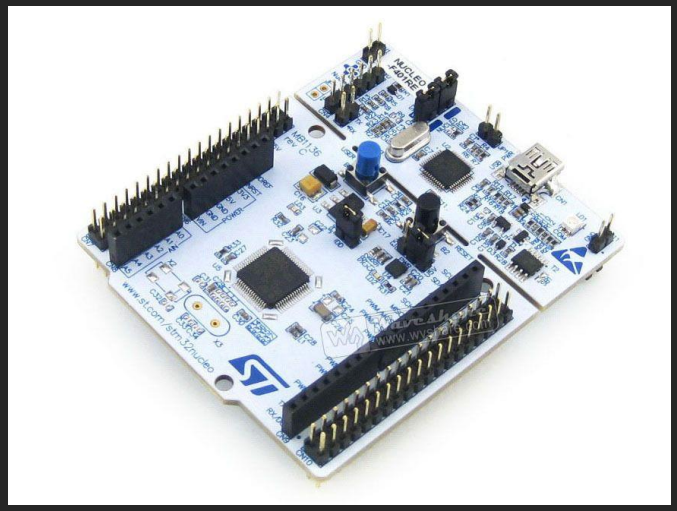
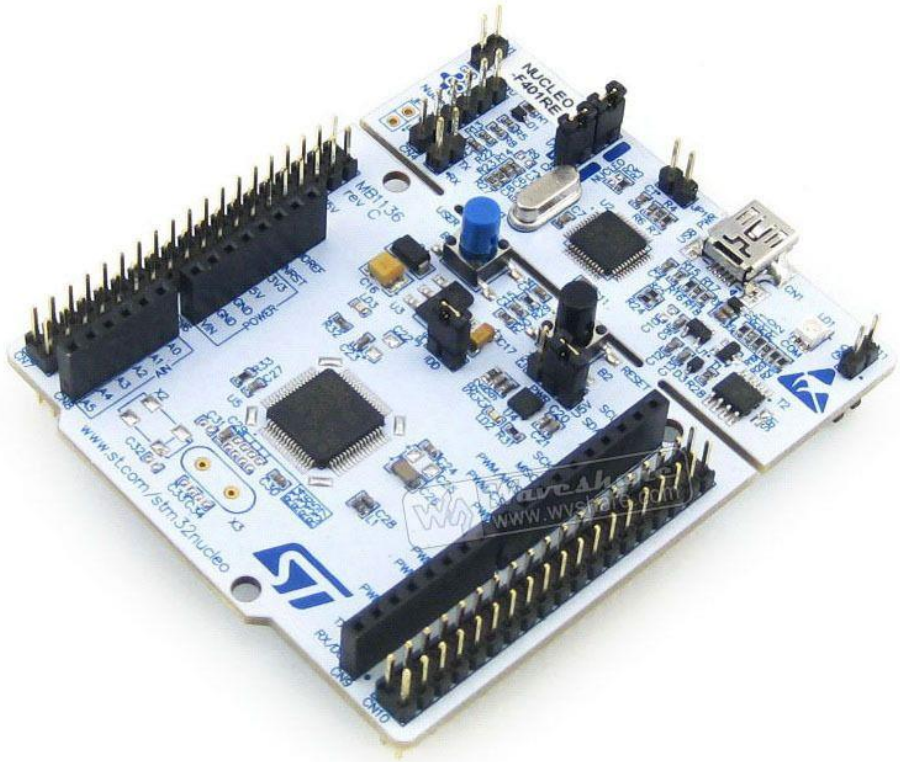
STM32F411
Nucleo Board

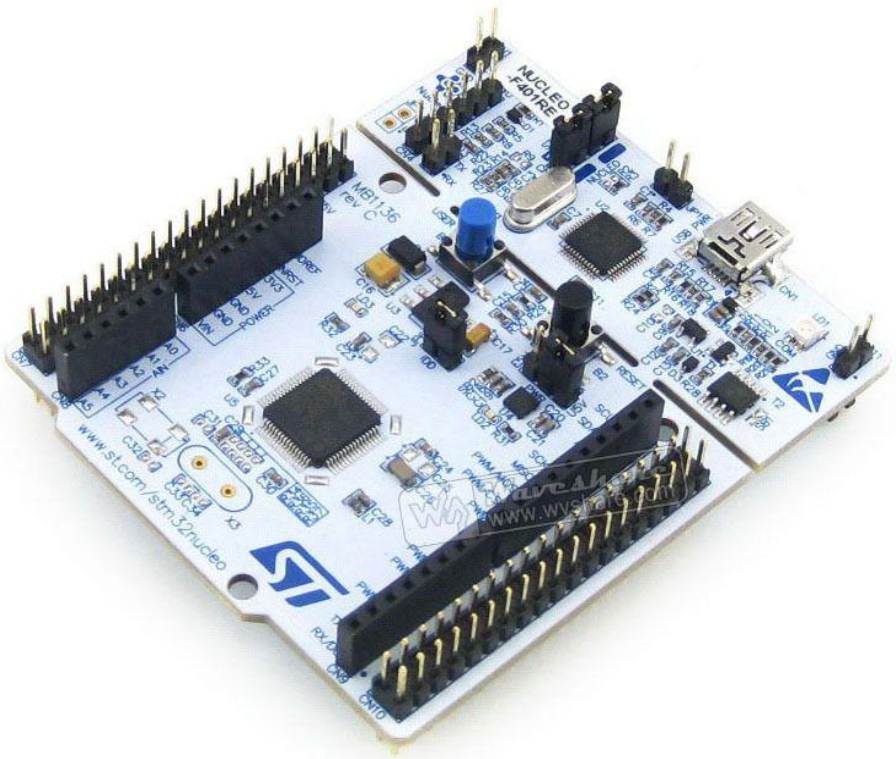
ATHANASIADIS ANGELOS
KARANASSOS DIMITRIOS

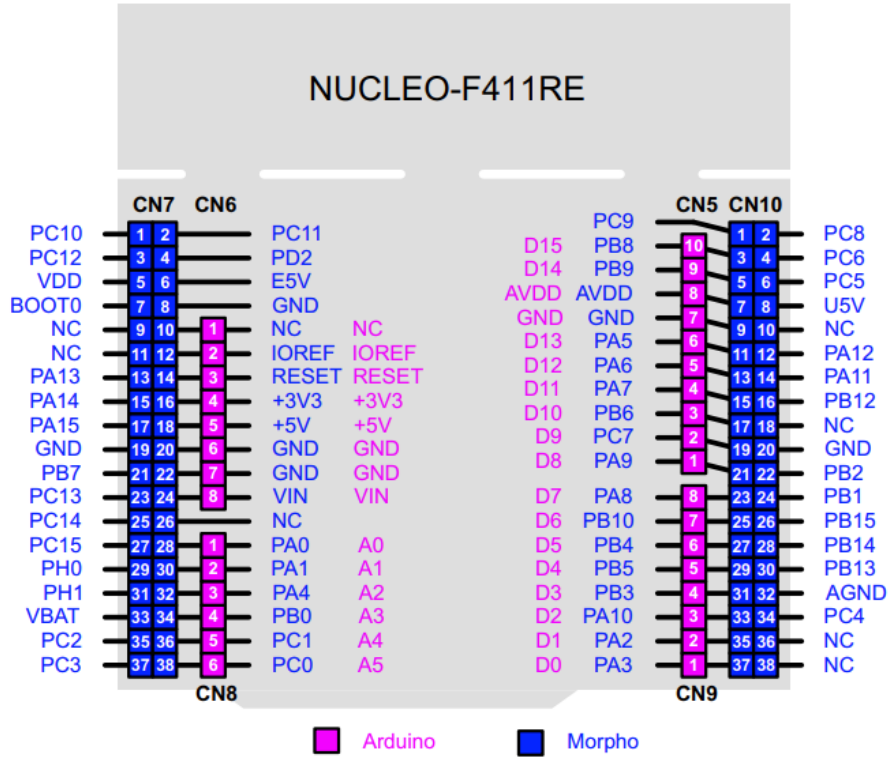


STM32F411
Nucleo Board

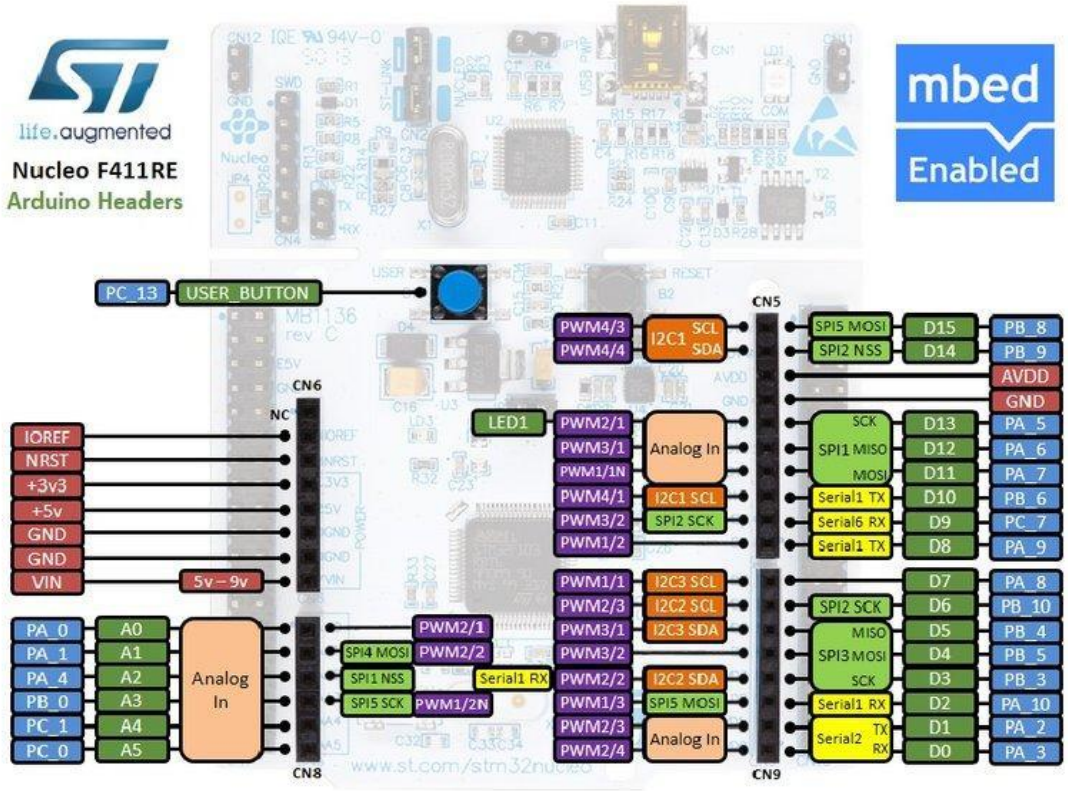


Board IO Pins
Ardunio Compatible





Nucleo Board Pinouts

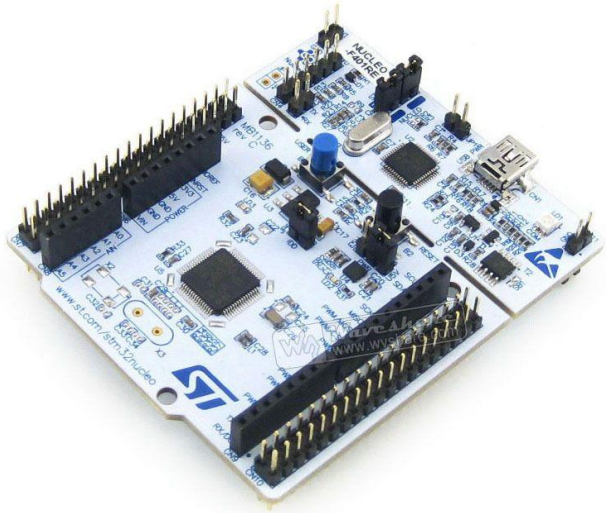


Labels usable in code

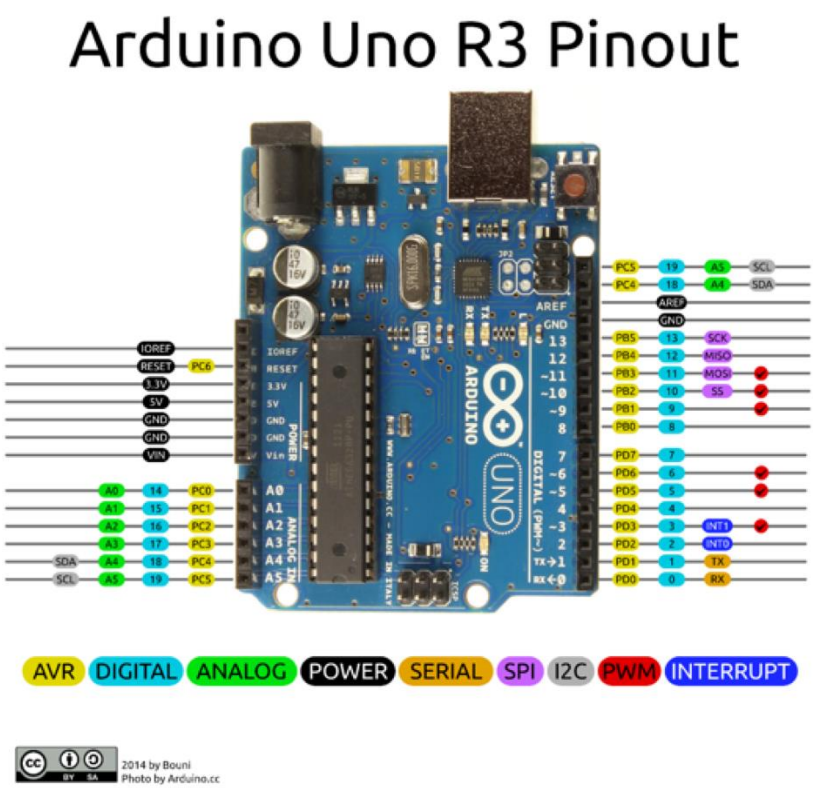
PX_Y	MCU pin without conflict
PX_Y	MCU pin connected to other components See **PeripheralPins.c** (link below) for more information
XXX	Arduino connector names (A0, D1, ...)
XXX	LEDs and Buttons (LED_1, USER_BUTTON, ...)

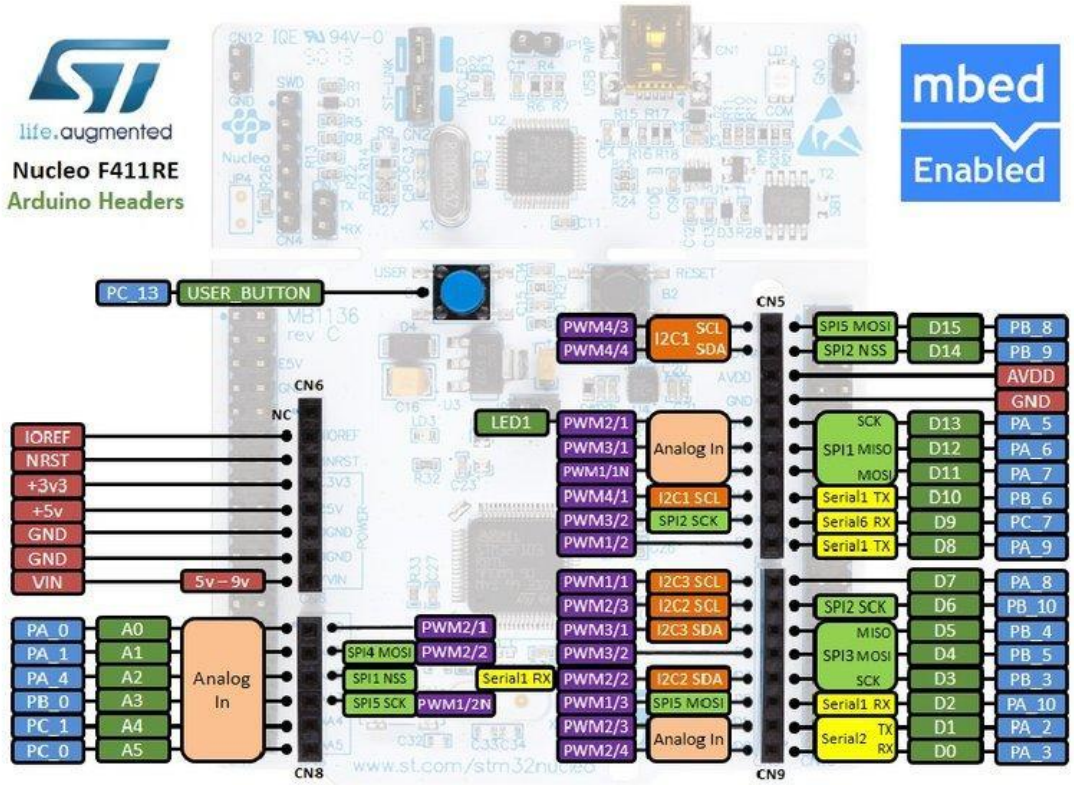
Labels not usable in code (for information only)

XXX	Serial pins (USART/UART)
XXX	SPI pins
XXX	I2C pins
XXX	PWMOut pins (TIMER n/c[N]) n = Timer number c = Channel N = Inverted channel
XXX	AnalogIn (ADC) and AnalogOut pins (DAC)
XXX	CAN pins
XXX	Power and control pins (3V3, GND, RESET, ...)



Comparing Arduino and Nucleo Board

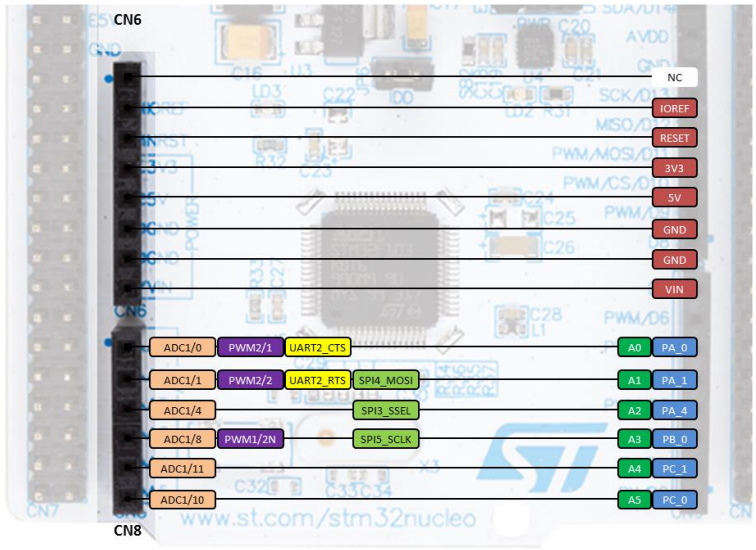




life.augmented

NUCLEO-F411RE

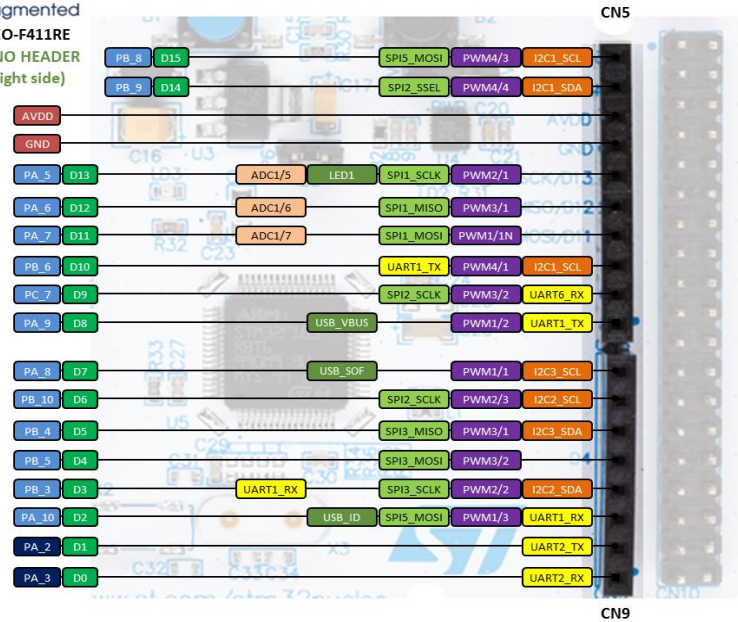
ARDUINO HEADER
(top left side)



life.augmented

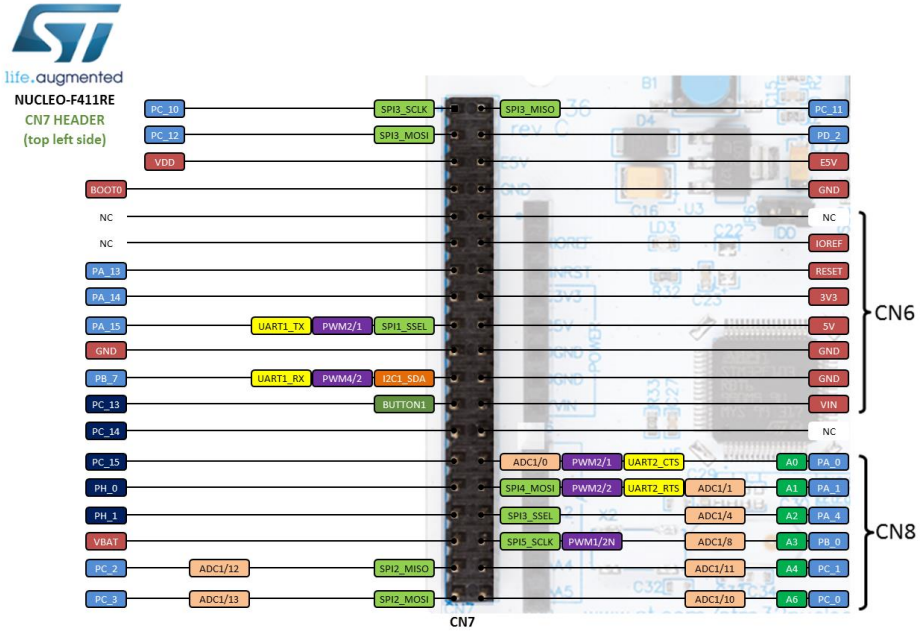
NUCLEO-F411RE

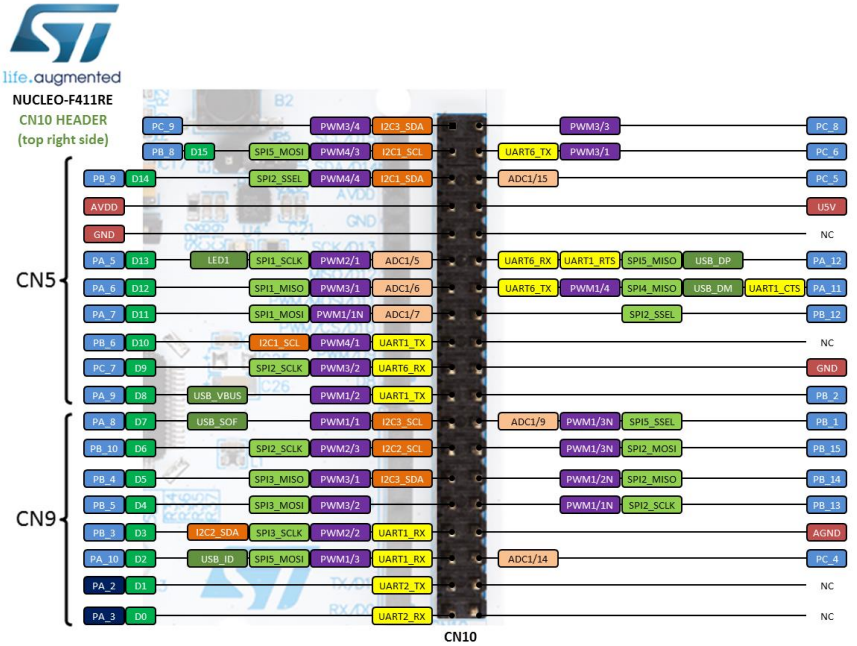
ARDUINO HEADER
(top right side)



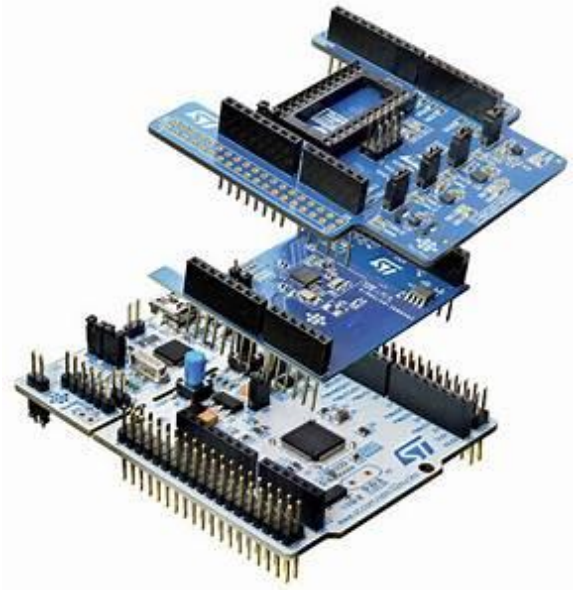
Arduino-compatible headers

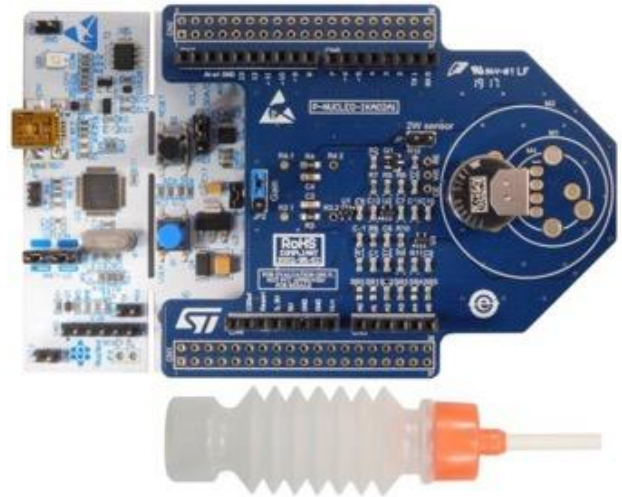
Morpho headers



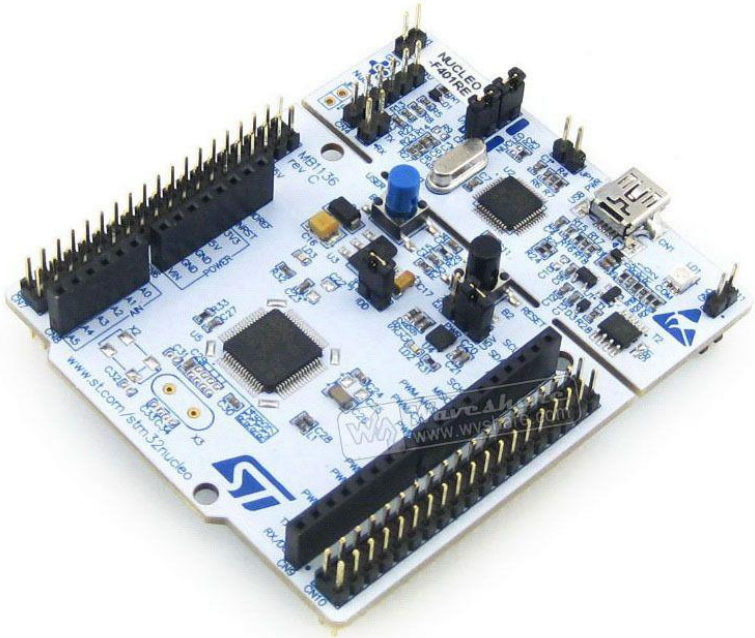


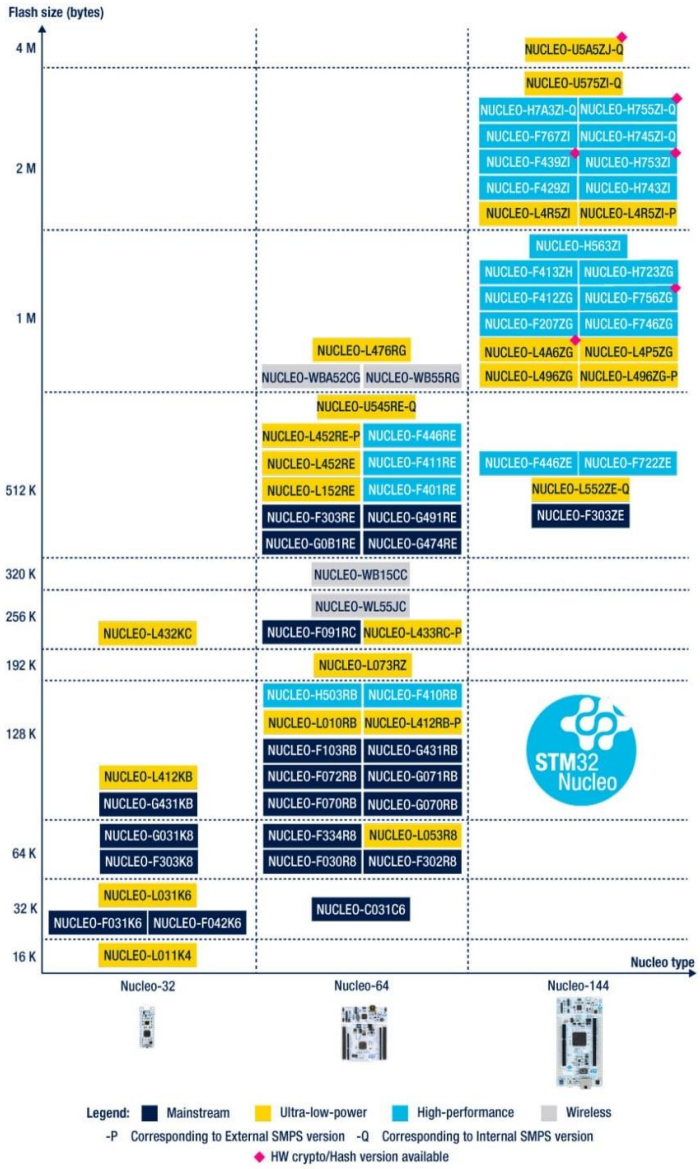
Expansion Shields for Nucleo Board

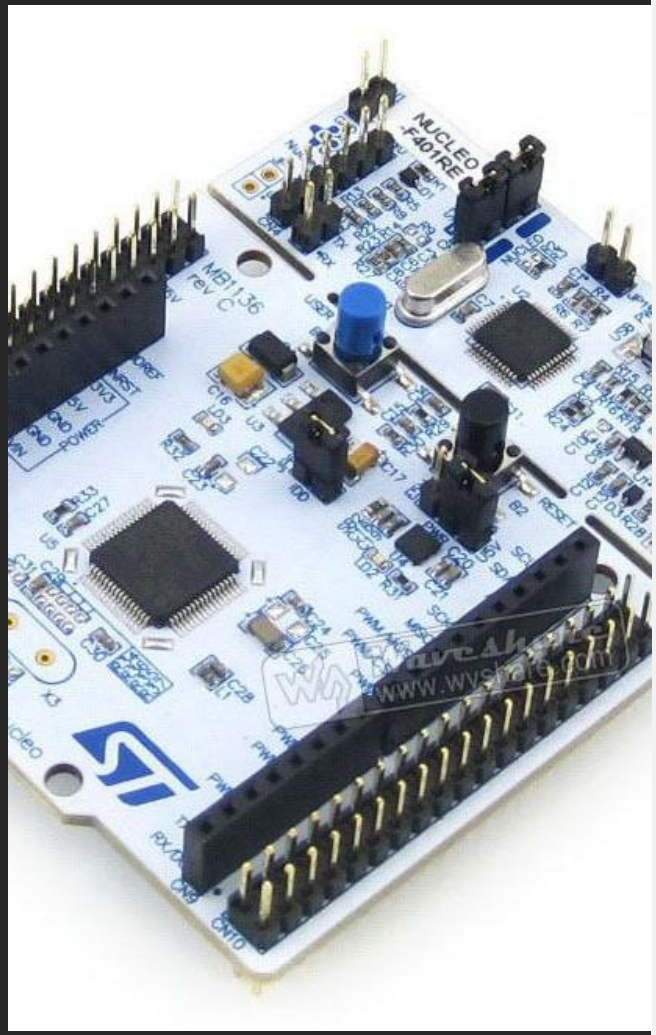




STM32 Nucleo Ecosystem



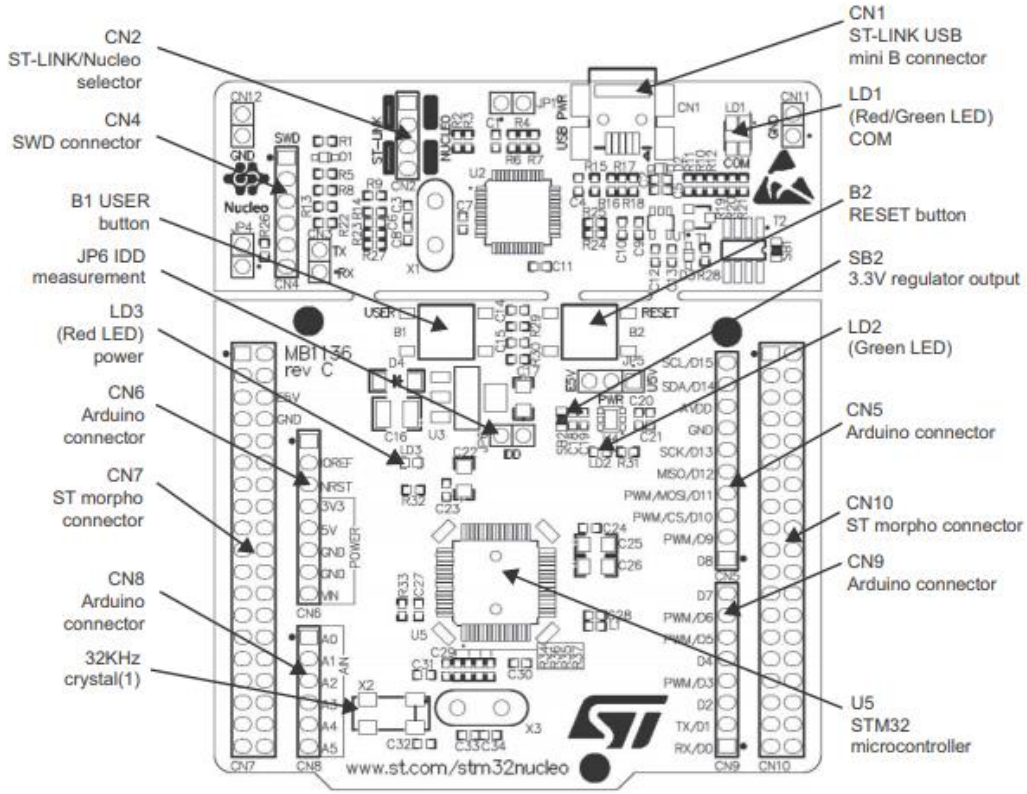




STM32F411RE MCU – STM32F411RET6

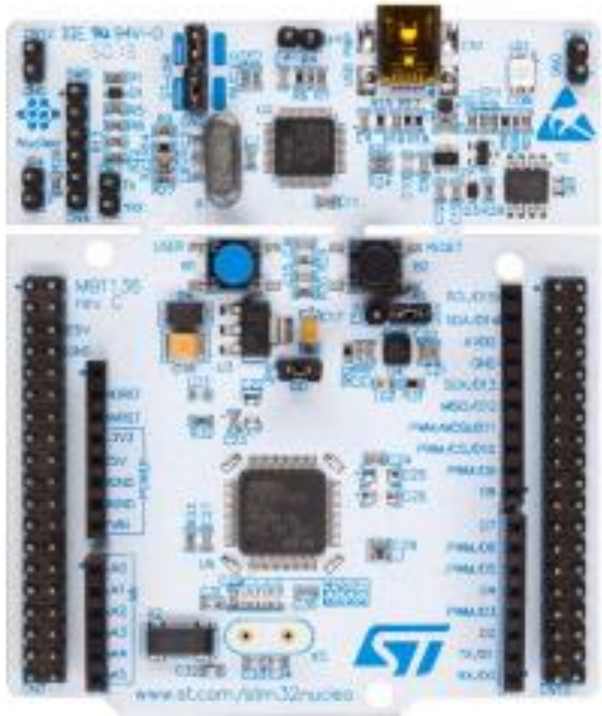
- STM32F411RET6 in LQFP64 package
- ARM®32-bit Cortex®-M4 CPU with FPU
- 100 MHz max CPU frequency
- VDD from 1.7 V to 3.6 V
- 512 KB Flash and 128 KB SRAM
- GPIO (50) with external interrupt capability
- 12-bit ADC with 16 channels
- RTC and Timers (8)
- I2C (3) - USART (3) - SPI (5)
- USB OTG Full Speed
- SDIO

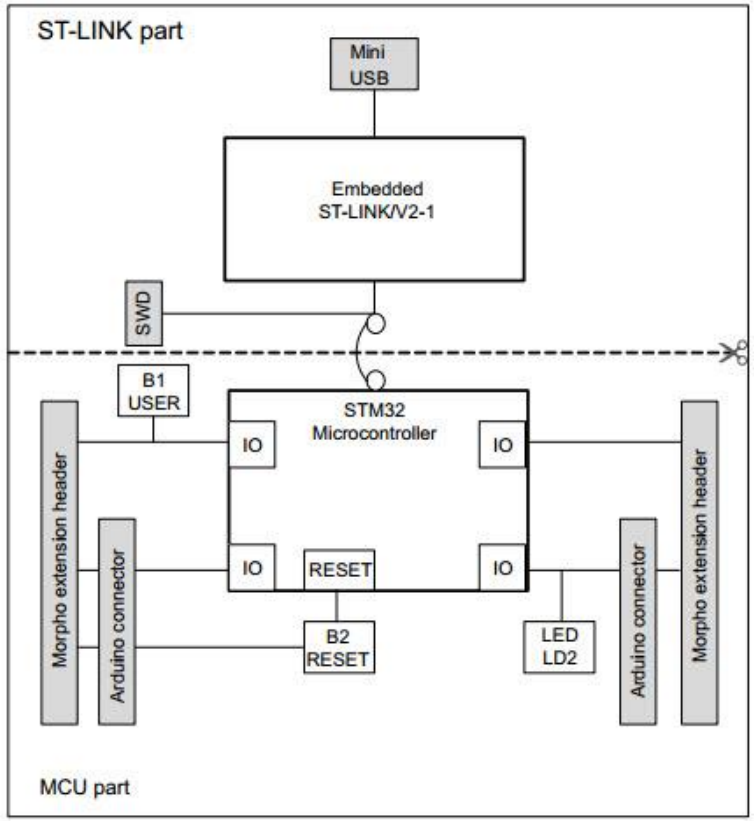
STM32F411RE MCU
Zooming in



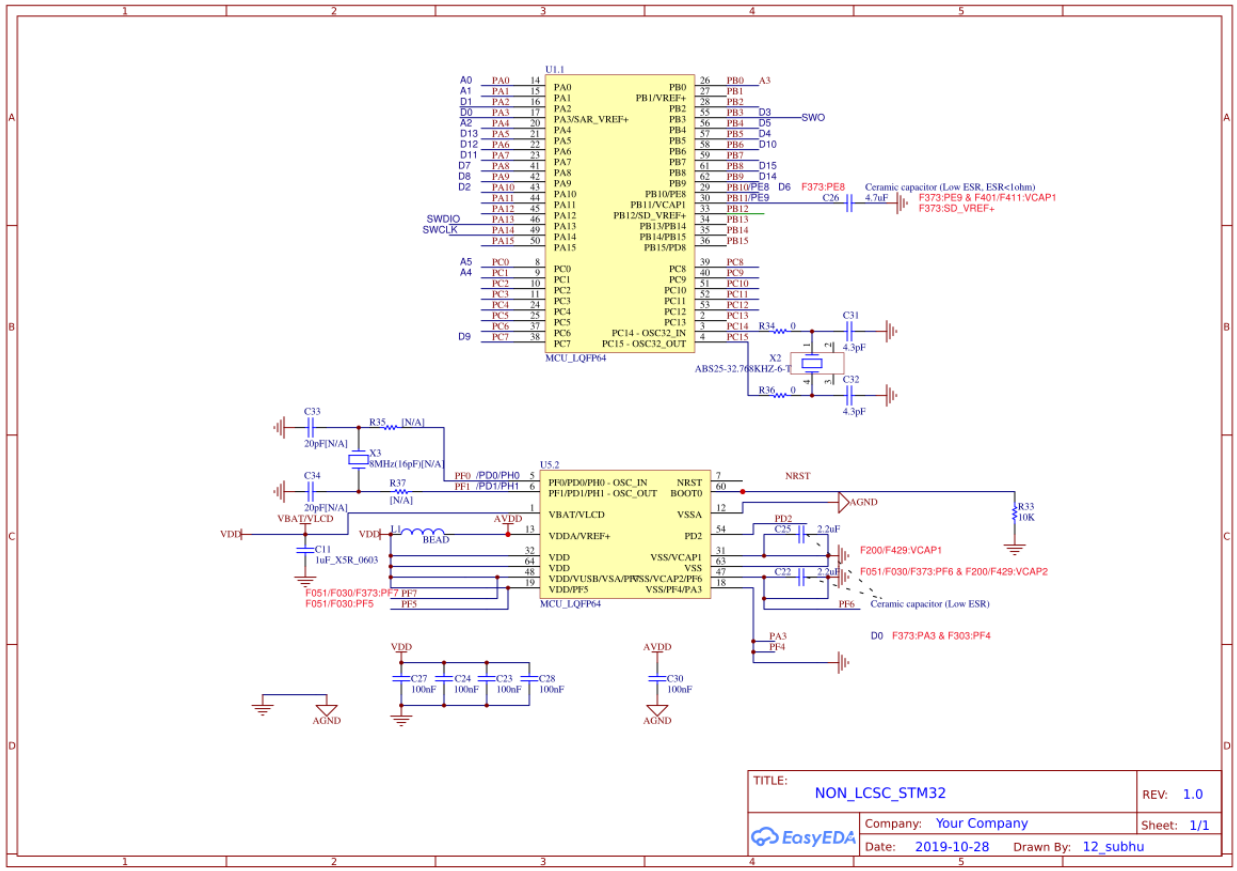
System View

- The ST-LINK allows us to program the board

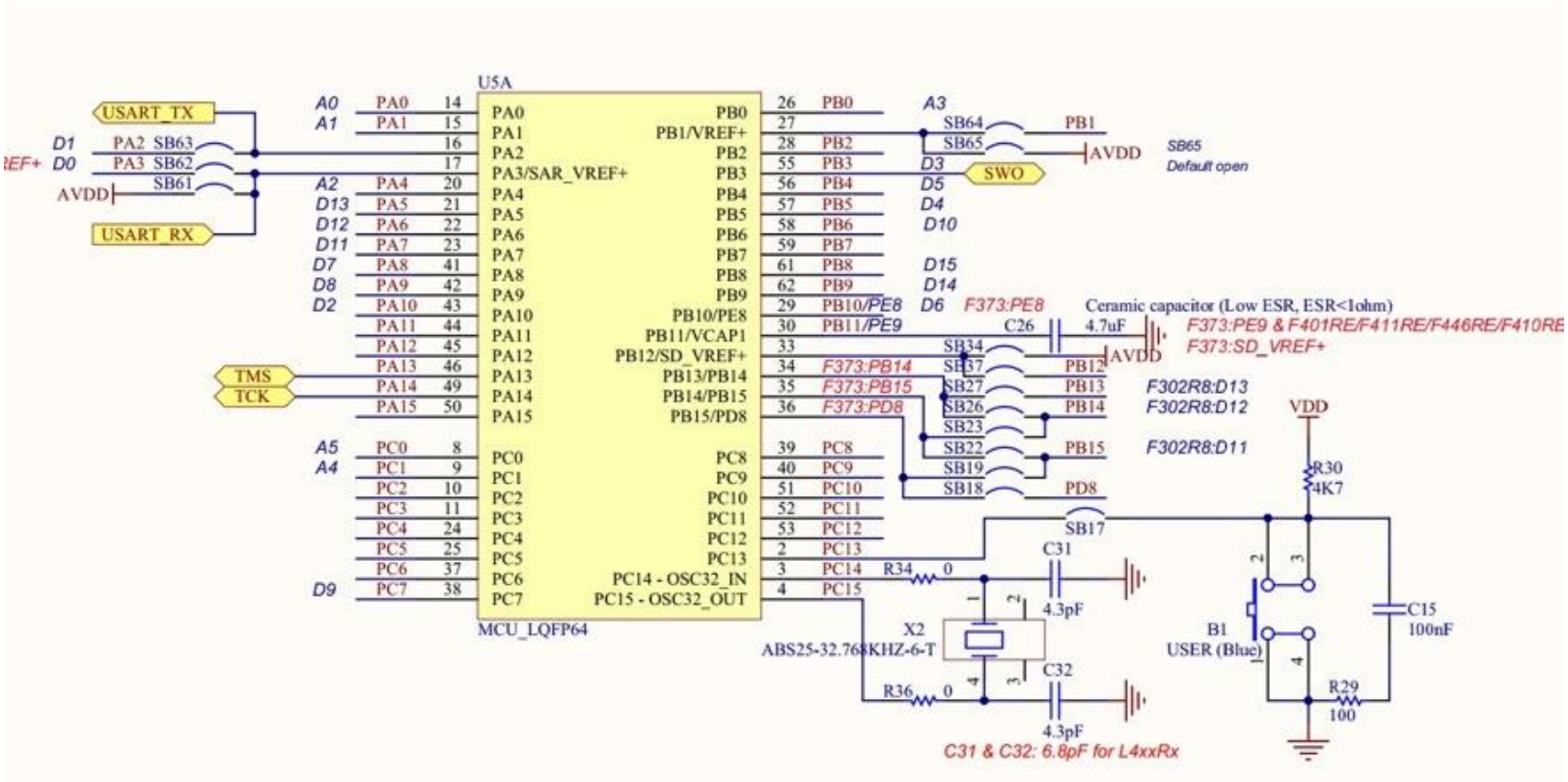




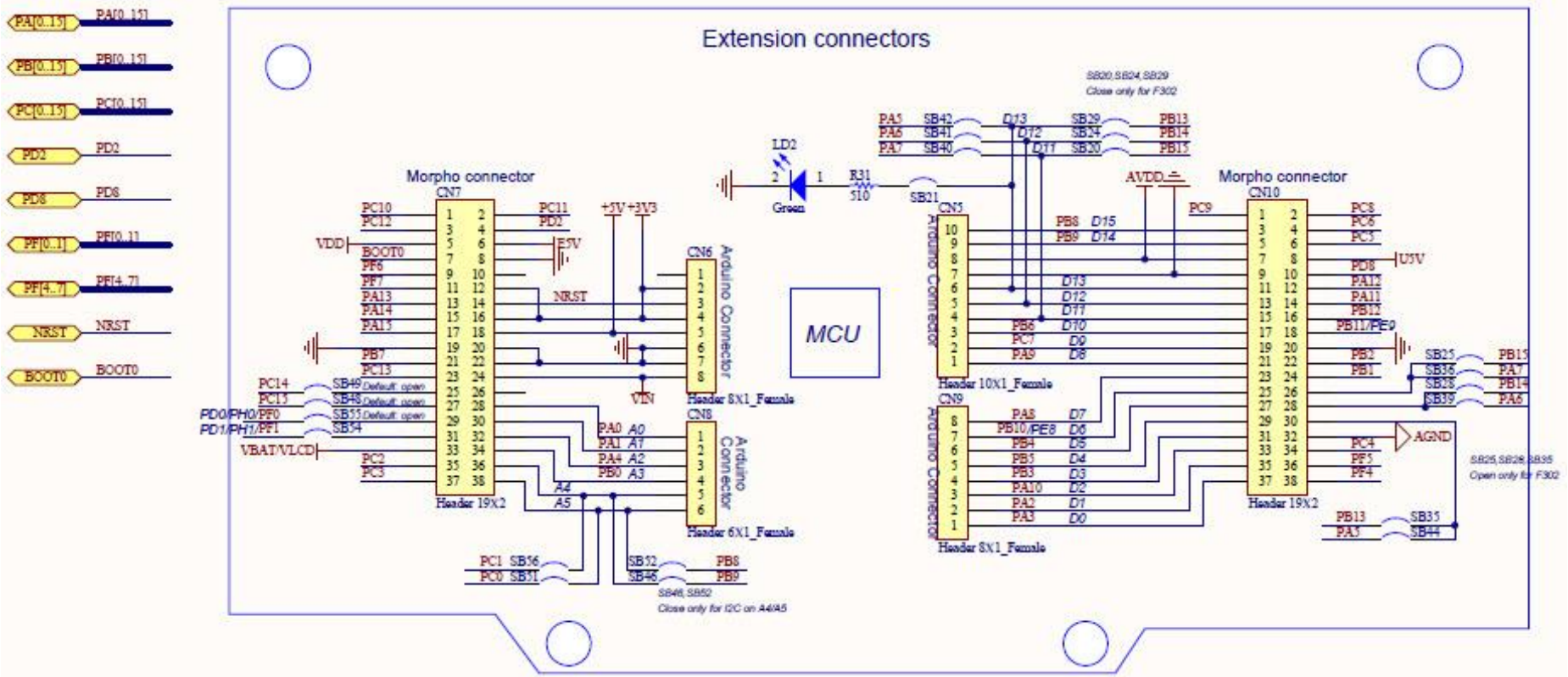
Board Schematic



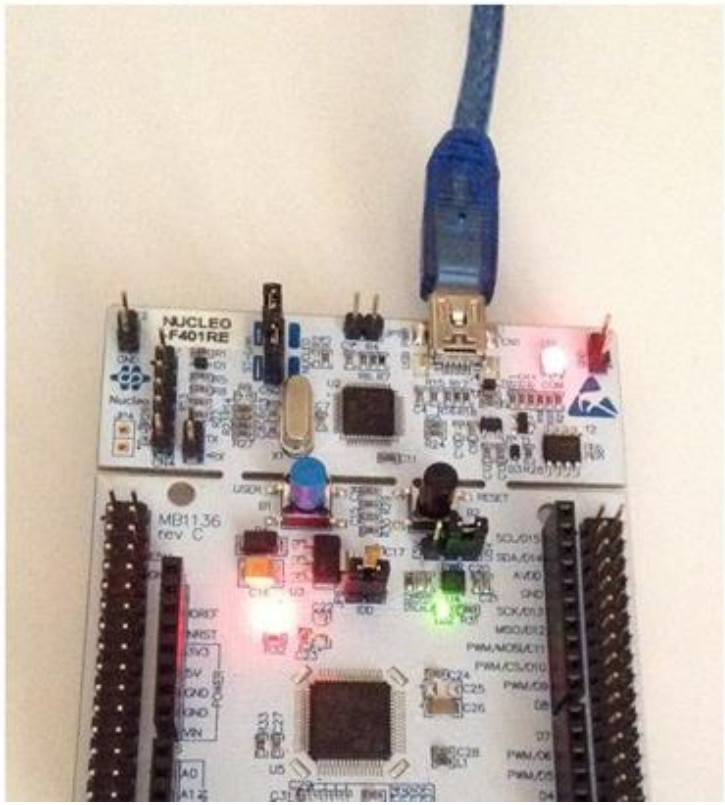
Board Schematic
Use the bush button

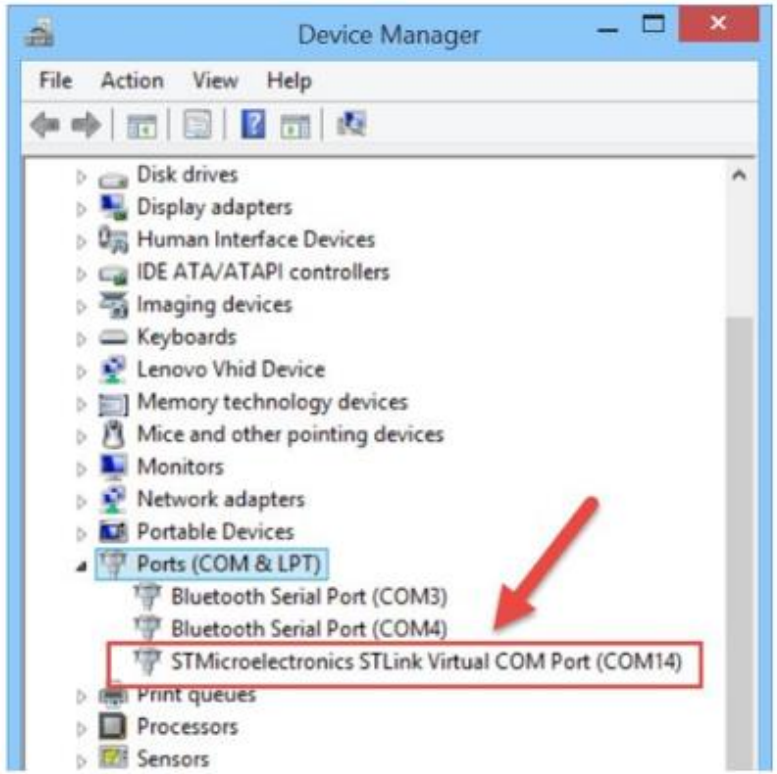


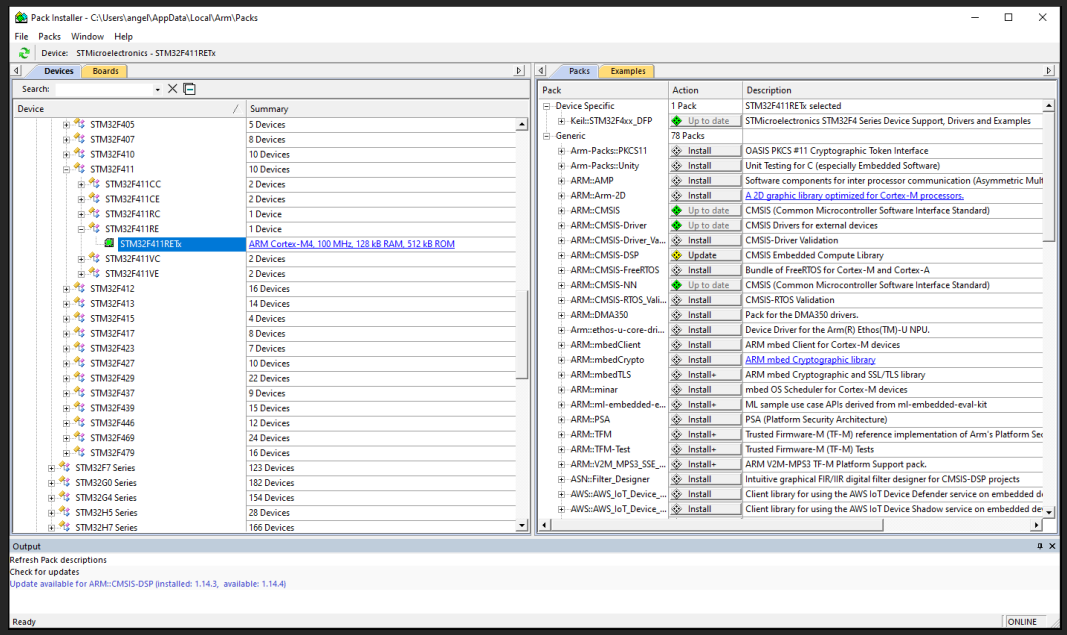
Extension Connectors



Checking your Connection





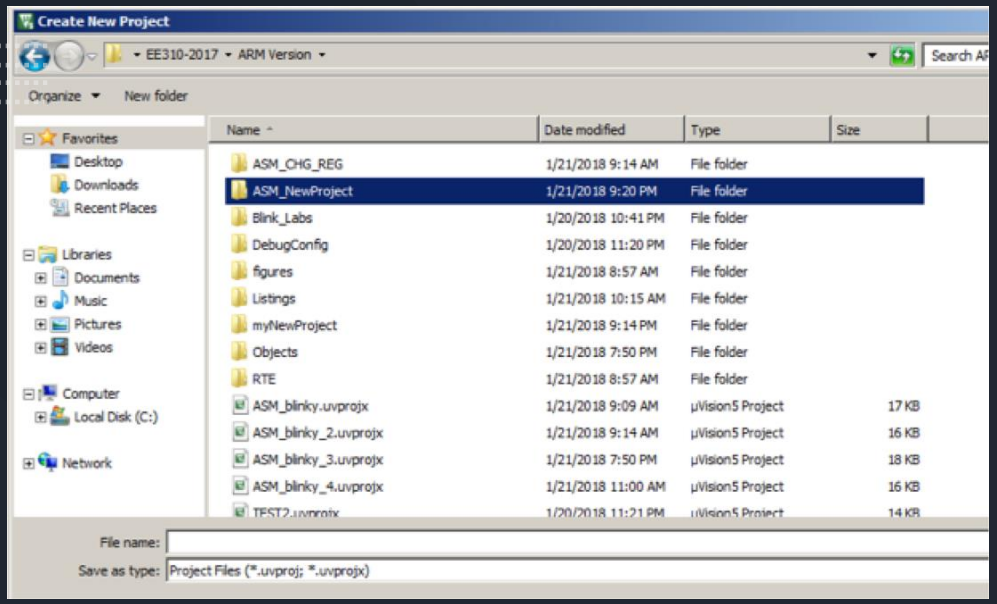


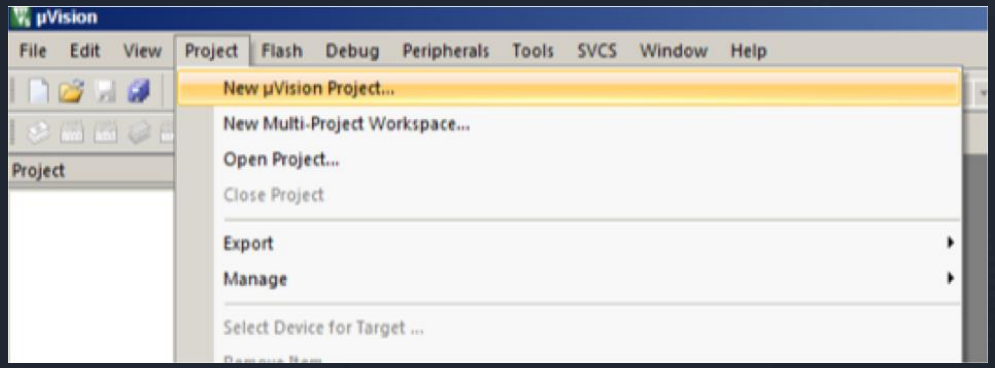
Start a NEW project in your Keil

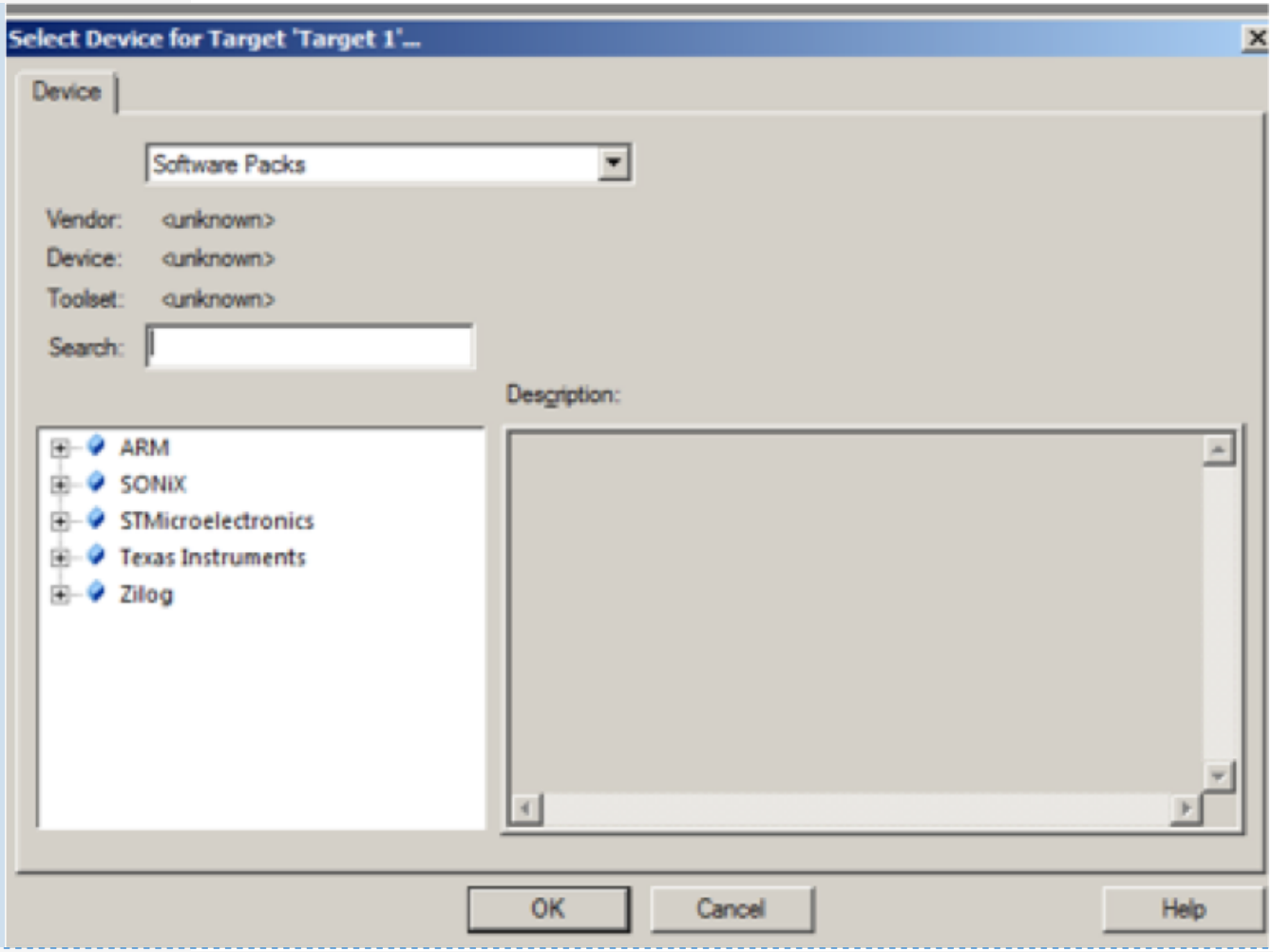
- On the left side of the window under Devices, click to expand STMicroelectronics.
- Find STM32F4 Series and expand it
- Double click on STM32F411RE and it should add it to the packs list on the right side.
- Click all the buttons that say Install or Update to get all the features/packages that Keil offers.

Start A New Project

- • Open Keil uVision 5 from your Windows
- Make sure all projects are closed:
- Project -> Close Project
- Click on Project -> new uVision Project …
- In the window click on New Folder
- Type the name of the project: ASM_NewProject
- Go to the new folder that you just created





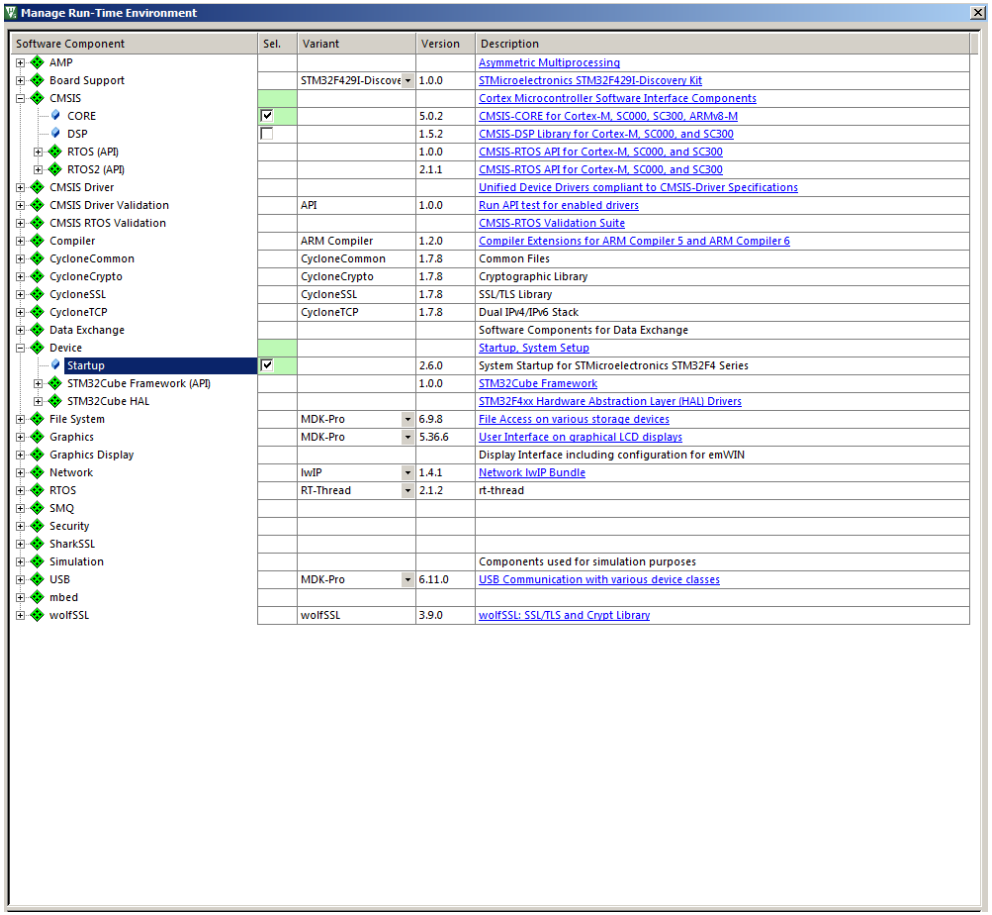


Start A New Project

- • The Select Device window pops up
- In the search type: STM32F411RETx
- Click on the device (STM32F411RETx) that showed up and then click OK

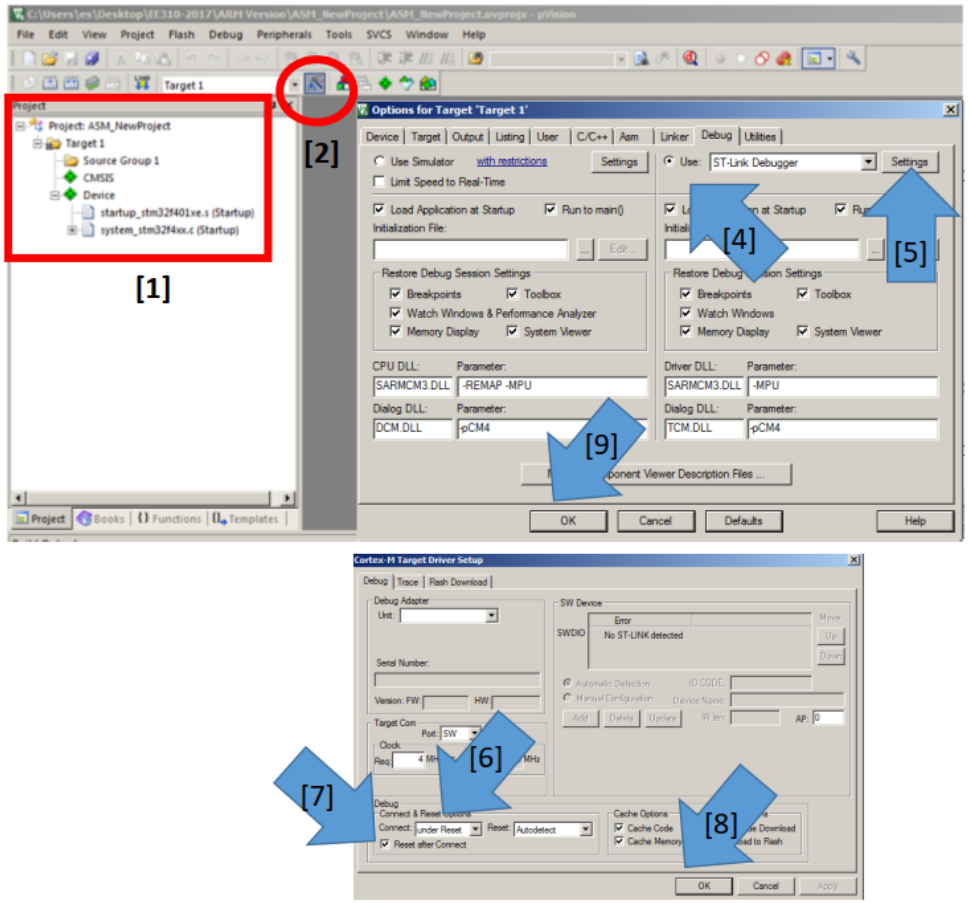
Start A New Project

- At this point the Manage Run window pops up
- CHECK CMSIS -> CORE and DEVICE->STARTUP boxes – note that the color changes
- Then click OK



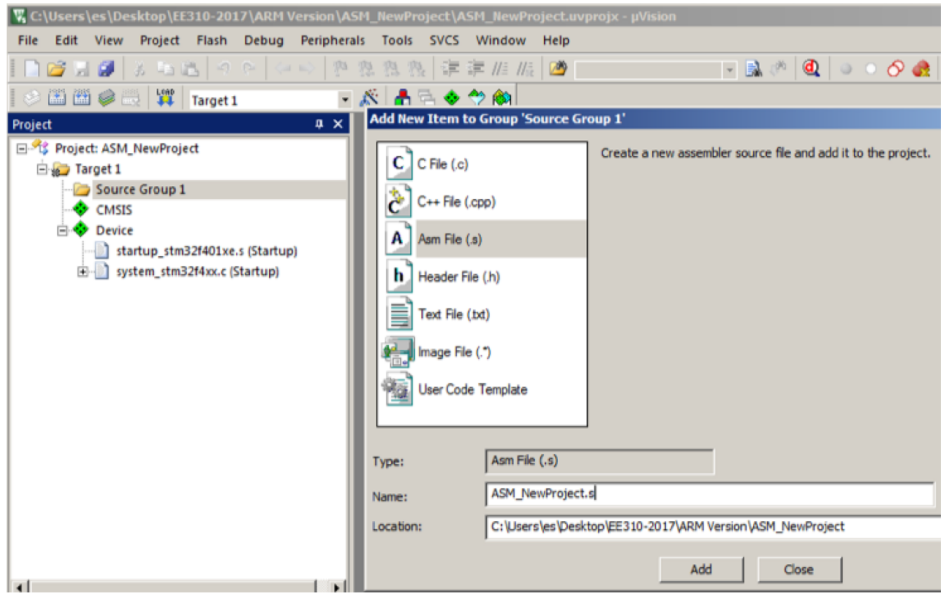
Start A New Project

- At tis point your Project window should show the CMSIS and DEVICE – these are the CMSIS system initialization files and startup files in order for the RUN TIME to operate **[1]**
  - Click on the OPTIONS FOR TARGET **[2]**
  - Click on DEBUG tab **[3]**
  - CHECK and Select the USE ST-LINK **[4]**
  - **Click on Setting [5]**
  - In the CORTEX-M4 Target Drive
  - Choose Under Reset [6]
  - Check RESET AFTER CONTROL Box [7]
  - Click ok [8]
  - In the OPTIONS window, Click on OK [9]
  - In the PROJECT window click on Source Group 1 and select ADD NEW ITEM….
  - In the new window select



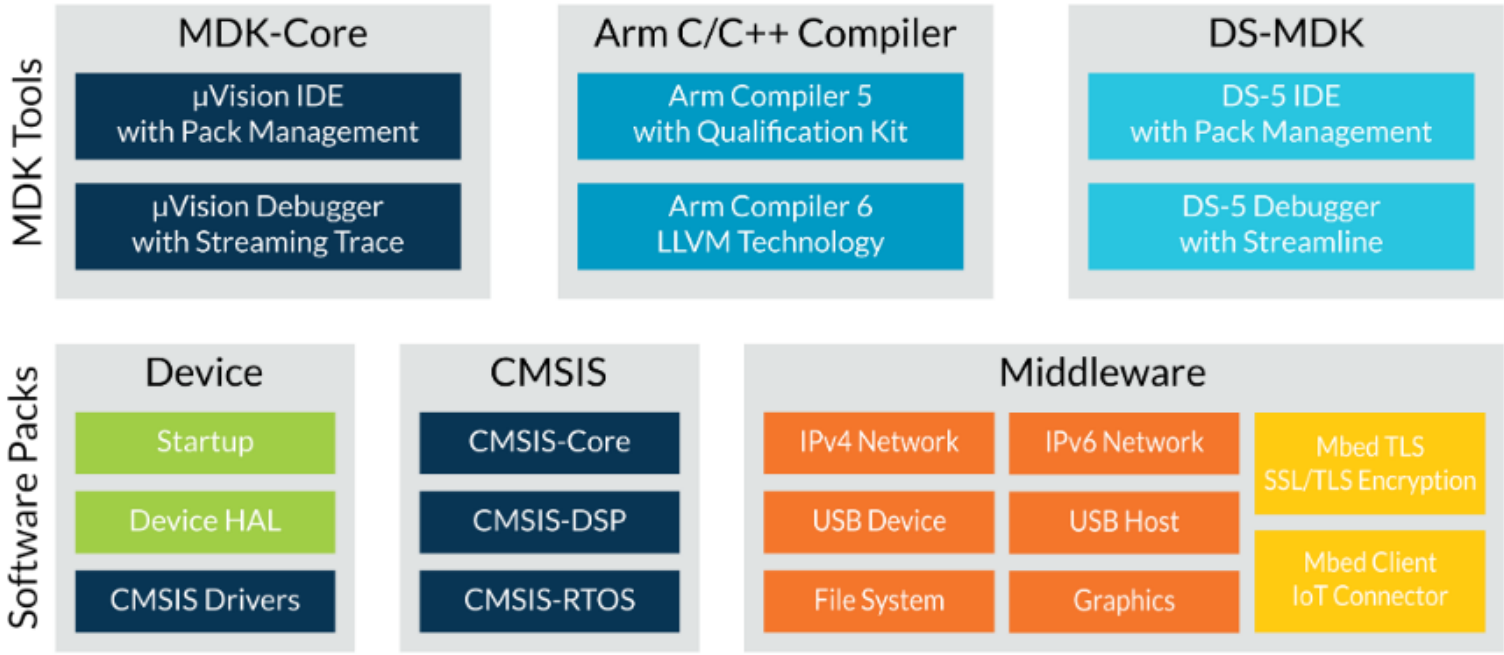
Start A New Project

- • In the PROJECT window click on Source Group 1 and select ADD NEW ITEM….
  • In the new window select ASM files
  • Then type the name of the ASM file: ASM_NewProject.s
  • Click on ADD



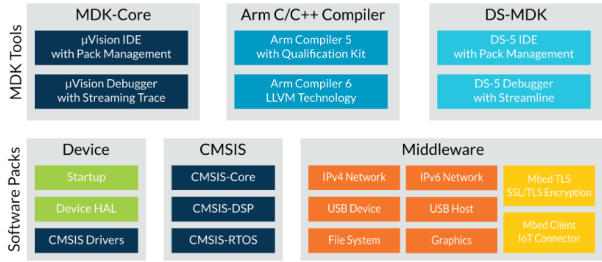
Arm Keil MDK

Arm® Keil® MDK is the most comprehensive software development solution for Arm Cortex®-based microcontrollers. Every MDK edition includes IDE, C/C++ compiler, debugger, software pack management, and CMSIS



Arm Keil MDK

MDK-Core offers leading support for over 3,750 Cortex-M based devices including the new Arm Cortex-M23/M33 cores. The µVision debugger includes event recorder and component viewer to show run-time behavior of software components. Together with the ULINKpro™ debug adapter, it offers full instruction trace and complete code coverage information.
DS-MDK contains the Eclipse-based DS-5 IDE/debugger and supports 32-bit Cortex-A processors or hybrid systems with 32-bit Cortex-A and Cortex-M. Streamline performance analyzer helps to get the best out of the system's resources and create high performance, energy efficient products.
Software packs can be added any time to MDK-Core or DS-MDK making new device support and middleware updates independent from the toolchain. They contain device support, CMSIS libraries, software components, middleware, board support, code templates, and example projects.
MDK middleware provides royalty-free, tightly-coupled software components that are specifically designed for communication peripherals in microcontrollers.

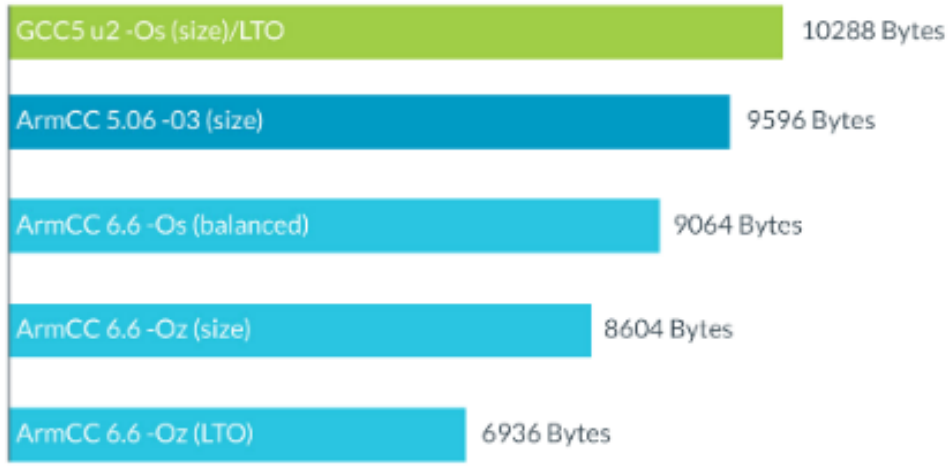


Arm Keil MDK

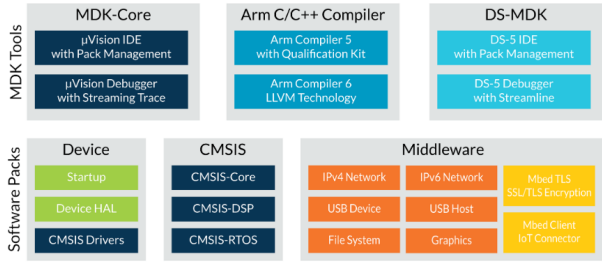
The industry-leading Arm C/C++ Compilers with assembler, linker, and highly optimized run-time libraries are tailored for optimum code size and performance.

Arm Compiler 5 is certified for functional safety applications and offers long-term maintenance and support.Arm Compiler 6 offer the best code size currently on the market.

It offers various optimization levels including Link Time Optimization. This diagram shows the code size.



developer.arm.com/armcompiler6

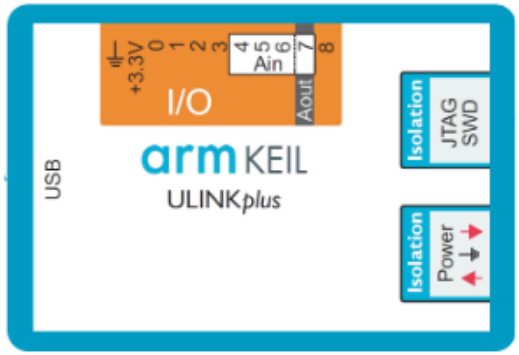


MDK editions and ULINK debug adapters

	Professional	Plus	Essential	Lite
µVision				
IDE with Editor, Pack Installer	✓	✓	✓	✓
Debugger	✓	✓	✓	32 KB
Fixed Virtual Platforms Simulation Models	✓			
DS-MDK (Windows® host + Linux® host)				
Support for heterogenous systems (Linux Cortex-A, RTOS Cortex-M)	✓	✓		
Streamline system analysis	✓			
Arm Compiler				
C/C++ Compilation Tools	✓	✓	✓	32 KB
Extended Maintenance and Qualification Kit	✓			
Arm processor support				
Arm Cortex-M0 .. Cortex-M7	✓	✓	✓	✓
Arm Cortex-M23/33 Non-Secure	✓	✓	✓	
Arm Cortex-M23/33 Secure + Non-Secure	✓	✓		
Armv8-M architecture + FastModel	✓			
Arm SecurCore™ (SC000, SC300)	✓	✓		
Arm7, Arm9, Arm Cortex-R4	✓	✓		
RTOS and Middleware				
CMSIS-RTOS RTX with full source code	✓	✓	✓	✓
Middleware (IPv4 Network, USB Device, File System, Graphics)	✓	✓		
Middleware (IPv6 Network, USB Host, IoT Connectivity)	✓			

www.keil.com/editions

MDK editions and ULINK debug adapters

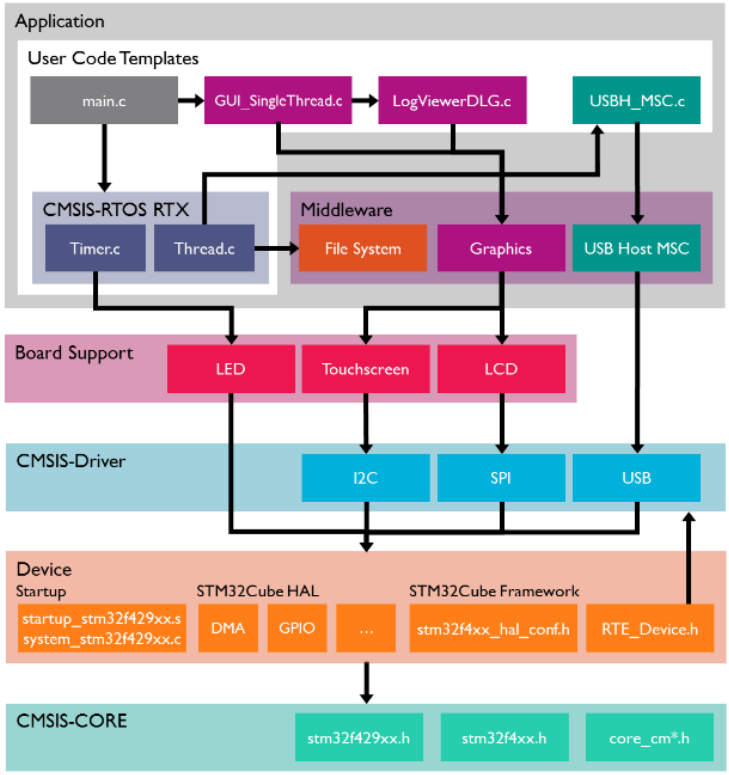


ULINKplus is a universal debug and trace adapter that enables test automation, software optimization for ultra-low-power applications, and isolation for debugging sensitive hardware systems. The compact enclosure allows usage in harsh environments and provides standard target connectors for JTAG, power measurement, and general purpose I/O.

Features	ULINKpro	ULINKplus	ULINK2
Run control	✓	✓	✓
Memory and breakpoint	✓	✓	✓
Data trace	✓	✓	✓
Instruction trace	✓		
Performance			
JTAG clock speed	50 MHz	50 MHz	50 MHz
Memory read/write	1 MB/s	3 MB/s	25 KB/s
Data and event trace	100 Mbit/s	50 Mbit/s	1 Mbit/s
Instruction trace	800 Mbit/s		
Analysis tools			
Component viewer	✓	✓	✓
Event recorder	✓	✓	✓
Power measurement		✓	
General purpose I/Os		✓	
Performance analyzer	✓		
Execution profiler	✓		
Code coverage	✓		

www.keil.com/ulink

CMSIS Components

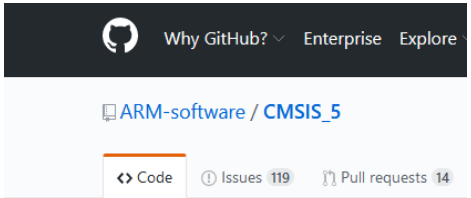


The **Cortex Microcontroller Software Interface Standard (CMSIS)** is a vendor-independent hardware abstraction layer for microcontrollers that are based on Arm® Cortex® processors. It defines generic tool interfaces and enables consistent device support. Its software interfaces simplify software re-use, reduce the learning curve for microcontroller developers, and improve time to market for new devices. CMSIS provides interfaces to processor and peripherals, real-time operating systems, and middleware components. It includes a delivery mechanism for devices, boards, and software and enables the combination of software components from multiple vendors.

CMSIS

CMSIS Components

CMSIS-...	Target Processors	Description
Core(M)	All Cortex-M, SecurCore	Standardized API for the Cortex-M processor core and peripherals. Includes intrinsic functions for Cortex-M4/M7/M33/M35P SIMD instructions.
Core(A)	Cortex-A5/A7/A9	Standardized API and basic run-time system for the Cortex-A5/A7/A9 processor core and peripherals.
Driver	All Cortex	Generic peripheral driver interfaces for middleware. Connects microcontroller peripherals with middleware that implements for example communication stacks, file systems, or graphic user interfaces.
DSP	All Cortex-M	DSP library collection with over 60 Functions for various data types: fixed-point (fractional q7, q15, q31) and single precision floating-point (32-bit). Implementations optimized for the SIMD instruction set are available for Cortex-M4/M7/M33/M35P.
NN	All Cortex-M	Collection of efficient neural network kernels developed to maximize the performance and minimize the memory footprint on Cortex-M processor cores.
RTOS v1	Cortex-M0/M0+/M3/M4/M7	Common API for real-time operating systems along with a reference implementation based on RTX. It enables software components that can work across multiple RTOS systems.
RTOS v2	All Cortex-M, Cortex-A5/A7/A9	Extends CMSIS-RTOS v1 with Armv8-M support, dynamic object creation, provisions for multi-core systems, binary compatible interface.
Pack	All Cortex-M, SecurCore, Cortex-A5/A7/A9	Describes a delivery mechanism for software components, device parameters, and evaluation board support. It simplifies software re-use and product life-cycle management (PLM).
SVD	All Cortex-M, SecurCore	Peripheral description of a device that can be used to create peripheral awareness in debuggers or CMSIS-Core header files.
DAP	All Cortex	Firmware for a debug unit that interfaces to the CoreSight Debug Access Port.
Zone	All Cortex-M	Defines methods to describe system resources and to partition these resources into multiple projects and execution areas.



CMSIS

Tips: 1st Assignment

1. Create an assembly (.s) file in your source folder
2. In the assembly source, declare the code as a global function using .global and .type:

```
     .global  myadd
     .p2align 2
     .type    myadd,%function

myadd:                      // Function "myadd" entry point.
     .fnstart
     add      r0, r0, r1    // Function arguments are in R0 and R1. Add together
                            // and put the result in R0.
     bx       lr            // Return by branching to the address in the link
                            // register.
     .fnend
```

3. In C code, declare the external function using extern:

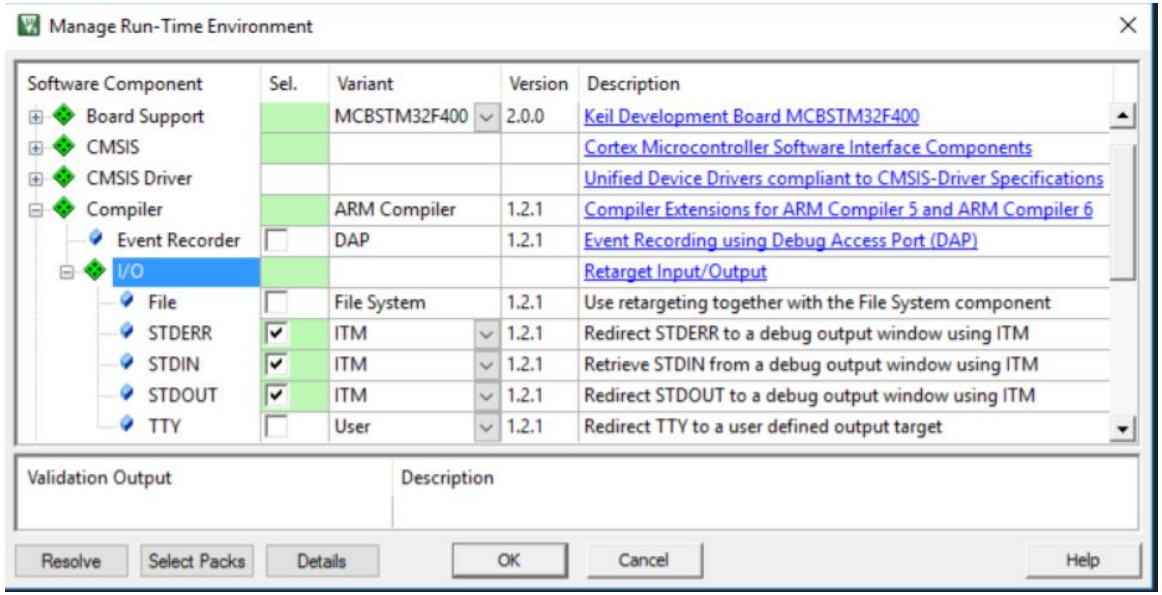
```
#include <stdio.h>

extern int myadd(int a, int b);

int main()
{
  int a = 4;
  int b = 5;
  printf("Adding %d and %d results in %d\n", a, b, myadd(a, b));
  return (0);
}
```

Tips: How to enable Keil-printf (1)

1. Open the Manage RTE dialog and expand the Compiler -- I/O folder
2. Check the STDERR, STDIN and STDOUT components and set the Variant to ITM for each component.

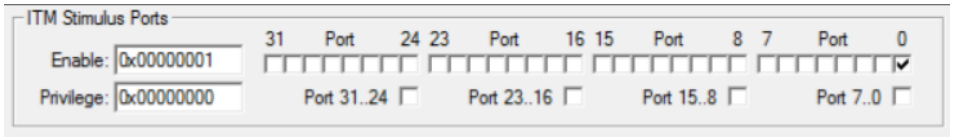


Tips: How to enable Keil-printf (2)

3. Add a debugging trace messages using printf

```
printf("AD value = 0x%04X\r\n", AD_value);
```

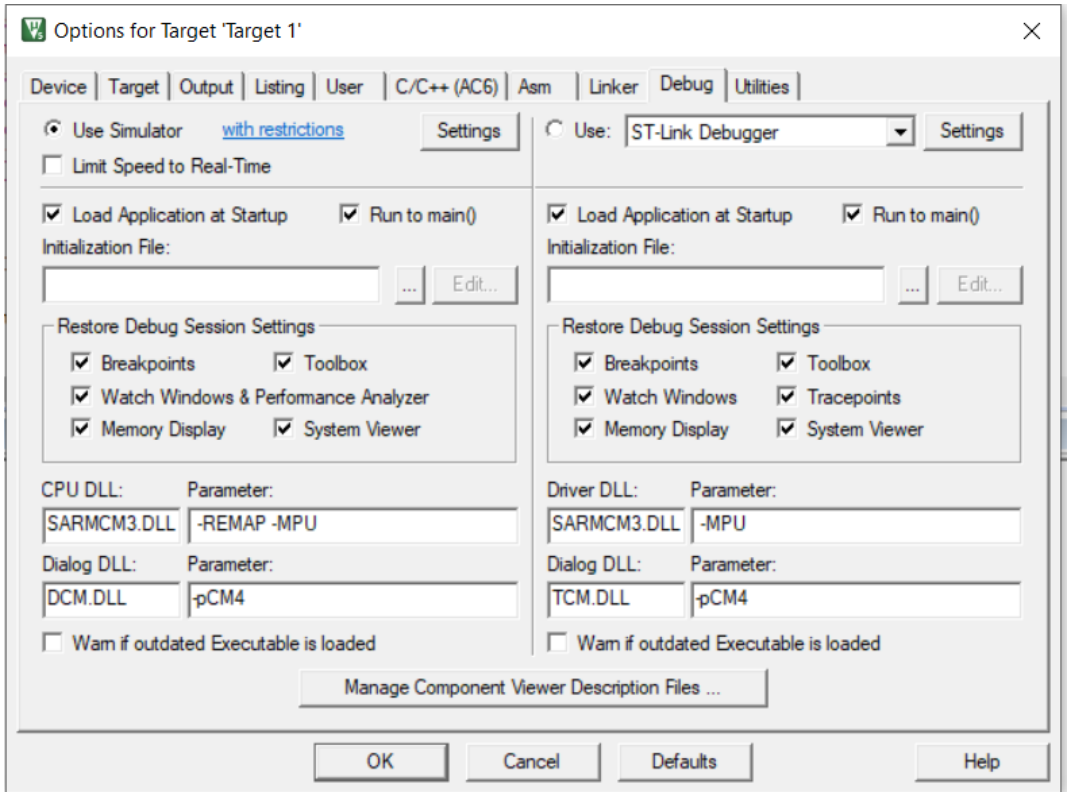
4. Set up the debug session for SWO trace
   - Trace Enable
   - Set Core Clock (usually 16MHz)
5. Set the ITM Port 0 to capture the information. Clear the Privilege Port 7..0 to access ITM Port 0 from User Mode.



6. Open the window from the menu View - Serial Windows - Debug (printf) Viewer.

Tips: How to enable Simulation

1. Click on Options for Target…
2. Click Debug
3. Enable the Use Simulator option



Download keil

https://www.keil.com/download/product/