Institut Universitaire de Technologie
Aix-Marseille Université

Embedded development using Arduino

Information Technology - 2nd year, H2

Instructor

Thibaut VIARD

- IUT Informatique option Systèmes Industriels (Reims)
- DU Image, Infographie et Communication (Reims)
- IUP Génie Mathématique et Informatique (Avignon)

Works in embedded since 2000 (Avionics, SmartCard, Digital TV/Set Top Box). Joined Atmel in 2009 as ARM Application Engineer. Involved in software deliveries for Cortex-M devices and managed the ports of Arduino Due and Arduino Zero.

Institut Universitaire de Technologie
Aix-Marseille Université

What will you learn?

You will discover what Embedded means, what a microcontroller is and composed of and finally how to develop simple applications on a flash MicroController Unit (MCU).

These basis will allow you to easily go further on more complex applications.

1. Introduction
2. Fundations
3. ARM Cortex-M families
4. ARM Cortex-M0+
5. Software API
6. Tools
7. Atmel WINC1500-XSTK
8. Arduino

What means embedded?

“An embedded system is a microprocessor based system that is built to control a function or a range of functions.”, Steve HEATH, Embedded Systems Design, 2003. Microprocessor and much of external peripherals have been integrated into a microcontroller.

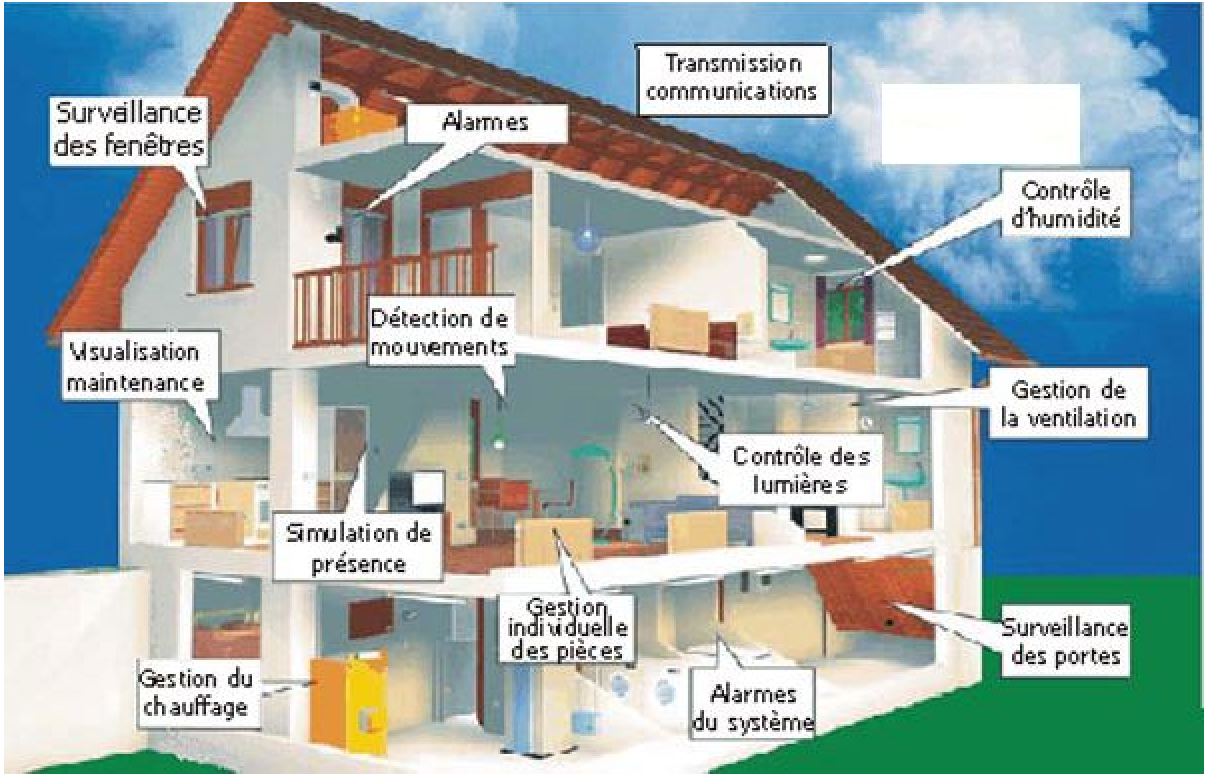
The microcontroller is connected to external peripherals like sensors, actuators, codecs and external storage.

Embedded systems are designed with efficiency constraints:

- Components placement
- Bill of Material
- Energy consumption

Embedded example 1/2

Embedded example 2/2



Why embedded (for you)?

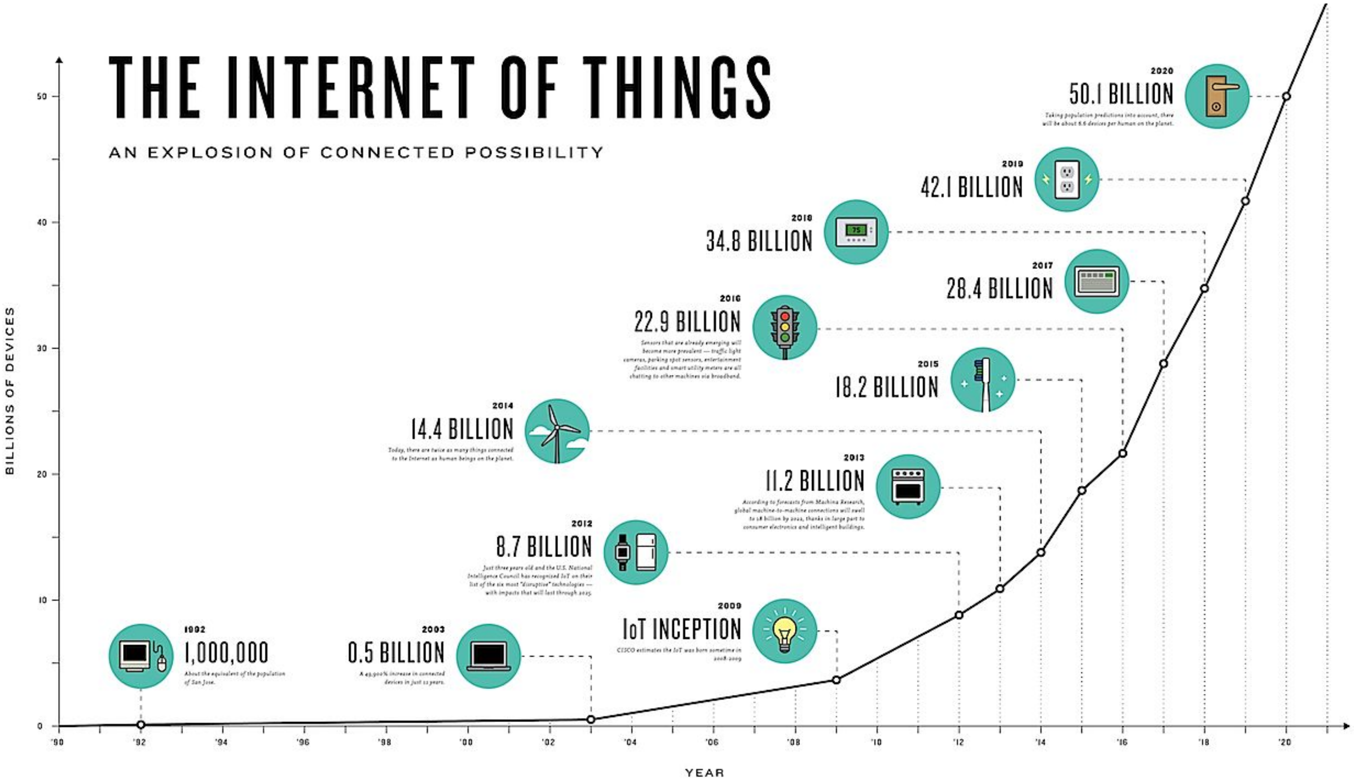
Embedded systems exponential growth is planned due to Internet of Things.

IoT will need much more embedded developers!

Embedded classes will allow you to be prepared to constrained systems using high level algorithms.

You will gain added value and improve valorization of diploma while looking for a job.

IoT predicted growth



1. Introduction
2. Fundations
3. ARM Cortex-M families
4. ARM Cortex-M0+
5. Software API
6. Tools
7. Atmel WINC1500-XSTK
8. Arduino

What is a microcontroller?

A **microcontroller** (sometimes abbreviated **µC**, **uC** or **MCU**) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.

Two common kinds of microcontroller architectures are instantiated: Von Neumann and Harvard.

Common point is system bus and peripheral bus, allowing CPU core to communicate with either memories or peripherals.

Microcontroller core architectures

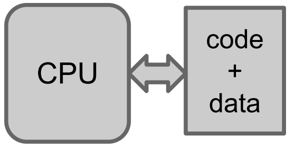
Question

What is the difference between Harvard Architecture and von Neumann Architecture?

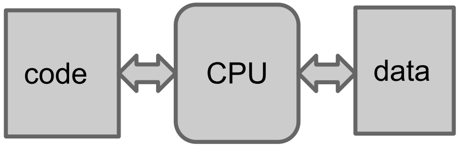
Answer

The name **Harvard Architecture** comes from the Harvard Mark I relay-based computer. The most obvious characteristic of the Harvard Architecture is that it has physically separate signals and storage for code and data memory. It is possible to access program memory and data memory simultaneously. Typically, code (or program) memory is read-only and data memory is read-write. Therefore, it is impossible for program contents to be modified by the program itself.

The **von Neumann Architecture** is named after the mathematician and early computer scientist John von Neumann. von Neumann machines have shared signals and memory for code and data. Thus, the program can be easily modified by itself since it is stored in read-write memory.



Von Neumann architecture

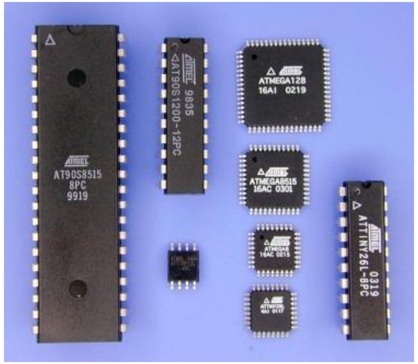


Harvard architecture

8 bitters

Flash MCU

- Atmel AVR



- Microchip PIC

32 bitters

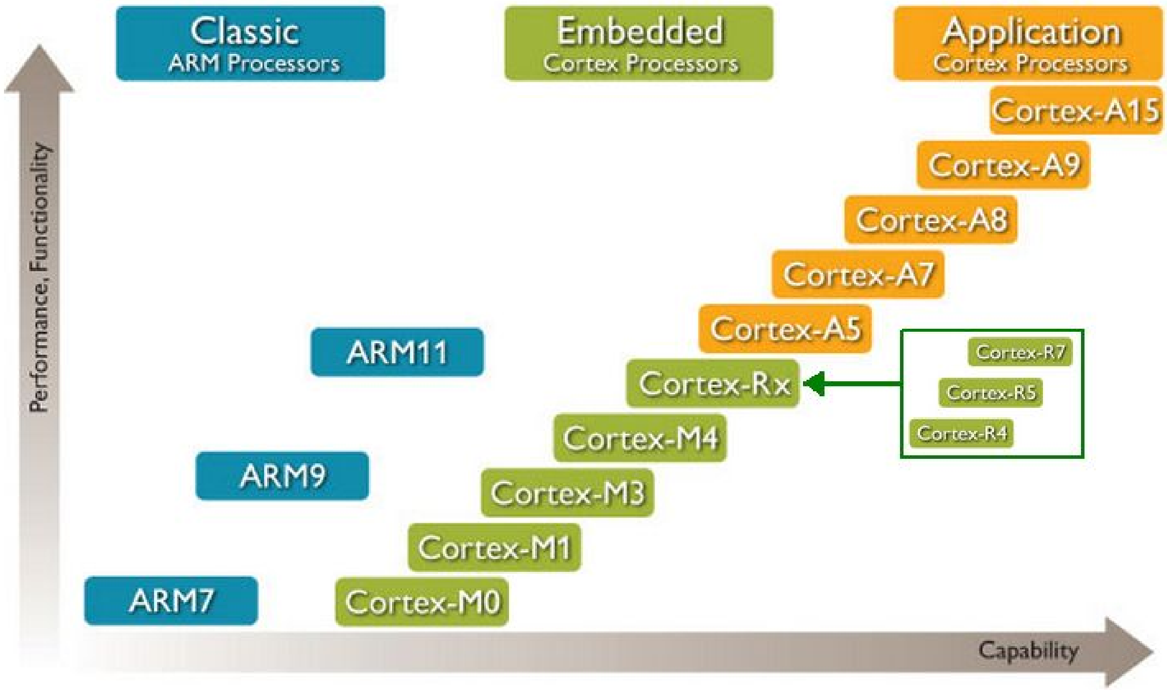
With embedded flash (MicroControllers Units)

- ARM Cortex-M families
- Mips 4k
- Renesas

With external storage (Application processors)

- ARM Cortex-A, 7, 9, 11
- Mips

ARM cores families



ARM Cortex-A

ARM Cortex-R

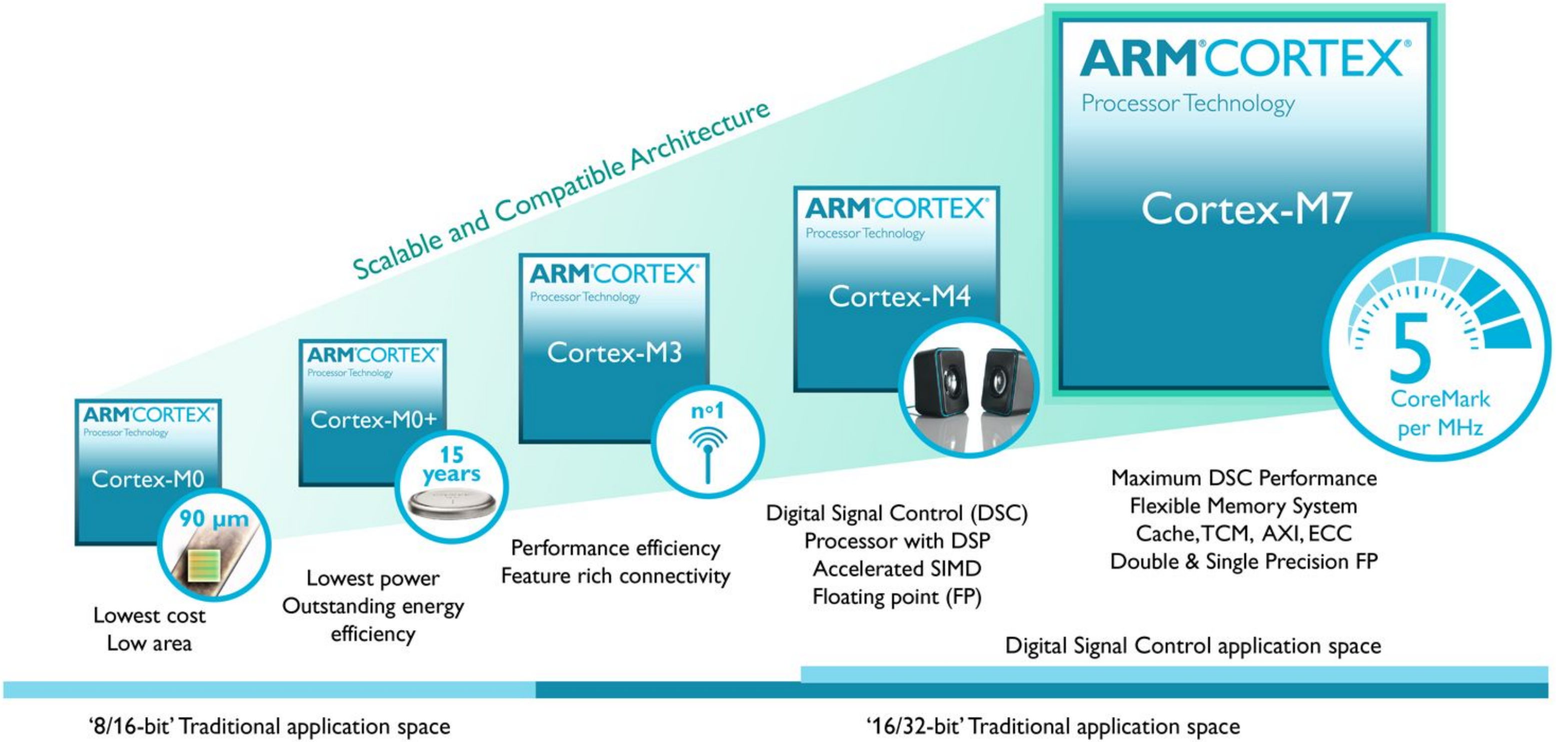

Cortex
Low-Power Leadership from ARM
High density storage
Mobile baseband
Automotive systems
3G
4G
HSPA
Lte
ARM® Cortex™ -R Real-time embedded processors
Cortex-R4 (2006)
• High-performance, real-time
• Deterministic interrupts
• Feature set configurable
• Dependable systems
Cortex-R5 (2011)
• Performance enhancing features
• Fast peripheral access
• I/O coherency
• Dual core configuration
• Extended error management
• Space-saving FPU
Cortex-R7 (2011)
• Large performance increase
• Advanced microarchitecture
• Higher clock frequency
• Quality of Service features
• Symmetric Multi-Processing
• Twin core and I/O coherency
• Extended real-time memory
• Hard error management
• Integrated interrupt controller

1. Introduction
2. Fundations
3. ARM Cortex-M families
4. ARM Cortex-M0+
5. Software API
6. Tools
7. Atmel WINC1500-XSTK
8. Arduino

Why ARM Cortex-M?

- Most available in volume
- Most versatile (many packages, from low-power to high processing)
- Most used and documented

ARM Cortex-M family



ARM Cortex-M family

Low-end (Von Neumann architecture)

- Cortex-M0/M0+

Middle end (Harvard architecture)

- Cortex-M3/M4

High end (Harvard architecture)

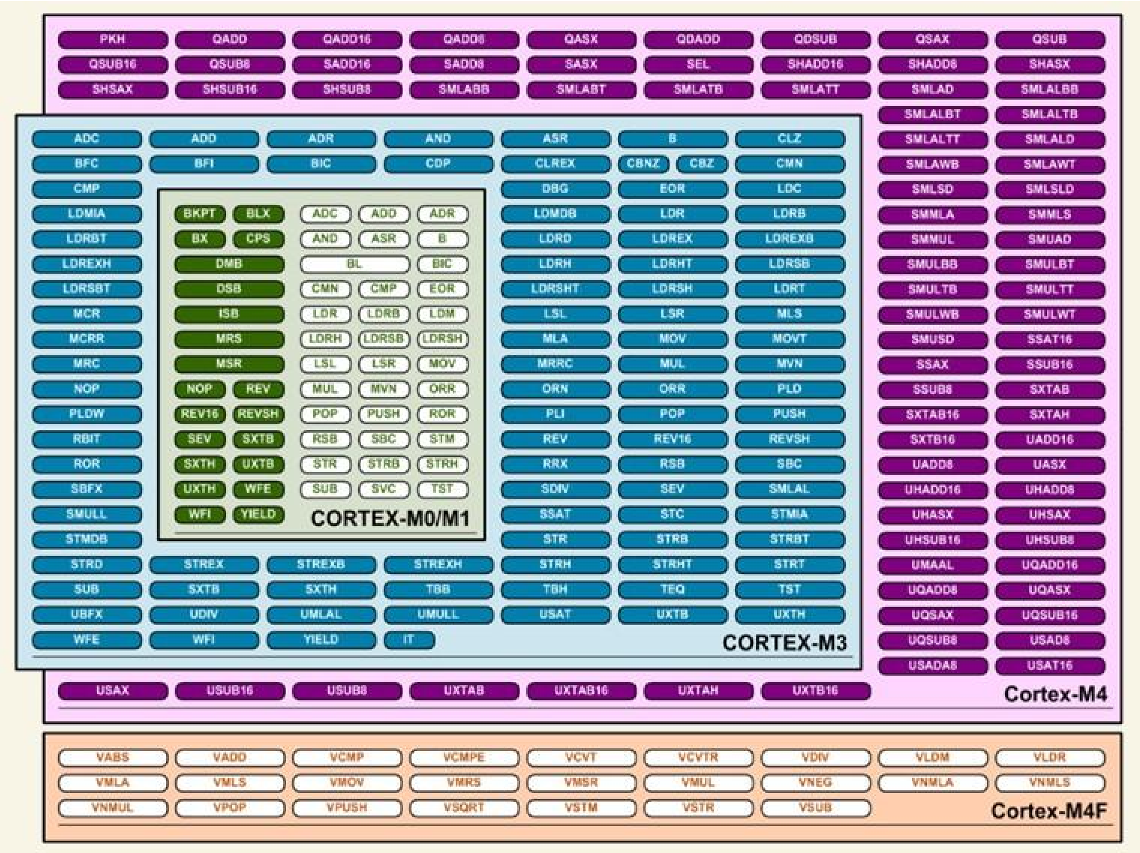
- Cortex-M7

Numbers for 2013: **2.9 billion of Cortex-M** devices sold out of 10 billion ARM global

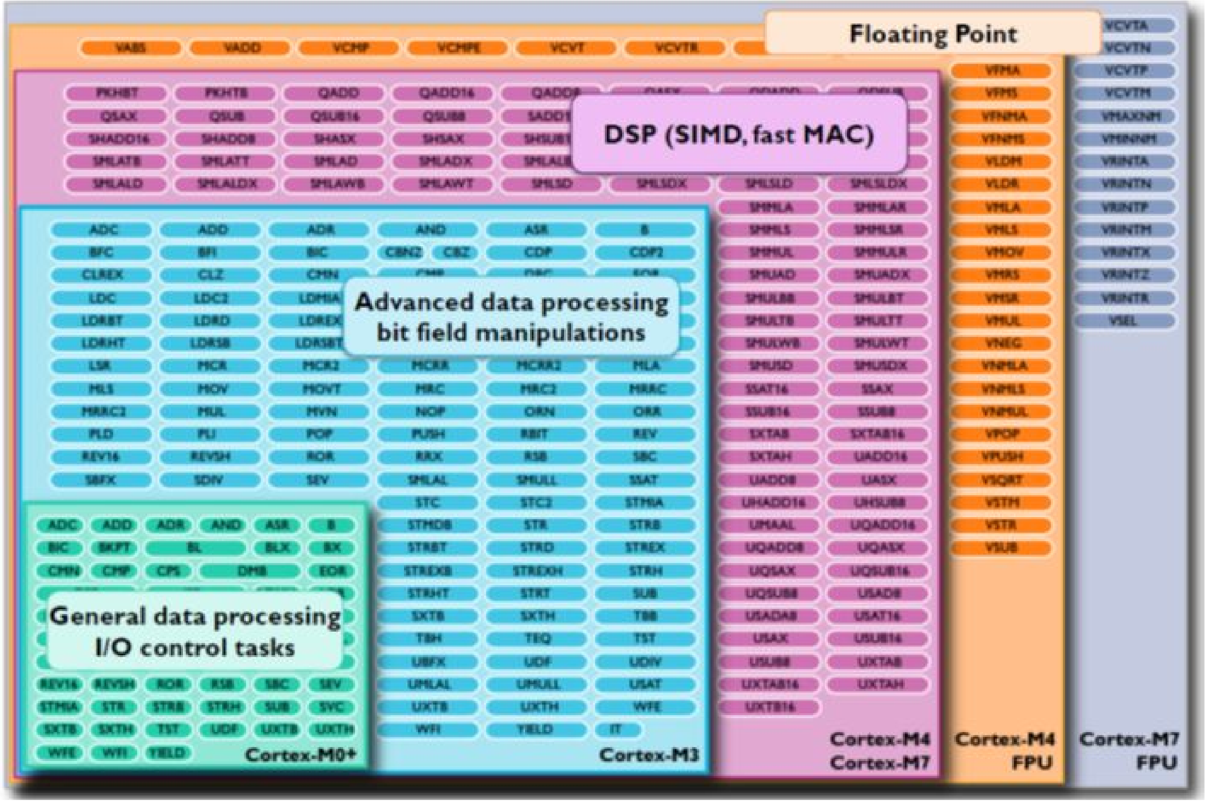
ARM Cortex-M family

	ARM Cortex-M0	ARM Cortex-M0+	ARM Cortex-M3	ARM Cortex-M4	ARM Cortex-M7
Architecture	ARMv6-M	ARMv6-M	ARMv7-M	ARMv7-M	ARMv7-M
CoreMark 1.0 / MHz	1.62	1.77	2.17	2.19	5
Typical DMIPS @ MHz	0.84	0.93	1.25	1.25	
Max Number of IRQ	32 + NMI	32 + NMI	240 + NMI	240 + NMI	
System exceptions	4	4	8	8	
Fault Handling exceptions	1 (HardFault)	1 (HardFault)	4 (HardFault + 3 other handlers)	4 (HardFault + 3 other handlers)	
Exception Priority levels	4(prog) + 2 (fixed)	4(prog) + 2 (fixed)	8 to 256(prog) +2(fixed)	8 to 256(prog) +2(fixed)	
Interrupt Latency (cycles)	16	15	12	12	
Register accesses	32-bit	32-bit	8/16/32-bit	8/16/32-bit	
SysTick timer	Yes (Optional)	Yes (Optional)	Yes	Yes	

Thumb-2 instruction set



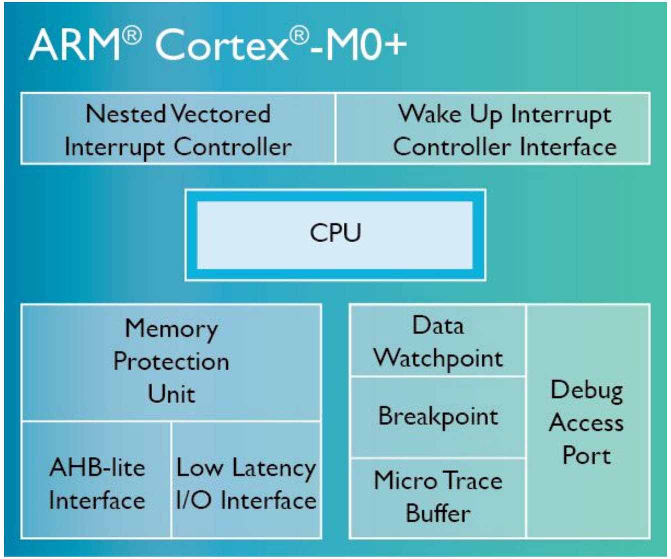
Thumb-2 instruction set



1. Introduction
2. Fundations
3. ARM Cortex-M families
4. ARM Cortex-M0+
5. Software API
6. Tools
7. Atmel WINC1500-XSTK
8. Arduino

Cortex-M0+ core and peripherals

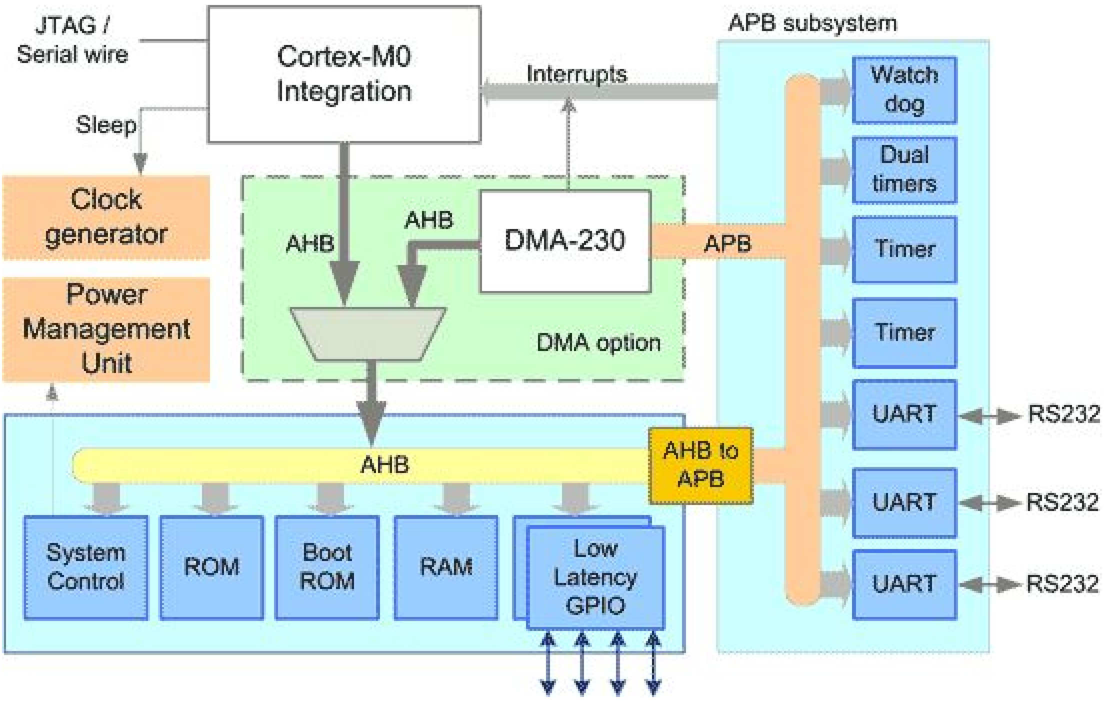
- Systick (System timer)
- NVIC (Nested Vector Interrupt Controller)
- SCB (System Control Block)
  - VTOR (Vector Table Offset Register)
- DWT (Data Watchpoint Timer)
- MTB (Micro Trace Buffer)
- MPU (Memory Protection Unit)



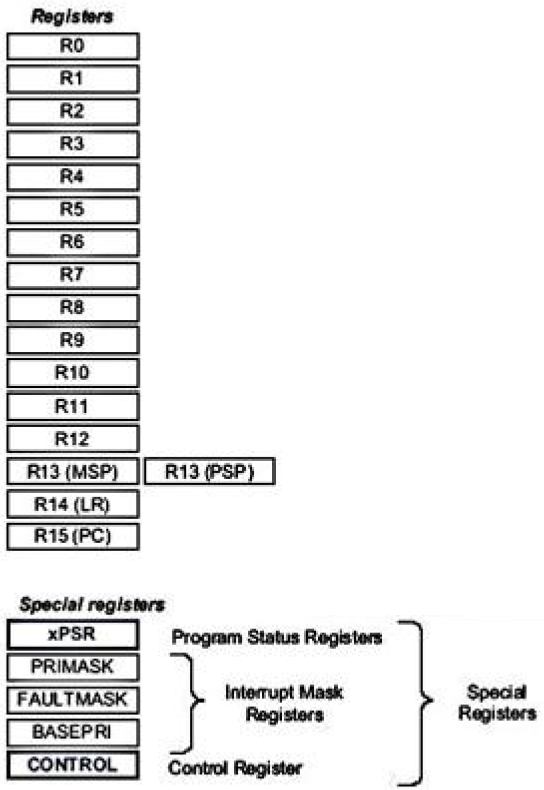
Cortex-M0+ buses

- I/D bus (instruction/data)
- Amba Peripheral Bus (APB) - slow peripherals, peripheral subsystem
- Amba High-performance Bus (AHB) - high speed peripherals

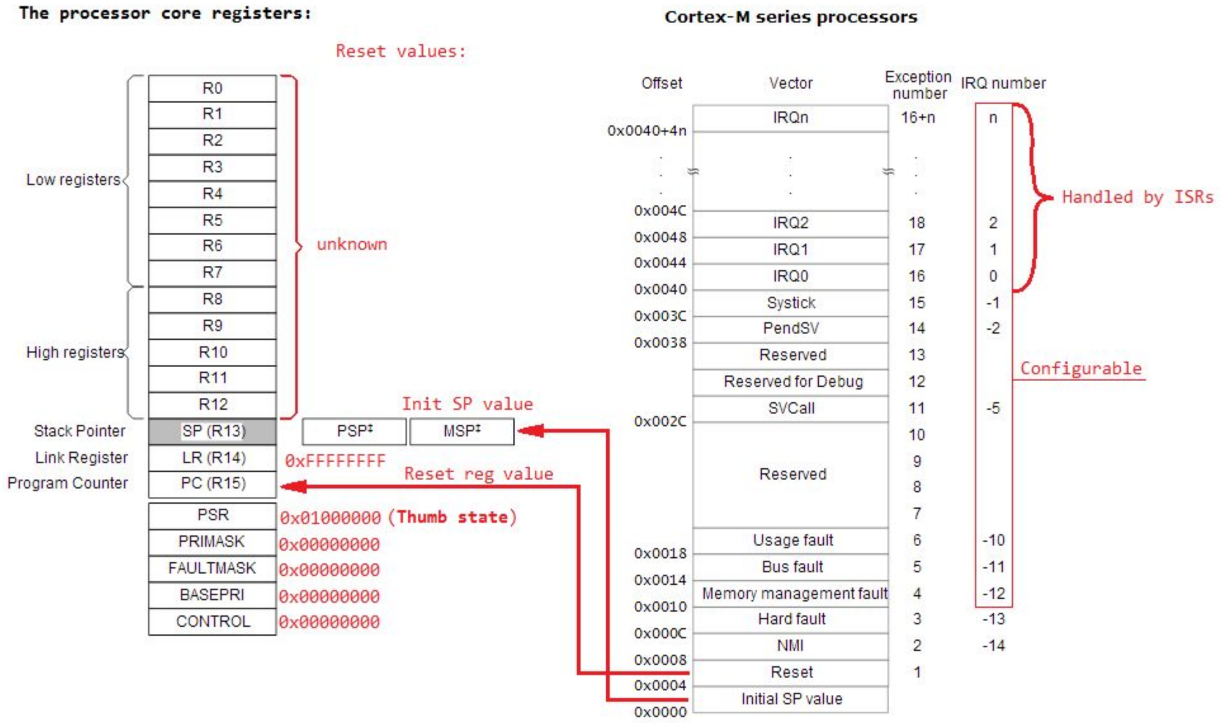
Cortex-M0+ integration



Cortex-M0+ registers



Cortex-M0+ boot sequence



1. Introduction
2. Fundations
3. ARM Cortex-M families
4. ARM Cortex-M0+
5. Software API
6. Tools
7. Atmel WINC1500-XSTK
8. Arduino

ARM CMSIS overview

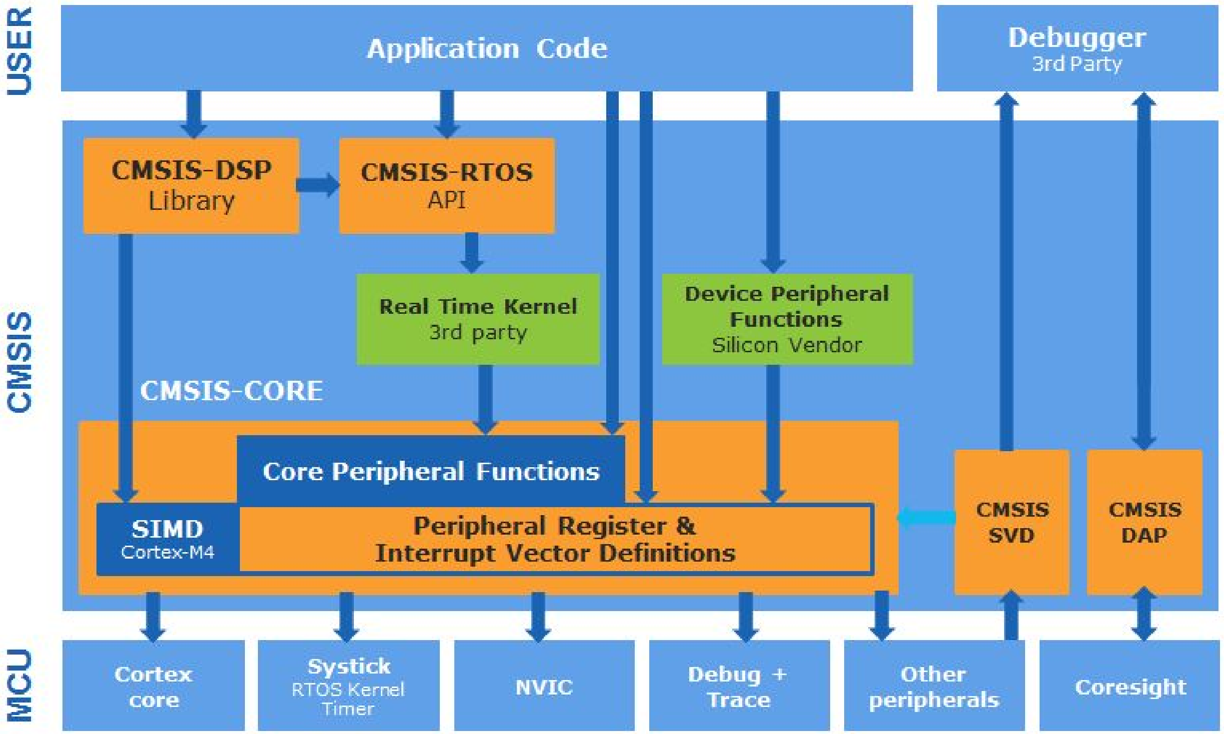
ARM® CONNECTED™

CMSIS COMPLIANT
ARM® Cortex™ Microcontroller Software Interface Standard

Cortex Microcontroller Software Interface Standard

- C language abstraction layer for all Cortex-M processor-based devices
- Developed in conjunction with silicon, tools and middleware partners
- Benefits to the embedded developer
  - Consistent software interfaces for silicon and middleware vendors
  - Simplifies re-use across Cortex-M processor-based devices
  - Reduces software development cost and time-to-market
  - Reduces learning curve for new Cortex microcontroller developers

CMSIS-based application architecture



CMSIS debug tools

CMSIS-SVD	• System View Description XML schema, describes the SoC and his peripheral API. These files are used by debuggers. • SVDConv tool: SVD check and headers generation.
CMSIS-DAP	• Allows low-cost core debug for Silicon Vendors using their own devices. • Open source on GitHub since November, 2013.

CMSIS low-level software components

CMSIS-CORE	Core peripherals register descriptions and API (NVIC, SCB, SysTick, ITM, DWT, …).
CMSIS-DRIVER	• It is intended to standardize the Peripheral Driver Interfaces as part of a future CMSIS release. • Could be subject to changes soon.
CMSIS-PACK	• The Pack Concept embeds every information from device to board needed for application development. • Composed of **Device Family Pack** and **Board Support Pack**

CMSIS high-level software components

CMSIS-DSP	This API provides SIMD instructions. Using Core instructions for CM4/CM7(F) and software for CM0/CM0+/CM3.
CMSIS-RTOS	This API provides RTOS abstraction by defining common ressources and functions (Abstraction layer).
Mbed	Although not initially based on CMSIS, Mbed turned open source (Github) in 2013 and community integrated CMSIS-RTOS.

ARM CMSIS package V3.x+

- CMSIS-Core API (CM0, CM0+, CM3, CM4): core peripherals descriptions and access (NVIC, SCB, SysTick, ITM, DWT, …).
  - CMSIS link
- CMSIS-DSP API and examples, libraries in binary format.
- CMSIS-SVD schemas and SVDConv tool.
- CMSIS-RTOS abstraction layer definition (cmsis_os.h)
  - Keil RTX port is provided under BSD-like license with some examples.
- Whole CMSIS API Doxygen-generated HTML documentation.
- CMSIS-DAP available in a dedicated distribution (GitHub).

<Vendor> CMSIS package

- CMSIS/
  - Device/
    - <Vendor>/
      - <device series>/
        - include/
          - <part>.h
          - system_<series>.h
        - source/
          - system_<series>.c
          - <toolchain>/
            - <part[x]>_flash.ld
            - <part[x]>_sram.ld
            - startup_<series>.c

- Init interface
- Initialization of device clock, mainly
- <toolchain>/
  - Linker scripts for flash and ram applications
  - Boot steps and vectors handlers

1. Introduction
2. Fundations
3. ARM Cortex-M families
4. ARM Cortex-M0+
5. Software API
6. Tools
7. Atmel WINC1500-XSTK
8. Arduino

Institut Universitaire de Technologie
Aix-Marseille Université

Embedded development tools

- ARM DS5 (Eclipse based) and ARM/Keil µVision
- IAR EWARM
- Atollic TrueSTUDIO (Eclipse based)
- Rowley CrossStudio
- Eclipse CDT + GNU ARM plugin
- Atmel Studio, Freescale Kinetis Design Studio (Eclipse based), TI Code Composer Studio (Eclipse based), STM32 Cube (Eclipse based)
- GCC command line or integrated in a IDE/PDE

Embedded debugger

A.K.A. ICE: In Circuit Emulator

- Segger JLink
- IAR JtagJet
- ARM ULink & D-Stream
- ARM CMSIS-DAP, open source project
- Black Magic Probe, open source project
- etc...

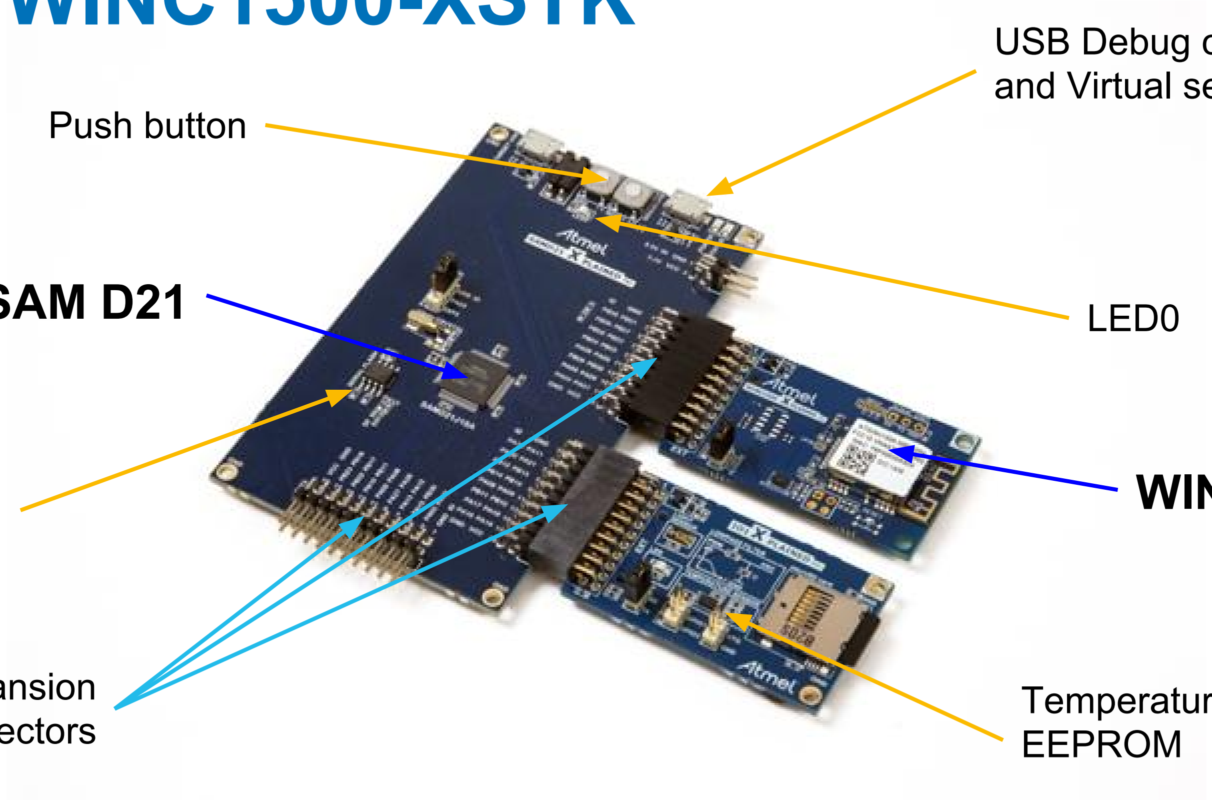
Product datasheet

The product datasheet is the bible containing all information you need to use the microcontroller chosen for a specific application:

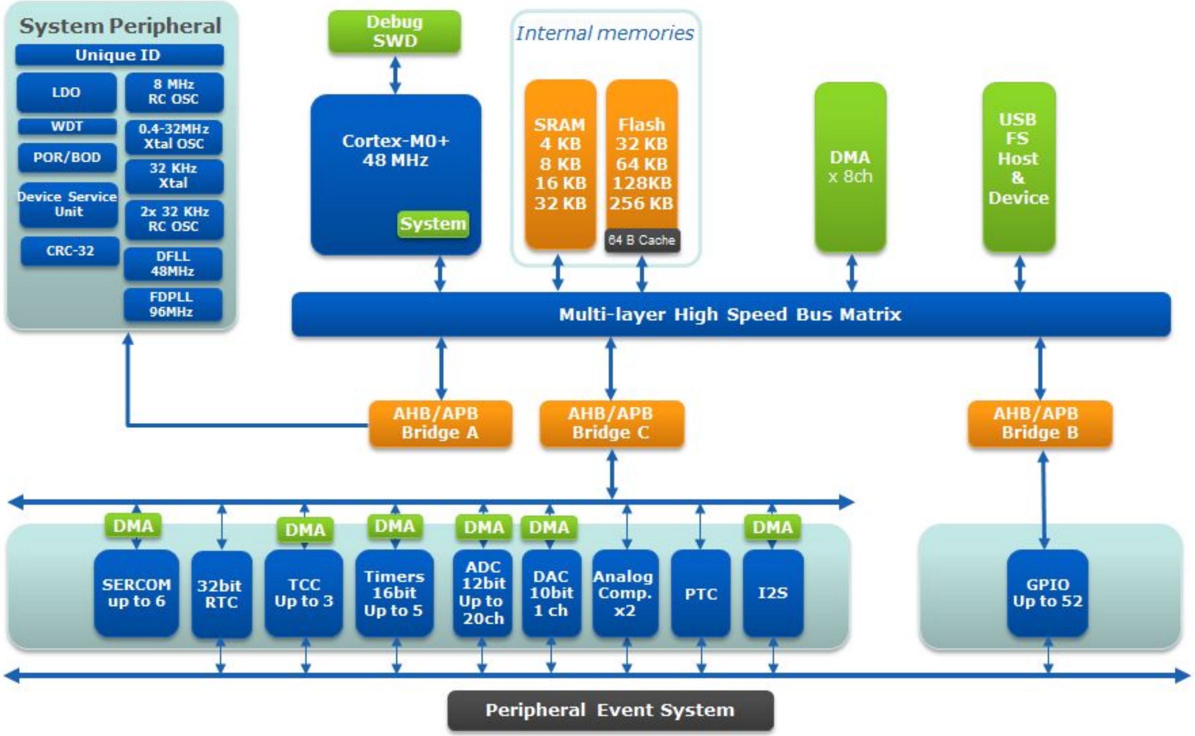
- architecture
- memory mapping
- peripherals and their registers
- electricals
- schematic checklist

1. Introduction
2. Fundations
3. ARM Cortex-M families
4. ARM Cortex-M0+
5. Software API
6. Tools
7. Atmel WINC1500-XSTK
8. Arduino

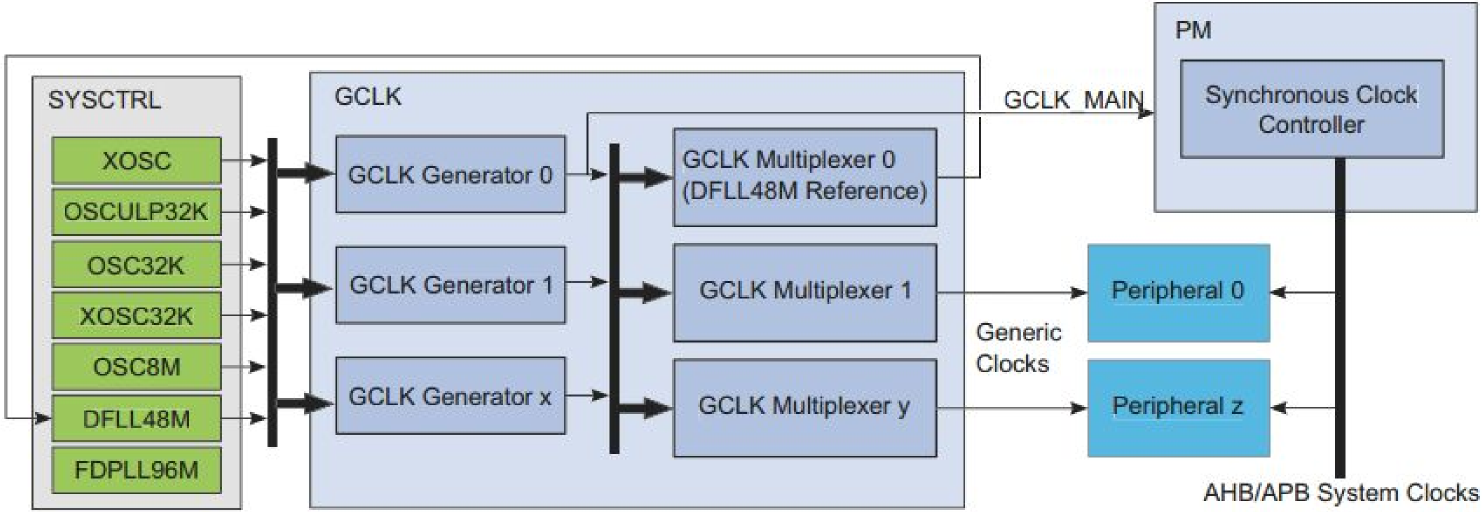
Atmel WINC1500-XSTK



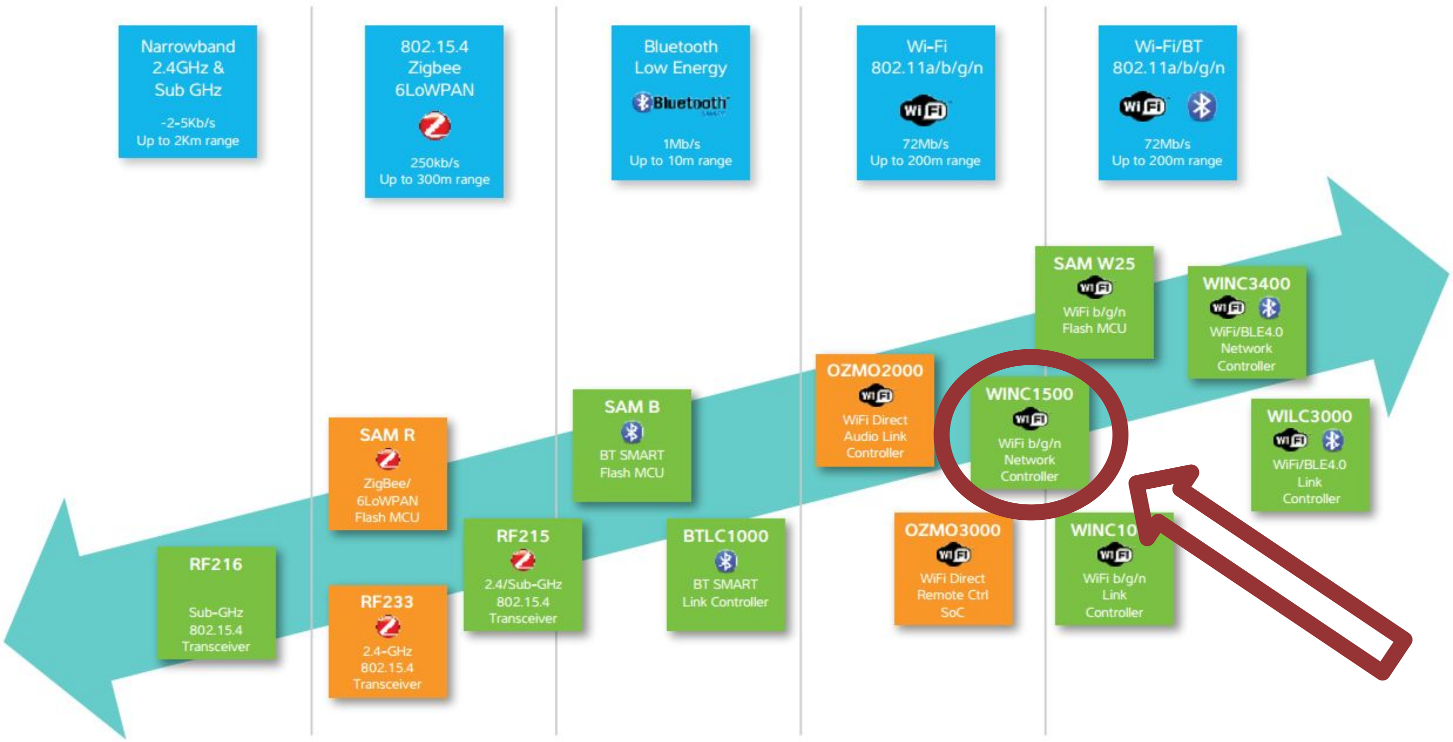
ATSAM D21 architecture



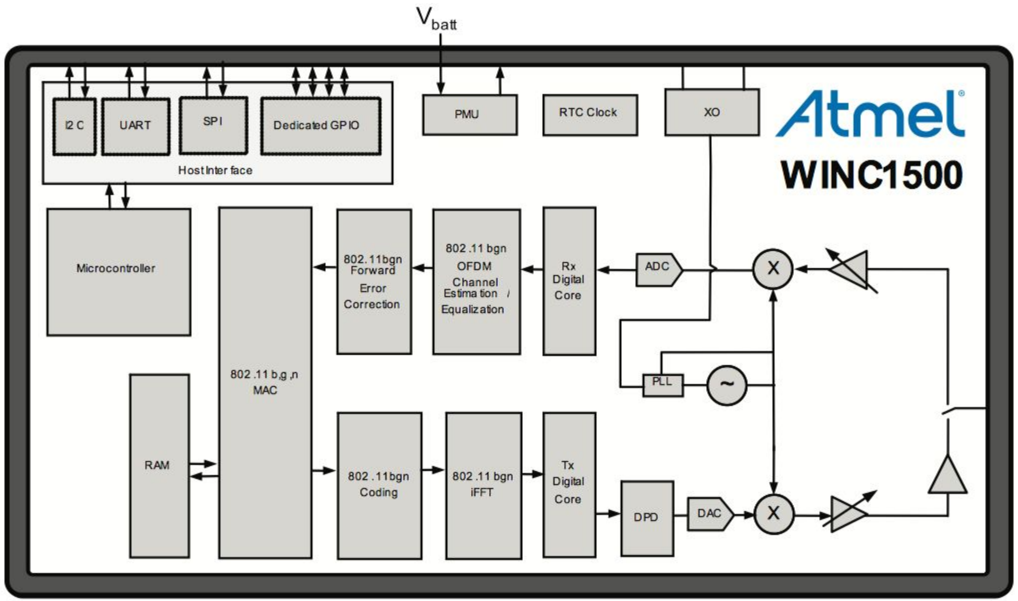
ATSAM D21 clocks



Wireless communications in IoT



WINC1500 XPlained Pro extension



Xplained Pro Extension header pinout

ID	1		2	GND
ADC+	3		4	ADC-
GPIO0	5		6	GPIO1
PWM+	7		8	PWM-
IRQ	9		10	SPI_SS_B
TWI_SDA	11		12	TWI_SCL
UART_RX	13		14	UART_TX
SPI_SS_A	15		16	SPI_MOSI
SPI_MISO	17		18	SPI_SCK
GND	19		20	VCC

What is Arduino?

Arduino is an open-source prototyping platform based on easy-to-use hardware and software, mostly dedicated to education.

Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

You can tell your board what to do by sending a set of instructions to the microcontroller on the board.

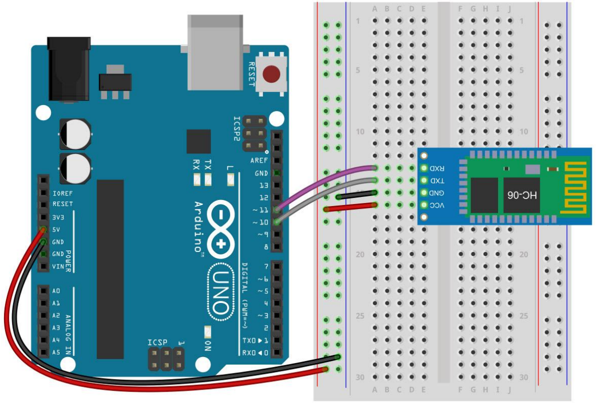
Arduino Uno and Zero comparison

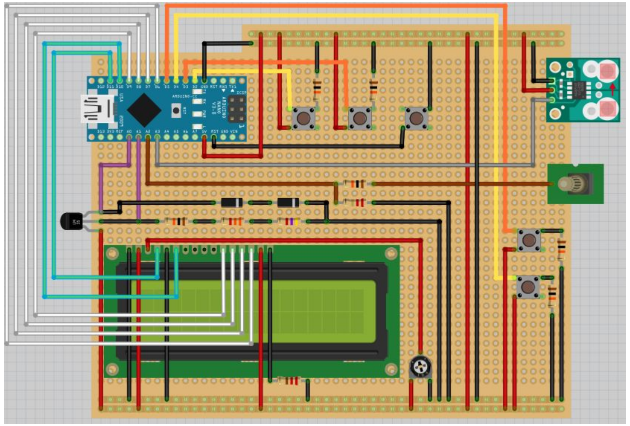
Feature	ATMega 328P - Uno	SAMD21 - Zero
uP core	AVR 8 bit RISC	ARM Cortex M0+ 32 bit
Max core speed	16 MHz	48 MHz
CoreMark/MHz	0.54	2.46
Flash	32 kB	256 kB
RAM	2 kB	32 kB
GPIO	23	38
Voltage	1.8 – 5.5	1.62 – 3.6V
USB	no	Host and device
On board debug	no	Yes

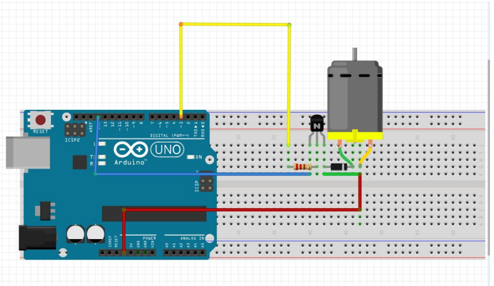
Approx 12x CPU power

What can we do with Arduino?

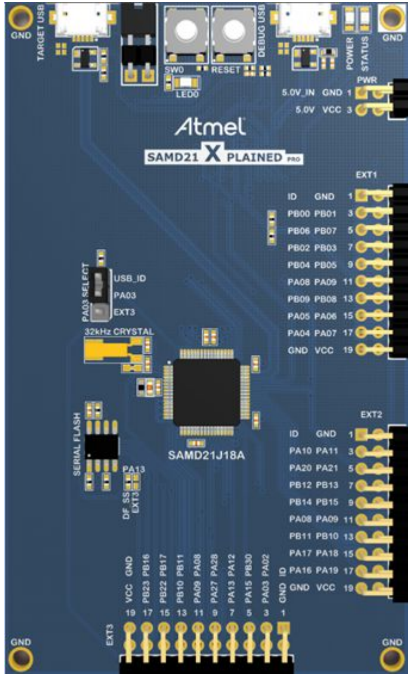
The electronic boards being based on industrial grade microcontroller, they can be connected to any industrial peripheral via General Purpose Inputs/Outputs or the different buses like serial port (RS-232, RS-485, …), SPI (Serial Peripheral Interface), I²C (Inter-Integrated Circuit), Parallel bus, Ethernet, WiFi, 802.15.4, CAN, etc… On top of these buses, we can see industrial protocols like ModBus, EtherCat, etc...







SAM D21 XPro seen with Arduino eyes



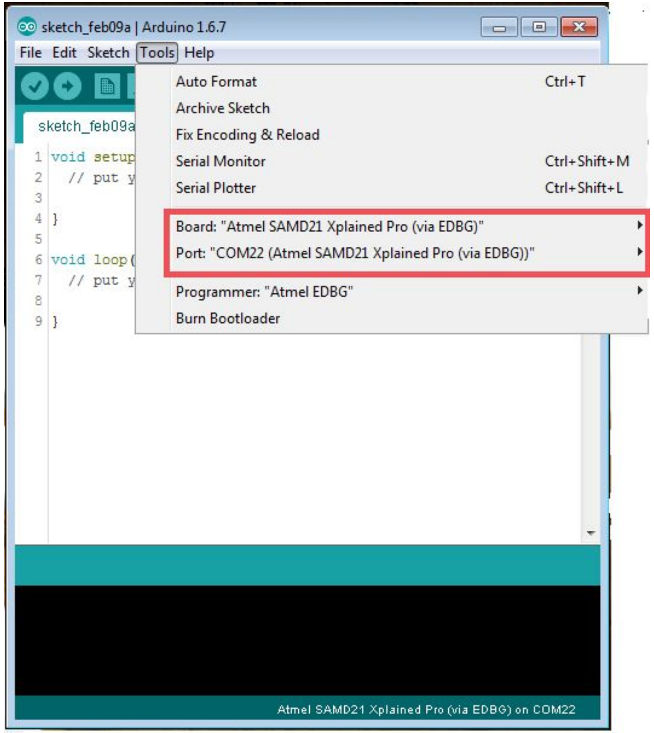
A0-A1
D4-D5
D16-D17
D6-D7
Wire
Serial1
SPI

A2-A3
D8-D9
D18-D19
D10-D11
Wire
Serial2
SPI1

A4-A5
D12-D13
D20-D21
D14-D15
Wire
Serial2
SPI2

Select the SAM D21 Xplained Pro

- Launch Arduino IDE
- Verify the SAMD21 Board is identified in the IDE under Tools menu



Flashing the Arduino bootloader

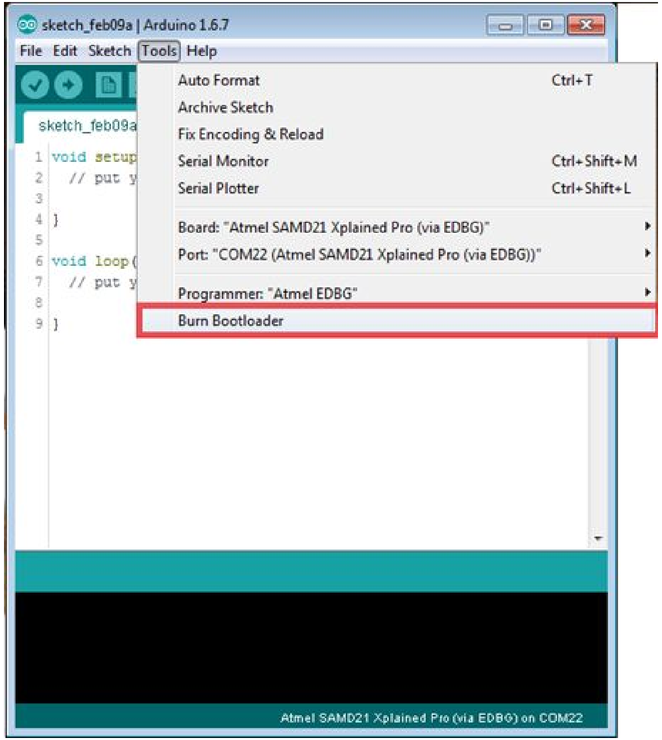
1. Select “Atmel EDBG” in Tools->Programmer
2. Select “Burn Bootloader”
3. After completion the following lines will appear:

** Verified OK **

** Resetting Target **

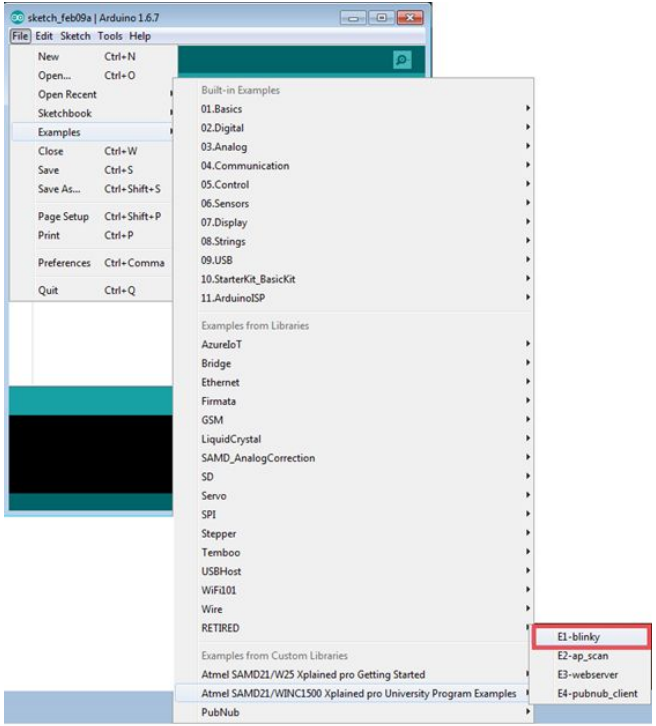
shutdown command invoked

The board is ready to welcome your applications



First example: Blinky

- Open a serial terminal window (TeraTerm, Putty, etc) at 115200 baud, 8 bits, no parity 1 stop bit
- “blink” should be printed every second along with the double blinking LED



Second example: WiFi Access Point scan

This sketch will scan for all available WiFi access points and reports the signal strength. Uses the WINC1500 in Station (normal mode).

1. Disconnect SAMD21 XPRO from USB power cable
2. Connect WINC1500 XPRO to EXT2 of SAMD21 XPRO
3. Re-connect USB cable to “Debug USB” port on SAMD21 XPRO
4. Load, build and upload the E2-ap_scan sketch from Atmel University
5. Open a serial console at 115200 8N1, the output should be similar to the following slide

Second example (console)

```
Scanning available networks...
** Scan Networks **
number of available networks:6
0) TestWPAN     Signal: -46 dBm Encryption: 2
1) AVRGUEST     Signal: -68 dBm Encryption: 2
2)      Signal: -70 dBm Encryption: 4
3) HP_Pro_Portable_Tablet_Dock 005      Signal: -72 dBm Encryption: 1
4)      Signal: -83 dBm Encryption: 2
5) TouchLab24   Signal: -51 dBm Encryption: 2
Scanning available networks...
** Scan Networks **
number of available networks:6
0) TestWPAN     Signal: -46 dBm Encryption: 2
1) AVRGUEST     Signal: -70 dBm Encryption: 2
2)      Signal: -71 dBm Encryption: 2
3) HP_Pro_Portable_Tablet_Dock 005      Signal: -72 dBm Encryption: 1
4)      Signal: -70 dBm Encryption: 4
5) TouchLab24   Signal: -50 dBm Encryption: 2
Scanning available networks...
```

Third example: webserver

This sketch will create a webserver. it uses the WINC1500 in Station mode to serve a tiny web page directly by reading analog inputs from the SAM D21.

1. Load, build and upload the E3-webserver sketch from Atmel University
2. We need to edit some things, like the SSID and password. Look for the following lines at the beginning of the sketch:

```
char ssid[] = "?? ssid ??";       // your network SSID (name)
char pass[] = "?? password ??";  // your network password
```

3. Replace the ssid and password strings with the access point information that's outlined by the instructor.
4. Build and upload the sketch, keep the serial terminal open from the E2-ap_scan exercise. We need information from the serial port to complete this exercise.
5. The sketch will connect to the WiFi network and then display the IP address that it obtained.
6. Use a browser on a computer or smartphone connected to the same WiFi network and point directly at the IP address given in the serial terminal output.

Summary

Basic WiFi operation with the SAM D21 Xplained Pro and its WINC1500 extension has been shown in a simple device function (ap_scan) along with a pretty basic webserver.

It is now up to you to go further!

Thank you!