RISC

Гарвардская архитектура – шина данных и шина инструкций

D-bus

I-bus

STM32

ARM Cortex-M

32-битное микропроцессорное ядро

Подготовка к работе

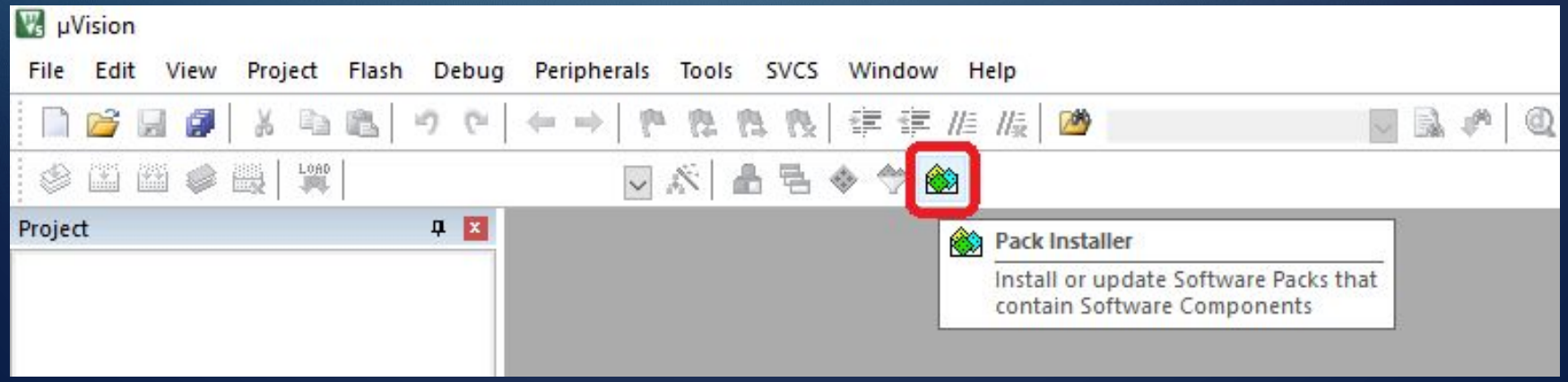
Создание шаблонного проекта *в среде Keil 5, библиотека SPL*

Создадим проект с помощью библиотеки StdPeriph (SPL – standart peripherals driver) (кроме STM32F7 – серии)

Для работы понадобится сама библиотека SPL, ее можно найти на сайте www.st.com.

С создания подобного шаблона начинается разработка каждого нового проекта. Старайтесь придерживаться единой структуры во всех проектах

В Keil необходимо скачать драйверы под микроконтроллер. Для этого запускаем "Packinstaller" и нажимаем на кнопку "Update"



После обновления выбираем производителя микроконтроллера (STmicroelectronics) и указываем нужную серию микроконтроллера (STM32F0), нажимаем "Установить"

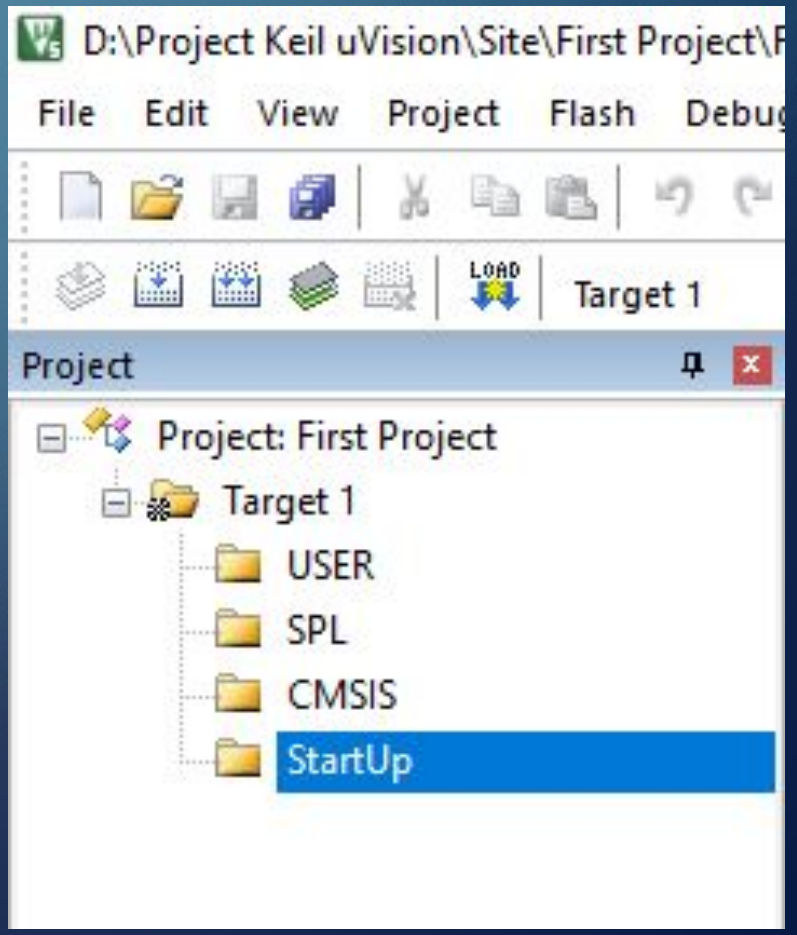
Далее в меню "Project" выбираем "New project"

Откроется диалоговое окно сохранения проекта. В поиске необходимо найти модель своего микроконтроллера и нажать "OK". В новом окне не выбирая ничего из предложенного, нажимаем "OK"

Удобно разбивать проект на группы файлов. Для создания группы нужно в контекстном меню имени проекта (Target 1 по умолчанию) выбрать пункт "AddGroup". Создайте группы, как на образце.

Открываем архив с библиотекой и папку с проектом. В папке с проектом создадим папку с именем "User". В папке "User" создадим две папки "Include" и "Source". В папке "Source" создадим файл "main.c". Возвращаемся в корневую папку проекта. В архиве с библиотекой переходим в папку "Libraries" и перетягиваем папки "CMSIS" и "STM32F0xx_StdPeriph_Driver" в папку с проектом

В архиве с библиотекой возвращаемся в корневой каталог, заходим в папку с примерами "Projects" -> "STM32F0xx_StdPeriph_Examples" -> "ADC" -> "ADC_AnalogWatchdog". В этом примере нас интересует файл "stm32f0xx_conf.h", перетаскиваем его в проект, в папку "STM32F0xx_StdPeriph_Driver"



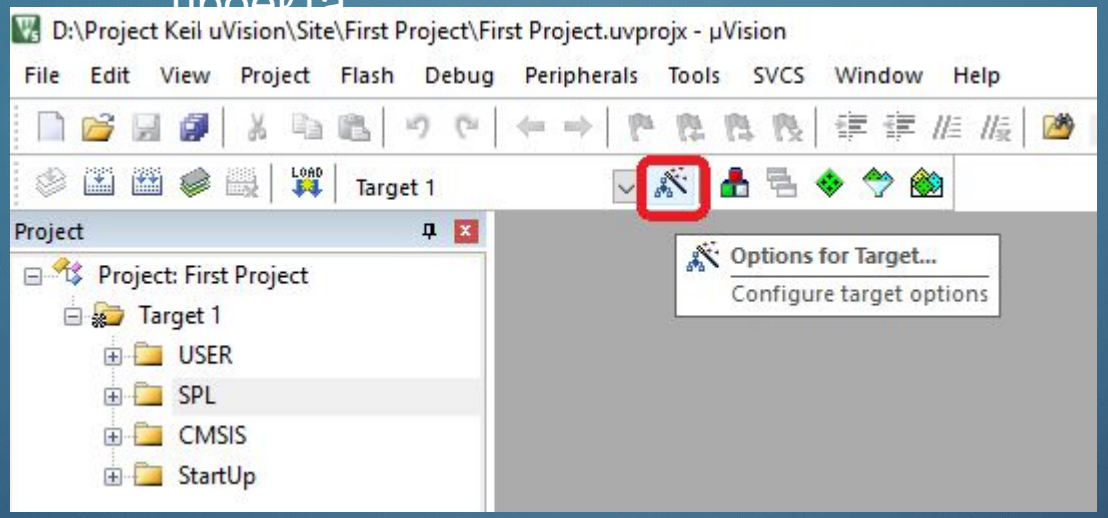
Переходим в Keil, двойным щелчком по группе "USER" вызываем окно добавления файла, выбираем файл "main.c" в папке "User" -> "Source", нажимаем "Add", закрываем окно

Далее аналогично с группой "SPL". Дважды щелкаем по ней, и в открывшемся окне переходим в папку "STM32F0xx_StdPeriph_Driver". Тут нужно переключить фильтр видимости файлов на "AllFiles", выбираем файл "stm32f0xx_conf.h", нажимаем "Add". Также добавляем все файлы из папки "src", закрываем окно

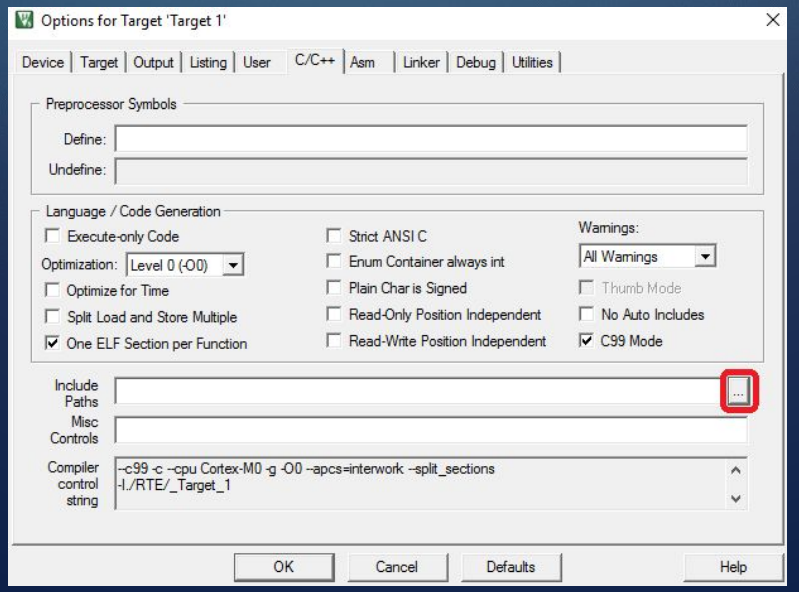
В группе "CMSIS" точно также добавляем файл "system_stm32f0xx.c" из папки "CMSIS" -> "Device" -> "ST" -> "stm32f0xx" -> "Source" -> "Template«

В группе "StartUp" добавляем файл "startup_stm32f030.s" из папки "CMSIS" -> "Device" -> "ST" -> "stm32f0xx" -> "Source" -> "Template" -> "arm" (здесь тоже нужно будет переключить фильтр видимости файлов на "AllFles")

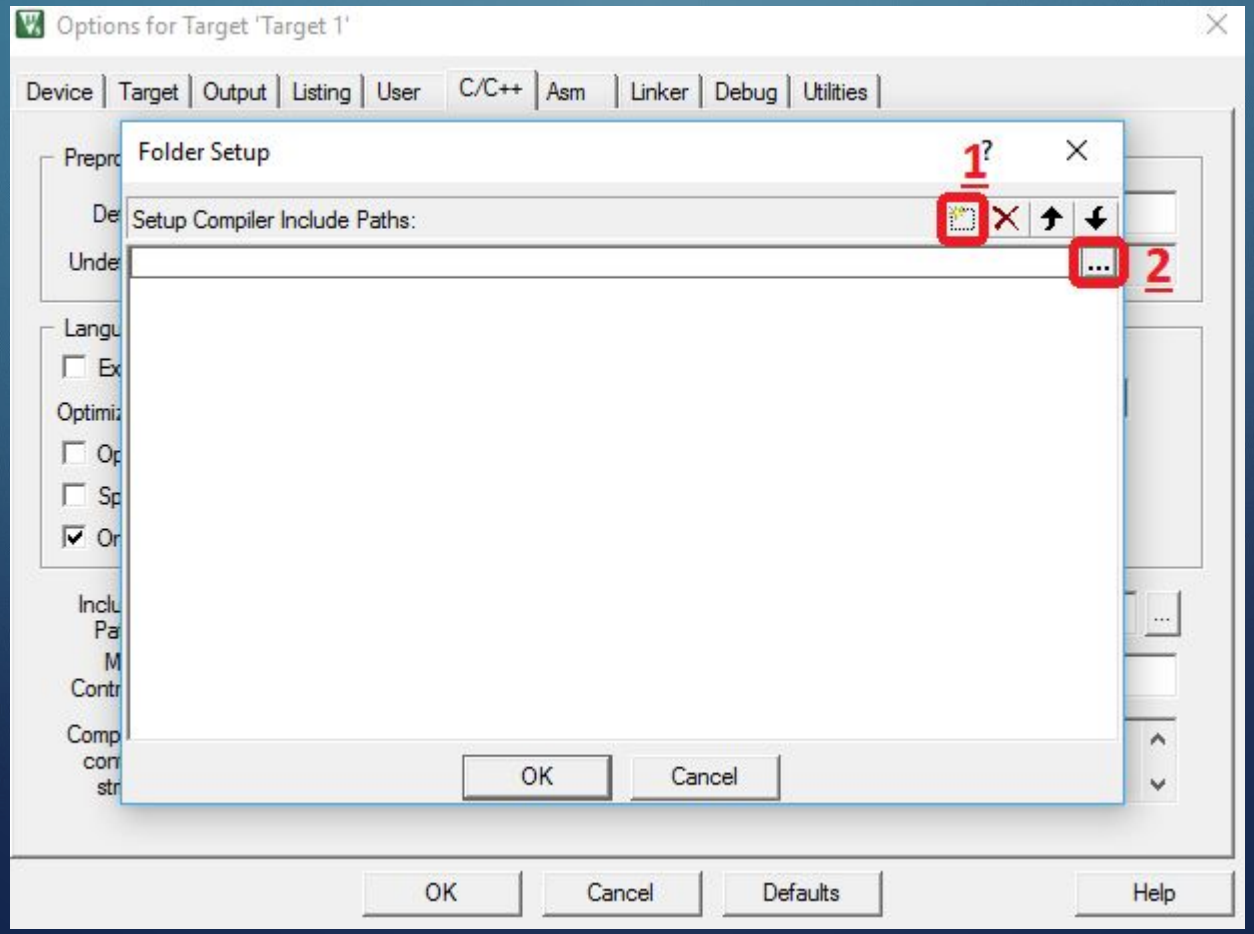
Нажимаем кнопку настройки проекта



В открывшемся окне переходим на вкладку "C/C++" и в графе "IncludePaths" нажимаем кнопку "…"



В появившемся окне нажимаем на кнопку "New" (№1)
и на кнопку "…" (№2)



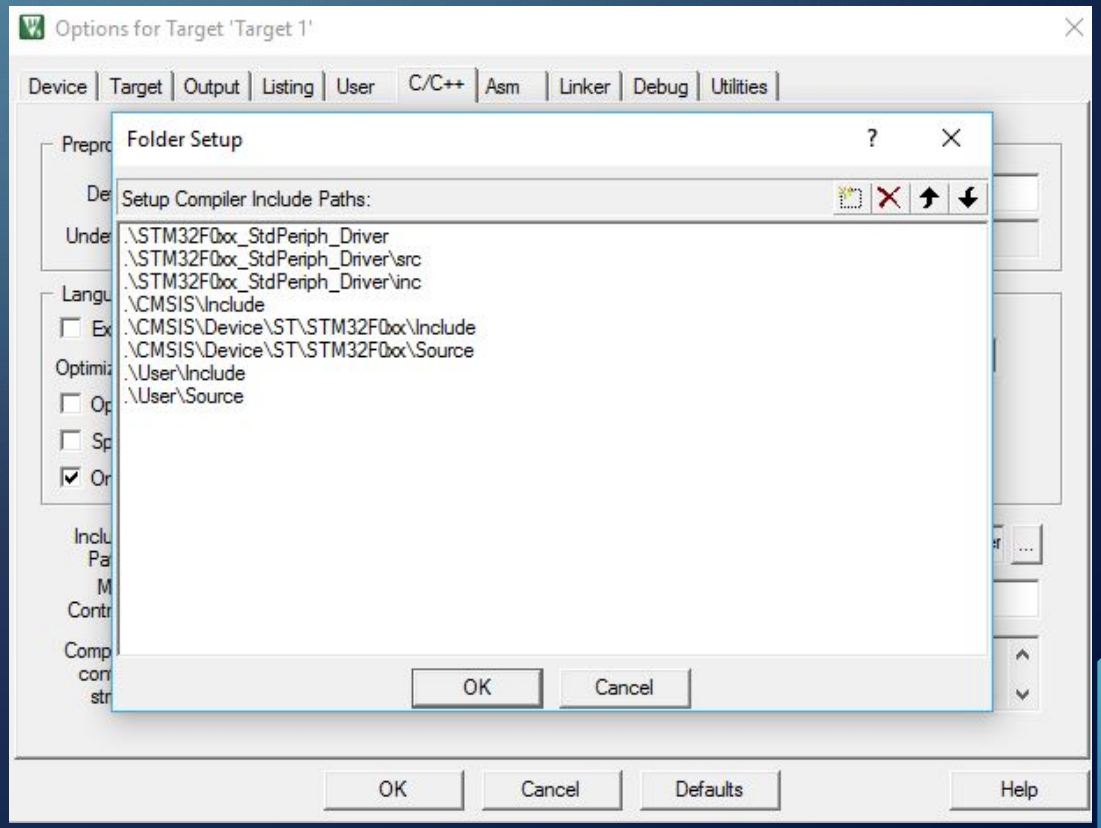
Откроется окно выбора папки, в нем выбираем папку
"STM32F0xx_StdPeriph_Driver"

Далее опять нажимаем кнопку "New" и кнопку "…" в новой строчке, в открывшемся окне заходим в папку "STM32F0xx_StdPeriph_Driver" и выбираем папку "src"

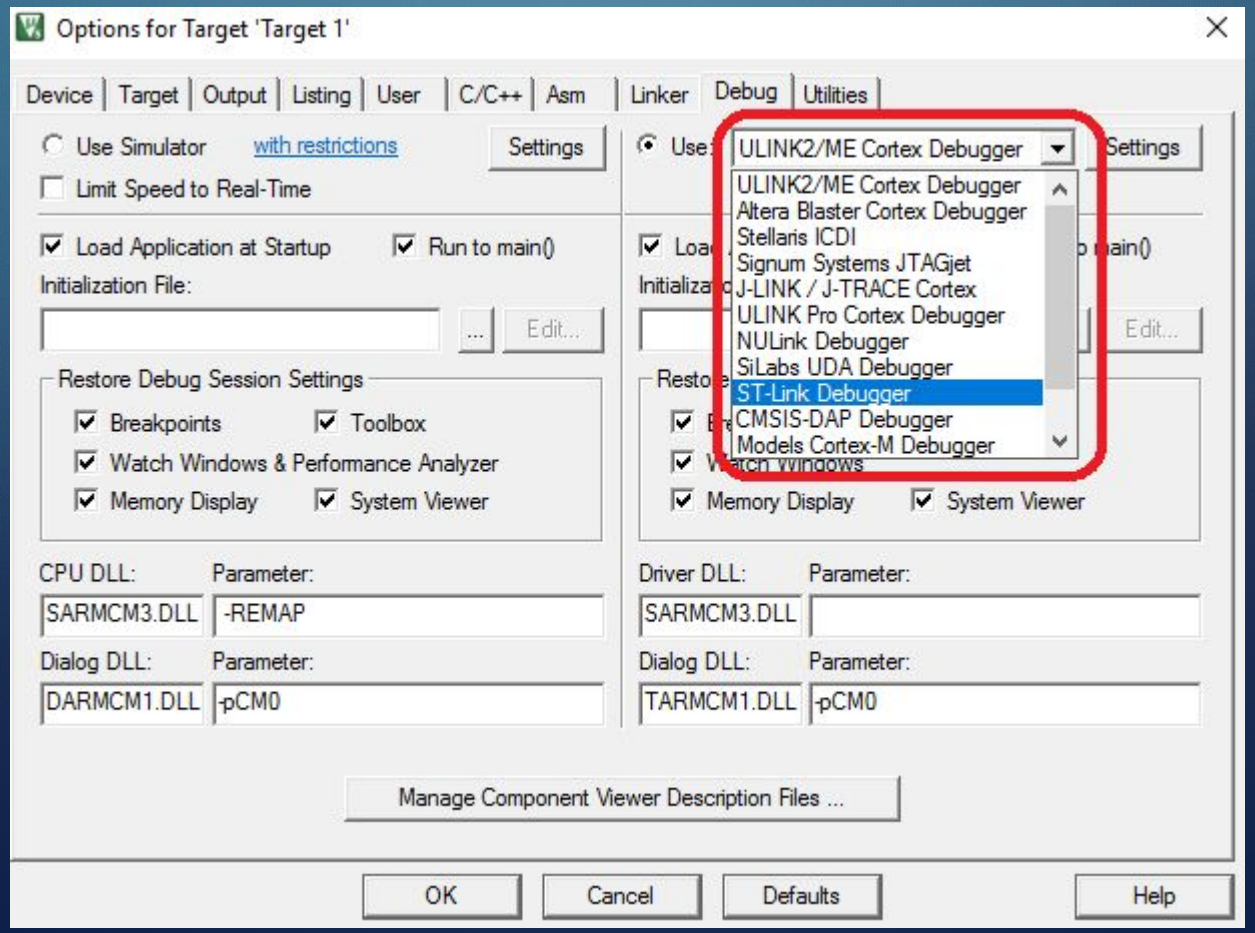
Добавляем еще:

- из папки "STM32F0xx_StdPeriph_Driver" папку "inc";
- из папки "CMSIS" папку "Include";
- из папки "CMSIS" -> "Device" -> "ST" -> "stm32f0xx" папку "Include";
- из папки "CMSIS" -> "Device" -> "ST" -> "stm32f0xx" папку "Source";
- из папки "User" папку "Include";
- из папки "User" папку "Source".

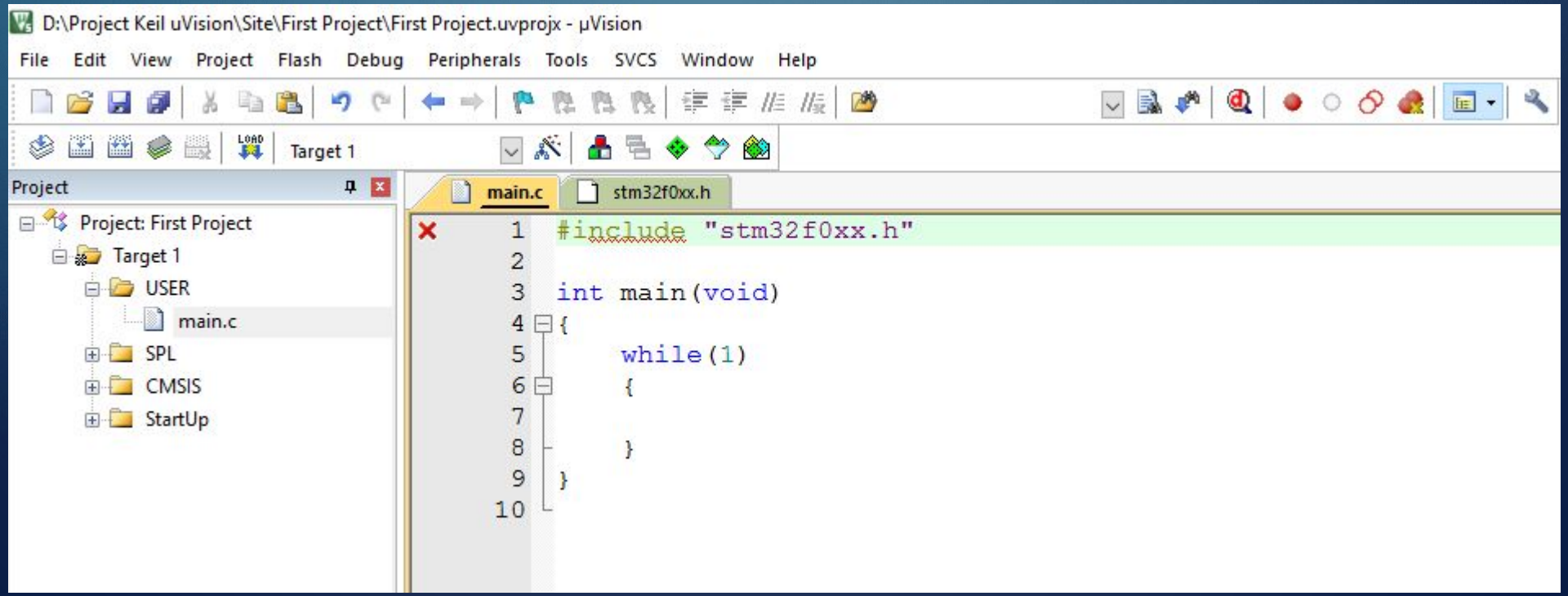
В итоге должно получиться



Нажимаем OK и переходим на вкладку "Debug", выбираем программатор-отладчик из списка.



Теперь можно приступать к написанию кода. Раскрываем группу "USER", дважды щелкаем по файлу "main.c" и пишем в нем типовую заготовку кода

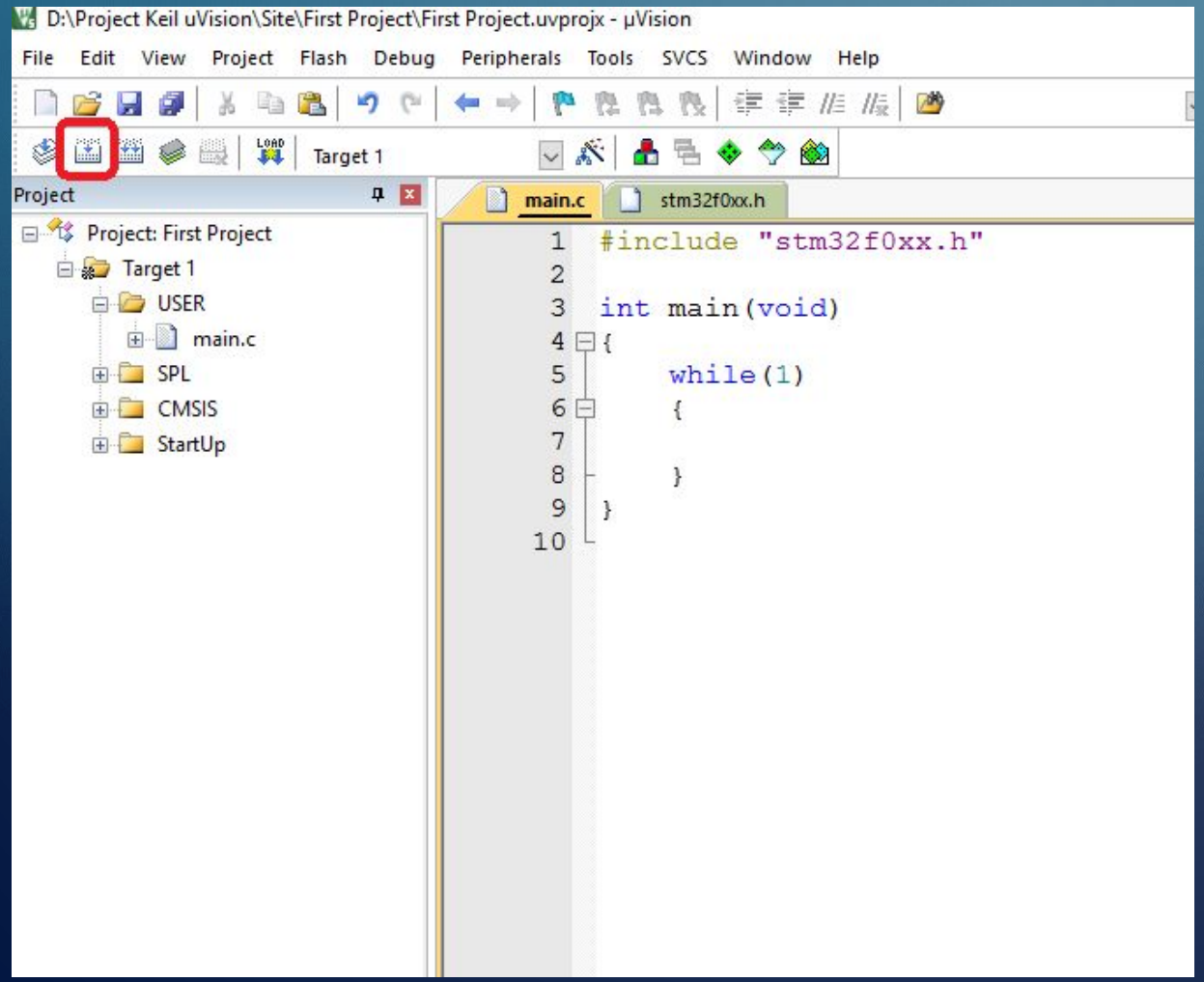


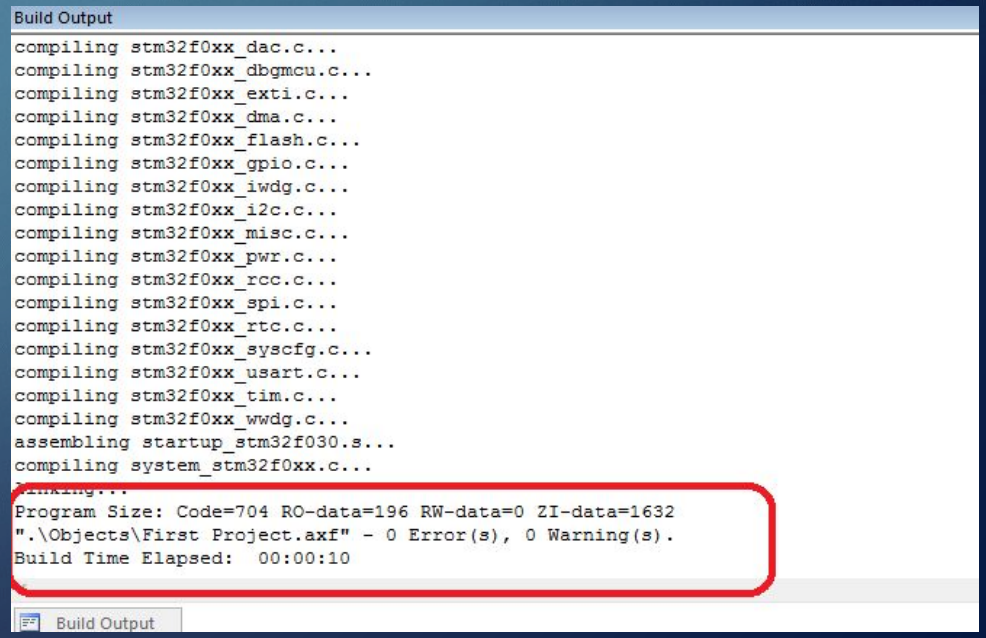
Для того, чтобы узнать, что писать на первой строчке в "include", открываем папку с проектом и переходим в папку "CMSIS" -> "Device" -> "ST" -> "stm32f0xx" -> "Include". Видим два файла - "stm32f0xx.h" и "system_stm32f0xx.h". Нам нужен первый, его название и нужно вставить в "include" на первой строке

Открываем первый файл в редакторе, на 64-ой строке видим "нужно раскомментировать используемое устройство". Находим нашу модель микроконтроллера и раскомментируем ее

Ниже находим строчку "/*#define USE_STDPERIPH_DRIVER*/" и раскомментируем ее тоже. Сохраняем файл и идем в Keil

Попробуем собрать проект. Нажимаем кнопку "Build" и смотрим в окно вывода





Ошибок и замечаний нет, проект собрался!

While (1)
{
} – бесконечный цикл

Основы программирования STM32

Программа для микроконтроллера пишется на одном из языков программирования в виде текстового файла

Специальная программа-транслятор преобразует исходн текст программы в машинные коды, понятные микропроцессору, этот процесс называется **компиляцией**

В результате компиляции будет создан так называемый **hex**-файл

«hexadecimal» – «шестнадцатеричная система счисления»

Двоичное число	*Шестнадцатеричное число*
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Структура – это особый тип данных, состоящий из нескольких разнотипных переменных (полей). В общем случае объявление структуры имеет следующий вид:

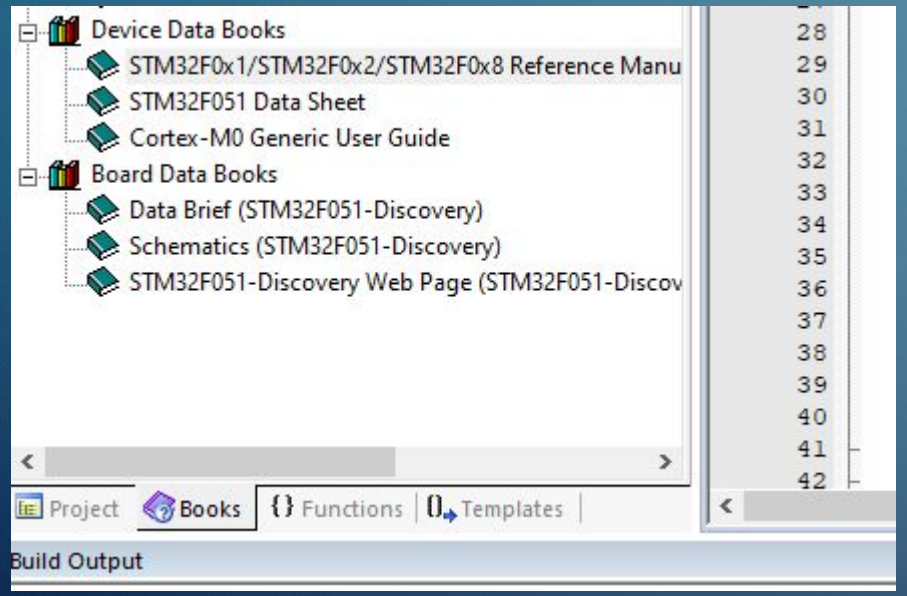
```
struct имя_структуры
{
тип поле_1;
…
тип поле_N;
};
```

```
struct DATE
{
int Day;
int Month;
int Year;
}
structDATEMyBirthday = {7, 8, 1974};
```

Для доступа к полям структуры в программе используют запись вида имя_структуры.поле. То есть, в представленном выше примере структуры DATE для инициализации полей можно было воспользоваться следующими операторами:

```
MyBirthday.Day = 7;
MyBirthday.Month = 8;
MyBirthday.Year = 1974;
```

По умолчанию вся перефирия МК отключена для энергосбережения
Необходимо включить тактирование



6.4.6 AHB peripheral clock enable register (RCC_AHBENR)

Address offset: 0x14

Reset value: 0x0000 0014

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCEN	Res.	IOPFEN	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Res.
							rw		rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	FLITFEN	Res.	SRAMEN	DMA2EN	DMAEN
									rw		rw		rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TSCEN:** Touch sensing controller clock enable
Set and cleared by software.
0: TSC clock disabled
1: TSC clock enabled

Bit 23 Reserved, must be kept at reset value.

Bit 22 **IOPFEN:** I/O port F clock enable
Set and cleared by software.
0: I/O port F clock disabled
1: I/O port F clock enabled

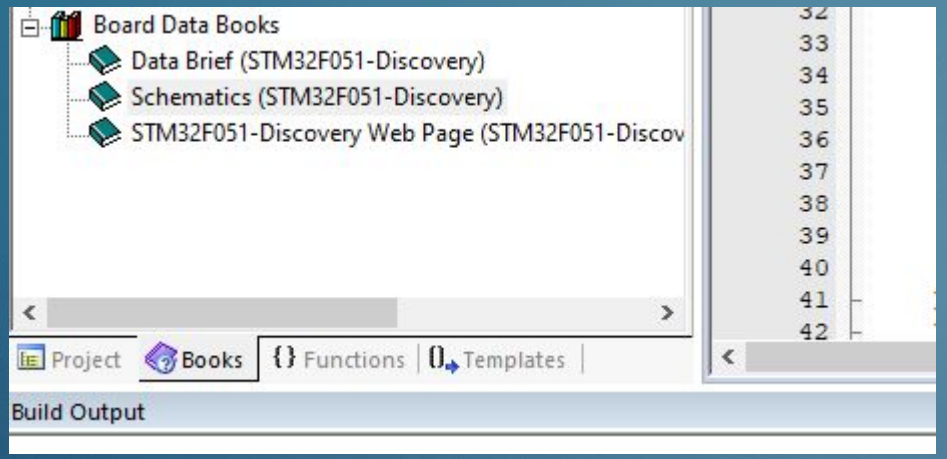
Bit 21 **IOPEEN:** I/O port E clock enable
Set and cleared by software.
0: I/O port E clock disabled
1: I/O port E clock enabled

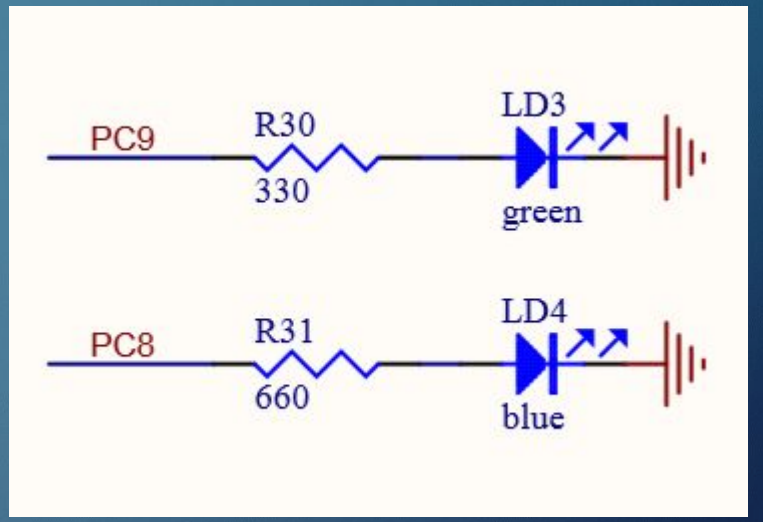
Bit 20 **IOPDEN:** I/O port D clock enable
Set and cleared by software.
0: I/O port D clock disabled
1: I/O port D clock enabled

Bit 19 **IOPCEN:** I/O port C clock enable
Set and cleared by software.
0: I/O port C clock disabled
1: I/O port C clock enabled

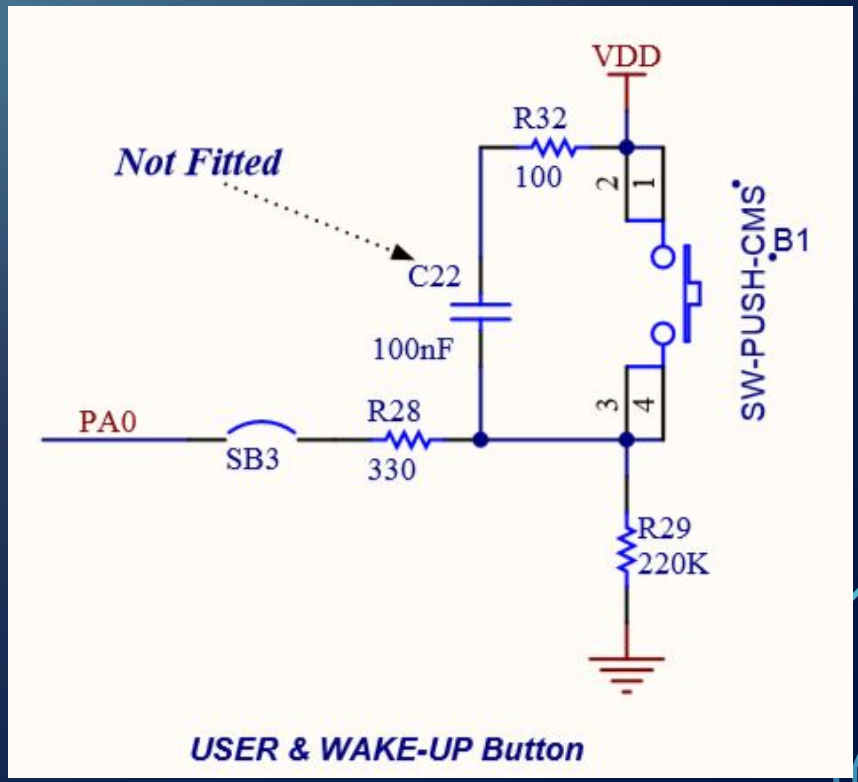
Bit 18 **IOPBEN:** I/O port B clock enable
Set and cleared by software.
0: I/O port B clock disabled
1: I/O port B clock enabled

```
RCC->AHBENR |= RCC_AHBENR_GPIOCEN;
RCC->AHBENR |= RCC_AHBENR_GPIOAEN;
```





Порты могут работать в двух режимах: вход (прием сигнала) и выход (передача сигнала).
Только 0 или 1
Порты микроконтроллера могут выдать ток не более 20 мА. Для подключения более мощных нагрузок следует использовать силовые ключи
R29 – подтяжка к земле
R30-31 – ограничивают ток через светодиоды



General Purpose Input/Output (GPIO) - основной способ связи с внешней средой

Настройка порта на

8.4.1 GPIO port mode register (GPIOx_MODER) (x =A..F)

Address offset:0x00

Reset values:

- 0x2800 0000 for port A
- 0x0000 0000 for other ports

Номер пина

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O mode.
00: Input mode (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

На выход

```
GPIOC ->MODER = 0x55500;
```

За инициализацию каждого регистра отвечает два бита

Подтягивание

8.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..F)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output type.

Подтягивание ← 0: Output push-pull (reset state)

1: Output open-drain → Без подтягивания

Настройка скорости

8.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A..F)

Address offset: 0x08

Reset value:

- 0x0C00 0000 for port A
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEEDR7 [1:0]		OSPEEDR6 [1:0]		OSPEEDR5 [1:0]		OSPEEDR4 [1:0]		OSPEEDR3 [1:0]		OSPEEDR2 [1:0]		OSPEEDR1 [1:0]		OSPEEDR0 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **OSPEEDRy[1:0]**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output speed.

x0: Low speed

01: Medium speed

11: High speed

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

Установка 1 на

8.4.6 GPIO port output data register (GPIOx_ODR) (x = A..F)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy:** Port output data bit (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A..F).

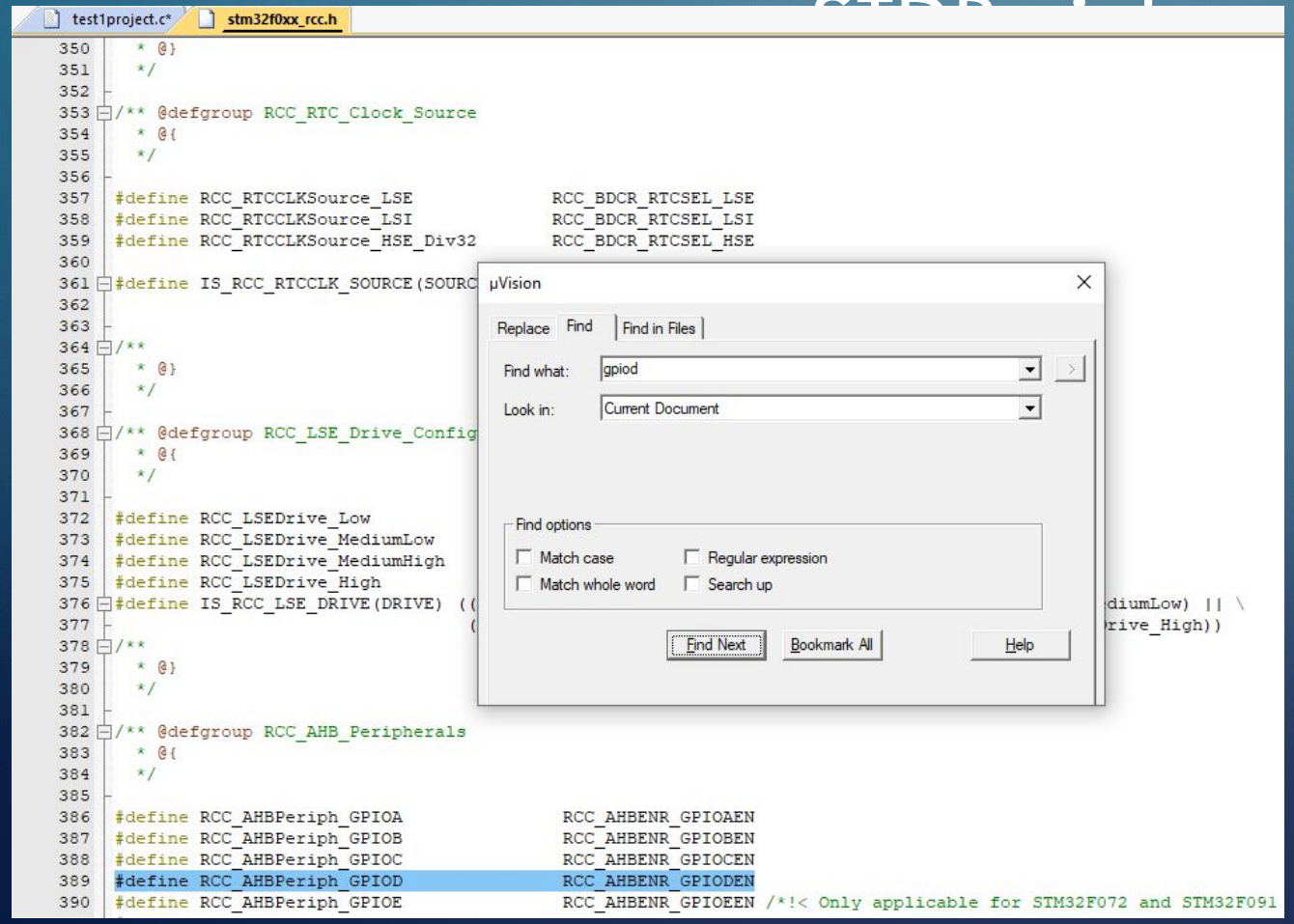
GPIOC->ODR = 0x100; - установка 1

for (int i=0; i<50000; i++){} – задержка (тратим время впустую)

GPIOC->ODR = 0x000; - сброс 1

for (int i=0; i<50000; i++){} - задержка

Использование библиотеки



RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIO**C**, ENABLE);

RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIO**C**, DISABLE);

```
595  void RCC_AHBPeriphClockCmd(uint32_t RCC_AHBPeriph, FunctionalState NewState);
```

Вход/выход

test1project.c* | stm32f0xx_gpio.h

```
34  extern "C" {
35  #endif
36
37  /* Includes ------------------------------------------------------------------
38  #include "stm32f0xx.h"
39
40  /** @addtogroup STM32F0xx_StdPeriph_Driver
41    * @{
42    */
43
44  /** @addtogroup GPIO
45    * @{
46    */
47  /* Exported types ------------------------------------------------------------
48
49  #define IS_GPIO_ALL_PERIPH(PERIPH) (((PERIPH) == GPIOA) || \
50                                      ((PERIPH) == GPIOB) || \
51                                      ((PERIPH) == GPIOC) || \
52                                      ((PERIPH) == GPIOD) || \
53                                      ((PERIPH) == GPIOE) || \
54                                      ((PERIPH) == GPIOF))
55
56  #define IS_GPIO_LIST_PERIPH(PERIPH) (((PERIPH) == GPIOA) || \
57                                       ((PERIPH) == GPIOB))
58
59  /** @defgroup Configuration_Mode_enumeration
60    * @{
61    */
62  typedef enum
63  {
64    GPIO_Mode_IN    = 0x00, /*!< GPIO Input Mode                */
65    GPIO_Mode_OUT   = 0x01, /*!< GPIO Output Mode               */
66    GPIO_Mode_AF    = 0x02, /*!< GPIO Alternate function Mode */
67    GPIO_Mode_AN    = 0x03  /*!< GPIO Analog In/Out Mode        */
68  }GPIOMode_TypeDef;
```


```
64    GPIO_Mode_IN    = 0x00, /*!< GPIO Input Mode                */
65    GPIO_Mode_OUT   = 0x01, /*!< GPIO Output Mode               */
66    GPIO_Mode_AF    = 0x02, /*!< GPIO Alternate function Mode */
67    GPIO_Mode_AN    = 0x03  /*!< GPIO Analog In/Out Mode        */
68  }GPIOMode_TypeDef;
```

```
#include "stm32f0xx.h"

int main(void)
{
```

GPIO_InitTypeDef GPIO_Init_LED; - создаем переменную GPIO_Init_LED с типом GPIO_InitTypeDef

RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIO**C**, ENABLE); - включаем тактирование на порту C

```
GPIO_Init_LED.GPIO_Pin=GPIO_Pin_9;
GPIO_Init_LED.GPIO_Mode = GPIO_Mode_OUT;
GPIO_Init_LED.GPIO_Speed = GPIO_Speed_2MHz;
GPIO_Init_LED.GPIO_OType = GPIO_OType_PP;
GPIO_Init_LED.GPIO_PuPd = GPIO_PuPd_NOPULL;
```

GPIO_Init (GPIO**C**, & GPIO_Init_LED); - инициализируем порт, & GPIO_Init_LED – указатель на нашу **структуру**

```
        while(1)
        {

                GPIO_SetBits(GPIOC, GPIO_Pin_9);
                    for (int i=0; i<50000; i++){}
                GPIO_ResetBits(GPIOC, GPIO_Pin_9);
                    for (int i=0; i<50000; i++){}

        }
```

SysTick

SysTick является частью микропроцессорного ядра Cortex-M

Описан в StartUp файле

SysTick_Config(SystemCoreClock); - инициализация, срабатывание каждую секунду.
SystemCoreClock – число тактов МК в секунду

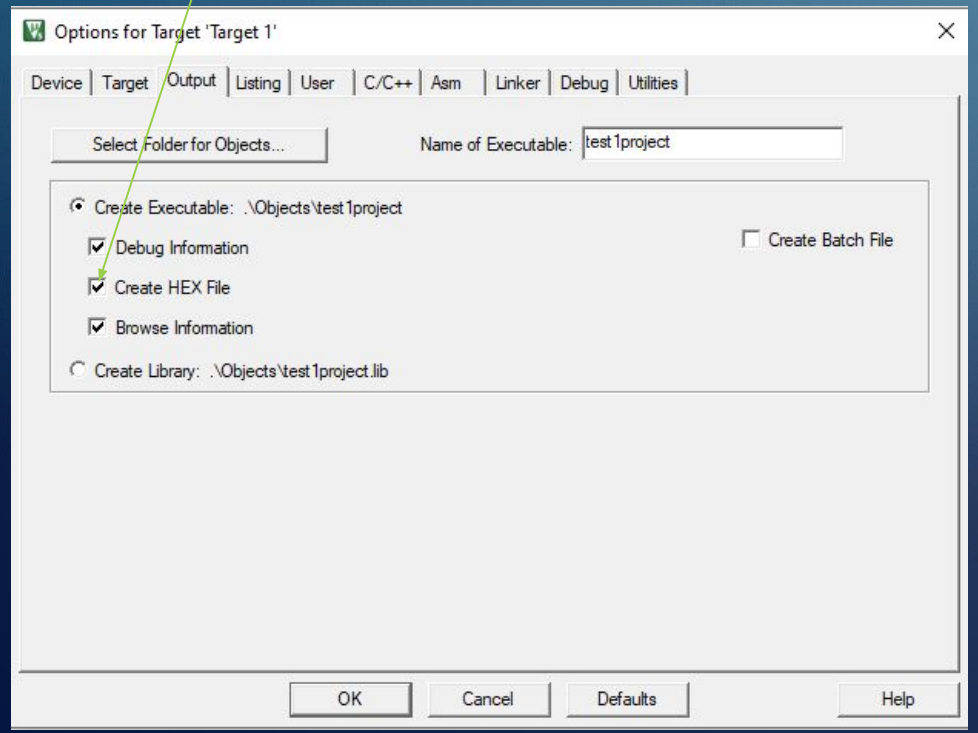
SysTick_Config(SystemCoreClock/1000); - одно срабатывание в миллисекунду

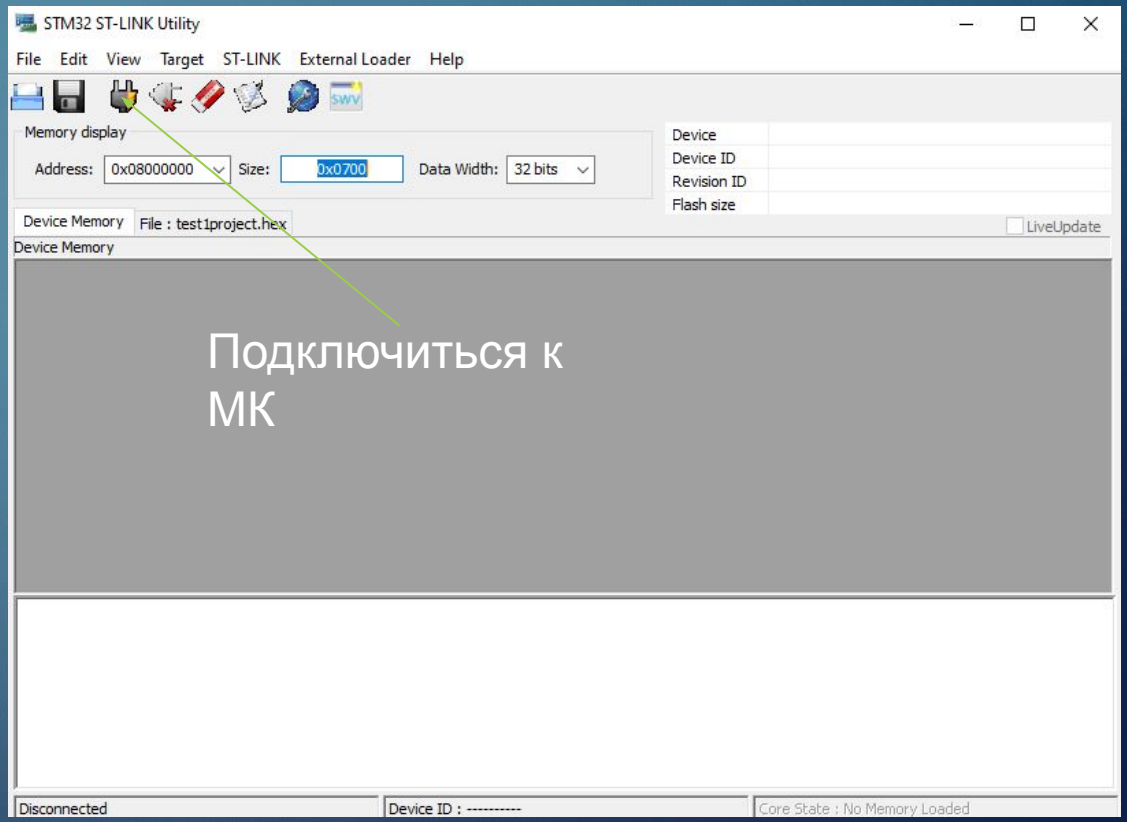
24-битный вычитающий счетчик с функциями автоматической перезагрузки и генерации прерывания

```
#include "stm32f0xx.h"
uint16_t delay_count=0;
void SysTick_Handler(void)
{
        if (delay_count>0)
        {
        delay_count--;
        }
}
void delay_ms(uint16_t delay_temp)
{
delay_count=delay_temp;
while(delay_count>0){}
}
int main(void)
{
GPIO_InitTypeDef GPIO_Init_LED;
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);
SysTick_Config(SystemCoreClock/1000);
GPIO_Init_LED.GPIO_Pin=GPIO_Pin_8;
GPIO_Init_LED.GPIO_Mode = GPIO_Mode_OUT;
GPIO_Init_LED.GPIO_Speed = GPIO_Speed_2MHz;
GPIO_Init_LED.GPIO_OType = GPIO_OType_PP;
GPIO_Init_LED.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_Init (GPIOC, & GPIO_Init_LED);
        while(1)
        {
        GPIO_SetBits(GPIOC, GPIO_Pin_8);
        delay_ms(100);
        GPIO_ResetBits(GPIOC, GPIO_Pin_8);
        delay_ms(100);
        }
}
```

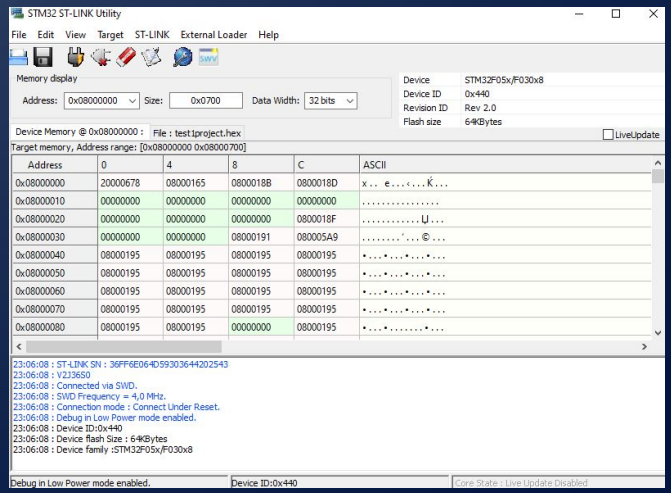
St-link Utility

Для создания HEX файла в настройках проекта KEIL:





Успешное подключение:



Target- Program & Verify (Ctrl+P)

Выбрать HEX файл, нажать OK