

Design and development of embedded systems for the Internet of Things (IoT)

Fabio Angeletti Fabrizio Gattuso



SAPIENZA
UNIVERSITÀ DI ROMA



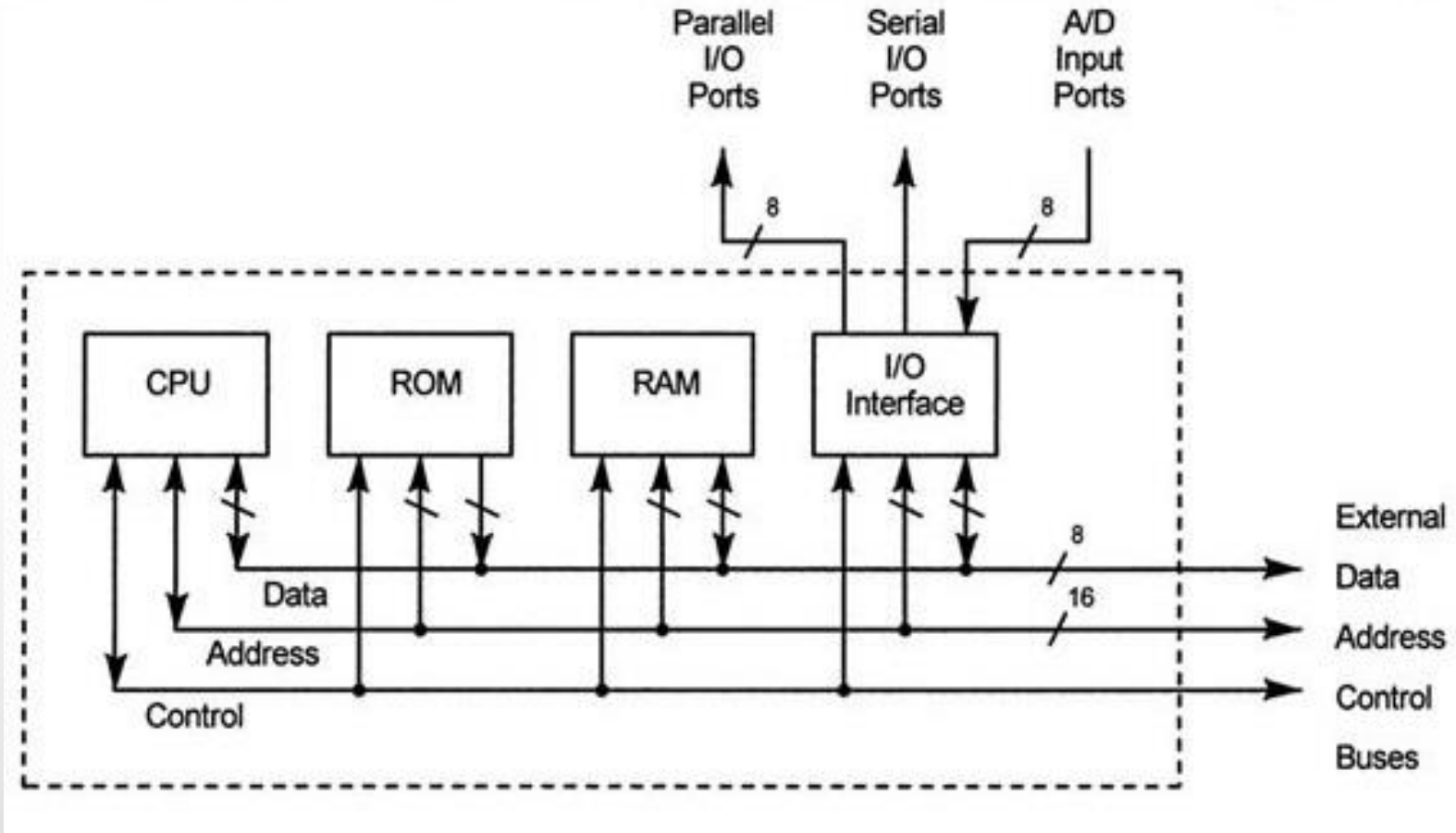
W • S E N S E
INTEGRATED CABLELESS SOLUTIONS

Microcontroller

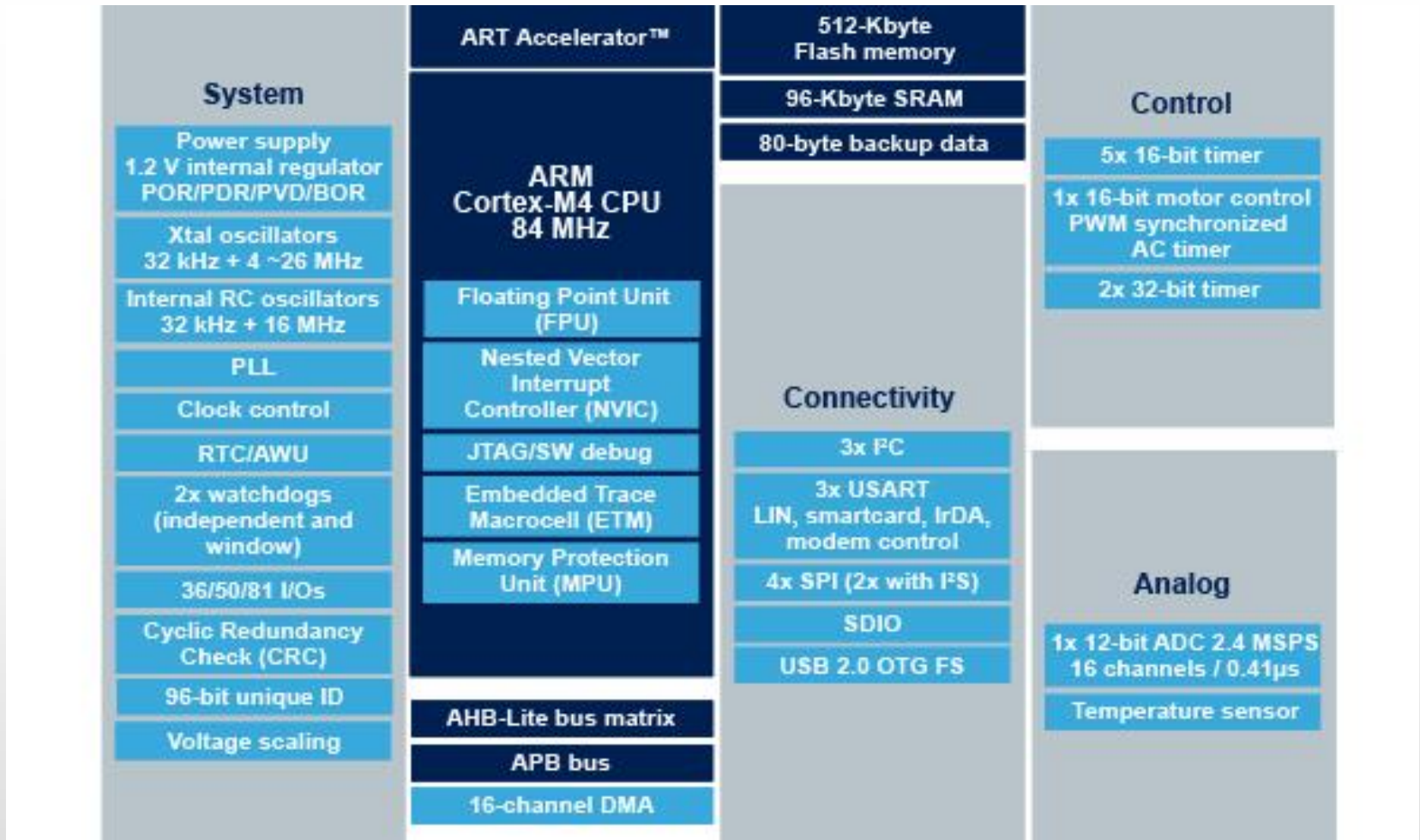
- It is essentially a small computer on a chip
- Like any computer, it has memory, and can be programmed to do calculations, receive inputs, and generate outputs
- Unlike a PC, it incorporates memory, a CPU, peripherals and I/O interfaces into a single chip



Generic architecture



STM32F401RE - Architecture



Duplex

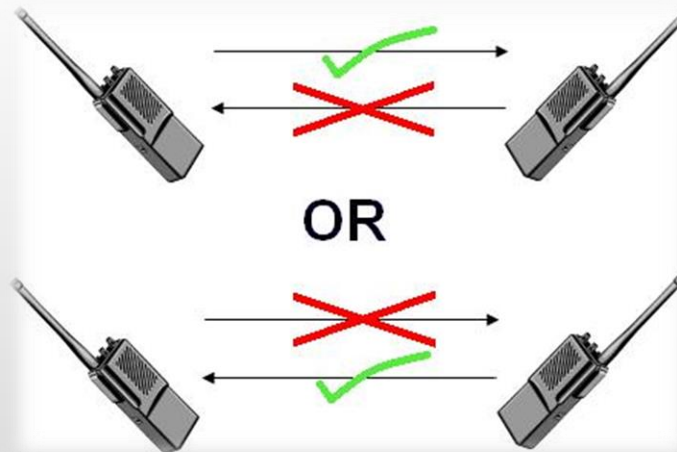
Send AND receive at the same time

FULL DUPLEX

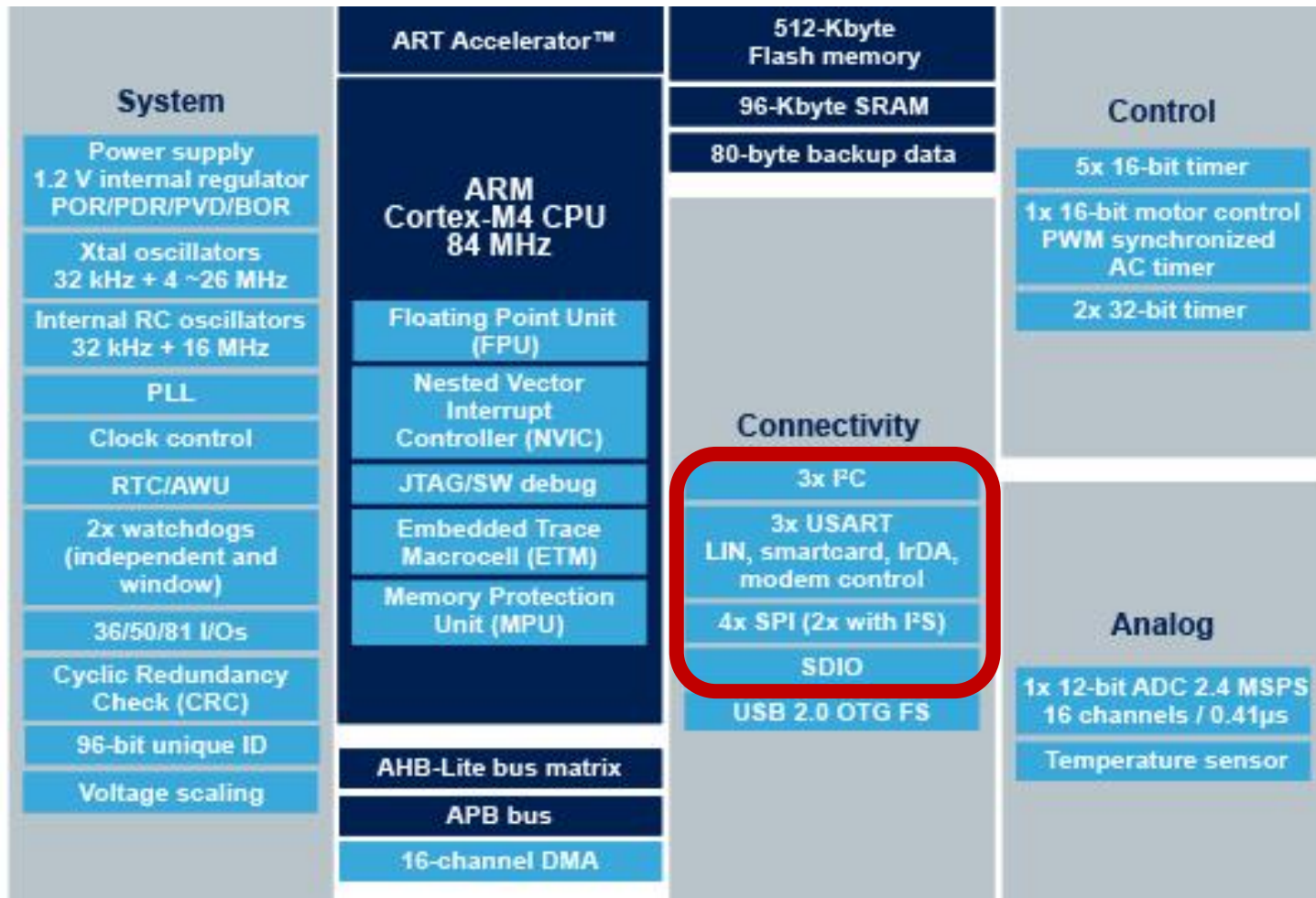


Send OR receive at the same time

HALF DUPLEX



Connectivity



I²C

Inter-Integrated Circuit - is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial computer bus.

HALF DUPLEX



I²C

It is appropriate for peripherals where simplicity and low manufacturing cost are more important than speed, like:

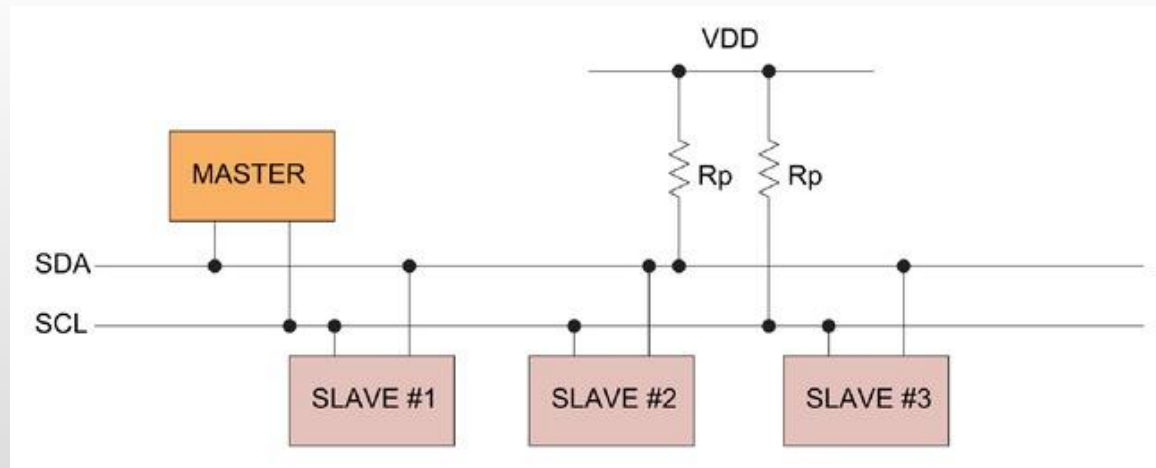
- EEPROMs
- ADC / DAC
- Small displays
- Digital potentiometers
- ...



I²C

Uses only two bi-directional open-drain lines, pulled up with resistors:

- Serial Data Line (SDA)
- Serial Clock Line (SCL)



I²C

The reference design has a 7 bits addressing scheme that can be extended to 10 bits (thus allowing up to 128 -> 1024 devices on the bus).

Multiple clock speeds are supported: 100kHz, 400kHz, 1MHz... During each clock cycle, a bit is transferred:

$$100\text{kHz} = 100\text{kbit/s}$$

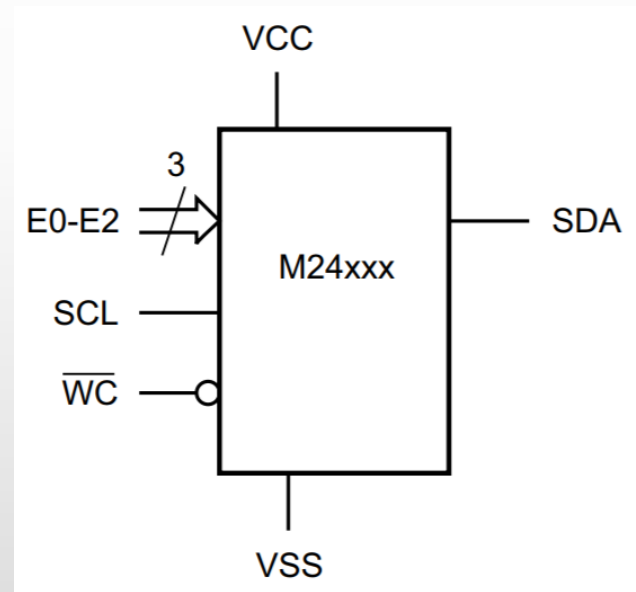
There is an overhead for addressing and other conditions.



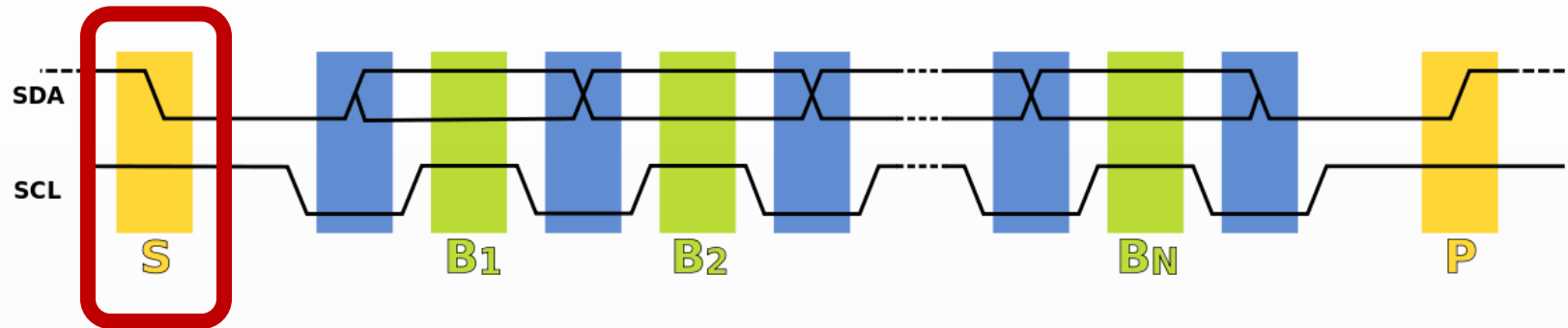
I²C

To mitigate the chance of address collision with the standard 7 bits addressing, some ICs have device address definition PINs to mitigate the collision problem.

E0, E1, E2 set the last 3 bits



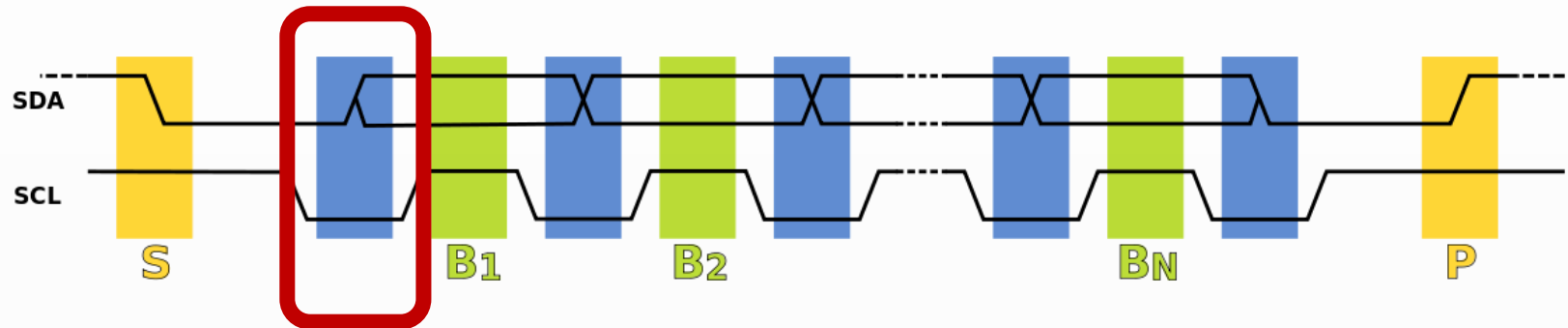
I²C - Timing



1. Data transfer is initiated with a *start* bit (S) signaled by SDA being pulled low while SCL stays high.



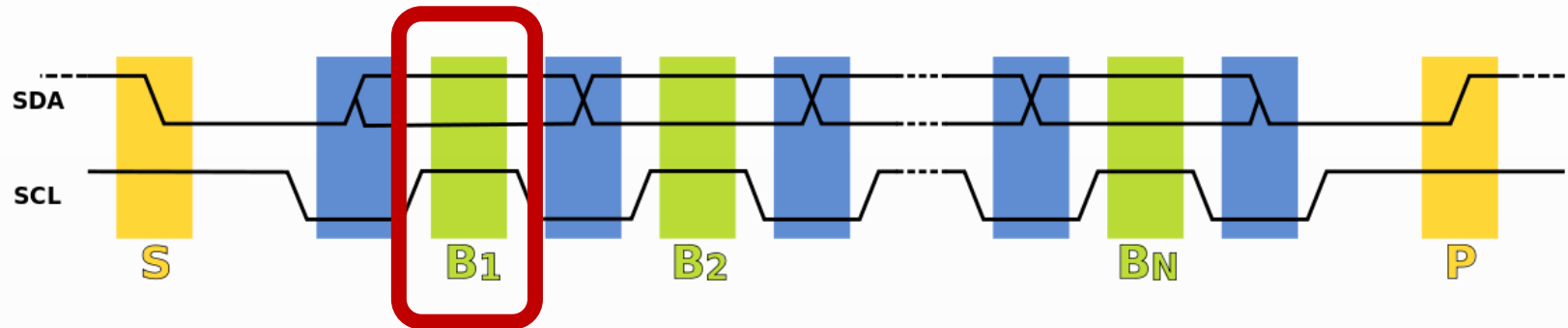
I²C - Timing



2. SCL is pulled low, and SDA sets the first data bit level while keeping SCL low (during blue bar time).



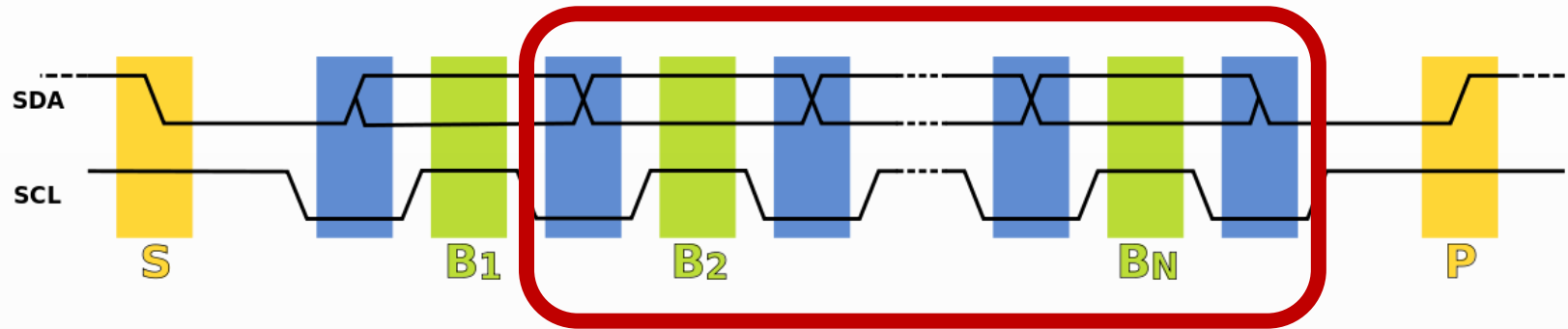
I²C - Timing



3. The data are sampled (received) when SCL rises for the first bit (B1). For a bit to be valid, SDA must not change between a rising edge of SCL and the subsequent falling edge (the entire green bar time).



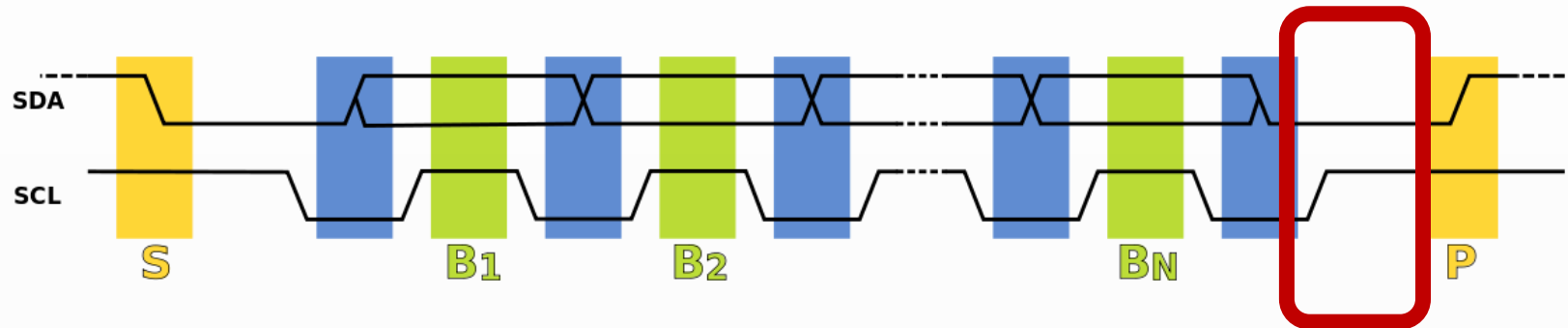
I²C - Timing



4. This process repeats, SDA transitioning while SCL is low, and the data being read while SCL is high (B2, ...Bn).



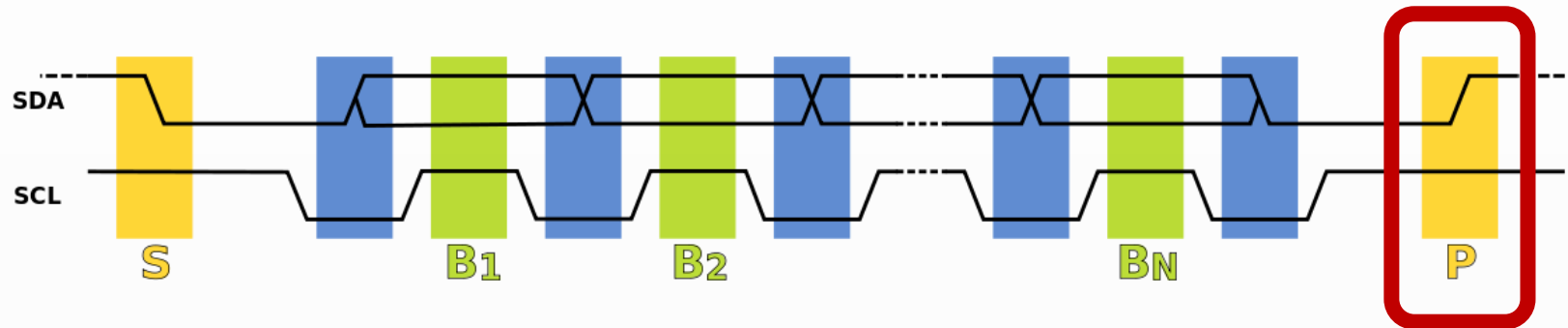
I²C - Timing



5. The final bit is followed by a clock pulse, during which SDA is pulled low in preparation for the stop bit.



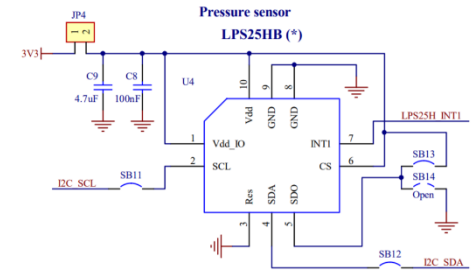
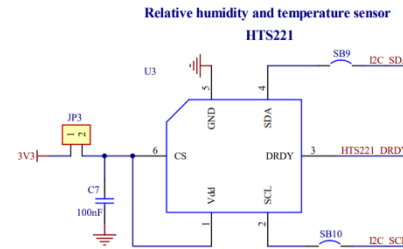
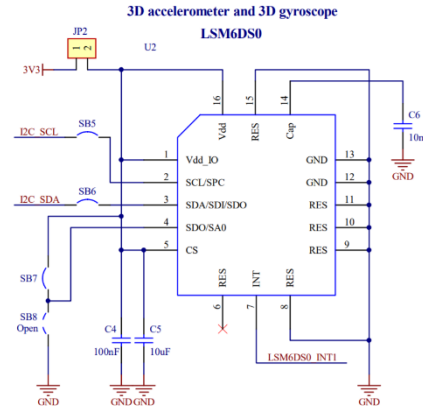
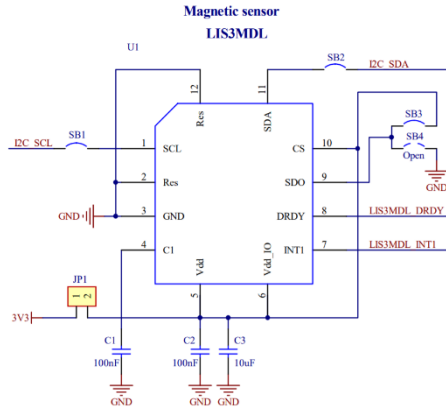
I²C - Timing



6. A *stop* bit (P) is signaled when SCL rises, followed by SDA rising.



I²C - Example



USART

Universal Synchronous and Asynchronous Receiver-Transmitter - is a type of a serial interface device that can be programmed to communicate asynchronously or synchronously.

We are interested in the asynchronous mode.



HALF DUPLEX AND FULL DUPLEX



USART

The basic bi-directional communication requires two lines:

- **TX – Transmit**
- **RX – Receive**

Hardware data flow control requires two additional lines:

- **RTS – Request To Send**
- **CTS – Clear To Send**

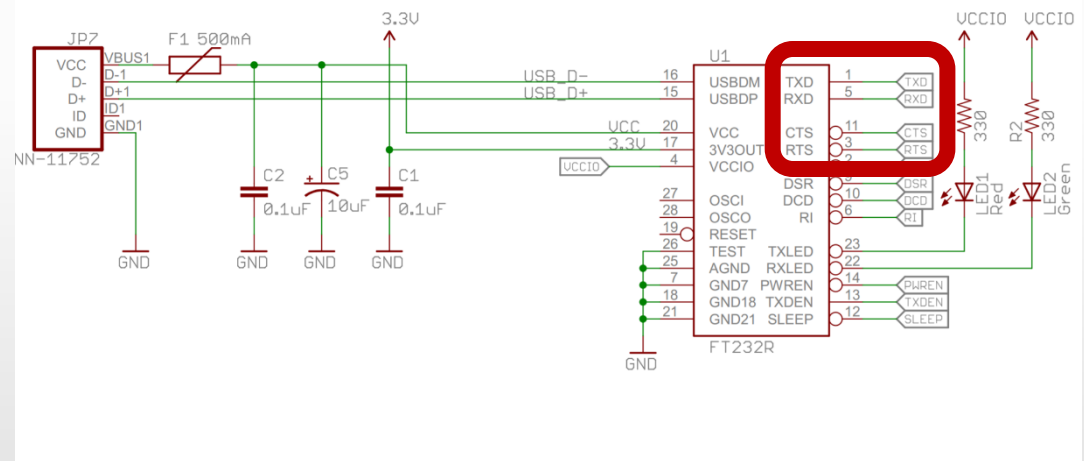
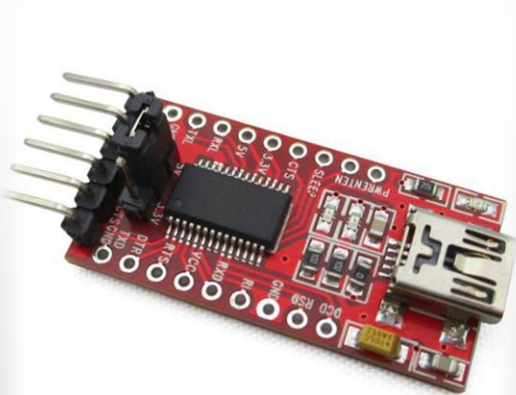
Other less used lines are:

- **DTR – Data Terminal Ready**
- **DSR – Data Set Ready**
- **DCD – Data Carrier Detect**



USART

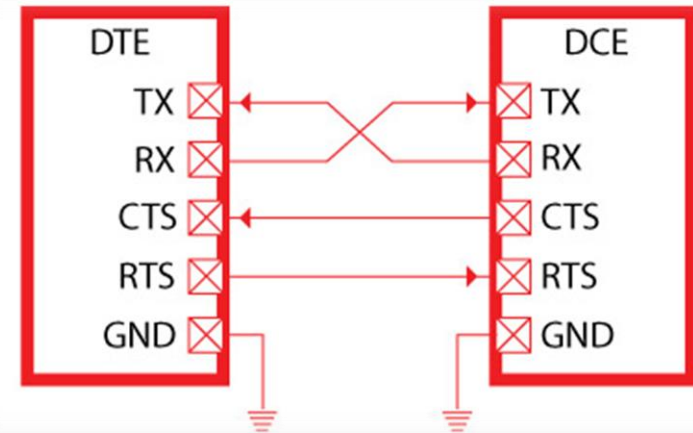
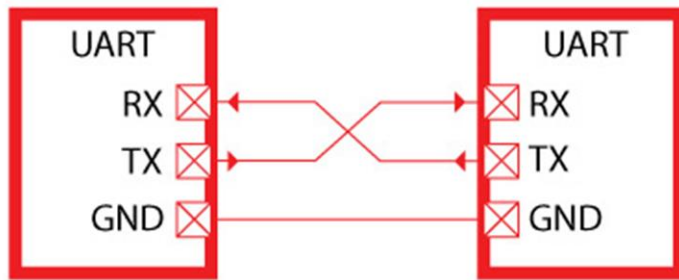
It allows the communication with a computer equipped with RS232 port through a level shifter, or using an USB-to-RS232 adapter, like the IC FTDI's FT232R.



USART - Connections

Simple

With flow control



We will use the simple connection



USART

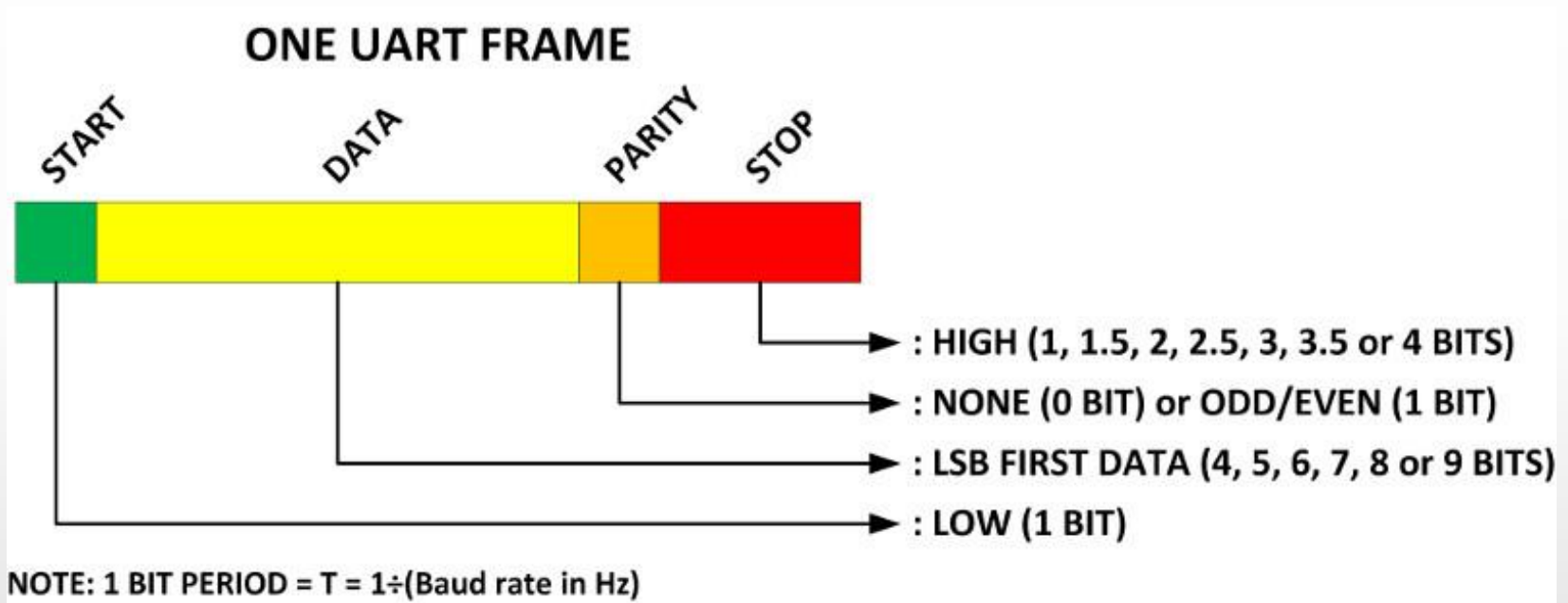
Data is transferred within **frames**.

A frame is composed by:

- **START** a single bit
- **DATA** from 4 to 9 bits
- **PARITY** from none to 1 bit
- **STOP** from 1 to 4 bits (with 0.5 bit resolution)



USART



USART

Asynchronous = **NO CLOCK LINE**

Thus, both RX and TX must know all the parameters of the channel in order to communicate effectively.

- **BAUD RATE**
- **PARITY**
- **DATA SIZE**
- **STOP BITS**



USART

Example:

9600,8,N,1

- 9600 is the baud rate
- 8 is the number of bits in data field
- N is the parity, no parity used
- 1 is the number of stop bits



SPI

Serial Peripheral Interface - consists in a single master device and at least one slave device.

FULL DUPLEX



SPI

Only four signal lines are required:

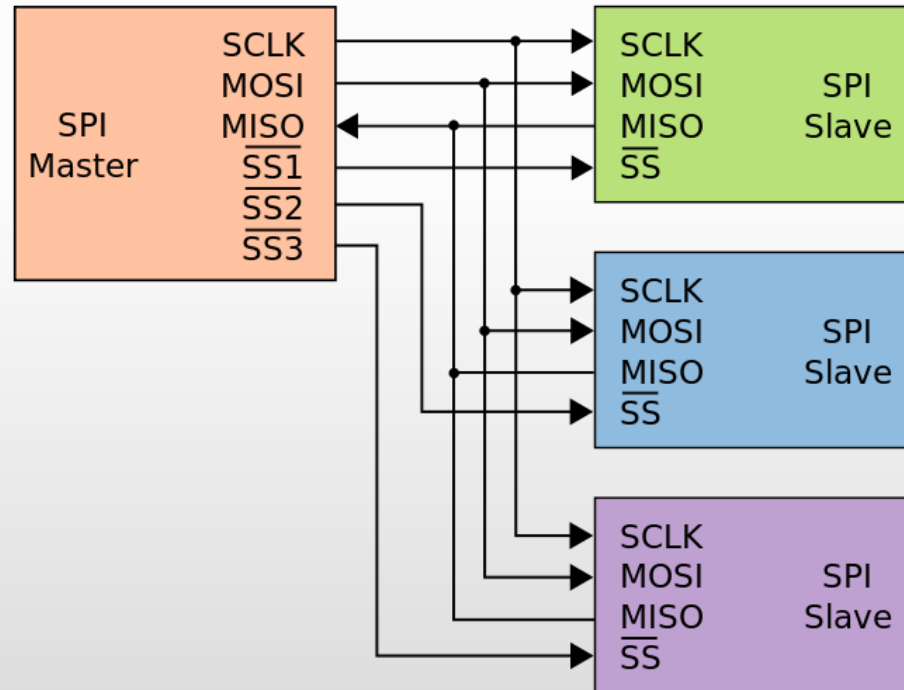
- MISO – Master Input Slave Output
- MOSI – Master Output Slave Input
- SCLK – Serial Clock
- SS – Slave Select

It is used for short distance communications, MISO and MOSI should be tri-state GPIOs.



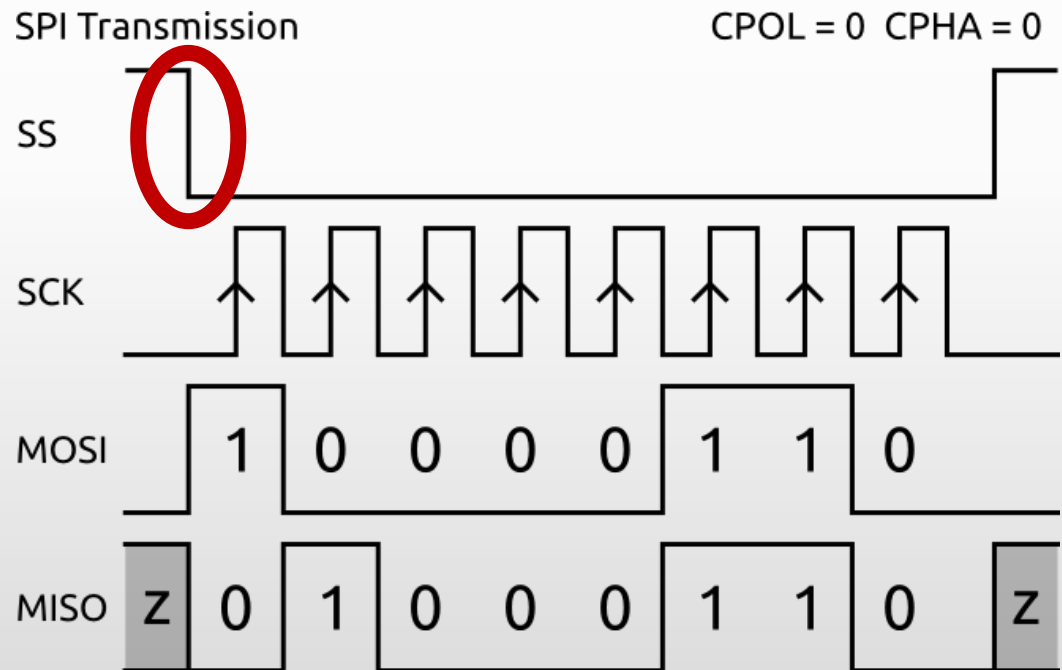
SPI

It is a shared bus with low GPIO requirements. Moreover it is sensibly faster than I2C (some peripherals exceed 10Mbit/s).



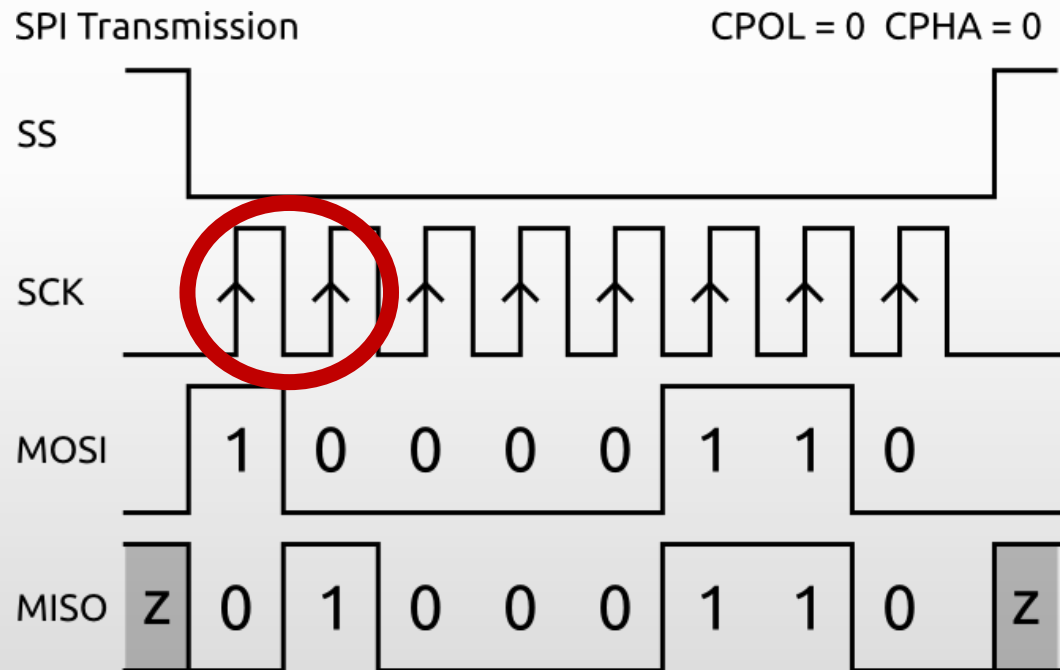
SPI - Timing

The MASTER initiates a communication with a specific SLAVE by pulling low the corresponding SS line while the MOSI line is set up with value 0



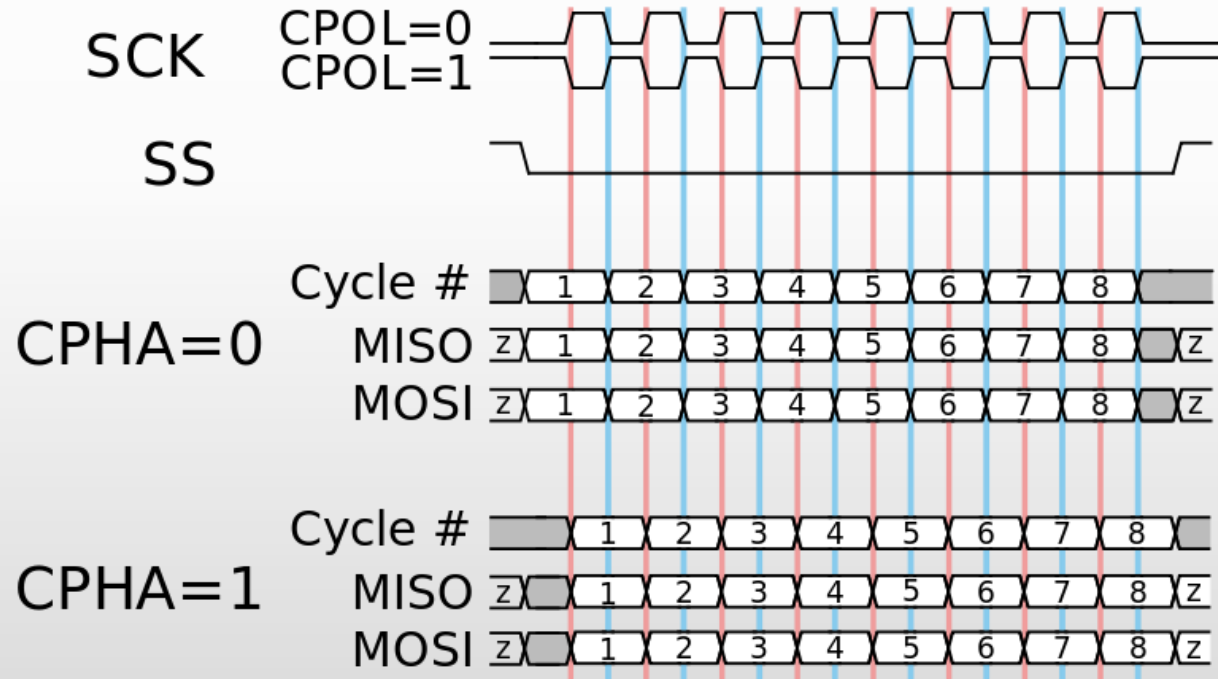
SPI - Timing

Then, the MASTER must configure the clock with a frequency supported by the SLAVE device



SPI - Timing

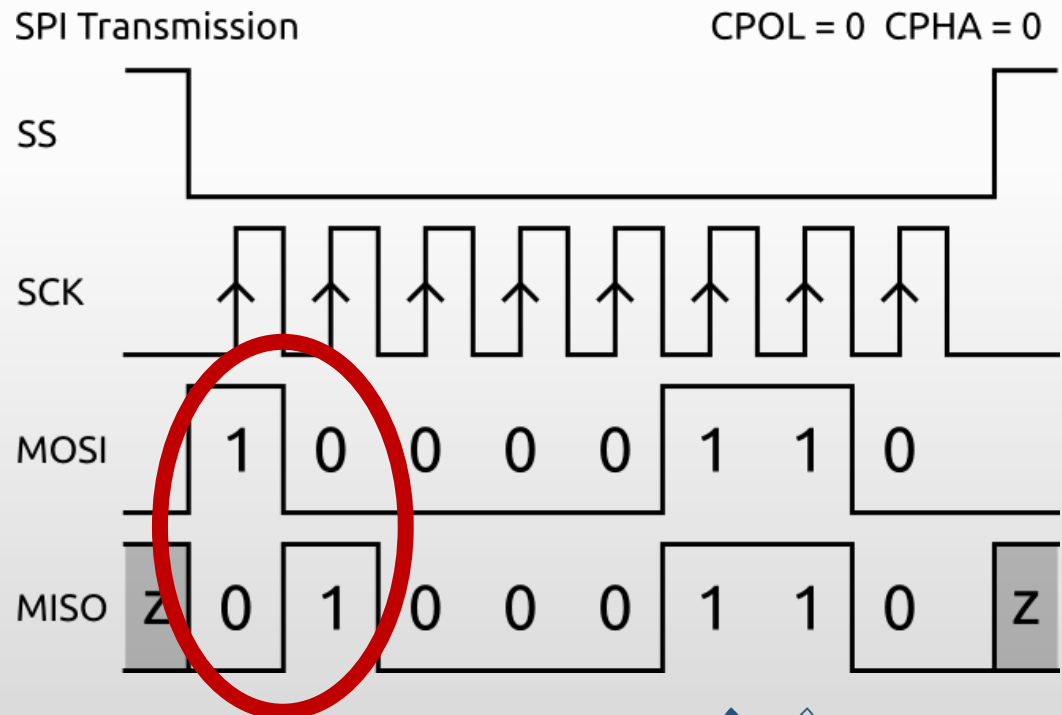
In addition to setting the clock frequency, the MASTER must also configure the clock polarity CPOL and clock phase CPHA to match the SLAVE



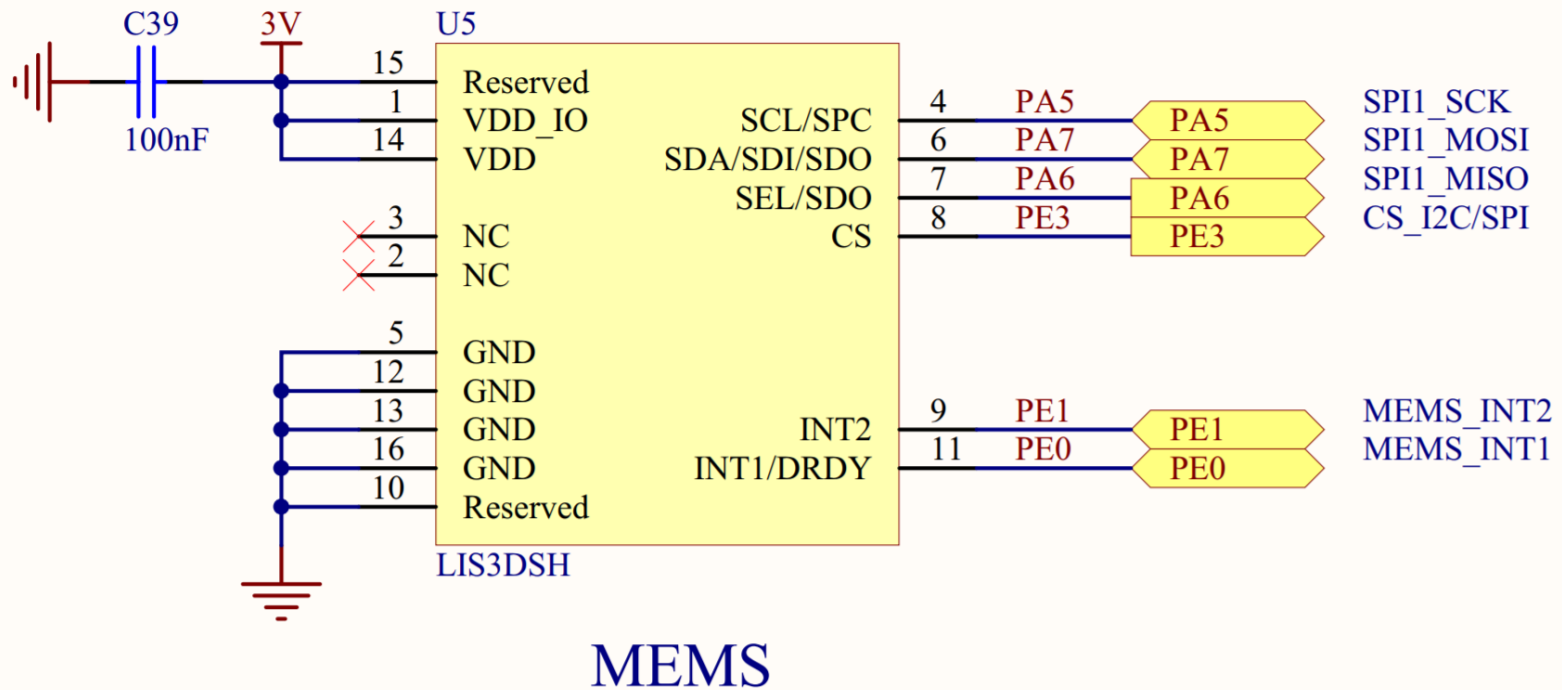
SPI - Timing

During each clock cycle, a bit is transferred from the MASTER to the SLAVE and a bit is transferred from the SLAVE to the MASTER.

FULL
DUPLEX



SPI – Example



I²S

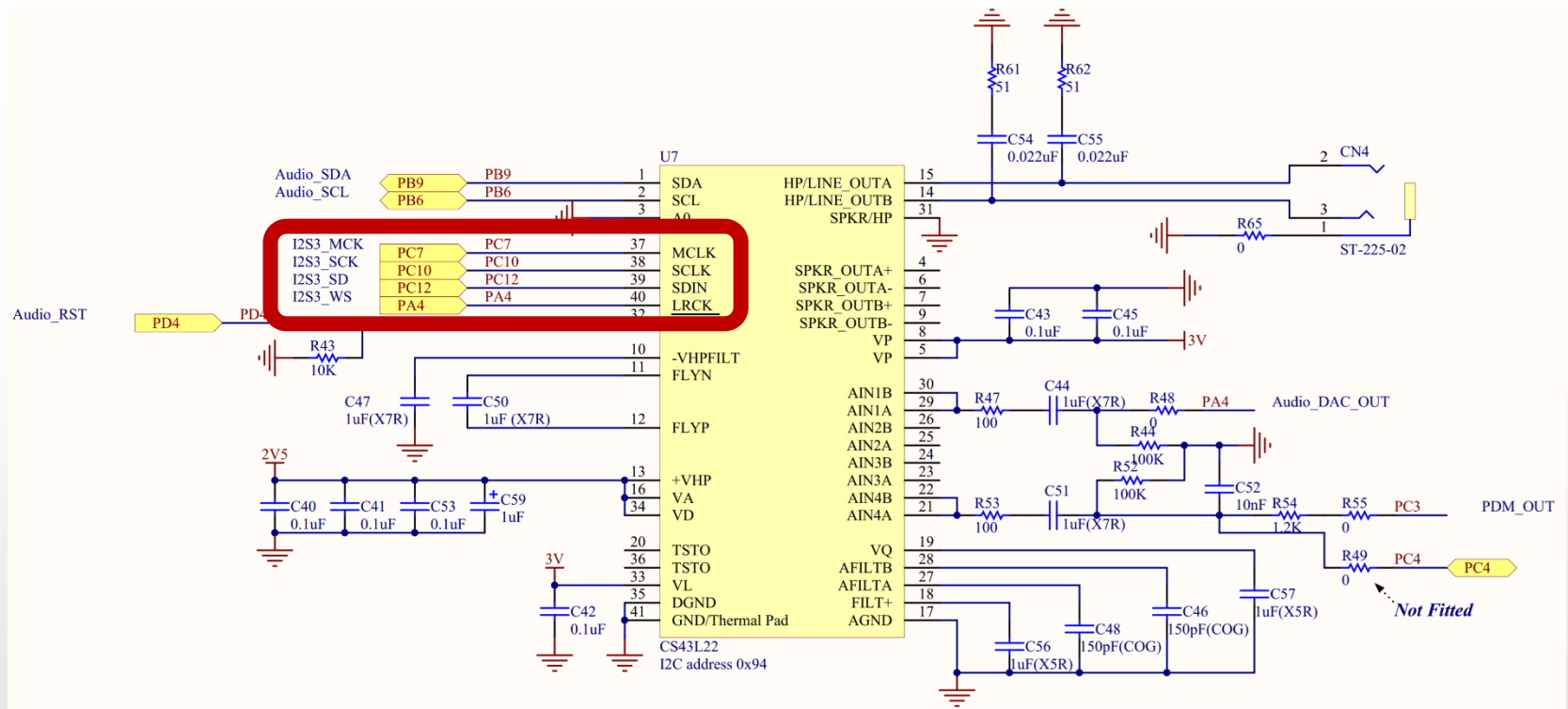
Inter-IC Sound – it is used for connecting digital audio devices together, communicates PCM audio data between integrated circuits.

Uses at least three lines:

- SCK – Continuous serial clock
- LRCLK – Left/Right channel selection
- SD – Multiplexed data



I²S - Example



SDIO

Secure Digital Input Output - is an extension of the SD specification to cover I/O functions.

SD is exactly the interface of common memory cards used in electronic devices.

HALF DUPLEX



SDIO

Requires from 3 to 10 lines, depending on how many bits can be transferred each clock cycle.

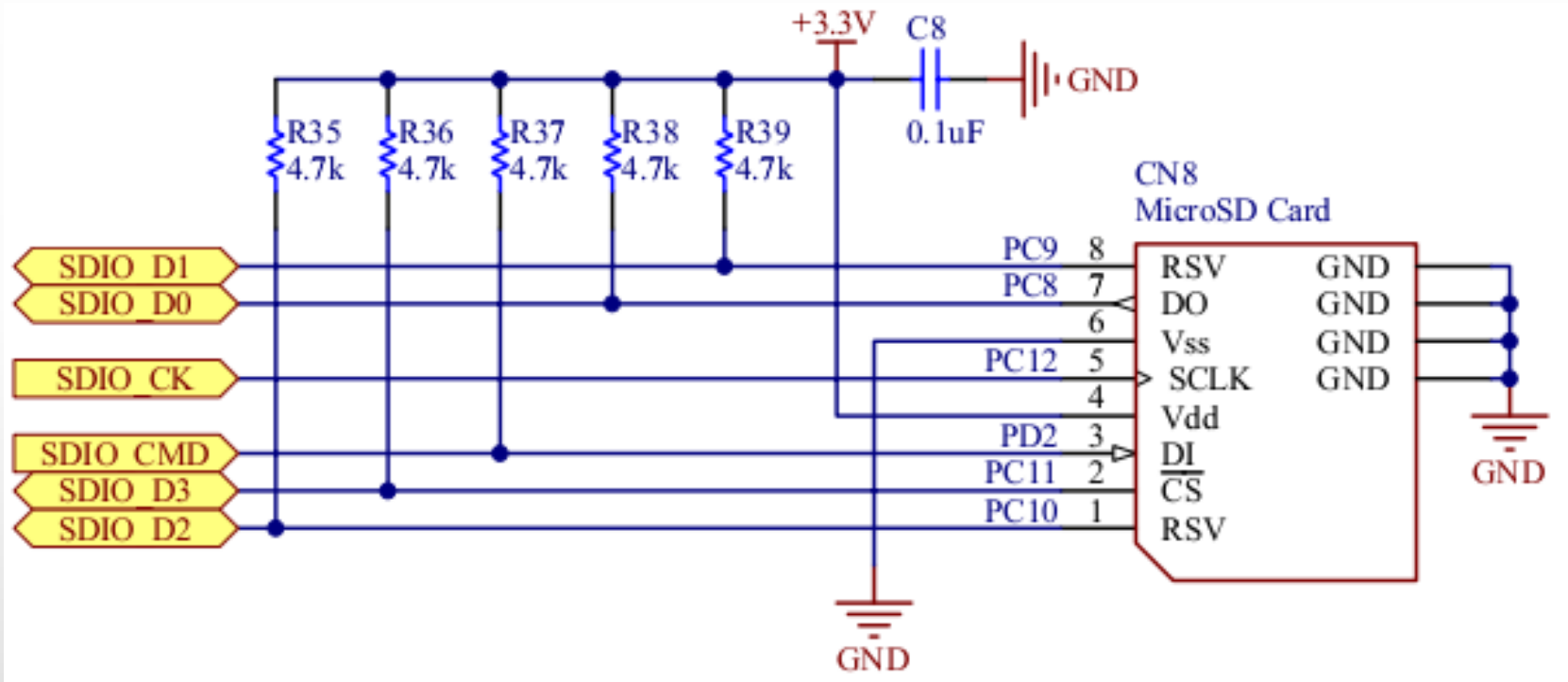
- **CLOCK** – clock signal from the host
- **COMMAND** – command/response signal
- **DATA** – from 1 to 8 lines

Each line, apart from **CLOCK**, requires a **PULL-UP** resistor.

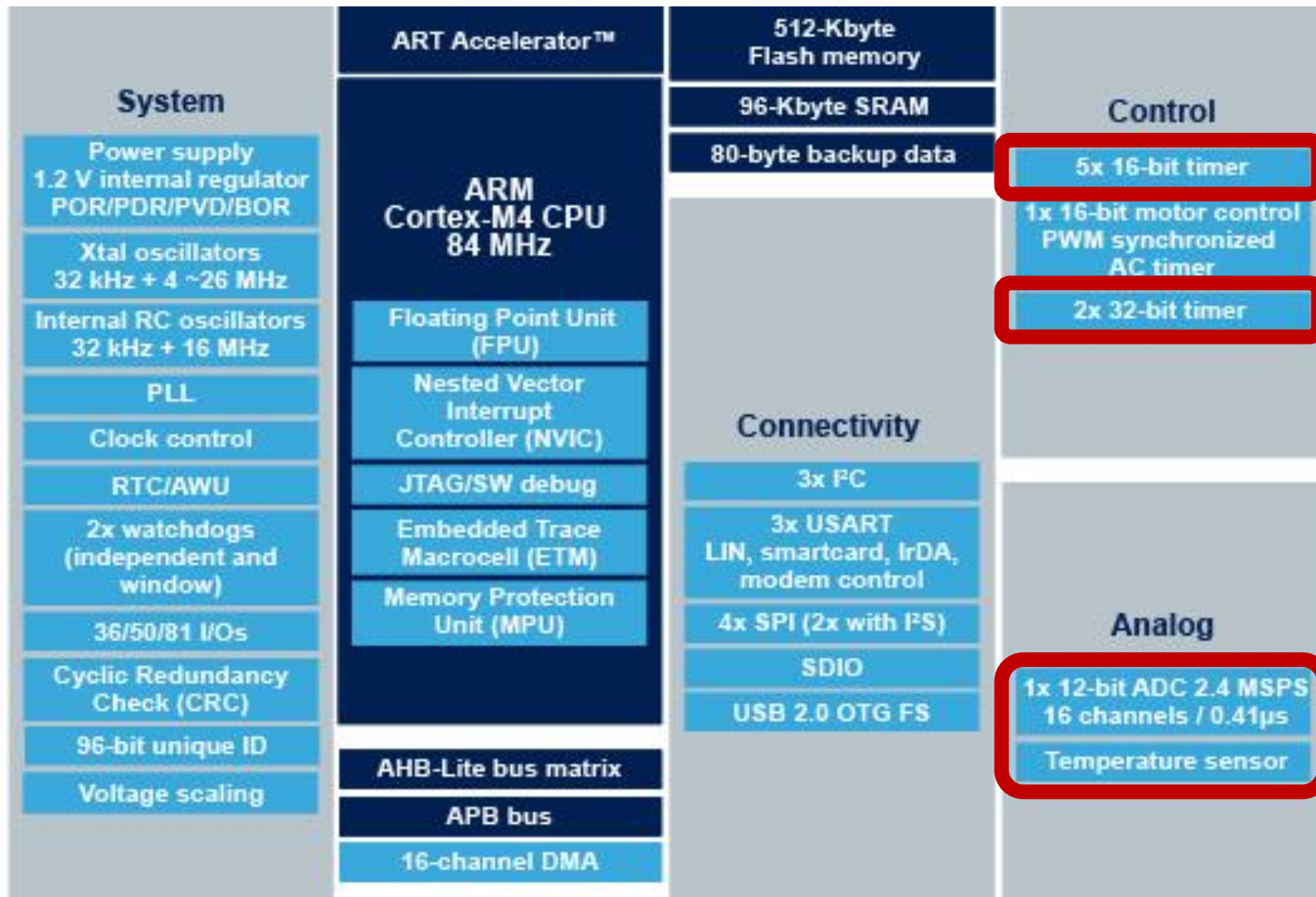


SDIO - Example

4-bit SDIO interface



Timers and analog



Timers

A timer is a digital counter that increments or decrements a variable according to a fixed input frequency – the **clock source**

The variable can be of any size, we use 16 bit and 32 bit timers. The size affects the maximum number of input clocks allowed before reaching an overflow or underflow situation



Timers

Advanced timers may have comparison logic to compare the timer value against a specific value, set by software, that triggers some action when the timer value matches the preset value.

One specialist use of hardware timers in computer systems is as watchdog timers, that are designed to perform a hardware reset of the system if the software fails.



Timer

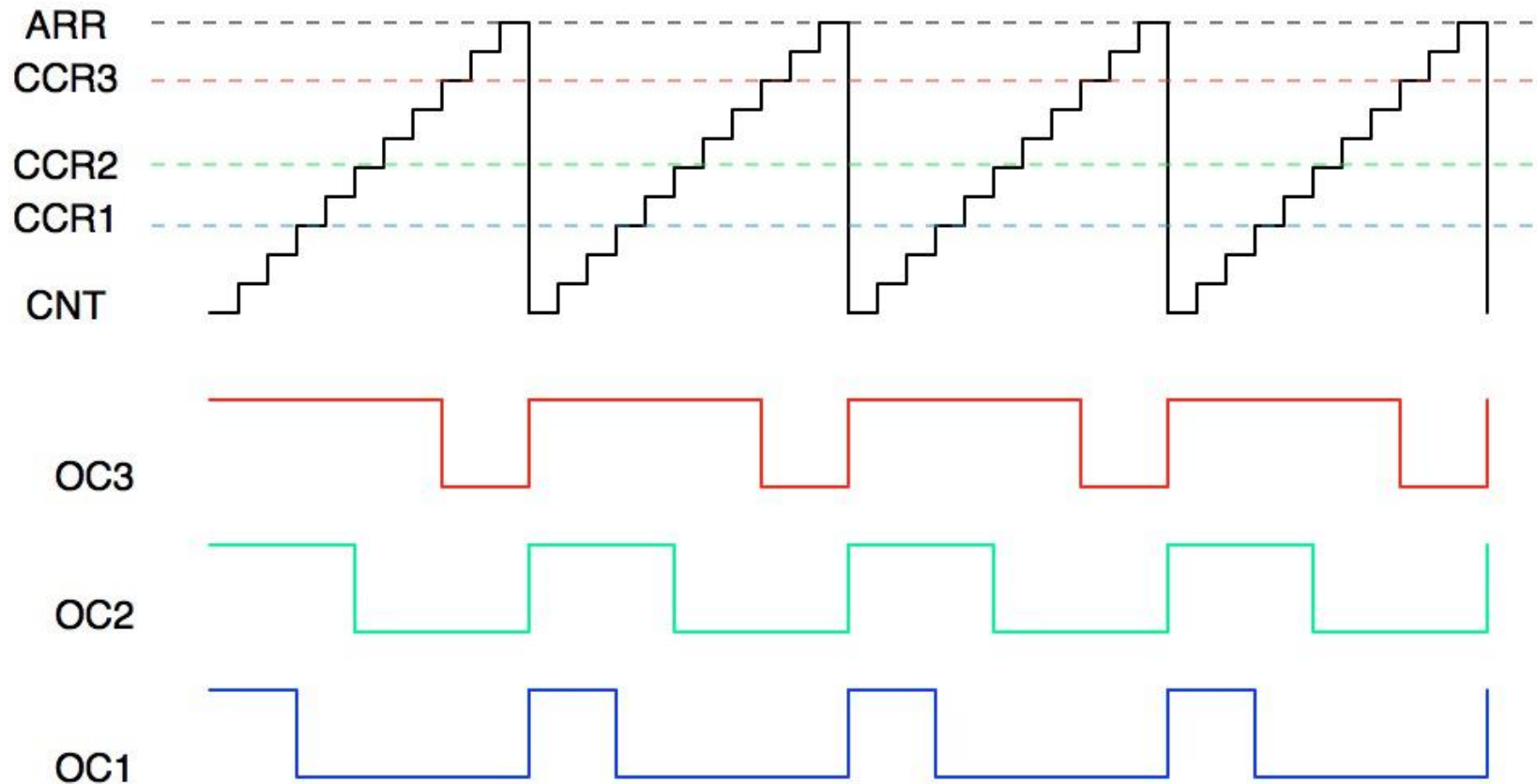
STM32F401RE implements advanced timers. It is possible to use a timer to:

- Generate a delay of a specific time interval
- Generate an interrupt at a fixed frequency
- Generate PWM signals
- Read a quadrature encoder
- ...



Timers - PWM

Three PWM signals from the Output Compare Channels of a general purpose timer



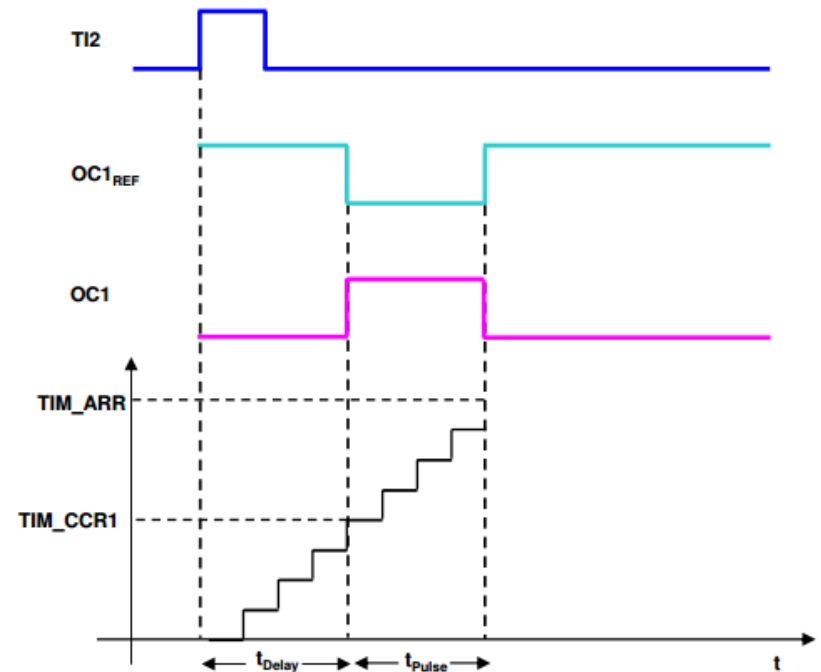
Timers - Pulse Mode

One Pulse Mode (OPM) is a particular case of the previous modes: Output Compare and Input Capture.

It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

There are two One Pulse Mode waveforms selectable by software:

- Single Pulse
- Repetitive Pulse



Timers - SYSTICK

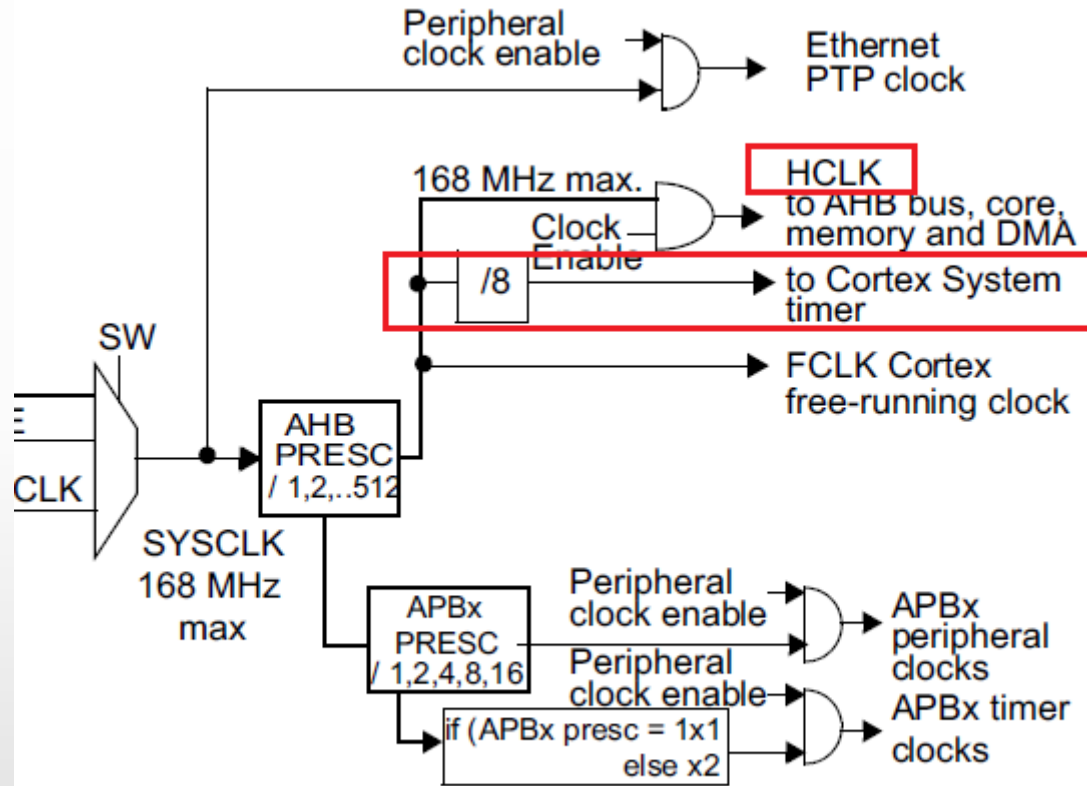
Cortex-M cores support the SYSTICK function.

In brief, the CPU clock source is used to keep track of timing and feeds a timer that can only increment a variable and generate an interrupt every programmable time interval.

It is often used to schedule the execution of functions and to realize delays without using a dedicated timer.

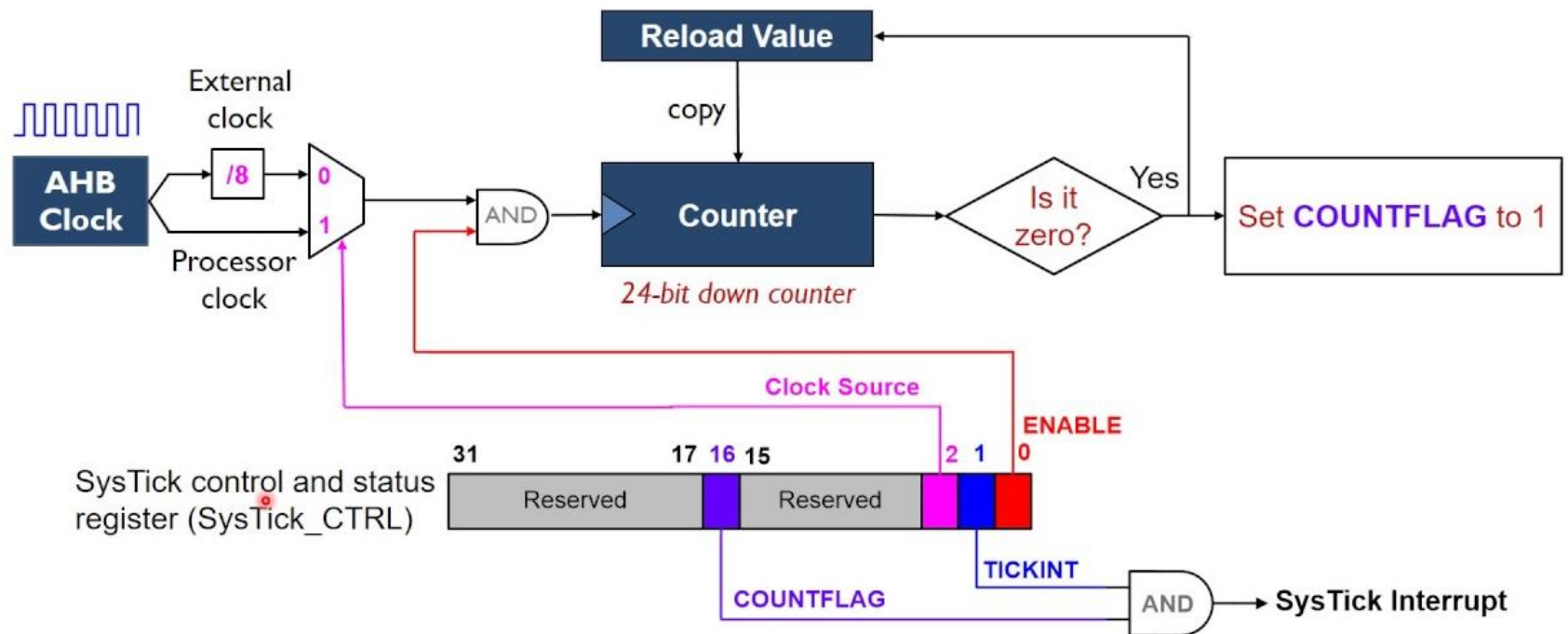


Timers - SYSTICK



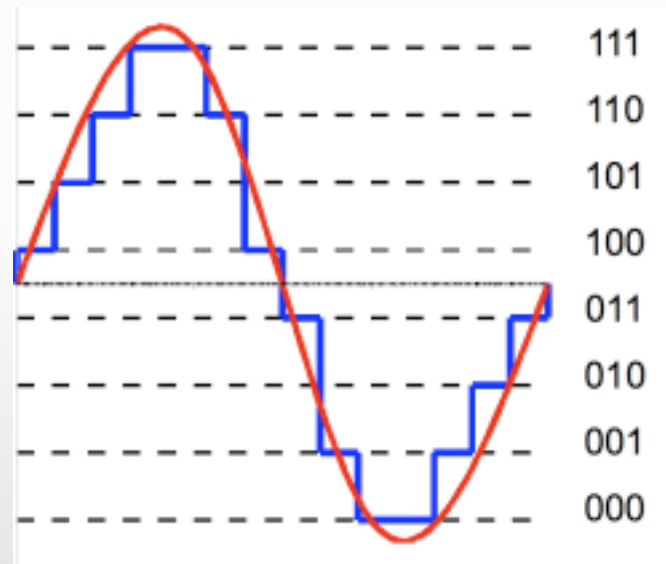
Timers - SYSTICK

Diagram of System Timer (SysTick)



ADC

Analog-to-Digital Converter - is a system that converts an analog signal into a digital signal.



ADC

Physical values are often **analog**. A digital circuit needs to convert them into **digital** values in order to handle them.

Different realizations of an ADC circuit exist:

- Direct-conversion
- Sigma Delta
- Ramp-compare
- Successive-Approximation
- ...



ADC - Resolution

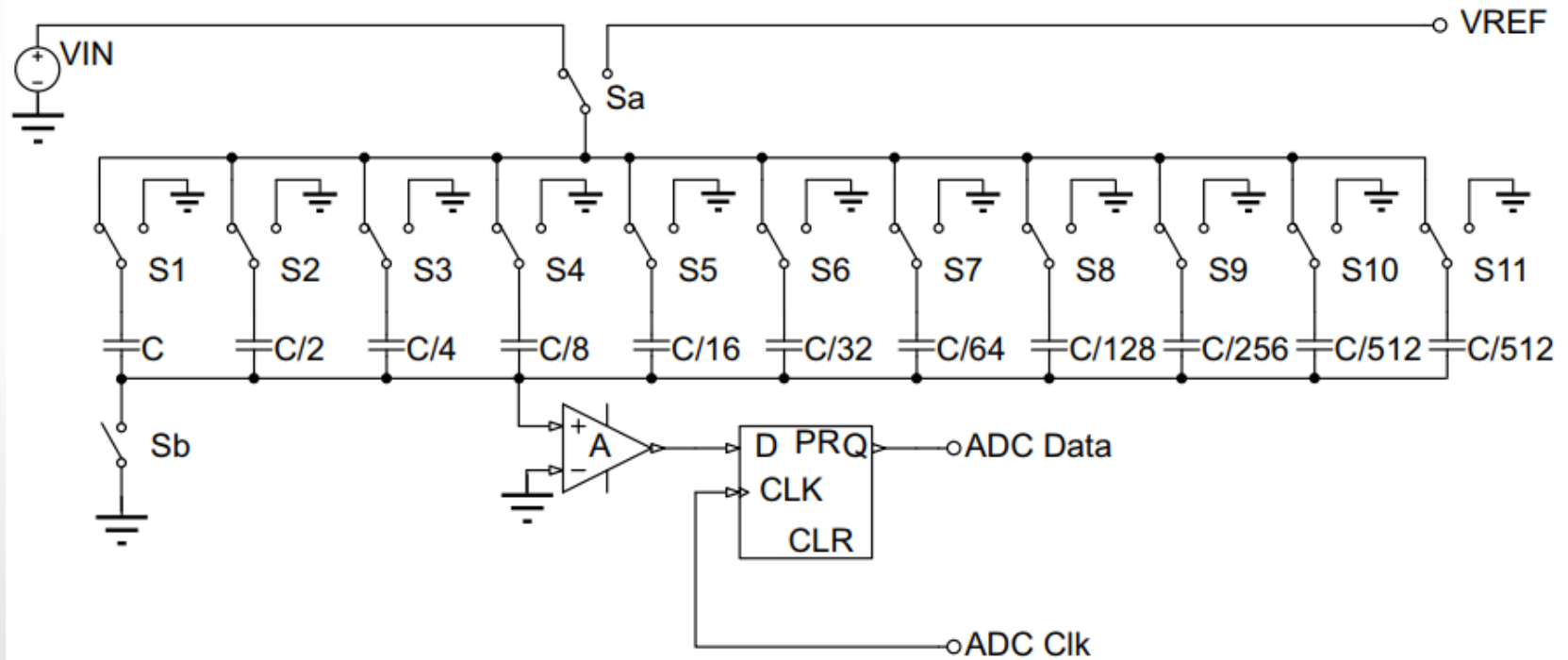
Our MCU has a single ADC with 12bit resolution and 2.4 Msps. 12 bit means that the result of an AD conversion gives up to 4096 different values.

If the working range of our ADC is from 0V to 4.096V, then each bit represents 1 mV.

If the working range of our ADC is from -1.5V to 1.5V, then each bit represents $(1.5 + 1.5) / (2^{12}) = 3 / 4096 = 0.73$ mV.

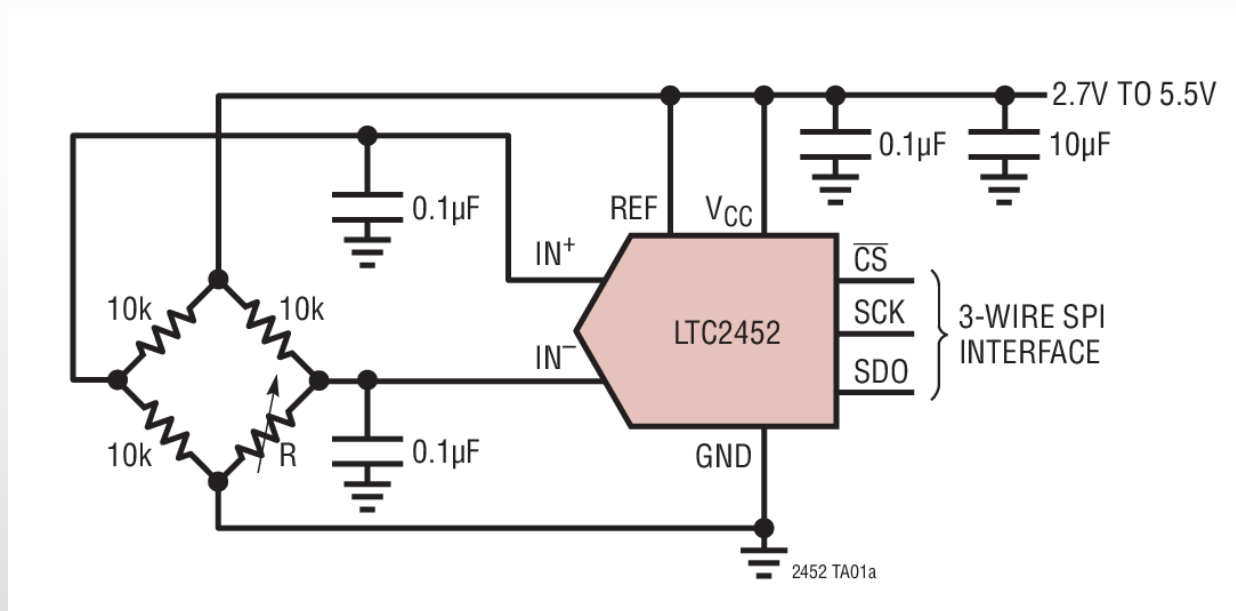


ADC - Successive-Approximation



ADC

For high performance analog applications, it is often mandatory to use an external ADC. In this case the ADC could be connected through I2C, SPI or other communication interfaces.



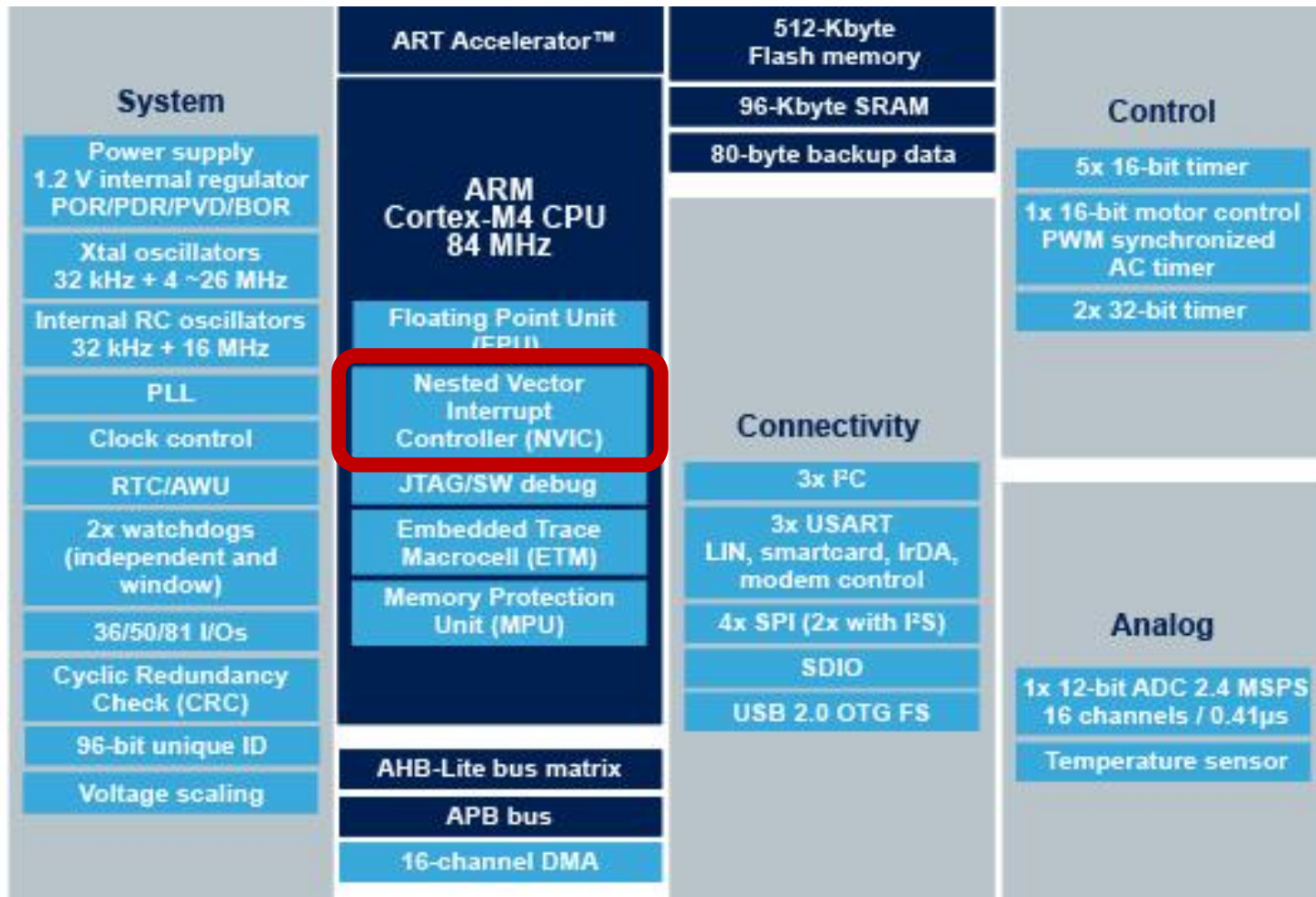
ADC - Sampling sequence

ADC is considered a slow peripheral, thus a reading sequence should follow this approach:

1. Initialize the ADC peripheral
2. Define an interrupt routine
3. Send the sampling command and do other stuff while waiting
4. The interrupt routine will be executed on ADC sampling completed



Interrupts and NVIC



Interrupts

Sometimes an application **MUST** react quickly to events.

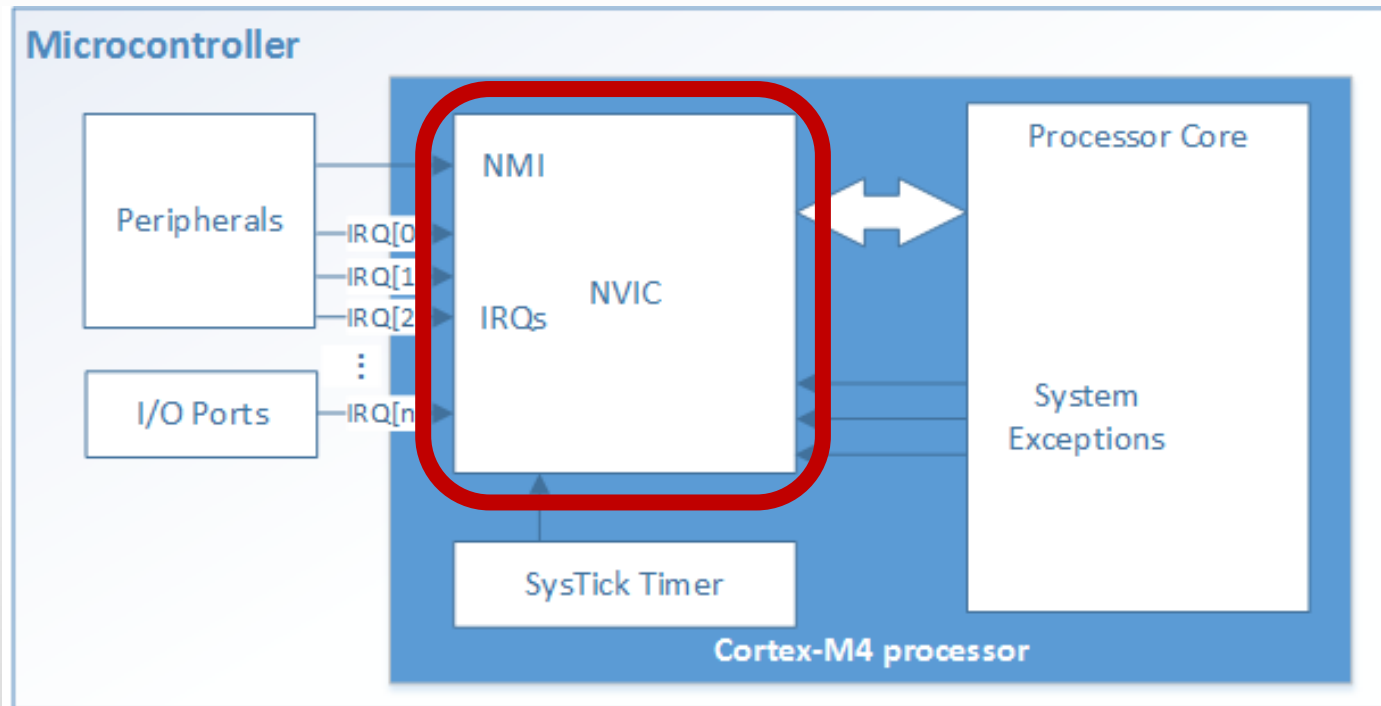
Interrupts associate events like timer overflow, ADC sampling completion or the press of a button to a function.

When an interrupt happens, the current function is stopped and instead the interrupt handler is executed.



NVIC

Nested Vector Interrupt Controller - is an advanced system for handling interrupts and their priorities.



NVIC

All the exceptions are numbered, from 1 to 15 are system exceptions, from 16 to 255 are external interrupts.

Most of them have a programmable priority.

During the execution of an interrupt, another interrupt with lower priority number can preempt the first interrupt.



NVIC

The priority register is 8 bits wide and is subdivided into preempt priority and subpriority.

These two sections can have different lengths according to the priority group that the programmer chooses.

We will focus on NVIC configured to have all bits assigned to preempt priority and no one to sub priority field.

The STM32F4xx series implements the NVIC with 4 bits for priorities, thus 16 possible priority values. It supports 82 different interrupts.



NVIC

- The **preempt priority** level defines whether an interrupt can be serviced when the processor is already running another interrupt handler. In other words, preempt priority determines if one interrupt can preempt another.
- The **subpriority** level value is used only when two exceptions with the same preempt priority level are pending (because interrupts are disabled, for example). When the interrupts are re-enabled, the exception with the lower subpriority (higher urgency) will be handled first.



NVIC

Interrupt signal: from device to CPU

In each peripheral device:

- Each potential interrupt source has a separate **arm (enable)** bit
 - Set for devices from which interrupts, are to be accepted
 - Clear to prevent the peripheral from interrupting the CPU
- Each potential interrupt source has a separate **flag** bit
 - hardware sets the flag when an “event” occurs
 - Interrupt request = (flag & enable)
 - ISR software must clear the flag to acknowledge the request
 - **test flags in software if interrupts not desired**

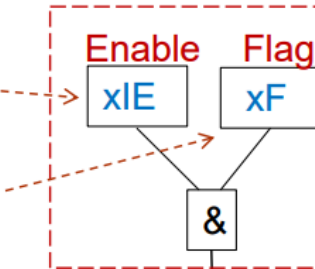
Nested Vectored Interrupt Controller (NVIC)

- Receives all interrupt requests
- Each has an enable bit and a priority within the VIC
- Highest priority enabled interrupt sent to the CPU

Within the CPU:

- Global interrupt enable bit in PRIMASK register
- Interrupt if priority of IRQ < that of current thread
- Access interrupt vector table with IRQ#

Peripheral Device Registers:



Peripheral IRQn

