Introduction to L.A.P. 1

Corrado Santoro

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

L.A.P. 1 Course

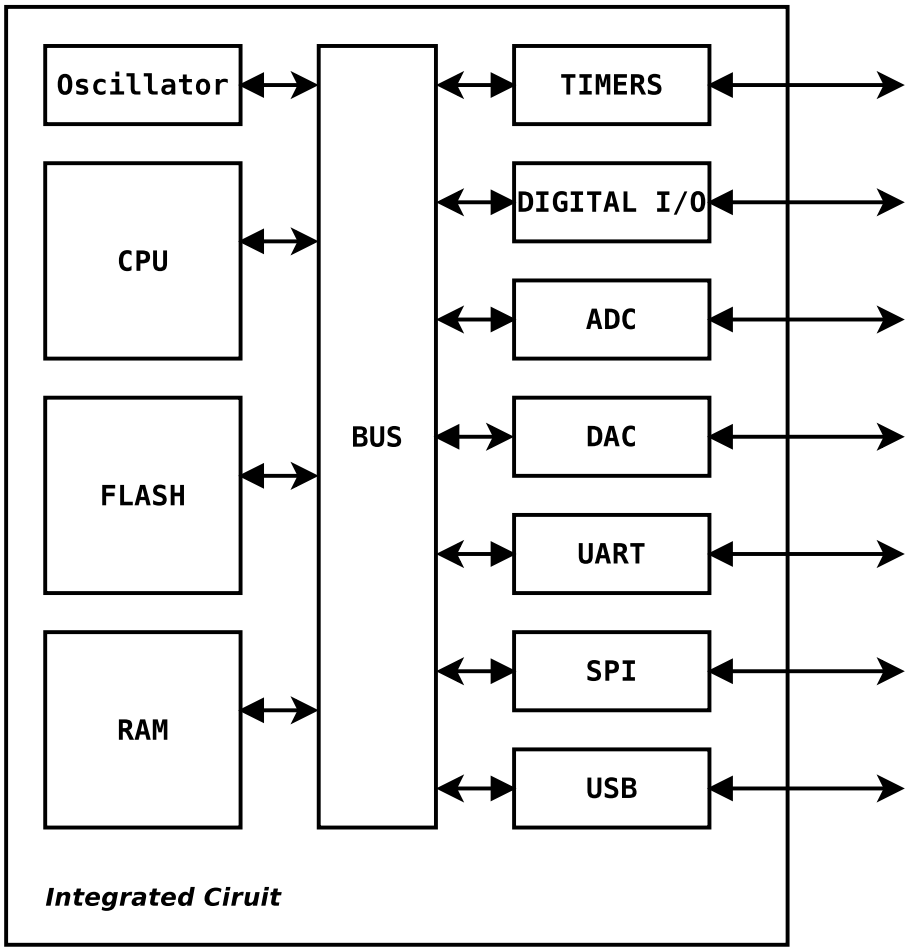
What is a “Microcontroller”?

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
- **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;
- ...

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

How are microcontrollers programmed?

- A specific term exists for MCU software: **firmware**
- Usually they are programmed in C or assembly through a development tool running in a host computer which includes:
  - A compiler
  - A hardware tool to transfer the code into the flash memory
  - An in-circuit debugger (optional)
- When the firmware is written in C, the MCU, at power-up, runs the program directly from the `main()` function.

Microcontrollers: manufacturers and families

There are many manufacturers of microcontrollers:

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

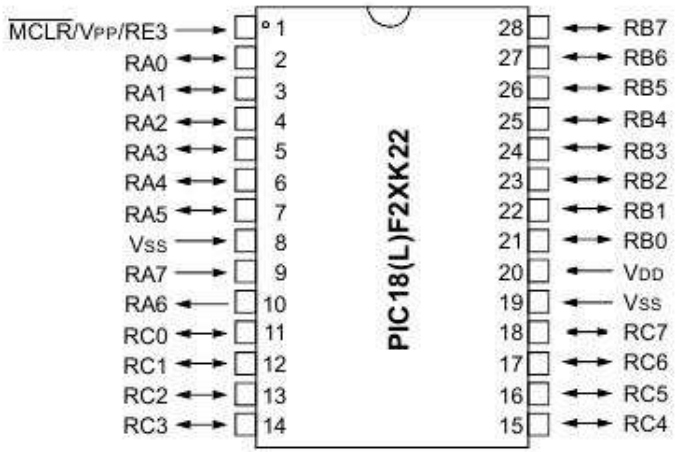
A specific microcontroller (the specific chip) is identified by:

- 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 the MCU **`PIC18F25K22`** by Microchip.

- 8-bit ARM-Cortex CPU
- CPU clock from 8 to 64 MHz
- 32K of flash memory
- 1532 bytes of RAM
- Several peripherals (digital, ADC, timers, SPI, I^2C)

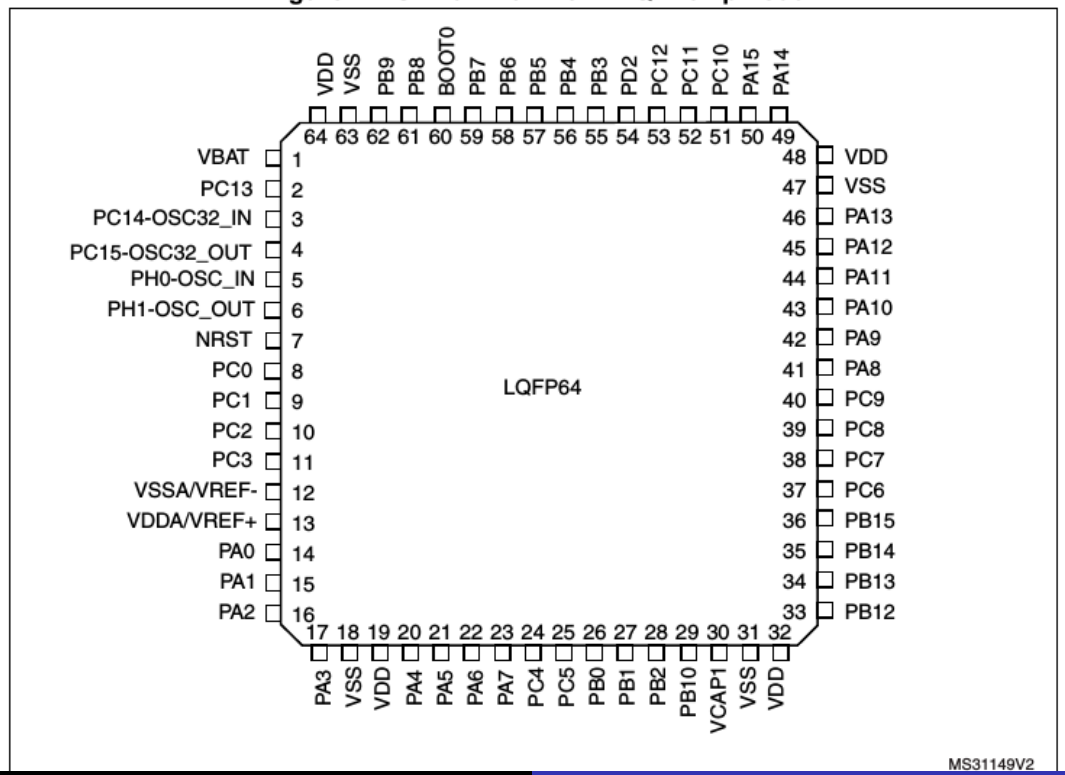


The MCU we will use!

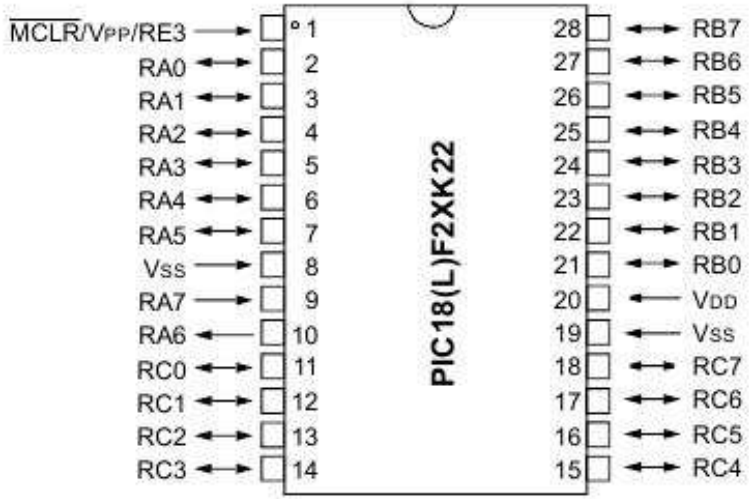
We will use a MCU of the **`STM32Fx`** family by STMicroelectronics.

- 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



The PIC18F25K22



The figure shows the “pinout” of the PIC18F25K22 microcontroller:

- **VDD**, **VSS**. These are the pins to power-up the MCU (the voltage used is 5V or 3.3V)
- **MCLR**. The **Master Clear (RESET)** pin; connecting this pin to ground (VSS, 0V) causes a CPU reset.
- All other pins are relative to **I/O peripherals**.

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**

Some SFR of the PIC18F25K22

TABLE 5-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F2X/4XK22 DEVICES

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FD7h	TMR0H	FAFh	SPBRG1	F87h	—(2)	F5Fh	CCPR3H
FFEh	TOSH	FD6h	TMR0L	FAEh	RCREG1	F86h	—(2)	F5Eh	CCPR3L
FFDh	TOSL	FD5h	T0CON	FADh	TXREG1	F85h	—(2)	F5Dh	CCP3CON
FFCh	STKPTR	FD4h	—(2)	FACh	TXSTA1	F84h	PORTE	F5Ch	PWM3CON
FFBh	PCLATU	FD3h	OSCCON	FABh	RCSTA1	F83h	PORTD(3)	F5Bh	ECCP3AS
FFAh	PCLATH	FD2h	OSCCON2	FAAh	EEADRH(4)	F82h	PORTC	F5Ah	PSTR3CON
FF9h	PCL	FD1h	WDTCON	FA9h	EEADR	F81h	PORTB	F59h	CCPR4H
FF8h	TBLPTRU	FD0h	RCON	FA8h	EEDATA	F80h	PORTA	F58h	CCPR4L
FF7h	TBLPTRH	FCFh	TMR1H	FA7h	EECON2(1)	F7Fh	IPR5	F57h	CCP4CON
FF6h	TBLPTRL	FCEh	TMR1L	FA6h	EECON1	F7Eh	PIR5	F56h	CCPR5H
FF5h	TABLAT	FCDh	T1CON	FA5h	IPR3	F7Dh	PIE5	F55h	CCPR5L
FF4h	PRODH	FCCh	T1GCON	FA4h	PIR3	F7Ch	IPR4	F54h	CCP5CON
FF3h	PRODL	FCBh	SSP1CON3	FA3h	PIE3	F7Bh	PIR4	F53h	TMR4
FF2h	INTCON	FCAh	SSP1MSK	FA2h	IPR2	F7Ah	PIE4	F52h	PR4
FF1h	INTCON2	FC9h	SSP1BUF	FA1h	PIR2	F79h	CM1CON0	F51h	T4CON
FF0h	INTCON3	FC8h	SSP1ADD	FA0h	PIE2	F78h	CM2CON0	F50h	TMR5H
FEFh	INDF0(1)	FC7h	SSP1STAT	F9Fh	IPR1	F77h	CM2CON1	F4Fh	TMR5L
FEEh	POSTINC0(1)	FC6h	SSP1CON1	F9Eh	PIR1	F76h	SPBRGH2	F4Eh	T5CON
FEDh	POSTDEC0(1)	FC5h	SSP1CON2	F9Dh	PIE1	F75h	SPBRG2	F4Dh	T5GCON
FECh	PREINC0(1)	FC4h	ADRESH	F9Ch	HLVDCON	F74h	RCREG2	F4Ch	TMR6

The meaning of each SFR is reported in the programmer's manual of the specific MCU

How can I access SFR?

- Using C pointers!! But ...
- The Microchip C compiler exports a **global variable** for each SFR, whose name matches the name of the SFR.
- Therefore, a SFR can be accessed, in C program, by using the relevant variable directly.

Resources for the PIC microcontroller

- Microchip MPLABX IDE:
  `http://www.microchip.com/mplabx`
- Microchip C compiler for 8-bit MCU, XC8:
  `http://www.microchip.com/mplabx`
- A **terminal emulator** program, such as:
  - **minicom** or **cutecom**, for Linux;
  - **picocomlap1** for Linux (see LAP1 web page);
  - **ZOC Terminal**, for Win and MacOS.
  - **TeraTerm**, for Win.
- The “Data Sheet” of the MCU **PIC18F25K22**:
  `http://www.microchip.com/`

Resources for the STM32 microcontroller

- OpenSTM32 IDE: `http://www.openstm32.org`
- The “Data Sheet” of the MCU **STM32Fx**: `http://www.st.com/`

Resources for the LAP1 Course

- Course Web Page (from `http://www.dmi.unict.it/~santoro`)
- A reference book: Dogan Ibrahim, *PIC Microcontroller Projects in C - Basic to Advanced*, Newnes
- The “Web”!

LAP1 Program

- Elements of circuit analysis; basics of digital circuits
- The digital I/O port of an MCU
- Programming models in MCU environments
- Managing time in MCUs: how to program and use timers
- Interrupt Management and programming
- Communication Interfaces in MCU: UART, I^2C, SPI
- 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
- Skills:
  - **Programming!**
  - English

Exams!

- A **test** which several questions (6–10) with both open answers and multiple choices
- A **practical exam** with a program to be developed onto a MCU board
- An optional **course project**

Introduction to L.A.P. 1

Corrado Santoro

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

L.A.P. 1 Course