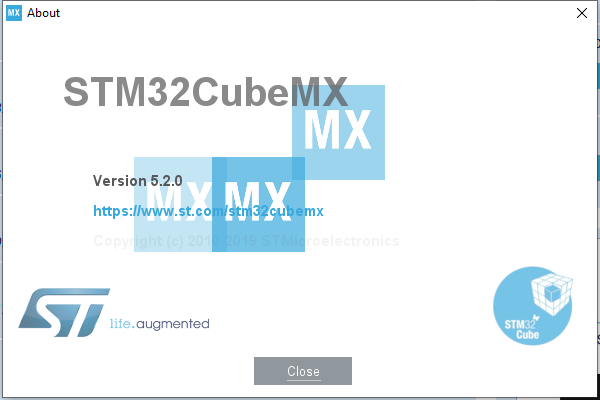
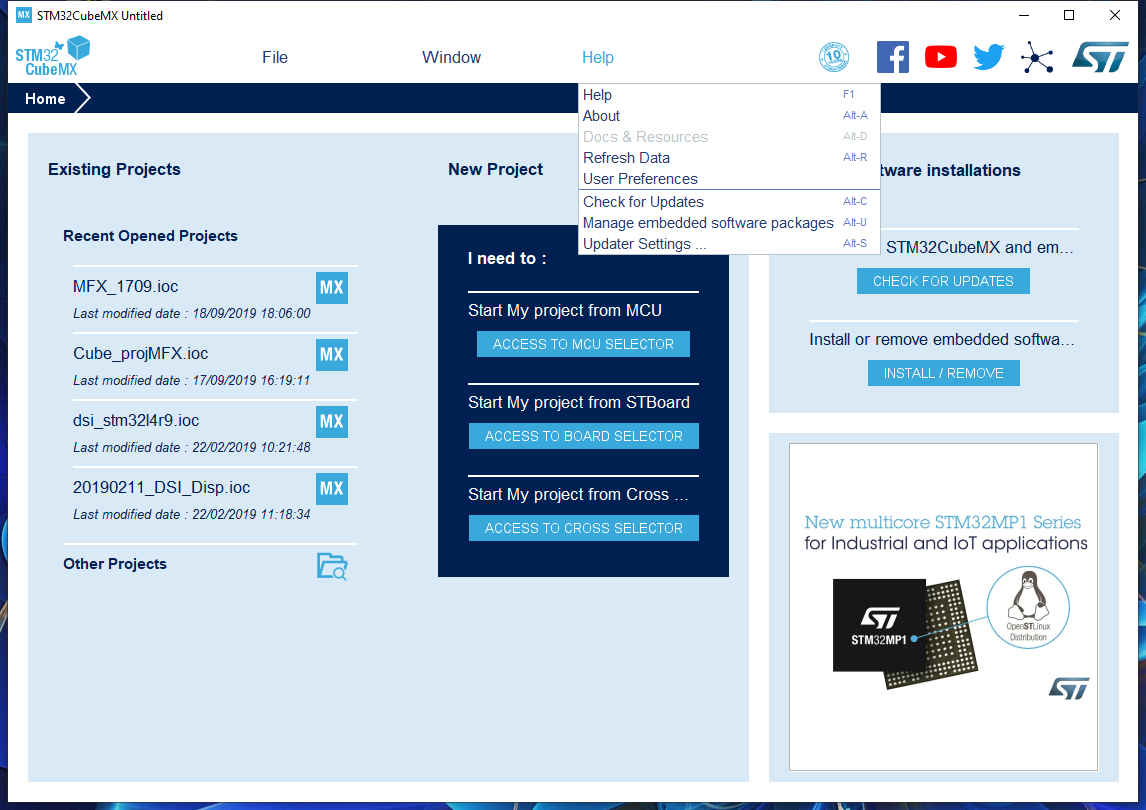
Генерация проекта осуществляется через CubeMX. Там необходимо установить расширение MEMS1. Для этого нужно:

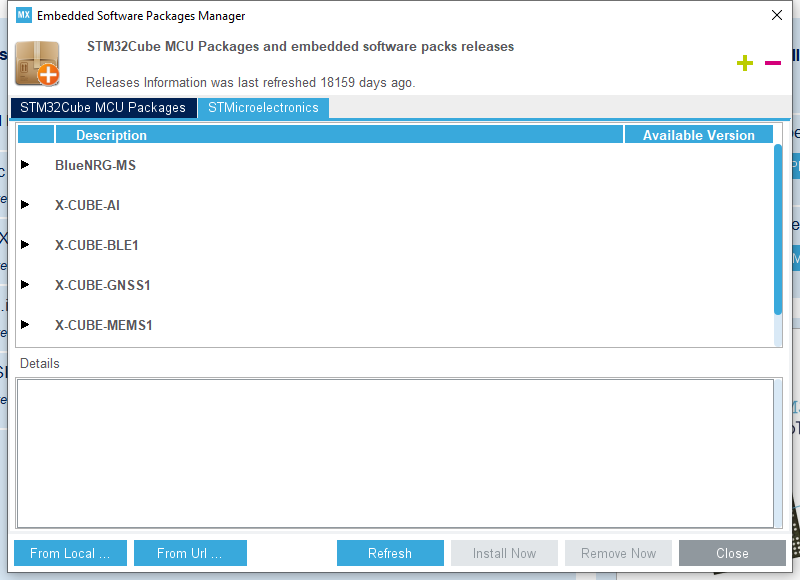
1. Убедиться, что у вас установлена версия CubeMX 5.2.0 или выше.



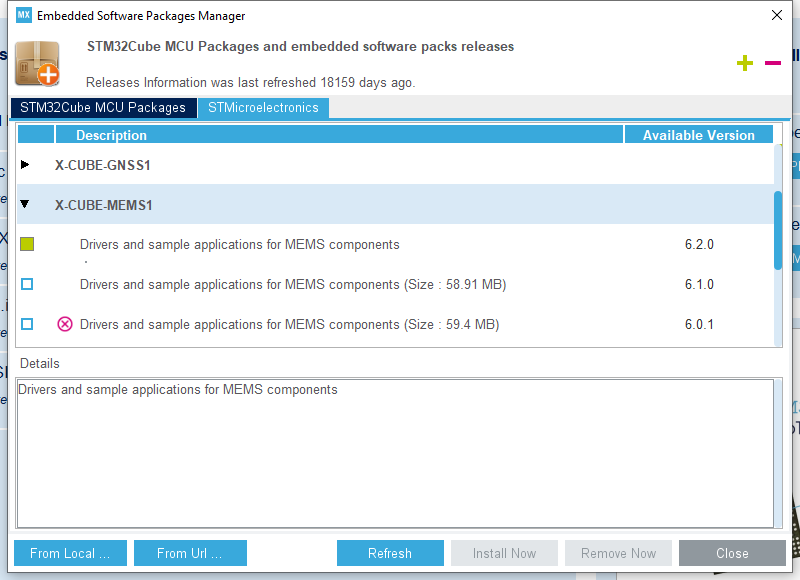
1. Во вкладке Help выбрать пункт Manage embedded software packages.



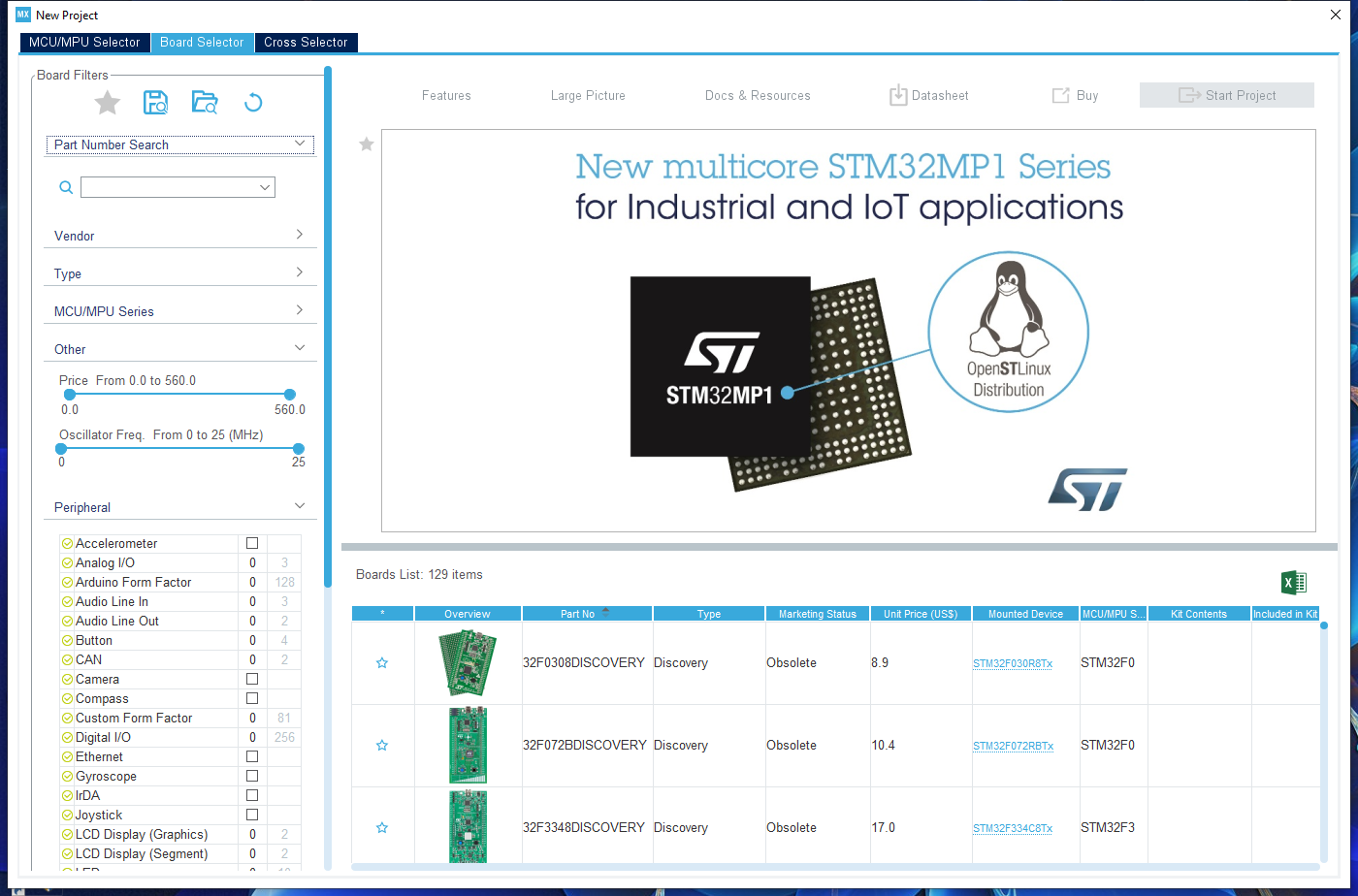
1. Там перейти во вкладку STMicroelectronics



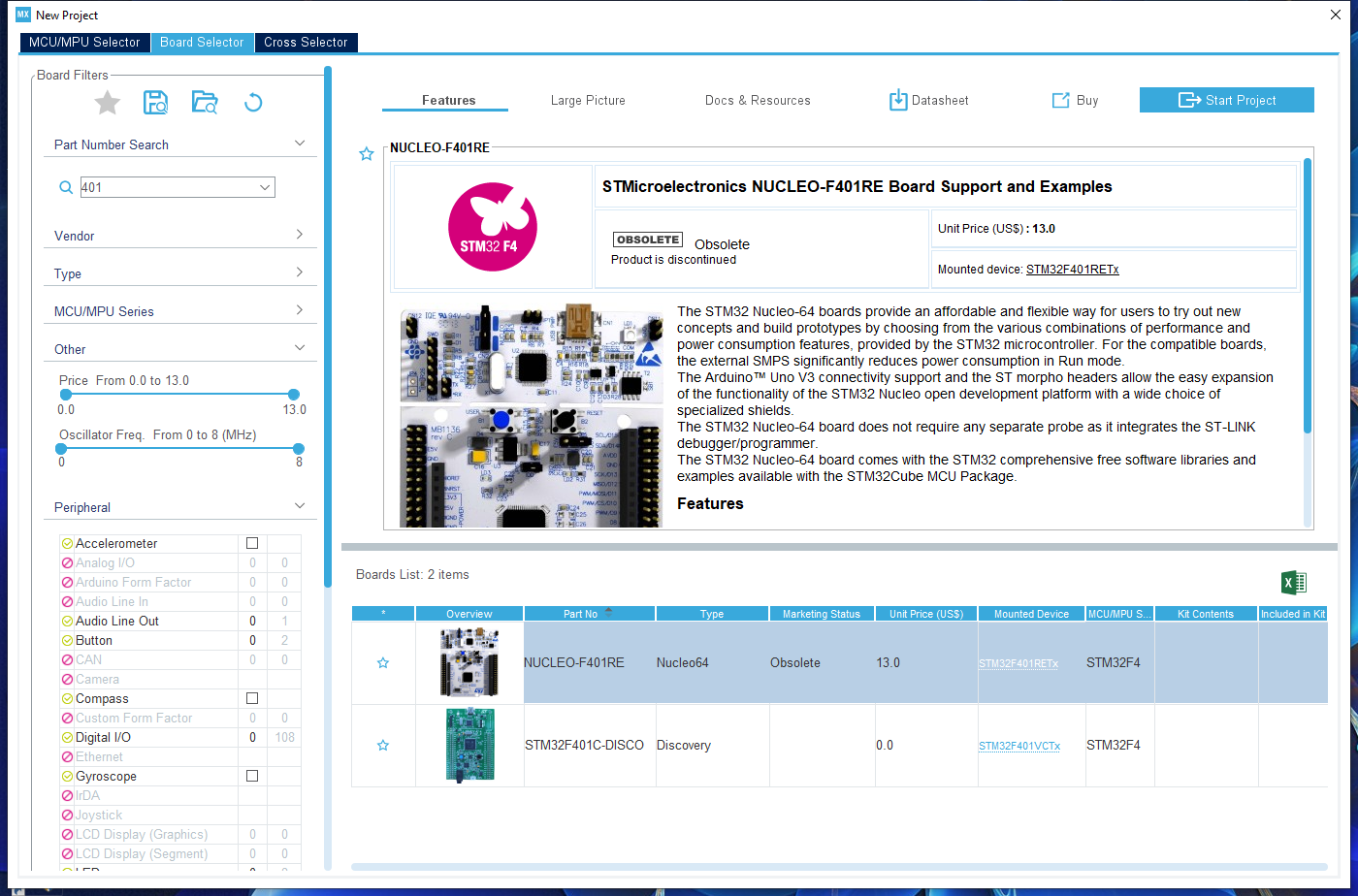
1. Выбрать пунктX-CUBE-MEMS1 и установить новейшую версию фреймворка.



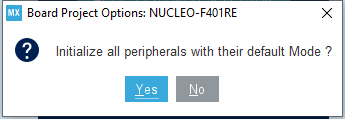
1. После установки обновлений лучше перезапустить программу и перезагрузить компьютер. Далее выбрать создание нового проекта через выбор платы



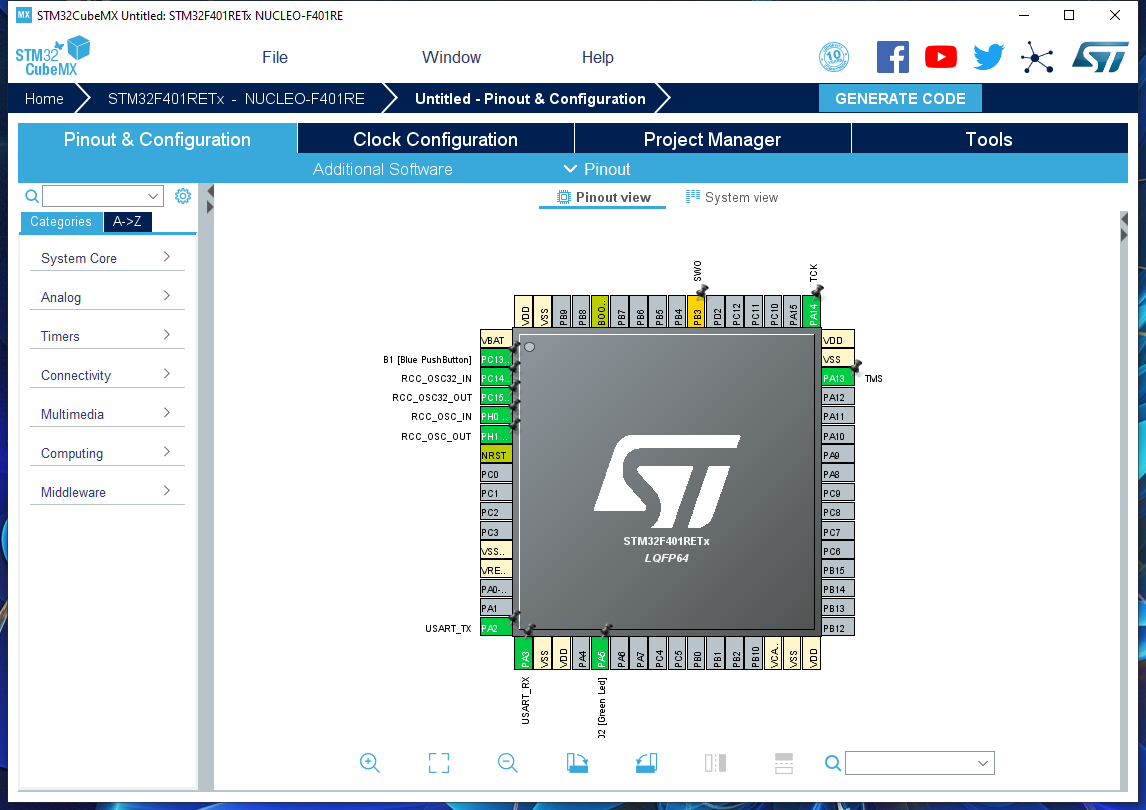
1. Здесь выбираем нашу плату



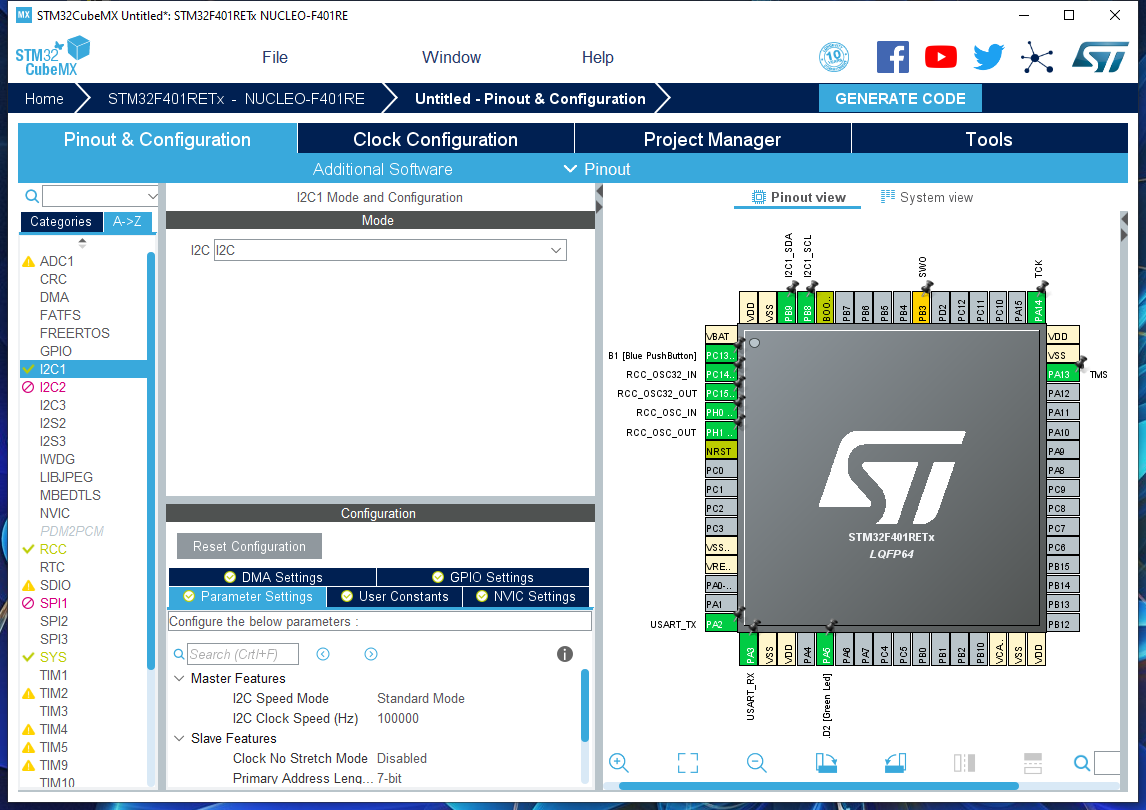
1. Говорим ДА



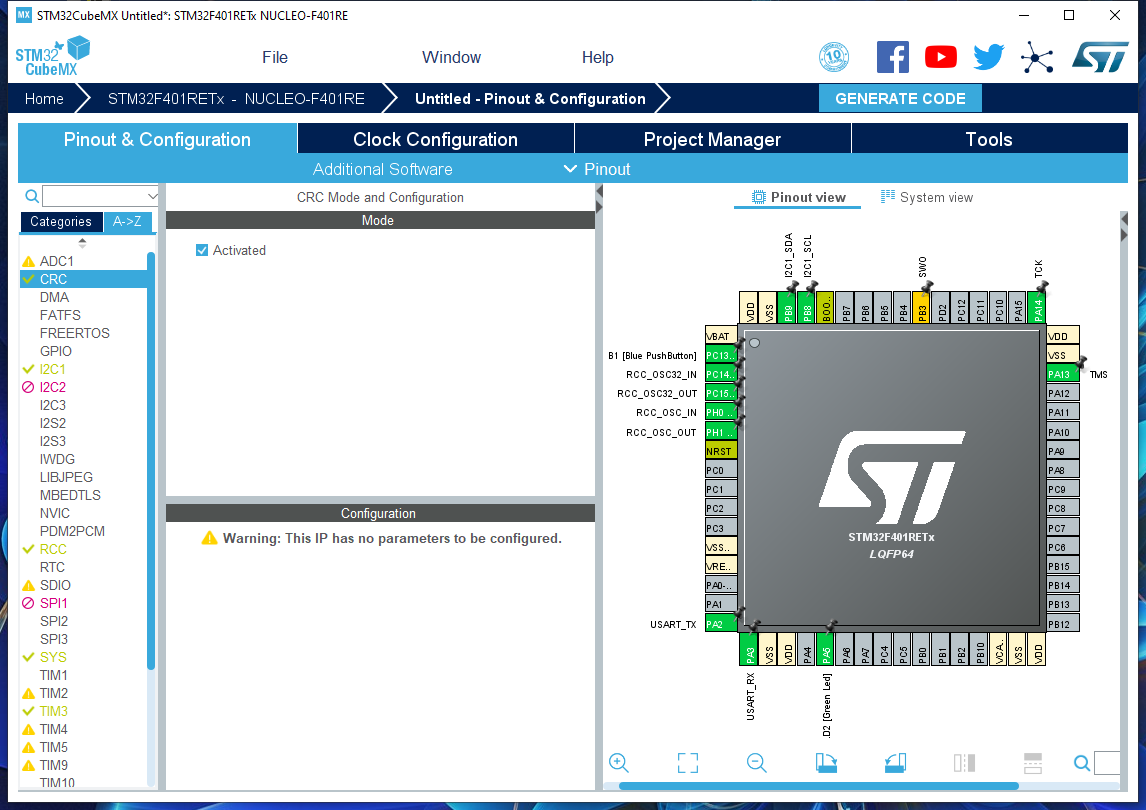
Открывается вот такое окно



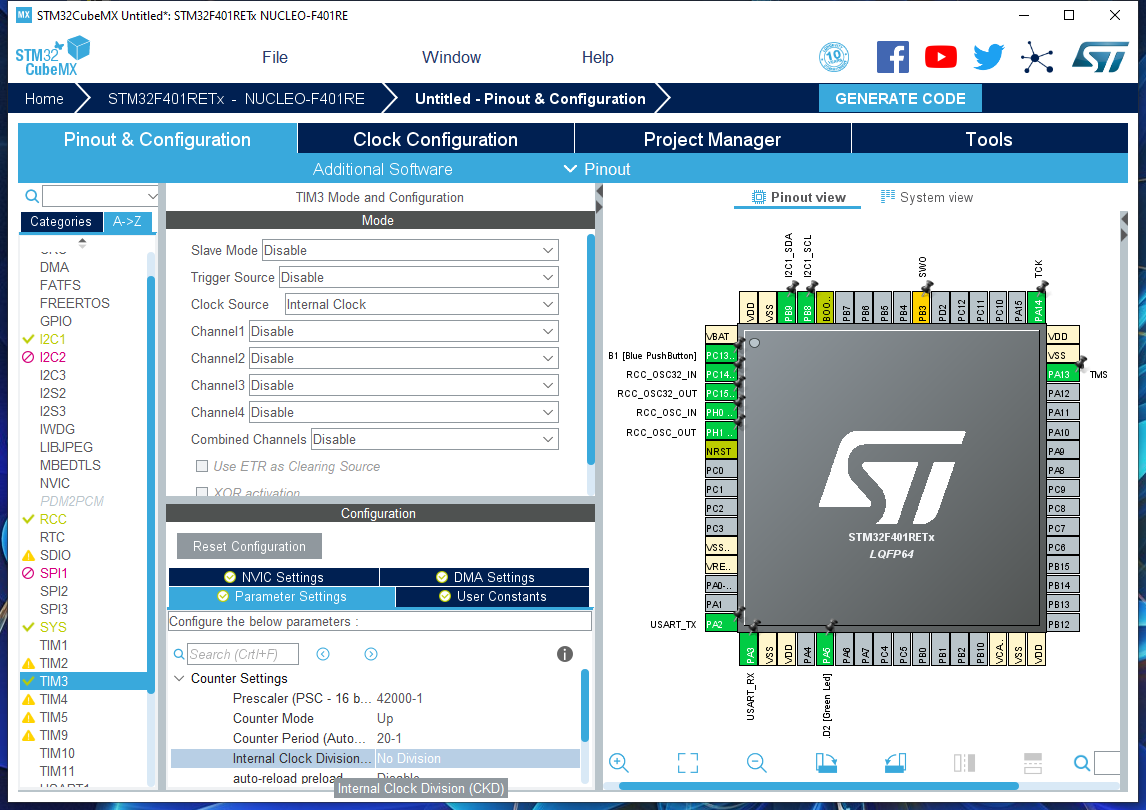
1. Слева выбираем необходимый протокол передачи и, при выборе i2c1 необходимо перенести ножки на pb8 и pb9, так как на плате i2c веден на них



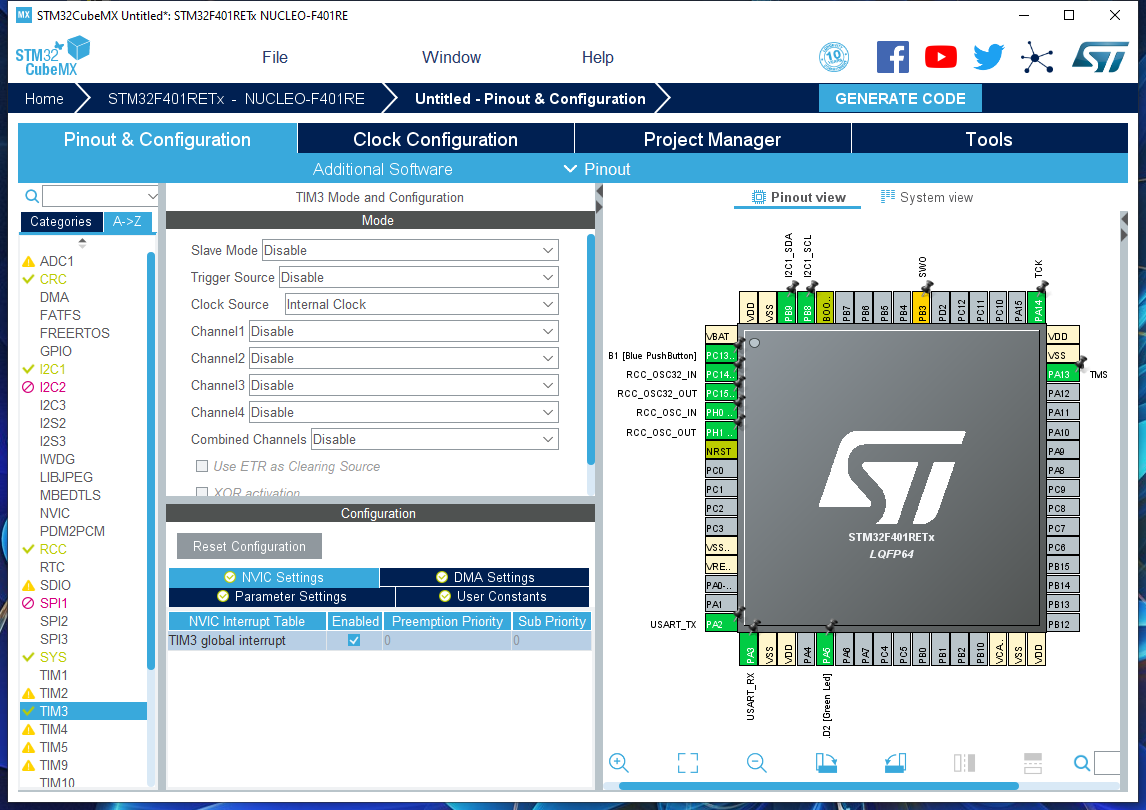
1. Для работы алгоритмов калибровки обязательно должен быть инициализирован CRC и таймер.



1. Настройки таймера необходимо установить как показано на картинке

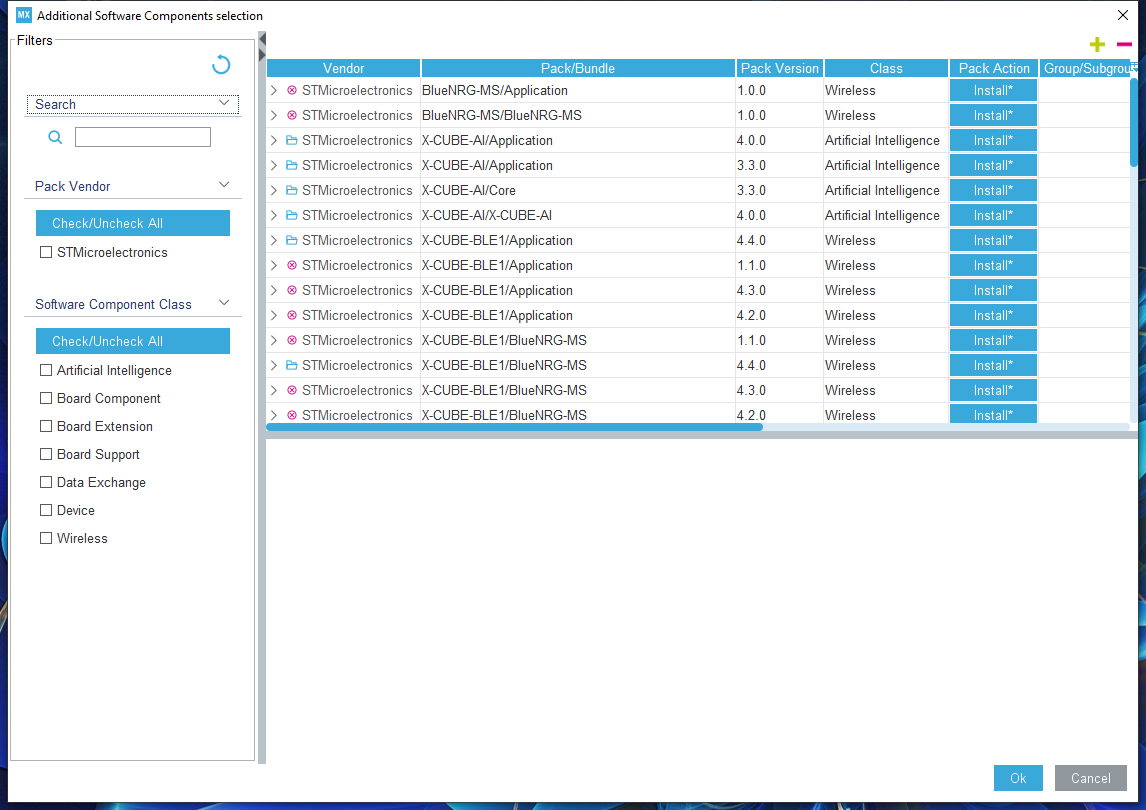


1. И включить прерывание таймера

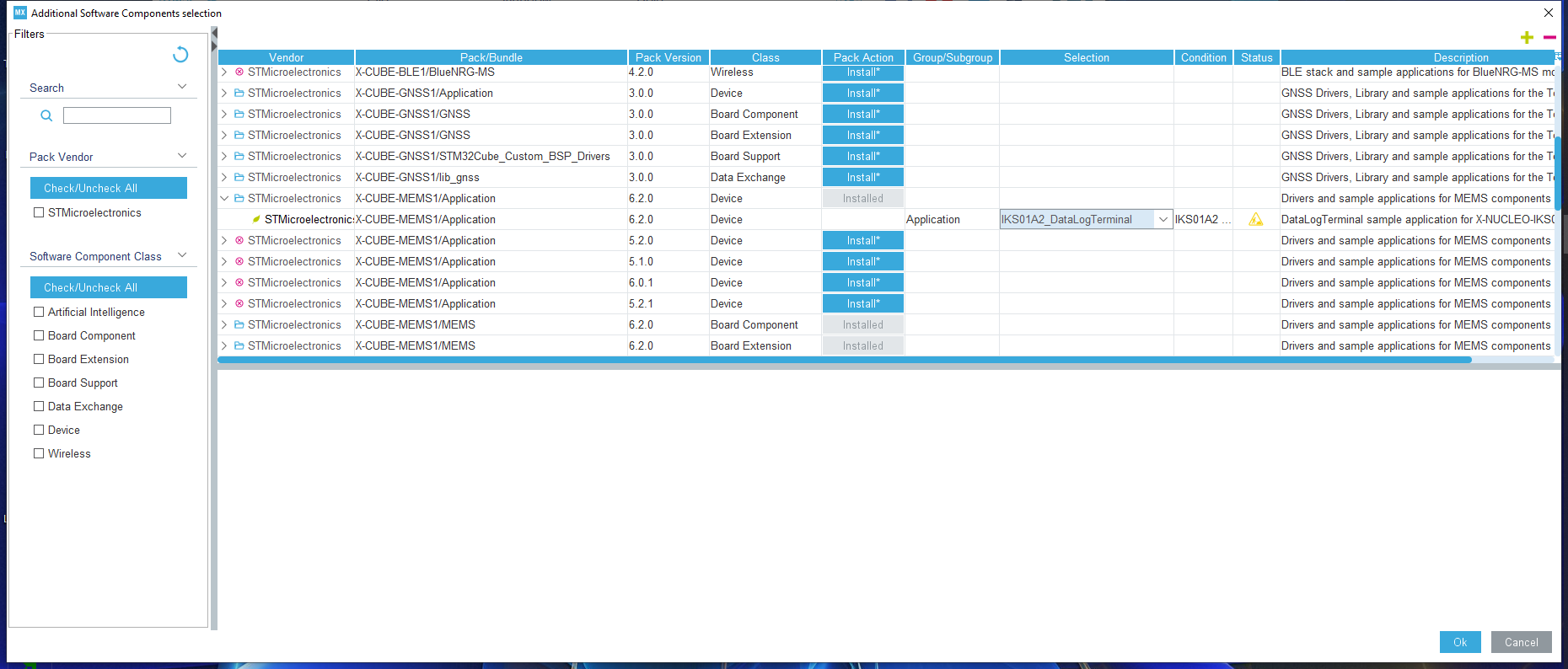


Это обязательные настройки, после этого можно добавить другую периферию, если это необходимо.

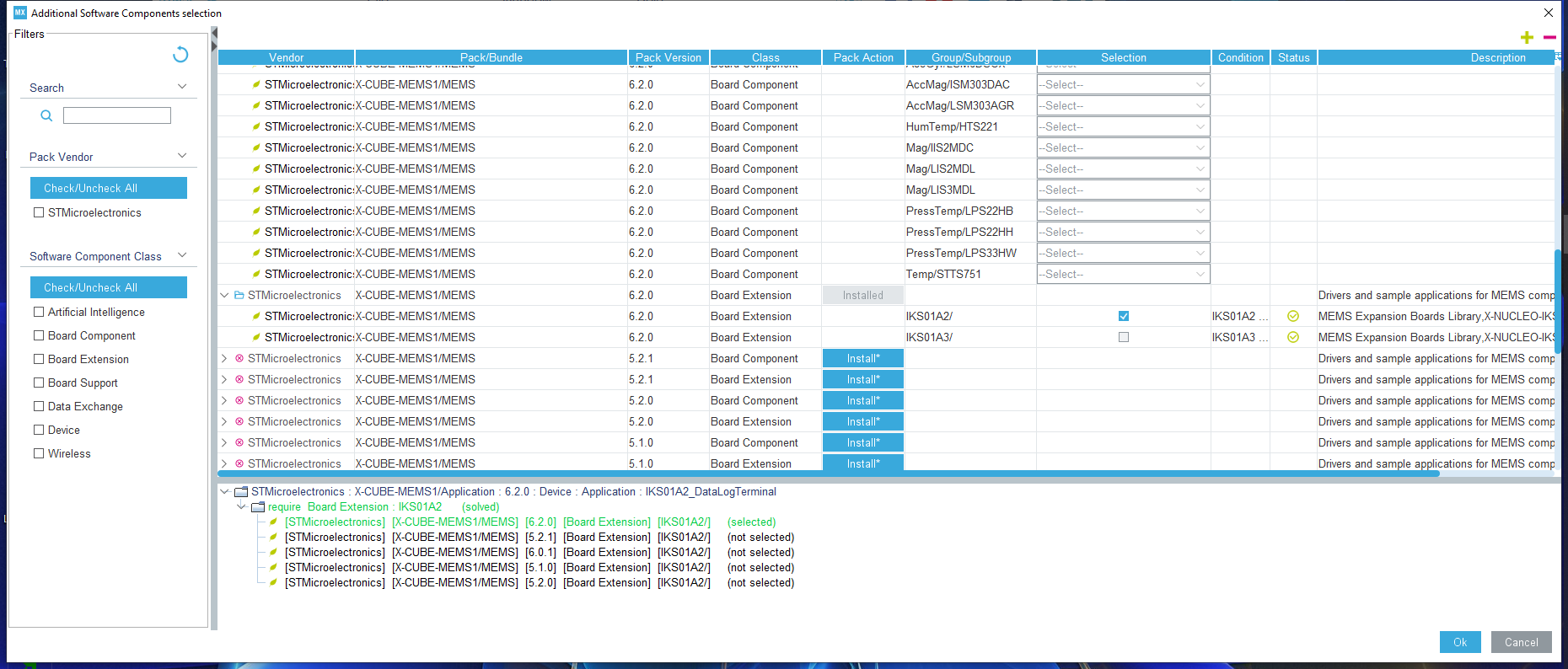
1. Вверху окна находится вкладка **Additional Software.** Переходим туда. Должно открыться вот такое окно



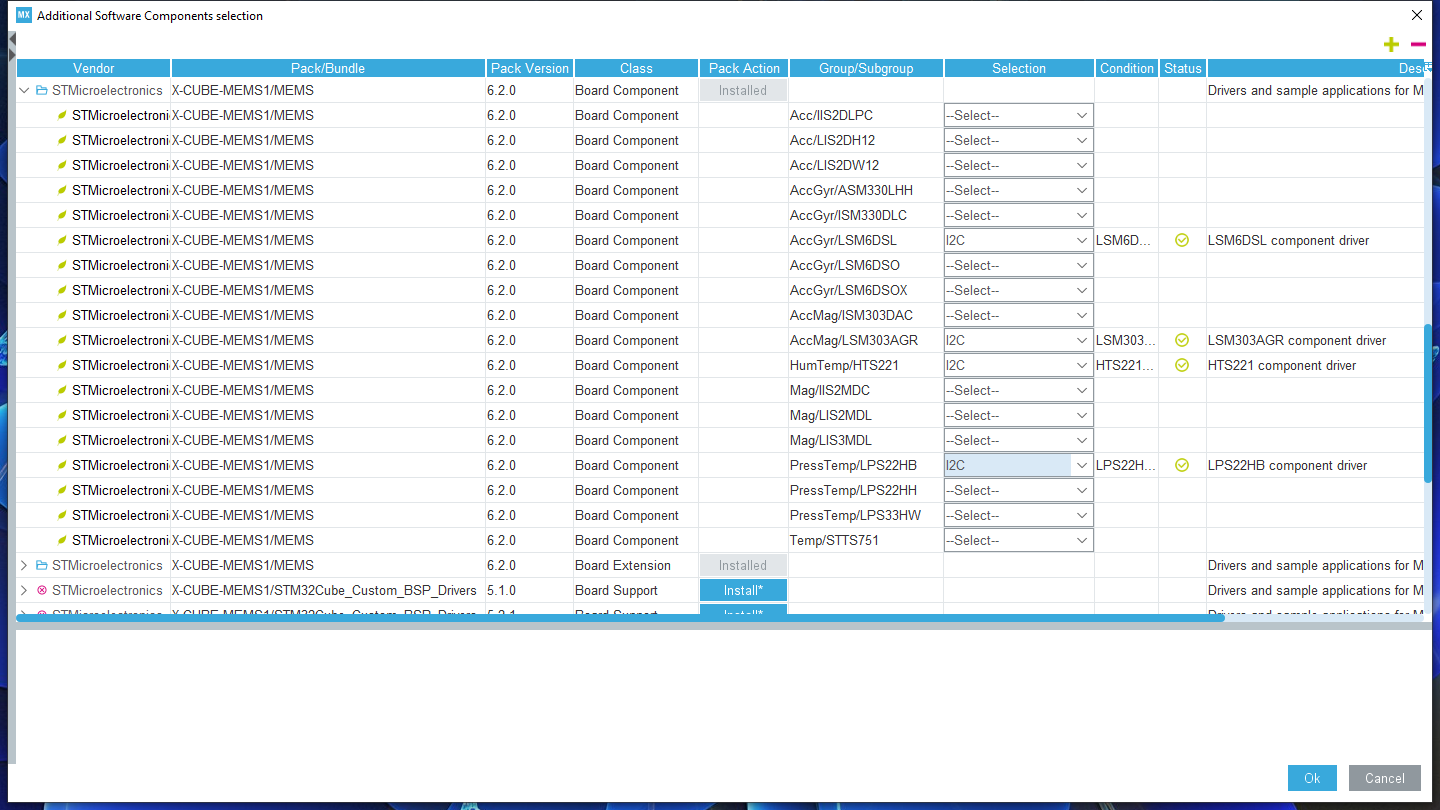
1. Находим нужный пункт и выбираем приложение, на основе которого будет создан проект. Это IKS01A2\_ DataLogTerminal.



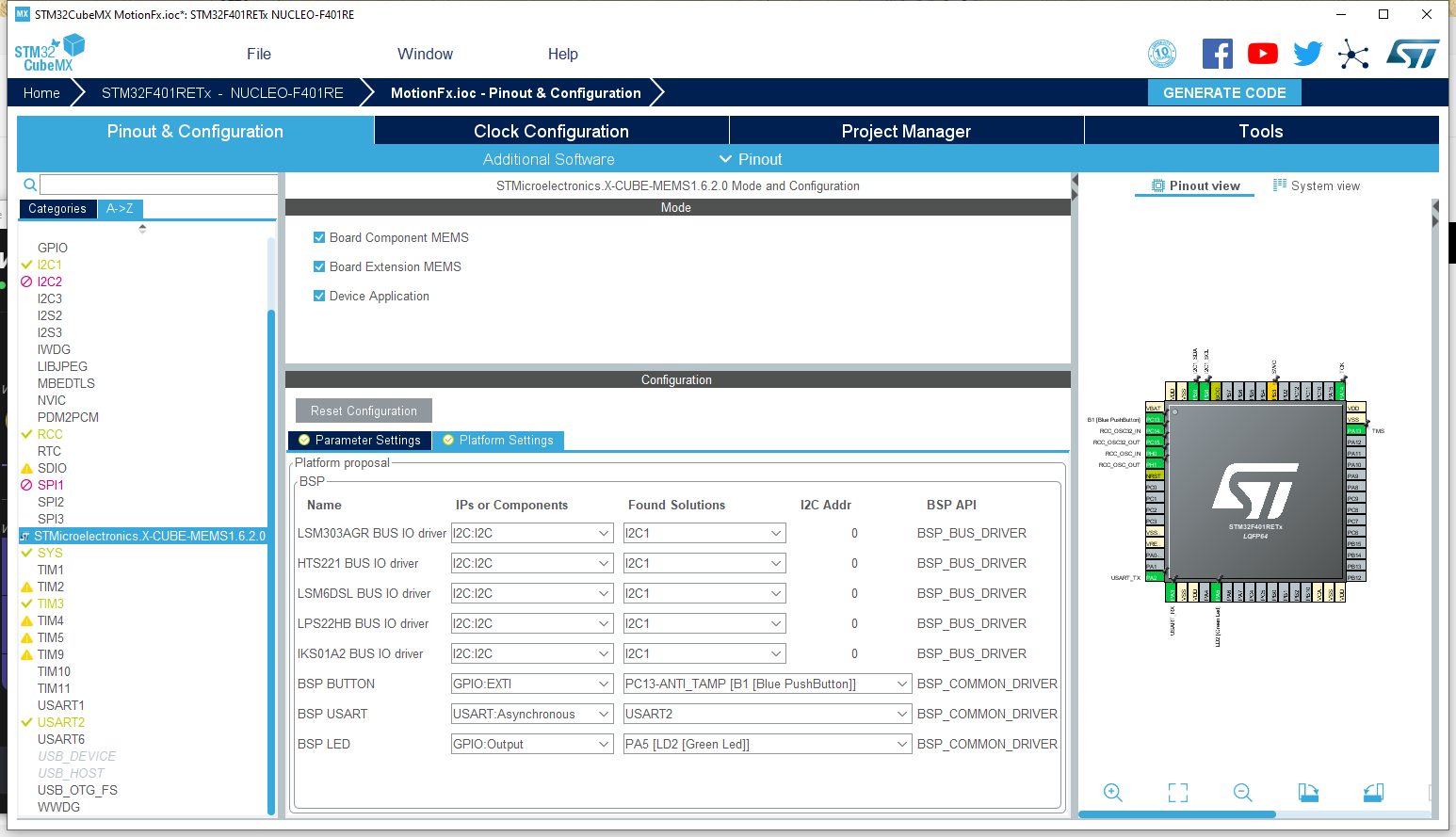
1. И, листая чуть ниже, ставим галочку напротив выбора платы расширений – IKS01A2



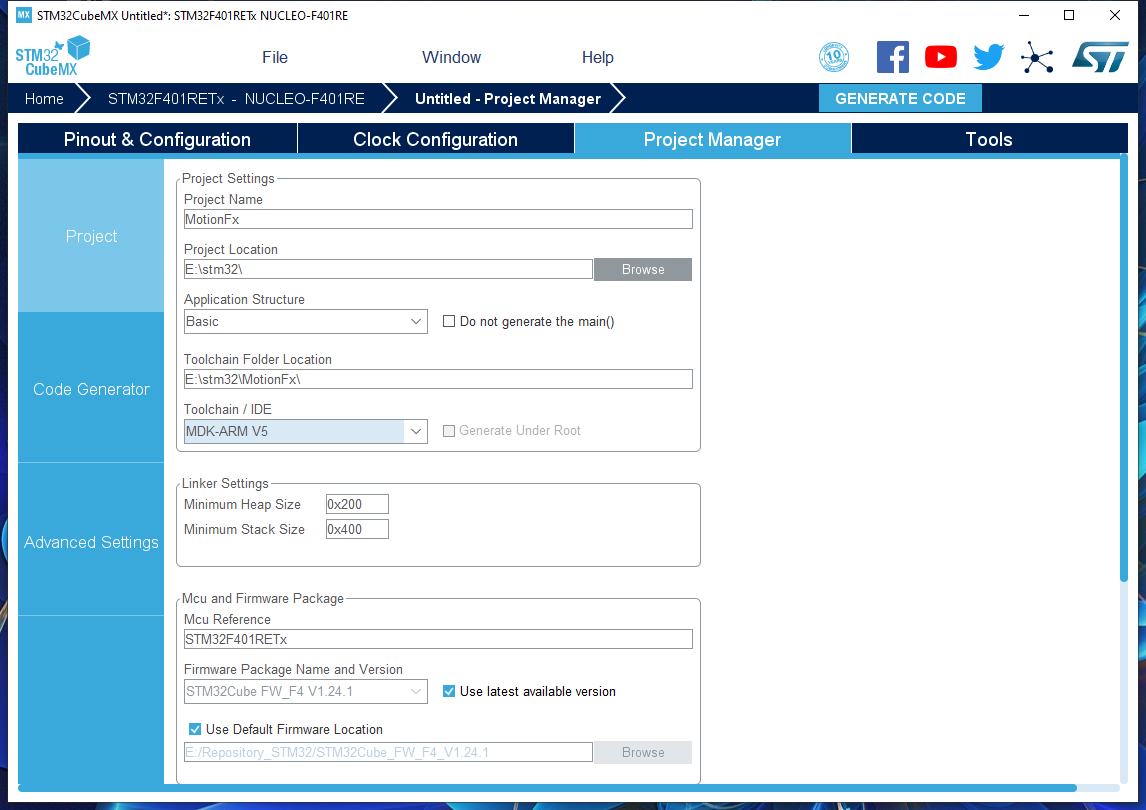
1. Листаем, находим класс **Board Component**, там выбираем используемые датчики и протокол обмена – AccGyr/LSM6DSL, AccMag/LSM303AGR, HumTemp/HTS221, PressTemp/LPS22HB.



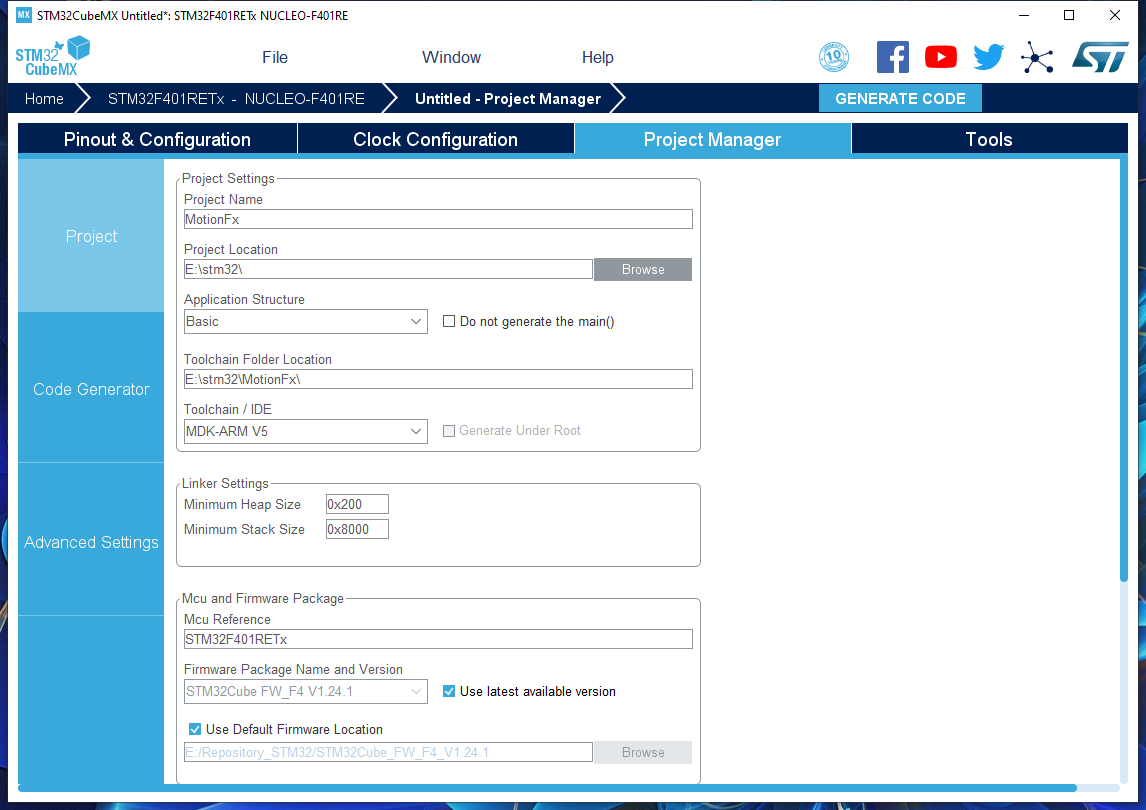
1. Ставим OK, выходим из этого меню. Слева в окне у нас появилась эта библиотека. Нажимаем на нее и в открывшемся окне вверху ставим 3 галочки, а в нижней части Platform Proposal напротив каждого параметра выбираем используемый протокол I2C и, если используется, UART.



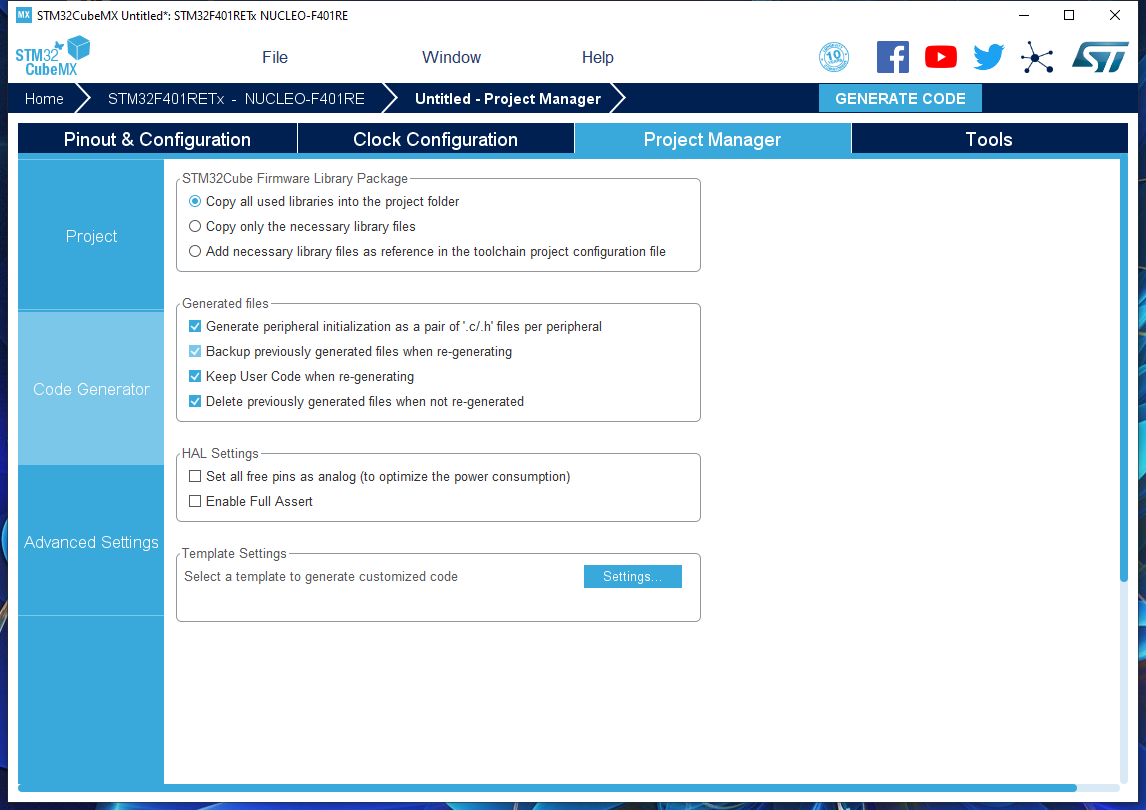
1. Переходим во вкладку **Project Manager** в верхней строке меню. Здесь задается имя проекта, его расположение и под какую IDE он будет сконфигурирован



1. Здесь же, очень важный пункт – нужно указать размер стека не менее чем 0х8000, без этого алгоритм калибровки работать не будет!



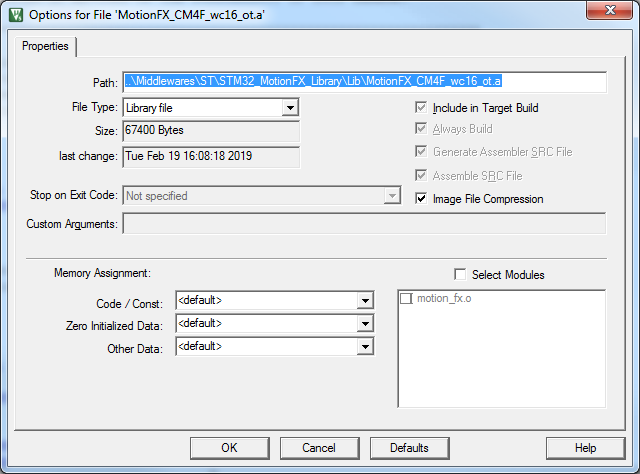
1. Слева следующий пункт меню **Code Generator**. Выбираем **Copy all used libraries into the project folder** и, ниже, ставим галочки



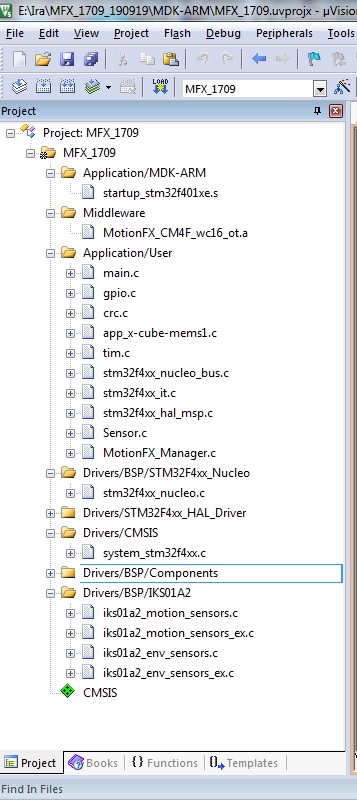
1. Нажимаем кнопу Generate code.

Теперь необходимо подключить библиотеку MotionFX и настроить проект. Для этого необходимо:

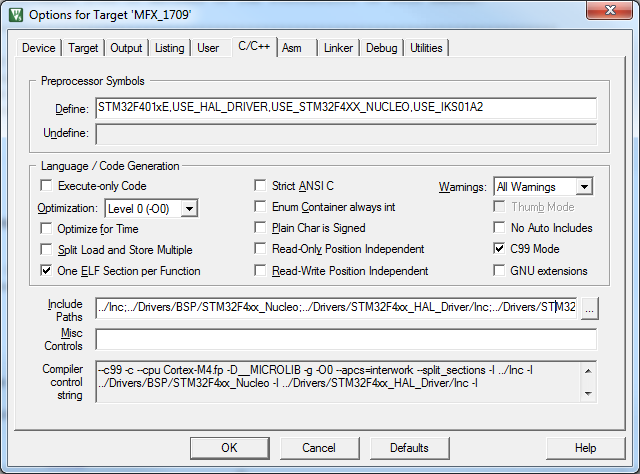
1. Скачать на сайте stm <https://www.st.com/en/embedded-software/x-cube-mems1.html#overview> папку с софтом.
2. Распаковать и скопировать Middlewares/ST/STM32\_MotionFX\_Library в папку проекта.
3. Необязательно, но облегчит работу: скопировать файлы MotionFX\_Manager.c и MotionFX\_Manager.h из папок Projects/STM32F401RE-Nucleo/Applications/IKS01A2/DataLogFusion/Inc и Src в папки Inc и Src своего проекта, а также наши файлы Sensor.h и Sensor.c.
4. Открыть проект в Keil.
5. Добавить в проект пустую папку (слева в дереве проекта нажать правой кнопкой мыши на названии проекта, в выпадающем меню выбрать пункт **Add Group…**)
6. В нее добавить файл библиотеки (в выпадающем меню выбрать пункт **Add Existing files to the group…** ) из Middlewares\ST\STM32\_MotionFX\_Library\Lib.
7. Нажать на появившийся файл в дереве проекта правой кнопкой мыши и в открывшемся окне выбрать первый пункт – **Options for File…** В графе **File** **Type** выбрать Library file из списка.



1. В папку Application/User в дереве проекта добавить файлы MotionFX\_Manager.c и Sensor.c, которые были скопированы ранее.



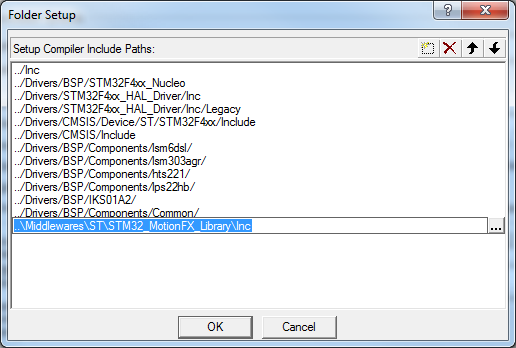
Примерно так должно выглядеть дерево проекта на данный момент.

1. Открыть Свойства проекта ( правой кнопкой мыши на имени проекта в дереве, в выпадающем меню пункт **Options for Target**…), перейти на вкладку **C/C++**, в графу **Define** добавить директивы, как указано на картинке. 
2. Далее, внизу напротив графы **Include Paths** нажать справа на значок с многоточием. В открывающемся окне нажать на кнопку с изображением квадрата и внизу списка появится пустая графа с многоточием. Необходимо нажать на кнопку с многоточием и выбрать:

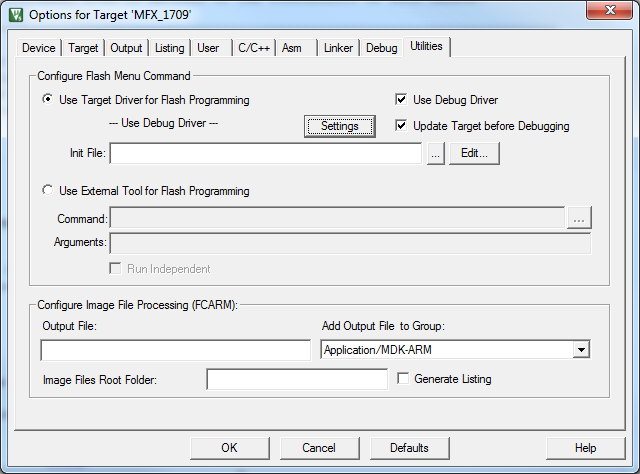
- папку Middlewares\ST\STM32\_MotionFX\_Library\Inc,

- Drivers\STM32F4xx-Nucleo,

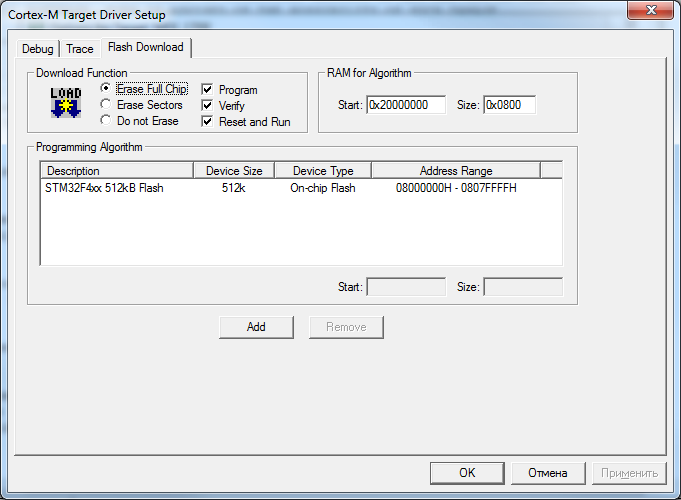
- После чего окошко должно выглядеть так:



1. Нажимаем OK, переходим на вкладку **Utilities**. Вверху область под названием **Configure Flash Menu Command**, на ней кнопка **Settings**.



Нажимаем ее, в открывшемся окне на вкладке **Flash Download** выбираем самый первый пункт **Erase Full Chip**. (Потому что при прошивке необходимо стереть и записать чип полностью)



1. Нажимаем OK, сохраняем настройки.

Далее переходим к правке кода.

В файле main.c, после инициализации всей периферии, в бесконечном цикле находится вызов функции MX\_MEMS\_Process(), в которой происходит чтение показаний датчиков и отправка сообщений по юарту.

while (1)

{

/\* USER CODE END WHILE \*/

MX\_MEMS\_Process();

/\* USER CODE BEGIN 3 \*/

}

Эта функция не подходит для работы с библиотекой MotionFX. Вместо этого удобнее закомментировать или удалить этот код, а вместо него добавить следующие функции:

**lib\_init(&htim3);**

**Lib\_Process();**

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

// while (1)

// {

// /\* USER CODE END WHILE \*/

// MX\_MEMS\_Process();

// /\* USER CODE BEGIN 3 \*/

// }

Эти функции находятся в наших файлах Sensor.h и Sensor.c. В функции lib\_init происходят вызовы функций для старта работы функций библиотеки MotionFX, а также начинает работу таймер, который необходим для правильной калибровки.

**void lib\_init(TIM\_HandleTypeDef\* AlgoTimHandle)**

**{**

/\* Sensor Fusion API initialization function \*/

MotionFX\_manager\_init();

/\* OPTIONAL \*/

/\* Get library version \*/

MotionFX\_manager\_get\_version(lib\_version, &lib\_version\_len);

/\* Enable magnetometer calibration \*/

MotionFX\_manager\_MagCal\_start(ALGO\_PERIOD);

/\*Start timer \*/

HAL\_TIM\_Base\_Start\_IT(AlgoTimHandle);

/\*Start fusion calibration \*/

MotionFX\_manager\_start\_9X();

**}**

Функция **Lib\_Process(void)** содержит в себе бесконечный цикл, что позволило нам убрать его из функции main(), где, с частотой срабатывания таймера происходит опрос датчиков, самостоятельная калибровка магнитометра и калибровка с учетом всех трех датчиков.

**void Lib\_Process(void)**

**{**

for (;;)

{

if (SensorReadRequest == 1U)

{

SensorReadRequest = 0;

Accelero\_Sensor\_Handler(IKS01A2\_LSM6DSL\_0);

Gyro\_Sensor\_Handler(IKS01A2\_LSM6DSL\_0);

Magneto\_Sensor\_Handler(IKS01A2\_LSM303AGR\_MAG\_0);

/\* Sensor Fusion specific part \*/

FX\_Data\_Handler();

}

}

**}**

Переменная SensorReadRequest устанавливается в функции обработки прерывания таймера.

**void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)**

**{**

if (htim->Instance == TIM3)

{

Lib\_ReadRequest();

}

**}**

**void Lib\_ReadRequest()**

**{**

SensorReadRequest = 1;

**}**

В функциях Accelero\_Sensor\_Handler и Gyro\_Sensor\_Handler происходит только чтение показаний датчиков, путем вызова библиотечной функции из файла iks01a2\_motion\_sensors.h

**void Accelero\_Sensor\_Handler(uint32\_t Instance)**

**{**

(void)IKS01A2\_MOTION\_SENSOR\_GetAxes(Instance, MOTION\_ACCELERO, &AccValue);

**}**

**void Gyro\_Sensor\_Handler(uint32\_t Instance)**

**{**

(void)IKS01A2\_MOTION\_SENSOR\_GetAxes(Instance, MOTION\_GYRO, &GyrValue);

**}**

Функция Magneto\_Sensor\_Handler включает в себя калибровку магнитометра и применение калибровочных значений по указанной в документации инструкции (документ Getting started with MotionFX sensor fusion library in X-CUDE-MEMS1 expansion for STM32Cube).

**void Magneto\_Sensor\_Handler(uint32\_t Instance)**

**{**

float ans\_float;

#if ((defined (USE\_STM32F4XX\_NUCLEO)) || (defined (USE\_STM32L4XX\_NUCLEO)) || (defined (USE\_STM32L1XX\_NUCLEO)))

MFX\_MagCal\_input\_t mag\_data\_in;

MFX\_MagCal\_output\_t mag\_data\_out;

#elif (defined (USE\_STM32L0XX\_NUCLEO))

MFX\_CM0P\_MagCal\_input\_t mag\_data\_in;

MFX\_CM0P\_MagCal\_output\_t mag\_data\_out;

#else

#error Not supported platform

#endif

(void)IKS01A2\_MOTION\_SENSOR\_GetAxes(Instance, MOTION\_MAGNETO, &MagValue);

if (MagCalStatus == 0U)

{

mag\_data\_in.mag[0] = (float)MagValue.x \* FROM\_MGAUSS\_TO\_UT50;

mag\_data\_in.mag[1] = (float)MagValue.y \* FROM\_MGAUSS\_TO\_UT50;

mag\_data\_in.mag[2] = (float)MagValue.z \* FROM\_MGAUSS\_TO\_UT50;

#if ((defined (USE\_STM32F4XX\_NUCLEO)) || (defined (USE\_STM32L4XX\_NUCLEO)) || (defined (USE\_STM32L1XX\_NUCLEO)))

mag\_data\_in.time\_stamp = (int)MagTimeStamp;

MagTimeStamp += (uint32\_t)ALGO\_PERIOD;

#endif

MotionFX\_manager\_MagCal\_run(&mag\_data\_in, &mag\_data\_out);

#if ((defined (USE\_STM32F4XX\_NUCLEO)) || (defined (USE\_STM32L4XX\_NUCLEO)) || (defined (USE\_STM32L1XX\_NUCLEO)))

if (mag\_data\_out.cal\_quality == MFX\_MAGCALGOOD)

#elif (defined (USE\_STM32L0XX\_NUCLEO))

if (mag\_data\_out.cal\_quality == MFX\_CM0P\_MAGCALGOOD)

#else

#error Not supported platform

#endif

{

MagCalStatus = 1;

ans\_float = (mag\_data\_out.hi\_bias[0] \* FROM\_UT50\_TO\_MGAUSS);

MagOffset.x = (int32\_t)ans\_float;

ans\_float = (mag\_data\_out.hi\_bias[1] \* FROM\_UT50\_TO\_MGAUSS);

MagOffset.y = (int32\_t)ans\_float;

ans\_float = (mag\_data\_out.hi\_bias[2] \* FROM\_UT50\_TO\_MGAUSS);

MagOffset.z = (int32\_t)ans\_float;

/\* Disable magnetometer calibration \*/

MotionFX\_manager\_MagCal\_stop(ALGO\_PERIOD);

}

}

MagValue.x = (int32\_t)(MagValue.x - MagOffset.x);

MagValue.y = (int32\_t)(MagValue.y - MagOffset.y);

MagValue.z = (int32\_t)(MagValue.z - MagOffset.z);

**}**

Переменная MagCalStatus объявлена в файле Sensor.c

static uint8\_t MagCalStatus = 0;

В функции FX\_Data\_Handler() происходит вся работа по калибровке значений всех трех датчиков путем преобразования всех считанных значений в необходимые единицы измерения и вызова функции **MotionFX\_manager\_run.**

**void FX\_Data\_Handler()**

**{**

#if ((defined (USE\_STM32F4XX\_NUCLEO)) || (defined (USE\_STM32L4XX\_NUCLEO)) || (defined (USE\_STM32L1XX\_NUCLEO)))

MFX\_input\_t data\_in;

MFX\_input\_t \*pdata\_in = &data\_in;

MFX\_output\_t data\_out;

MFX\_output\_t \*pdata\_out = &data\_out;

#elif (defined (USE\_STM32L0XX\_NUCLEO))

MFX\_CM0P\_input\_t data\_in;

MFX\_CM0P\_input\_t \*pdata\_in = &data\_in;

MFX\_CM0P\_output\_t data\_out;

MFX\_CM0P\_output\_t \*pdata\_out = &data\_out;

#else

#error Not supported platform

#endif

data\_in.gyro[0] = (float)GyrValue.x \* FROM\_MDPS\_TO\_DPS;

data\_in.gyro[1] = (float)GyrValue.y \* FROM\_MDPS\_TO\_DPS;

data\_in.gyro[2] = (float)GyrValue.z \* FROM\_MDPS\_TO\_DPS;

data\_in.acc[0] = (float)AccValue.x \* FROM\_MG\_TO\_G;

data\_in.acc[1] = (float)AccValue.y \* FROM\_MG\_TO\_G;

data\_in.acc[2] = (float)AccValue.z \* FROM\_MG\_TO\_G;

data\_in.mag[0] = (float)MagValue.x \* FROM\_MGAUSS\_TO\_UT50;

data\_in.mag[1] = (float)MagValue.y \* FROM\_MGAUSS\_TO\_UT50;

data\_in.mag[2] = (float)MagValue.z \* FROM\_MGAUSS\_TO\_UT50;

/\* Run Sensor Fusion algorithm \*/

BSP\_LED\_On(LED2);

MotionFX\_manager\_run(pdata\_in, pdata\_out, MOTIONFX\_ENGINE\_DELTATIME);

BSP\_LED\_Off(LED2);

**}**

Файл Sensor.h содержит необходимые для работы библиотеки

#include "stdint.h"

#include "main.h"

#include "MotionFX\_Manager.h"

#include "iks01a2\_motion\_sensors.h"

Константы

#define ALGO\_FREQ 100U /\* Algorithm frequency [Hz] \*/

#define ALGO\_PERIOD 10 /\* Algorithm period [ms] \*/

#define MOTIONFX\_ENGINE\_DELTATIME 0.01f

#define FROM\_MG\_TO\_G 0.001f

#define FROM\_G\_TO\_MG 1000.0f

#define FROM\_MDPS\_TO\_DPS 0.001f

#define FROM\_DPS\_TO\_MDPS 1000.0f

#define FROM\_MGAUSS\_TO\_UT50 (0.1f/50.0f)

#define FROM\_UT50\_TO\_MGAUSS 500.0f

И объявления указанных выше функций.

Для работы с этими файлами их нужно добавить в соответствующие папки проекта Inc и Src, после чего добавить файл Sensor.c в дерево проекта, в папку Application/User, а также добавить соответствующий инклуд в файл main.c

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

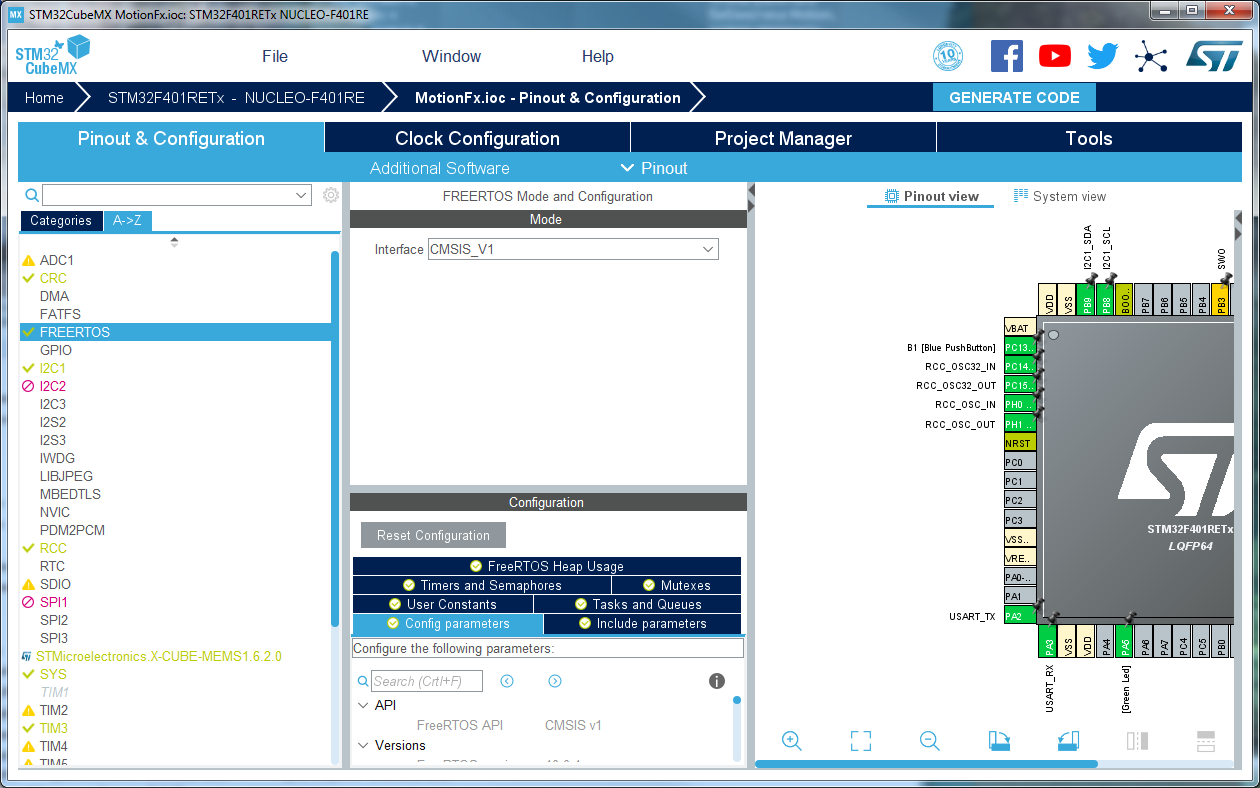
#include "Sensor.h"

/\* USER CODE END Includes \*/

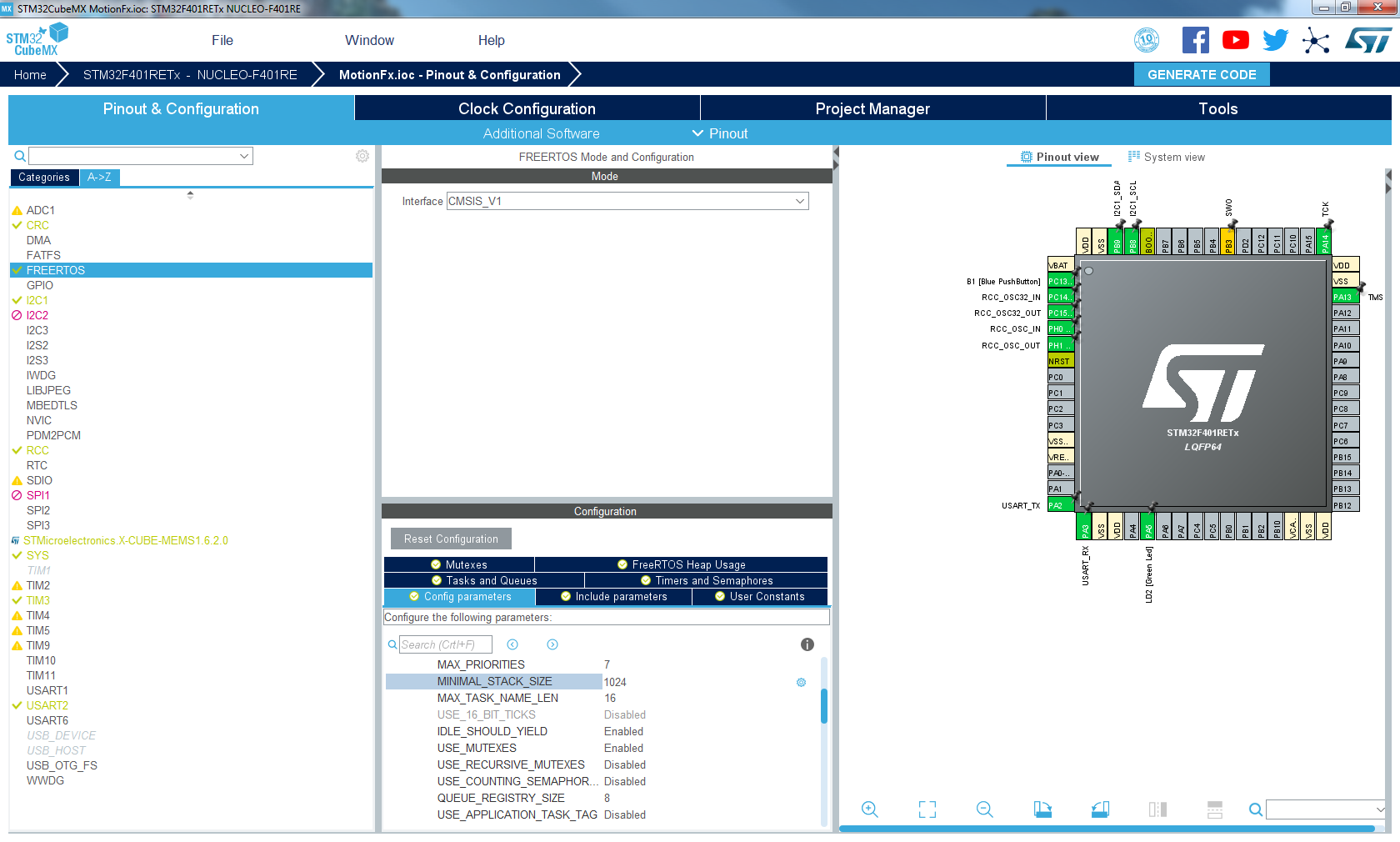
Собираем проект и прошиваем плату.

Подключение к проекту FreeRTOS

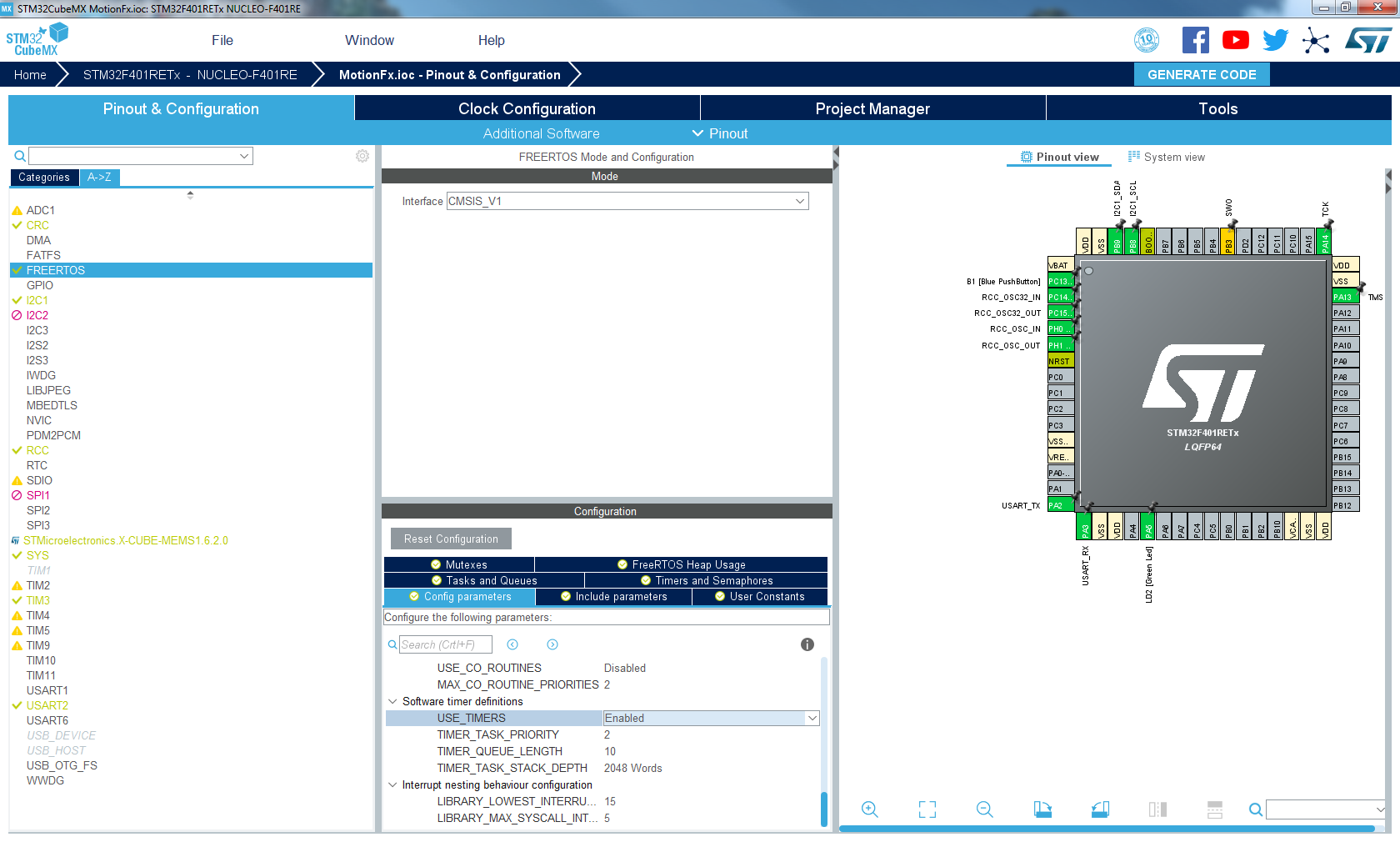
1. Открыть файл проекта CubeMX. Слева выбрать FREERTOS, в открывшемся справа окне выбрать нужный интерфейс.



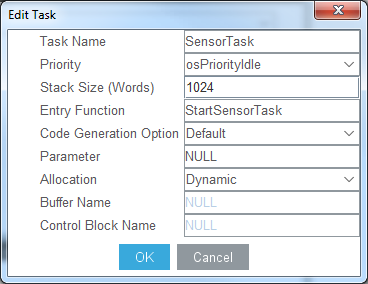
1. Ниже в конфигурации увеличить минимальный размер стека до 1024 слов

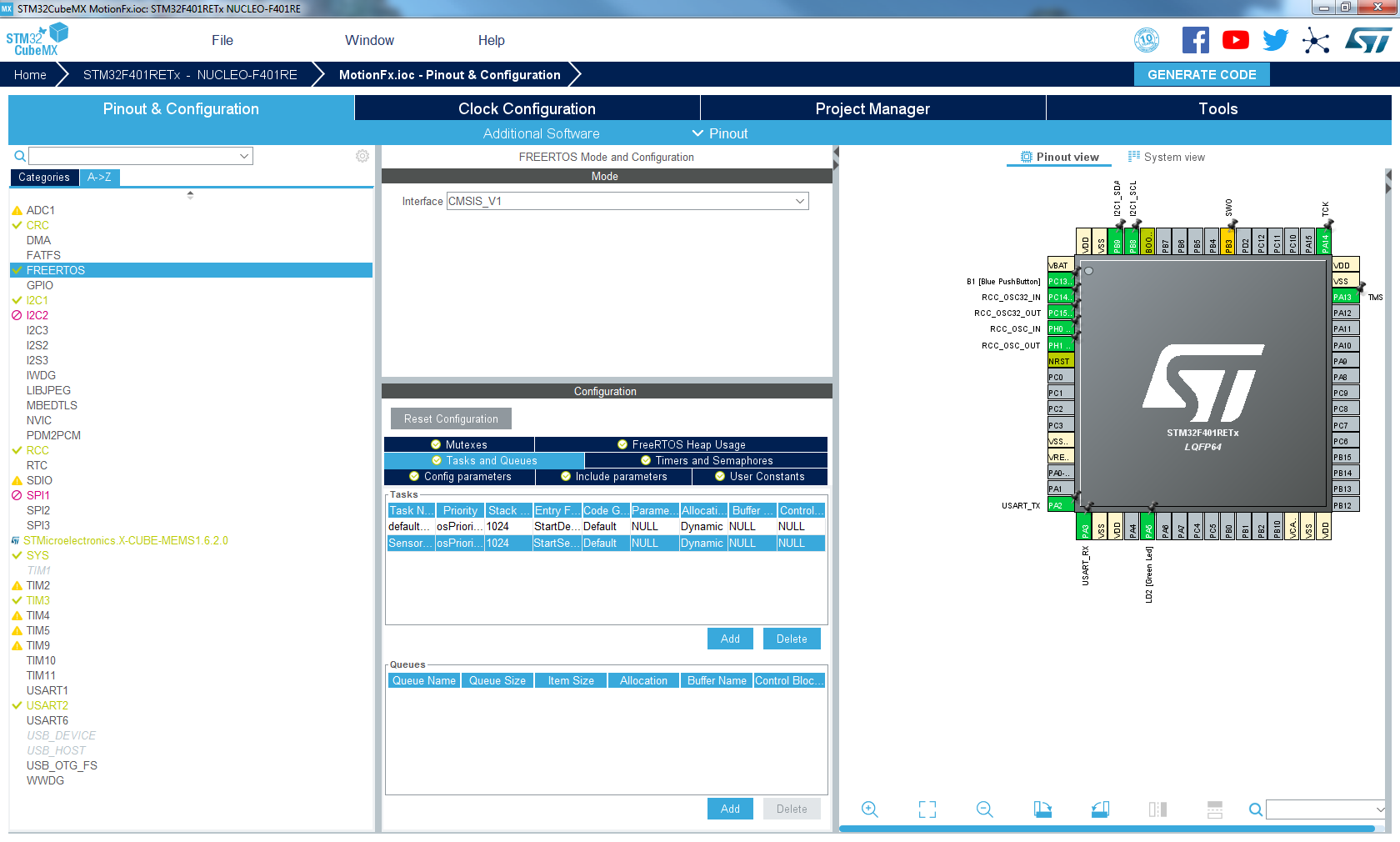


1. Пролистав ниже, находим USE\_TIMERS. Выбираем Enabled.

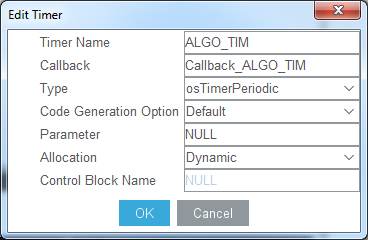


1. Если есть необходимость в нескольких задачах, переходим во вкладку Tasks and Queues, нажимаем кнопку Add, добавляя задачи.

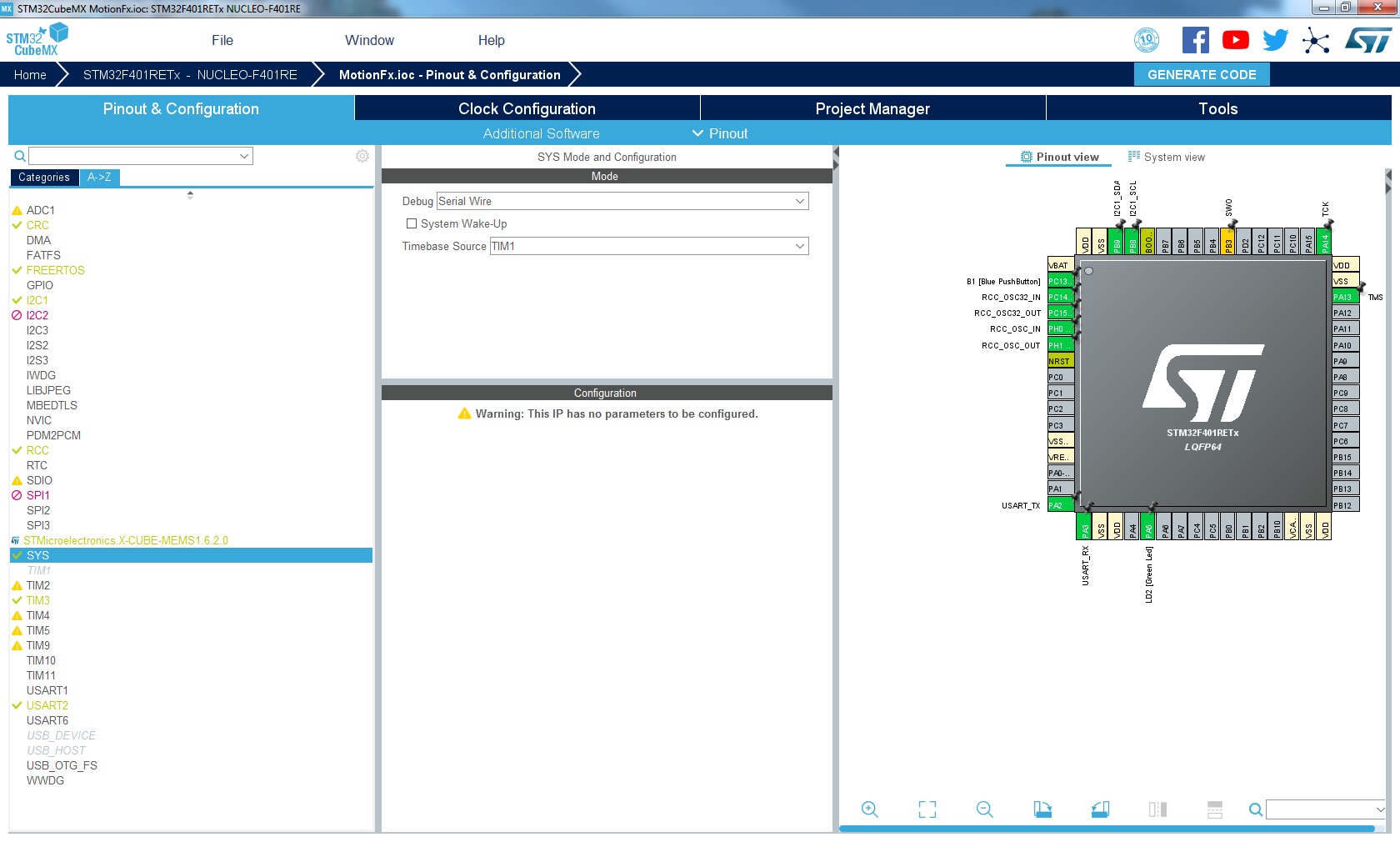




1. Теперь переходим во вкладку Timers and Semaphores. Точно также нажимаем кнопку Add, для удобства переименовываем название таймера и его функции. Тип таймера – osTimerPeriodic.



1. Нажимаем OK.Слева в общем окне выбираем SYS. Меняем Timebase Source на любой свободный таймер.



1. Таймер 3, который использовался ранее в нашем проекте можно убрать, он больше не понадобится.
2. Нажимаем вверху кнопку GENERATE CODE.

Работа с кодом.

1. Нам понадобится видоизмененный файл Sensor.c. В папке проекта нужно добавить/заменить его в папке Src.
2. В файле main.c нужно удалить вызов функции Lib\_Process(). Ее содержимое теперь находится в функции Lib\_ReadRequest(). Здесь остается только вызов функции lib\_init()Теперь часть кода с инициализацией периферии выглядит так:

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_CRC\_Init();

/\* USER CODE BEGIN 2 \*/

**lib\_init();**

/\* USER CODE END 2 \*/

/\* Call init function for freertos objects (in freertos.c) \*/

MX\_FREERTOS\_Init();

1. Код, который ранее обрабатывался в функции Lib\_Process(), теперь обрабатывается в функции Lib\_ReadRequest().

**void Lib\_ReadRequest()**

**{**

Accelero\_Sensor\_Handler(IKS01A2\_LSM6DSL\_0);

Gyro\_Sensor\_Handler(IKS01A2\_LSM6DSL\_0);

Magneto\_Sensor\_Handler(IKS01A2\_LSM303AGR\_MAG\_0);

/\* Sensor Fusion specific part \*/

FX\_Data\_Handler();

**}**

1. В файле freertos.c, который появился в новом проекте, нужно добавить:

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

#include "app\_x-cube-mems1.h"

#include "Sensor.h"

/\* USER CODE END Includes \*/

1. В задачу, которую мы выделили для обработки датчиков, добавляем вызов osTimerStart(ALGO\_TIMHandle, 10).

**void StartSensorTask(void const \* argument)**

**{**

/\* USER CODE BEGIN StartSensorTask \*/

**osTimerStart(ALGO\_TIMHandle, 10);**

/\* Infinite loop \*/

for(;;)

{

osDelay(1);

}

/\* USER CODE END StartSensorTask \*/

**}**

Последний аргумент этой функции – период срабатывания прерывания в миллисекундах, аналогичен периоду срабатывания таймера из первого варианта проекта без freeRTOS.

1. В функцию Callback\_ALGO\_TIM добавляем вызов нашей функции обработки и калибровки датчиков.

/\* Callback\_ALGO\_TIM function \*/

**void Callback\_ALGO\_TIM(void const \* argument)**

**{**

/\* USER CODE BEGIN Callback\_ALGO\_TIM \*/

**Lib\_ReadRequest();**

/\* USER CODE END Callback\_ALGO\_TIM \*/

**}**

1. Проект собираем и прошиваем.