ETH zürich

Embedded systems with drones – Hands-on lecture

Project-based Learning Center | ETH Zurich

Tommaso Polonelli tommaso.polonelli@pbl.ee.ethz.ch

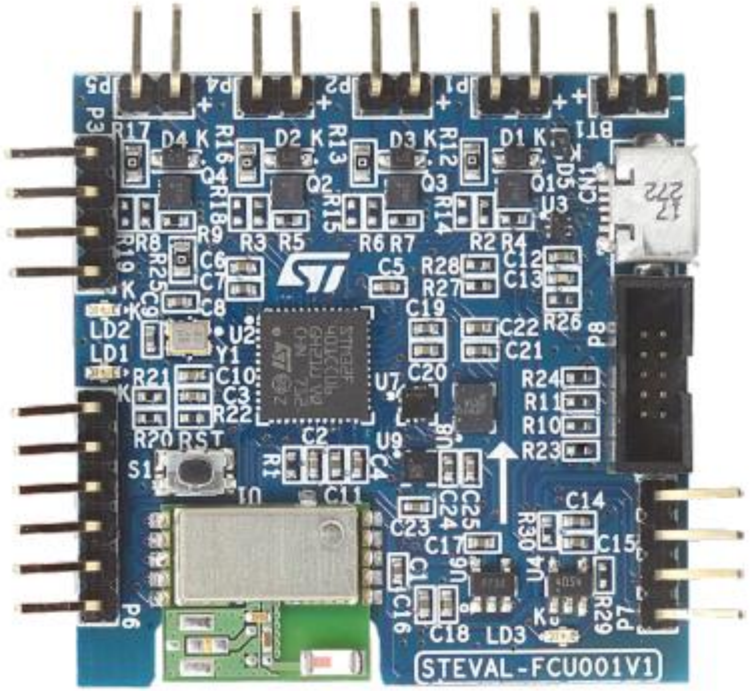
Michele Magno michele.magno@pbl.ee.ethz.ch

Vlad Niculescu vladn@iis.ee.ethz.ch

Evaluation board – STEVAL-FCU001V1

Key Features

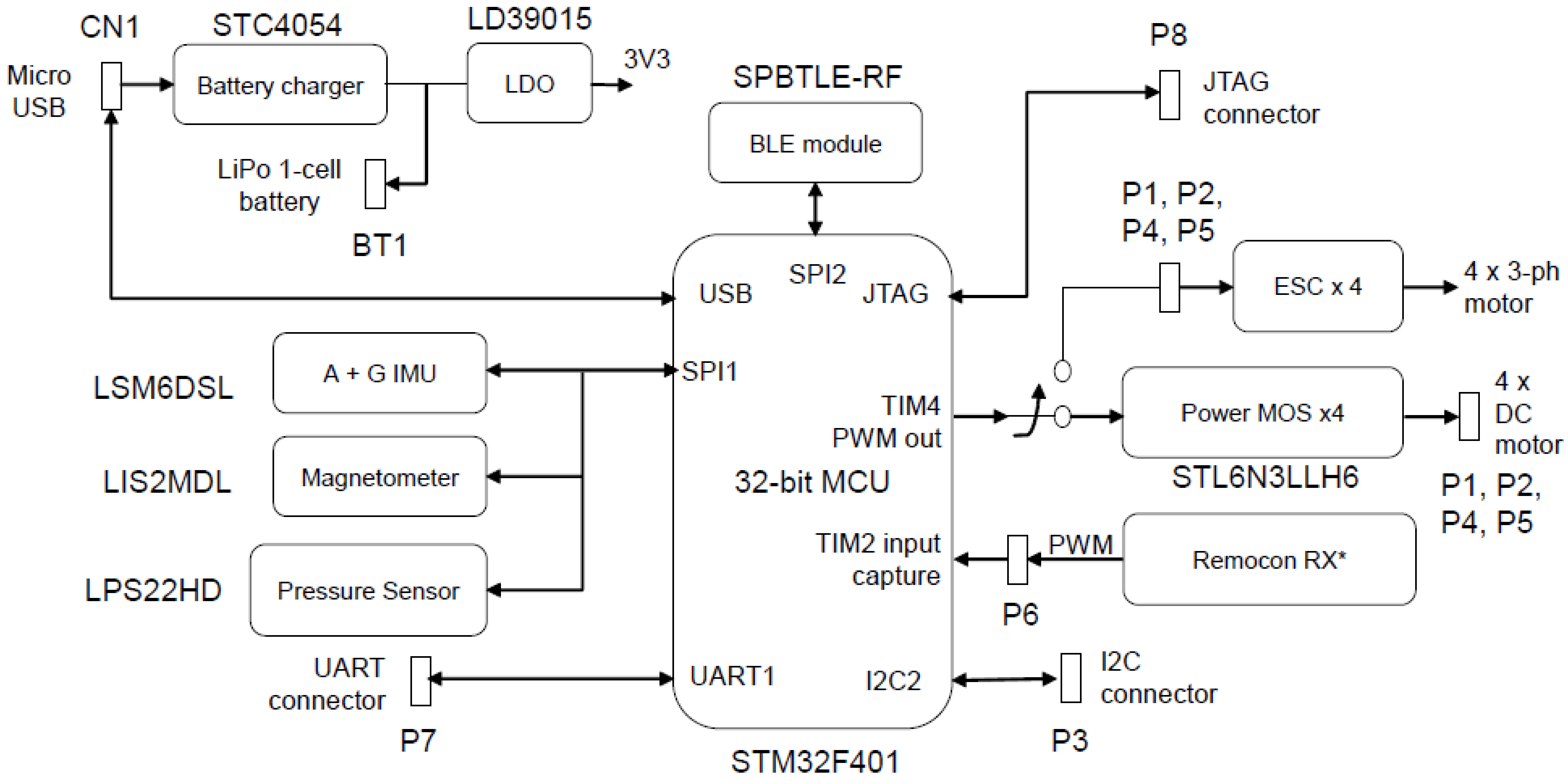
- Compact flight controller unit (FCU) evaluation board complete with sample firmware for a small or medium sized quadcopter
- On-board LiPo 1-cell battery charger
- Possibility to directly drive 4 DC brushed motors through the low voltage onboard MOSFET or alternatively use external ESC for DC brushless motor configuration



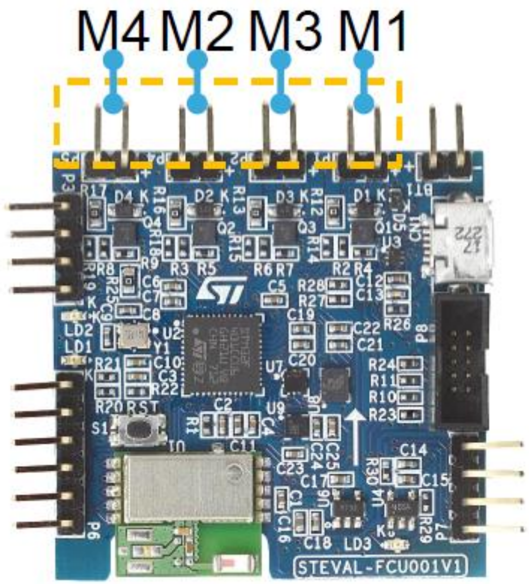
Main Components

- STM32F401 – 32-bit MCU with ARM® Cortex®
- LSM6DSL – iNEMO intertial module: 3D accelerometer and 3D gyroscope
- LIS2MDL – High performance 3D magnetometer
- LPS22HD – MEMS pressure sensor: 260-1260 hPa absolute digital output barometer
- SPBTLE-RF – Very low power module for Bluetooth Smart v4.1
- STL6N3LLH6 - N-channel 30 V, 6 A STripFET H6 Power MOSFET
- STC4054 - 800 mA standalone linear Li-Ion battery charger

Evaluation board – STEVAL-FCU001V1



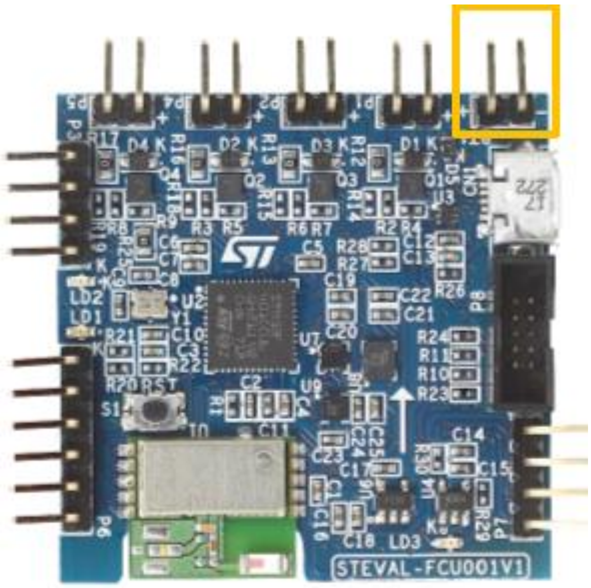
DC motors connected to P5, P4, P2 and P1 connectors.



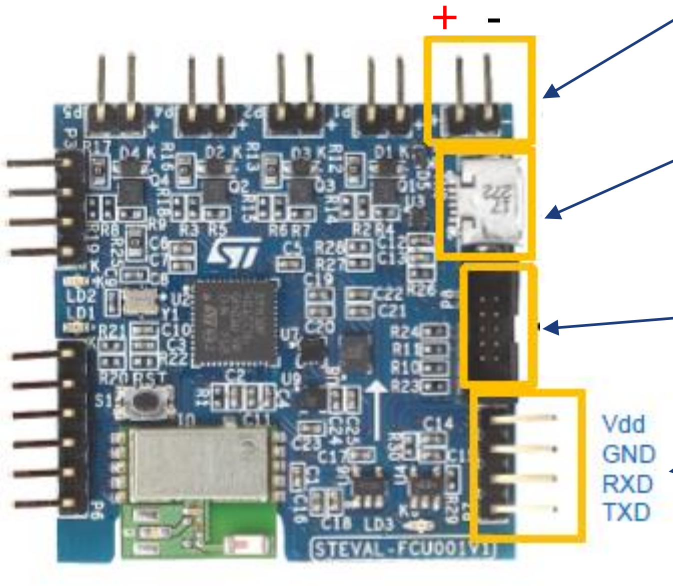
Battery connector

Warning

a reverse battery protection diode is not mounted, check carefully before connecting the battery!



JTAG connection and SW download



battery must be connected

Connect microUSB cable to avoid to discharge the battery.

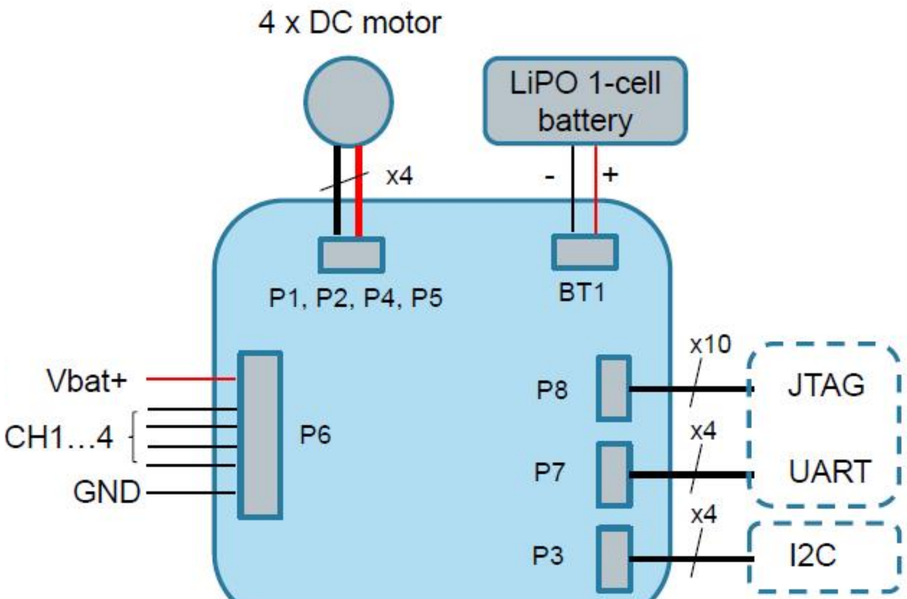
Connector for STLink/V2 → Program and Debug

(optionally) UART can be used for debug and datalog

Note: if only the microUSB cable is connected, the STM32 supply voltage may be unstable and not work properly in debugging.

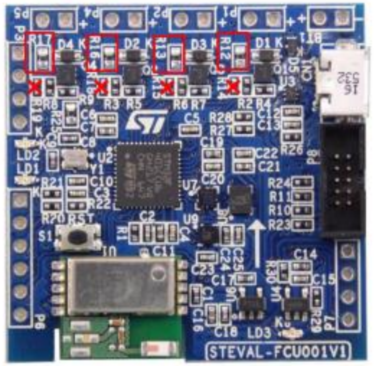
Note: in order to avoid complete LiPo battery discharge during a debugging session, always connect the microUSB cable (connected to PC or charger) to keep charging the battery

STEVAL-FCU001V1 application connection



Used only for debug purposes

Available for future use



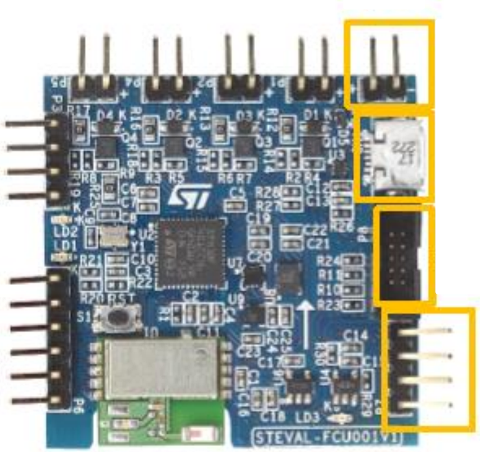
Note: this is the Out of the Box configuration

Resistor Configuration R12, R13, R16, R17 mounted
R14, R15, R18, R19 NOT mounted

Tools for the course

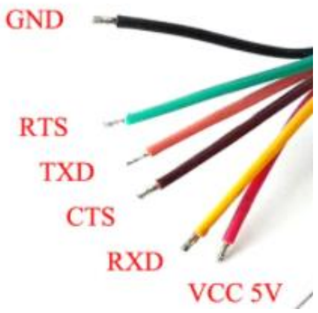
Hardware

Board: STEVAL-FCU001V1



ST-LINK/V2
The JTAG/SWD interfaces are used to communicate with any STM32 microcontroller located on an application board.

TTL-232R-3V3



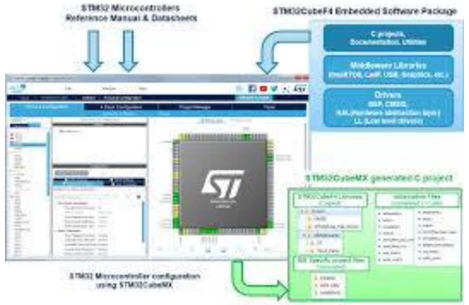
Important document:
Datasheet STM32F401
Reference Manual RM0368

Software

IDE: STM32CubeIDE

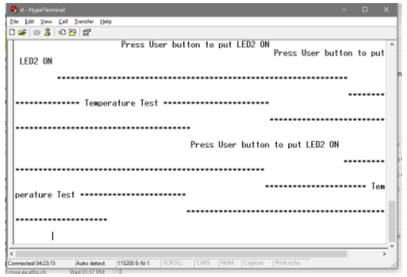
STM32 CubeIDE

STM32CubeMX


STM32 Microcontrollers Reference Manual & Datasheets
STM32CubeF4 Embedded Software Package
C projects, Documentation, Utilities
Middleware Libraries (FreeRTOS, LwIP, USB, Graphics, etc.)
Drivers BSP, CMSIS, HAL (Hardware abstraction layer) LL (Low level drivers)
STM32CubeMX generated C project
STM32 Microcontroller configuration using STM32CubeMX

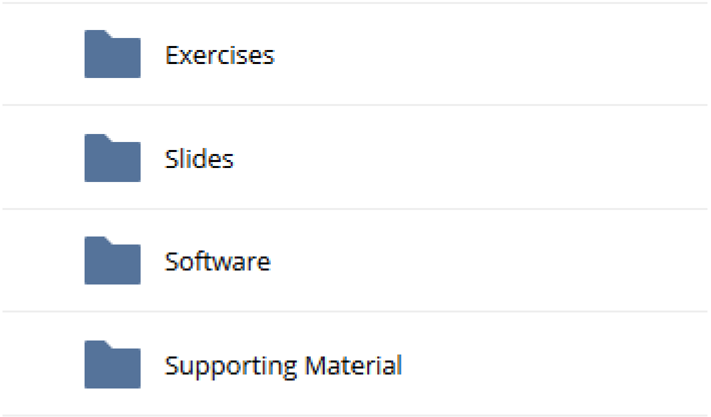
ST BLE Drone

Hyperterminal
Display data flow from serial on your PC



```
                    Press User button to put LED2 ON
                                                      Press User button to put
 LED2 ON
      ****************************************************************
                                                                        ********
************** Temperature Test ***********************
                                                       *************************
***************************************
                                   Press User button to put LED2 ON
                                                                        ********
*******************************************************
                                                      ********************** Tem
perature Test ***********************
                                     ***********************************************
********************
      |
```

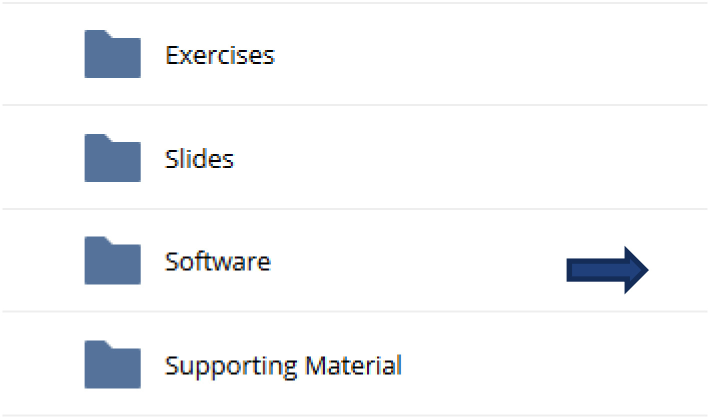
Where to find what? - POLYBOX



- Exercises
- Slides
- Software
- Supporting Material

- Every week new exercises
- Typically before the next session a solution

Where to find what? - Software


Exercises

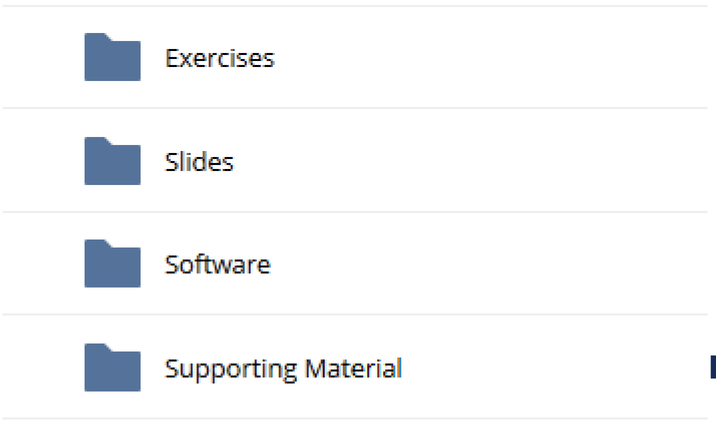
Slides

Software

Supporting Material

- IDE:
  - WINDOWS, LINUX, MAC: STM32CubeIDE - https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-ides/stm32cubeide.html
- Fast HW configuration:
  - STM32CubeMX – https://www.st.com/en/development-tools/stm32cubemx.html
- Drivers and Examples:
  - STM32CubeL4 – (will be automatically downloaded by CubeMX)
- Visualization and HW debugging:
  - Teraterm: https://osdn.net/projects/ttssh2/releases/
  - STM-STUDIO-STM32 – https://www.st.com/en/development-tools/stm-studio-stm32.html

Where to find what? – Important documents

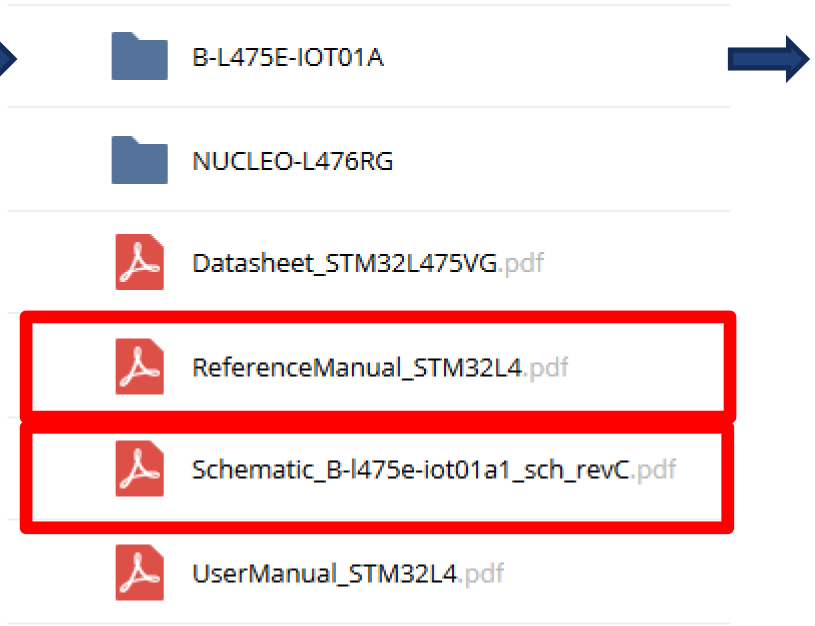


Exercises

Slides

Software

Supporting Material



B-L475E-IOT01A

NUCLEO-L476RG

Datasheet_STM32L475VG.pdf

ReferenceManual_STM32L4.pdf

Schematic_B-l475e-iot01a1_sch_revC.pdf

UserManual_STM32L4.pdf

STEVAL-STWINKT1_UserManual.pdf

Which peripheral?

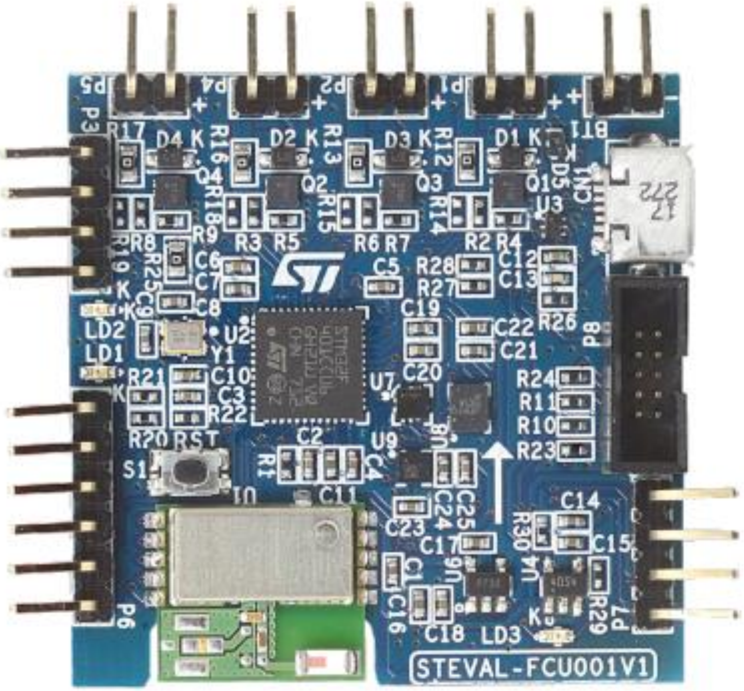
Which GPIO?

- Where is the LED?
- Where is the button?
- Where is the programmer connector?

Important documents

- RM0368 (Reference manual for STM32F4 series)
  - It provides complete information on how to use the STM32F4x microcontroller memory and peripherals.
- UM1884 (Description of HAL and LL for STM32F4 series)
  - It provides complete information on the HAL and LL libraries
- Board schematics for **STEVAL-FCU001V1**
  - schematic diagram shows the components and interconnections of the circuit. Useful to find the pin connection and sensor diagram

  - Useful links:
    - https://www.st.com/en/microcontrollers-microprocessors/stm32f401vc.html
    - https://www.st.com/content/st_com/en/products/evaluation-tools/solution-evaluation-tools/sensor-solution-eval-boards/steval-fcu001v1.html#overview



Programming Tips

The C programming language is the **most popular** programming language for **programming embedded systems**

- C is very popular among microcontroller developers due to the **code efficiency** and **reduced development time**.
- C offers **low-level control** and is more readable than assembly.
- Additionally, using C increases **portability**, since C code can be compiled for different types of processors.
- Wide availability of existing **libraries** and **examples** ready to use.

Cross Compiler

Cross compiler → Compiler capable to create executable code for **a platform other than one in which the compiler is running.** Cross compiler tools are used to generate executable code for embedded system or multiple platforms:

- The **GNU Compiler Collection** (**GCC**) is an open source compiler system produced by the **GNU Project** supporting various programming languages.

Integrated Design Evironment (**IDE**) is a software application that **provides comprehensive facilities to computer programmers for software development**

An IDE normally consists of:

- source code **editor**
- **compiler** (or **cross-compiler**)
- **Assembler**
- **Linker**
- **Debugger**

KEIL™ Tools by ARM

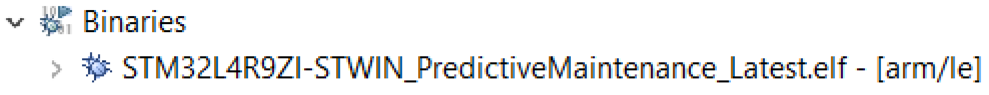
IAR SYSTEMS

CodeWarrior™

eclipse

ST life.augmented

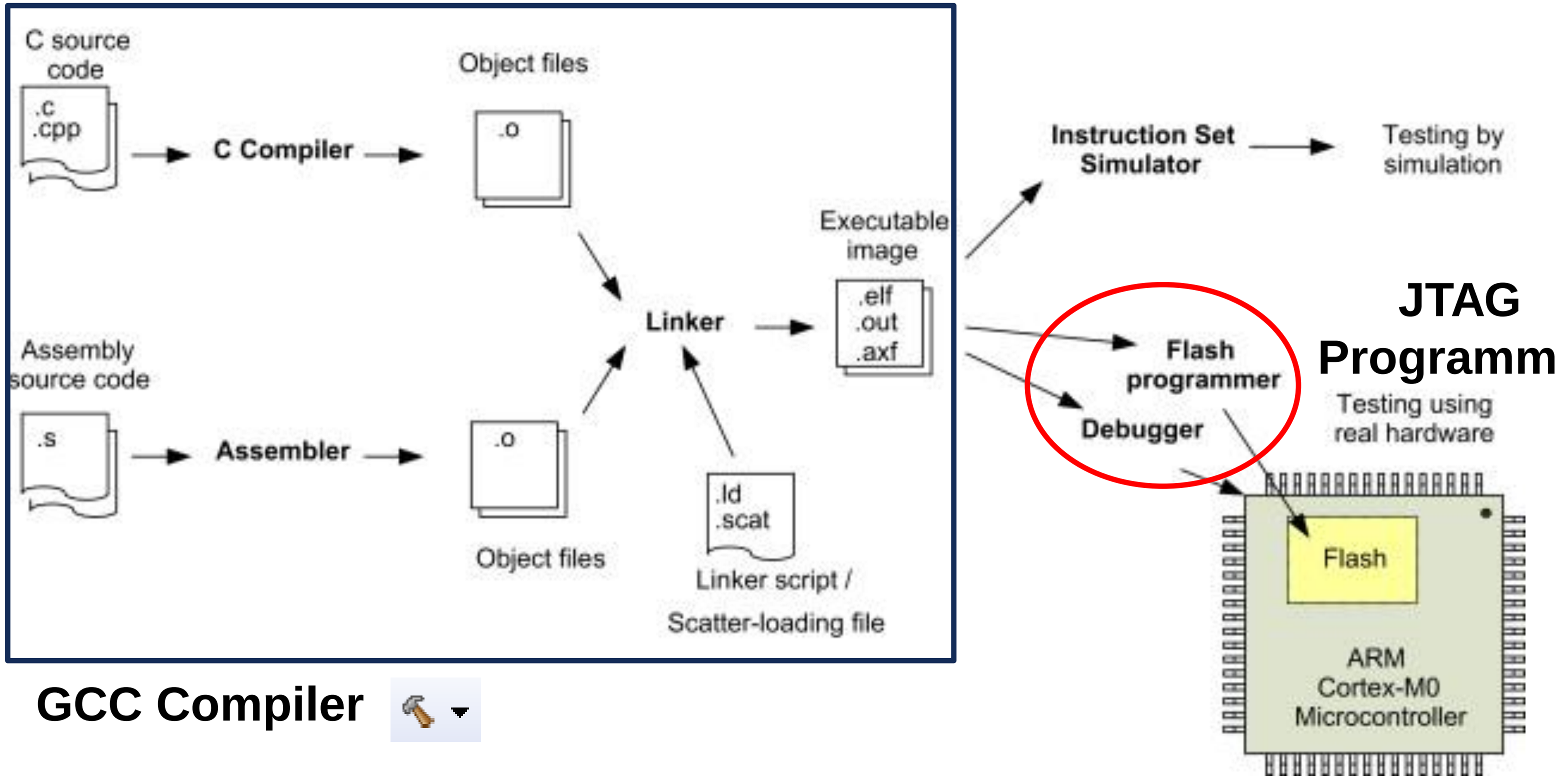
Cross Compiler



Binaries

STM32L4R9ZI-STWIN_PredictiveMaintenance_Latest.elf - [arm/le]

You are here

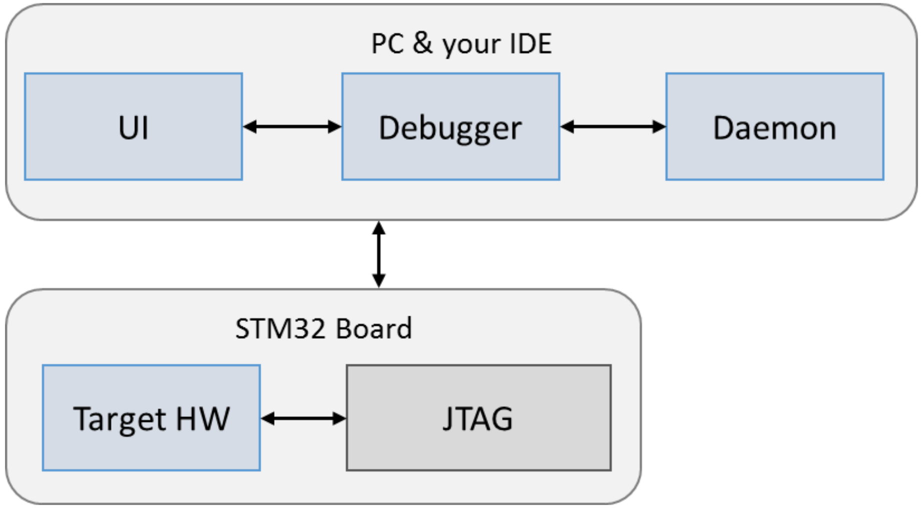


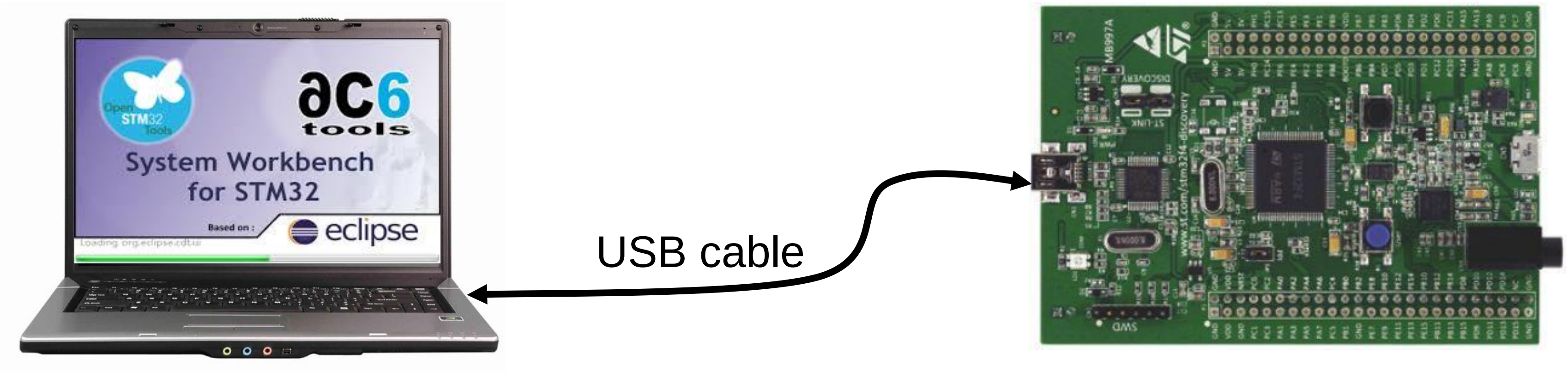
JTAG Programmer

GCC Compiler

JTAG and SWD

- JTAG (Joint Test Action Group) implements standards for **on-chip instrumentation for device testing**.
- It specifies the use of a **dedicated debug port** implementing a **serial communications interface for low-overhead access** to the system components (registers and memory).
- It allows the device programmer to **transfer data** into the **internal memory**.
- It is used for device programming and for active debugging (testing) of the executed program.





Writing the code

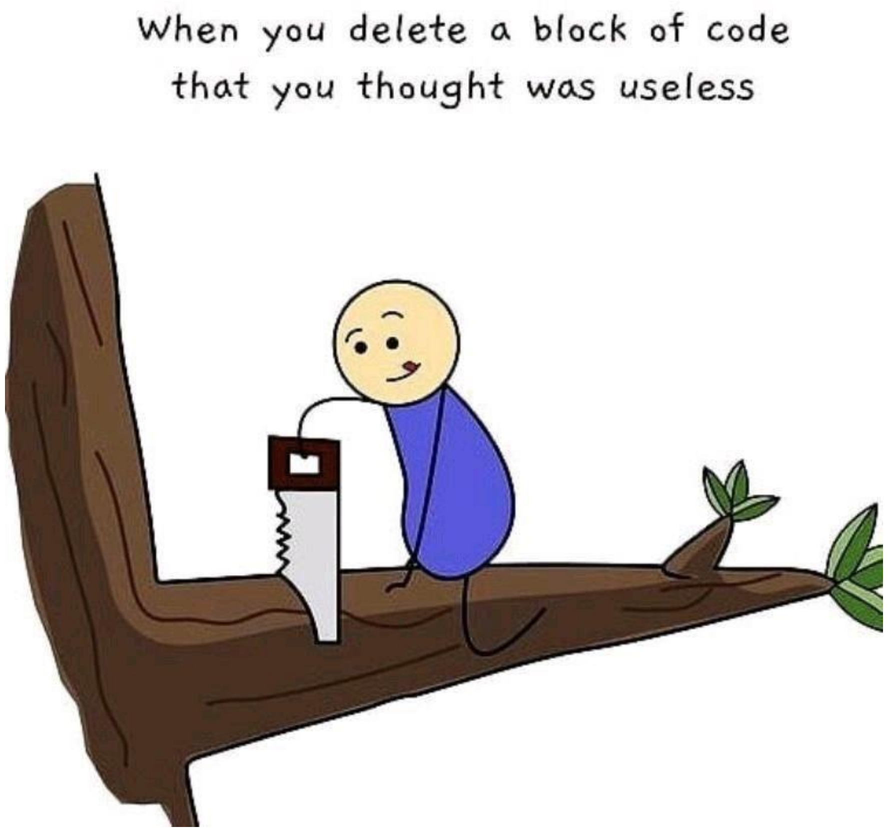
- Use **lots of comments** to provide enough information to fully understand the code

```
/* This is a comment */
```

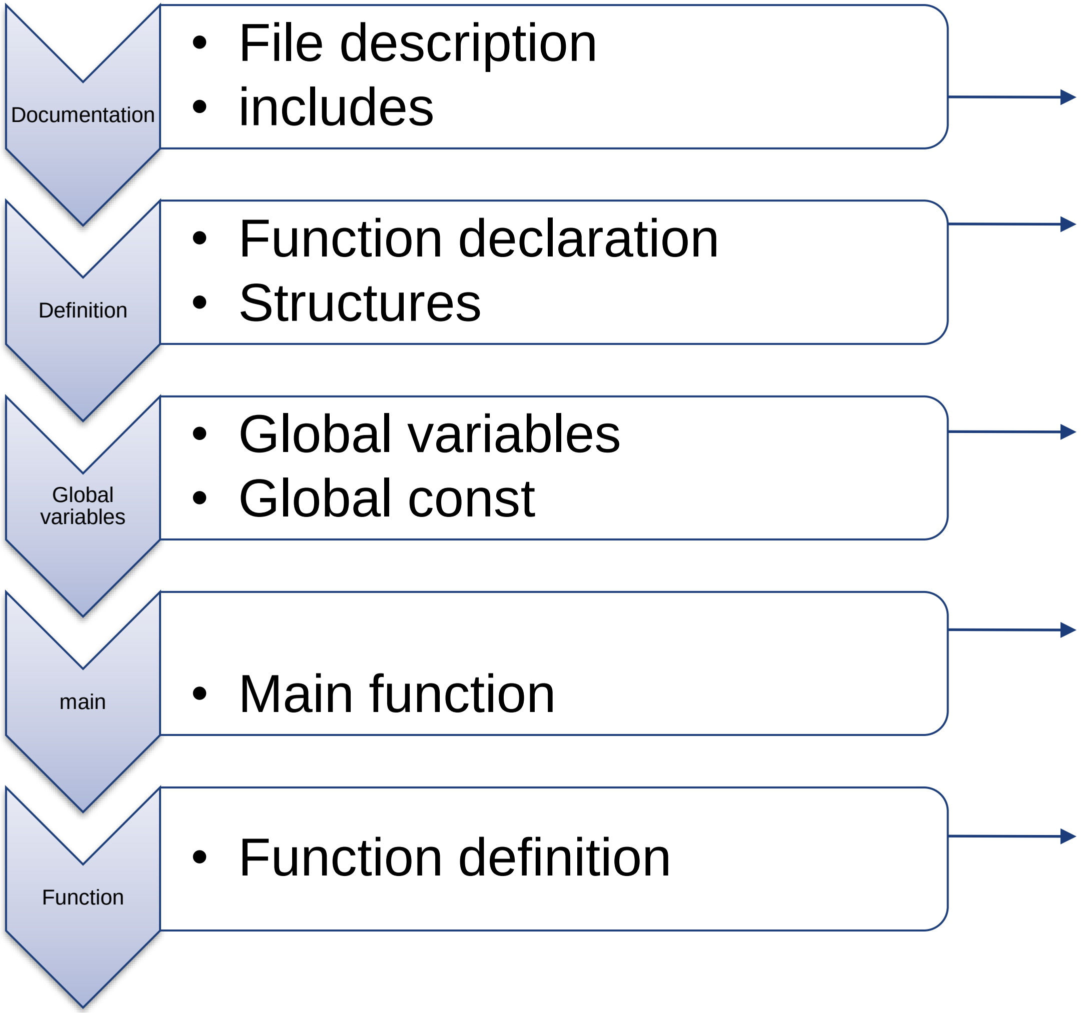
- Comment each procedure telling:

```
/*----------------------------------*/
/* ProcedureName – what it does     */
/* Parameters:                      */
/*   Param1 – what param1 is        */
/*   Param2 – what param2 is        */
/* Returns:                         */
/*   What is returned, if anything  */
/*----------------------------------*/
```

- Use lots of **white space** or blank lines
- Always use **indentation**:
  - **Align** code blocks to **highlight** the functionality
  - Each new scope is indented 2 spaces from previous
  - Put { on end of previous line or start of next line
  - Line matching } up below

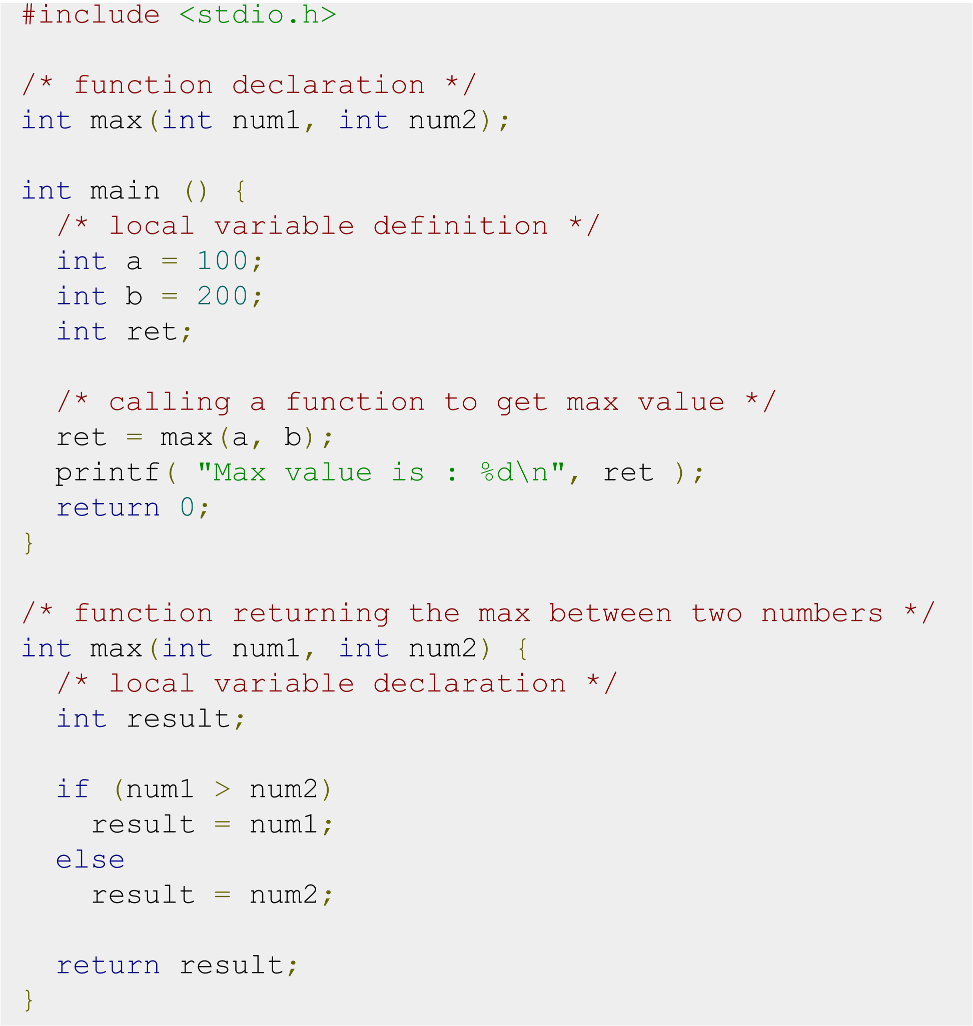


Basic C structure



```
******************************************************************************
 * @file    iis2mdc.c
 * @author  MEMS Software Solutions Team
 * @brief   IIS2MDC driver file
 ******************************************************************************

/* Includes ------------------------------------------------------------------*/
#include "iis2mdc.h"


/* Private function prototypes -----------------------------------------------*/
static void SystemClock_Config(void);


/* Exported Variables -------------------------------------------------------------
*/
volatile uint8_t AccIntReceived= 0;
volatile uint8_t FifoEnabled = 0;
volatile uint32_t PredictiveMaintenance = 0;


int main(void)
{
 SystemClock_Config();
 FifoEnabled = 1;
}


void SystemClock_Config(void)
{
…
…
}
```

Calling a function



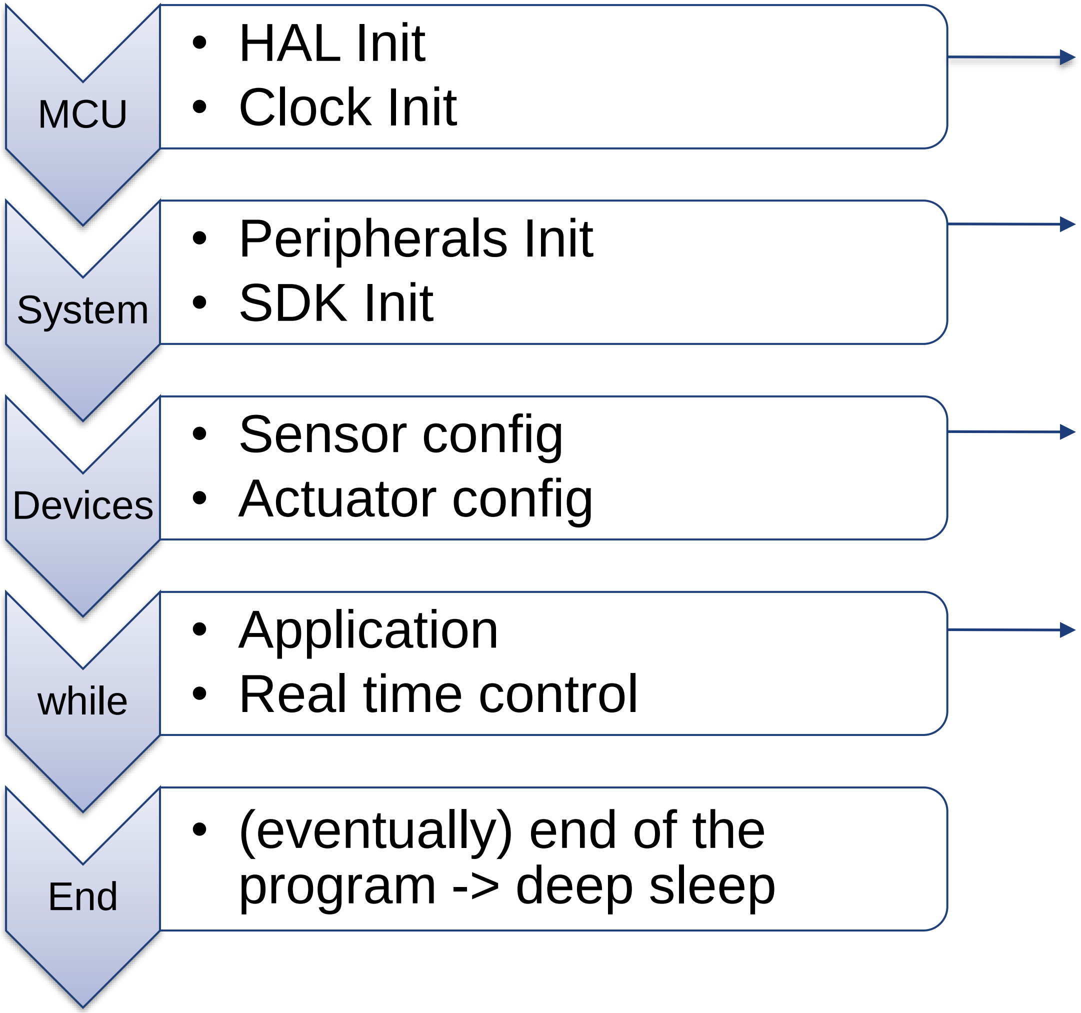
```
#include <stdio.h>

/* function declaration */
int max(int num1, int num2);

int main () {
  /* local variable definition */
  int a = 100;
  int b = 200;
  int ret;

  /* calling a function to get max value */
  ret = max(a, b);
  printf( "Max value is : %d\n", ret );
  return 0;
}

/* function returning the max between two numbers */
int max(int num1, int num2) {
  /* local variable declaration */
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

While creating a C function, you give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task.

When a program calls a function, the program control is transferred to the called function. A called function performs a defined task and when its return statement is executed or when its function-ending closing brace is reached, it returns the program control back to the main program.

To call a function, you simply need to pass the required parameters along with the function name, and if the function returns a value, then you can store the returned value.

Basic application structure - main.c



MCU
- HAL Init
- Clock Init

System
- Peripherals Init
- SDK Init

Devices
- Sensor config
- Actuator config

while
- Application
- Real time control

End
- (eventually) end of the program -> deep sleep

```
int main(void)
{
  HAL_Init();

  /* Configure the System clock */
  SystemClock_Config();

 /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_ADC1_Init();
  MX_DAC1_Init();

 /* Set Accelerometer Full Scale to 2G */
  Set2GAccelerometerFullScale();
  /* initialize timers */
  InitTimers();
  /* set unconnectable */
  set_connectable = FALSE;

 /* Infinite loop */
  while (1){
    /* Led Blinking when there is not a client connected */

   … your application code here …

  }
```

Declaration: Local versus Global

- **Global variables:**
  - Declared outside of a function
  - Accessible throughout the code
  - Stored in Global Data Section of memory
  - Scope is entire program
  - Initialized to zero
- **Local variables:**
  - Declared within one function and are only accessible during its execution
  - Declared at the beginning of a block
  - Stored on the stack
  - Scope is from point of declaration to the end of the block
  - Un-initialized
- **Some tips:**
  - Keep declarations as close as you can to where you use the variables– keep the scope as small as you can.
  - It’s better to explicitly pass parameters to functions, rather than use global variables.

`const`:

- Used to declare a constant (content is not changed in the course of code implementation);
- Stored in program section memory.

`extern`:

- Used to make reference to variables declared elsewhere, for example in another module.

`register`:

- Used to store a variable in a processor’s register;
- Promotes faster access to the contents of the variable;
- Only used locally and depends on the register’s availability.

`static`:

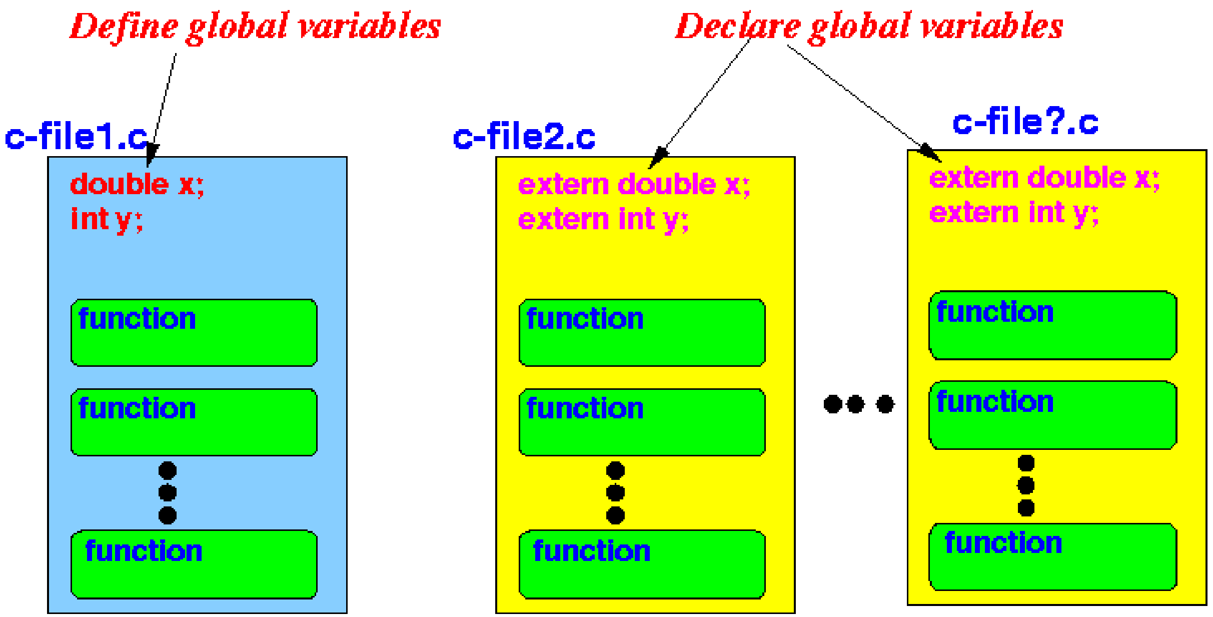
- Declared within a function or a program block;
- Preserves the variable even after a function or block has been executed.

`volatile`:

- A statement using this descriptor informs the compiler that this variable should not be optimized.

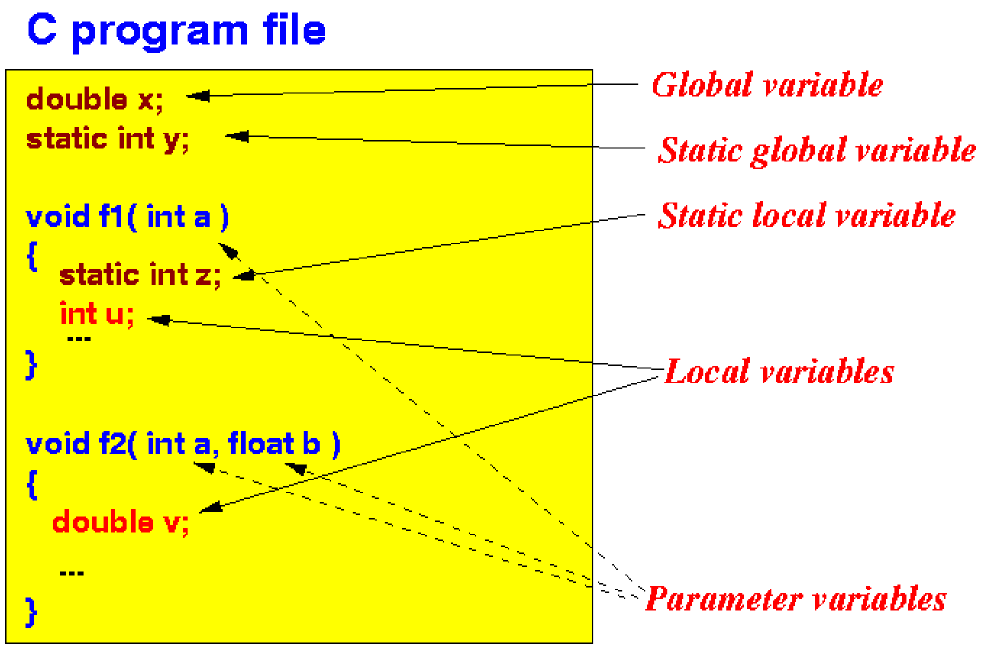
Declaration: Local versus Global

extern:



https://www.mathcs.emory.edu/

static:



Primitive Numeric Data Types

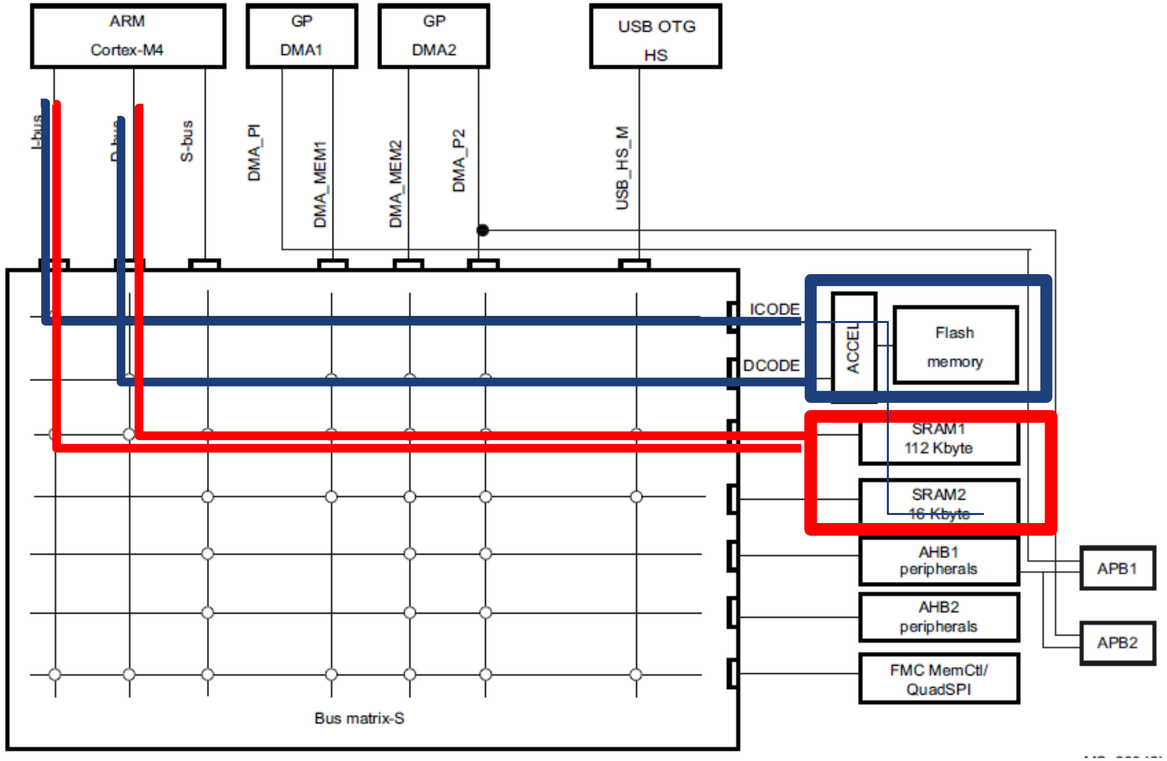
Data Type		Size	Range
byte	u8/s8	1 byte	Integers in the range of -128 to +128
short	u16/s16	2 bytes	Integers in the range of -32,768 to +32,767
int	u32/s32	4 bytes	Integers in the range of -2,147,483,648 to +2,147,483,647
long	u64/s64	8 bytes	Integers in the range of -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807
float	float	4 bytes	Floating-point numbers between ±3.4x10^{-38} to ±3.4x10^{38} with 7 digits of accuracy
double	double	8 bytes	Floating-point numbers between ±1.7x10^{-308} and ±1.7x10^{308} with 15 digits of accuracy

Memory Organization (Physical)

Embedded systems have memory constraints

↓

Memory allocation is very important



Physical allocation:

- RAM (**Data**)
- Flash (**Code**)

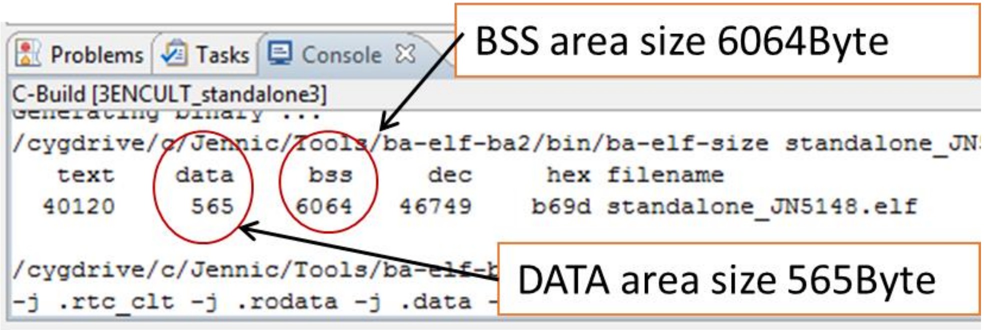
With exceptions!

(some code can be stored in RAM and some Data can be stored in Flash)

Memory Segments

In classical architectures there are 4 basic memory areas:

- **DATA → initialized global and static variables.**
- **BSS** (*Block Started by Symbol*) → **uninitialized global and static variables**.
- **HEAP → Dynamically allocated memory**.
  (Malloc operations are **<u>not recommended</u>** in embedded software)
- **TEXT →**Code segment that contains **executable instructions**.



C-Build [3ENCULT_standalone3]

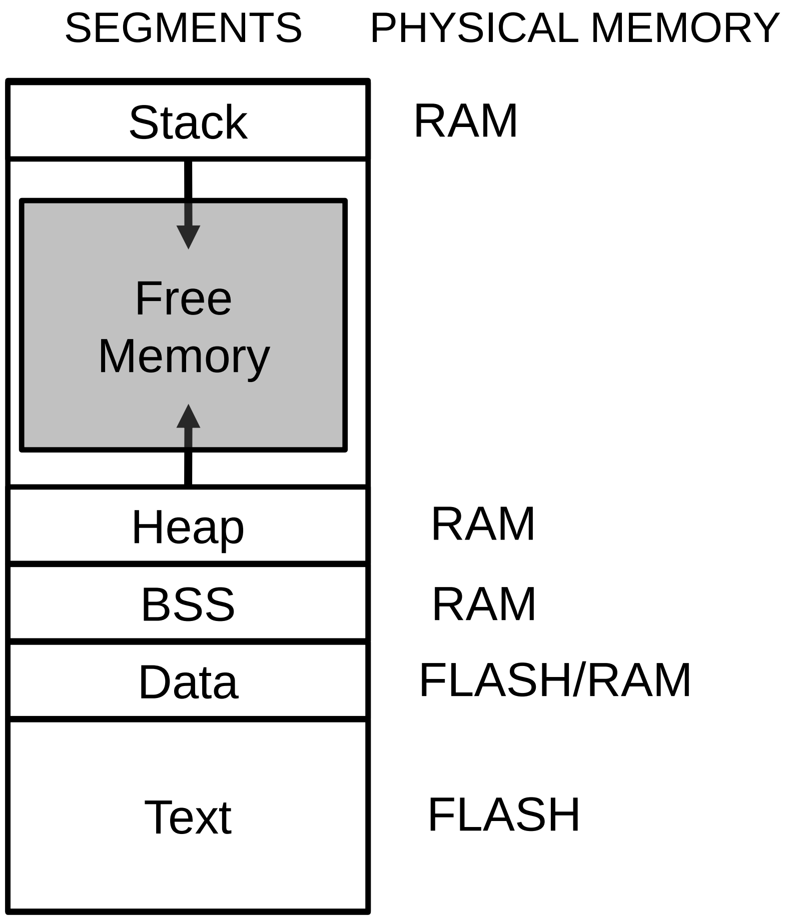
```
Generating binary ...
/cygdrive/c/Jennic/Tools/ba-elf-ba2/bin/ba-elf-size standalone_JN
   text    data     bss     dec     hex filename
  40120     565    6064   46749    b69d standalone_JN5148.elf

/cygdrive/c/Jennic/Tools/ba-elf-b
-j .rtc_clt -j .rodata -j .data -
```

BSS area size 6064Byte

DATA area size 565Byte

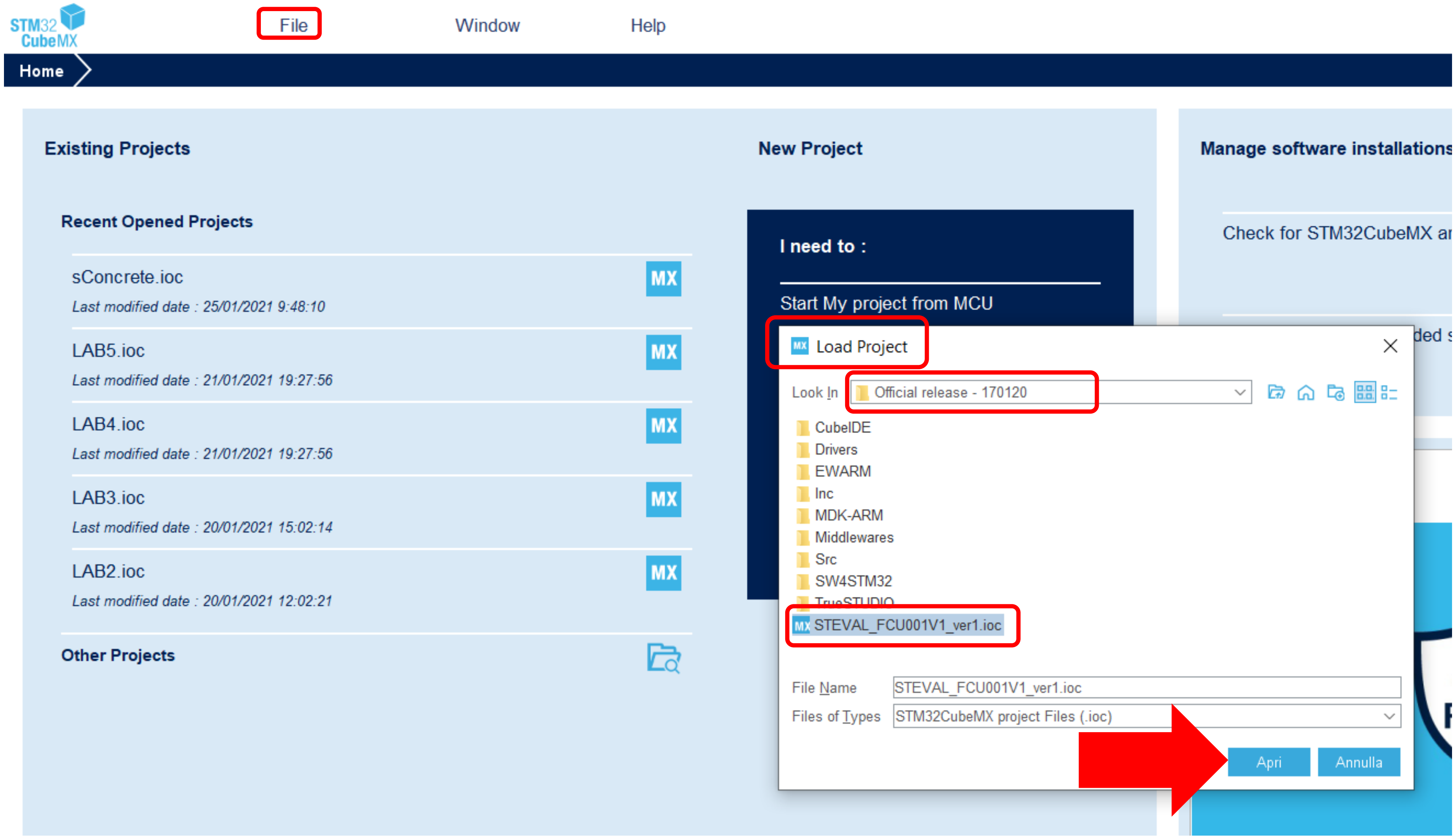
Memory allocations

```
int data_bss;            }  ←BSS
int iSize;

int data_data=32;           ←DATA


void main(void)
{
   int a =2;             }  ←STACK
   char *p;

   static int b;         }  ←DATA
   static int c=11;

   iSize = 8;
   p = malloc(iSize);       ←HEAP (8 Bytes)

   a=example();

}
```



Let‘s go!

STM32CubeMx



File Window Help

Home

Existing Projects

Recent Opened Projects

sConcrete.ioc
Last modified date : 25/01/2021 9:48:10

LAB5.ioc
Last modified date : 21/01/2021 19:27:56

LAB4.ioc
Last modified date : 21/01/2021 19:27:56

LAB3.ioc
Last modified date : 20/01/2021 15:02:14

LAB2.ioc
Last modified date : 20/01/2021 12:02:21

Other Projects

New Project

I need to :

Start My project from MCU

Manage software installations

Check for STM32CubeMX ar

ded s

Load Project

Look In: Official release - 170120

CubeIDE
Drivers
EWARM
Inc
MDK-ARM
Middlewares
Src
SW4STM32
TrueSTUDIO
STEVAL_FCU001V1_ver1.ioc

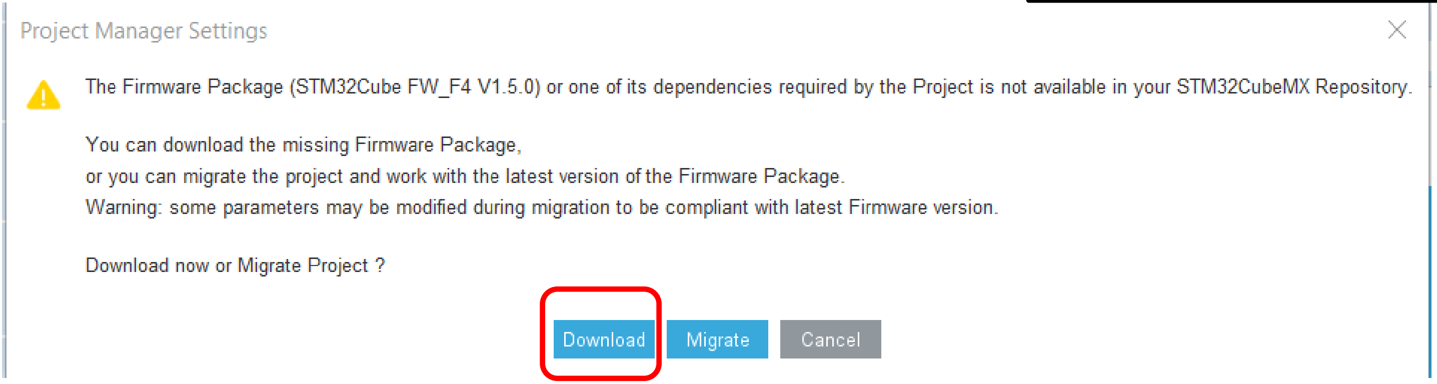
File Name: STEVAL_FCU001V1_ver1.ioc

Files of Types: STM32CubeMX project Files (.ioc)

STM32CubeMx

Supported versions:

- STM32CubeMx v6.0.1
- STM32Cube MCU Package for STM32F4 Series 1.5.0
- STM32IDE v1.01



Project Manager Settings

The Firmware Package (STM32Cube FW_F4 V1.5.0) or one of its dependencies required by the Project is not available in your STM32CubeMX Repository.

You can download the missing Firmware Package,
or you can migrate the project and work with the latest version of the Firmware Package.
Warning: some parameters may be modified during migration to be compliant with latest Firmware version.

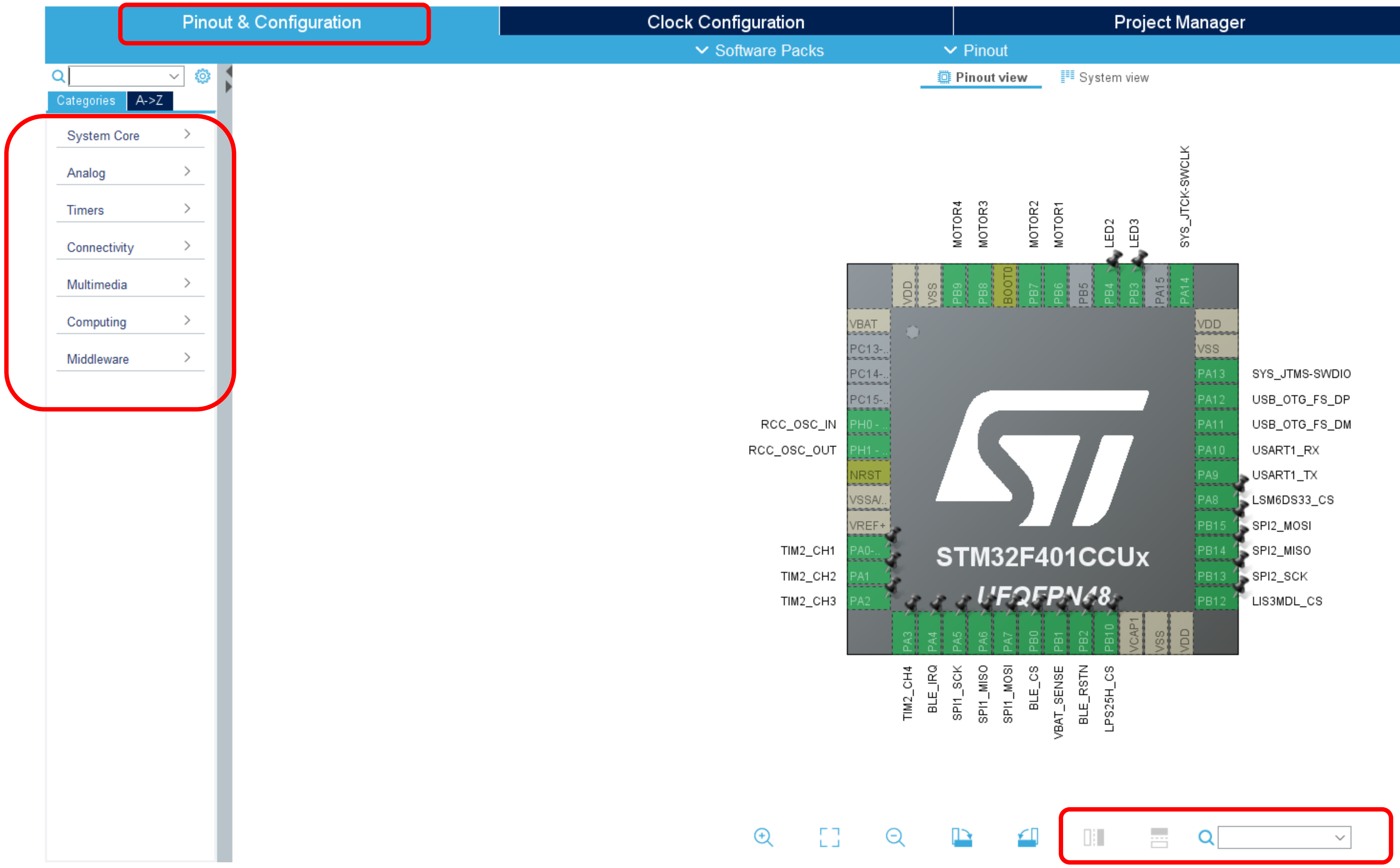
Download now or Migrate Project ?

Default settings of:

- GPIOs
- Clock
- Peripherals

Default settings following the **STEVAL-DRONE01** schematic.
Cube MX knows which GPIO is connected to a specific interface, sensor, serial BUS, etc.

STM32CubeMx



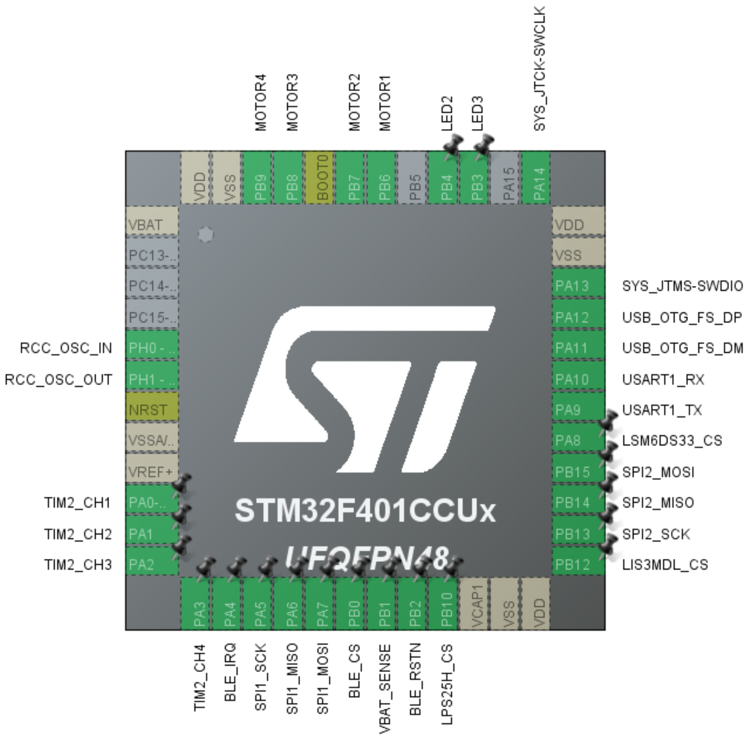
Default settings of:

- GPIOs
- Clock
- Peripherals

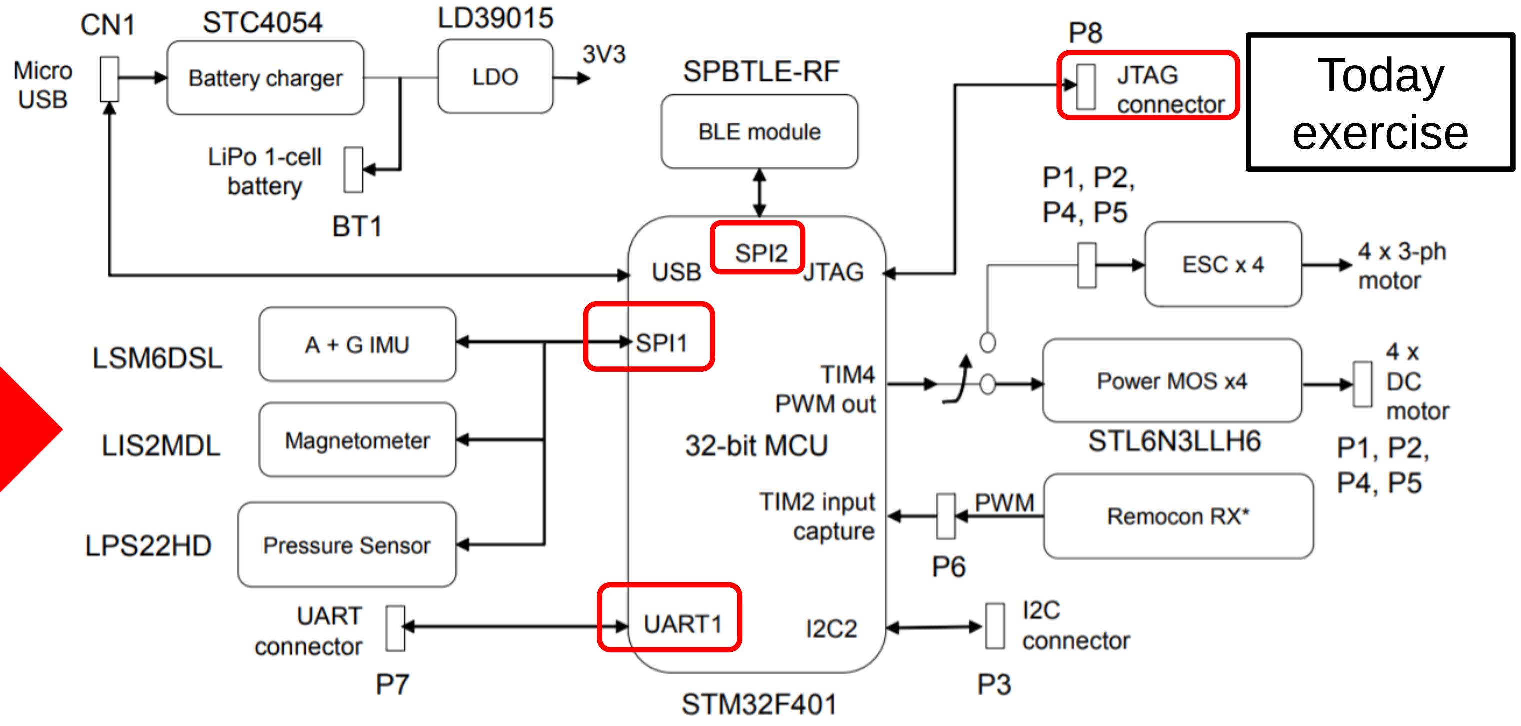
Linked with STEVAL-FCU001V1 block diagram

Search for a specific GPIO name

STM32CubeMx

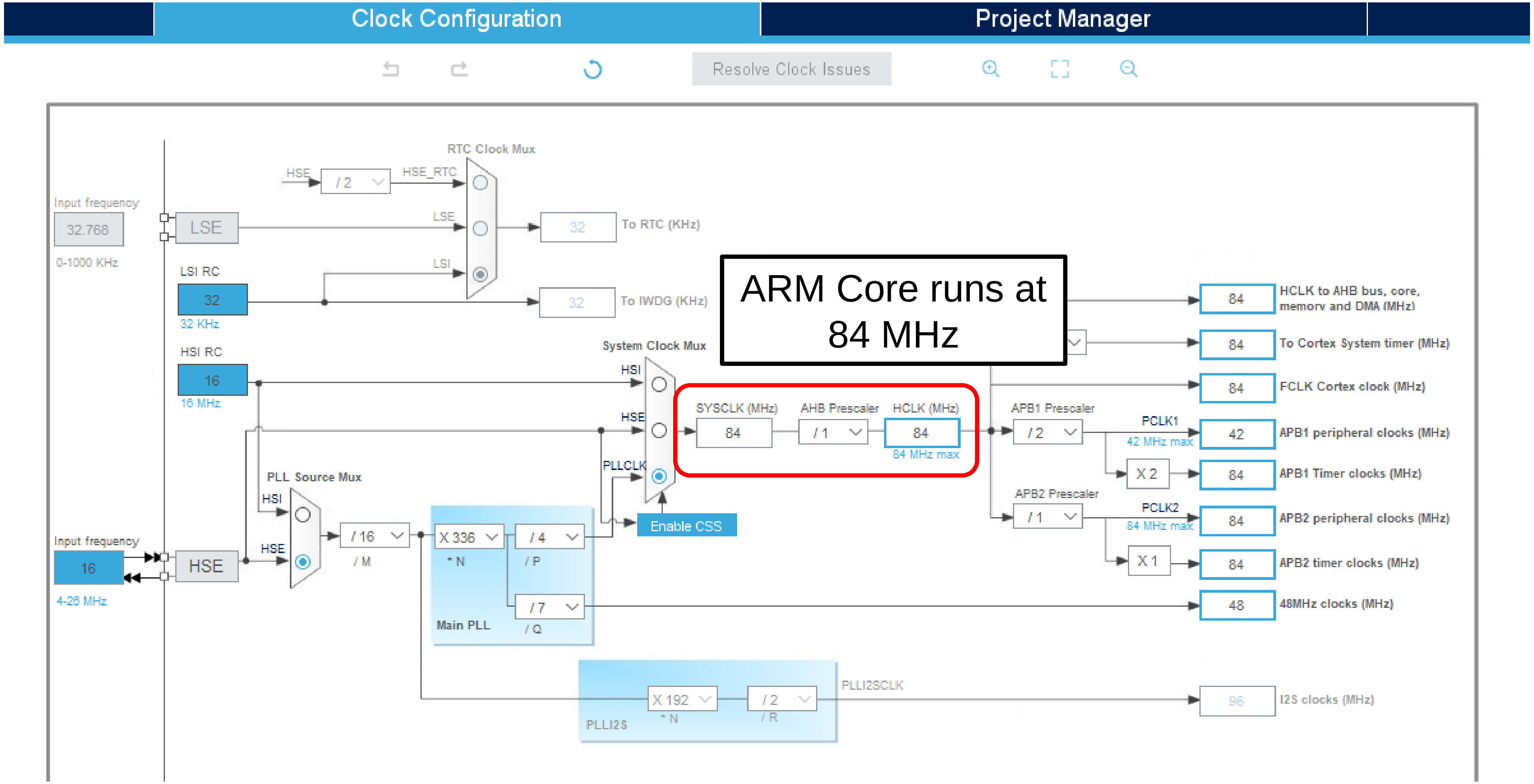


steval-fcu001v1.pdf



Today exercise

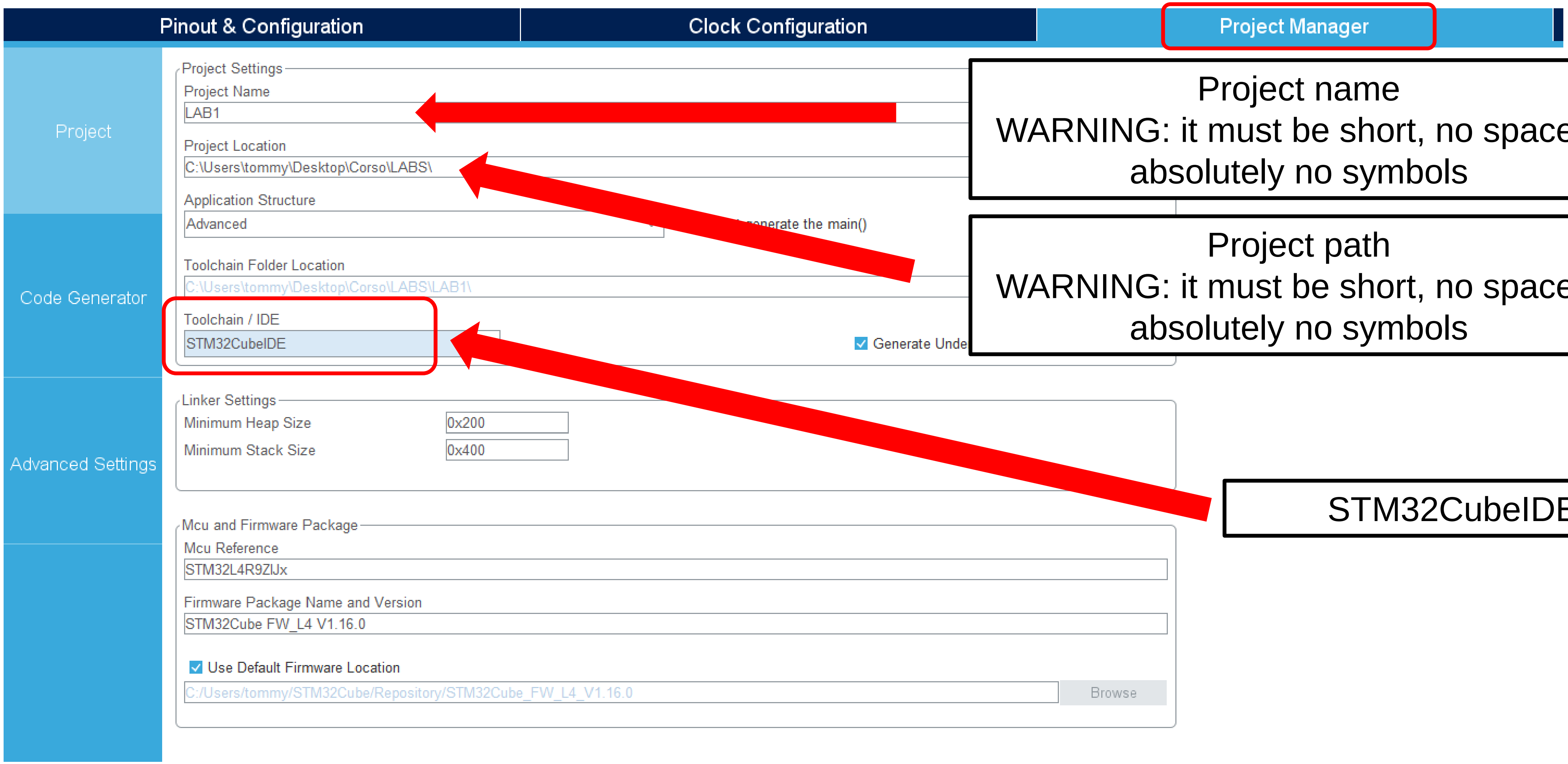
STM32CubeMx



ARM Core runs at 84 MHz

STM32CubeMx

Manages the project folder and IDE



Project Settings

Project Name: LAB1

Project Location: C:\Users\tommy\Desktop\Corso\LABS\

Application Structure: Advanced ... nerate the main()

Toolchain Folder Location: C:\Users\tommy\Desktop\Corso\LABS\LAB1\

Toolchain / IDE: STM32CubeIDE

☑ Generate Unde

Linker Settings

Minimum Heap Size: 0x200

Minimum Stack Size: 0x400

Mcu and Firmware Package

Mcu Reference: STM32L4R9ZIJx

Firmware Package Name and Version: STM32Cube FW_L4 V1.16.0

☑ Use Default Firmware Location

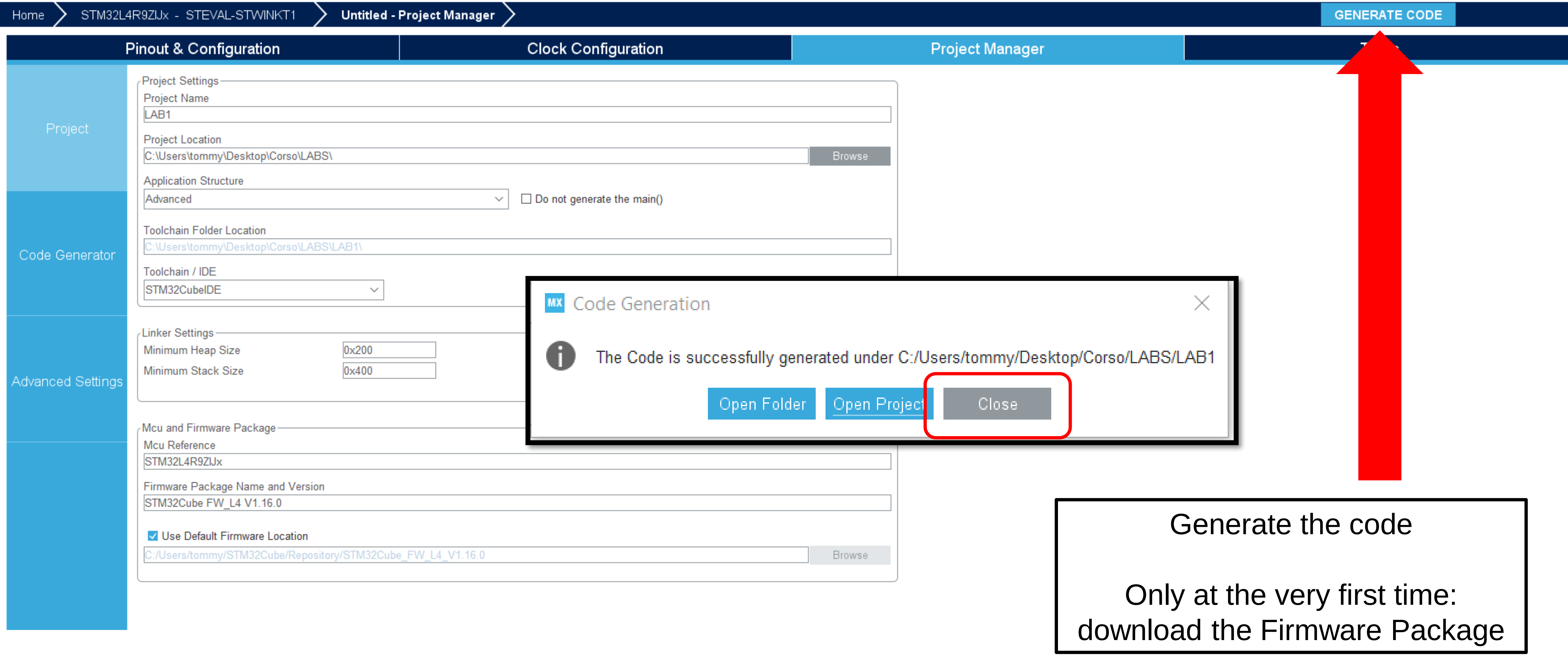
C:/Users/tommy/STM32Cube/Repository/STM32Cube_FW_L4_V1.16.0

Project name
WARNING: it must be short, no spaces, absolutely no symbols

Project path
WARNING: it must be short, no spaces, absolutely no symbols

STM32CubeIDE

STM32CubeMx



Project Settings

Project Name: LAB1

Project Location: C:\Users\tommy\Desktop\Corso\LABS\

Application Structure: Advanced

□ Do not generate the main()

Toolchain Folder Location: C:\Users\tommy\Desktop\Corso\LABS\LAB1\

Toolchain / IDE: STM32CubeIDE

Linker Settings

Minimum Heap Size: 0x200

Minimum Stack Size: 0x400

Mcu and Firmware Package

Mcu Reference: STM32L4R9ZIJx

Firmware Package Name and Version: STM32Cube FW_L4 V1.16.0

☑ Use Default Firmware Location

C:/Users/tommy/STM32Cube/Repository/STM32Cube_FW_L4_V1.16.0

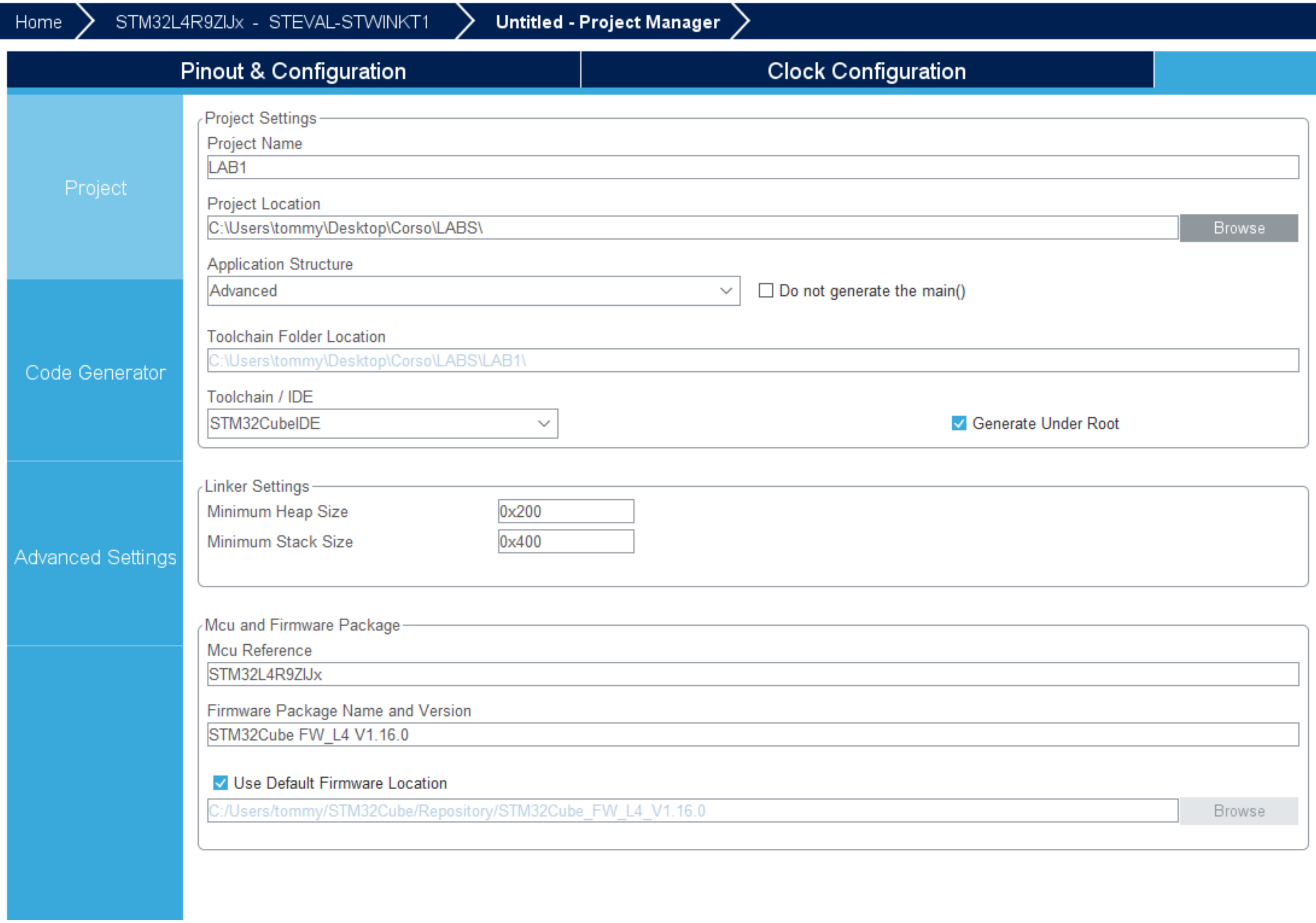
Code Generation

The Code is successfully generated under C:/Users/tommy/Desktop/Corso/LABS/LAB1

Generate the code

Only at the very first time: download the Firmware Package

STM32CubeMx



Project Settings

Project Name: LAB1

Project Location: C:\Users\tommy\Desktop\Corso\LABS\

Application Structure: Advanced

☐ Do not generate the main()

Toolchain Folder Location: C:\Users\tommy\Desktop\Corso\LABS\LAB1\

Toolchain / IDE: STM32CubeIDE

☑ Generate Under Root

Linker Settings

Minimum Heap Size: 0x200

Minimum Stack Size: 0x400

Mcu and Firmware Package

Mcu Reference: STM32L4R9ZIJx

Firmware Package Name and Version: STM32Cube FW_L4 V1.16.0

☑ Use Default Firmware Location

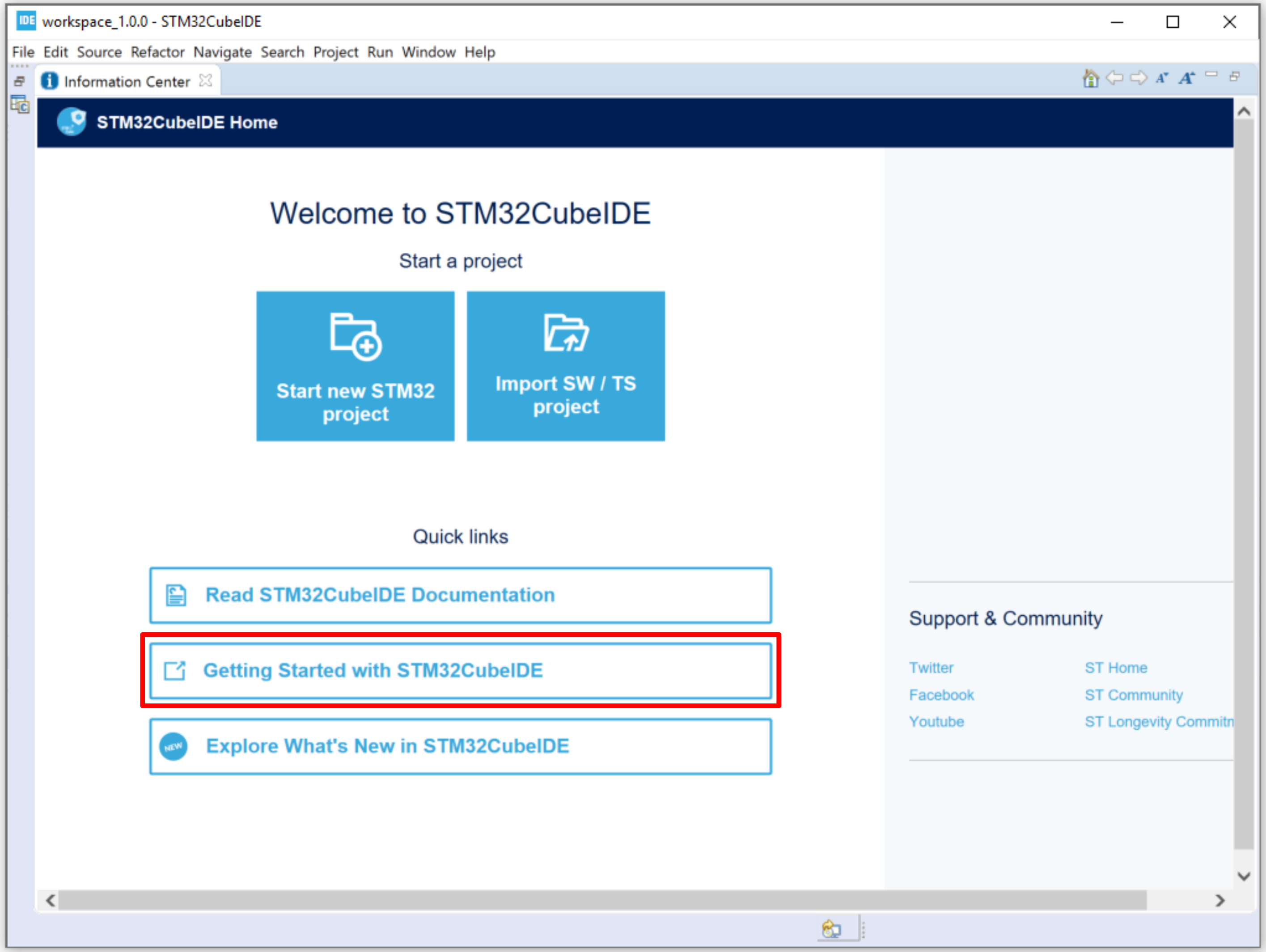
C:/Users/tommy/STM32Cube/Repository/STM32Cube_FW_L4_V1.16.0

CODE is already generated!!
Tool useful for future projects

at e

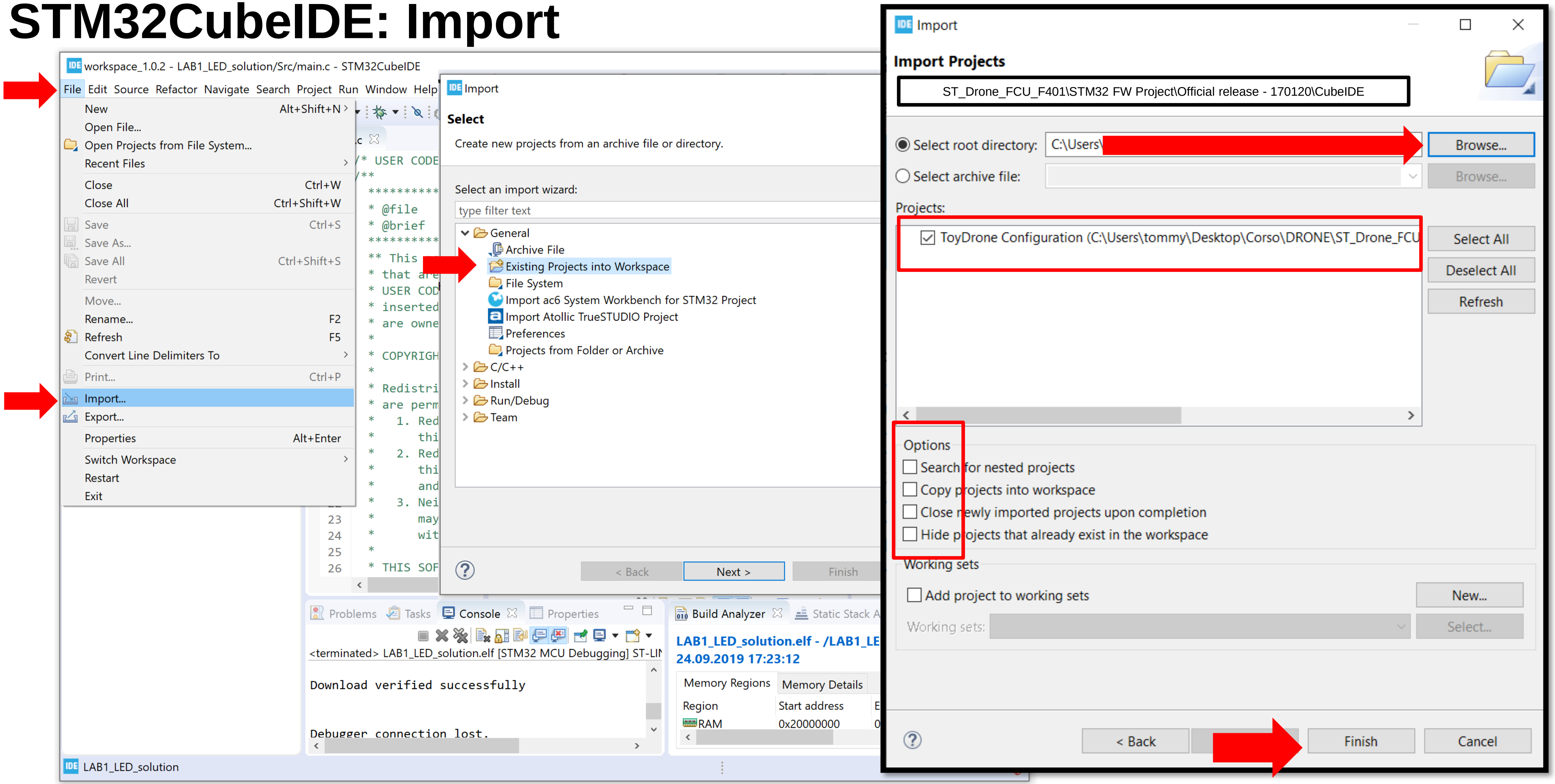
Only a first time:
downlo re Package

STM32CubeIDE: Quickstart Guide



If you use eclipse the first time have a look

STM32CubeIDE: Import



Create new projects from an archive file or directory.

Select an import wizard:

type filter text

```
/* USER CODE
/**
 **********
 * @file
 * @brief
 **********
 ** This
 * that are
 * USER COD
 * inserted
 * are owne
 *
 * COPYRIGH
 *
 * Redistri
 * are perm
 *   1. Red
 *      thi
 *   2. Red
 *      thi
 *      and
 *   3. Nei
 *      may
 *      wit
 *
 * THIS SOF
```

```
<terminated> LAB1_LED_solution.elf [STM32 MCU Debugging] ST-LIN

Download verified successfully


Debugger connection lost.
```

LAB1_LED_solution.elf - /LAB1_LE
24.09.2019 17:23:12

Region	Start address	E
RAM	0x20000000	0

Import Projects

ST_Drone_FCU_F401\STM32 FW Project\Official release - 170120\CubeIDE

● Select root directory: C:\Users\

○ Select archive file:

Projects:

☑ ToyDrone Configuration (C:\Users\tommy\Desktop\Corso\DRONE\ST_Drone_FCU

Options

□ Search for nested projects

□ Copy projects into workspace

□ Close newly imported projects upon completion

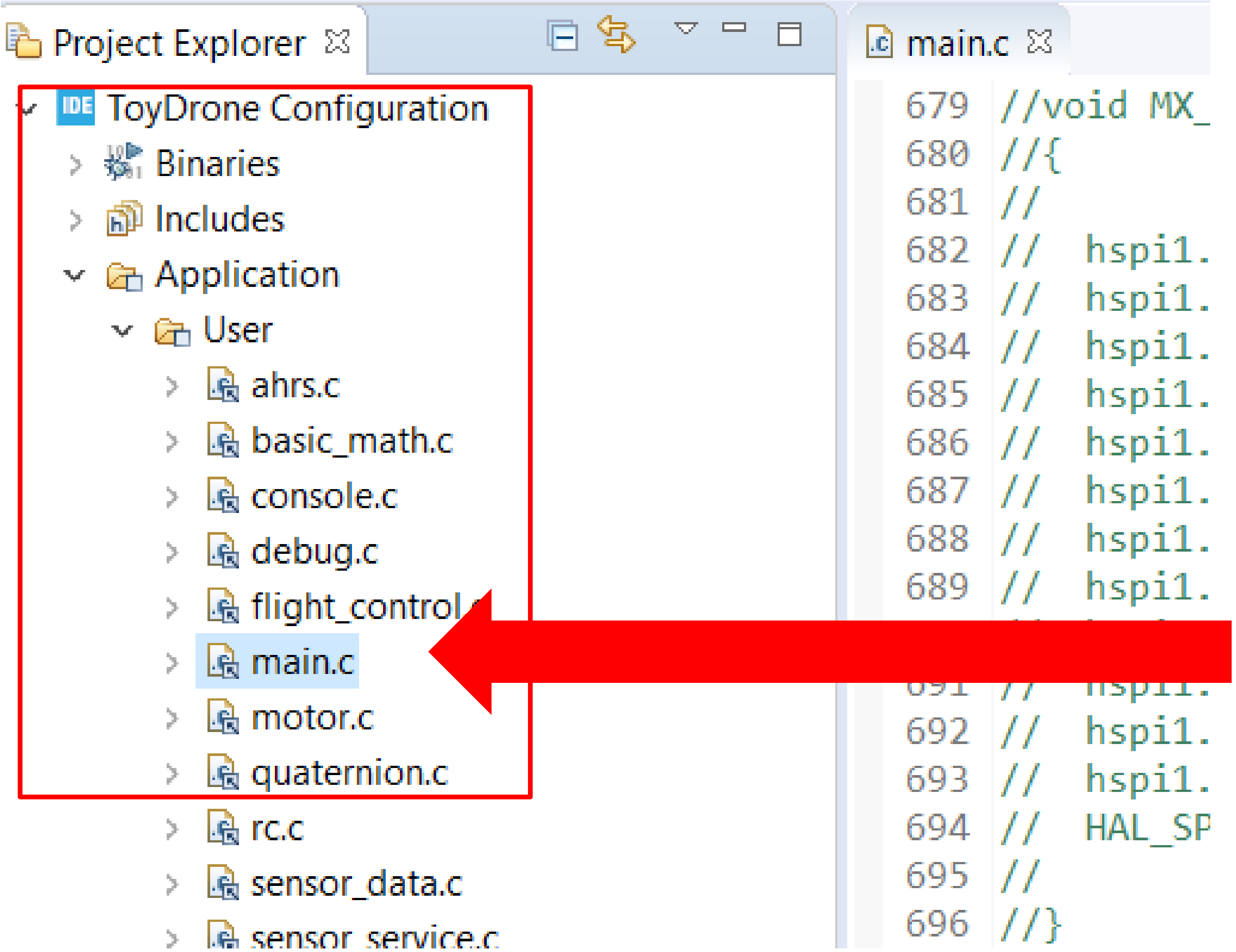
□ Hide projects that already exist in the workspace

Working sets

□ Add project to working sets

Working sets:

STM32CubeIDE: Project Files



```
//void MX_
//{
//
//   hspi1.
//   hspi1.
//   hspi1.
//   hspi1.
//   hspi1.
//   hspi1.
//   hspi1.
//   hspi1.
//   hspi1.
//   hspi1.
//   hspi1.
//   HAL_SP
//
//}
```

main.c is the application file of your project. You will work here

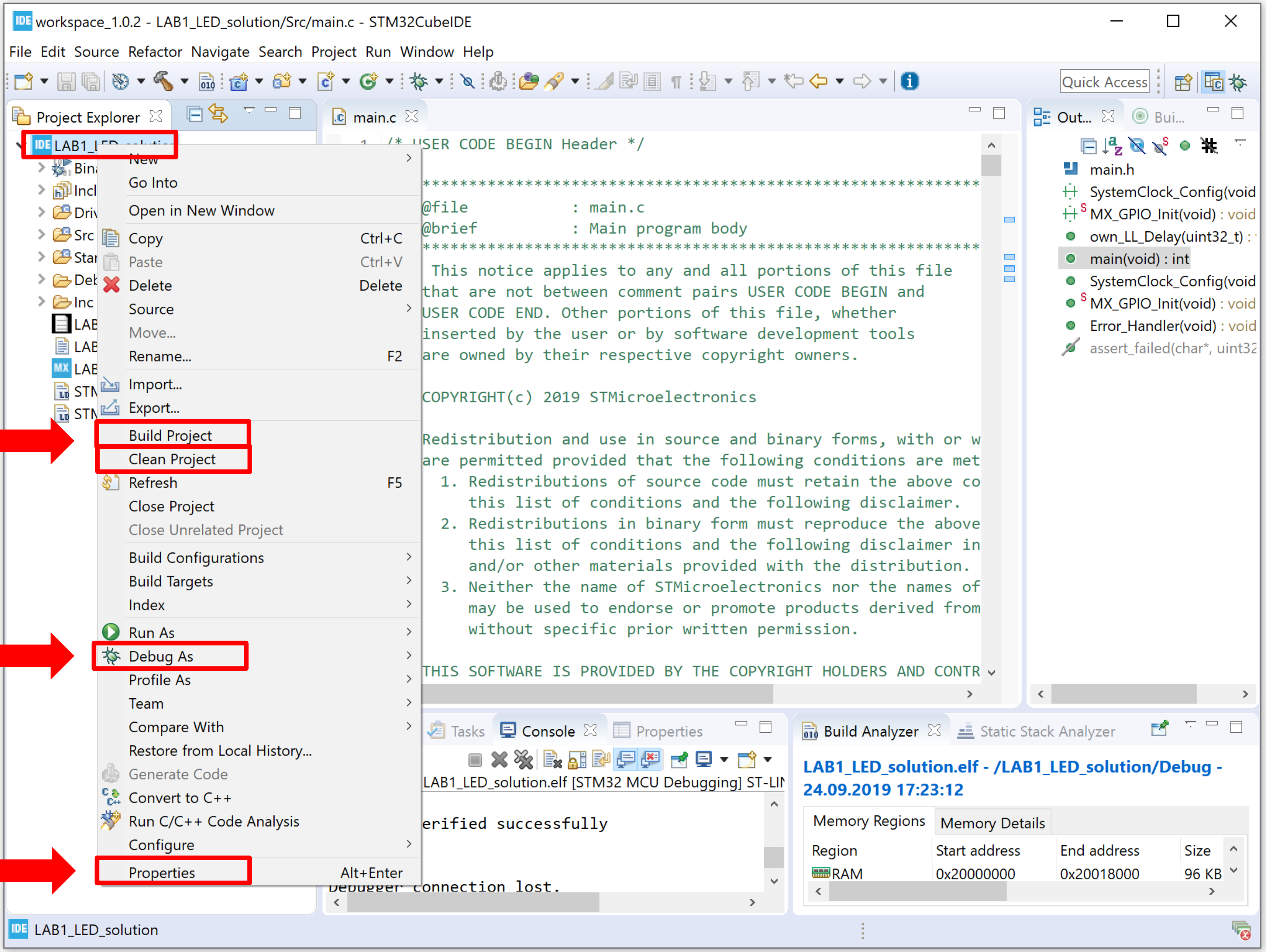
STM32CubeIDE: Key Functions

1

BUILD PROJECT

DEBUG

PROJECT PROPERTIES (ADVANCED)



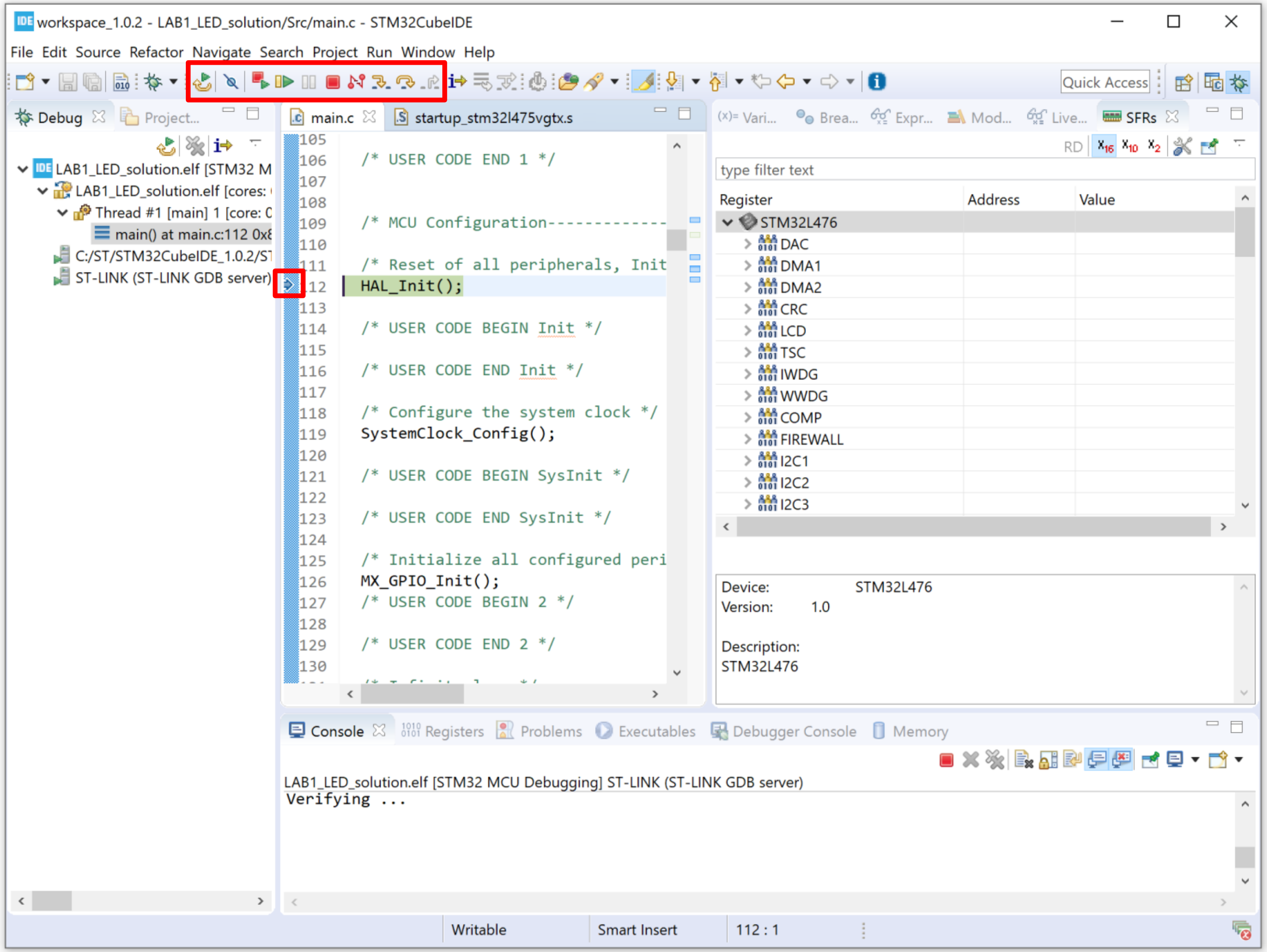
```
/* USER CODE BEGIN Header */
******************************************************************
@file           : main.c
@brief          : Main program body
******************************************************************
 This notice applies to any and all portions of this file
that are not between comment pairs USER CODE BEGIN and
USER CODE END. Other portions of this file, whether
inserted by the user or by software development tools
are owned by their respective copyright owners.

COPYRIGHT(c) 2019 STMicroelectronics

Redistribution and use in source and binary forms, with or w
are permitted provided that the following conditions are met
  1. Redistributions of source code must retain the above co
     this list of conditions and the following disclaimer.
  2. Redistributions in binary form must reproduce the above
     this list of conditions and the following disclaimer in
     and/or other materials provided with the distribution.
  3. Neither the name of STMicroelectronics nor the names of
     may be used to endorse or promote products derived from
     without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTR
```

```
LAB1_LED_solution.elf [STM32 MCU Debugging] ST-LIN

erified successfully

Debugger connection lost.
```

LAB1_LED_solution.elf - /LAB1_LED_solution/Debug - 24.09.2019 17:23:12

Region	Start address	End address	Size
RAM	0x20000000	0x20018000	96 KB

STM32CubeIDE: Debug



- LAB1_LED_solution.elf [STM32 M
  - LAB1_LED_solution.elf [cores:
    - Thread #1 [main] 1 [core: 0
      - main() at main.c:112 0x8
  - C:/ST/STM32CubeIDE_1.0.2/ST
  - ST-LINK (ST-LINK GDB server)

```
  /* USER CODE END 1 */


  /* MCU Configuration--------------

  /* Reset of all peripherals, Init
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peri
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */
```

type filter text

Register	Address	Value
STM32L476		
DAC		
DMA1		
DMA2		
CRC		
LCD		
TSC		
IWDG		
WWDG		
COMP		
FIREWALL		
I2C1		
I2C2		
I2C3		

Device: STM32L476
Version: 1.0

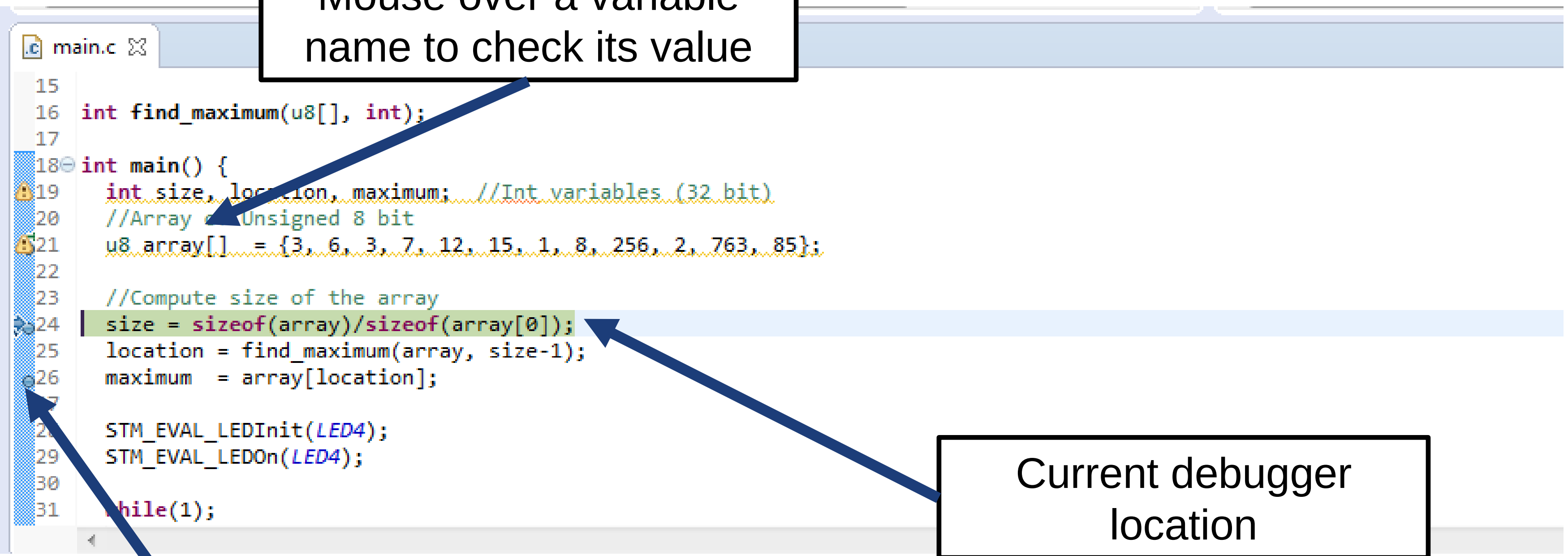
Description:
STM32L476

LAB1_LED_solution.elf [STM32 MCU Debugging] ST-LINK (ST-LINK GDB server)

```
Verifying ...
```

Writable | Smart Insert | 112 : 1

STM32CubeIDE: Debug



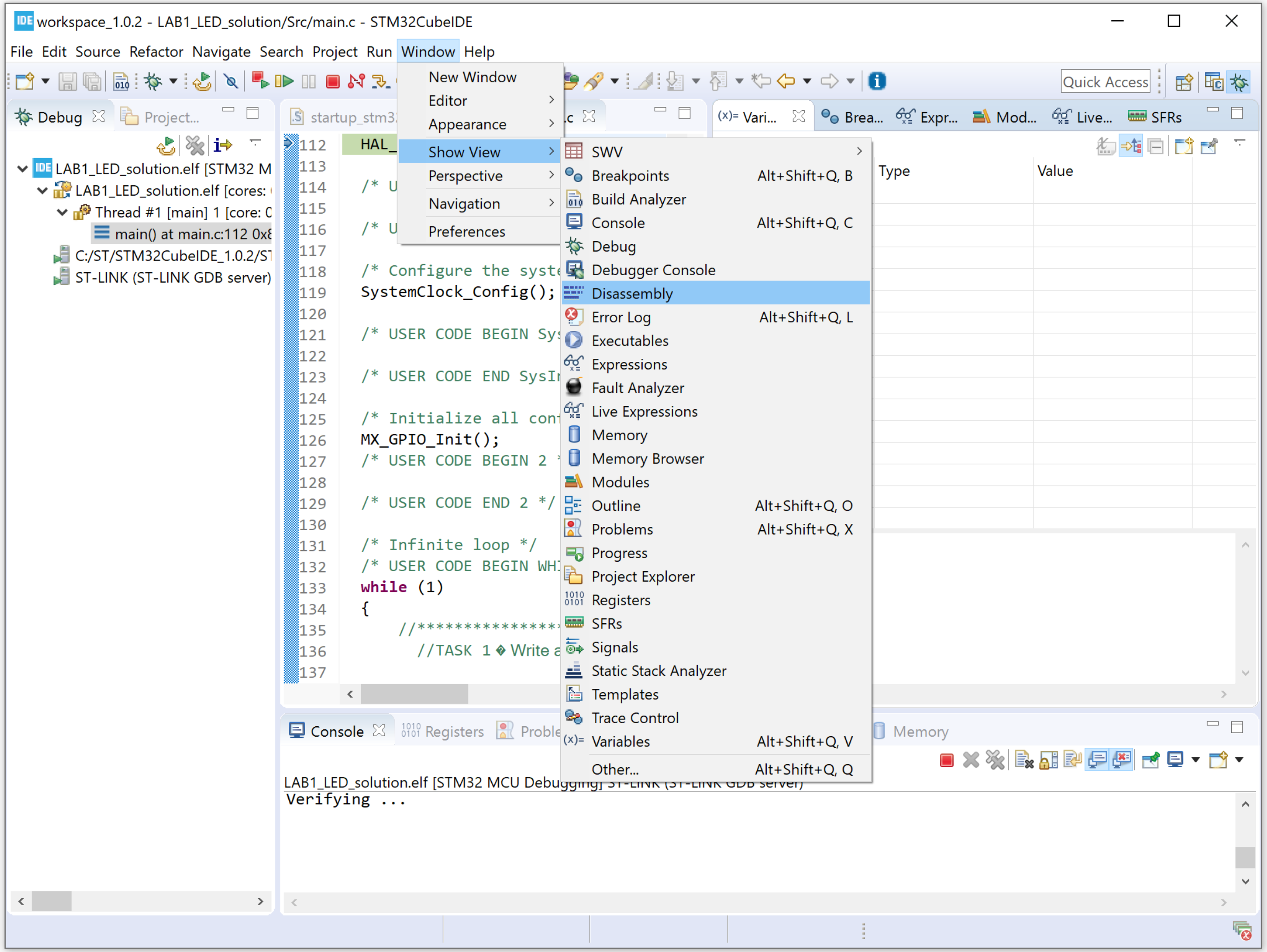
```c

int find_maximum(u8[], int);

int main() {
  int size, location, maximum;  //Int variables (32 bit)
  //Array of Unsigned 8 bit
  u8 array[]  = {3, 6, 3, 7, 12, 15, 1, 8, 256, 2, 763, 85};

  //Compute size of the array
  size = sizeof(array)/sizeof(array[0]);
  location = find_maximum(array, size-1);
  maximum  = array[location];

  STM_EVAL_LEDInit(LED4);
  STM_EVAL_LEDOn(LED4);

  while(1);
```

Mouse over a variable name to check its value

Current debugger location

Breakpoints
(code execution pauses here)

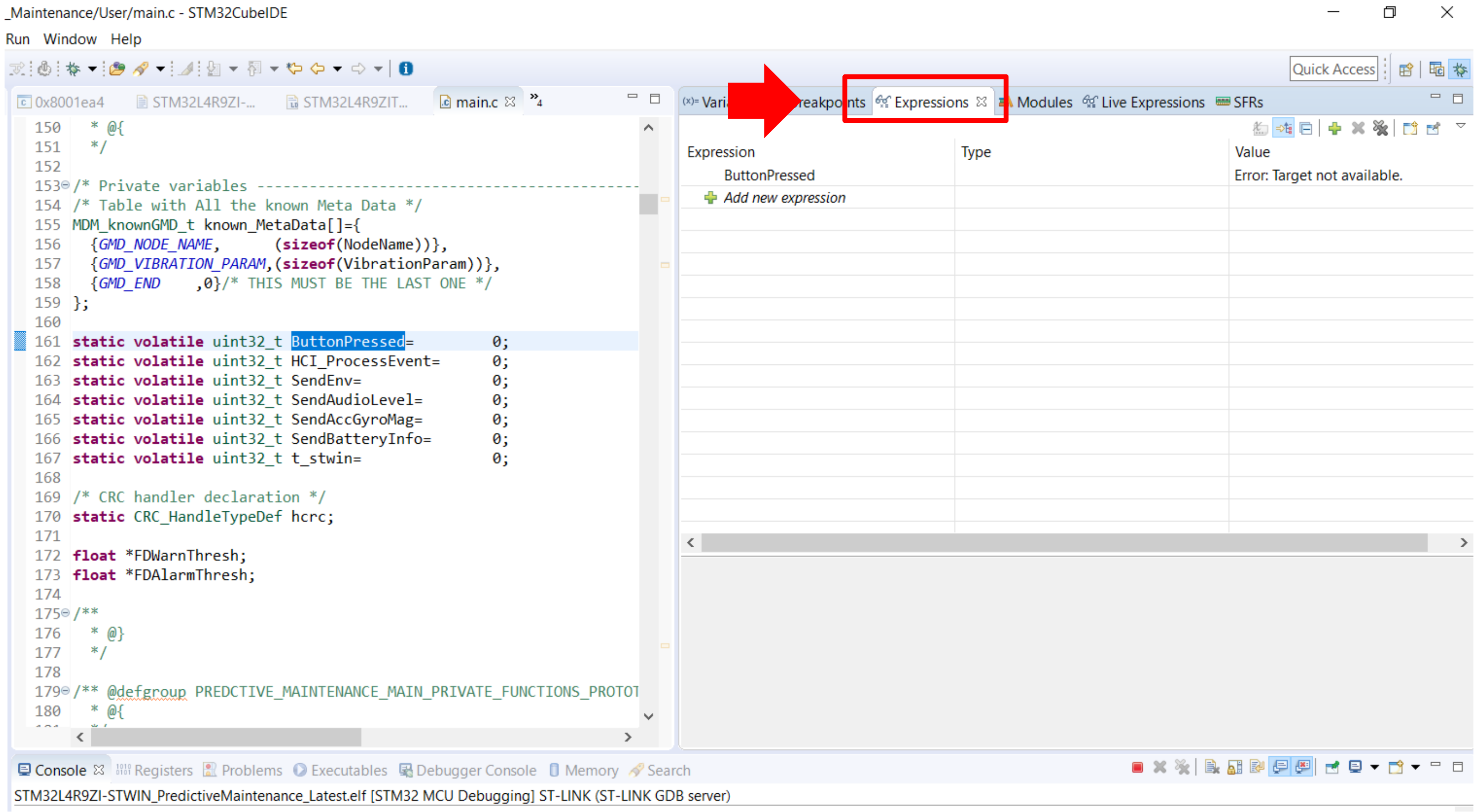
STM32CubeIDE



```
HAL_

/* U

/* U

/* Configure the syste
SystemClock_Config();

/* USER CODE BEGIN Sys

/* USER CODE END SysI

/* Initialize all conf
MX_GPIO_Init();
/* USER CODE BEGIN 2

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHI
while (1)
{
    //***************
      //TASK 1 ◆ Write a
```

```
LAB1_LED_solution.elf [STM32 MCU Debugging] ST-LINK (ST-LINK GDB server)
Verifying ...
```

STM32CubeIDE – Variables Monitoring

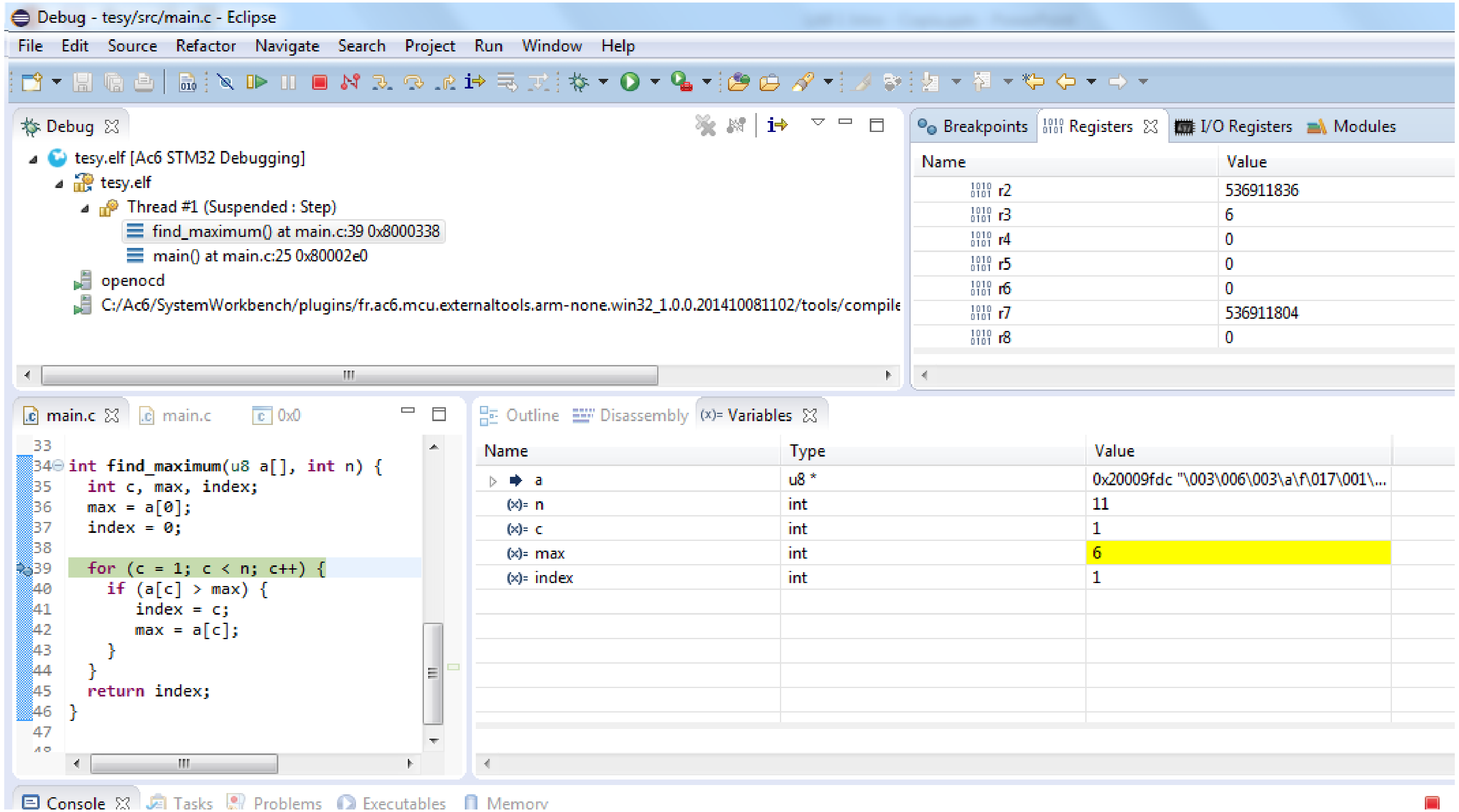


```
  * @{
  */

/* Private variables ---------------------------------------------------------
/* Table with All the known Meta Data */
MDM_knownGMD_t known_MetaData[]={
  {GMD_NODE_NAME,       (sizeof(NodeName))},
  {GMD_VIBRATION_PARAM,(sizeof(VibrationParam))},
  {GMD_END    ,0}/* THIS MUST BE THE LAST ONE */
};

static volatile uint32_t ButtonPressed=         0;
static volatile uint32_t HCI_ProcessEvent=      0;
static volatile uint32_t SendEnv=               0;
static volatile uint32_t SendAudioLevel=        0;
static volatile uint32_t SendAccGyroMag=        0;
static volatile uint32_t SendBatteryInfo=       0;
static volatile uint32_t t_stwin=               0;

/* CRC handler declaration */
static CRC_HandleTypeDef hcrc;

float *FDWarnThresh;
float *FDAlarmThresh;

/**
  * @}
  */

/** @defgroup PREDCTIVE_MAINTENANCE_MAIN_PRIVATE_FUNCTIONS_PROTOT
  * @{
```

Expression	Type	Value
ButtonPressed		Error: Target not available.
Add new expression		

STM32L4R9ZI-STWIN_PredictiveMaintenance_Latest.elf [STM32 MCU Debugging] ST-LINK (ST-LINK GDB server)

STM32CubeIDE – Variables Monitoring



```
tesy.elf [Ac6 STM32 Debugging]
  tesy.elf
    Thread #1 (Suspended : Step)
      find_maximum() at main.c:39 0x8000338
      main() at main.c:25 0x80002e0
  openocd
  C:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.externaltools.arm-none.win32_1.0.0.201410081102/tools/compile
```

Name	Value
r2	536911836
r3	6
r4	0
r5	0
r6	0
r7	536911804
r8	0

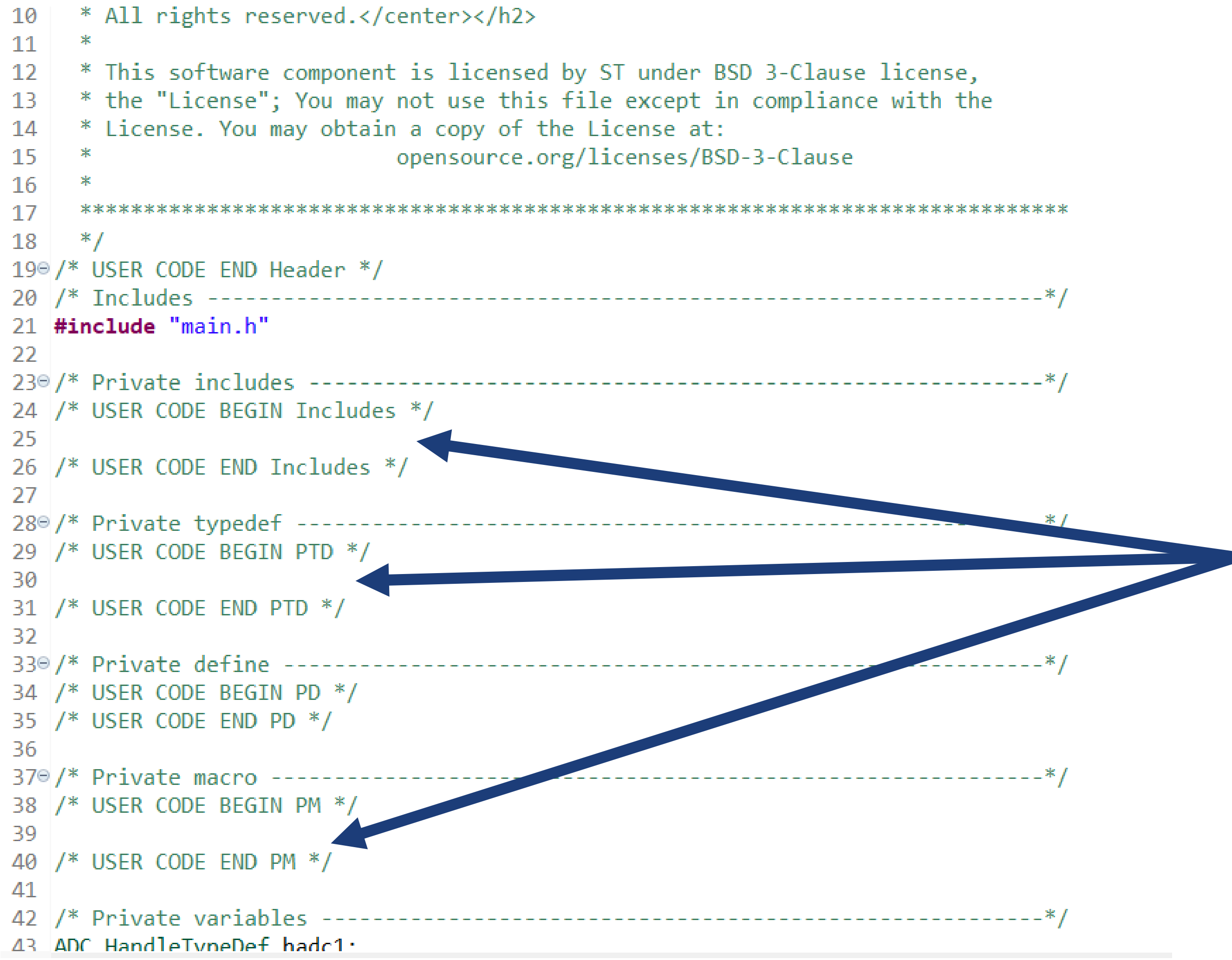
```c
int find_maximum(u8 a[], int n) {
  int c, max, index;
  max = a[0];
  index = 0;

  for (c = 1; c < n; c++) {
    if (a[c] > max) {
       index = c;
       max = a[c];
    }
  }
  return index;
}
```

Name	Type	Value
a	u8 *	0x20009fdc "\003\006\003\a\f\017\001\...
n	int	11
c	int	1
max	int	6
index	int	1

LAB1: Exercise overview

Exercise	Assignment	Concept
Exercise 1	- Hands-on **STEVAL-FCU001V1** evaluation board - Open the project with CubeMX - Understanding software architecture by STM - Control LEDs with GPIO	Start program, Debug, GPIO
Questions:	- What is a debugger? - In which page of the STWIN schematic is the MCU? - Explain the programming pipeline of the STM32 MCU - If a variable is defined as static int, what does it mean?	Programming and debugging

STM32CubeIDE – User Code

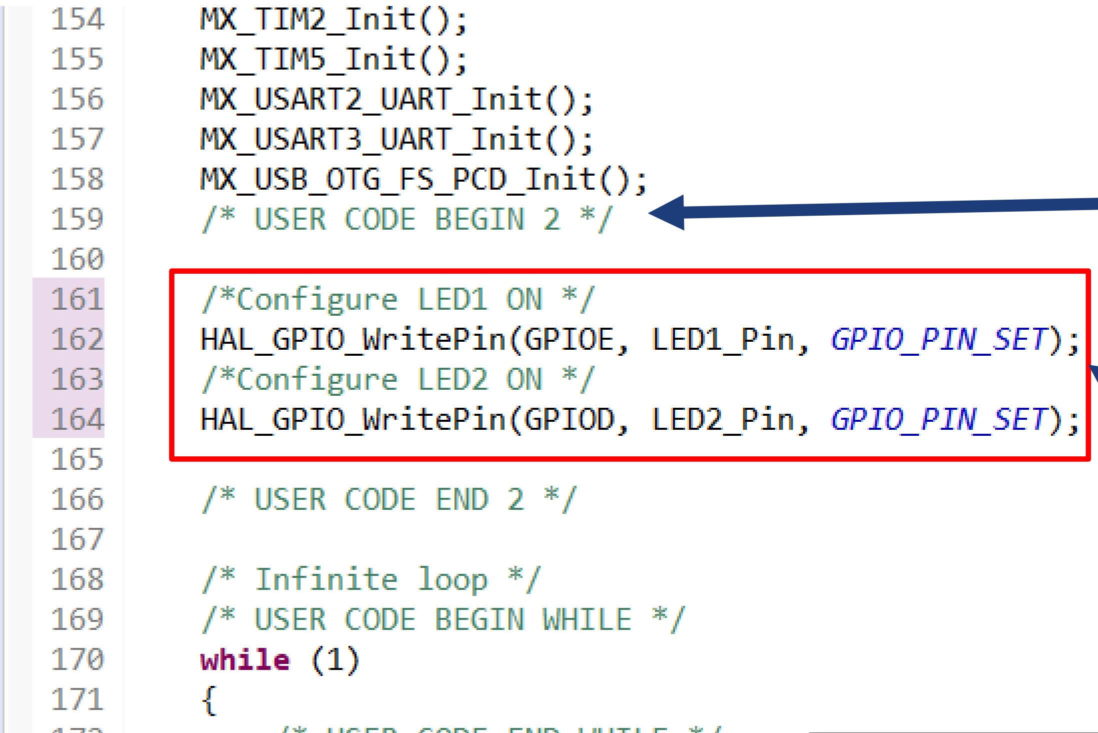


```
 * All rights reserved.</center></h2>
 *
 * This software component is licensed by ST under BSD 3-Clause license,
 * the "License"; You may not use this file except in compliance with the
 * License. You may obtain a copy of the License at:
 *                        opensource.org/licenses/BSD-3-Clause
 *
 ******************************************************************************
 */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;
```

STMCubeMX automatically generates source files but,
it deletes all your code placed outside the USER CODE space

Write the code within USER CODE BEGIN and USER CODE END

STM32CubeIDE – LEDs GO!



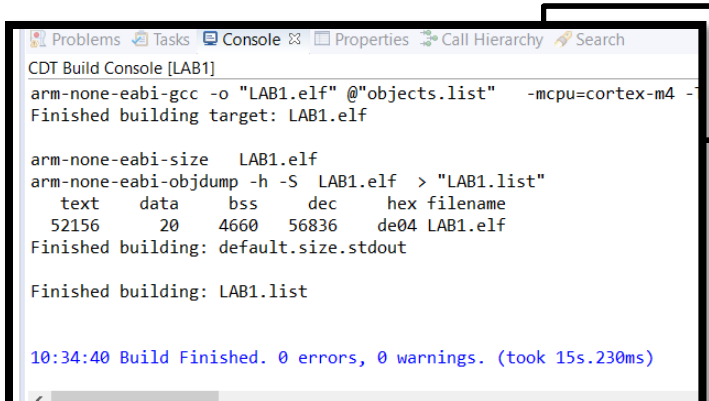
```
    MX_TIM2_Init();
    MX_TIM5_Init();
    MX_USART2_UART_Init();
    MX_USART3_UART_Init();
    MX_USB_OTG_FS_PCD_Init();
    /* USER CODE BEGIN 2 */

    /*Configure LED1 ON */
    HAL_GPIO_WritePin(GPIOE, LED1_Pin, GPIO_PIN_SET);
    /*Configure LED2 ON */
    HAL_GPIO_WritePin(GPIOD, LED2_Pin, GPIO_PIN_SET);

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        /* USER CODE END WHILE */

        /* USER CODE BEGIN 3 */
    }
    /* USER CODE END 3 */
}
```

Enable
STWIN Green (LED1) and Orange (LED2) LEDs

Write the code within USER CODE BEGIN and USER CODE END

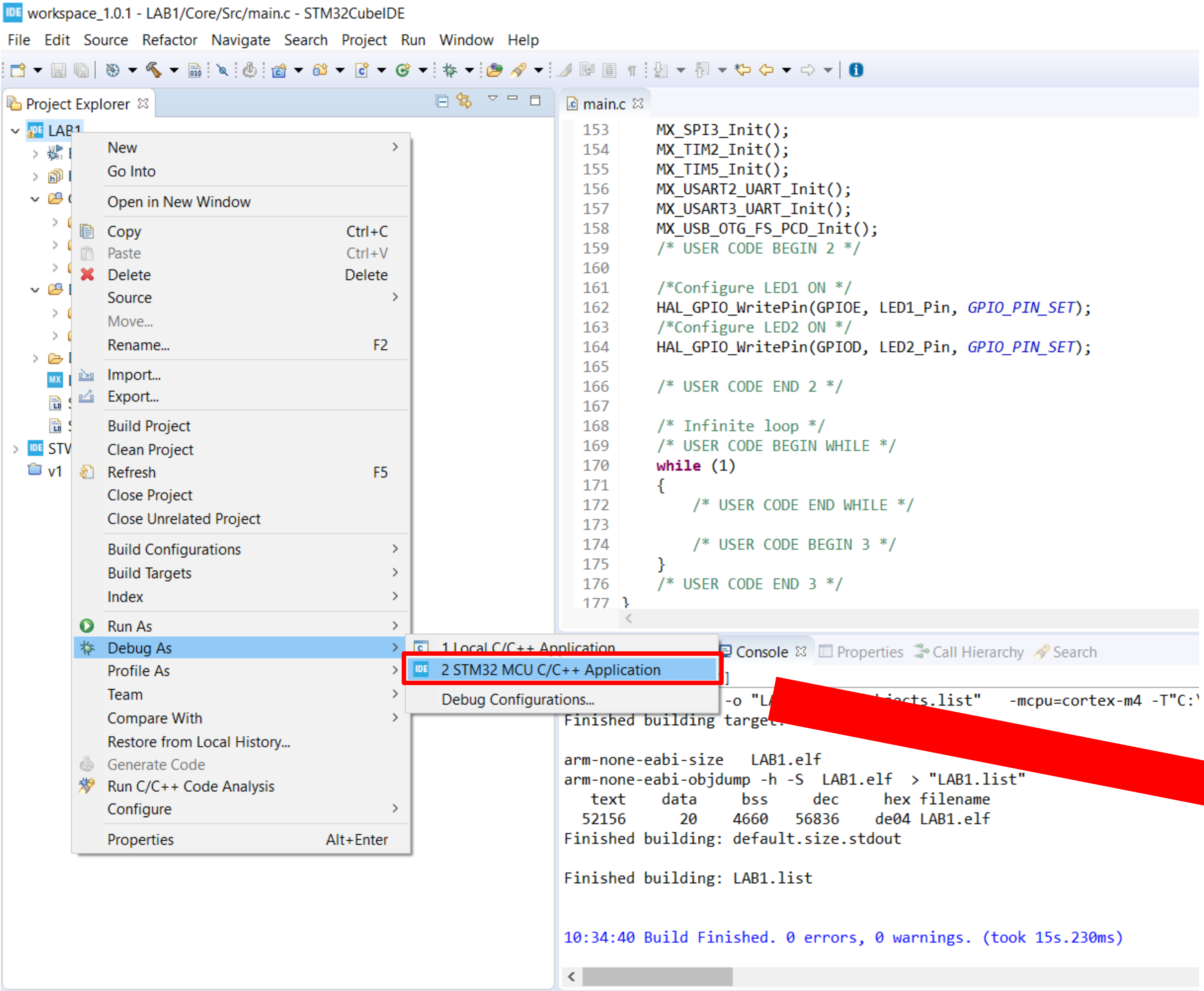
Add the GPIO_WritePin functions, they configure the GPIO pin connected to the corresponding LED to 1, a logical value of 3.3V

BUILD-> DEBUG -> TEST!



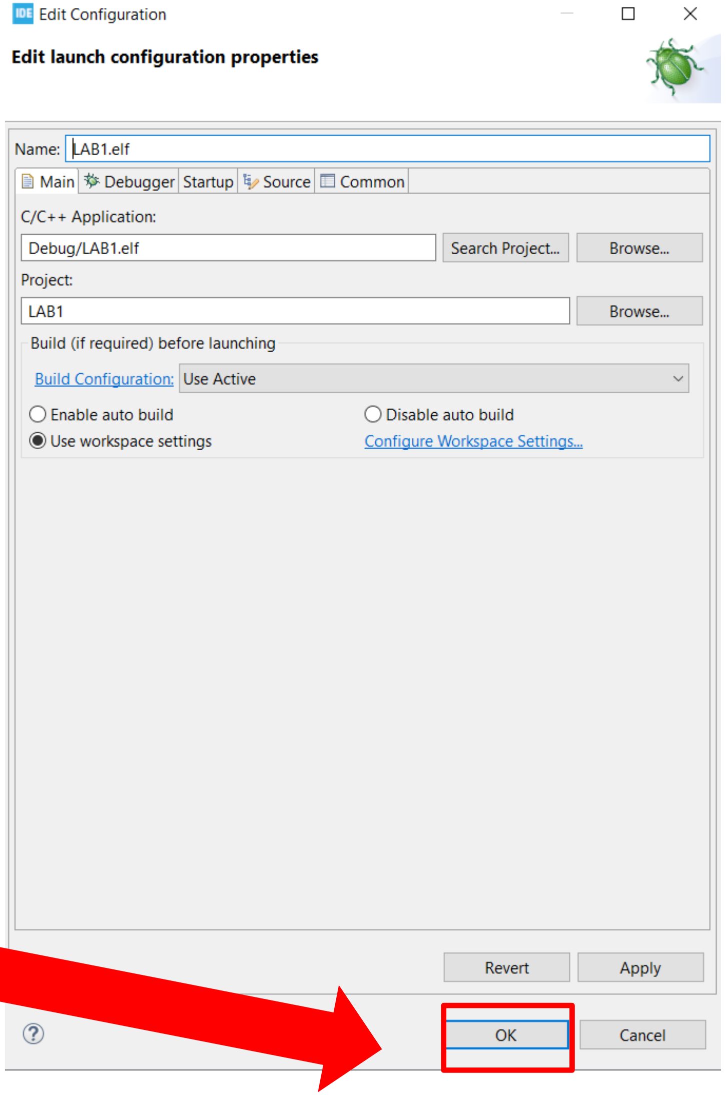
```
CDT Build Console [LAB1]
arm-none-eabi-gcc -o "LAB1.elf" @"objects.list"   -mcpu=cortex-m4 -
Finished building target: LAB1.elf

arm-none-eabi-size   LAB1.elf
arm-none-eabi-objdump -h -S  LAB1.elf  > "LAB1.list"
   text    data     bss     dec     hex filename
  52156      20    4660   56836    de04 LAB1.elf
Finished building: default.size.stdout

Finished building: LAB1.list


10:34:40 Build Finished. 0 errors, 0 warnings. (took 15s.230ms)
```

STM32CubeIDE – LEDs GO!



```c
  MX_SPI3_Init();
  MX_TIM2_Init();
  MX_TIM5_Init();
  MX_USART2_UART_Init();
  MX_USART3_UART_Init();
  MX_USB_OTG_FS_PCD_Init();
  /* USER CODE BEGIN 2 */

  /*Configure LED1 ON */
  HAL_GPIO_WritePin(GPIOE, LED1_Pin, GPIO_PIN_SET);
  /*Configure LED2 ON */
  HAL_GPIO_WritePin(GPIOD, LED2_Pin, GPIO_PIN_SET);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

```
]
-o "LA          jects.list"   -mcpu=cortex-m4 -T"C:\
Finished building target

arm-none-eabi-size   LAB1.elf
arm-none-eabi-objdump -h -S  LAB1.elf  > "LAB1.list"
   text    data     bss     dec     hex filename
  52156      20    4660   56836    de04 LAB1.elf
Finished building: default.size.stdout

Finished building: LAB1.list


10:34:40 Build Finished. 0 errors, 0 warnings. (took 15s.230ms)
```



Name: LAB1.elf

C/C++ Application: Debug/LAB1.elf

Project: LAB1

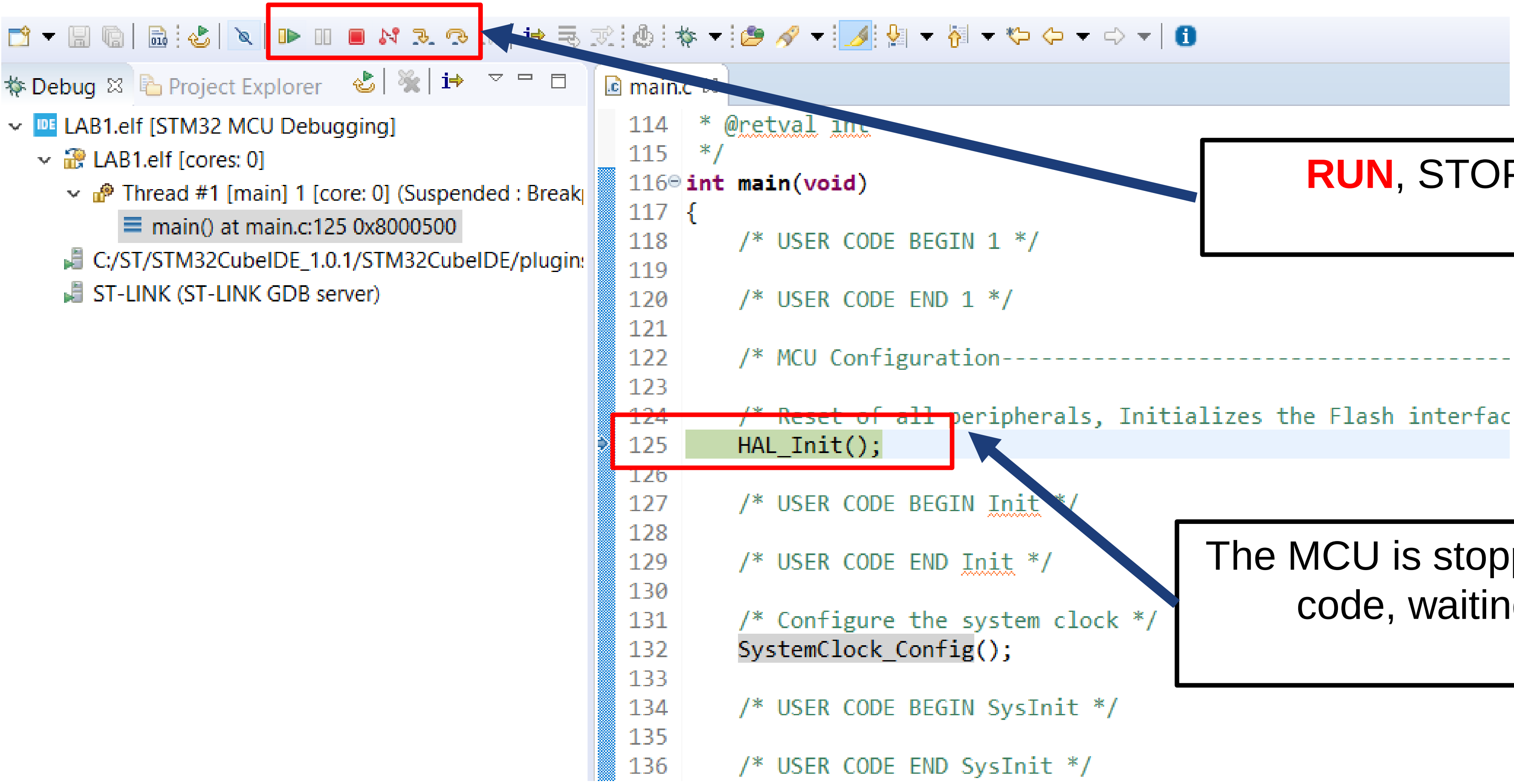
Build (if required) before launching

Build Configuration: Use Active

○ Enable auto build ○ Disable auto build

● Use workspace settings Configure Workspace Settings...

STM32CubeIDE – LEDs GO! DEBUGGING



```
LAB1.elf [STM32 MCU Debugging]
  LAB1.elf [cores: 0]
    Thread #1 [main] 1 [core: 0] (Suspended : Break
      main() at main.c:125 0x8000500
  C:/ST/STM32CubeIDE_1.0.1/STM32CubeIDE/plugin
  ST-LINK (ST-LINK GDB server)
```

```c
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration----------------------------------------

  /* Reset of all peripherals, Initializes the Flash interfac
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */
```

RUN, STOP, OR STEP BY STEP

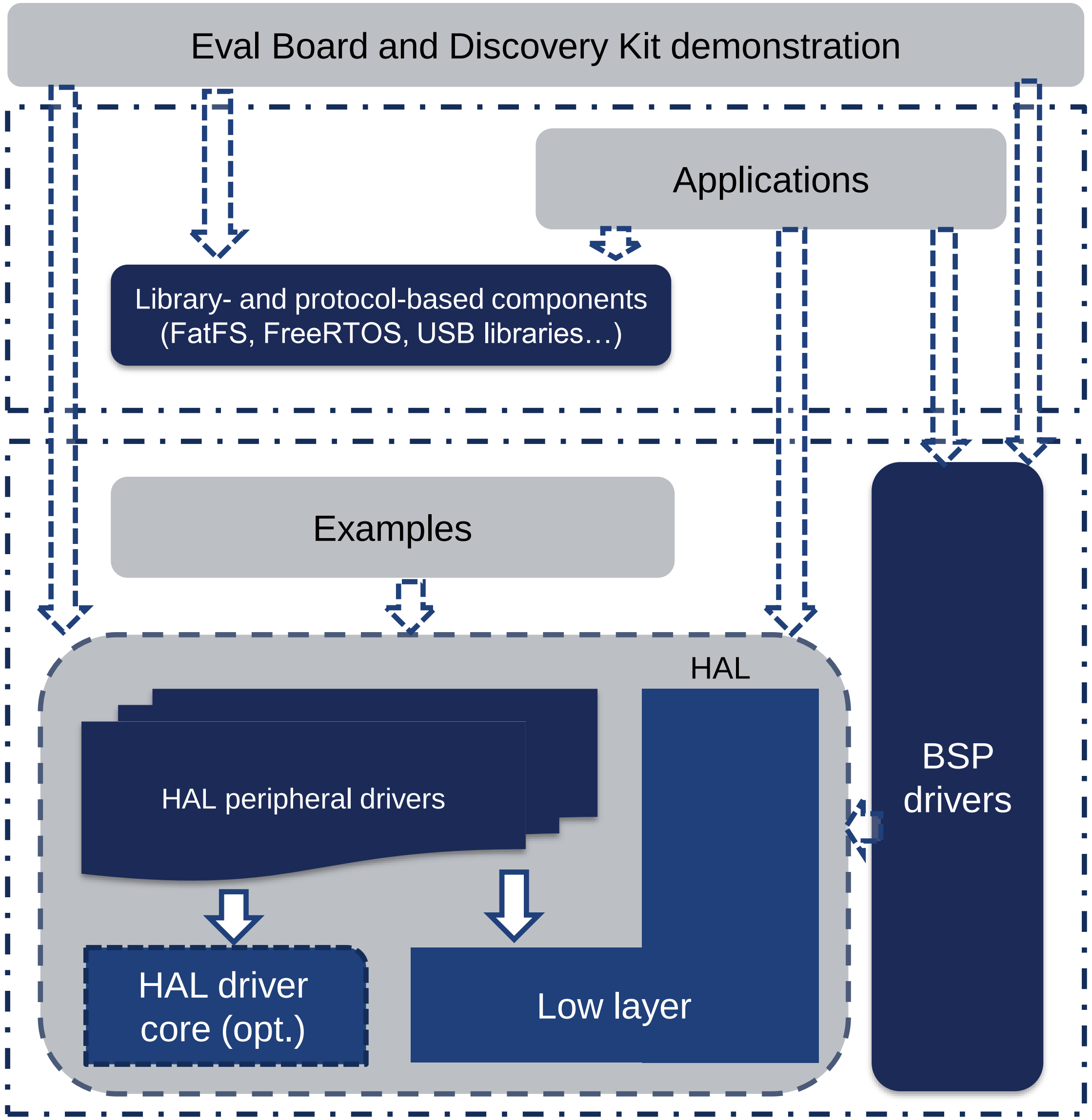
The MCU is stopped at the first line of the code, waiting for a user request

STM32 software architecture

Application Level 2

Middleware Level 1

Drivers HAL or LL Level 0


Eval Board and Discovery Kit demonstration
Applications
Library- and protocol-based components
(FatFS, FreeRTOS, USB libraries…)
Examples
HAL
HAL peripheral drivers
HAL driver
core (opt.)
Low layer
BSP
drivers

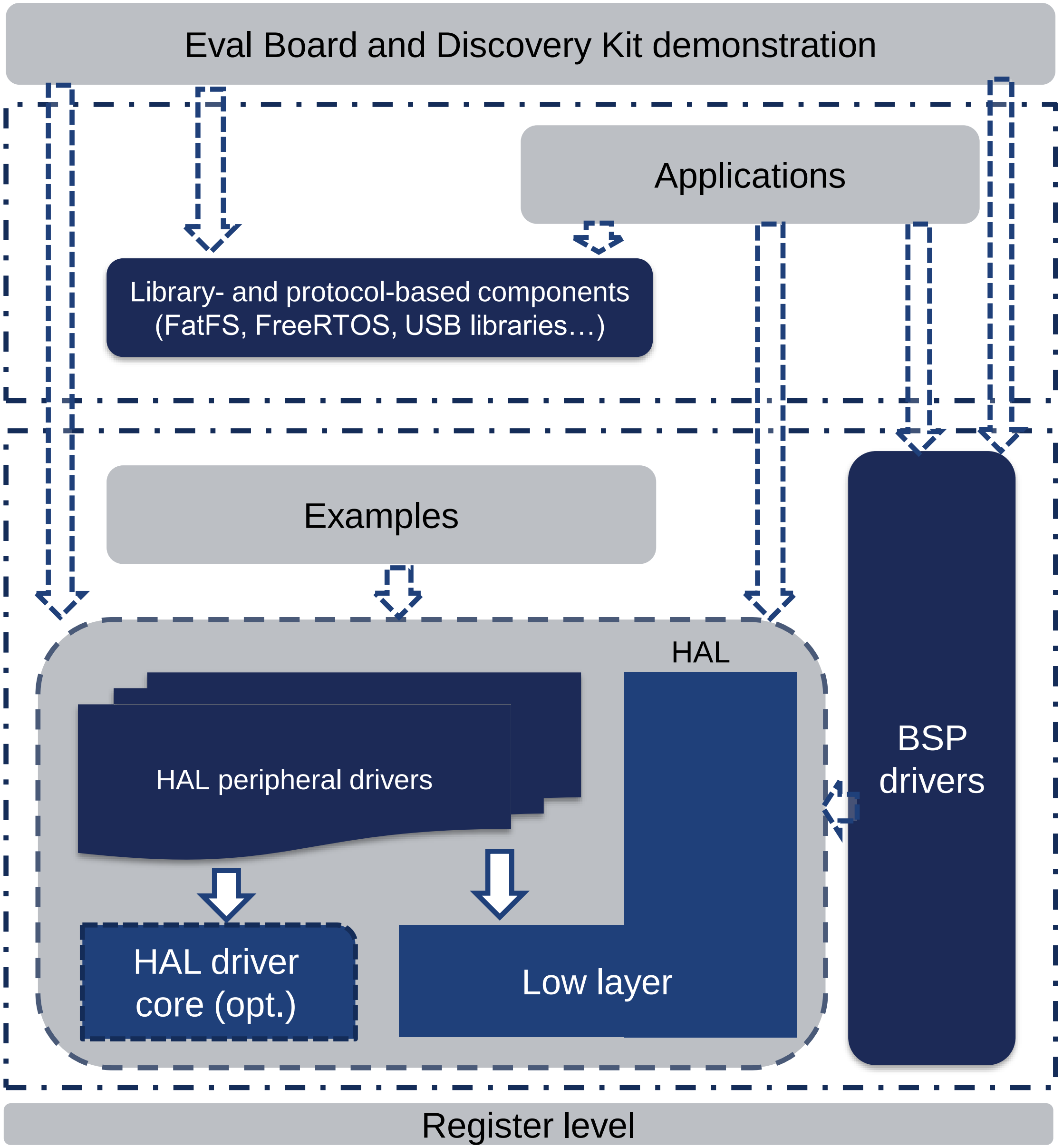
STM32 software architecture

Application Level 2

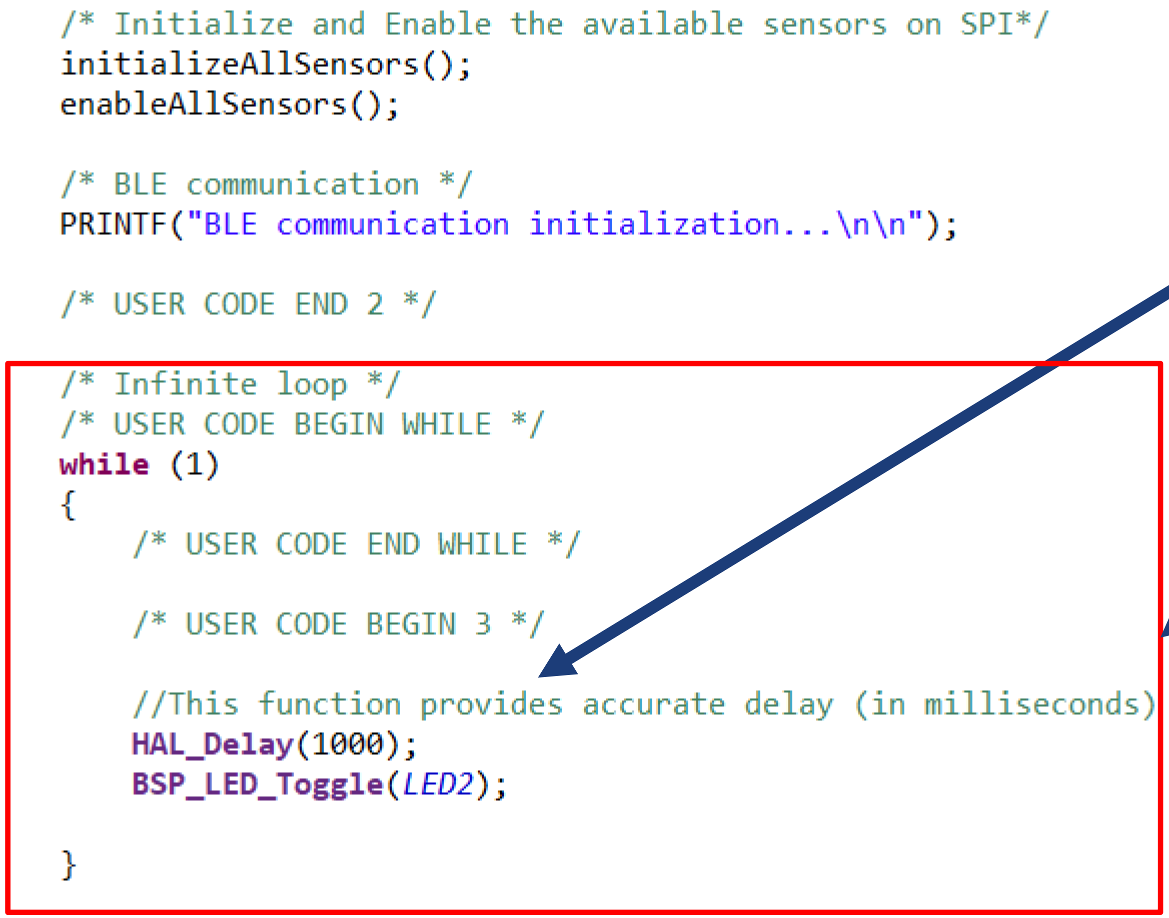
Middleware Level 1

Drivers HAL or LL Level 0

Advanced


Eval Board and Discovery Kit demonstration
Applications
Library- and protocol-based components (FatFS, FreeRTOS, USB libraries…)
Examples
HAL
HAL peripheral drivers
HAL driver core (opt.)
Low layer
BSP drivers
Register level

STM32CubeIDE – LEDs GO!



```
/* Initialize and Enable the available sensors on SPI*/
initializeAllSensors();
enableAllSensors();

/* BLE communication */
PRINTF("BLE communication initialization...\n\n");

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

    //This function provides accurate delay (in milliseconds)
    HAL_Delay(1000);
    BSP_LED_Toggle(LED2);

}
```

Write the code within USER CODE BEGIN and USER CODE END

Add the GPIO_Toggle functions for LED1, change the BLINK frequency using HAL_Delay

BUILD-> DEBUG -> TEST!