# STM32 PMSM SDK 5.2 training

T.O.M.A.S. team

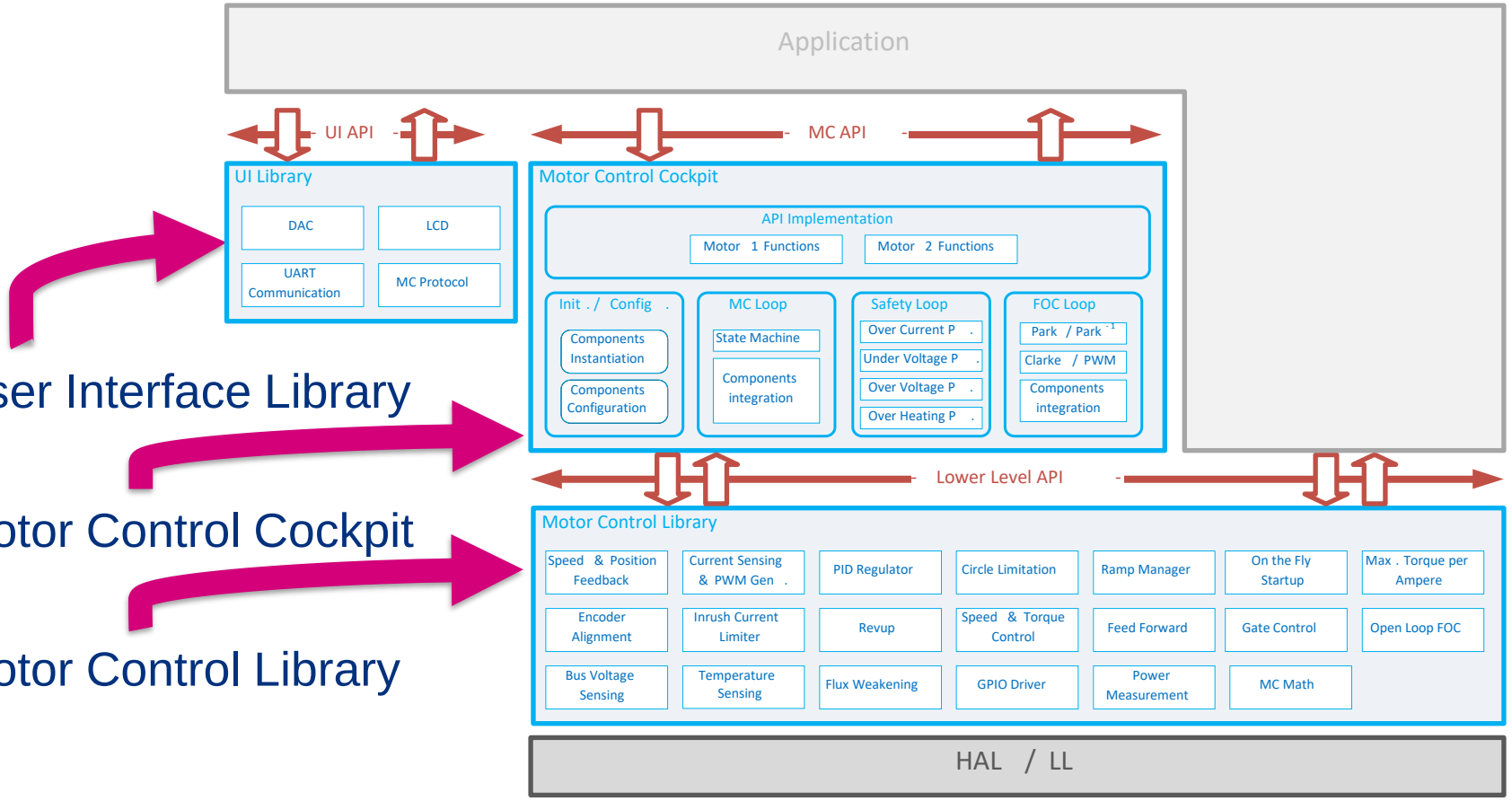
life.augmented

# 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



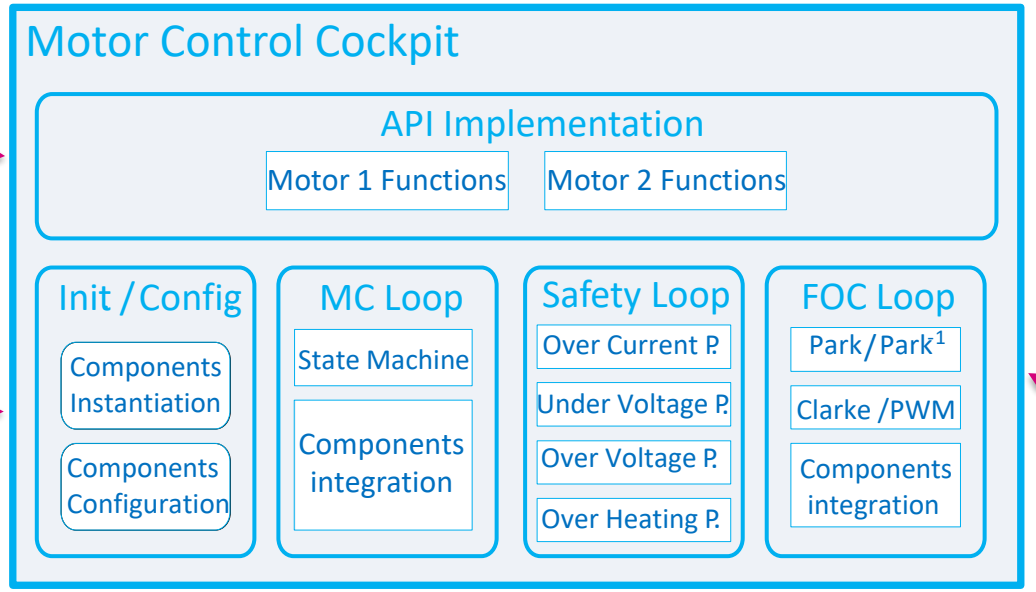
# Motor Control Cockpit - Introduction



- 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)

# Application Programming Interface API

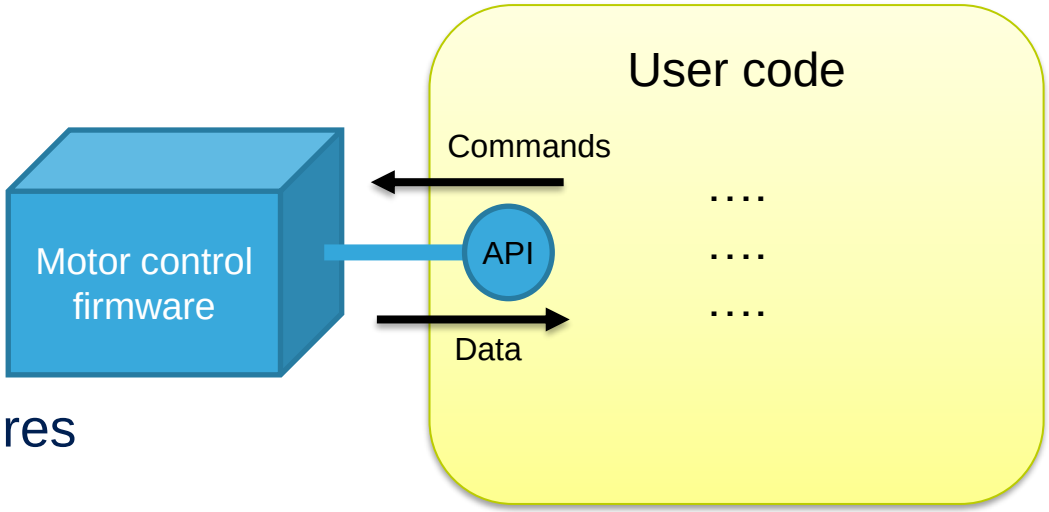
life.augmented

# API - Application Programming Interface



## What is an API?

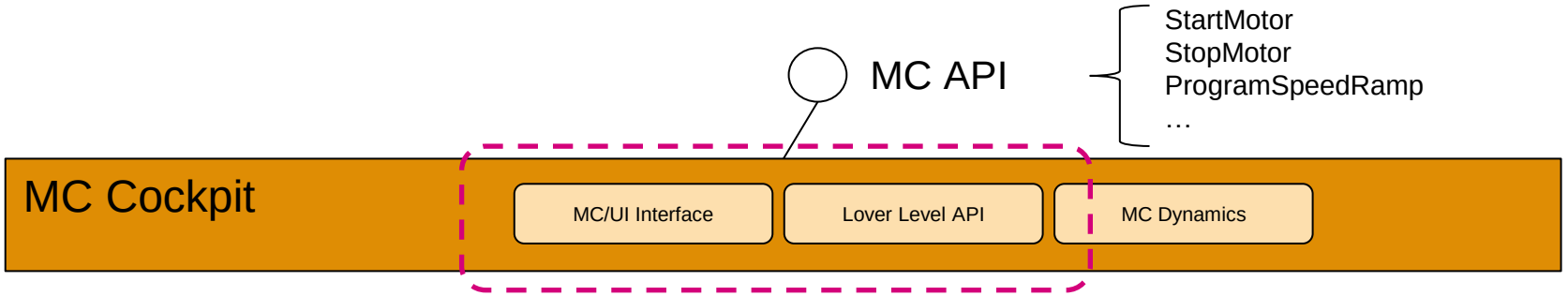
- An Application Programming Interface (API) specifies how software components should interact with each other.
- It provide a consistent, programmatic method for accessing a resource.
- It is a structured way of exposing functionalities.
- Unlike an user interface the API is a macchine to macchine interface.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



# Motor Control (MC) API



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




| | Functionality | Intended use | Example functions |
|---|---|---|---|
| **MC/UI Interface** | Basic | Basic user code | MC_StartMotor1<br>MC_StopMotor2<br>MC_ProgramSpeedRampMotor1 |
| **LL API** | Full | Tuning<br>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 discarted.

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

# 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();
...
}
```

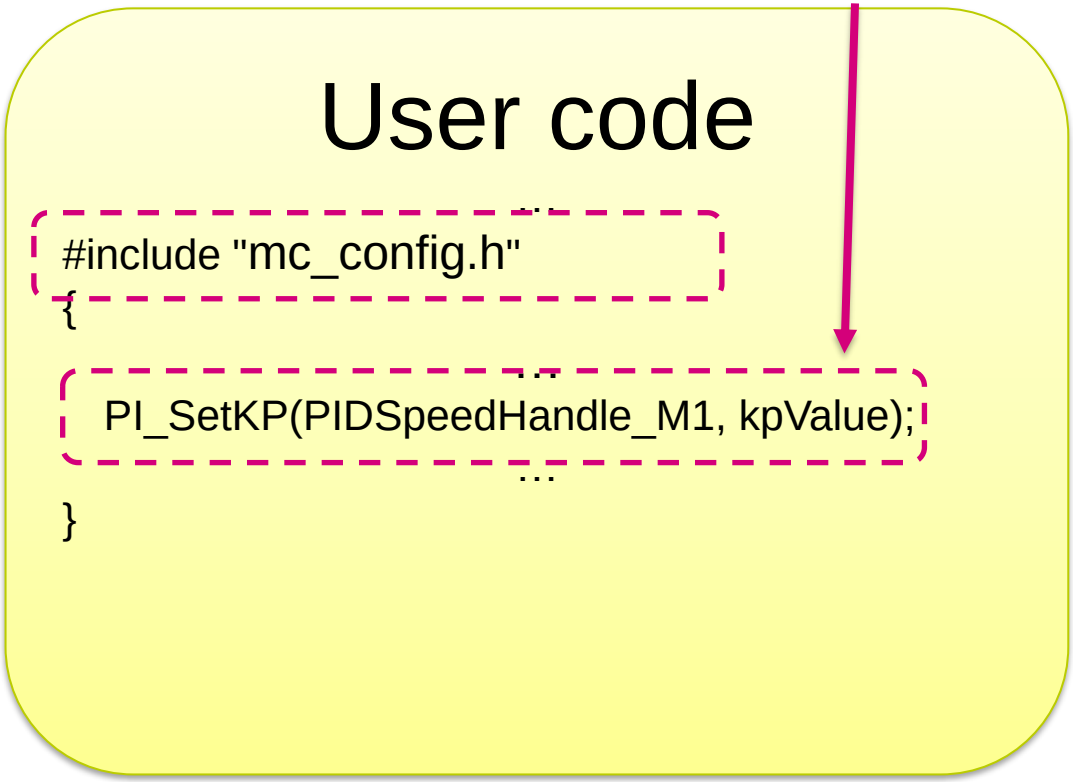
# MC Low Level API functions



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

***M1 means motor 1***

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




PIDSpeedHandle_M1 =
= pointer of LL_API 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

- Current / Torque units implementation: Unit is s16A

$$(Current_{Amps} * 2^{16} * R_{shunt} * Gain) / V_{dd\_micro} = Current_{s16A}$$

# MC Application Commands



Example list of APIs Commands:

```
MC_StartMotor1();
MC_StopMotor1();
MC_GetSTMStateMotor1();
MC_ProgramSpeedRampMotor1();
MC_GetMecSpeedAverageMotor1();
```

```
PID_SetKI(PIDIdHandle_M1, number);
number =PID_GetKI(PIDIdHandle_M1);
```

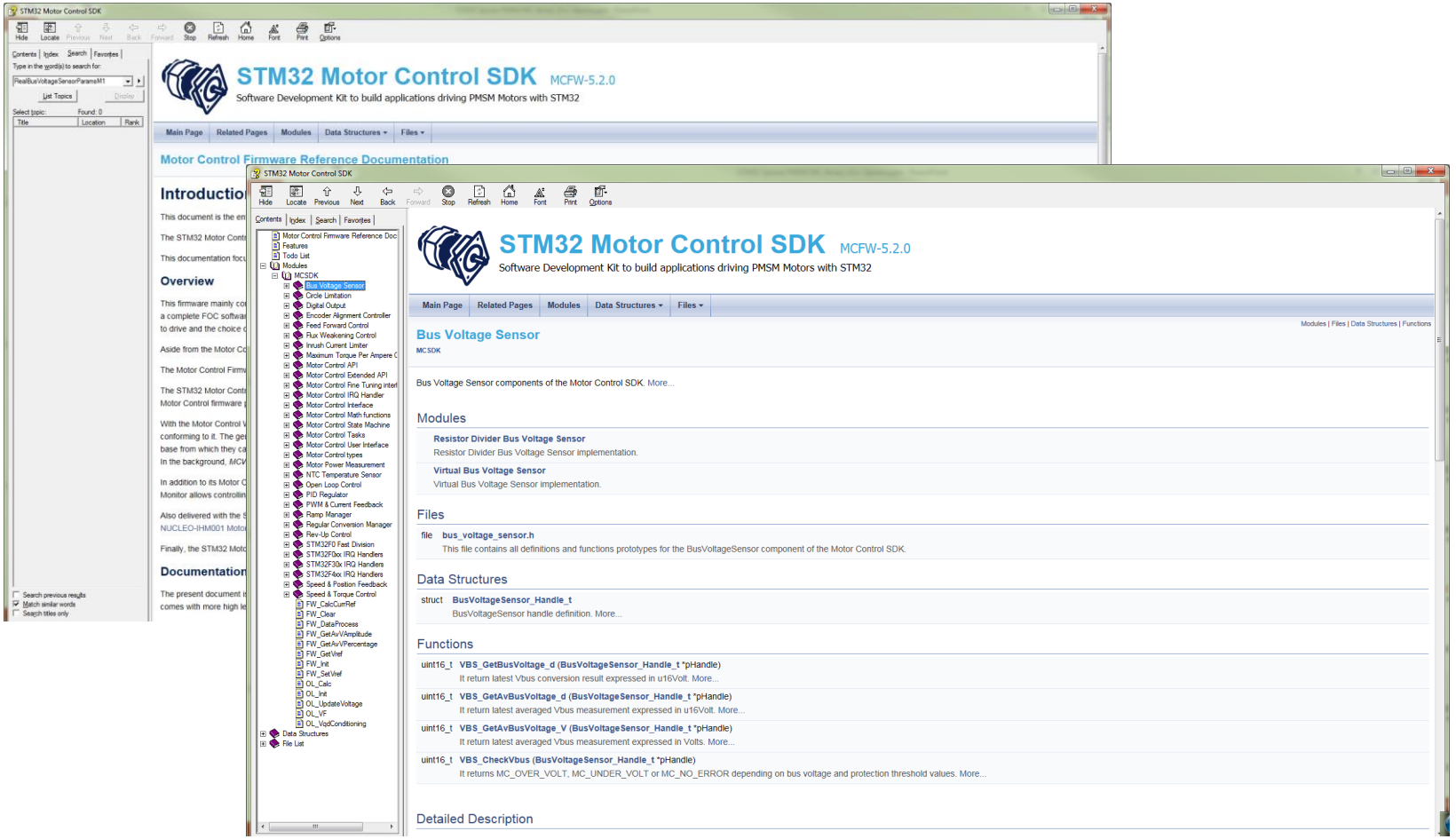
```
NTC_GetAvTemp_C(TempSensorParamsM1);
```

```
VBS_GetAvBusVoltage_V(&(RealBusVoltageSensorParamsM1._Super));
```

# MC API help file



- The complete and detailed list of MCI and MCT functions can be found in the ***STM32 Motor Control SDK library Help file*** present in the DOC folder.



# API HandsOn

life.augmented

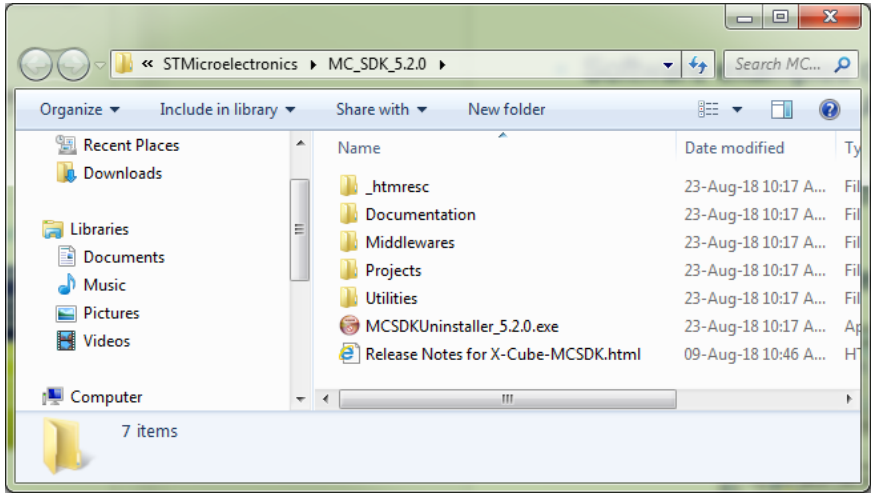
# Example project

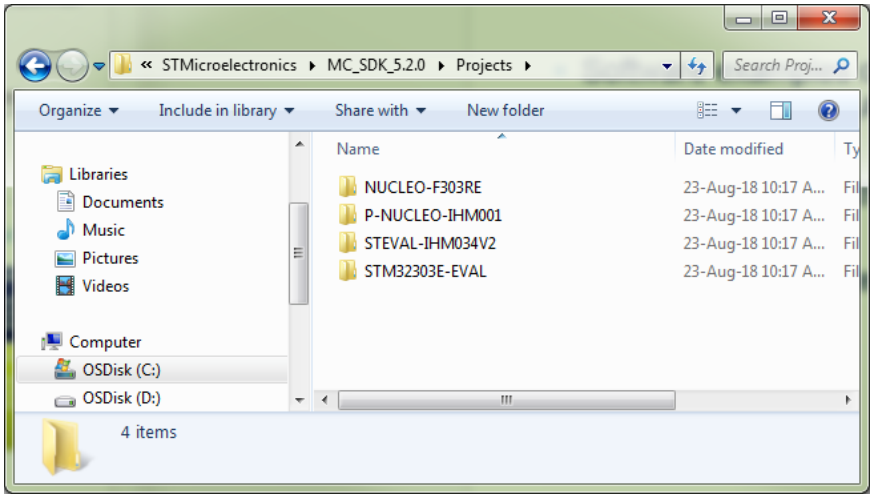
# Software examples



- Software examples can be used as staring point for new design or guideline to understand the MC API.

- You can found in installed path MC_SDK_5.x.x/Projects



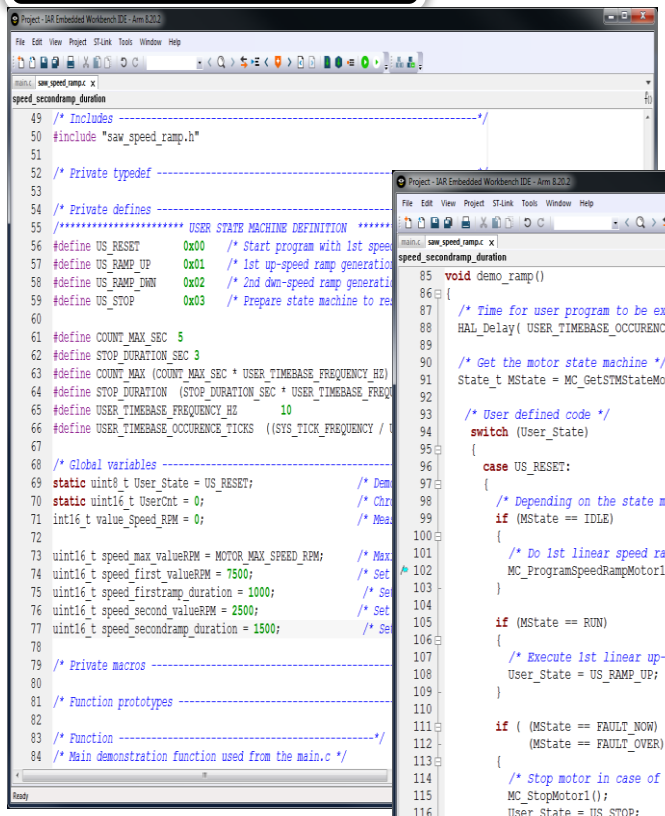
# Saw Speed Ramp – project example



How to write an application layer based on STM32 MC FW library

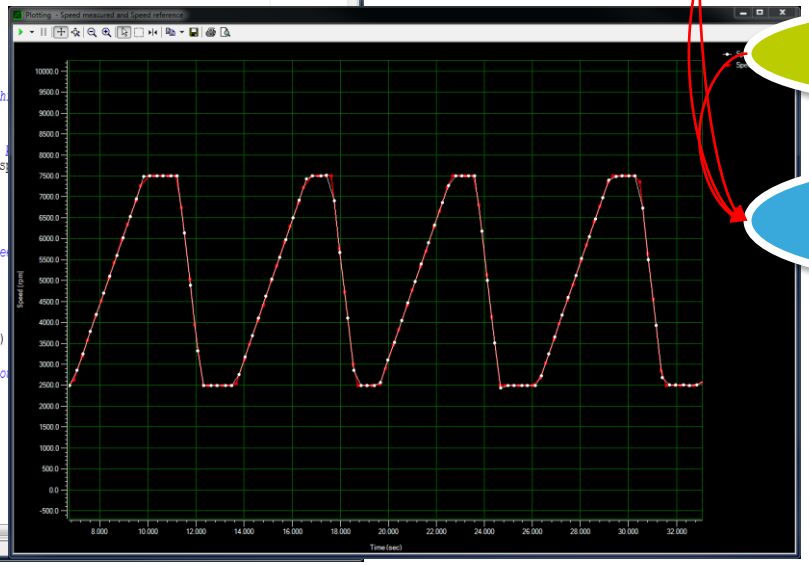
Saw Speed Ramp

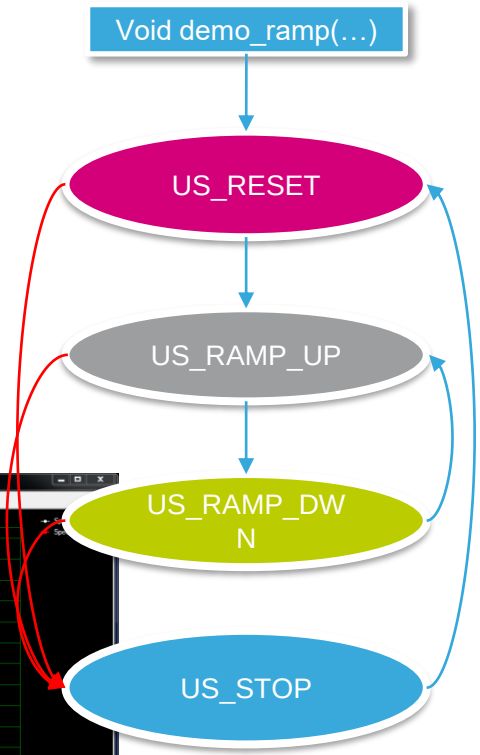
HW setup :
NUCLEO-F303RE
X-NUCLEO-IHM07M1
Bull Running motor



```
/* Includes ------------------------------------------------------------------*/
#include "saw_speed_ramp.h"

/* Private typedef -----------------------------------------------------------*/

/* Private defines -----------------------------------------------------------*/
/********************** USER STATE MACHINE DEFINITION ***********************/
#define US_RESET        0x00    /* Start program with 1st spee
#define US_RAMP_UP      0x01    /* 1st up-speed ramp generatio
#define US_RAMP_DWN     0x02    /* 2nd dwn-speed ramp generati
#define US_STOP         0x03    /* Prepare state machine to re

#define COUNT_MAX_SEC  5
#define STOP_DURATION_SEC 3
#define COUNT_MAX (COUNT_MAX_SEC * USER_TIMEBASE_FREQUENCY_HZ)
#define STOP_DURATION  (STOP_DURATION_SEC * USER_TIMEBASE_FREQ
#define USER_TIMEBASE_FREQUENCY_HZ        10
#define USER_TIMEBASE_OCCURENCE_TICKS  ((SYS_TICK_FREQUENCY / U

/* Global variables ----------------------------------------------------------*/
static uint8_t User_State = US_RESET;                 /* Demo
static uint16_t UserCnt = 0;                          /* Chro
int16_t value_Speed_RPM = 0;                          /* Mea

uint16_t speed_max_valueRPM = MOTOR_MAX_SPEED_RPM;    /* Max
uint16_t speed_first_valueRPM = 7500;                 /* Set
uint16_t speed_firstramp_duration = 1000;             /* Se
uint16_t speed_second_valueRPM = 2500;                /* Set
uint16_t speed_secondramp_duration = 1500;            /* Se

/* Private macros ------------------------------------------------------------*/

/* Function prototypes -------------------------------------------------------*/

/* Function ------------------------------------------------------------------*/
/* Main demonstration function used from the main.c */
```

```
void demo_ramp()
{
  /* Time for user program to be executed */
  HAL_Delay( USER_TIMEBASE_OCCURENCE_TICKS );

  /* Get the motor state machine */
  State_t MState = MC_GetSTMStateMotor1();

  /* User defined code */
  switch (User_State)
  {
    case US_RESET:
    {
      /* Depending on the state machi
      if (MState == IDLE)
      {
        /* Do 1st linear speed ramp
        MC_ProgramSpeedRampMotor1((sp
      }

      if (MState == RUN)
      {
        /* Execute 1st linear up-spe
        User_State = US_RAMP_UP;
      }

      if ( (MState == FAULT_NOW) ||
           (MState == FAULT_OVER)  )
      {
        /* Stop motor in case of tro
        MC_StopMotor1();
        User_State = US_STOP;
      }

      /* Reset Chronometer */
      UserCnt = 0;
    }
    break;
```






life.augmented

# Potentiometer – project example



How to write an application layer based on STM32 MC FW library

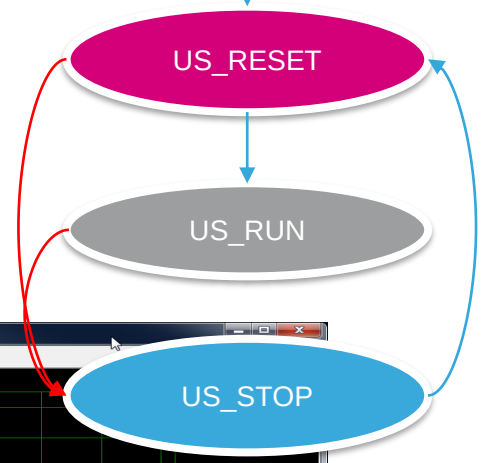
## Potentiometer

HW setup :
NUCLEO-F302R8
X-NUCLEO-IHM07M1
Bull Running motor

```
/* Includes ------------------------------------------------------------------*/
#include "Timebase.h"
#include "mc_api.h"
#include "regular_conversion_manager.h"
#include "parameters_conversion.h"

/* Private typedef -----------------------------------------------------------*/

/* Private defines -----------------------------------------------------------*/
/********************** DEFINE for USER STATE MACHINE  ************
#define US_RESET        0x00    /* Start program with speed ramp */
#define US_RUN          0x01    /* speed control */
#define US_STOP         0x02    /* Prepare state machine to restart */

#define COUNT_MAX_SEC  3
#define STOP_DURATION_SEC 2
#define COUNT_MAX (COUNT_MAX_SEC * USER_TIMEBASE_FREQUENCY_HZ)
#define STOP_DURATION  (STOP_DURATION_SEC * USER_TIMEBASE_FREQUENCY_HZ
#define USER_TIMEBASE_FREQUENCY_HZ      10
#define USER_TIMEBASE_OCCURENCE_TICKS  (SYS_TICK_FREQUENCY/USER_TIMEBA

/* Global variables ----------------------------------------------------------*/

static uint8_t User_State = US_RESET;                   /* Demonstrati
static uint16_t UserCnt = 0;                            /* Chronometer
uint16_t potentiometer_value = 0;                       /* Default pot

uint16_t speed_max_valueRPM = MOTOR_MAX_SPEED_RPM;      /* Maximum spe
uint16_t speed_min_valueRPM = 1000;                     /* Set the min
uint16_t speed_ramp_duration = 500;                     /* Set the dur

/* Private macros ------------------------------------------------------------*/

/* Function prototypes -------------------------------------------------------*/

/* Function ------------------------------------------------------------------*/
```

```
/* Main demonstration function used from the main.c */
void demo_potentiometer(uint8_t handle)
{
  /* Time for user program to be executed */
  HAL_Delay( USER_TIMEBASE_OCCURENCE_TICKS );
  /* Get the motor state machine */
   State_t MState = MC_GetSTMStateMotor1();

  /* User defined code */
  switch (User_State)
  {
    case US_RESET:
    {
      /* Depending on the state machin
      if (MState == IDLE)
      {
        /* Do linear speed ramp pre-co
        MC_ProgramSpeedRampMotor1((spe
      }

      if (MState == RUN)
      {
        /* Motor is ready for demo */
        User_State = US_RUN;
      }

      if ( (MState == FAULT_NOW) ||
           (MState == FAULT_OVER)  )
      {
        /* Stop motor in case of troub
        MC_StopMotor1();
        User_State = US_STOP;
      }

      /* Reset Chronometer */
      UserCnt = 0;
    }
    break;
```

Void demo_potentiometer(uint8_t handle)





life.augmented

# Survey

- **Now you have  5 minute for fill the SurveyMonkey**

https://www.surveymonkey.com/r/motrocontrol

**Do not worry, it is about 10 questions….☺ It is valuable feedback for us**

**Thank you!**

# Further Tips & Tricks



- You can find additional Tips & Tricks in F.A.Q. located on ST website!

# Thank you