Максим Рубченко @maxim_rubchenko
Архитектор .NET

14 декабря 2016 в 13:24

Установка FreeRTOS для Stm32vlDiscovery

tutorial

DIY или Сделай сам*

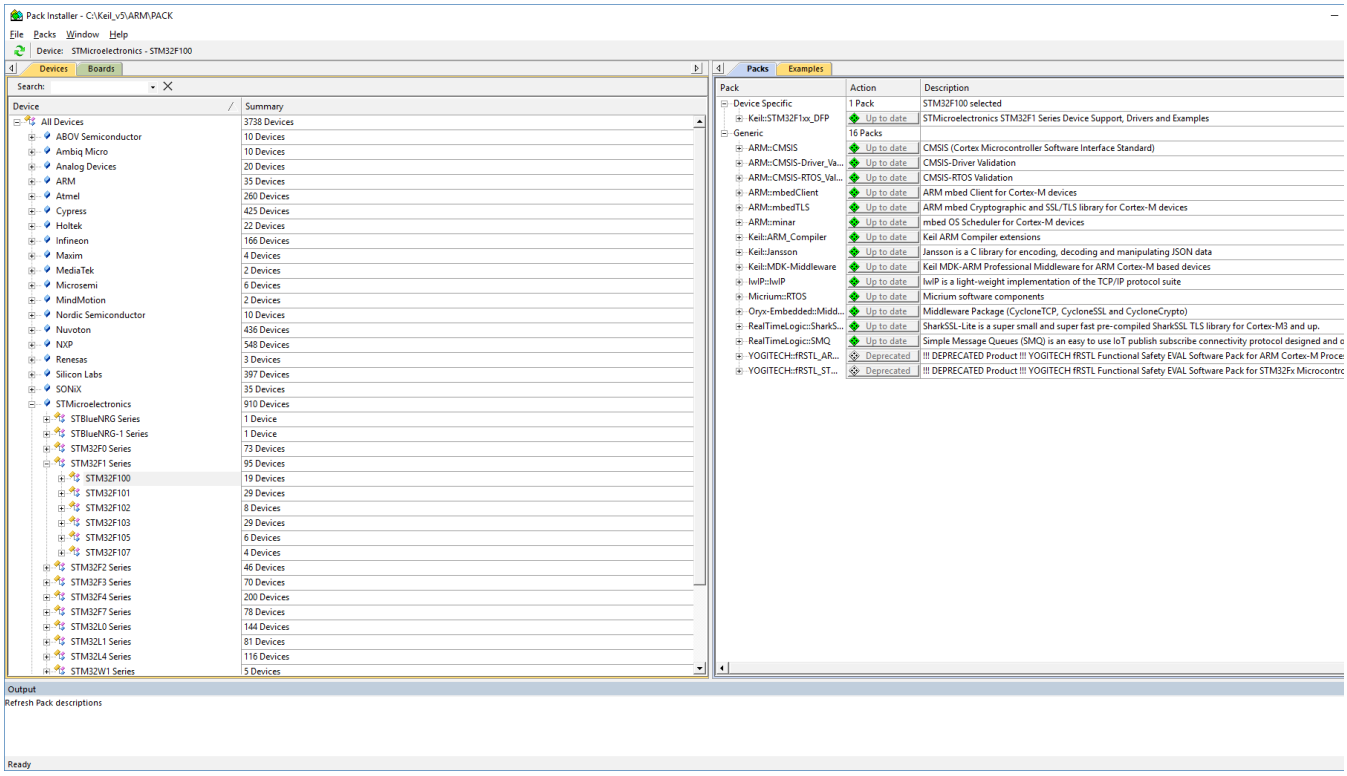
Доброго времени суток.
Разбираясь с программированием микроконтроллеров Stm32, решил попробовать установить (операционную систему реального времени) ОС FreeRTOS и столкнулся с рядом трудностей. В интернете есть множество статей, но внятной инструкции я не нашёл, поэтому всех заинтересовавшихся прошу под кат.

Итак для начала определимся что необходимо иметь:

- Отладочная плата Stm32VLDiscovery
- Установленную IDE Keil ARM 5 и выше
- Желание разбираться в новом

Если вы уже разрабатывали что-либо для семейства STM32F1x в Keil, то можете пропустить этот абзац. Для начала необходимо установить п для разработки для семейства STM32F1x, это делается следующим образом:

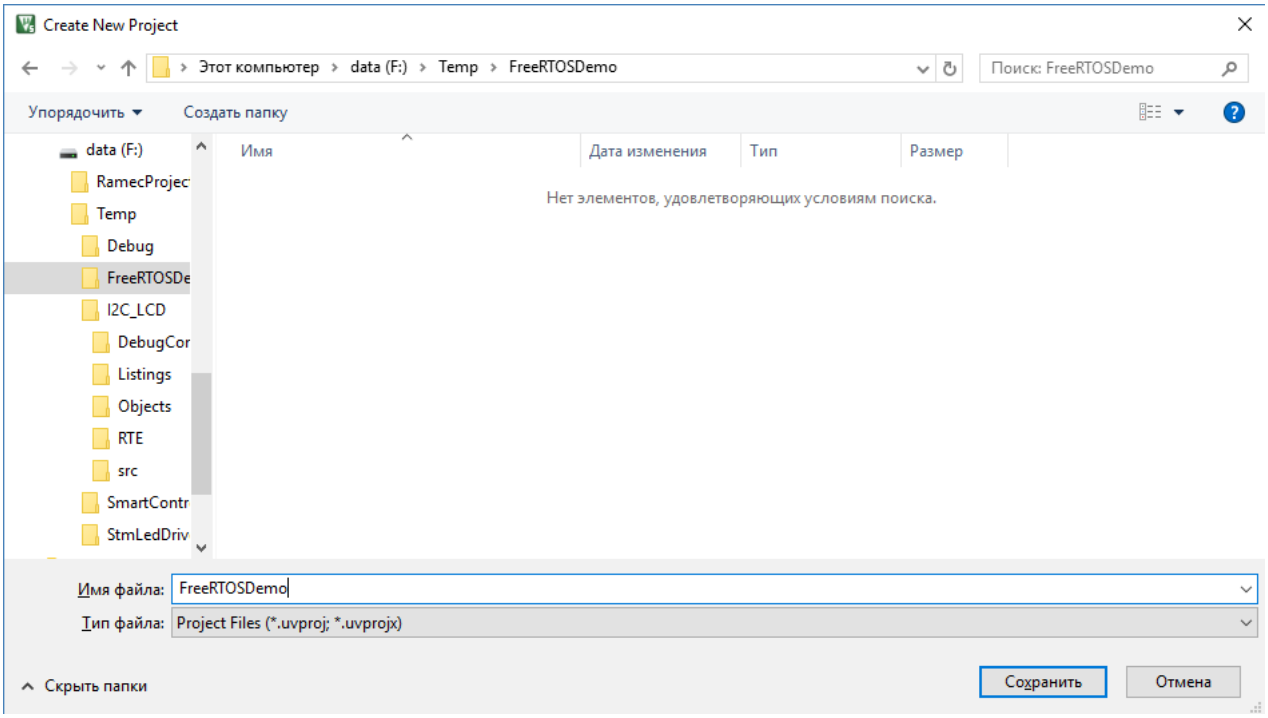
1. Запускаем Keil.
2. Нажимаем на кнопку Pack Installer в панели



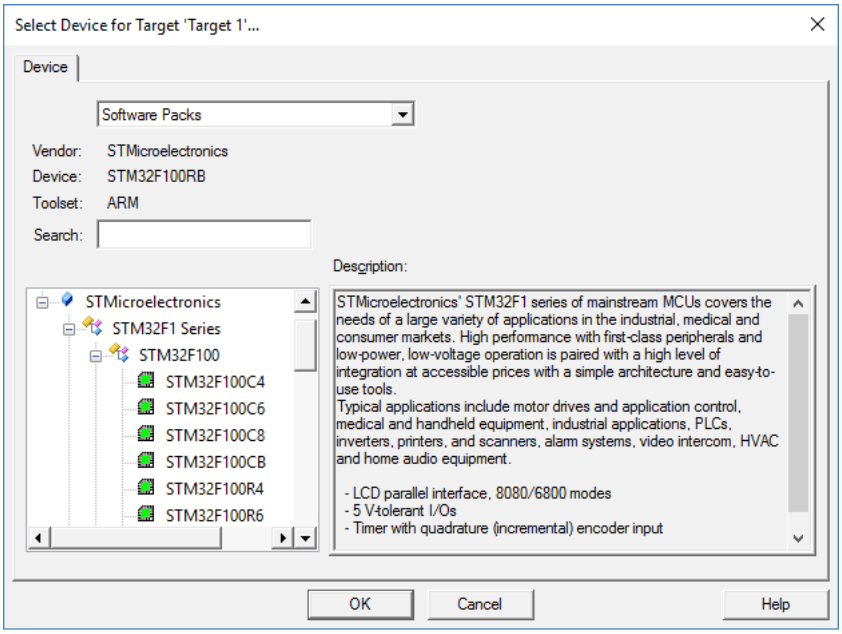
3. В открывшемся окне нажимаем кнопку Check For Updates (ждём обновления списка пакетов)
4. Выбираем в дереве STMicroelectronics->STM32F1 Series->STM32F100 и устанавливаем все пакеты

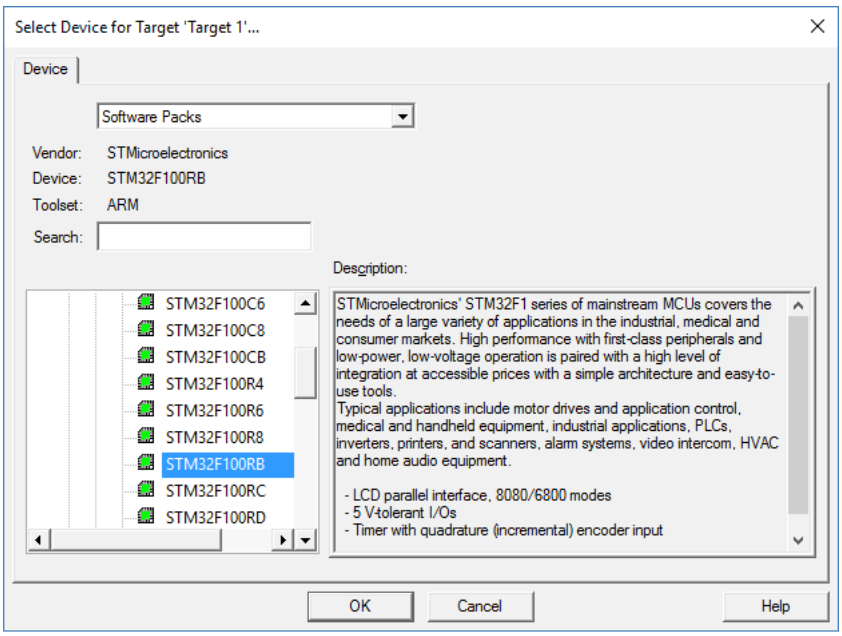
На этом подготовка среды разработки закончена.
Переходим непосредственно к созданию проекта с использованием FreeRTOS.
Первое что необходимо сделать это скачать свежий дистрибутив ОСРВ с официального сайта www.freertos.org, на момент написания статьи последней версией была FreeRTOSv9.0.0. Итак мы получили внушительный архив, распаковываем его в любое место (сразу скажу что 99% нам не понадобится, так что не пугайтесь рамера получившейся папки у меня более 200 Мб).

Теперь нам необходимо создать новый проект в Keil.

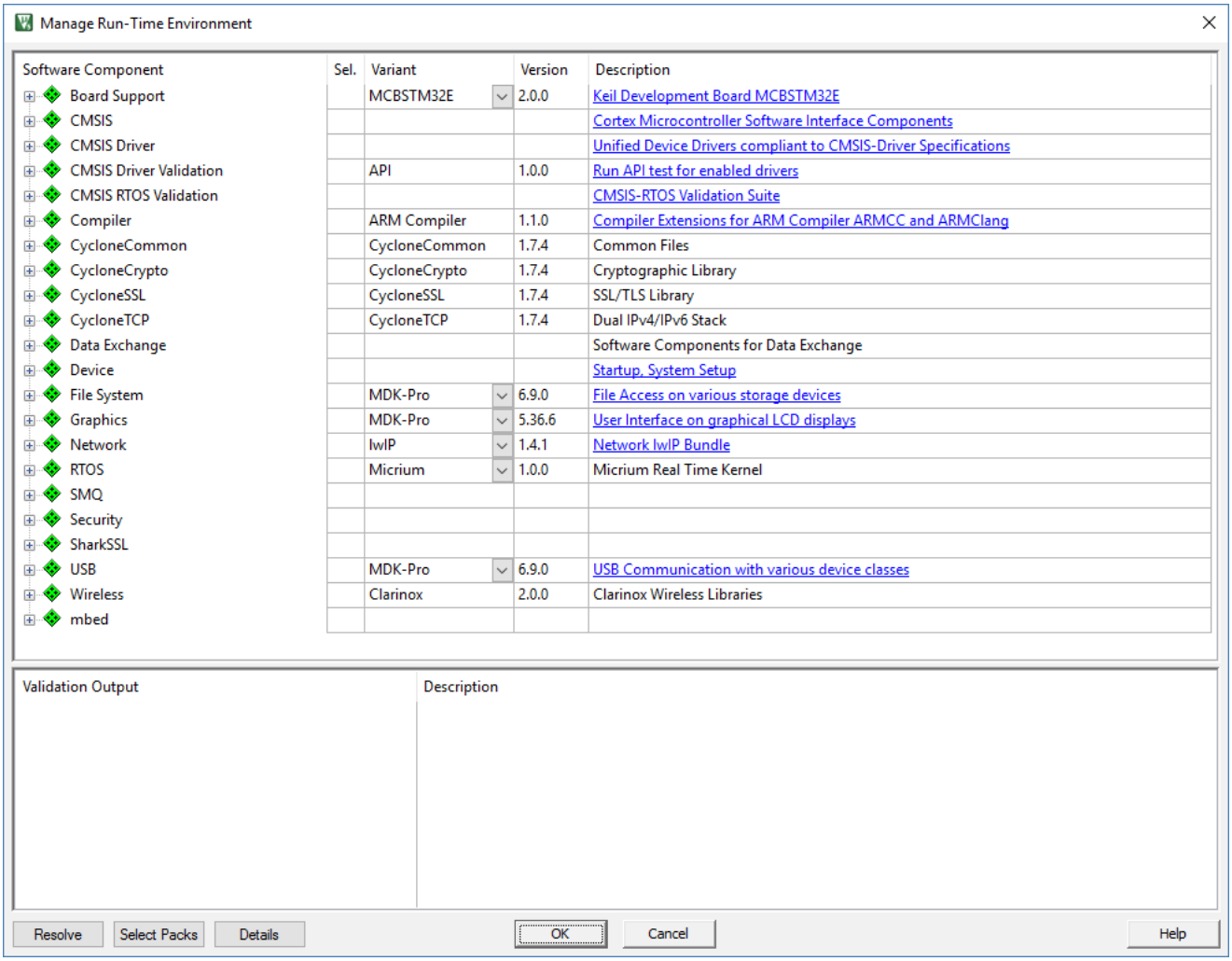


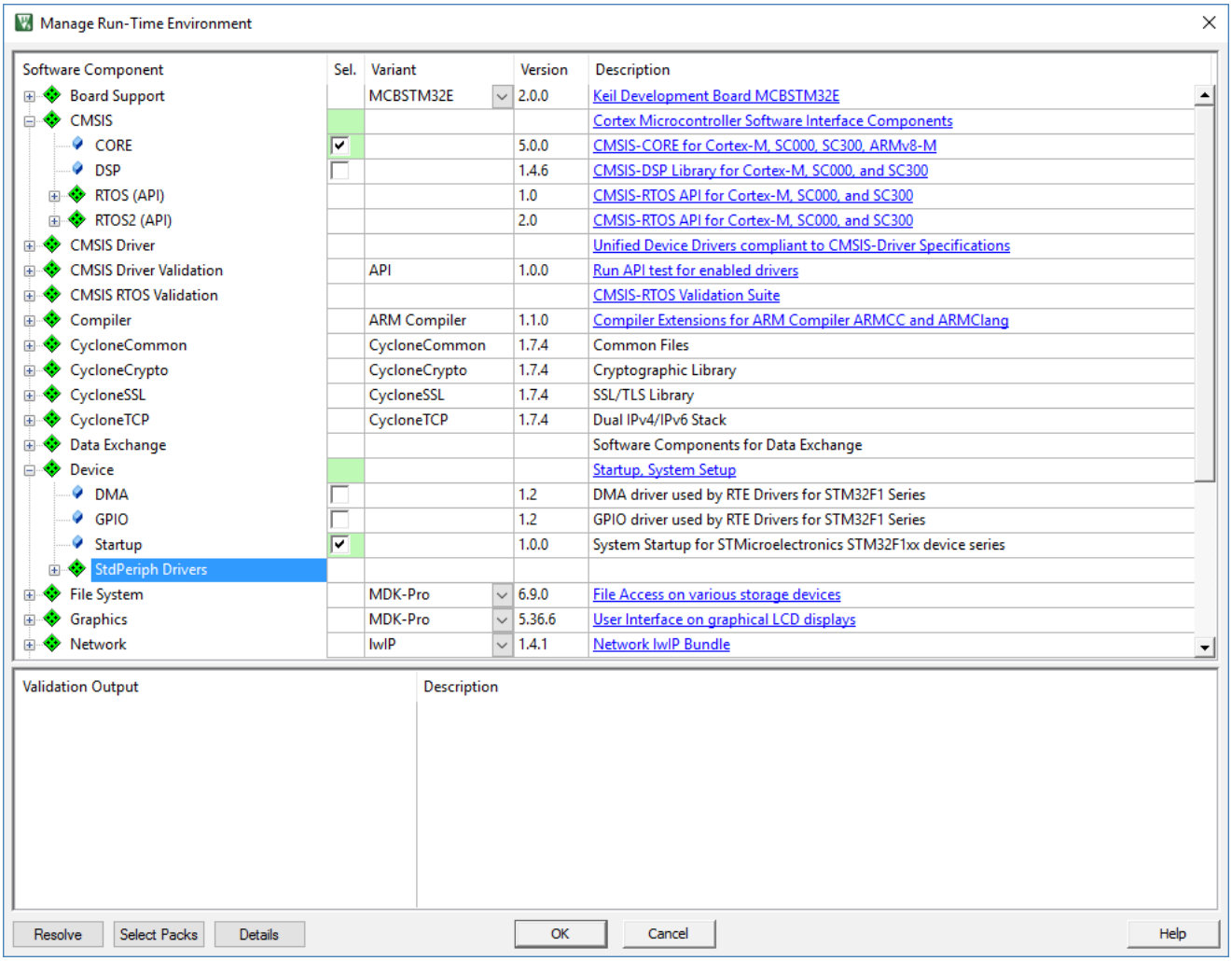
Я назвал его FreeRTOSDemo, вы можете выбрать любое подходящее название, главное чтобы в пути размещения проекта не было пробелов русских букв.
Теперь нам необходимо выбрать наш чип, я тестировал на отладочной плате STM32VLDiscovery с чипом STM32F100RB соответственно выбир именно его.

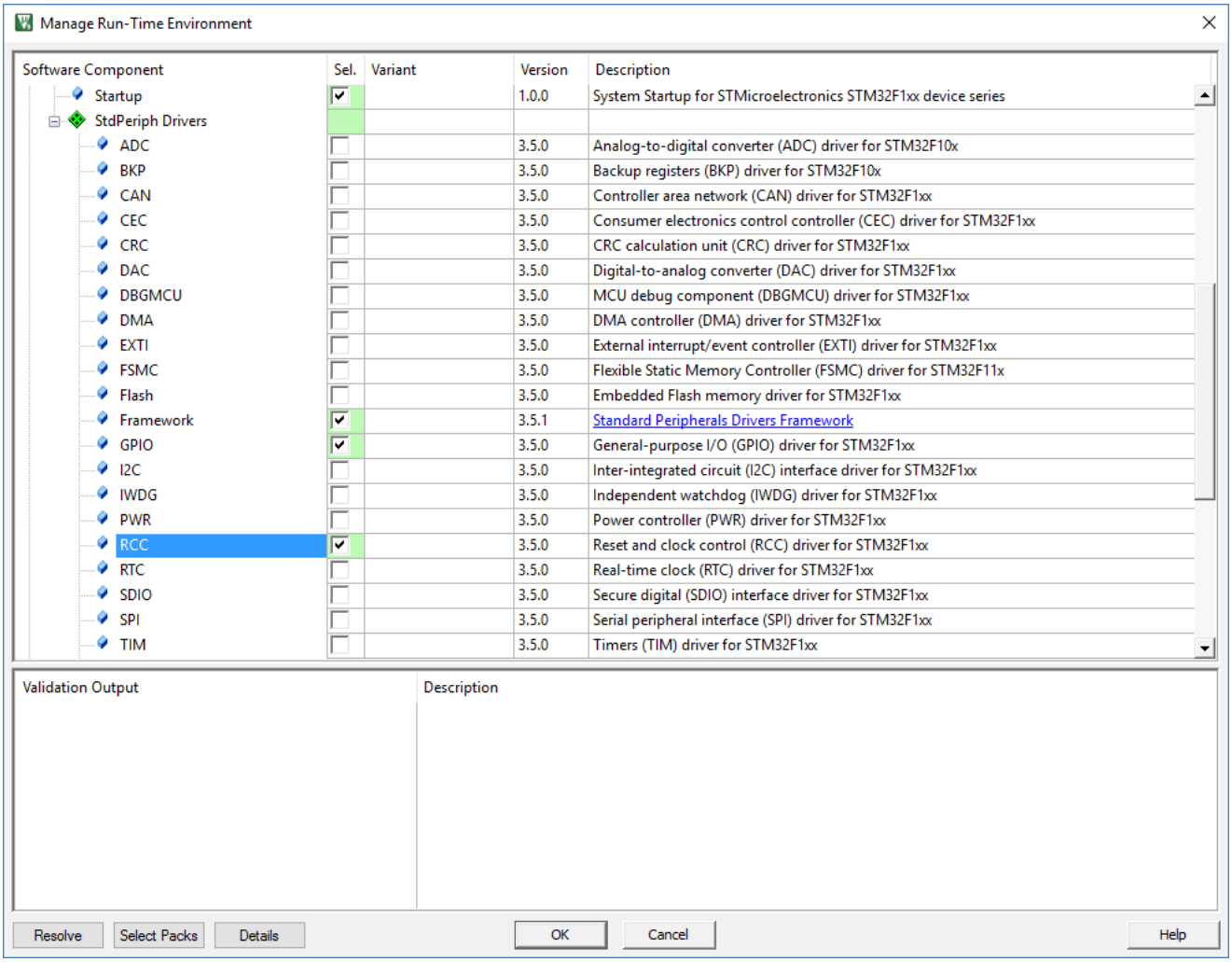




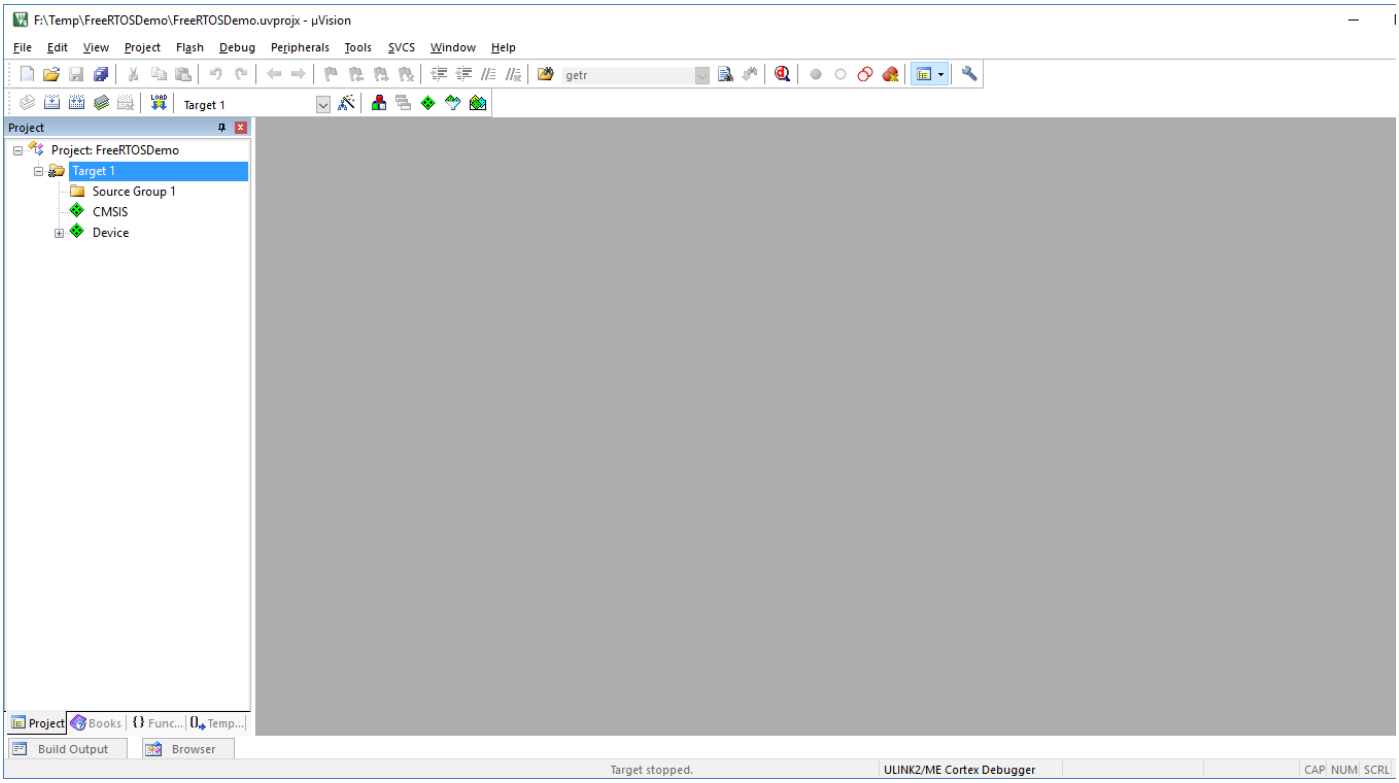
После выбора чипа, необходимо выбрать библиотеки которые мы будем использовать. Нам понадобится библиотека CMSIS и StdPeriph. На рисунках далее показан минимальный выбор компонентов, чтобы проверить работу ОСРВ и помигать диодиками (это касается библиотеки StdPeriph, CMSIS нужна обязательно).



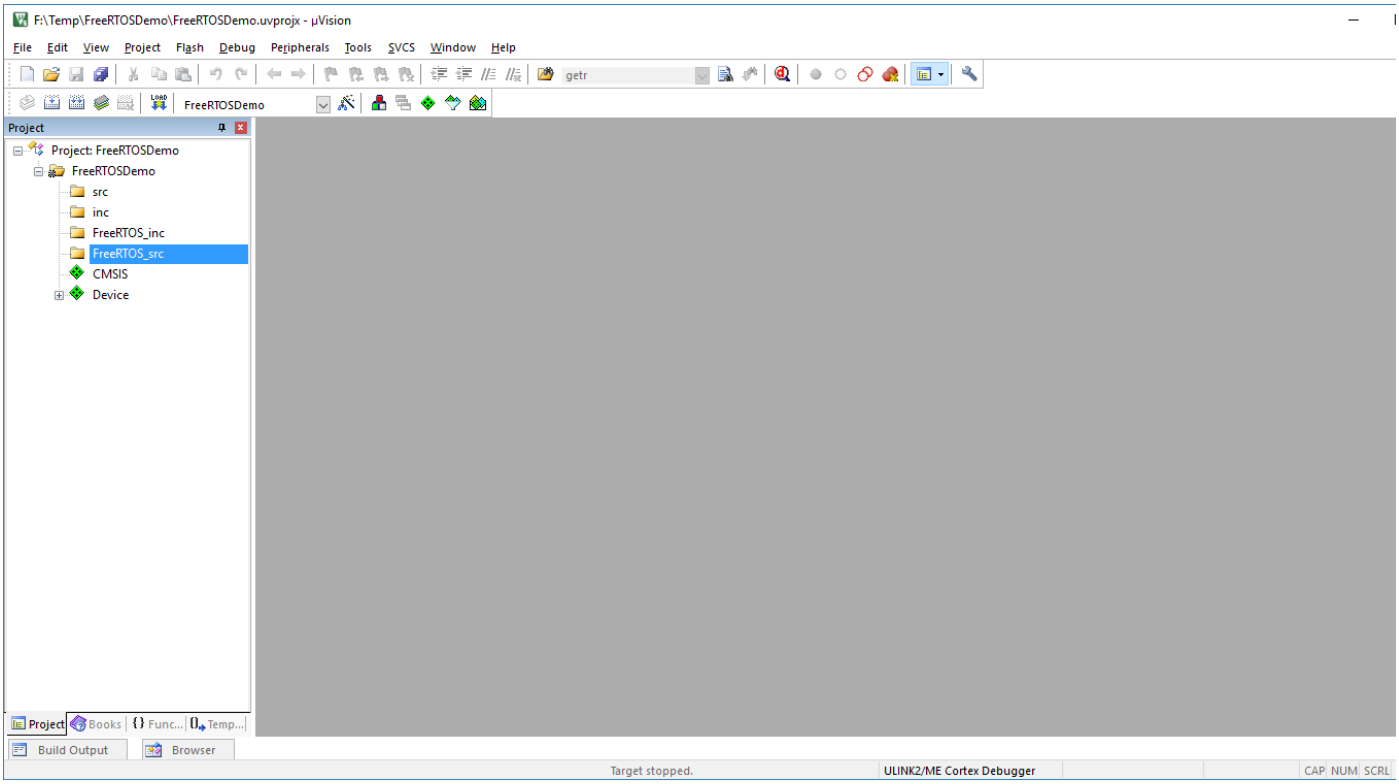

Manage Run-Time Environment
Software Component
Sel.
Variant
Version
Description
Board Support
MCBSTM32E
2.0.0
Keil Development Board MCBSTM32E
CMSIS
Cortex Microcontroller Software Interface Components
CORE
5.0.0
CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M
DSP
1.4.6
CMSIS-DSP Library for Cortex-M, SC000, and SC300
RTOS (API)
1.0
CMSIS-RTOS API for Cortex-M, SC000, and SC300
RTOS2 (API)
2.0
CMSIS-RTOS API for Cortex-M, SC000, and SC300
CMSIS Driver
Unified Device Drivers compliant to CMSIS-Driver Specifications
CMSIS Driver Validation
API
1.0.0
Run API test for enabled drivers
CMSIS RTOS Validation
CMSIS-RTOS Validation Suite
Compiler
ARM Compiler
1.1.0
Compiler Extensions for ARM Compiler ARMCC and ARMClang
CycloneCommon
CycloneCommon
1.7.4
Common Files
CycloneCrypto
CycloneCrypto
1.7.4
Cryptographic Library
CycloneSSL
CycloneSSL
1.7.4
SSL/TLS Library
CycloneTCP
CycloneTCP
1.7.4
Dual IPv4/IPv6 Stack
Data Exchange
Software Components for Data Exchange
Device
Startup, System Setup
DMA
1.2
DMA driver used by RTE Drivers for STM32F1 Series
GPIO
1.2
GPIO driver used by RTE Drivers for STM32F1 Series
Startup
1.0.0
System Startup for STMicroelectronics STM32F1xx device series
StdPeriph Drivers
File System
MDK-Pro
6.9.0
File Access on various storage devices
Graphics
MDK-Pro
5.36.6
User Interface on graphical LCD displays
Network
lwIP
1.4.1
Network lwIP Bundle
Validation Output
Description
Resolve
Select Packs
Details
OK
Cancel
Help


Manage Run-Time Environment
Software Component
Sel.
Variant
Version
Description
Startup
1.0.0
System Startup for STMicroelectronics STM32F1xx device series
StdPeriph Drivers
ADC
3.5.0
Analog-to-digital converter (ADC) driver for STM32F10x
BKP
3.5.0
Backup registers (BKP) driver for STM32F10x
CAN
3.5.0
Controller area network (CAN) driver for STM32F1xx
CEC
3.5.0
Consumer electronics control controller (CEC) driver for STM32F1xx
CRC
3.5.0
CRC calculation unit (CRC) driver for STM32F1xx
DAC
3.5.0
Digital-to-analog converter (DAC) driver for STM32F1xx
DBGMCU
3.5.0
MCU debug component (DBGMCU) driver for STM32F1xx
DMA
3.5.0
DMA controller (DMA) driver for STM32F1xx
EXTI
3.5.0
External interrupt/event controller (EXTI) driver for STM32F1xx
FSMC
3.5.0
Flexible Static Memory Controller (FSMC) driver for STM32F11x
Flash
3.5.0
Embedded Flash memory driver for STM32F1xx
Framework
3.5.1
Standard Peripherals Drivers Framework
GPIO
3.5.0
General-purpose I/O (GPIO) driver for STM32F1xx
I2C
3.5.0
Inter-integrated circuit (I2C) interface driver for STM32F1xx
IWDG
3.5.0
Independent watchdog (IWDG) driver for STM32F1xx
PWR
3.5.0
Power controller (PWR) driver for STM32F1xx
RCC
3.5.0
Reset and clock control (RCC) driver for STM32F1xx
RTC
3.5.0
Real-time clock (RTC) driver for STM32F1xx
SDIO
3.5.0
Secure digital (SDIO) interface driver for STM32F1xx
SPI
3.5.0
Serial peripheral interface (SPI) driver for STM32F1xx
TIM
3.5.0
Timers (TIM) driver for STM32F1xx
Validation Output
Description
Resolve
Select Packs
Details
OK
Cancel
Help

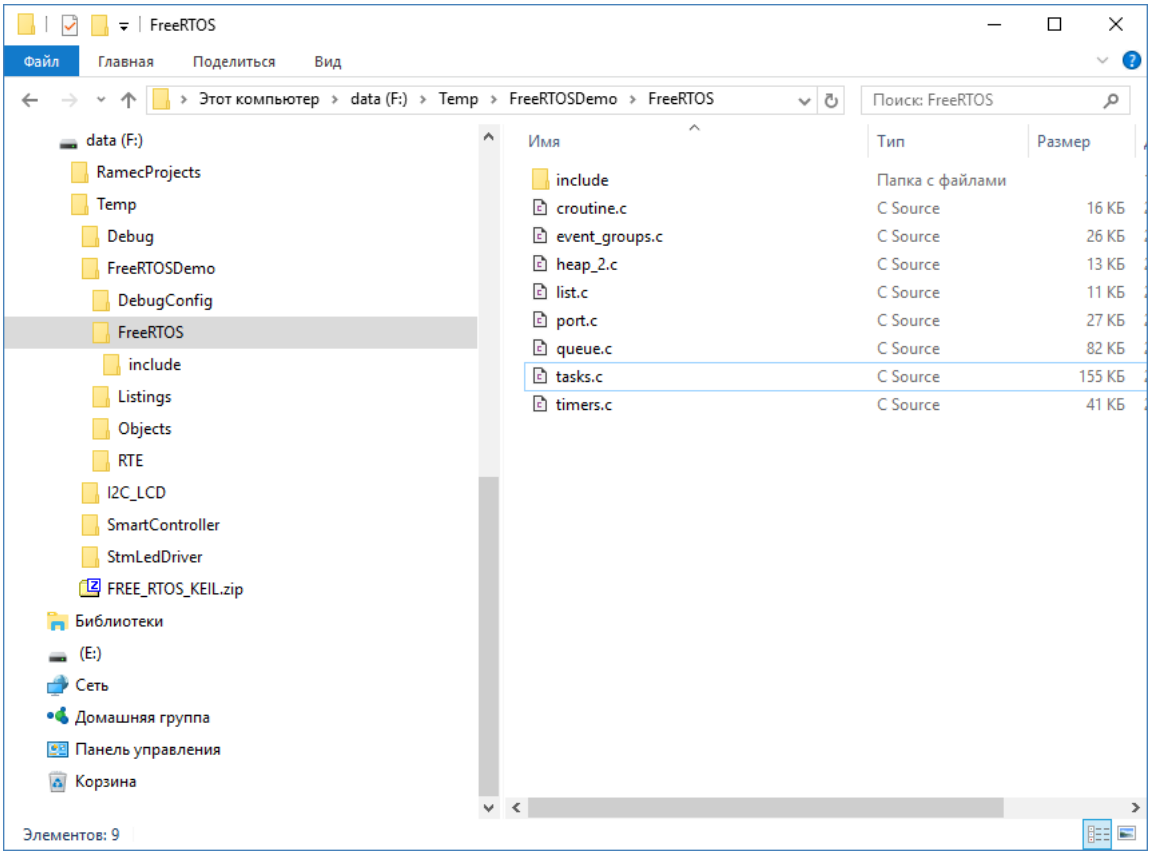
Итак мы создали проект, он выглядит следующим образом:

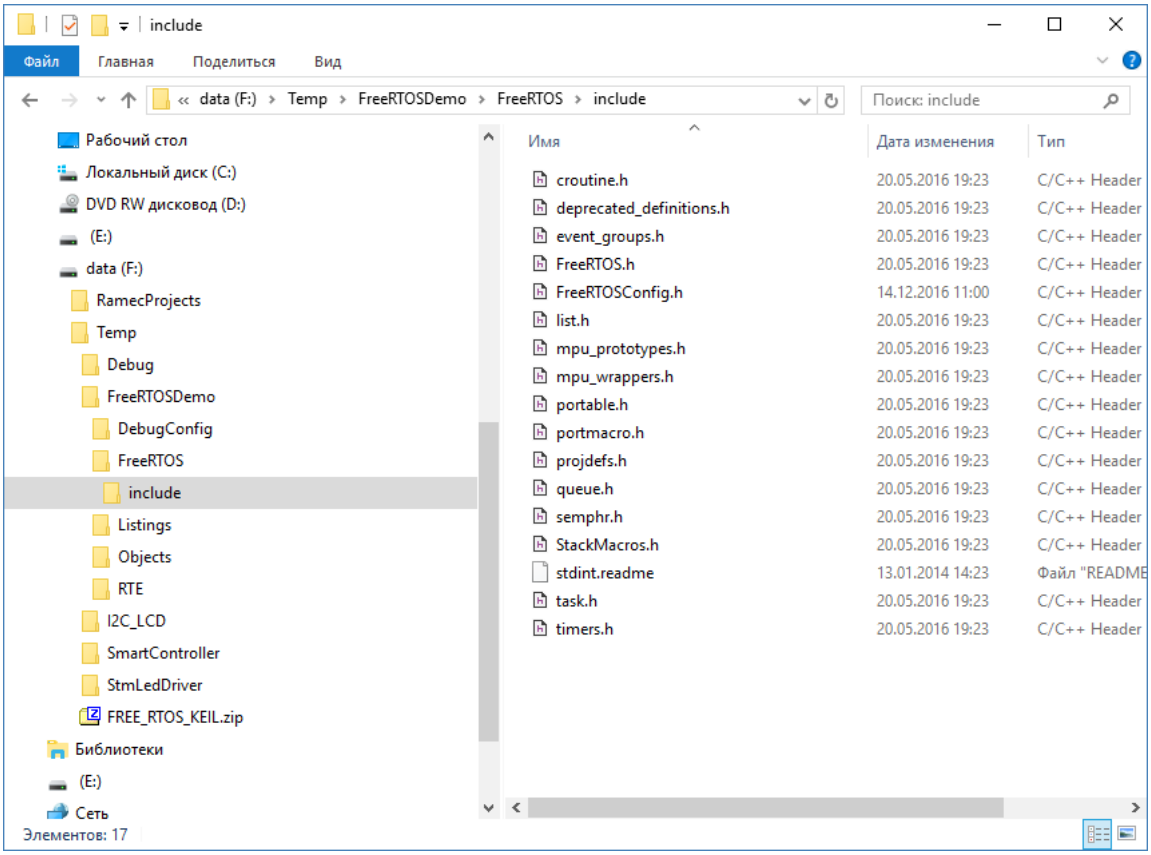


Далее для удобства в проекте создаем несколько групп, чтобы в последствии не получилась мешанина из файлов.

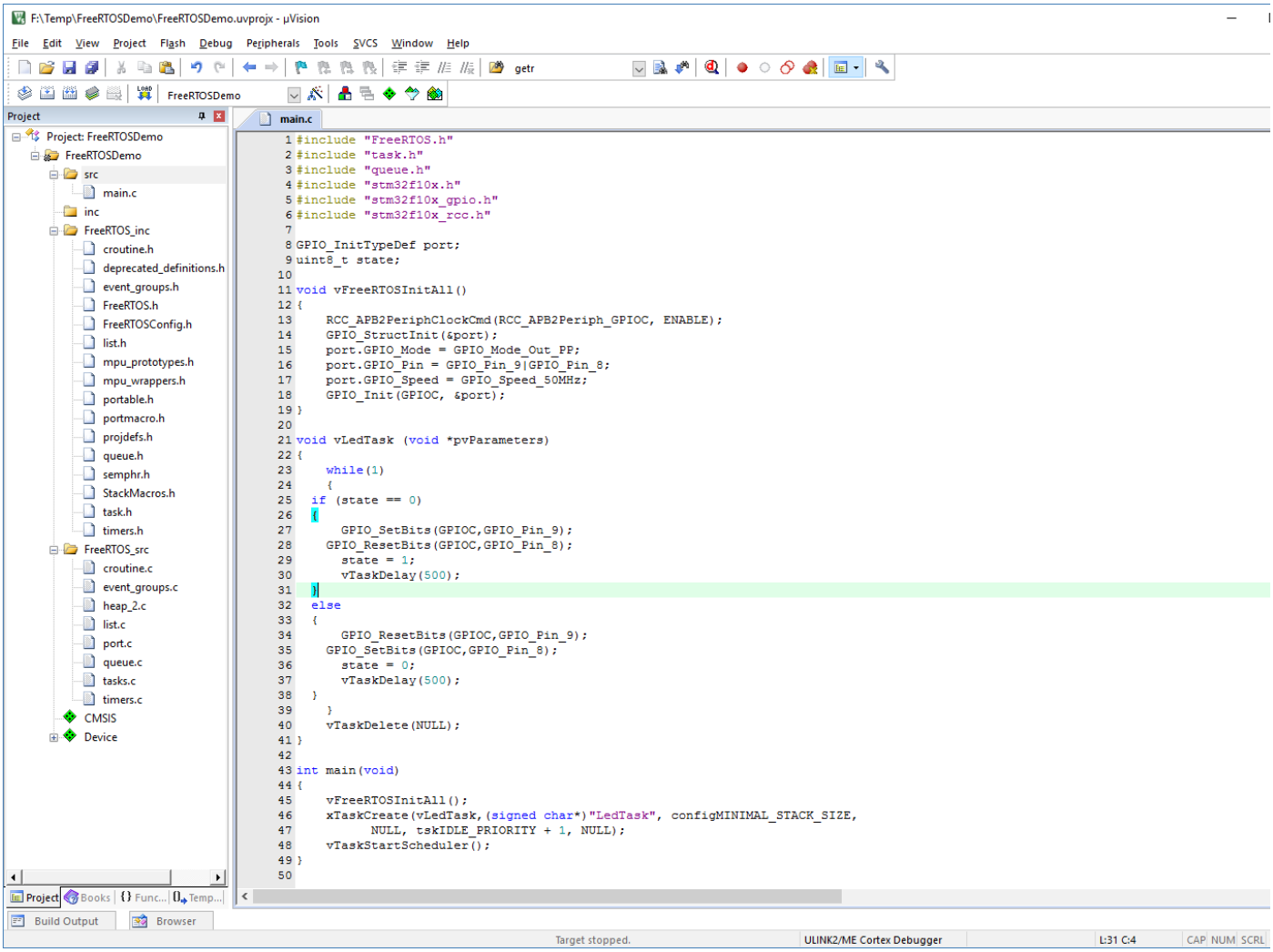


Всё готово к копированию файлов ОСРВ в проект. Создадим папку FreeRTOS в папке проекта.
Из папки FreeRTOSv9.0.0\FreeRTOS\Source\ копируем все *.c файлы в созданную папку FreeRTOS. Туда же копируем папку include.
Из папки FreeRTOSv9.0.0\FreeRTOS\Source\portable\RVDS\ARM_CM3\ копируем файлы *.c и *.h в соответвующие им места папки FreeRTOS н
проекта.
Из папки FreeRTOSv9.0.0\FreeRTOS\Source\portable\MemMang\ копируем файл heap_2.c.
Из папки FreeRTOSv9.0.0\FreeRTOS\Demo\CORTEX_STM32F103_Keil\ копируем файл FreeRTOSConfig.h.
После всех манипуляций должно получится следующее:

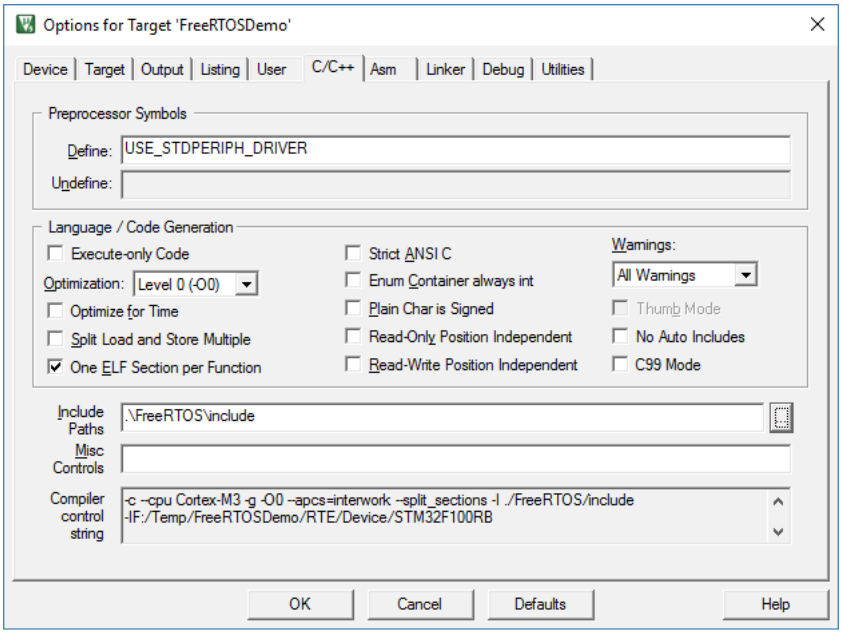




Далее нам необходимо добавить файлы в соответствующие группы в проекте, а также создать файл main.c с основным текстом программы. В итоге должна получится следующая структура проекта.



Для того чтобы проект компилировался в его настройках необходимо указать все пути к *.h файлам, а также прописать дерективу USE_STDPERIPH_DRIVER.



Теперь осталось изменить несколько строк в файле FreeRTOSConfig.h, для корректной работы:

```
#define configCPU_CLOCK_HZ                    ( ( unsigned long ) 72000000 )  /*на*/
#define configCPU_CLOCK_HZ                    ( ( unsigned long ) 24000000 )

#define configTOTAL_HEAP_SIZE          ( ( size_t ) ( 17 * 1024 ) )   /*на*/
#define configTOTAL_HEAP_SIZE          ( ( size_t ) ( 5 * 1024 ) )
```

И добавляем следующие строки после #define FREERTOS_CONFIG_H

```
#define xPortSysTickHandler SysTick_Handler
#define xPortPendSVHandler PendSV_Handler
#define vPortSVCHandler SVC_Handler
```

После этого в файле main.c пишем простейшую программу для мигания светодиодами.

```
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "stm32f10x.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"

GPIO_InitTypeDef port;
uint8_t state;

void vFreeRTOSInitAll()
{
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
    GPIO_StructInit(&port);
    port.GPIO_Mode = GPIO_Mode_Out_PP;
    port.GPIO_Pin = GPIO_Pin_9|GPIO_Pin_8;
    port.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &port);
}

void vLedTask (void *pvParameters)
{
    while(1)
    {
        if (state == 0)
        {
      GPIO_SetBits(GPIOC,GPIO_Pin_9);
                GPIO_ResetBits(GPIOC,GPIO_Pin_8);
            state = 1;
                        vTaskDelay(500);
        }
        else
        {
            GPIO_ResetBits(GPIOC,GPIO_Pin_9);
                GPIO_SetBits(GPIOC,GPIO_Pin_8);
            state = 0;
                        vTaskDelay(500);
        }
    }
    vTaskDelete(NULL);
}

int main(void)
{
                vFreeRTOSInitAll();
    xTaskCreate(vLedTask,(signed char*)"LedTask", configMINIMAL_STACK_SIZE,
                                        NULL, tskIDLE_PRIORITY + 1, NULL);
    vTaskStartScheduler();
}
```

Теперь можно собрать проект и залить прошивку в микроконтроллер, после перезагрузки наблюдать мигающие светодиоды.

stm32vldiscovery, freertos