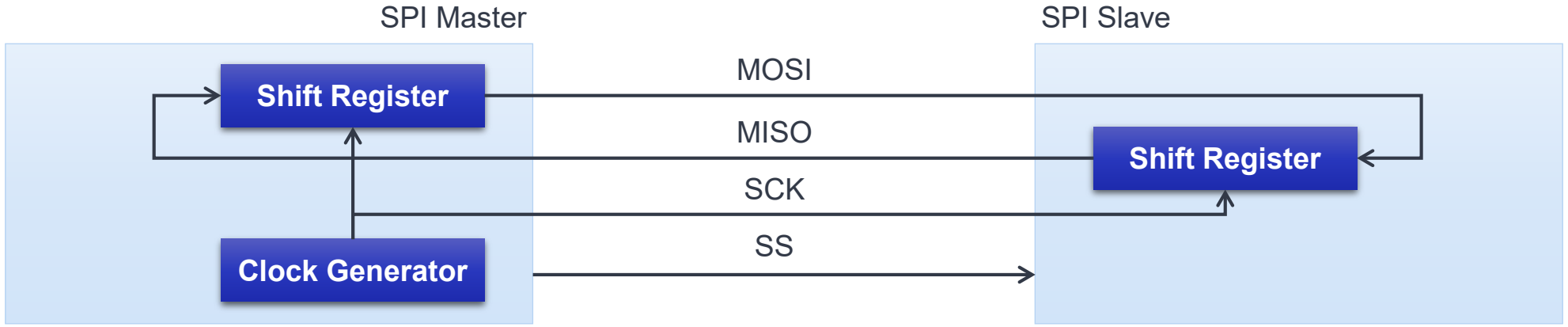
Communication

SPI Bus

SPI

Main characteristics

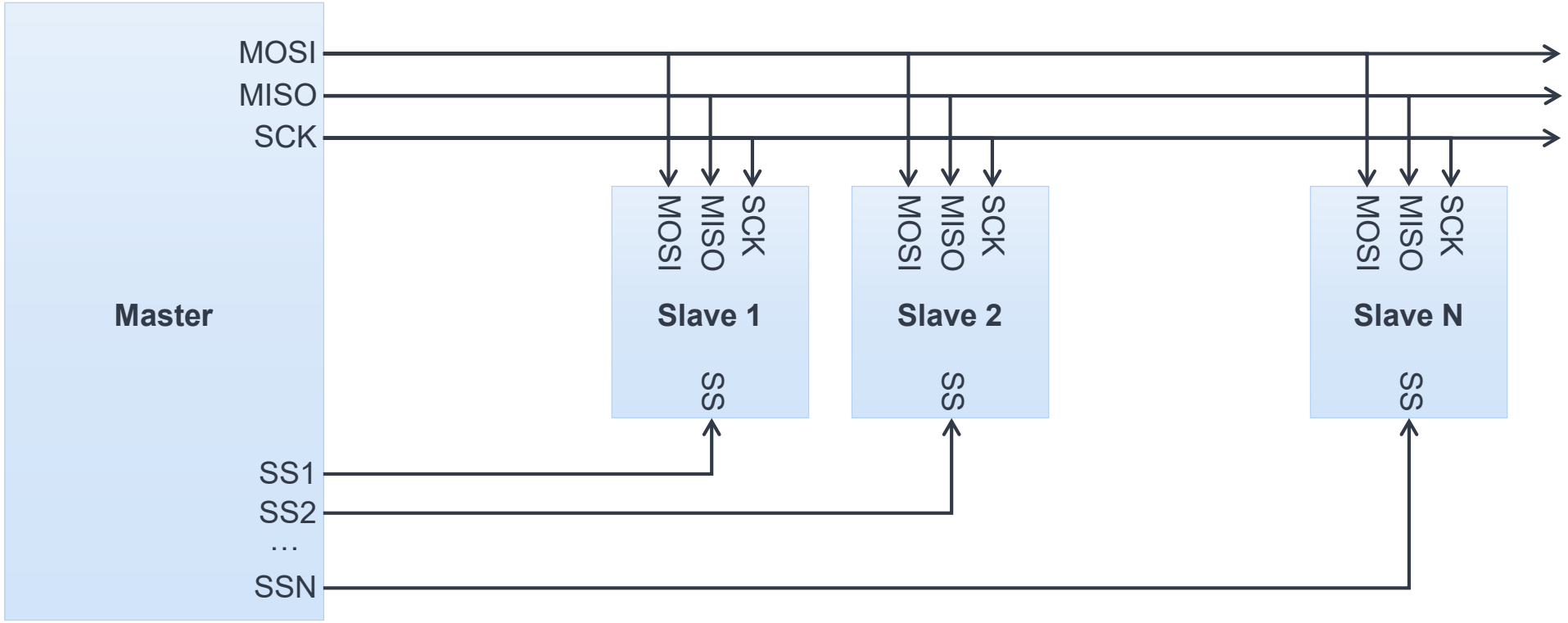
- Single master, multiple slaves
- Synchronous: all device share a clock signal generated by the master
- High bit-rates, up to 20Mbit/s
- 3 lines (MISO, MOSI, SKC) to create a distributed shift register
- 1 line (SS) to select slave device
- 4 operating modes identified by 2 parameters: Polarity and Phase



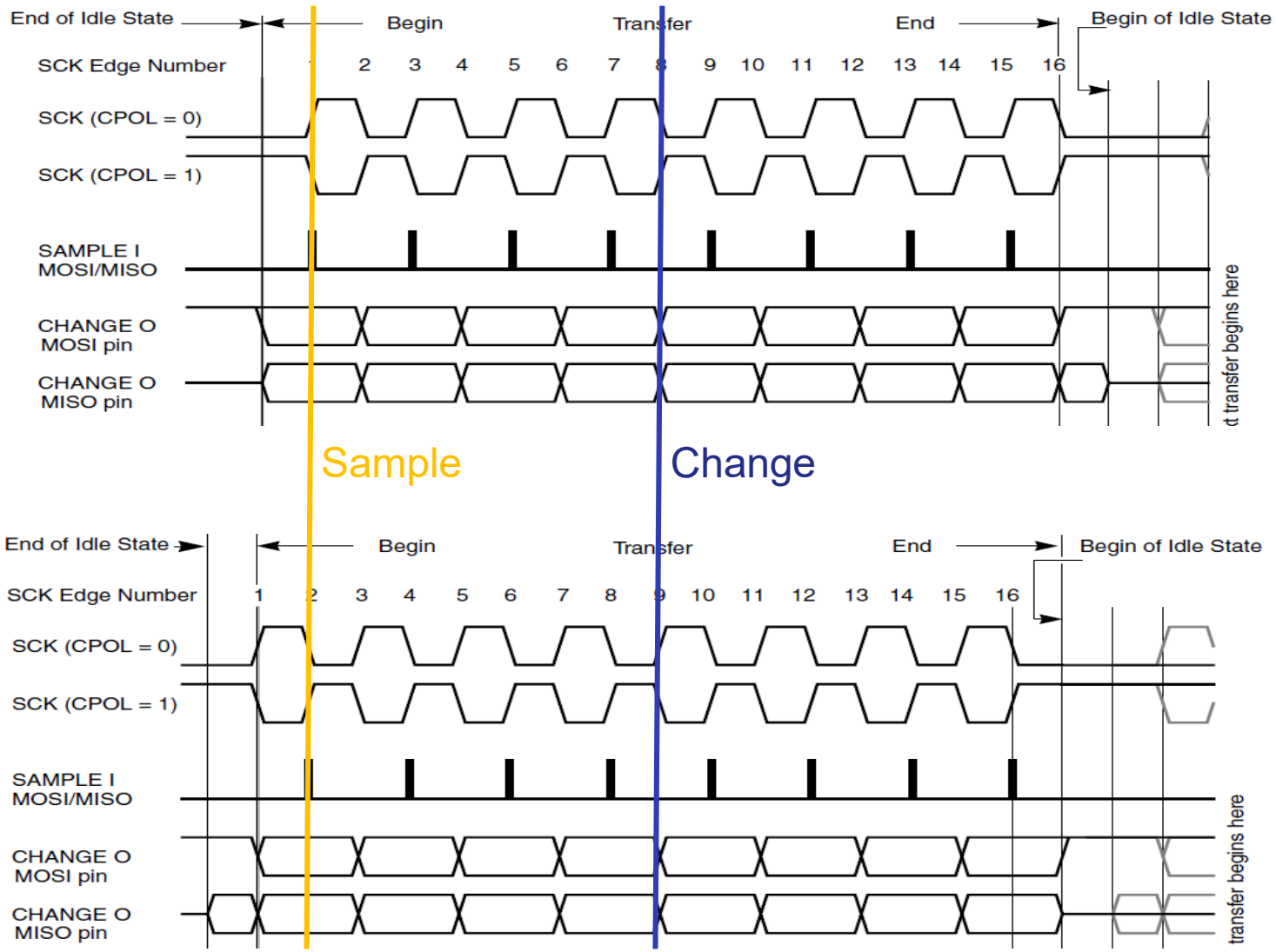
SPI

Multi-slave architecture

- Connecting N slaves requires 3 + N lines
- Devices may be operated at different speeds



SPI


End of Idle State
Begin
Transfer
End
Begin of Idle State
SCK Edge Number
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
SCK (CPOL = 0)
SCK (CPOL = 1)
SAMPLE I
MOSI/MISO
CHANGE O
MOSI pin
CHANGE O
MISO pin
Sample
Change
transfer begins here
End of Idle State
Begin
Transfer
End
Begin of Idle State
SCK Edge Number
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
SCK (CPOL = 0)
SCK (CPOL = 1)
SAMPLE I
MOSI/MISO
CHANGE O
MOSI pin
CHANGE O
MISO pin
transfer begins here

SPI Operation

Transmit

- Master asserts SS
- Master generates clock pulses
- Master shifts out data on MOSI
- Slave receives data on MOSI

Receive

- Master asserts SS
- Master generates clock pulses
- Master shifts out “dummy” data on MOSI
- Slaves shifts out data on MISO
- Master reads data on MISO

SPI Operation

Transfer

- Master asserts SS
- Master generates clock pulses
- Master shifts out data on MOSI
- Slave receives data on MOSI and shifts out data on MISO

Especially useful in maintaining two or more devices synchronized

- Common in industrial application, where state/command memory pages are continuously exchanged
- Dummy bytes at beginning/end of the frames can be added to account for delays between write/read operations

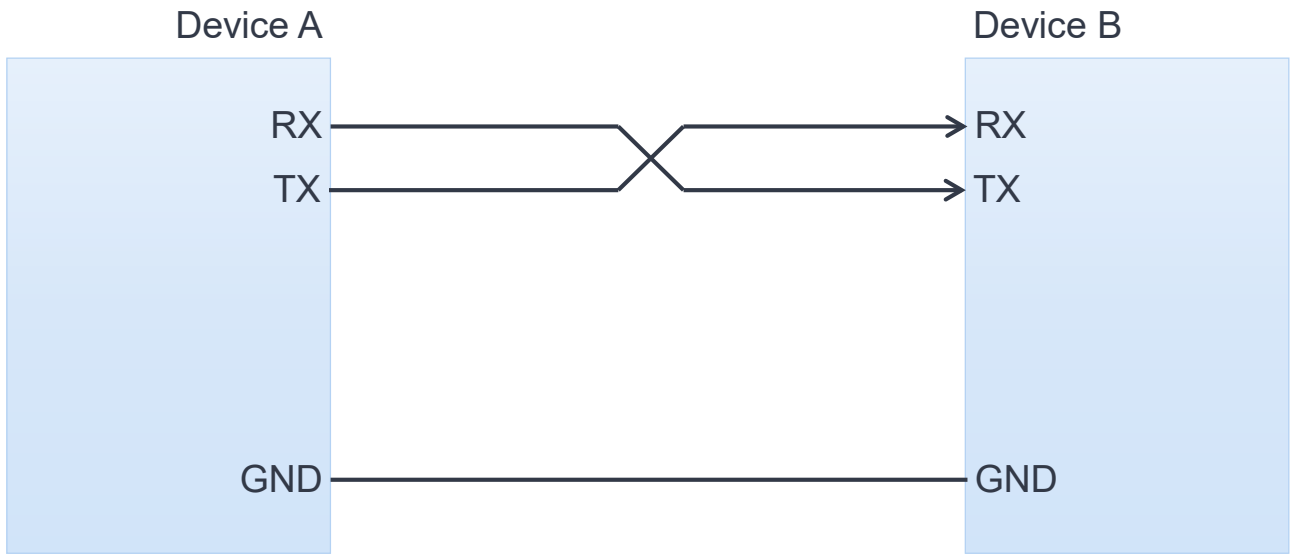
Communication

UART

UART

Main characteristics

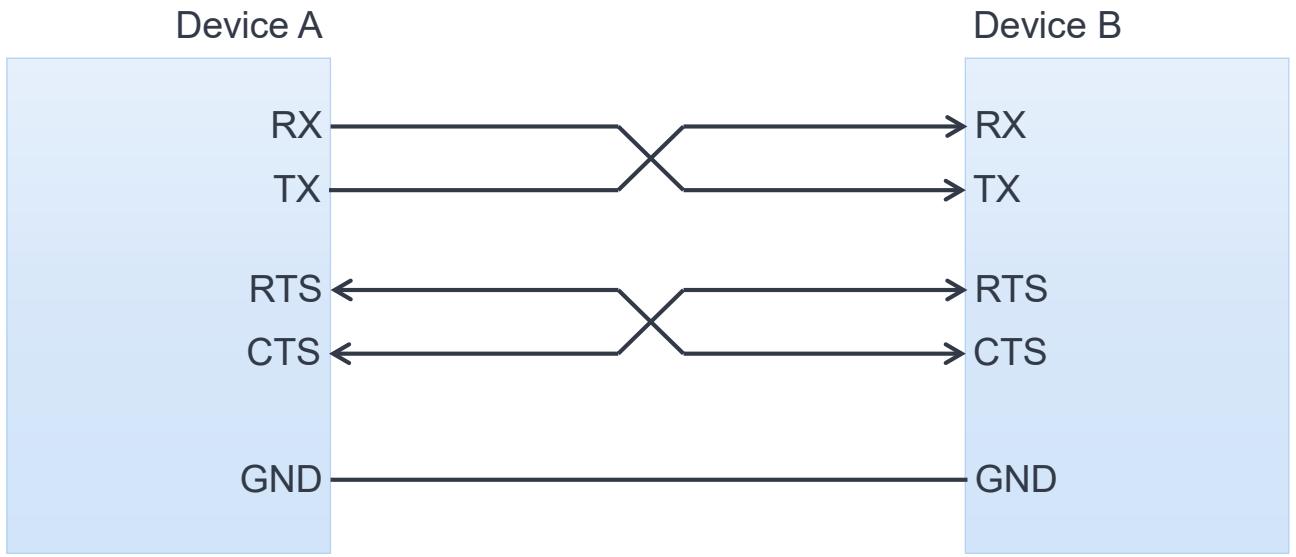
- Point-to-point, transmitter/receiver
- Asynchronous: transmitter and receiver do not share a common clock signal
- Low to medium bit-rates, up to 1Mbit/s, rearely more
- Simplified interface with two lines: TX, RX



UART

Hardware flow control

- Allows devices negotiating when to start/stop receiving data
- Fast tansmitters may be required to pause while slow receiver processes data
- RTS: Ready To Send. The transitters asks to transmit data
- CTS: Clear To Send. The receiver allows transmitter to start sending data



UART

Data trasmission is organized in «frames» composed of

- 1 start bit
- 7 or 8 data bits
- 0 or 1 parity bit
- 1 or 2 stop bits

Configurations are described by short codes

- 8N1: 8 data bits, No parity, 1 stop bit

- 7E2: 7 data bits, Even parity, 2 stop bits

UART Operation

Transmit / Receive

- Transmitter sends bits over TX line
- Receivers reads bits over RX line

Hardware flow control

- Transmitter asserts RTS
- When ready, receiver asserts CTS
- Transmitter sends bits over the TX line and receiver reads bits over the RX line
- Receiver requests transmitter to pause sending by de-asserting CTS
- Transmitter suspends sending bits
- Transmitter waits receiver to re-assert CTS

Communication

I2C

I2C

Main characteristics

- Synchronous protocol
- Multiple master, multiple slaves
  - Most applications are single mater
- One bidirectional data line (SDA)
- One clock line (SCL)

Mixes and improves UART and SPI

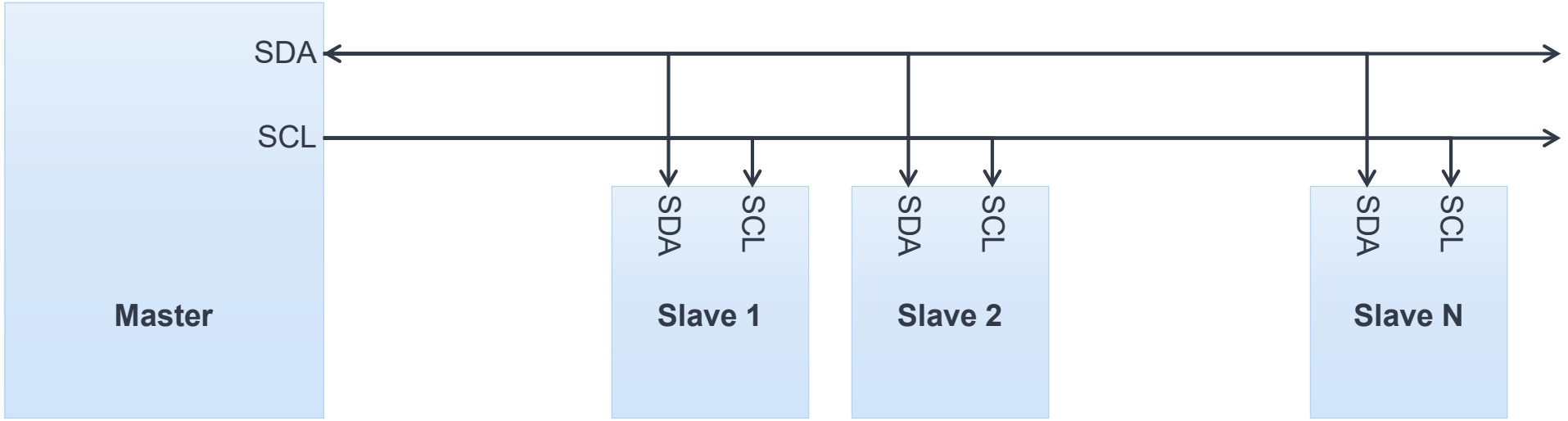
- Better than UART since supports multiple slave
- Better than SPI since does not require additional lines for additional devices

More complex than both

- Requires a specific protocol for executing read and write operations
- Lower data rates

I2C

The most common setup is single master, multiple slaves



I2C Protocol

Master-slaves

- The master initiates all transfer operation
- The data line is controlled both by the master and the slaves
- The clock is always generated by the master

Limited speeds supported by most devices

- 100 kbps in normal mode
- 400 kbps in fast mode
- 3.4 Mbps or 5Mbps available from standard but rarely supported

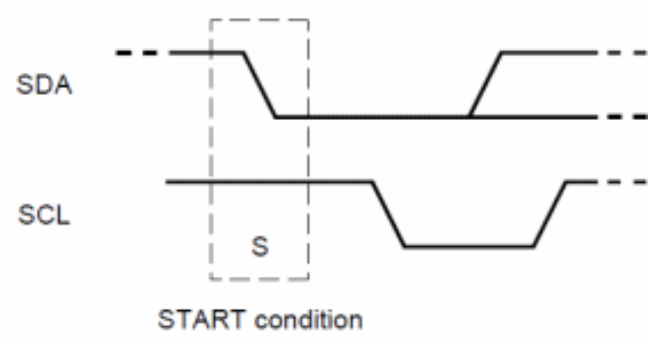
Two operations

- Write
- Read

I2C Protocol

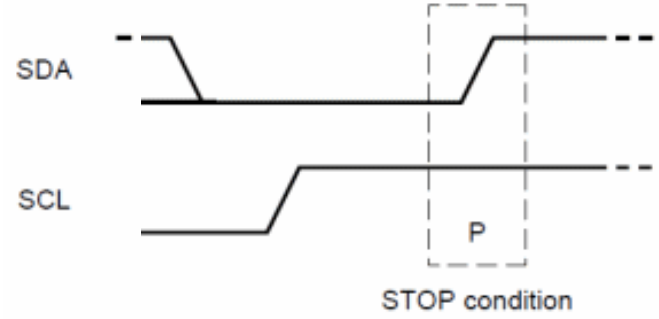
Start condition

- Falling edge on SDA while SCL is high



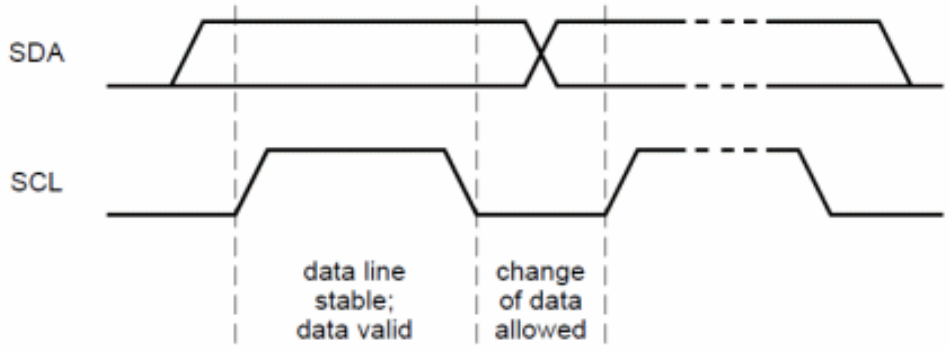
Stop condition

- Rising edge on SDA while SCL is high



Data sampling

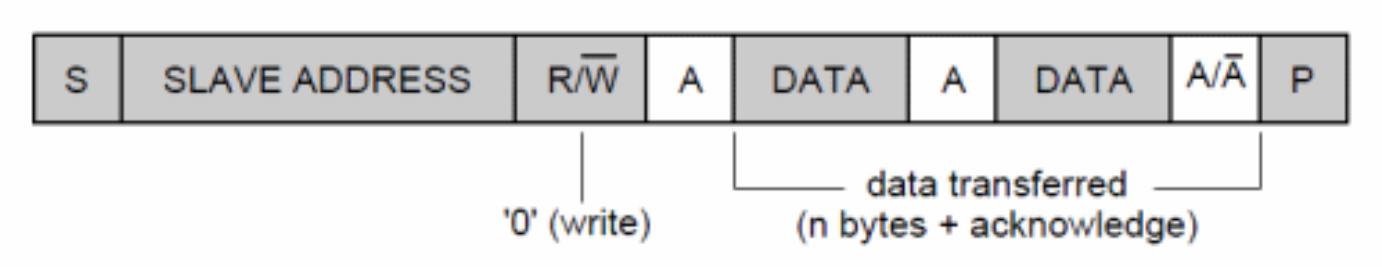
- Data stable while clock high
- Data can change while clock low



I2C Protocol

Write operation

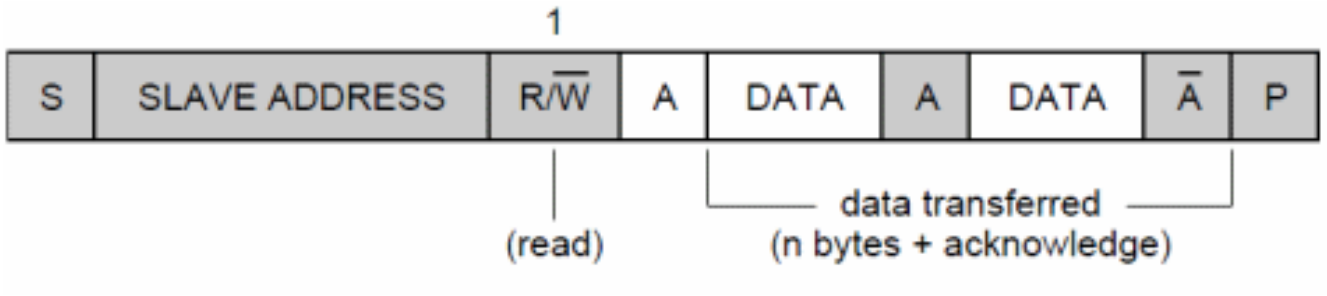
- Master generates start condition (S)
- Master setup-to-write (address, R/W=0), slave acknowledges
- Master sensd data chunks, slave acknowledges
- Master generates stop condition (P)



I2C Protocol

Read operation

- Master generates start condition
- Master setup-to-read (address, R/W=1), slave acknowledges
- Slave sensd data chunks, master acknowledges
- When done, slave do no acknowledges (not A)
- Master generates stop condition



I2C Protocol

Transfer operation: write and read

- Master generates start condition
- Master setup-to-write (address, R/W=0), slave acknowledges
- Master sensd data chunks, slave acknowledges
- Master generates a repeated start condition (Sr)
- Master setup-to-read (address, R/W=1), slave acknowledges
- Slaves sends data chunks, master acknowledges
- When done, slave do not acknowledges (not A)
- Master generates a stop condition (P)