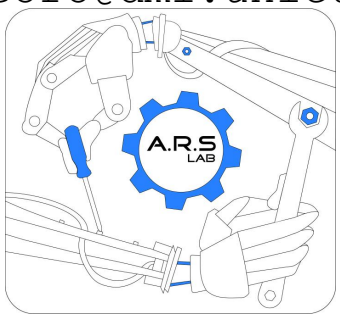
Introduction to Laboratory of Microcontrollered Systems

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



L.S.M. Course

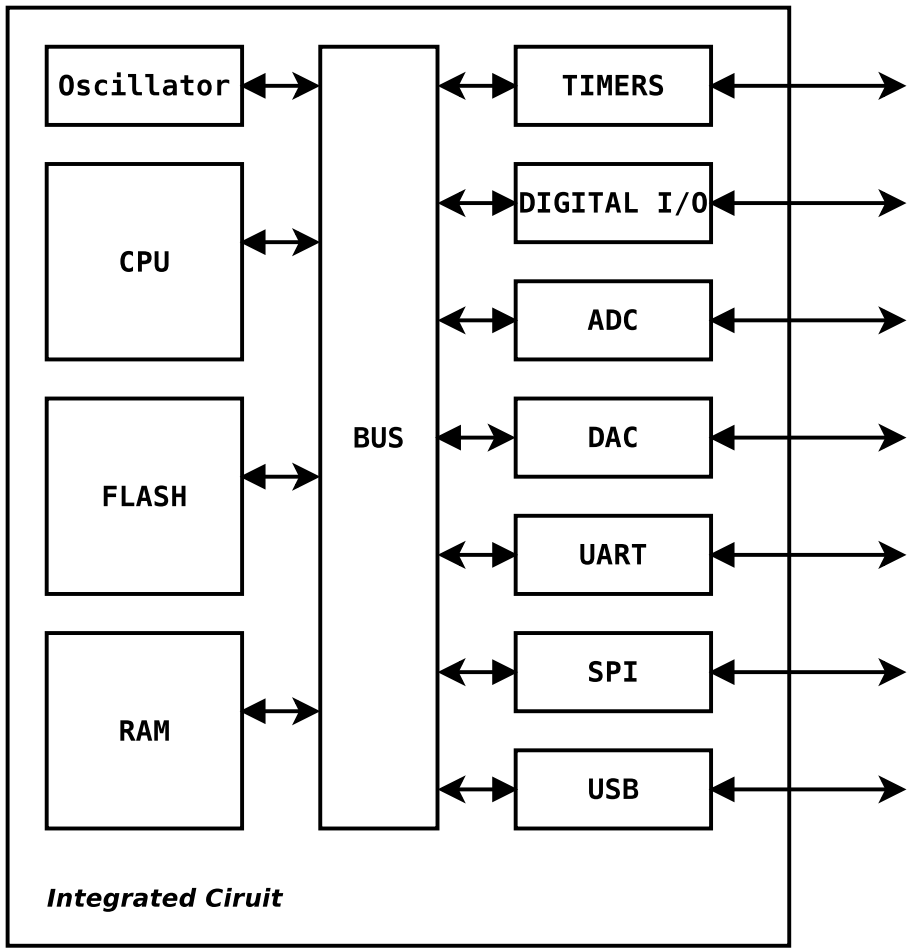
What is a “Microcontroller”?

A **Microcontroller (MCU)** is an integrated circuit including *all parts* of complete computer.
In includes:

- **CPU**
- Built-in **oscillator** for clock source
- **Flash memory** (in the order of KBytes/MBytes), to hold the program (acting as a ROM)
- **RAM**, in the order of KBytes/MBytes
- Several **I/O** peripherals for both generic and specific purposes

In its PINs, a **microcontroller** does not provides the BUS (as in normal CPUs) but the **I/O peripherals**.

What is a “Microcontroller”?



PINs of the integrated circuit

What are the typical peripherals?

- Digital (1-bit) lines
- Analog lines
  - Analog-to-Digital (ADC)
  - Digital-to-Analog (DAC)
- Timers
- Special digital lines (Pulse-Width-Modulation);
- Communication interfaces for other devices and/or sensors/actuators:
  - USB
  - UART (serial port)
  - SPI (Serial Peripheral Interface)
  - I2C (I-square-C)
  - CAN (Controller Area Network)
  - Ethernet
  - ...

Where are microcontrollers employed?

Special-purpose applications/equipments, such as:

- Measurement equipments;
- Cars (i.e. automotive industry, engine control, driver assistance);
- Household Appliances (TV sets, set-top-boxes, DVD, washing machines, microwave ovens, etc.);
- Previous-generation cellphones and smartphones;
- Industrial automation, robotics;
- Domotics, Internet-of-Things;
- ...

How are microcontrollers programmed?

- Generally, they run the software in **bare metal**, i.e. without an operating system.
- In some cases, they host a very small operating system (e.g. **FreeRTOS**) able to offer minimum functionalities: a simple driver layer, no MMU, cooperative or preemptive scheduling
- When the system is programmed in bare metal, the developer has to take care also of programming I/O peripherals

Typical MCU design pattern

```
#include "..."

/* global variables */

int main()
{
    /* initialization of peripherals */

    /* infinite loop */
    for (;;) {
      /* (wait events) */
      /* read inputs */
      /* process data */
      /* write outputs */
    }
}

/* Interrupt Handlers for peripherals */
void xxx_IRQHandler(void)
{
  ...
}
```

Microcontrollers: manufacturers and families

There are many manufacturers of microcontrollers:

- Microchip
- Atmel
- Freescale
- STMicroelectronics
- Intel
- ...

A specific microcontroller (the specific chip) is identified by:

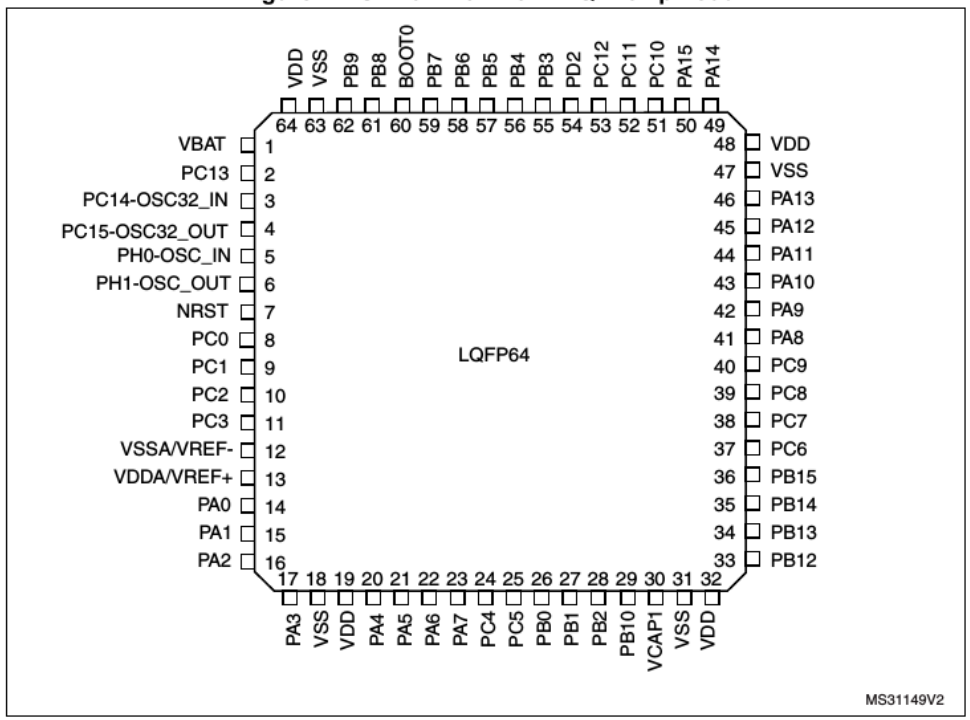
- The **core**, that is the **CPU**: 8-bit, 16-bit, 32-bit, etc.
- The core usually denotes also the **family**
- The amount of **flash memory** and **RAM**
- The **peripherals** which are included in the chip

The MCU we will use!

We will use a MCU of the **`STM32Fx`** family by STMicroelectronics.

- 32-bit ARM-Cortex CPU
- CPU clock from 80 to 240 MHz
- Flash memory from 512K to 2M
- RAM from 512K to 2M
- Several peripherals (digital, ADC, timers, SPI, I^2C, CAN, USB, Ethernet)

Figure 12. STM32F401xD/xE LQFP64 pinout



How can I use/program a MCU?

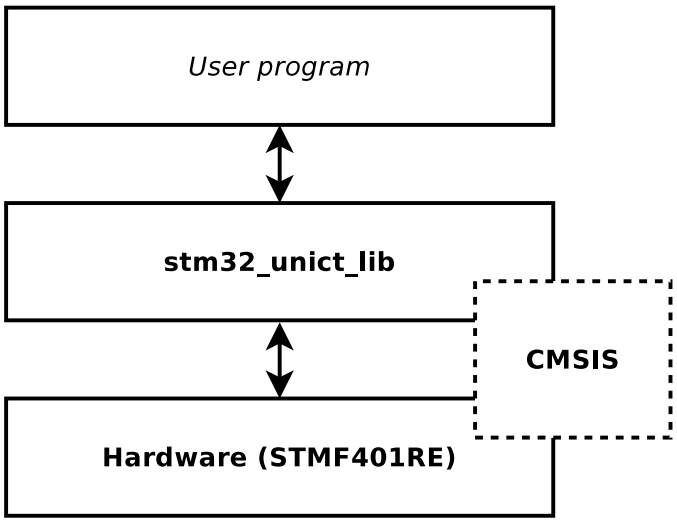
- Using MCU implies to use its peripherals
- Thus, we must learn how to program in C the MCU peripherals
- A certain region of the system memory is **reserved** for peripherals
- In this region, each memory location has a **specific meaning**
- These memory locations are called **Special Function Registers (SFRs)**
- Writing a data into a SFR implies to **program the behaviour of a specific peripheral**

How can I access SFR?

- Using C pointers!! But ...
- The “Cortex Microcontroller Software Interface Standard” (CMSIS) STM32 libraries export a **global variable** for each SFR.
- Therefore, a SFR can be accessed, in C program, by using the relevant variable directly.

The “stm32_unict_lib” Library

- At first, we do not use SFR directly
- Access to hardware will be “mediated” by a library written ad-hoc for the LSM course



Tools for the STM32 microcontroller

- OpenSTM32 IDE (Editor+Compiler): `http://www.openstm32.org`
- A **terminal emulator** program, such as:
  - **minicom** or **cutecom**, for Linux;
  - **picocomlap1** for Linux (see LSM web page);
  - **ZOC Terminal**, for Win and MacOS.
  - **TeraTerm**, for Win.
- The "Data Sheet" of the MCU **STM32F401RE**: `http://www.st.com/`
- The STM32F4Cube Libraries (including the CMSIS): `http://www.st.com/en/embedded-software/stm32cubef4.html`
- The STM32-UNICT-LIB Libraries: `http://www.dmi.unict.it/santoro/index.php?p=13`

Resources for the LSM Course

- Course Web Page
  `http://www.dmi.unict.it/santoro/index.php?p=13`
- The “Web”!

LSM Lessons

- The digital I/O port of an MCU
- Elements of circuit analysis; basics of digital circuits
- Interrupt Management and programming
- Programming models in MCU environments
- Managing time in MCUs: how to program and use timers
- The Analog-to-Digital Converter (ADC)
- Case-Studies
  - Special signal generation: PWM
  - How to drive a servo-motor
  - How to drive a DC motor
  - How to interface digital and analog sensors
  - How to interface I^2C/SPI sensors

Requirements

- Knowledge:
  - Computer Architectures
  - C language
  - Operating Systems
  - Software Engineering
- Skills:
  - **Programming!**
  - English

Exams!

- A **practical exam** with a program to be developed onto a MCU board
- An **oral exam**

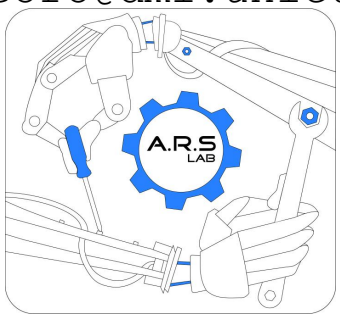
Other initiatives

- **Collaboration with STMicroelectronics**
  - **Internship and Thesis Projects** at STM Labs
- **UNICT-TEAM: The Robotic Student Team of the Unverisity of Catania**
  - **Eurobot Competition**, May 2023 (project start now, volunteers wanted!)

Introduction to Laboratory of Microcontrollered Systems

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



L.S.M. Course