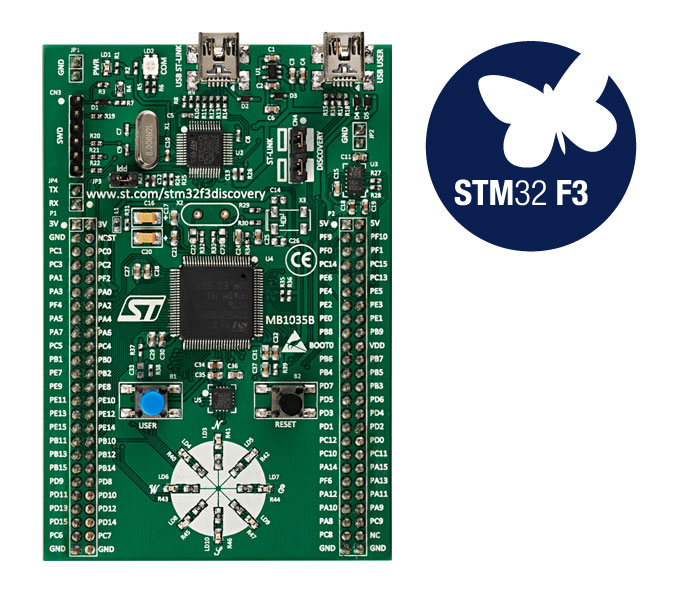
Создание базового проекта для STM32F3Discovery с помощью STM32CubeMX и Keil uVision 5



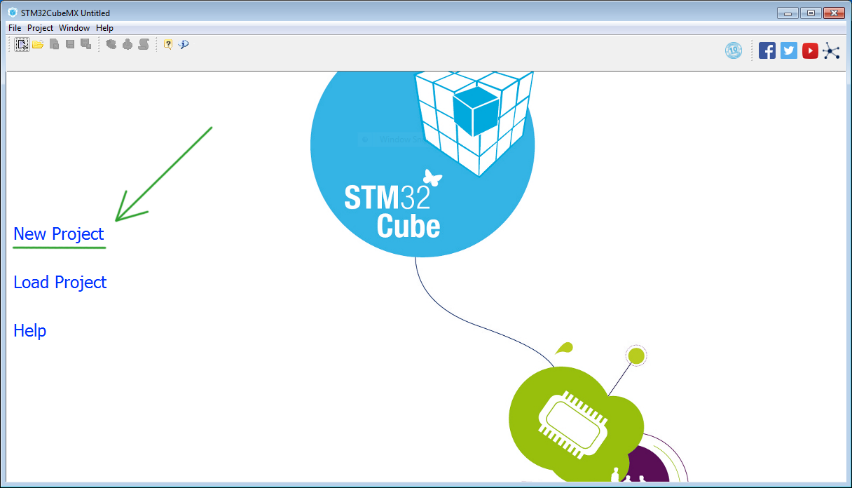
Введение

В последних версиях Keil (v5.25.2.0 or higher) и расширений для STM32F3 (pack version 2.1.0 or higher) инициализация проекта делегирована STM32CubeMX, который выполняет следующие шаги:

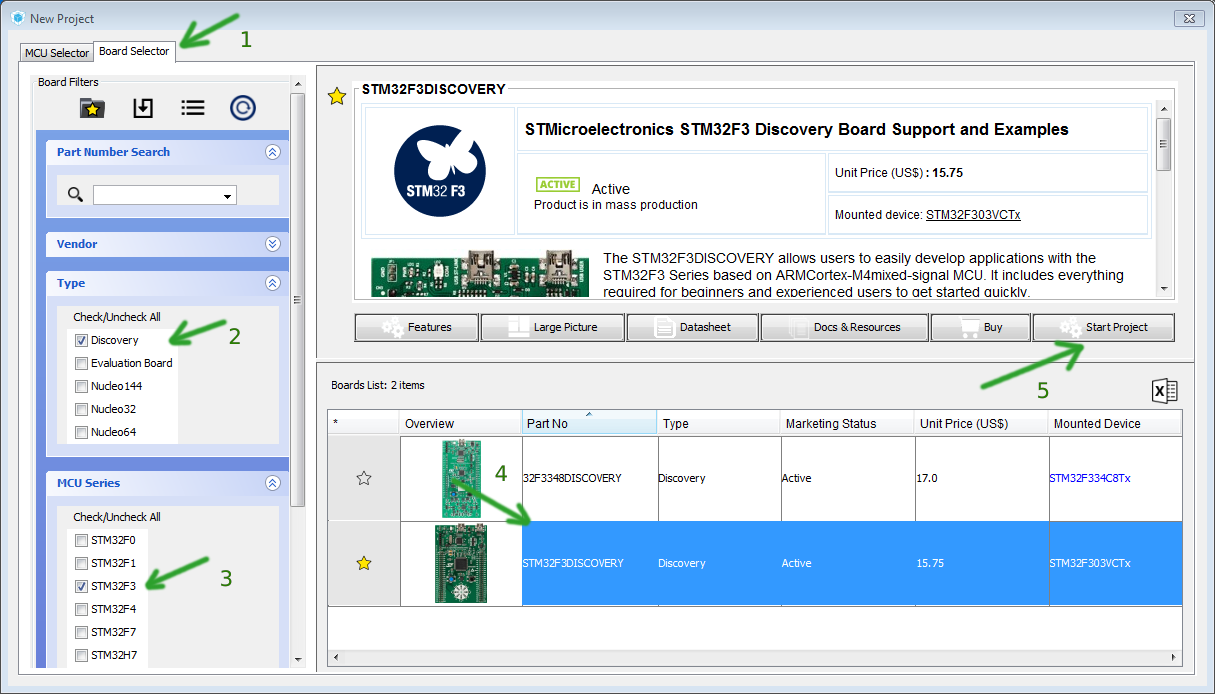
1. добавляется в проект файл с адресами регистров
2. добавляет в проект код инициализации, который запускается при старте микроконтроллера и потом вызывает main
3. добавляет в проект скрипт линковщика, в котором содержится информация об адресах flash и RAM

# Создание проекта с помощью STM32CubeMX

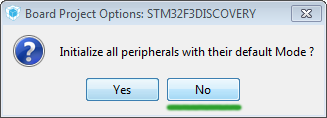
1. запустите STM32CubeMX и создайте новый проект



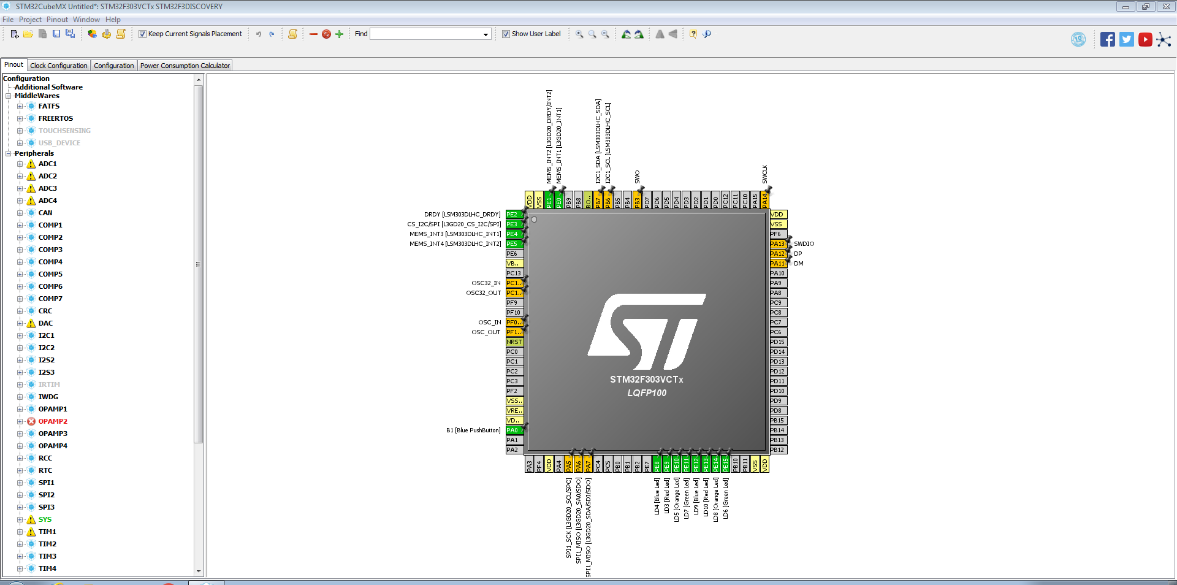
1. перейдите на вкладку board, выберите тип платы, серию MCU, найдите свою плату, и нажмите “Start Project”



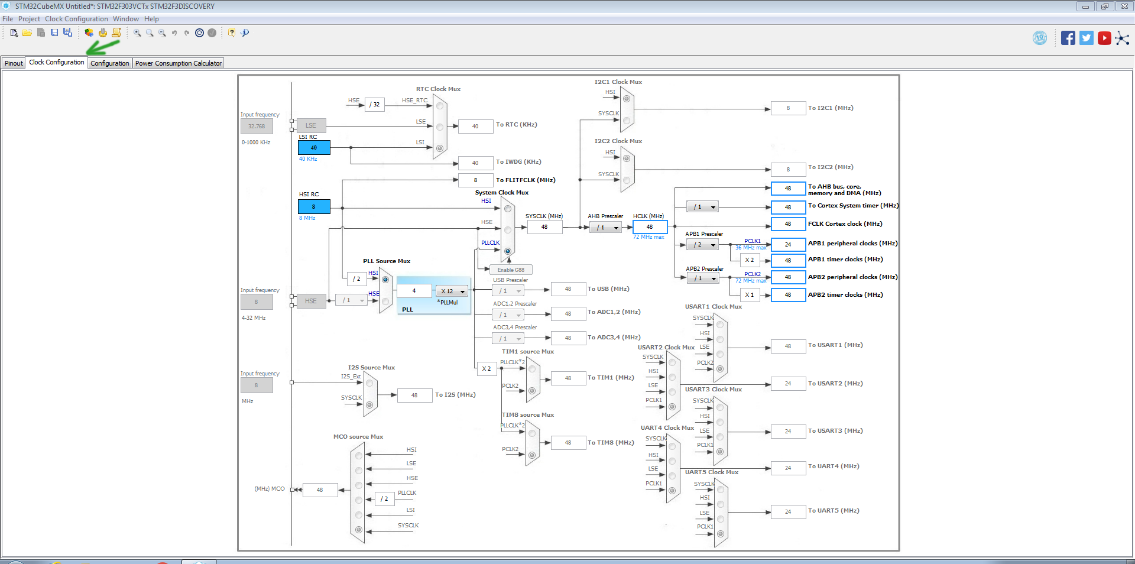
1. откажитесь от инициализации периферии по умолчанию



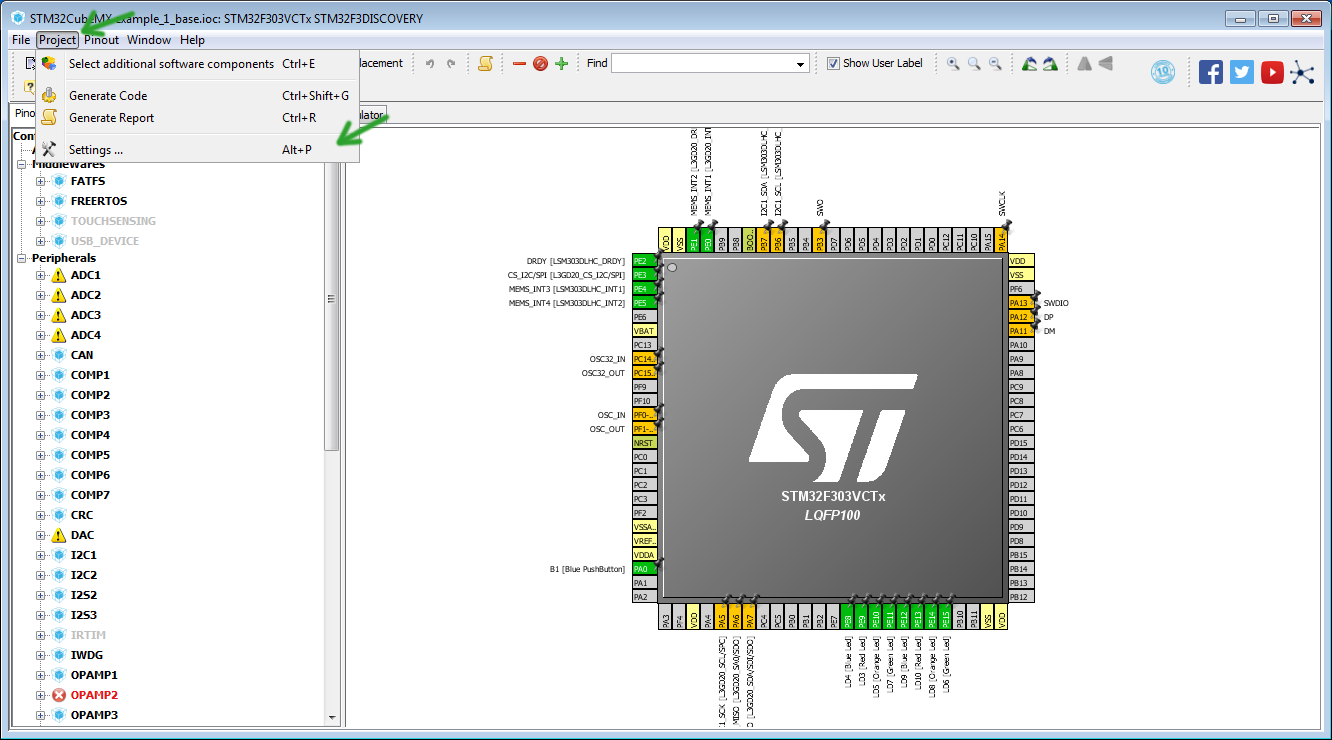
1. CubeMX вам покажет распиновку микроконтроллера. Ничего настраивать на данной вкладке не нужно



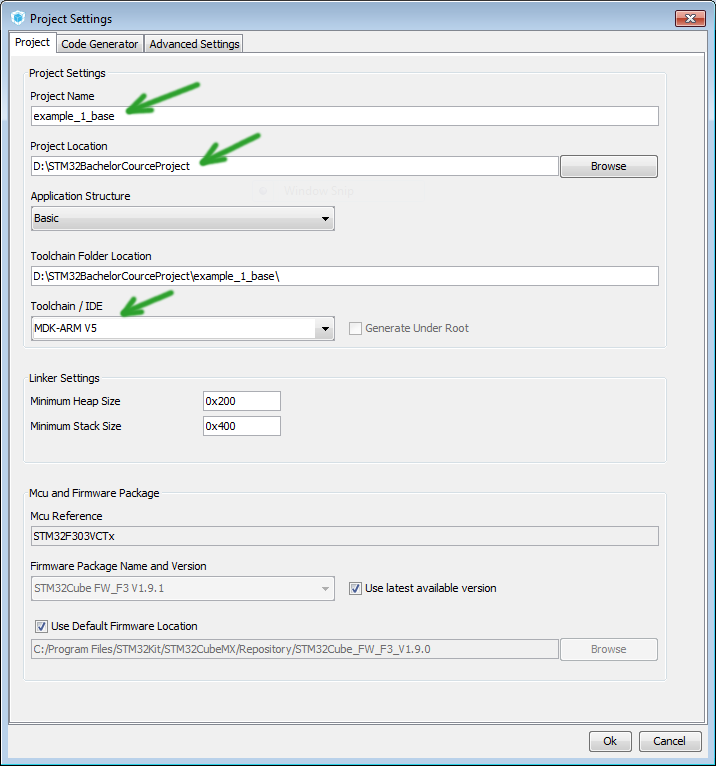
1. на вкладке “Clock Configuration” можно посмотреть схему конфигурации тактирующих сигналов. Она вам может понадобится для определения, входной тактовой частоты устройства, с которым вы работаете



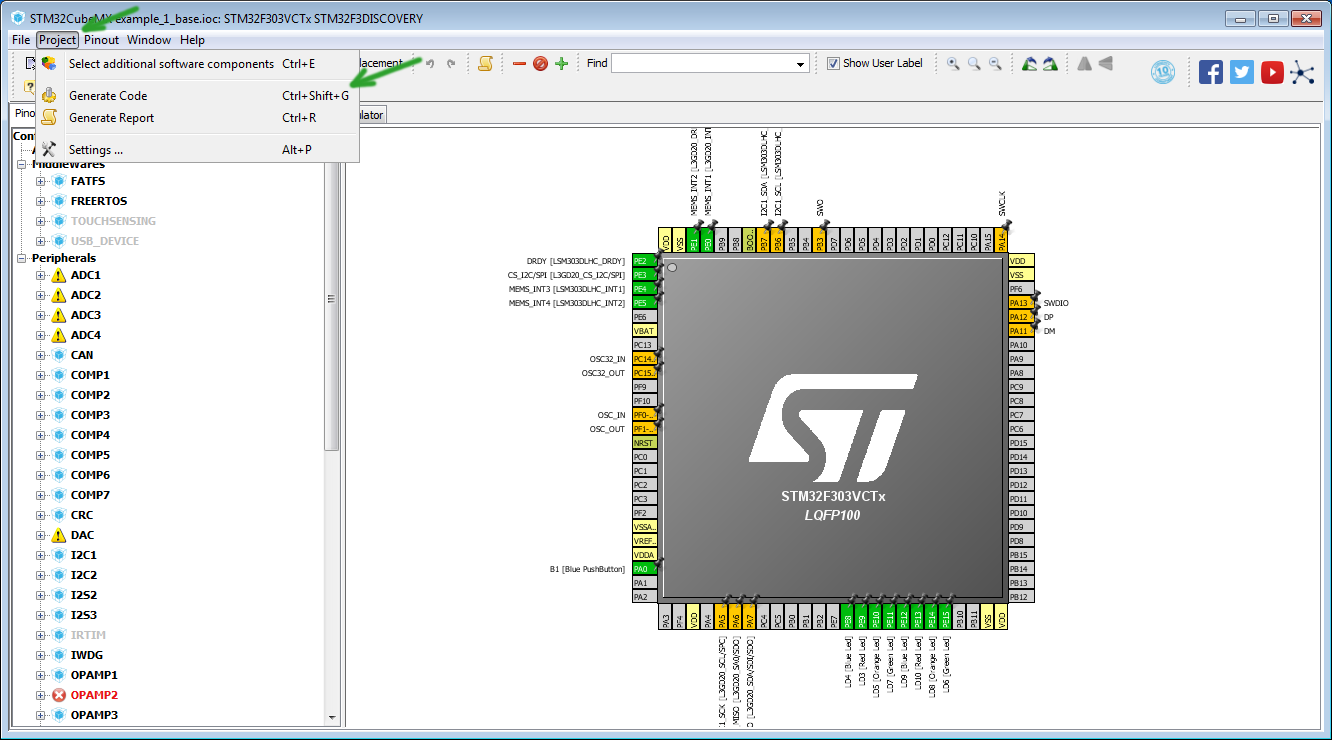
1. откройте настройки проекта



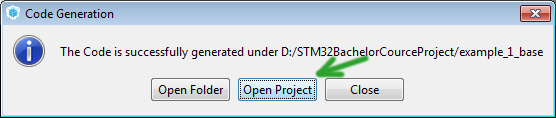
1. выберете имя проекта, папку куда его сохранить, и среду MDK-ARM 5 (keil uVision 5)



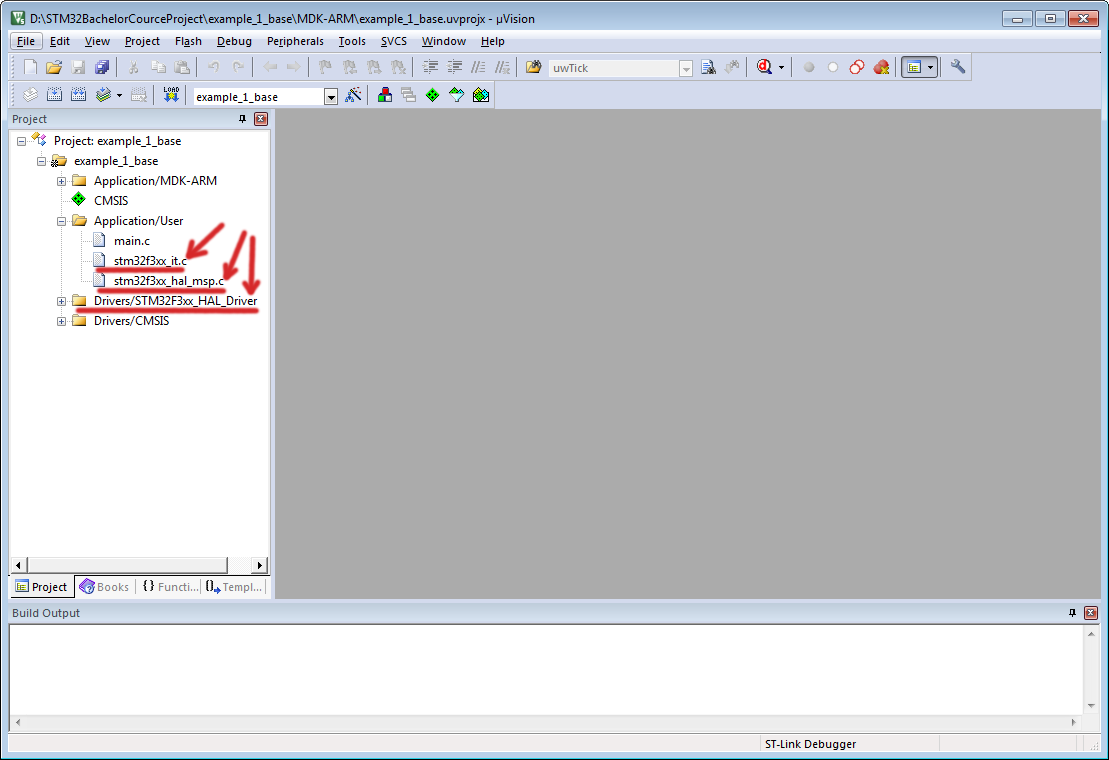
1. запустите генерацию кода проекта



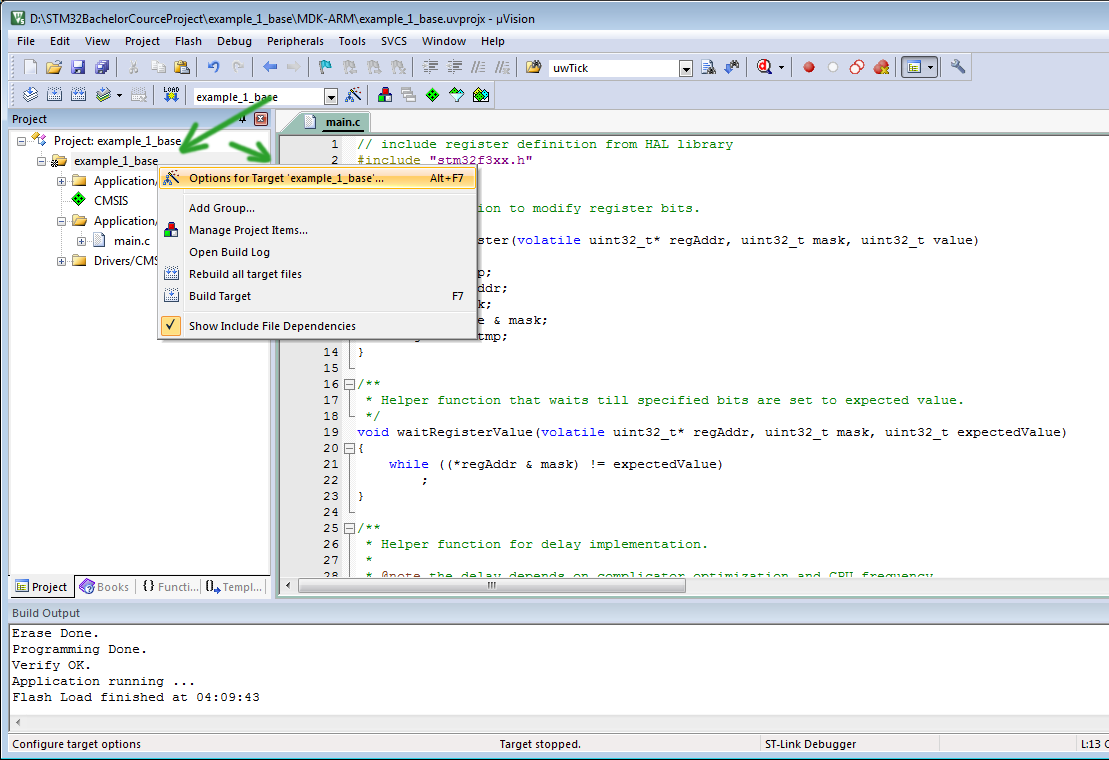
1. после генерации кода, CubeMX предложит открыть его в keil. Откройте проект и закройте CubeMX



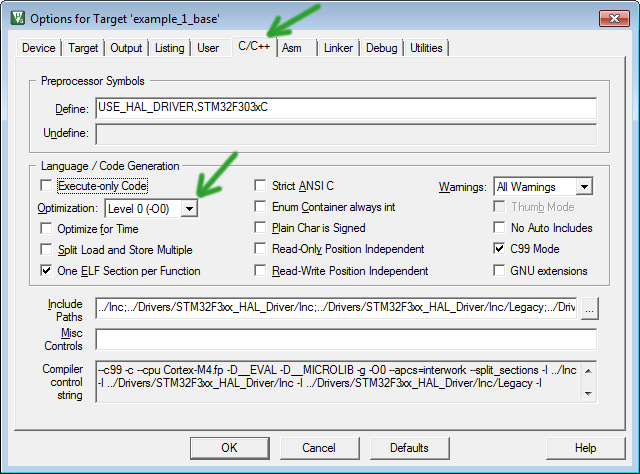
1. в созданном проекте удалите файлы stm32f3xx\_it.c, stm32f3\_hal\_msp.c и папку Drivers/STM32F3xx\_HAL\_driver



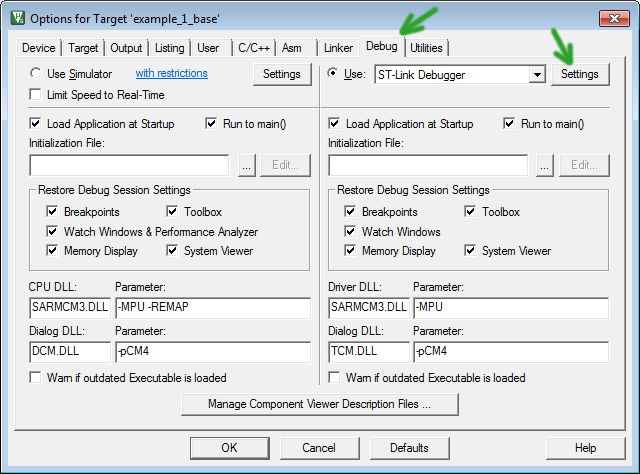
1. откройте настройки проекта



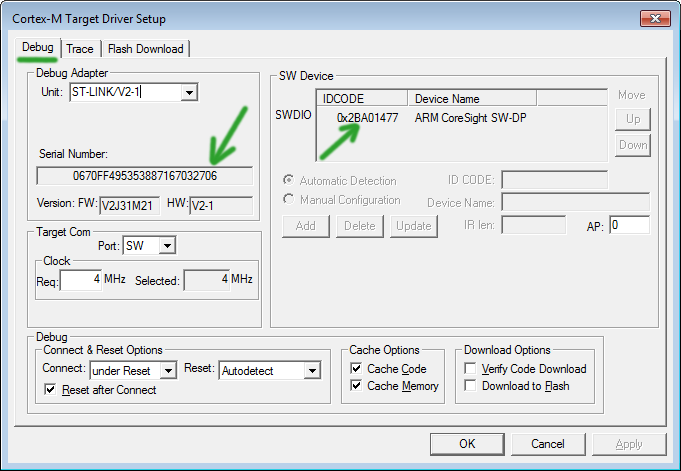
1. перейдите на вкладку C/C++ и выключите оптимизации. Это упростит отладку вашей программы



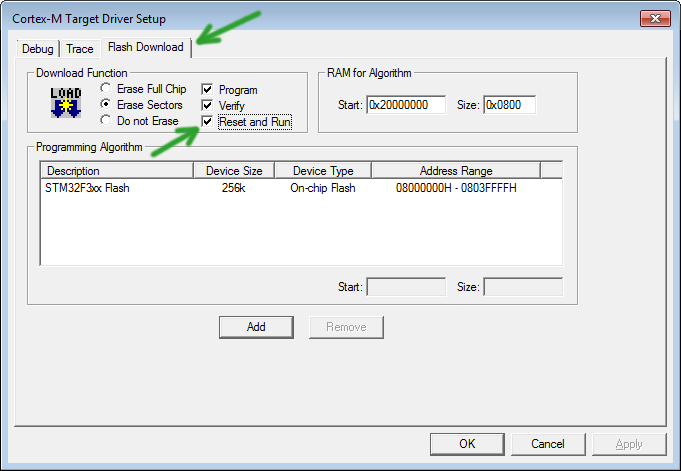
1. перейдите на вкладку Debug, и нажмите на Settings



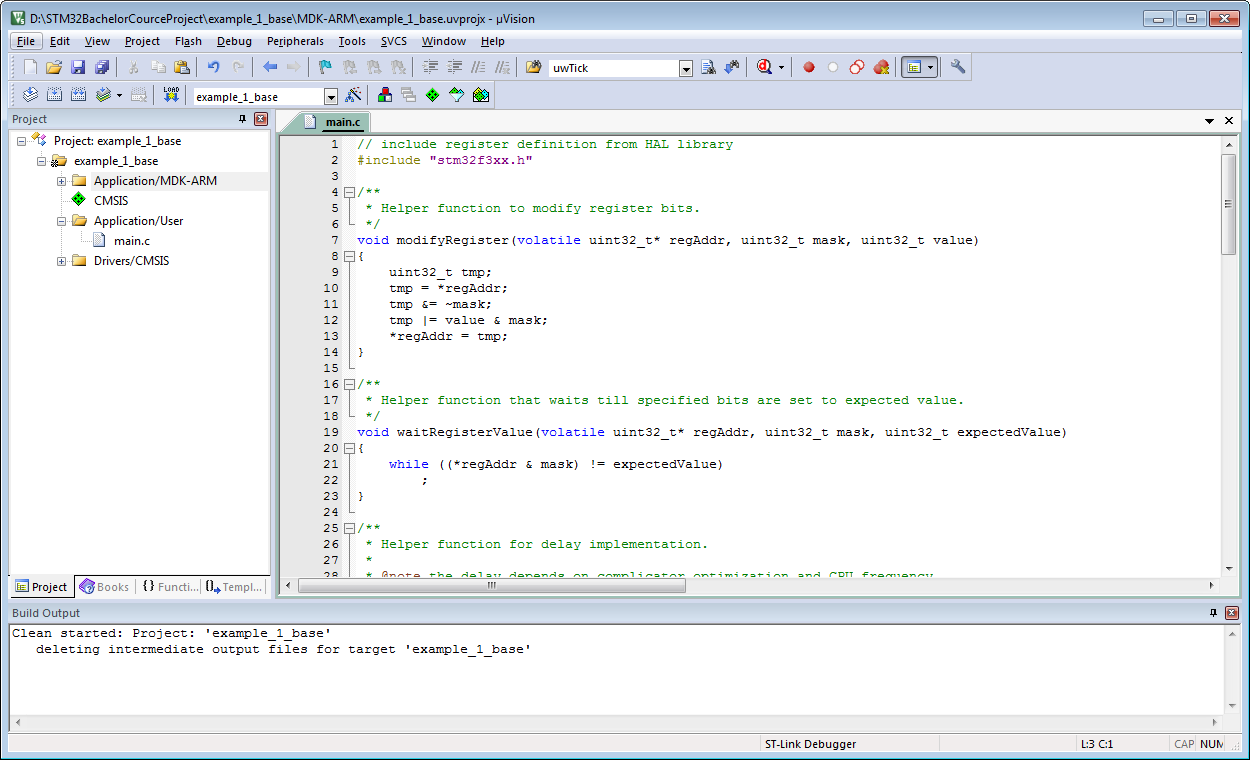
1. перед вами появится вкладка Debug. На ней не надо ничего настраивать, но по ней можно увидеть, видит ли Keil микроконтроллер



1. перейдите на вкладку с Flash download и включите автоматическую перезагрузку платы после ее перепрошивки



1. закройте данные окна, и замените код в main.c файле следующим



// include register definition from HAL library

#include "stm32f3xx.h"

/\*\*

\* Helper function to modify register bits.

\*/

void modifyRegister(volatile uint32\_t\* regAddr, uint32\_t mask, uint32\_t value)

{

uint32\_t tmp;

tmp = \*regAddr;

tmp &= ~mask;

tmp |= value & mask;

\*regAddr = tmp;

}

/\*\*

\* Helper function that waits till specified bits are set to expected value.

\*/

void waitRegisterValue(volatile uint32\_t\* regAddr, uint32\_t mask, uint32\_t expectedValue)

{

while ((\*regAddr & mask) != expectedValue)

;

}

/\*\*

\* Helper function for delay implementation.

\*

\* @note the delay depends on complicator optimization and CPU frequency.

\*/

void delay(uint32\_t num\_ops)

{

volatile uint32\_t count = 0;

while (count < num\_ops) {

count++;

}

}

/\*\*

\* Function to configure CPU frequency (SYSCLOCK).

\*

\* After configurration CPU frequency will be changed from 8 MHz to 48 MHz.

\*/

void configClock()

{

// HSI is enabled by default and used as SYSCLOCK.

// So currect SYSCLOCK frequency is 8 MHz

// 1. Configure PLL (phase-locked loop) to increase frequncy

// 1.1 Set CFGR bits 15-16 to 0b00 use HSI as source clock for PLL

// note: f\_PLL\_input = freq\_HSI / 2 = 4 MHz

modifyRegister(&(RCC->CFGR), 0x00018000, 0x00000000);

// 1.2 Set CFGR bits 18-21 to 0b1010 for PLL multiplication factor 12

// note: f\_PLL\_output = f\_PLL\_input \* 12 = 4 \* 12 = 48 MHz

modifyRegister(&(RCC->CFGR), 0x003C0000, 0x00280000);

// 1.3 Enable PLL (set CR bit 24 to 1).

modifyRegister(&(RCC->CR), 0x01000000, 0x01000000);

// 1.4 Wait till PLL is enable (check CR bit 25).

waitRegisterValue(&(RCC->CR), 0x02000000, 0x02000000);

// 2. Switch SYSCLK from HSI to PLL

// 2.1 Set PLL as SYSCLK source (set CFGR bits 0-1 to 0b10).

modifyRegister(&(RCC->CFGR), 0x00000003, 0x00000002);

// 2.2 Wait end of SYSCLK source configuration (check that CFGR bits 2-3 is 0b10).

waitRegisterValue(&(RCC->CFGR), 0x0000000C, 0x00000008);

// update global variable with system frequency

// note: we can use it check that set configure frequency correctly (the SystemCoreClock should be set to 0X02DC6C00)

SystemCoreClockUpdate();

}

/\*\*

\* Configure LED pins.

\*/

void configLEDS()

{

// 1. Enable clock of the GPIOE

modifyRegister(&(RCC->AHBENR), RCC\_AHBENR\_GPIOEEN, RCC\_AHBENR\_GPIOEEN);

// 2. Set high speed

modifyRegister(&(GPIOE->OSPEEDR), 0xFFFF0000, 0xFFFF0000);

// 3. Disable pull up/down registers

modifyRegister(&(GPIOE->PUPDR), 0xFFFF0000, 0x00000000);

// 4. Set pull/push mode

modifyRegister(&(GPIOE->OTYPER), 0x0000FF00, 0x00000000);

// 5. Set general output mode

modifyRegister(&(GPIOE->MODER), 0xFFFF0000, 0x55550000);

}

int main(void)

{

// configure CPU frequency to 48 MHz

configClock();

// configure leds

configLEDS();

// simple led blinking

uint8\_t ledMask = 0x01;

while (1) {

// update led mask

if (ledMask & 0x80) {

ledMask = ledMask ^ 0x01;

}

ledMask = (ledMask << 1) + 0x01;

// show animation

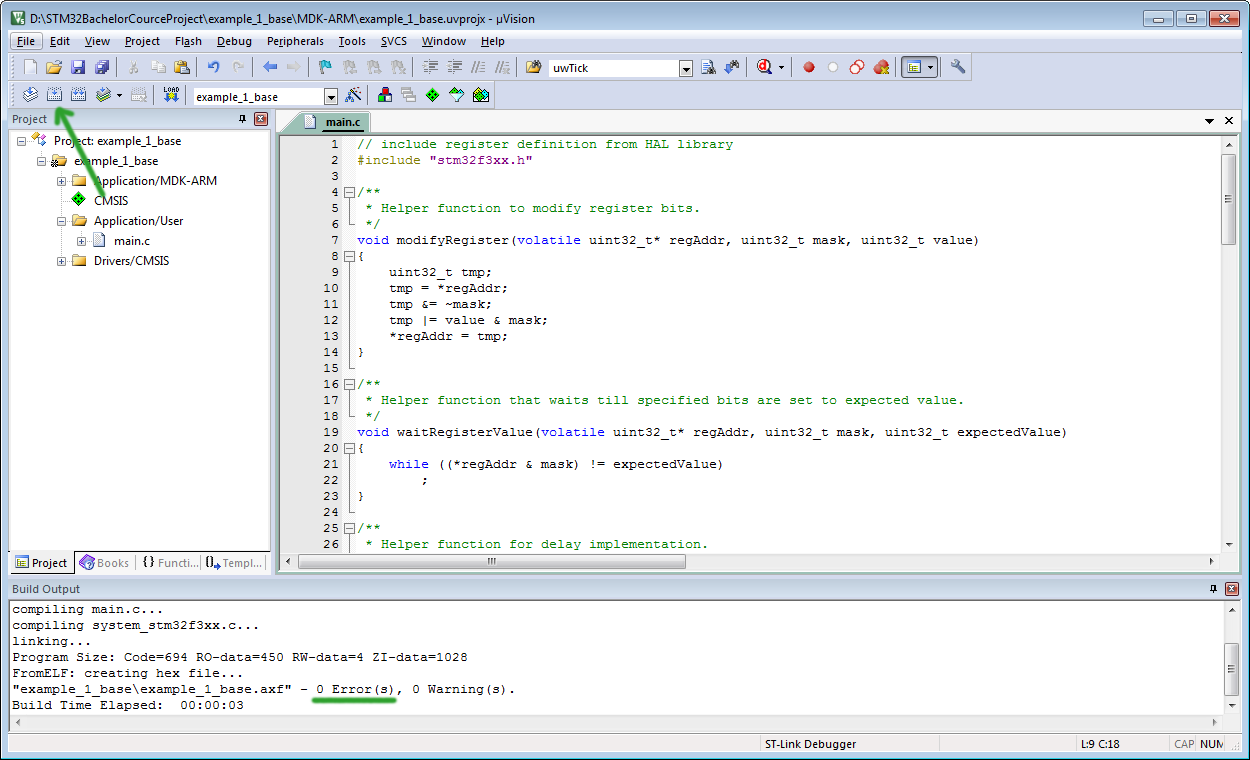
GPIOE->ODR = (GPIOE->ODR & 0xFFFF00FF) | (ledMask << 8);

delay(1000000);

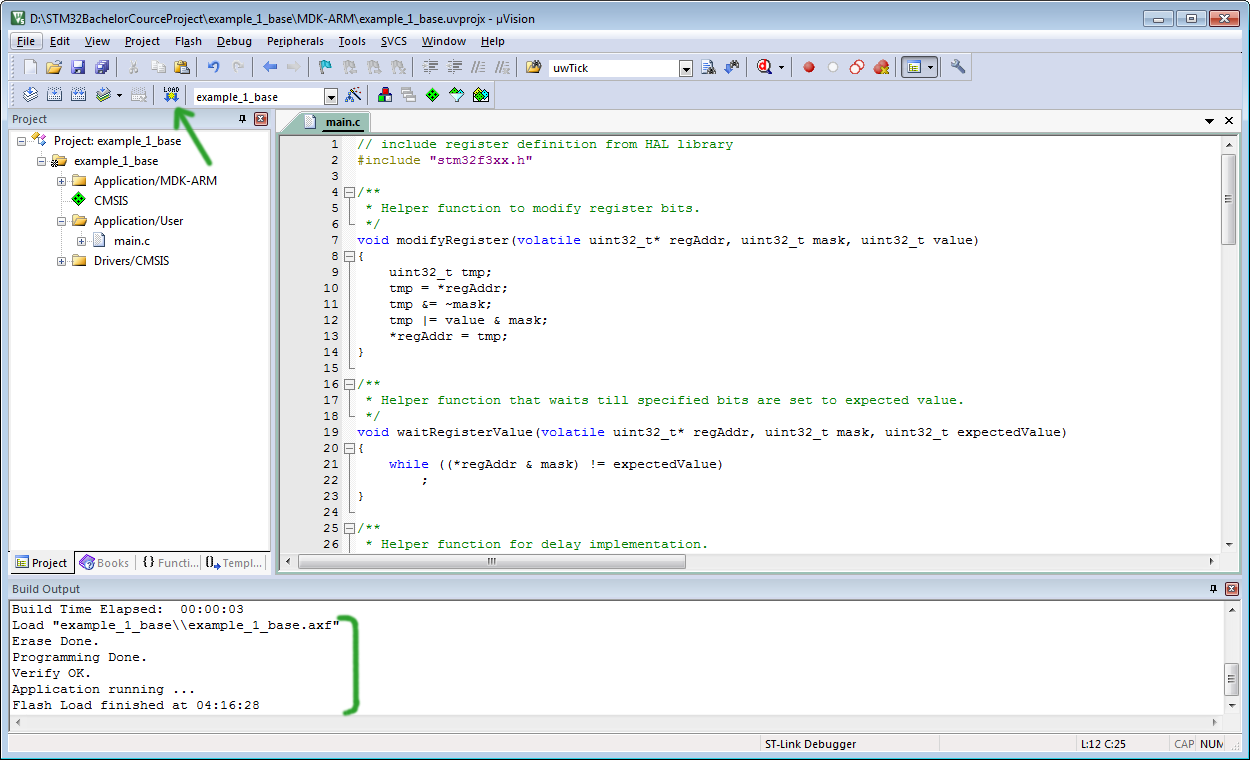
}

}

1. скомпилируйте проект (быстрая клавиша – F7). После компиляции убедитесь, что в окне логов нет ошибок компиляции



1. загрузите проект на плату (быстрая клавиша – F8)



Если все было сделано правильно, то на плате вы увидите мигание светодиодов.