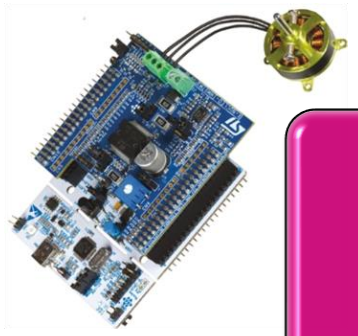
ST Motor Control FOC library API & STM32CubeMX

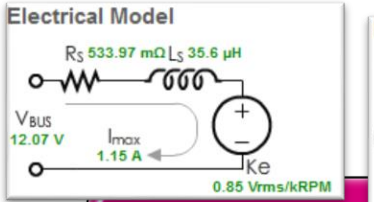
Lab 5: Using Motor Control library API’s

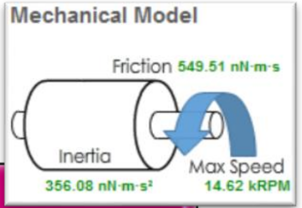
life.augmented

Motor Control Development Workflow

#6 – Application development







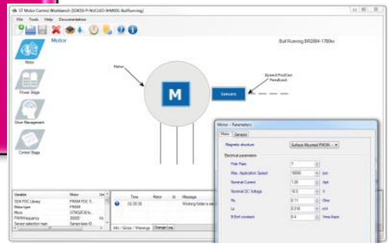
Hardware Setup

Motor Characterization

Motor Profiler
Motion Control Suite

System Configuration

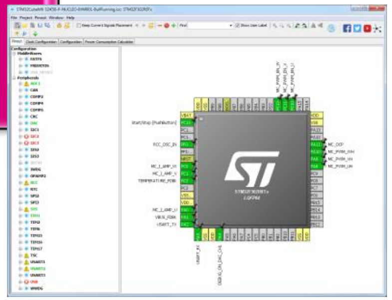
Motor Control Workbench



Project Configuration

CubeMX & IDE

STM32 CubeMX



ac6

TrueSTUDIO for STM32

IAR SYSTEMS

KEIL Tools by ARM

Motor Drive Tuning

Tune MC part

Final Application Development

life.augmented

LAB 5 tasks

- SDK5.x Firmware Architecture
- API - Application Programming Interface
- How to build an user project & firmware?
- Start / Stop – motor by API

  Hands-On section
- Speed control by API

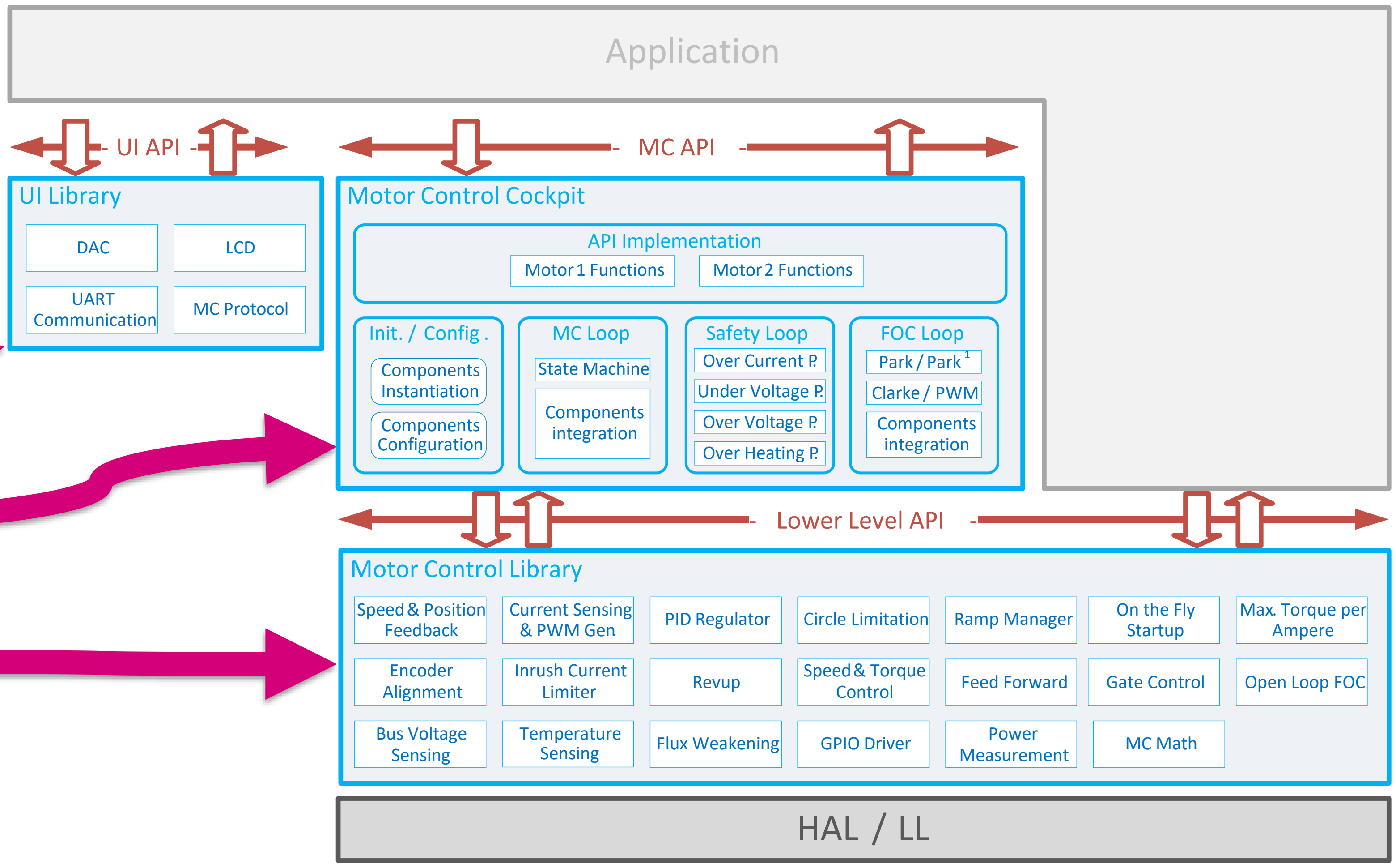
  Hands-On section

life.augmented

SDK5.x Firmware Architecture

SDK5.x FW Architecture Overview

- Motor control firmware is organized into 3 parts:



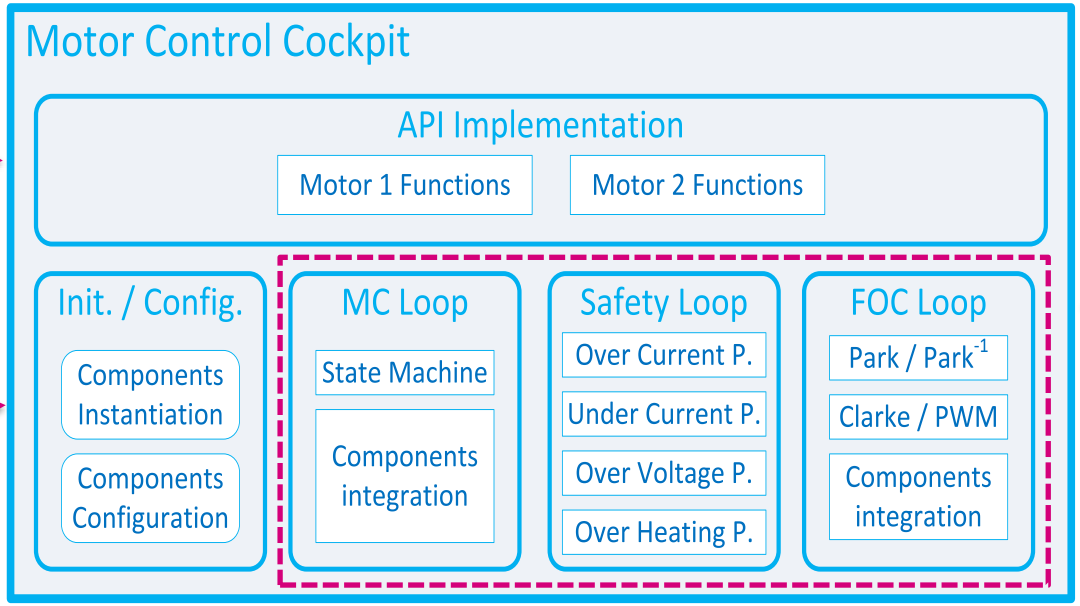
- User Interface Library
- Motor Control Cockpit
- Motor Control Library

Motor Control Cockpit - Introduction

- MC Cockpit is made of three parts

MC Interface
Implementation of the MC API

MC Configuration
Instantiation and configuration of all needed component



MC Dynamics
Implementation the Motor Control dynamic behavior/loops:
- FOC Loop (High Freq)
- MC Loop (Med Freq)
- Safety Loop (Safety tasks)

life.augmented

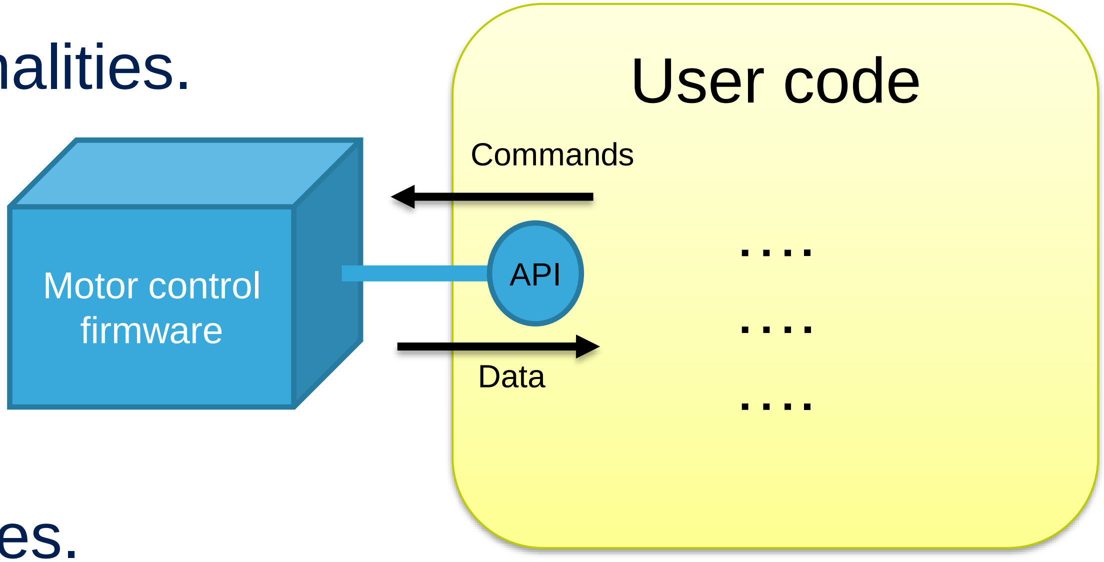
Application Programming Interface API

life.augmented

API - Application Programming Interface

What is an API?

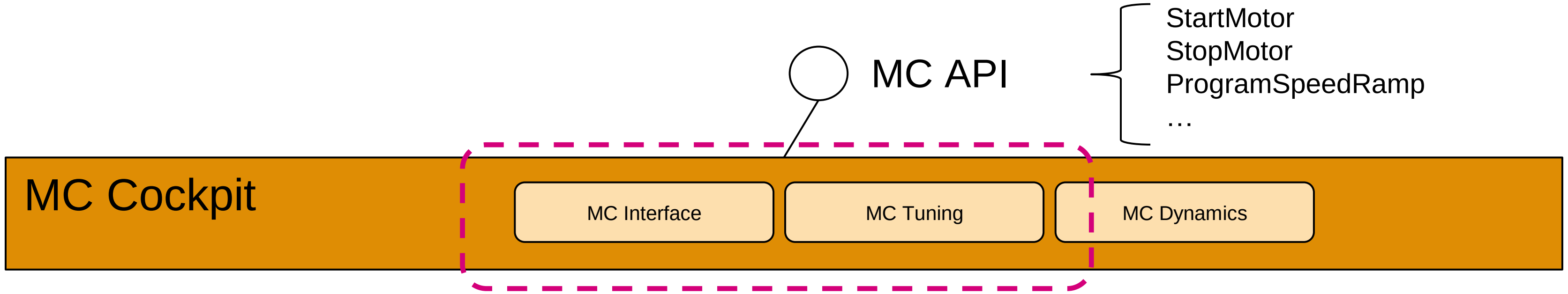
- An Application Programming Interface (API) specifies how software components should interact with each other.
- It provide a consistent, programmatic method for accessing a resource.
- It is a structured way of exposing functionalities.
- Unlike an user interface the API is a macchine to macchine interface. It allows developers to access the functionality of the software through well-defined data structures.



life.augmented

Motor Control (MC) API

- The MC API is the entry point to build user application
- It is split in two sections:
  - ***MC Interface*** is a set of basic functions that allows to build an user application.
  - ***MC Tuning*** contains full set of functions that can be used to interact with the motor control objects.



	Functionality	Intended use	Example functions
MC Interface	Basic	Basic user code	MC_StartMotor1 MC_StopMotor2 MC_ProgramSpeedRampMotor1
MC Tuning	Full	Tuning Advanced user code	PID_SetKP

MC Interface

- MC Interface contains 2 types of commands
  - ***Buffered*** - don't become active as soon as it is called but it will be executed when the state machine reach the RUN state.
  - ***Not buffered*** - is executed instantaneously if the state macchine is in the proper state otherwise it is discarded.

	Behavior	Example functions
Buffered commands	Command is buffered and executed when the state macchine reach the RUN state.	MC_ProgramSpeedRampMotor1 MC_ProgramTorqueRampMotor2 MC_SetCurrentReferenceMotor1
Not buffered commands	Command is executed instantaneously if the state machine is in the proper state otherwise it is discarded.	MC_StartMotor2 MC_StopMotor1 MC_AcknowledgeFaultMotor1

life.augmented

MC Interface functions

- All functions of the MC interface can be called by their self explaining names without passing the pointer to a data structure.

User code

```
....
{
....
MC_ProgramSpeedRampMotor1(final speed, ramp duration);
MC_StartMotor1();
...
}
```

MC Tuning functions

- MC tuning commands has to has at least one input parameter
- First input parameter is a pointer to item from MC Tuning list
- The called function will use data linked by this pointer
- The MC Tuning list in “mc_config.h” has to be included

  - The MC Tuning list you can find in the header file “mc_config.h” or also in the documentation

M1 means motor 1

User code

```
…
#include "mc_config.h"
{
…
PI_SetKP(PIDSpeedHandle_M1, kpValue);
…
}
```

PIDSpeedHandle_M1 =
= pointer of MC Tuning list

SDK C - variables formats

- Speed variables formats
  - Two formats are utilized in the firmware library:
    - 0.1Hz, this is the format utilized by speed PID and by the user interface layer.

      For example what value we have to use for speed ramp: after 1s run on 600 rpm.

      600 rpm [round per minute] ➔ 600 / 60 [round per second] ➔ 10 [rps]

      10 [rps] ➔ 10 [Hz]

      10 [Hz] * 10 ➔ **100 [0.1Hz → … per ten seconds]**

      MC_ProgramSpeedRampMotor1(100, 1000); //1000 ms ➔ 1 s

Digit Per PWM (dpp), it expresses the angle variation (s16) in a PWM period.
This format can be directly accumulated for getting the rotor angular position

$$F_{dpp} = F_{0.1Hz} \frac{65536}{10 \cdot F_{PWM\,(Hz)}}$$

- Current / Torque units implementation

$$(Current * 2^{16} * R_{shunt} * Gain) / V_{dd\_micro} = Torque_{Final}$$

life.augmented

How to build an user project & firmware?

life.augmented

How to write down an user code?

- The best place for customer code is in the **Infinite loop** inside **main** function
- An **initialization** code of added periphery have to be placed before Infinite loop
  - To the section /* USER CODE BEGIN 2 */
  - Or after definition of /* USER CODE BEGIN WHILE */
- **Functional** code have to be located inside the **Infinite loop**
  - To the section /* USER CODE BEGIN WHILE */ after the row with while (1)
  - Or to the section /* USER CODE BEGIN 3 */

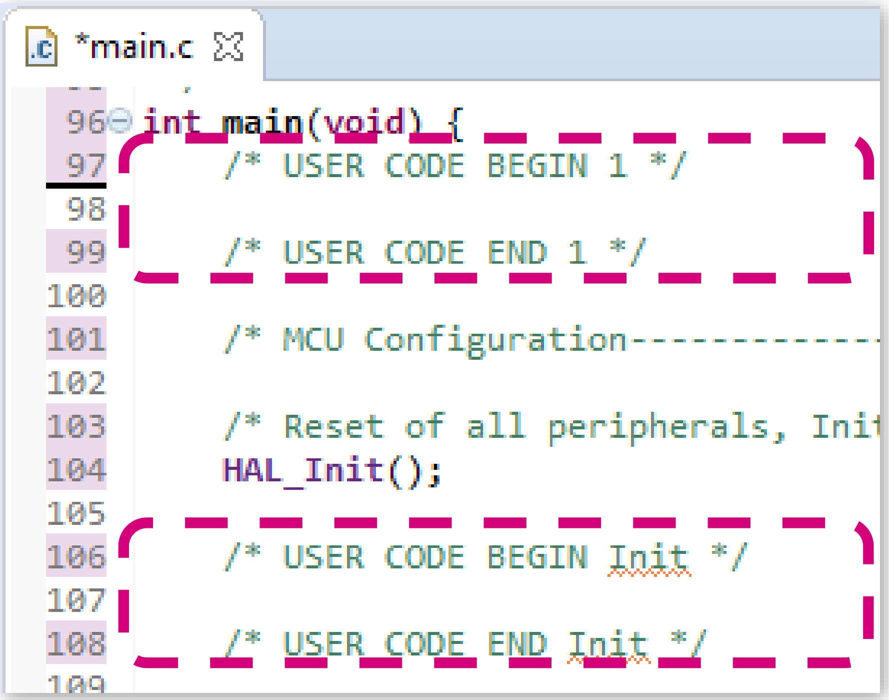
main.c

```
  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_DAC_Init();
  MX_TIM1_Init();
  MX_USART2_UART_Init();
  MX_MotorControl_Init();

  /* Initialize interrupts */
  MX_NVIC_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

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

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

  }
  /* USER CODE END 3 */

}
```

life.augmented

User Code Sections

- User Code sections have been placed where they were thought to be useful.
- At the beginning and end of such all user section are
  /* USER CODE BEGIN XXX */ and /* USER CODE END XXX */
- Applications developers can place the code they want in these sections.
- STM32CubeMx guarantees that this code is kept across regenerations.



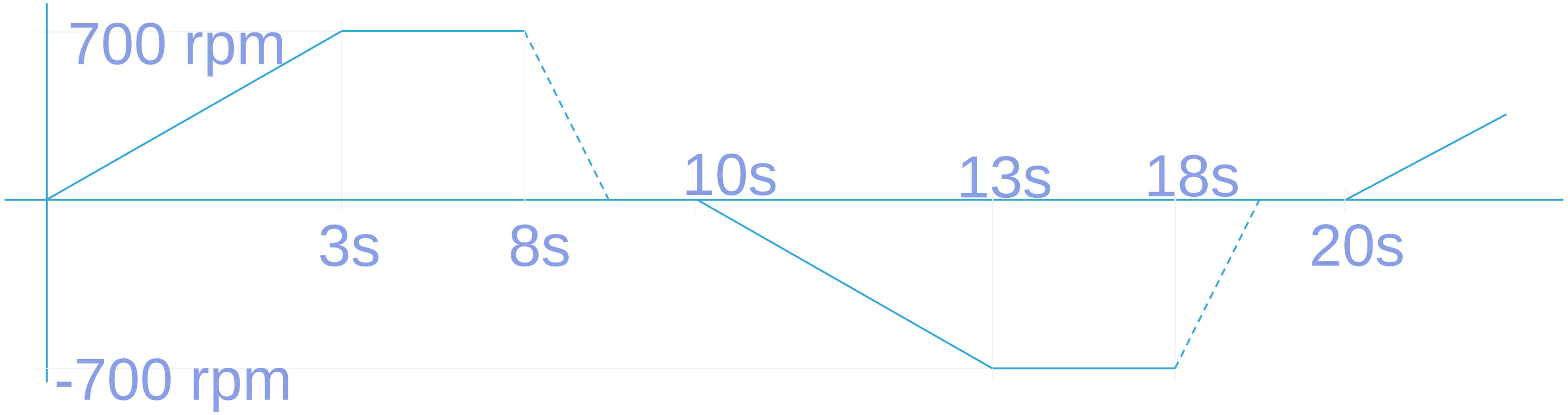
Hands-On section

Start / Stop – motor control by API

The tasks:
Use your project & MC Workbench setting for Nucleo F303RE & IHM16 & *GimBal* motor

Make sequence code with will do following steps:

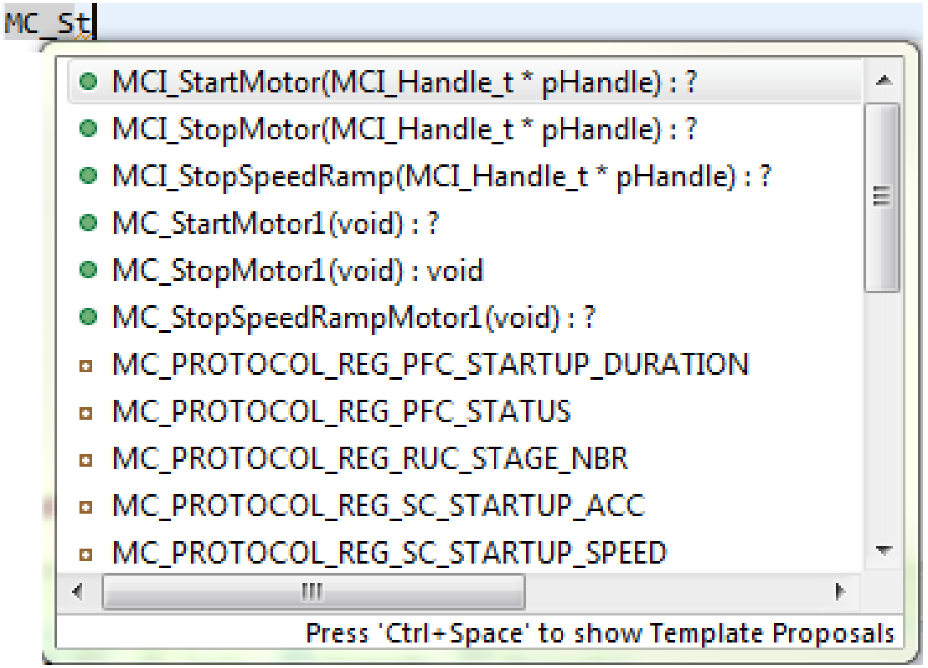
- **Start** the motor with start-up ramp up to 700 rpm during 3 seconds
- **Stay** on speed 700 rpm 5 second, then **stop** motor
- **Wait** 2 second, then **start** motor with previous start-up ramp, but to the **opposite direction!**
- **Stay** on speed 700 rpm 5 second, then **stop** motor
- **Wait** 2 second, than **repeat all** previous points again and again …



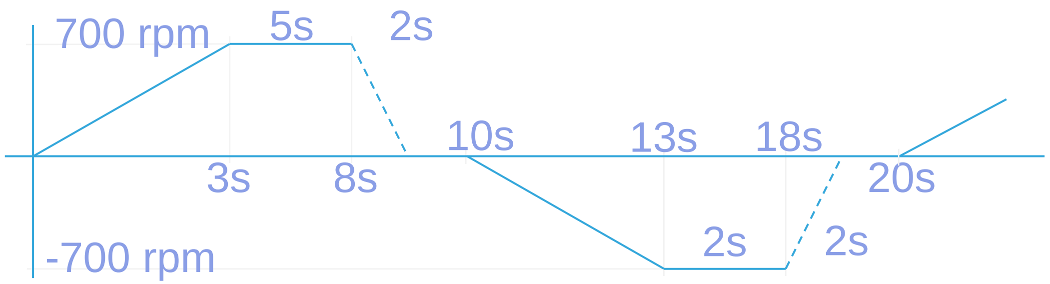
life.augmented

ST TRUEStudio hint

- If you need show Template Proposals you have to use **Ctrl+Space**



Start / Stop – motor control by API



- Open TrueSTUDIO project **WorkShop01** and main.c file.
- Find the Infinite loop inside the main function.
- Type the function for speed ramp and for start motor:

  // hFinalSpeed is mechanical rotor speed reference at the end of the speed ramp
  // it is expressed in tenths of HZ. (The rpm value has to be divided by 6)

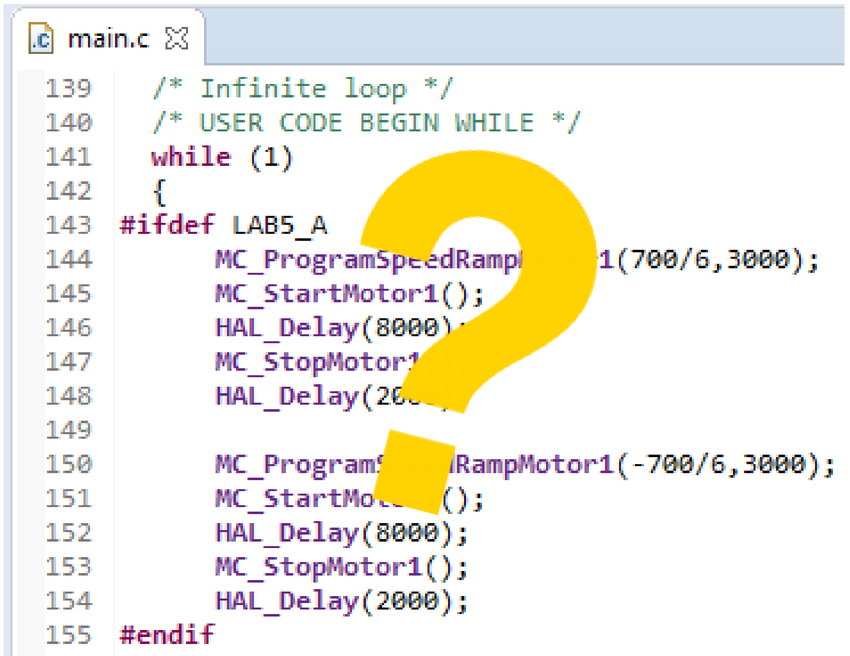
  void **MC_ProgramSpeedRampMotor1**(int16_t hFinal**Speed**, uint16_t h**Duration**ms);
  bool **MC_StartMotor1**(void);

- Use HAL function for waiting till 5th second then stop motor:

  HAL_Delay(8000); //delay contains the start-up time and 5 second run ➔ 8000 ms
  MC_StopMotor1();
  HAL_Delay(2000);

- Write down four similar lines for the reverse direction.

  Reverse direction?
  ➔ Speed is with minus signature



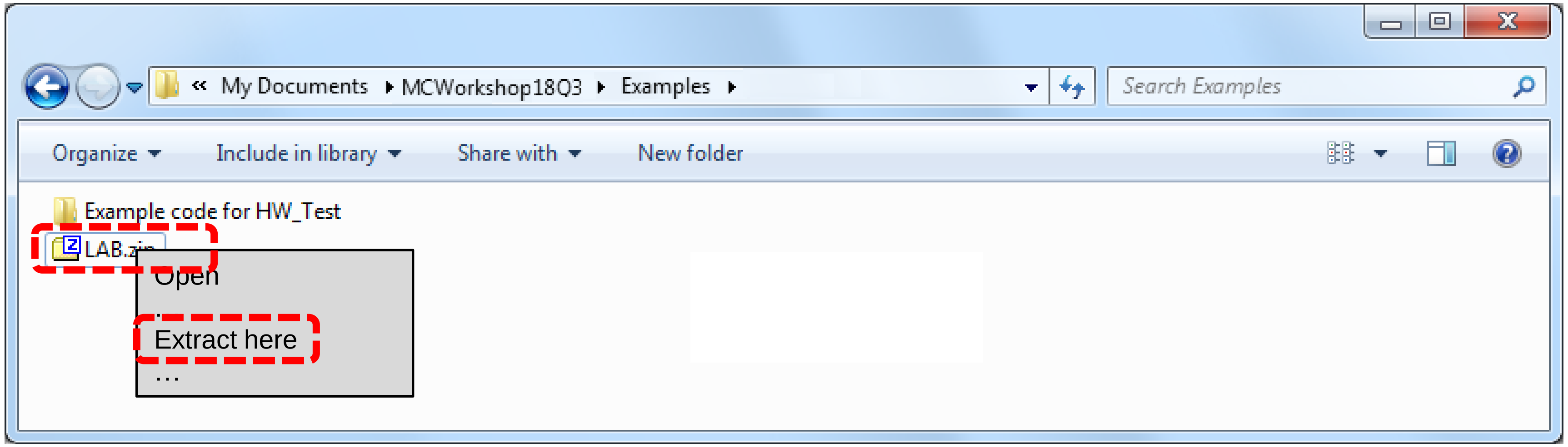
- Time to work (10mins) on the example. **Let's go!**

Hands-On section

Finalized examples

- You had not enough time, you have lost somewhere… ☹
- Don’t worries! We have prepared for you finalized examples! ☺
- Open Workshop folder **..\Documents\MCWorkshop18Q3\Examples**
- Find package file LAB.zip
- Use right button on mouse
- Select ”Extract here”

  Password is …



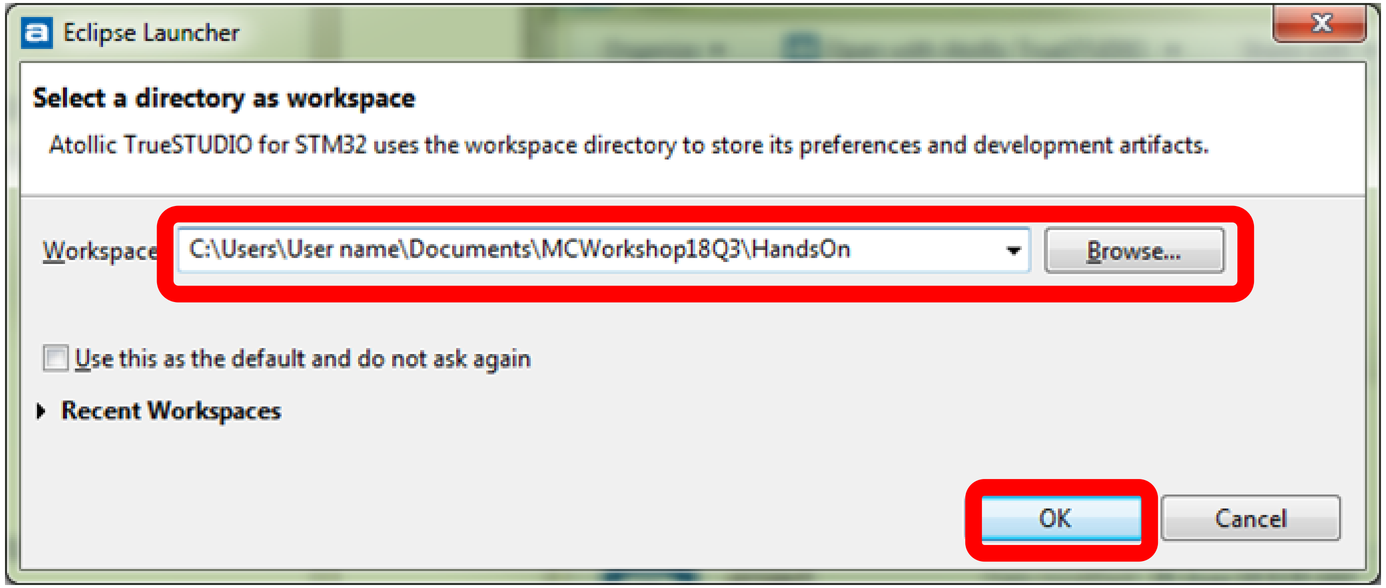
Hands-On section

How open finalized examples in ST TrueSTUDIO

- Open Atollic TrueSTUDIO
- Select Workspace

C:\Users\User Name**Documents\MCWorkshop18Q3\HandsOn**

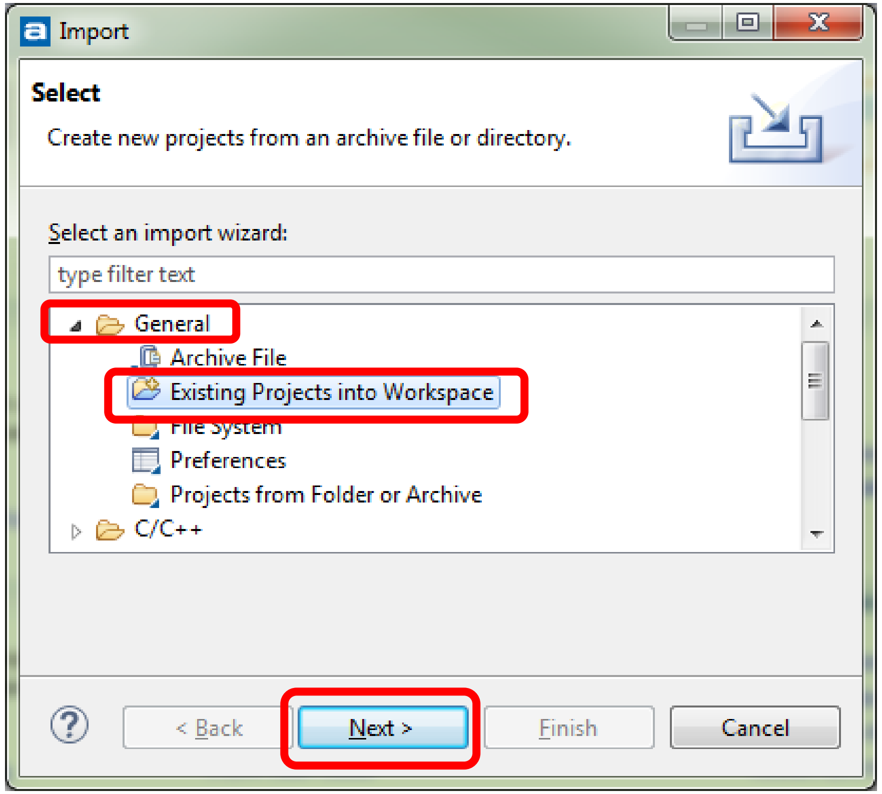
- Click on OK

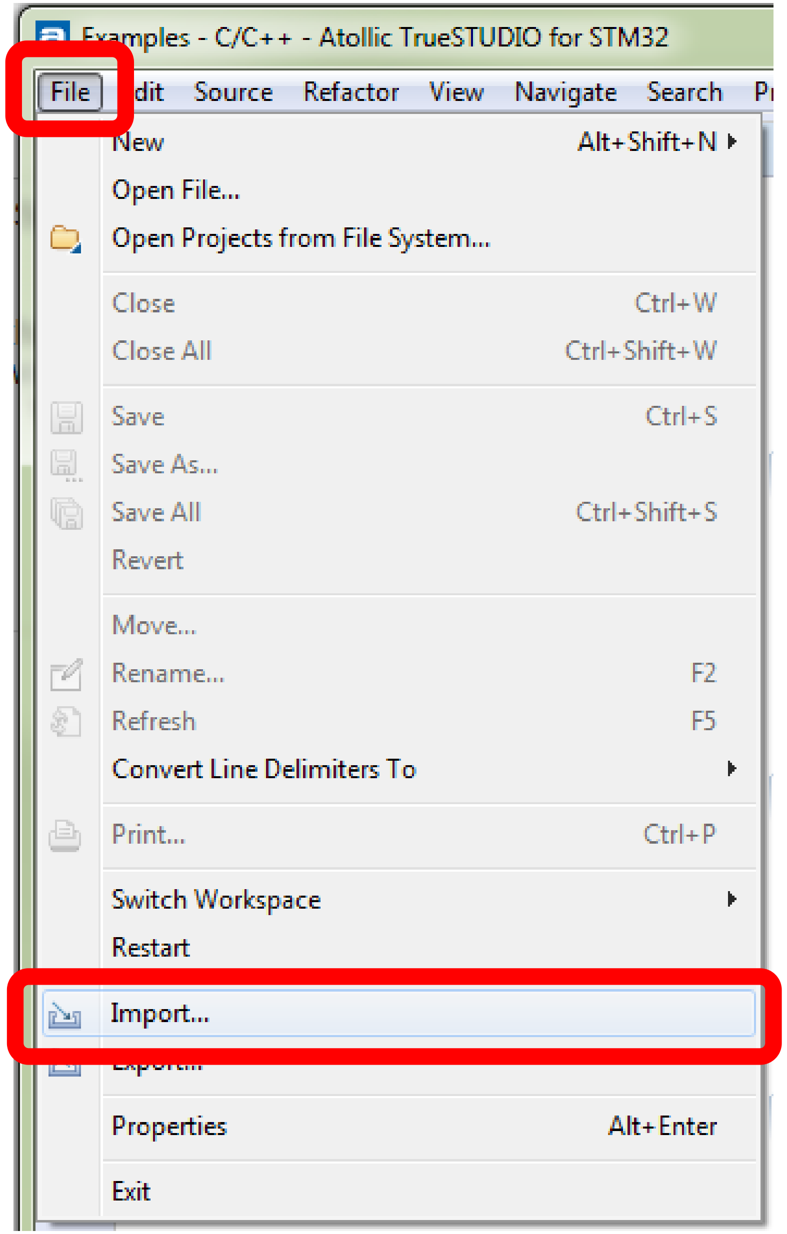


Hands-On section

How open finalized examples in ST TrueSTUDIO

- Click on **File** and **Import**
- Select **General**, **Existing Projects into Workspace** and **Next**





life.augmented

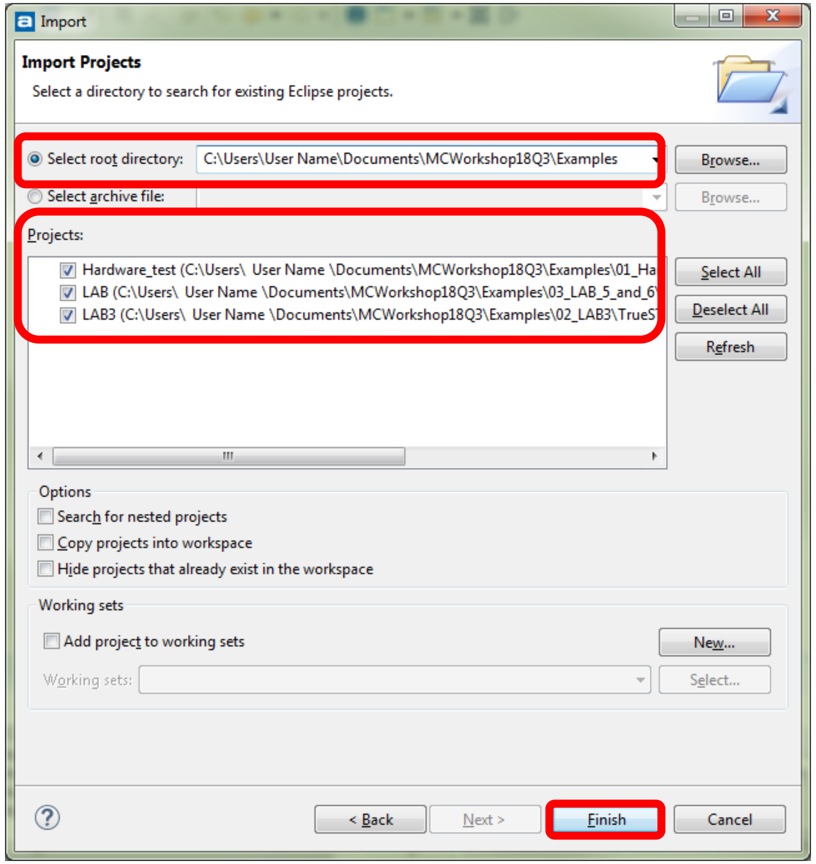
Hands-On section

How open finalized examples in ST TrueSTUDIO

- Select root directory:

*C:\Users\User Name**Documents\MCWorkshop18Q3\Examples***

- You should see three examples
- Click on **Finish**

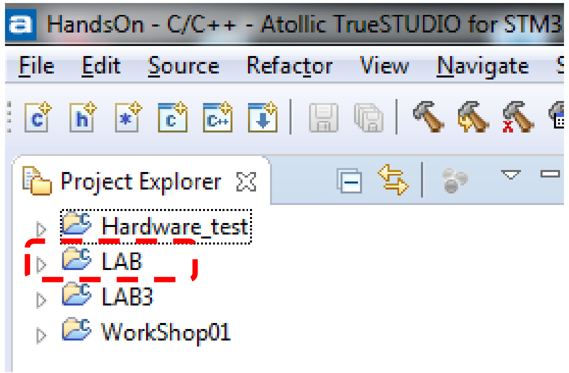


life.augmented

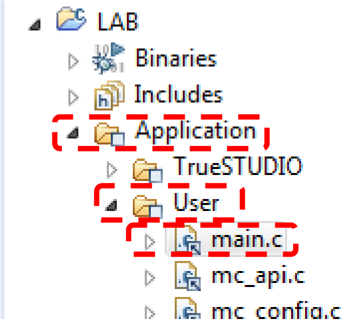
Hands-On section

How use finalized examples in ST TrueSTUDIO

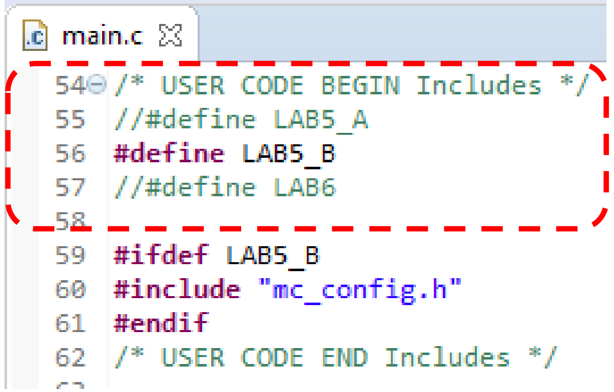
- After import you should see three Project



- Open LAB Project
- Click on Application, User and main.c



- In the USER CODE BEGIN Includes section are predefined LAB5_A, LAB5_B and LAB6.
  You can select one by (un)comment and test it



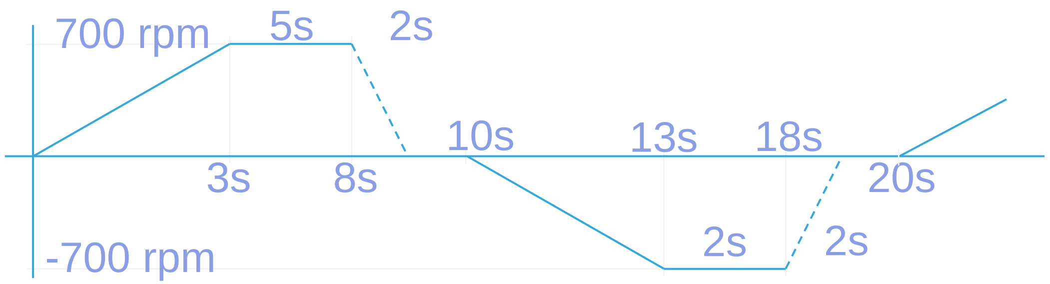
life.augmented

Hands-On section

How use finalized examples in ST TrueSTUDIO

main.c

```c
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
#ifdef LAB5_A
      MC_ProgramSpeedRampMotor1(700/6,3000);
      MC_StartMotor1();
      HAL_Delay(8000);
      MC_StopMotor1();
      HAL_Delay(2000);

      MC_ProgramSpeedRampMotor1(-700/6,3000);
      MC_StartMotor1();
      HAL_Delay(8000);
      MC_StopMotor1();
      HAL_Delay(2000);
#endif
```



- ST TrueSTUDIO – Compile and load your MC application

Build 'Debug' for project

Click on the icon "**Build**" or press the keys "Ctrl+B"

Debug

- Click on the icon "**Debug**" or press the key "F11"

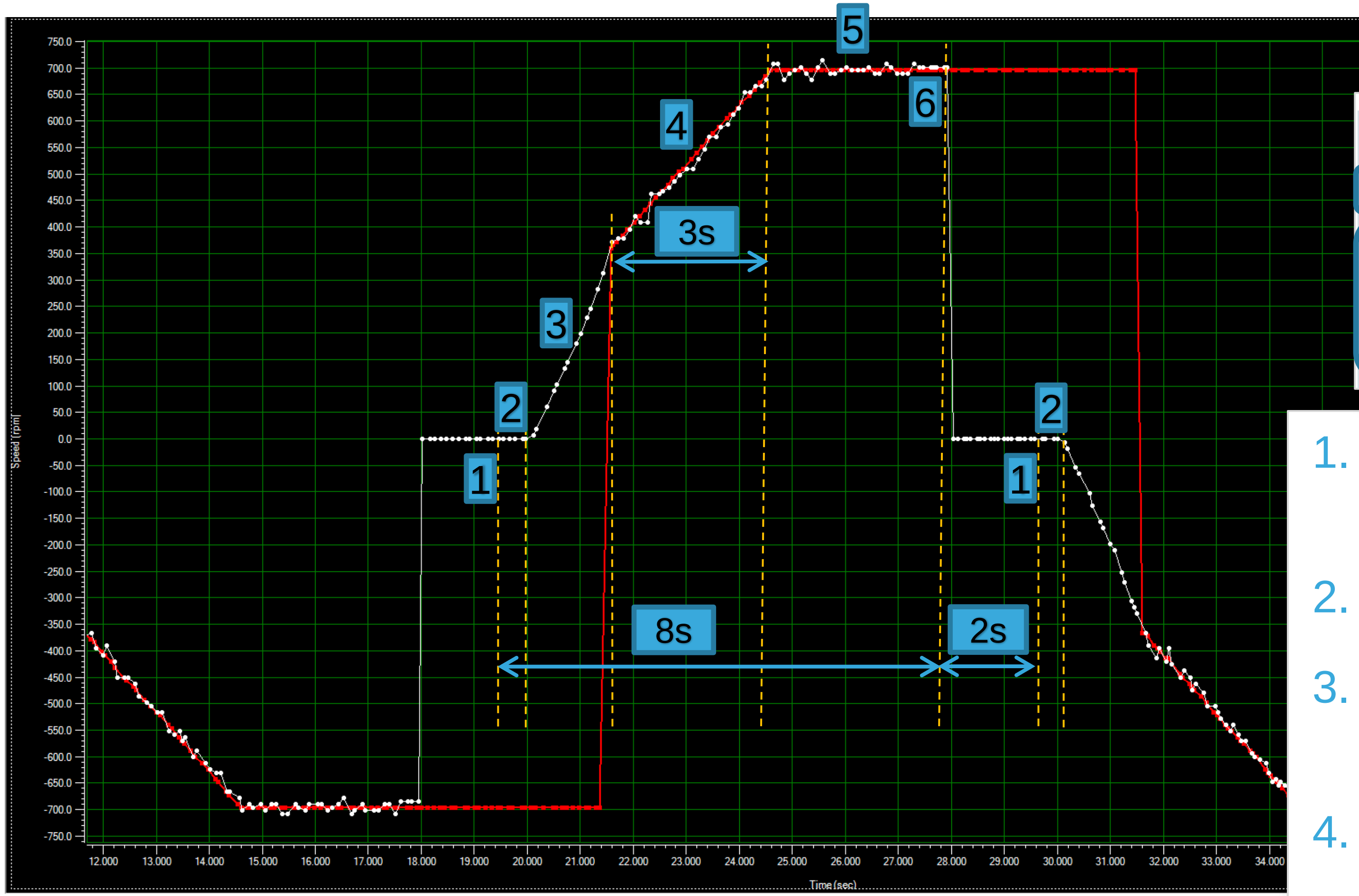
Terminate (Ctrl+F2)

- **Close the Debug** to start your MC Application

life.augmented

Hands-On section

LAB5 - Start / Stop – motor control by API – time overview



Revup Configuration

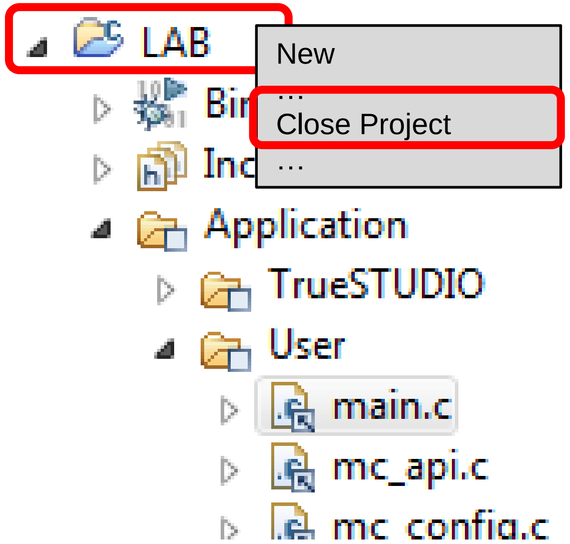
Num	Final Speed (rpm)	Final Torque	Duration (ms)
1	0	1504	500
2	120	1504	500
3	246	1504	500
4	372	1504	500
5	498	1504	500

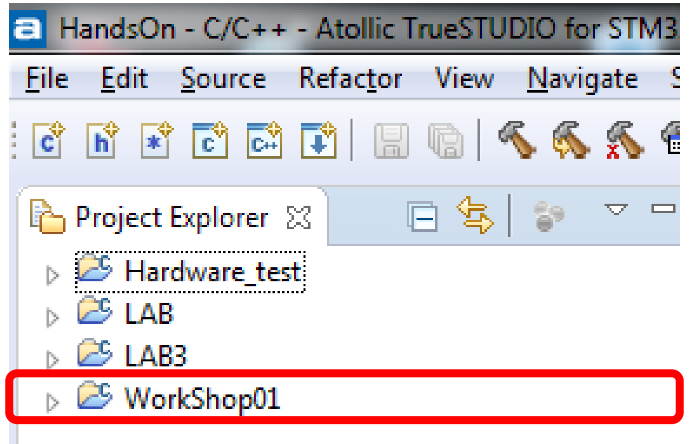
1. Calling of functions SpeedRamp and StartMotor
2. Start-up step 1 – alignment → zero speed
3. Start-up ramp – switching from open to close loop is over speed threshold 350rpm
4. Applied SpeedRamp 700rpm is reached during 3 seconds
5. Reached 700 and keeping in stable speed
6. Calling StopMotor and wait 2 seconds

Hands-On section

Next task

- You will have time to do next task.
- Close the finalized examples
  - Select “LAB”
  - Use right button on mouse
  - Select ”Close Project”
  - Open WorkShop01
- Please follow the instructions.
- Do not worry if you will lose somewhere. ☹
- You know, you can use prepared finalized examples. ☺





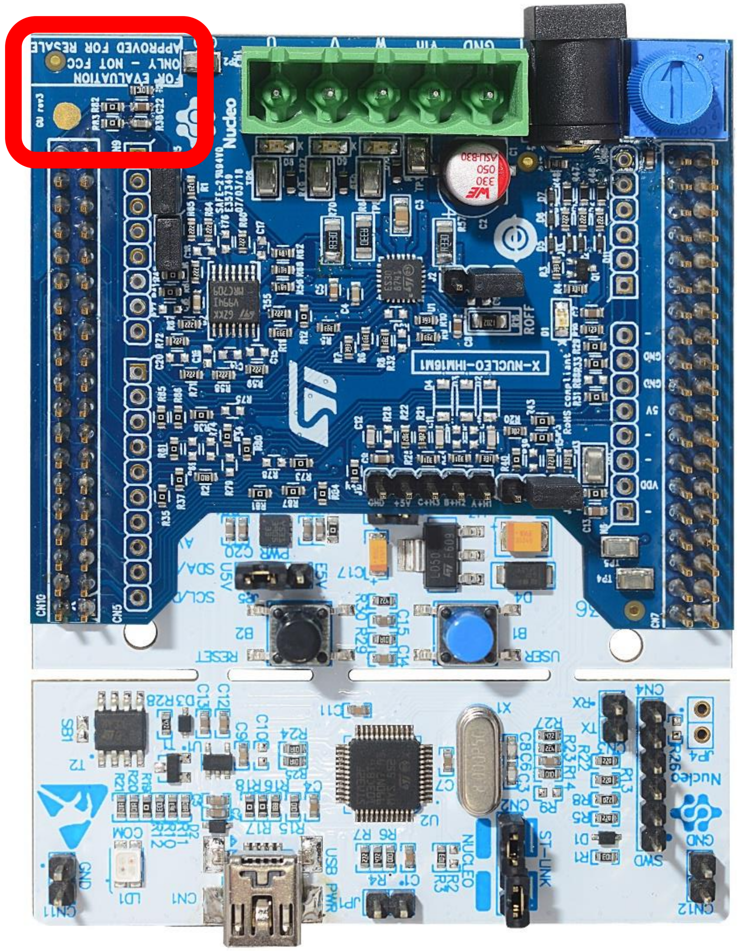
life.augmented

Speed control motor by API

Adapt previous example to swap between two speed values 700 rpm and 1400 rpm depending on temperature

- Read Power Board temperature
  - NTC sensor is located next to green motor connector

    Use function NTC_GetAvTemp_C



- Compare measured temperature
  - If (> 29°C) then spin slower (700 rpm)
  - else spin faster (1400 rpm)

Hands-On section

Speed control motor by API

- Time to work (10mins) on the example.
- Use previous example with ramps
- You have to get the pointer of MC Tuning list.

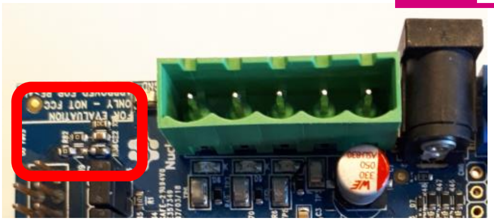
  #include “mc_config.h” //write it before Infinite loop

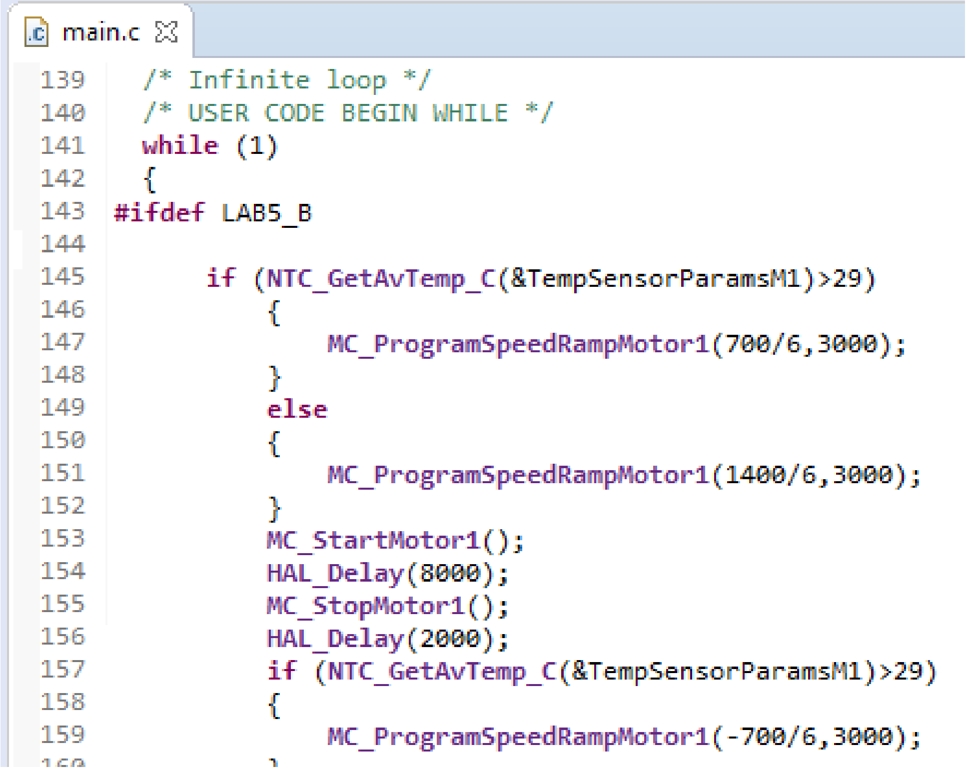
- Add temperature reading and compare result with 29°C

  Use finction NTC_GetAvTemp_C and structure from “mc_config.h”

  // write the if condition before rows with the speed ramp

```
if (NTC_GetAvTemp_C(&TempSensorParamsM1)>29)
{                  // If temp is great then use slower speed 700 rpm
   MC_ProgramSpeedRampMotor1(700/6, 3000);
}
else               // If not then use faster speed 1400 rpm
{                  // write down the second case with faster speed
   MC_ProgramSpeedRampMotor1(1400/6, 3000);
}
```

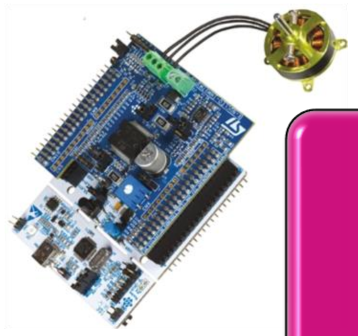



main.c

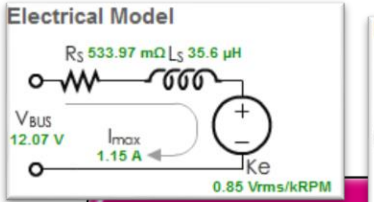
```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
#ifdef LAB5_B

        if (NTC_GetAvTemp_C(&TempSensorParamsM1)>29)
          {
            MC_ProgramSpeedRampMotor1(700/6,3000);
          }
          else
          {
            MC_ProgramSpeedRampMotor1(1400/6,3000);
          }
          MC_StartMotor1();
          HAL_Delay(8000);
          MC_StopMotor1();
          HAL_Delay(2000);
          if (NTC_GetAvTemp_C(&TempSensorParamsM1)>29)
          {
            MC_ProgramSpeedRampMotor1(-700/6,3000);
```

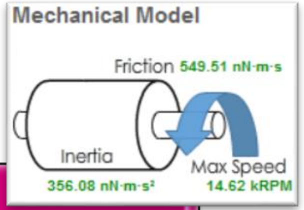
Motor Control Development Workflow

#6 – Application development



Hardware Setup



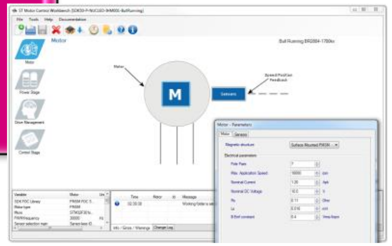


Motor Characterization

Motor Profiler
Motion Control Suite

System Configuration

Motor Control Workbench



Project Configuration

CubeMX & IDE

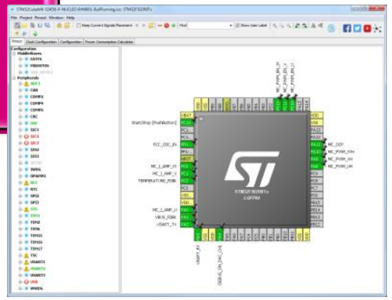
STM32 CubeMX

ac6

TrueSTUDIO for STM32

IAR SYSTEMS

KEIL Tools by ARM



Motor Drive Tuning

Tune MC part

Final Application Development

User code

```
#include "MC.h"
{
    CMCI oMCI = GetMCI(M1);
    MCI_ExecSpeedRamp(oMCI, final speed, ramp duration);
    MCI_StartMotor(oMCI);
}
```

life.augmented

LAB 6 tasks

Integration of additional IP's into your Motor Control project using STM32CubeMX

- Setup additional pins and hardware elements by STM32CubeMX
  - Blue USER button

Hands-On section

New HW setup in STM32CubeMX

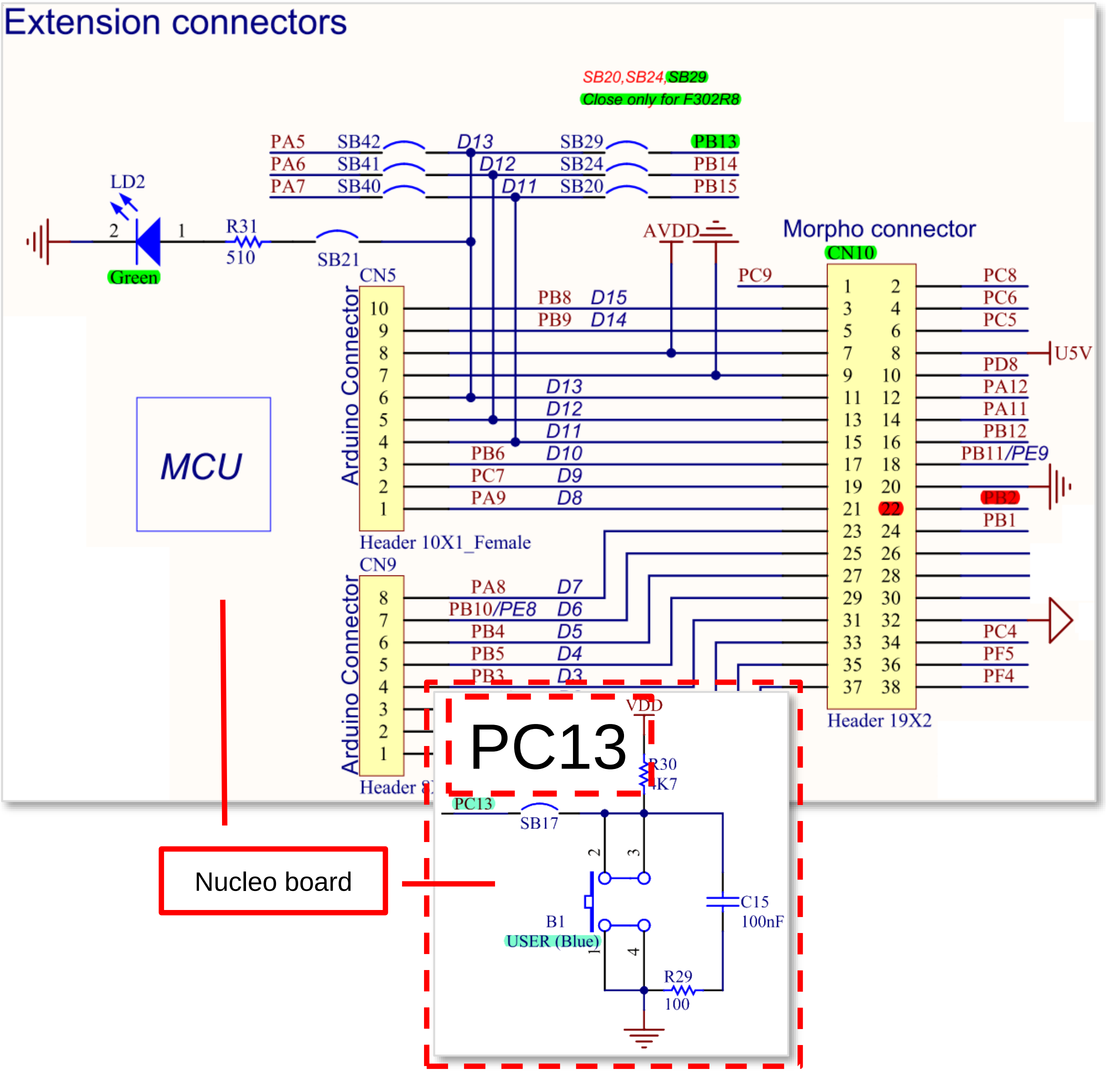
1. Use the Blue **USER button** to **start** and to **stop** the motor
    a) Find the connection of the blue USER button in the schematic
    b) Disable the control of the blue USER button by the MC Library
    c) Use the pin, select and set external interrupt function in STM32CubeMX
    d) Write down a code for handling the button and control the motor

Hands-On section

Find the I/O pins connection

In the schematics
of “NUCLEO-F303RE” board
find the connections of

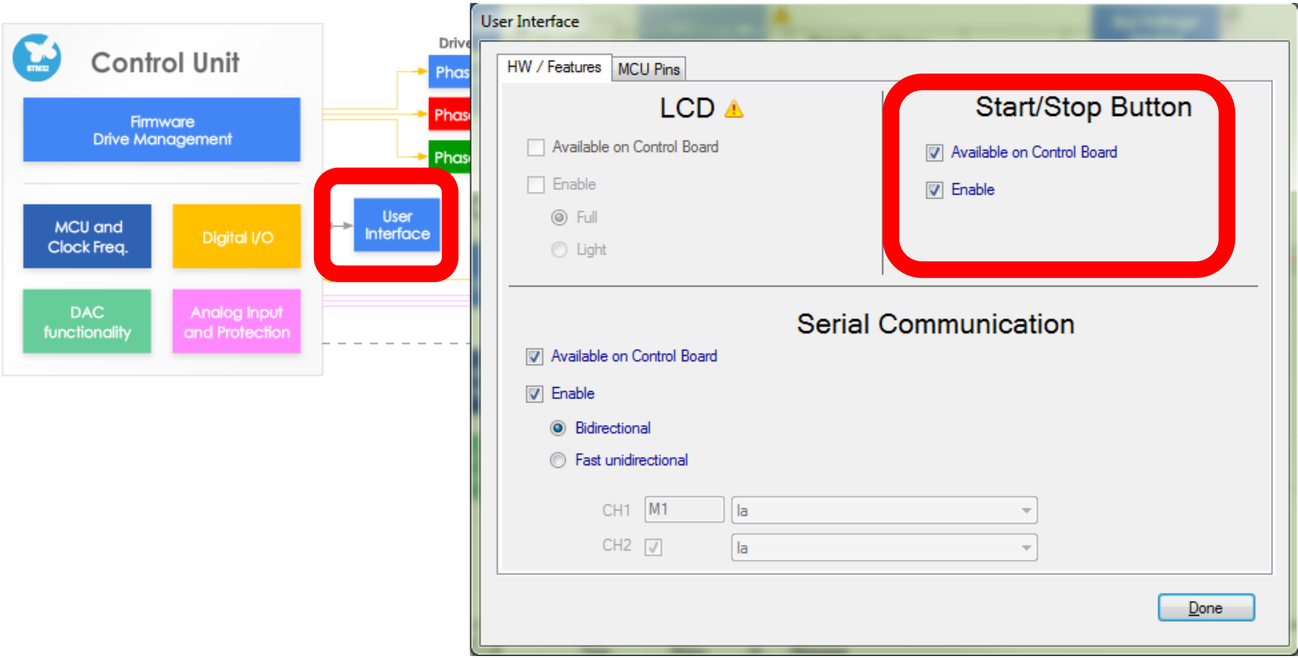
Blue USER button ➔ PC13



life.augmented

Setting in MC Workbench

- Open example project in WB for **Nucleo F303RE & IHM16 & *GimBal***
- In **User Interface unselect Start/Stop Button**
- **Save As "WorkShop02"** to the ***HandsOn*** *folder*

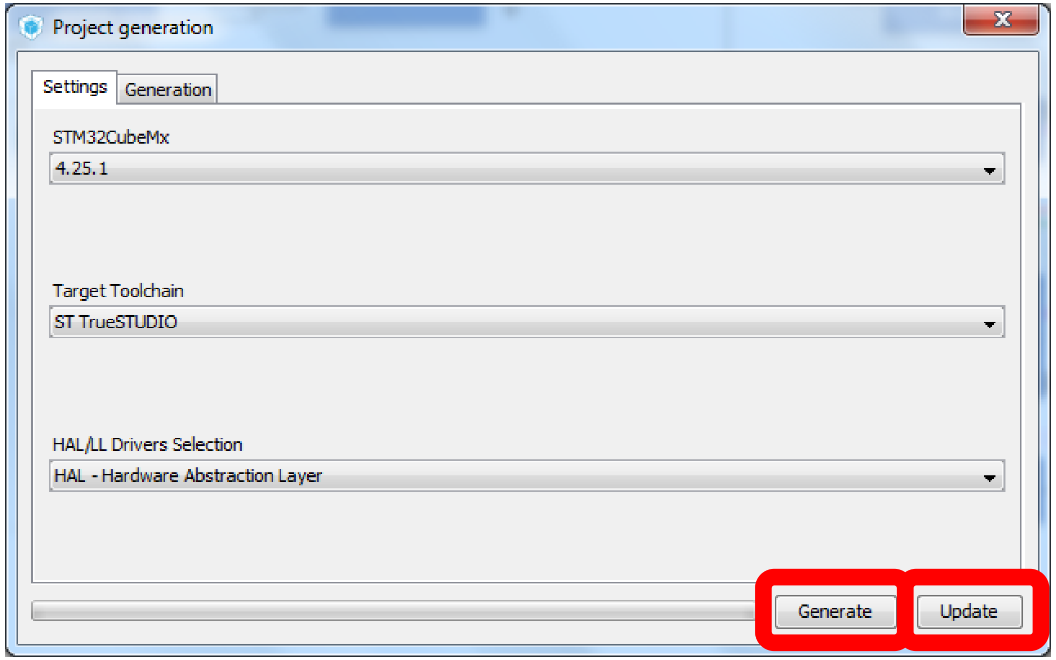


Hands-On section

Generate project from MC Workbench

- Generate project to the ***HandsOn*** *folder*

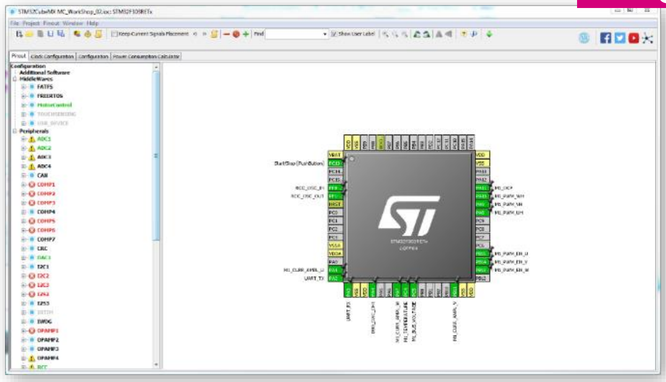
- *There are two buttons*
  - *Generate*
    - *It does not take into account modifications inside .ioc file*
    - *Simply generate new .ioc from WB project*
  - *Update*
    - *It generate motor control part*
    - *Rest of setting is reused from previous .ioc*
    - *Have to be used after you have made additional setting by Cube MX*



life.augmented

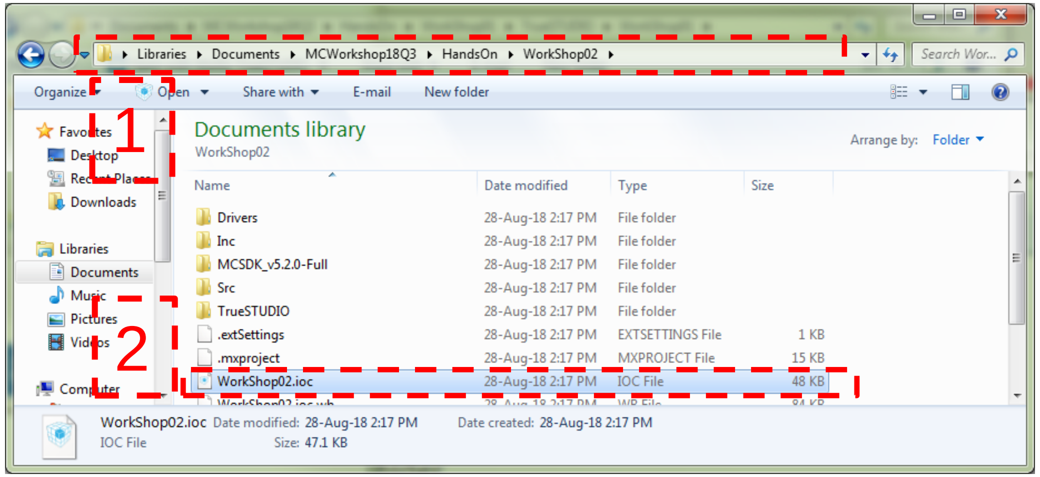
Hands-On section

STM32CubeMX project



- Open the generated Cube MX project

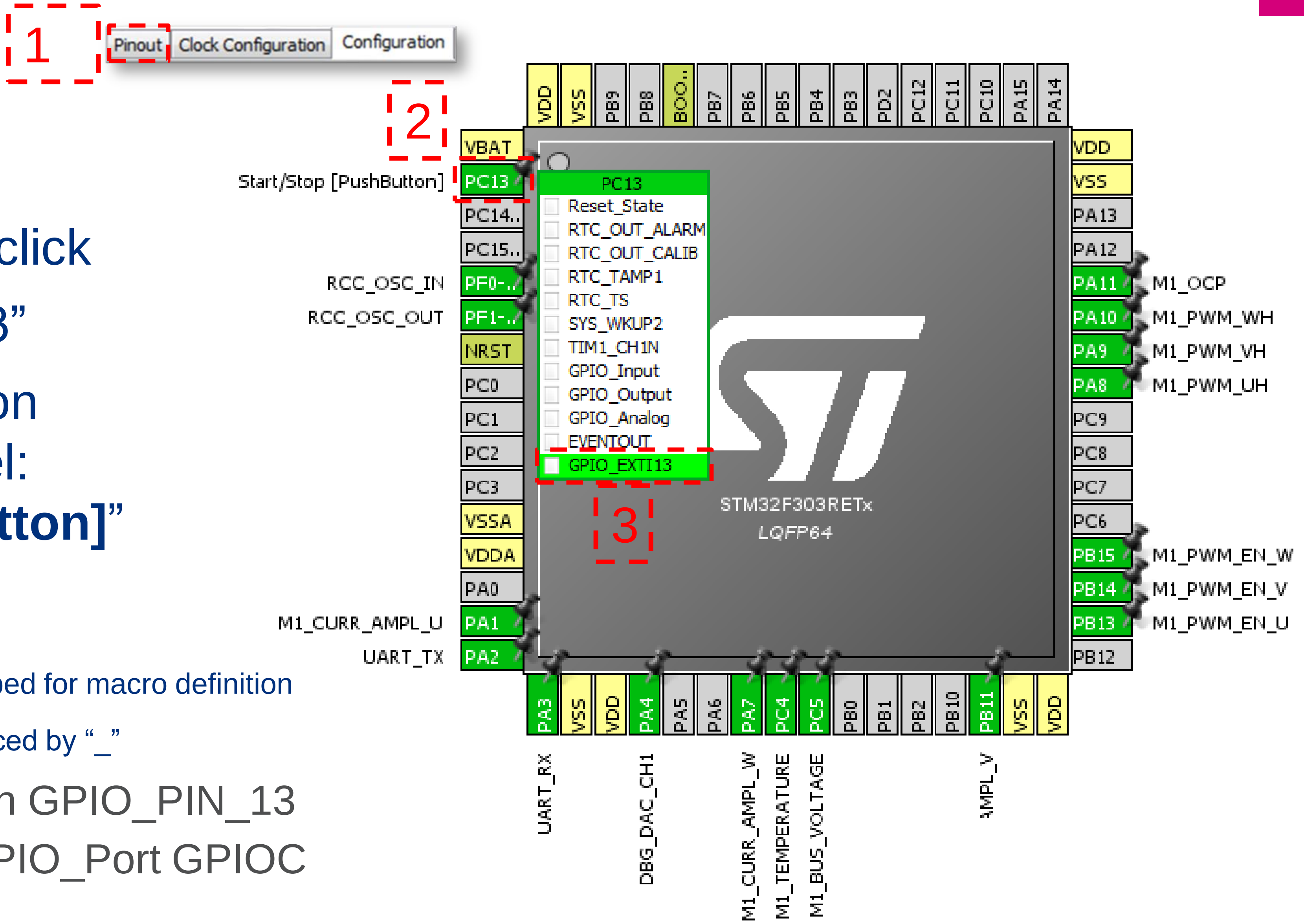
1. Open the project folder
   C:\Users\User Name**Documents\MCWorkshop18Q3\HandsOn**
2. Launch “STM32CubeMX” project initiated from MC Workbench using double click on .ioc file “**WorkShop02.ioc**”



Modify the pinout layout

Hands-On section

1. Select “Pinout” tab
2. Select the PC13 and click
3. Select “GPIO_EXTI13”
4. Use right mouse button and enter the pin label: “**Start/Stop [PushButton]**”



Comment will be skipped for macro definition

The forbidden characters in C will be replaced by “_”

➔ #define Start_Stop_Pin GPIO_PIN_13

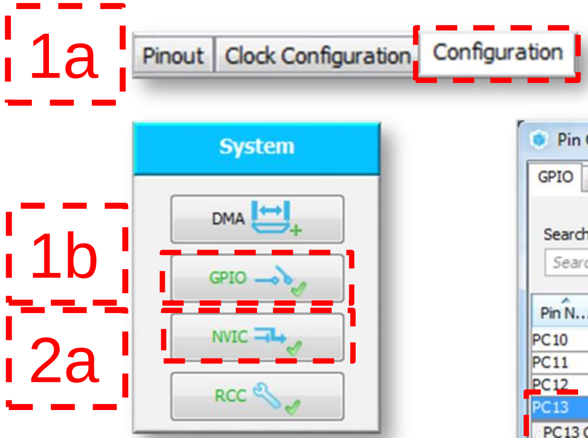
#define Start_Stop_GPIO_Port GPIOC

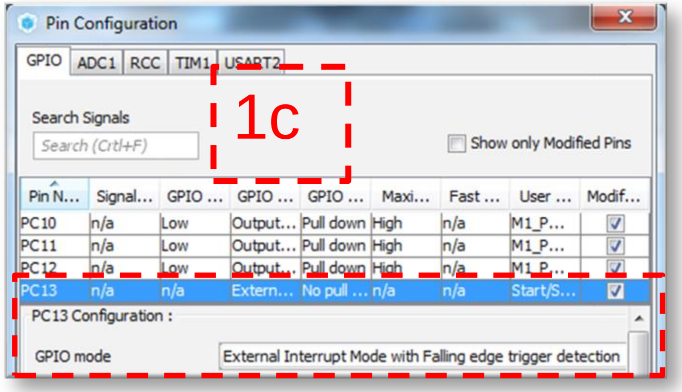
life.augmented

Modify the pinout setting of button

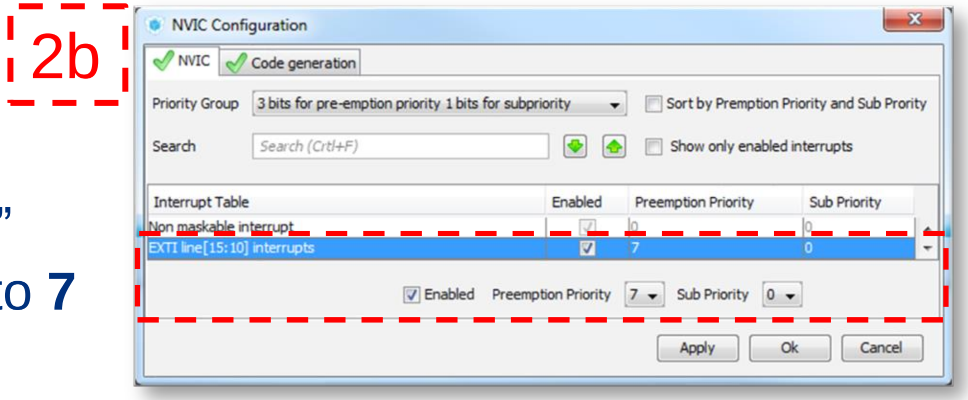
- We have to configure the external interrupt input usage of USER button

1. Configure **GPIO** PC13 as EXTI
   a) Select the "**Configuration**" tab
   b) Click on the "**GPIO**" **button**
   c) Select PC13 in the "**GPIO**" **tab** and set the **GPIO mode** to "**External Interrupt** Mode with **Falling edge** trigger detection"





2. Set **interrupt mode** in NVIC
   a) Click on the "**NVIC**" **button**
   b) **Enable** the "**EXTI line[15:10] interrupts**" group in the "**NVIC**" **tab** and set **Priority** to **7**

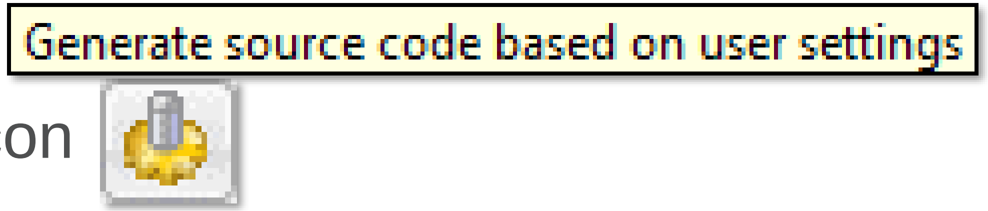


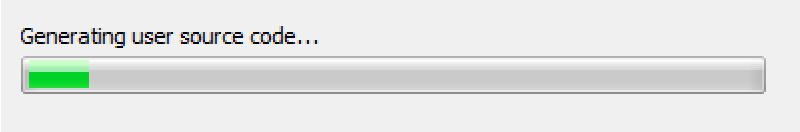
life.augmented

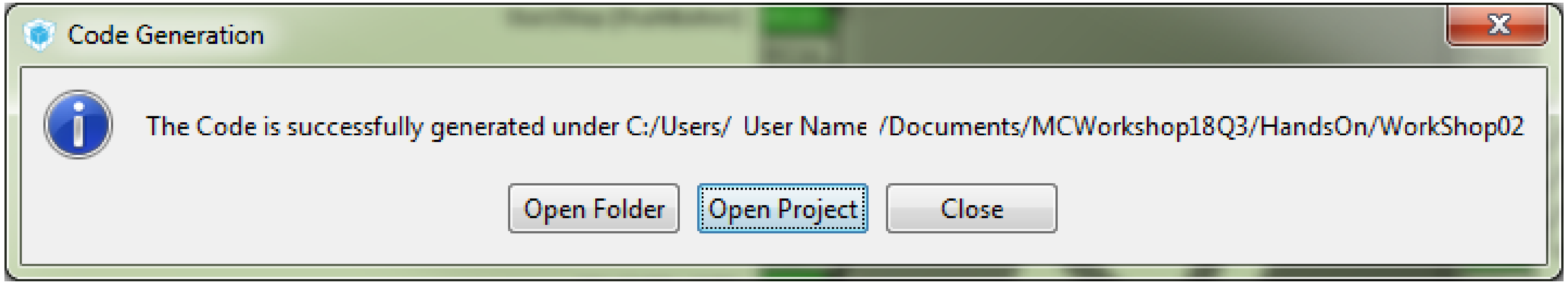
Hands-On section

Generate the firmware

- STM32CubeMX – Generate the firmware
  - Start STM32CubeMX **generation** from icon or use short-keys Ctrl+Shift+G







Hands-On section

Push Button code

- Write down the code for the Button
  - Use function for getting the status of motor 1 **MC_GetSTMStateMotor1** and compere it with the state **IDLE**
  - **Code**

if(GPIO_Pin == Start_Stop_Pin) {

State_t MState = MC_GetSTMStateMotor1();

if (MState == IDLE)   {

MC_ProgramSpeedRampMotor1(700/6,500);

MC_StartMotor1(); }

else

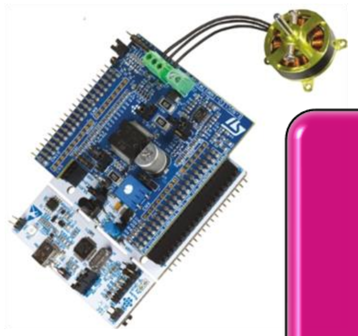
MC_StopMotor1();   }

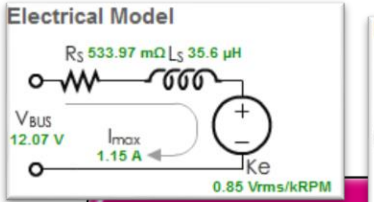
main.c

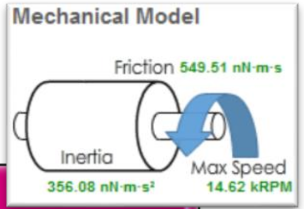
```c
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
#ifdef LAB6
    if(GPIO_Pin == Start_Stop_Pin)
    {
        State_t MState = MC_GetSTMStateMotor1();
        if (MState == IDLE)
        {
         MC_ProgramSpeedRampMotor1(700/6,500);
         MC_StartMotor1();
        }
        else
         MC_StopMotor1();
    }
#endif
}
/* USER CODE END 4 */
```

Motor Control Development Workflow

#6 – Application development







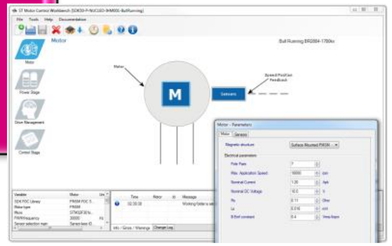
Hardware Setup

Motor Characterization

Motor Profiler
Motion Control Suite

System Configuration

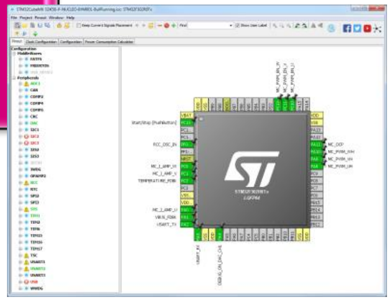
Motor Control Workbench



Project Configuration

CubeMX & IDE

STM32 CubeMX



ac6 TrueSTUDIO for STM32 IAR SYSTEMS KEIL Tools by ARM

Motor Drive Tuning

Tune MC part

Final Application Development

User code

```
#include "MC.h"
{
    CMCI oMCI = GetMCI(M1);
    MCI_ExecSpeedRamp(oMCI, final speed, ramp duration);
    MCI_StartMotor(oMCI);
}
```

life.augmented