

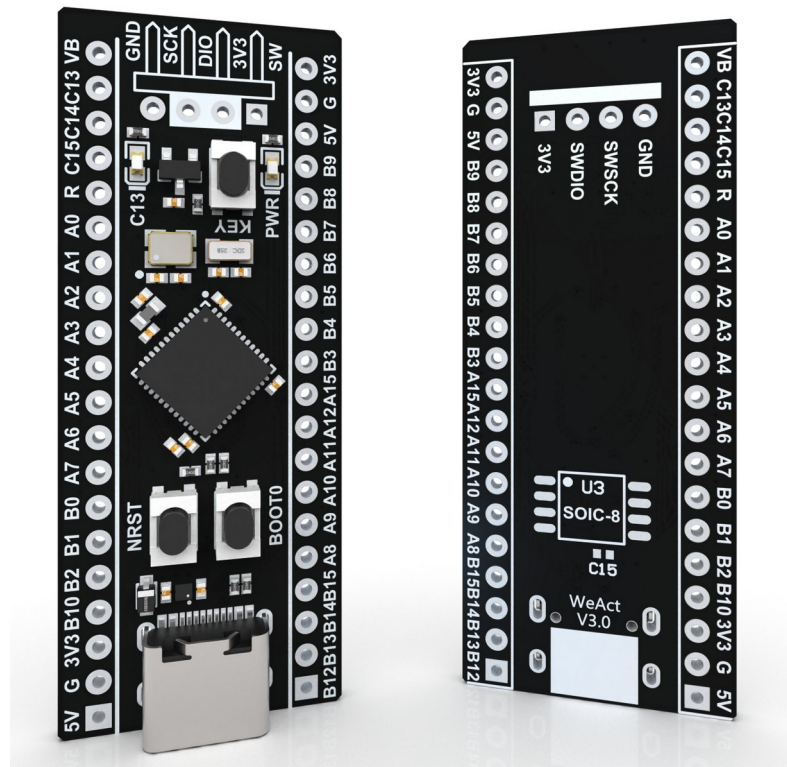
Обмен цифровой информацией

Шины и интерфейсы

Знакомство с инструментами и мигание лампочкой

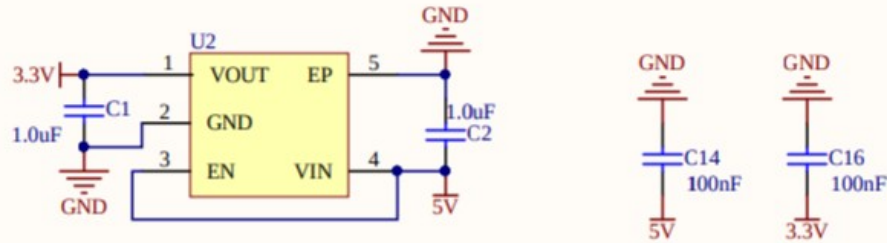
Плата WeActTC/MiniSTM32F4x1

- У нас вариант с STM32F411CEU6 на борту
- Плата имеет документацию!
<https://github.com/WeActTC/MiniSTM32F4x1>

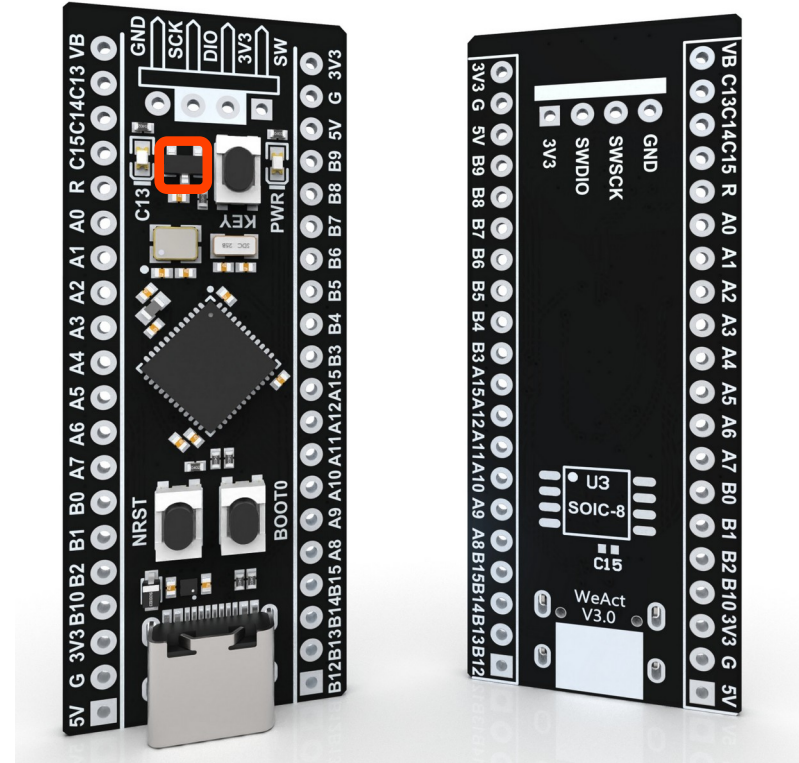


Что есть на плате

Линейный регулятор на 3.3 вольта



5V -> 3.3V 电源转换



Что есть на плате

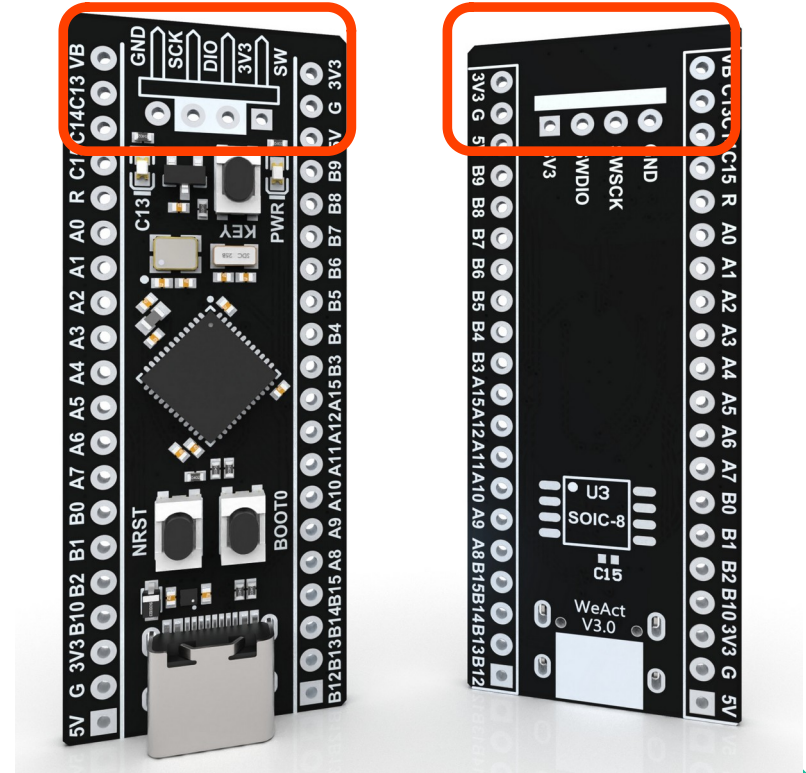
Разъем для программатора (Serial Wire Debug = SWD)

3V3 — питание 3.3В

GND — нулевой провод (земля)

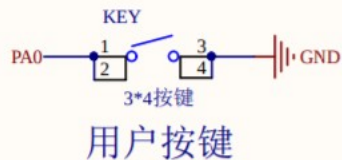
SCK — синхросигнал SWD

DIO — данные SWD

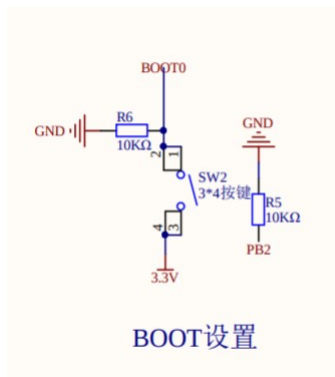


Что есть на плате

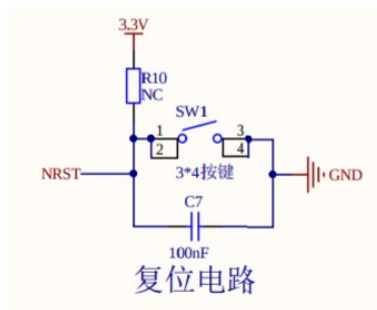
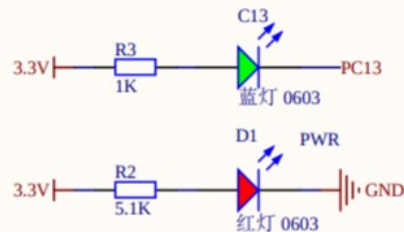
Лампочки и кнопки



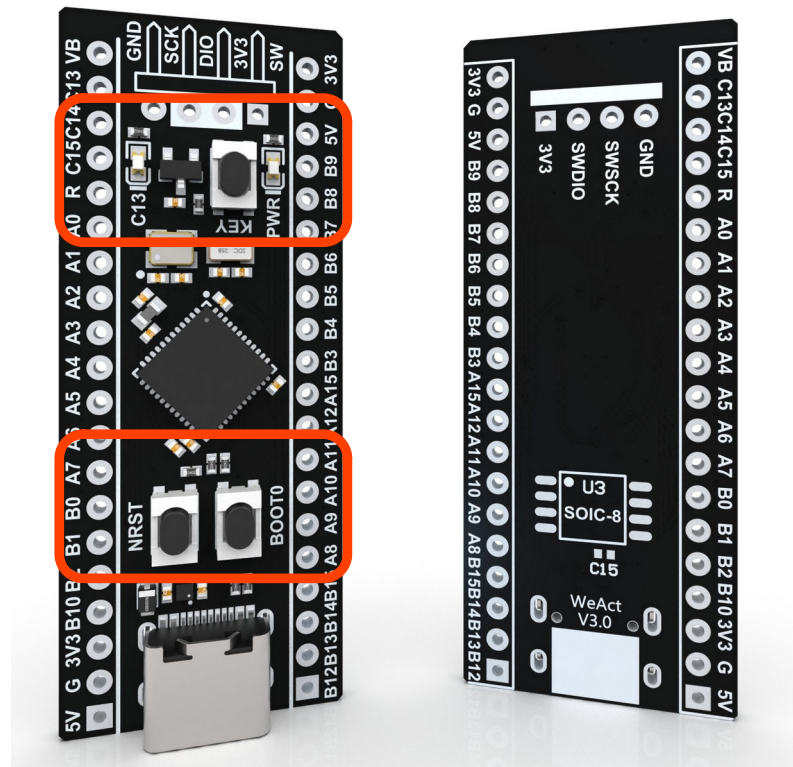
V1.3 增加用户按键 KEY



BOOT设置



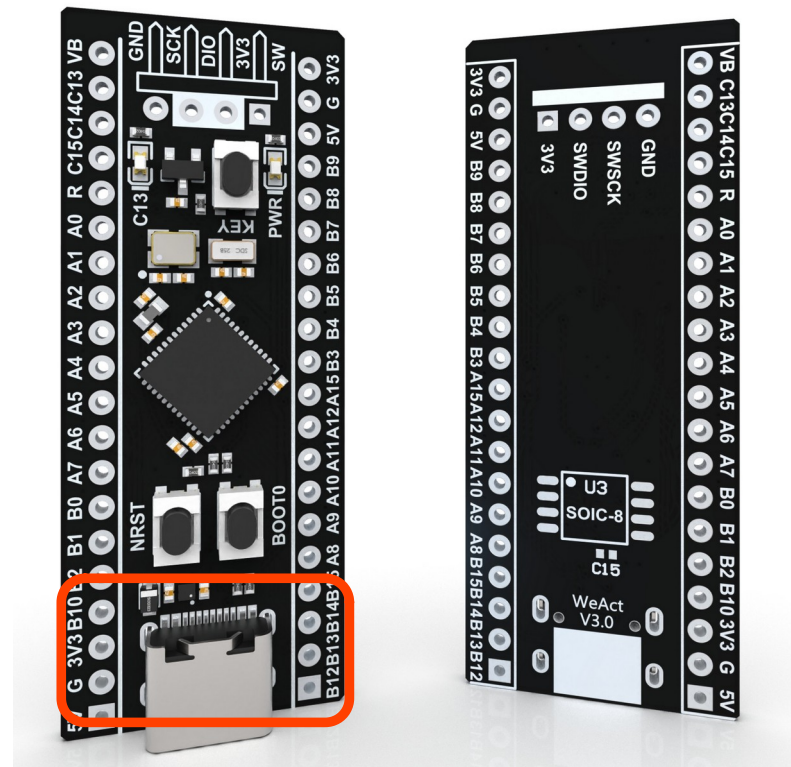
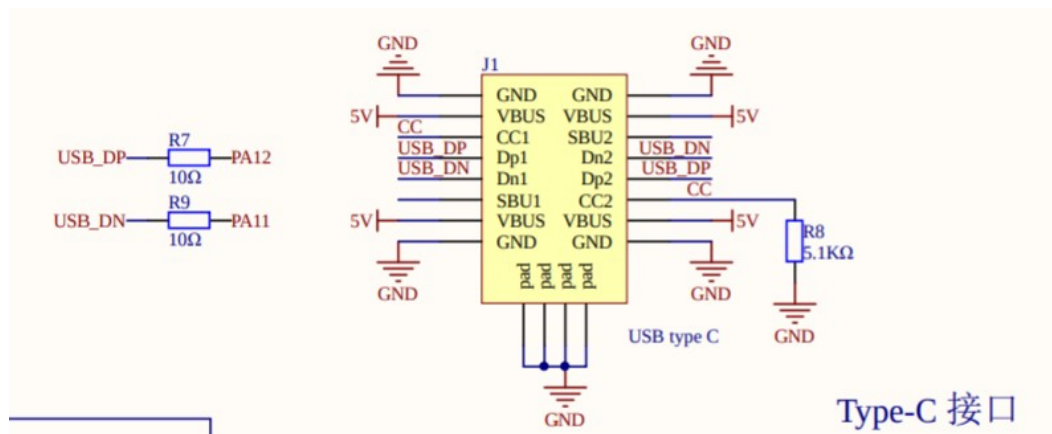
复位电路



Что есть на плате

USB OTG Full Speed (2.0) Type-C

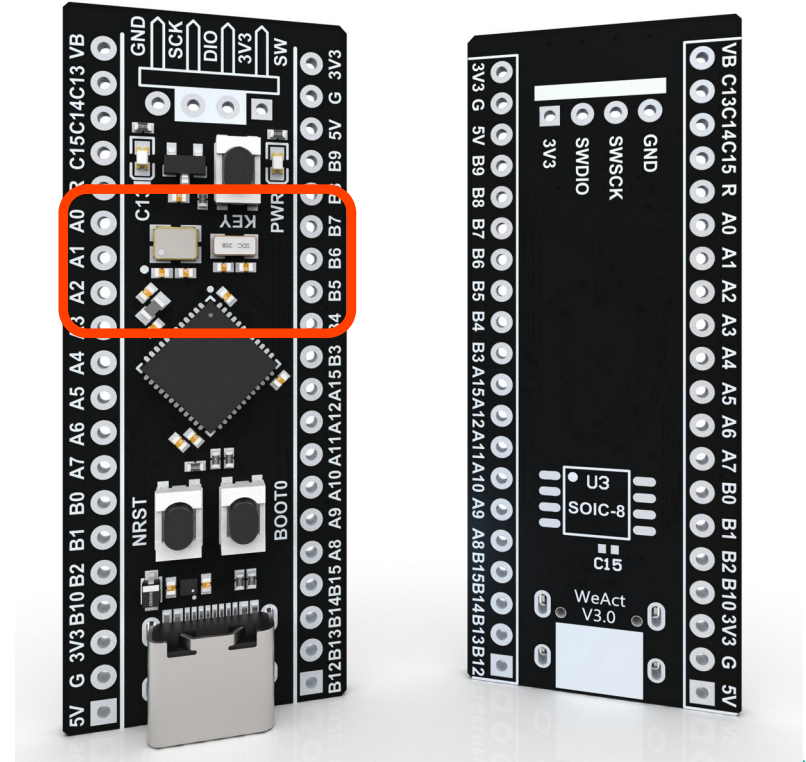
Можно брать оттуда питание например



Что есть на плате

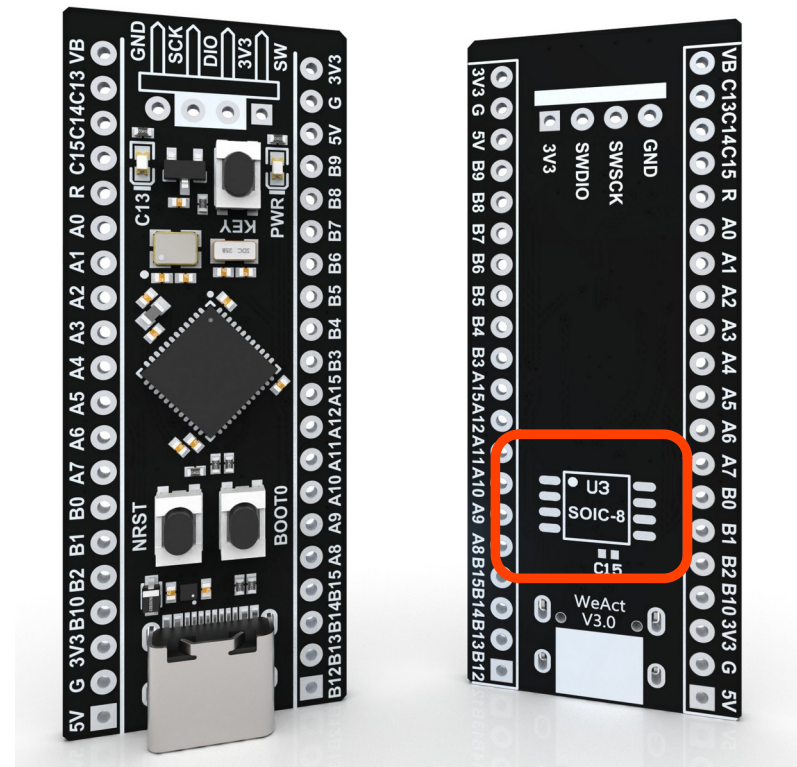
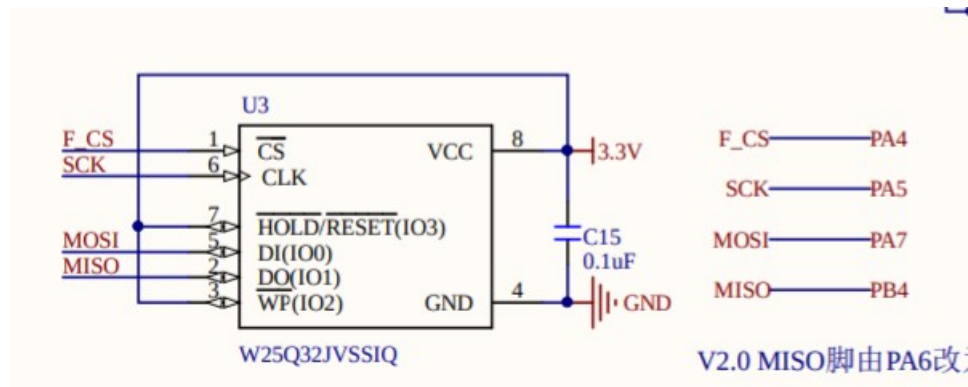
Кварцевые резонаторы

HSE (High Speed External) на 25 МГц
LSE (Low Speed External) на 32.768 КГц



Что есть на плате

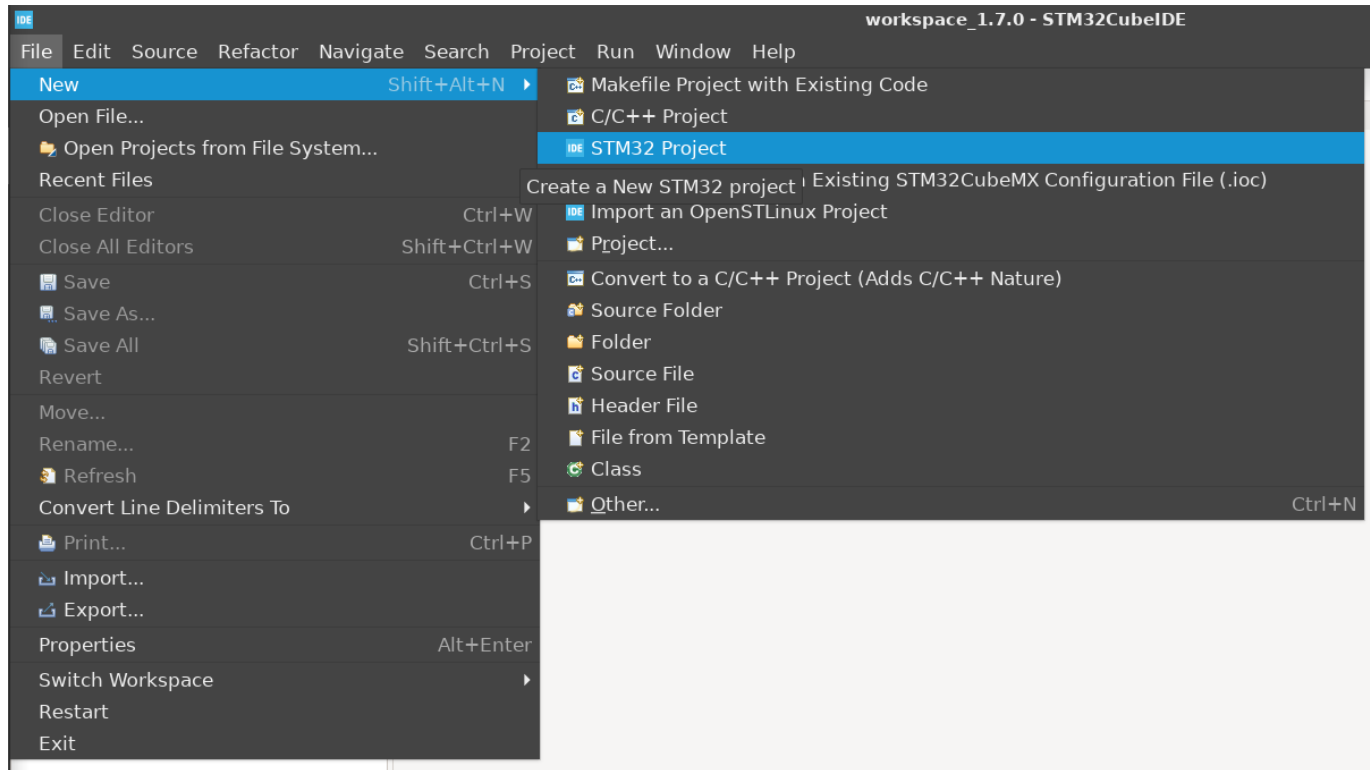
Место для флешки



STM32CubeIDE



Создание проекта



Создание проекта

STM32 Project

Target Selection

Select STM32 target or STM32Cube example

MCU/MPU Selector Board Selector Example Selector Cross Selector

MCU/MPU Filters

Part Number: STM32F411CE

Core >

Series >

Line >

Package >

Other >

Peripheral >

Features

Block Diagram

Docs & Resources

Datasheet

Buy

STM32F4 Series

STM32F411CE

High-performance access line, Arm Cortex-M4 core with DSP and FPU, 512 Kbytes of Flash memory, 100 MHz CPU, ART Accelerator

ACTIVE Active

Product is in mass

Unit Price for 10kU (US\$) : 3.382

MCUs/MPUs List: 2 items

+ Display similar items

Export

*	Part No	Reference	Mar...	Unit Pric...	Board	Package	Flash	RAM	IO	Freq.
☆	STM32F411CE	STM32F411CEUx	Active	3.382		UFQFPN48	512 kBytes	128 kBytes	36	100 MHz
☆	STM32F411CE	STM32F411CEYx	Active	3.382		WLCSP49	512 kBytes	128 kBytes	36	100 MHz

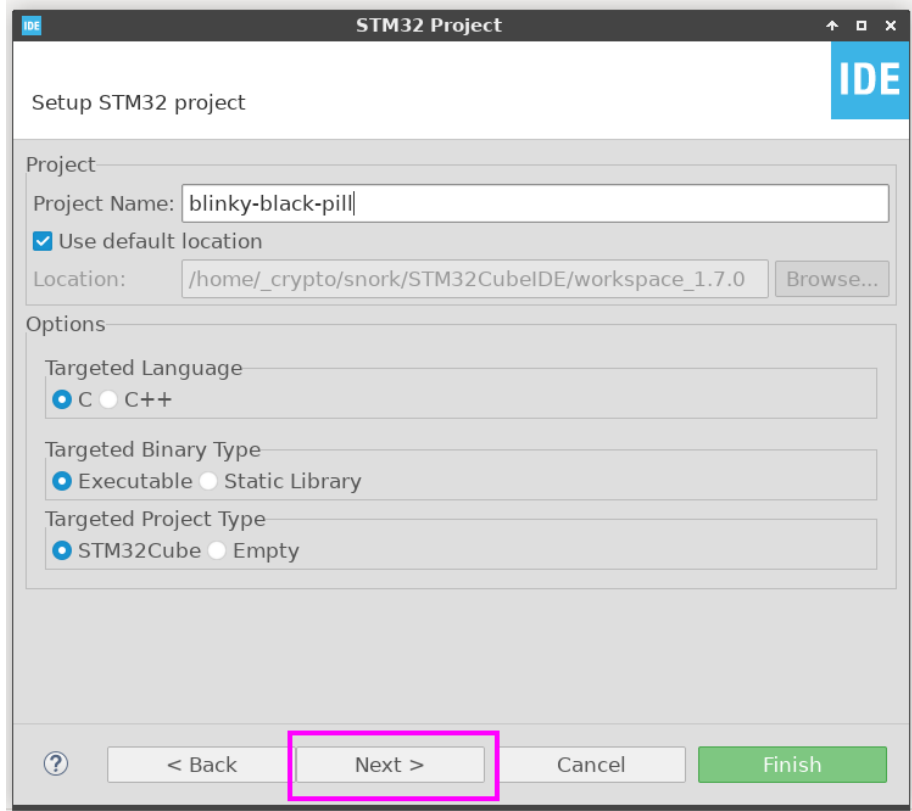
< Back

Next >

Cancel

Finish

Создание проекта



STM32 Project

Setup STM32 project

Project

Project Name:

☒ Use default location

Location:

Options

Targeted Language

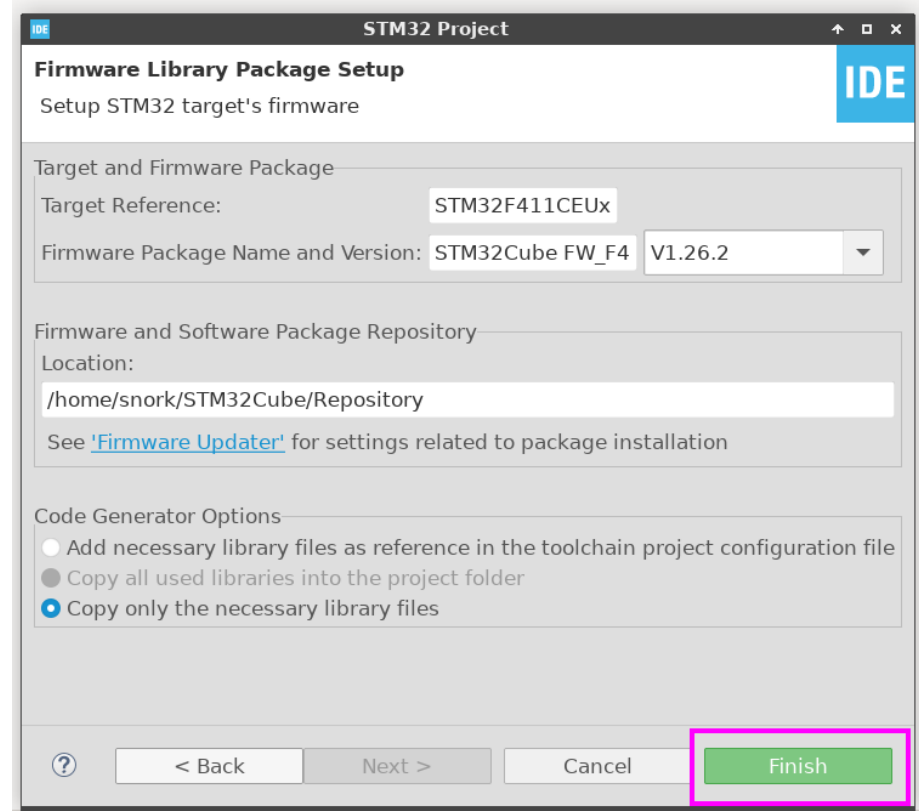
☒ C ☐ C++

Targeted Binary Type

☒ Executable ☐ Static Library

Targeted Project Type

☒ STM32Cube ☐ Empty



STM32 Project

Firmware Library Package Setup

Setup STM32 target's firmware

Target and Firmware Package

Target Reference:

Firmware Package Name and Version:

Firmware and Software Package Repository

Location:

See ['Firmware Updater'](#) for settings related to package installation

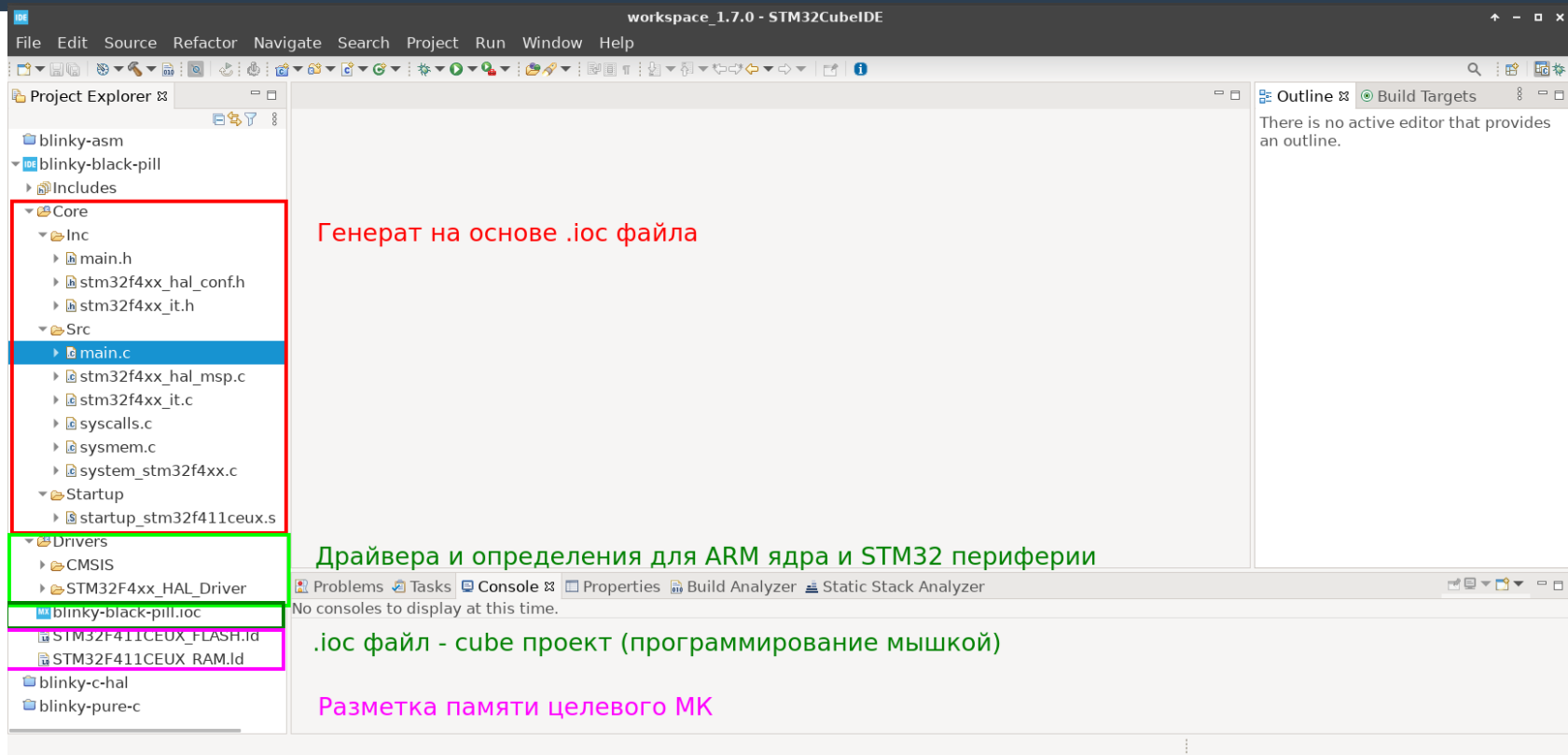
Code Generator Options

☐ Add necessary library files as reference in the toolchain project configuration file

☐ Copy all used libraries into the project folder

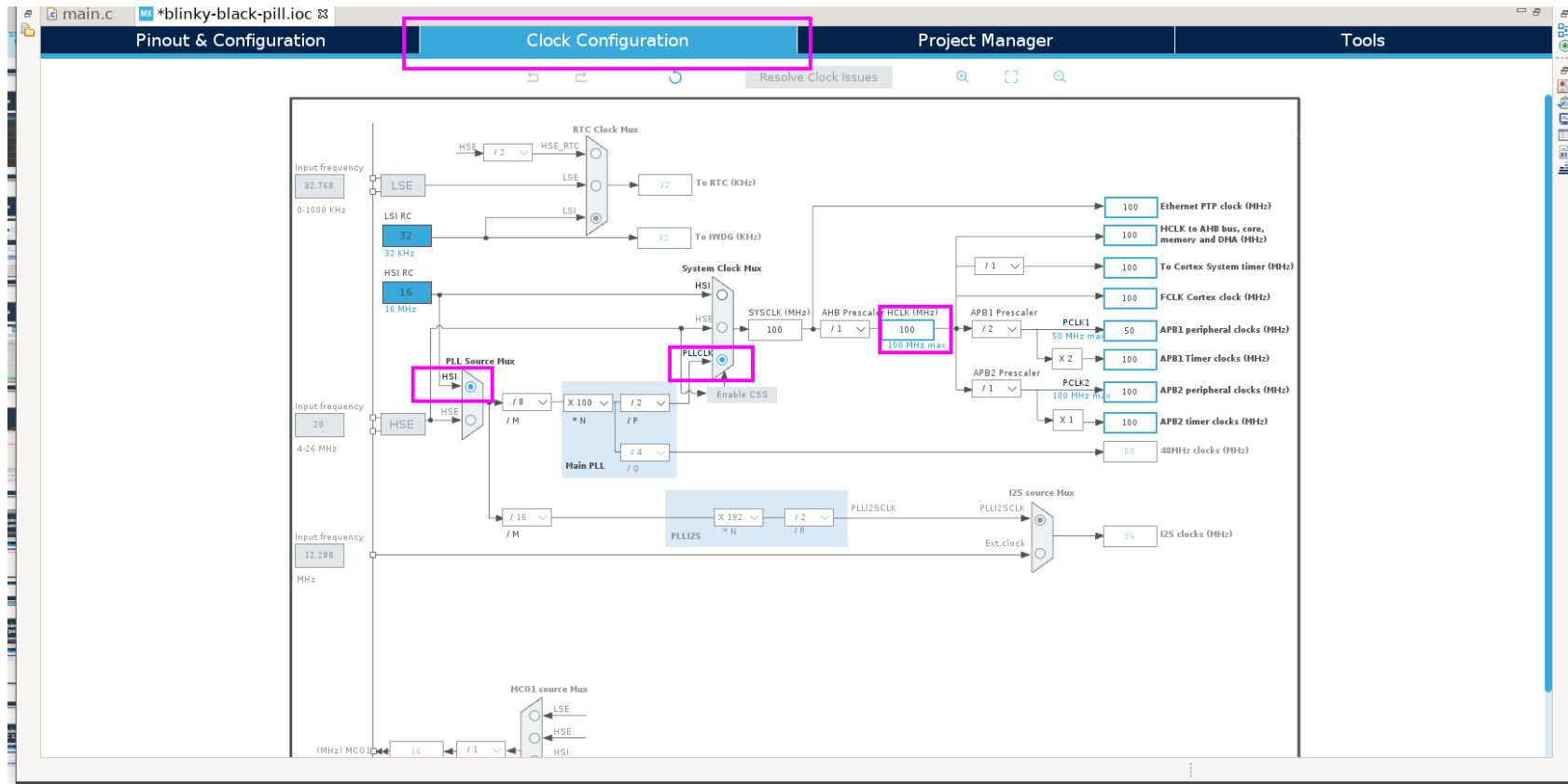
☒ Copy only the necessary library files

Обзор проекта

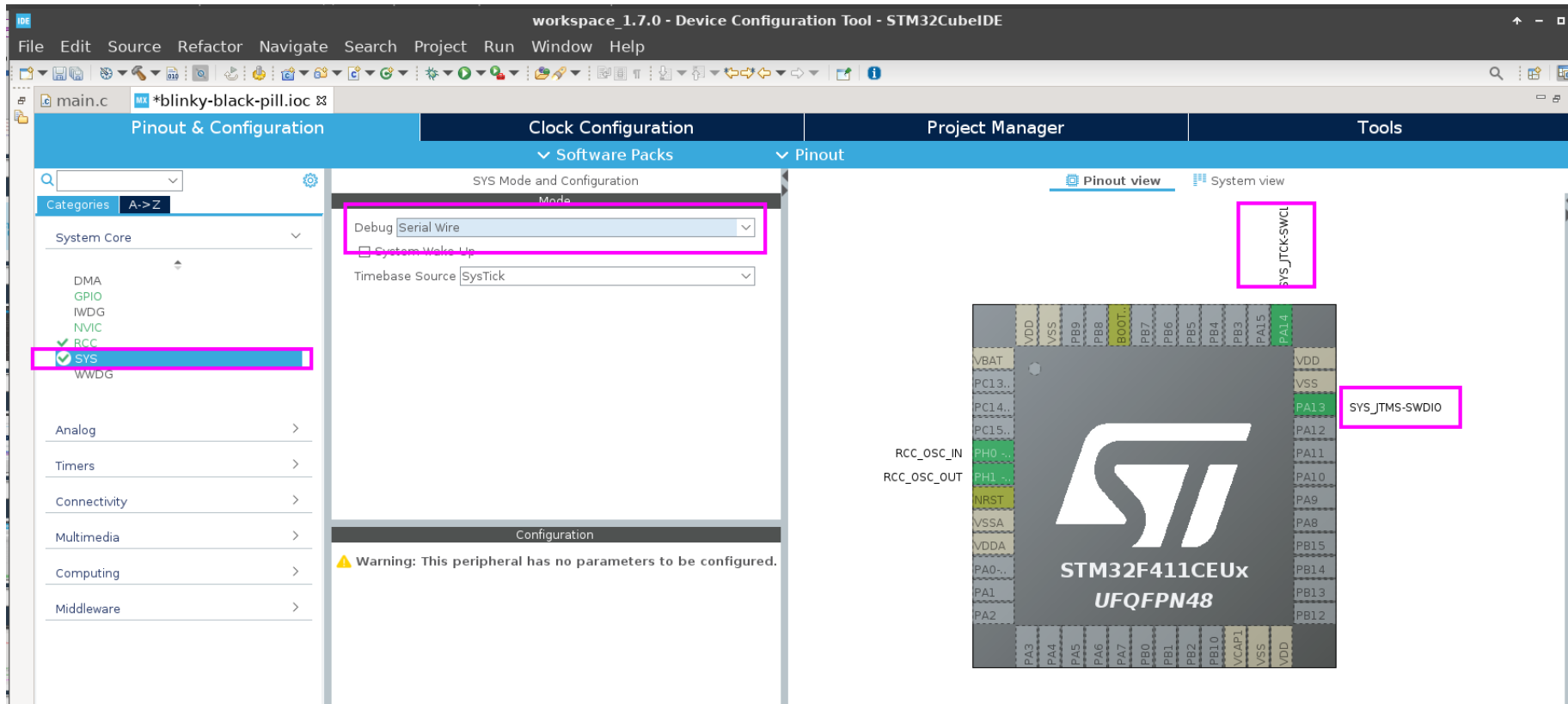


Первичная настройка Cube (.ioc) проекта

Настройка частот



Первичная настройка Cube (.ioc) проекта Разрешение программатора



Первичная настройка Cube (.ioc) проекта

Настройка GPIO C 13

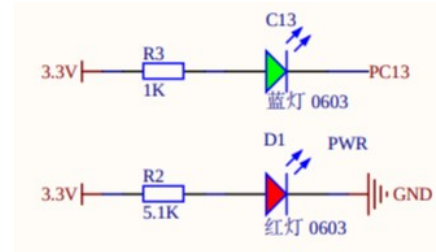
The screenshot shows the STM32CubeIDE Pinout & Configuration window. The left sidebar lists system components, with **GPIO** highlighted. The main panel shows the configuration for PC13, which is set to **Output** mode. The configuration table is as follows:

Pin	Signal	GPIO	Mode	Speed	Output Type	LED
PC13	n/a	High	Output	No pull-up and no pull-down	Low	<input checked="" type="checkbox"/>

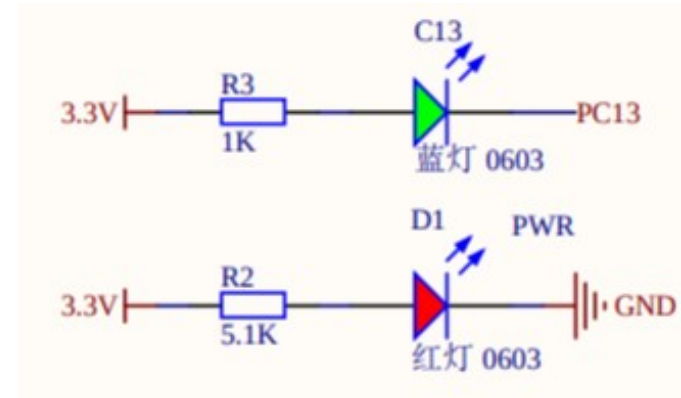
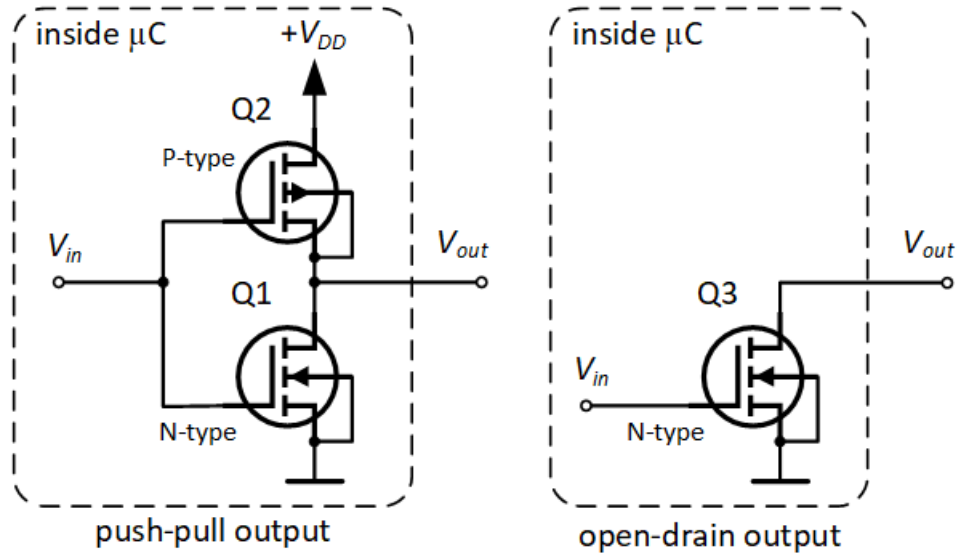
Below the table, the **PC13-ANTI_TAMP Configuration** is shown with the following settings:

- GPIO output level: High
- GPIO mode: Output Open Drain
- GPIO Pull-up/Pull-down: No pull-up and no pull-down
- Maximum output speed: Low
- User Label: LED

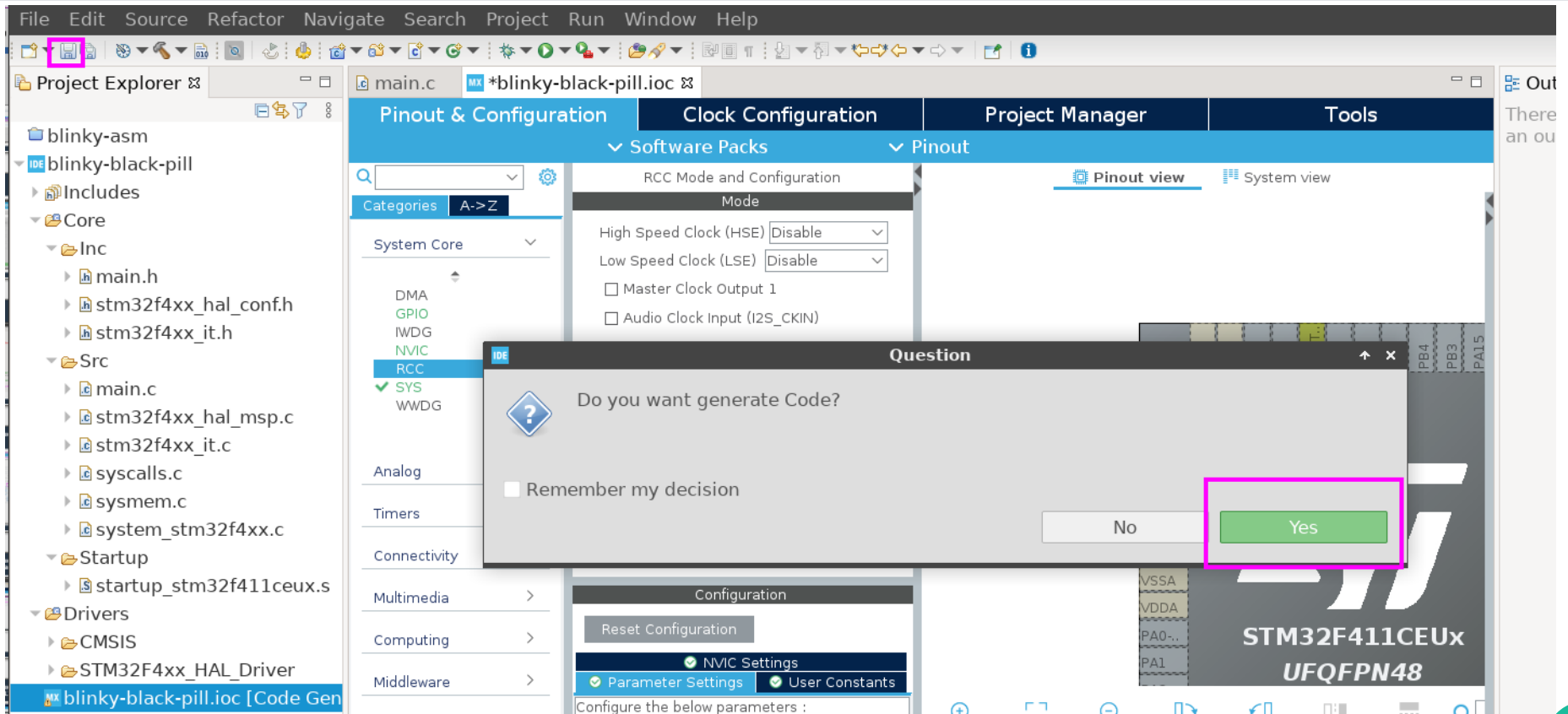
The right sidebar shows the **Pinout view** of the STM32F411CEUx microcontroller, with the **LED** pin highlighted.



Push Pull / Open Drain



Обновление генерата



Функция main()

Состоит из
инициализации
железа и
бесконечного
цикла

```
--
64 int main(void)
65 {
66     /* USER CODE BEGIN 1 */
67     /* USER CODE END 1 */
68     /* MCU Configuration-----*/
69     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
70     HAL_Init();
71     /* USER CODE BEGIN Init */
72     /* USER CODE END Init */
73     /* Configure the system clock */
74     SystemClock_Config();
75     /* USER CODE BEGIN SysInit */
76     /* USER CODE END SysInit */
77     /* Initialize all configured peripherals */
78     MX_GPIO_Init();
79     /* USER CODE BEGIN 2 */
80     /* USER CODE END 2 */
81     /* Infinite loop */
82     /* USER CODE BEGIN WHILE */
83     while (1)
84     {
85         /* USER CODE END WHILE */
86         /* USER CODE BEGIN 3 */
87         /* USER CODE END 3 */
88     }
89     /* USER CODE END 3 */
90 }
91
```

Функции для работы с GPIO

Вообще, документация для HAL живёт [тут](#), но это большой и сложный документ. Почти все что в нем написано — написано и в исходниках

```
/* Initialization and de-initialization functions *****/
void HAL_GPIO_Init(GPIO_TypeDef *GPIOx, GPIO_InitTypeDef *GPIO_Init);
void HAL_GPIO_DeInit(GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin);

/* IO operation functions *****/
GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
void HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState);
void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
```

Что нагенерил cube для инициализации?

Файл main.h

```
/* Private defines -----*/  
#define LED_Pin GPIO_PIN_13  
#define LED_GPIO_Port GPIOC
```

Файл main.c

```
static void MX_GPIO_Init(void)  
{  
    GPIO_InitTypeDef GPIO_InitStruct = {0};  
  
    /* GPIO Ports Clock Enable */  
    __HAL_RCC_GPIOC_CLK_ENABLE();  
    __HAL_RCC_GPIOA_CLK_ENABLE();  
  
    /*Configure GPIO pin Output Level */  
    HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);  
  
    /*Configure GPIO pin : LED_Pin */  
    GPIO_InitStruct.Pin = LED_Pin;  
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_OD;  
    GPIO_InitStruct.Pull = GPIO_NOPULL;  
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;  
    HAL_GPIO_Init(LED_GPIO_Port, &GPIO_InitStruct);  
}
```

GPIO output level

High

GPIO mode

Output Open Drain

GPIO Pull-up/Pull-down

No pull-up and no pull-down

Maximum output speed

Low

User Label

LED

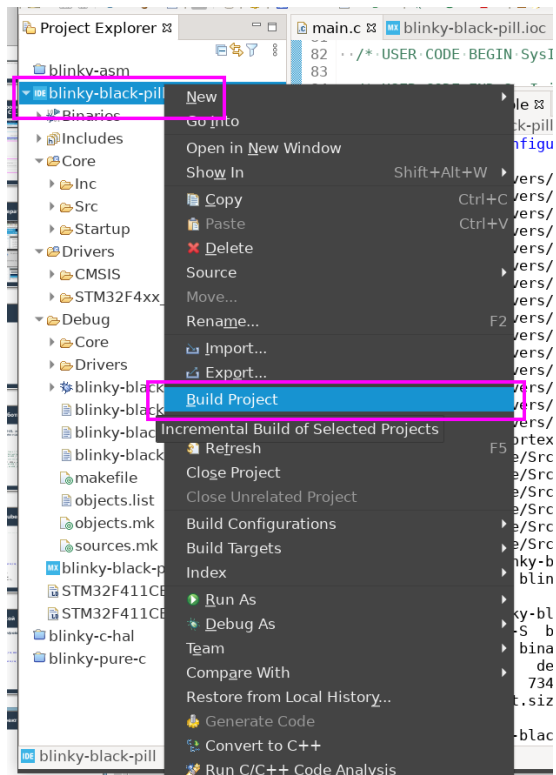
Мигаем лампочкой

Файл main.c функция main(void)

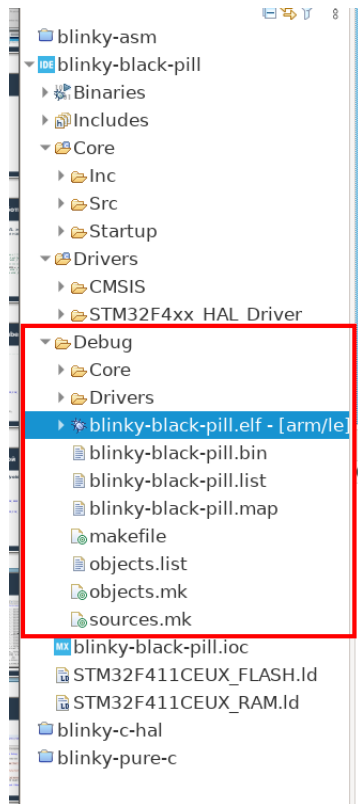
```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    // Зажигаем лампочку
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
    // Ждем 1000 миллисекунд
    HAL_Delay(1000);
    // Тушим лампочку
    HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
    // Ждем 500 миллисекунд
    HAL_Delay(500);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

Сборка проекта

[illegible]

Результаты сборки



.elf файл — Executable and Linkable Format
Формат .exe файла для POSIX платформ.
Содержит все необходимое для запуска
и отладки проекта

.bin — образ флеш памяти целевого МК

.list — asm листинг получившегося кода

.map — файл-карта. Показывает в какие
регионы памяти попали разные функции
и переменные

Сборка проекта неудачный билд

```
92  /* Infinite loop */
93  /* USER CODE BEGIN WHILE */
94  while (1)
95  {
96          // Зажигаем лампочку
97          la la la, error here!
98      HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
99          // Ждем 1000 миллисекунд
100     HAL_Delay(1000);
101         // Тушим лампочку
```

Problems Tasks Console Properties Build Analyzer Static Stack Analyzer

CDT Build Console [blinky-black-pill]

15:51:55 **** Incremental Build of configuration Debug for project blinky-black-pill ****

make -j4 all

arm-none-eabi-gcc "../Core/Src/main.c" -mcpu=cortex-m4 -std=gnu11 -g3 -DDEBUG -DUSE_HAL_DRIVER -DSTM32F411xE -

../Core/Src/main.c: In function 'main':

../Core/Src/main.c:97:5: error: unknown type name 'la'

```
97 |     la la la, error here!
```

```
    ^~
```

../Core/Src/main.c:97:11: error: expected '=', ',', ';', 'asm' or '__attribute__' before 'la'

```
97 |     la la la, error here!
```

```
    ^~
```

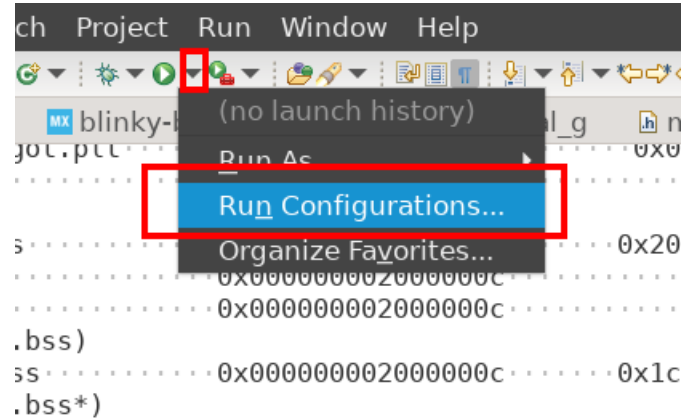
make: *** [Core/Src/subdir.mk:34: Core/Src/main.o] Error 1

"make -j4 all" terminated with exit code 2. Build might be incomplete.

15:51:56 Build Failed. 3 errors, 0 warnings. (took 746ms)

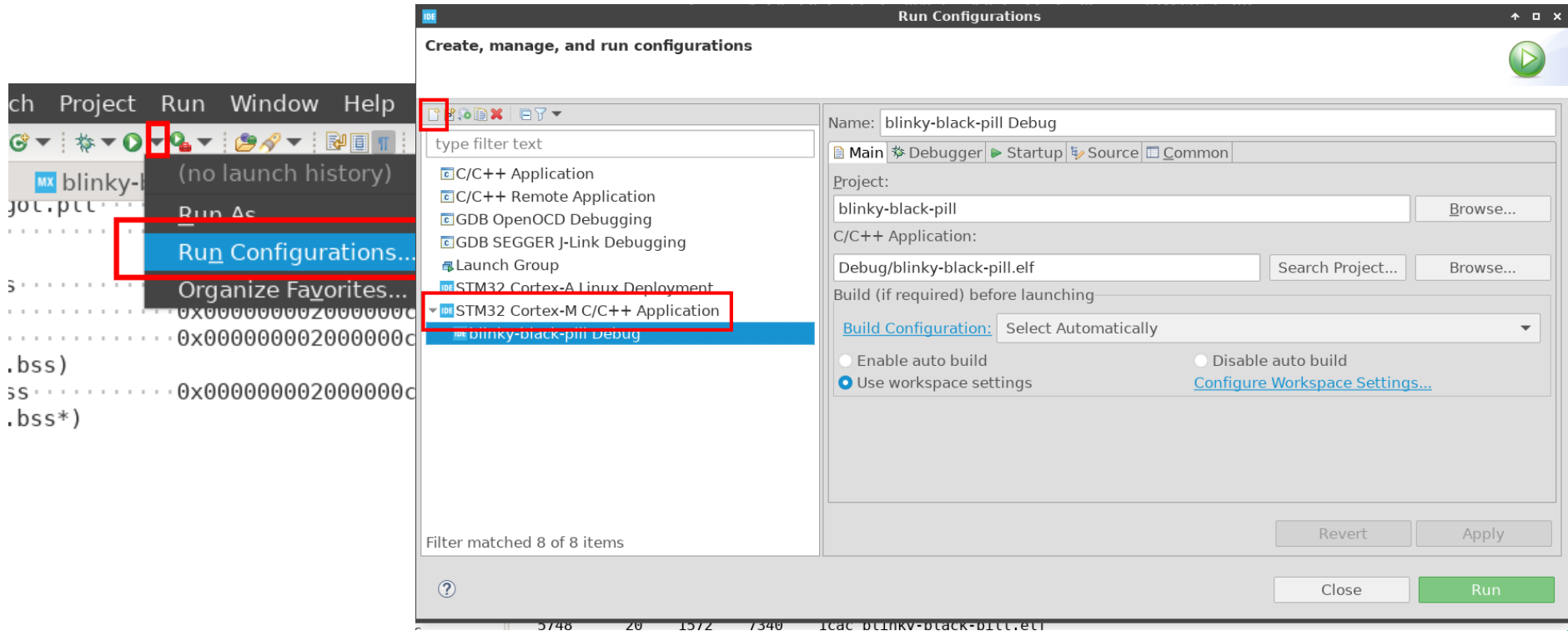
Прошивка проекта

Создание конфигурации запуска



Прошивка проекта

Создание конфигурации запуска



Прошивка проекта

Настройка конфигурации запуска

Name: blinky-black-pill Debug

Main **Debugger** Startup Source Common

GDB Connection Settings

☒ Autostart local GDB server Host name or IP address localhost

☐ Connect to remote GDB server Port number 61234

Debug probe ST-LINK (ST-LINK GDB server)

GDB Server Command Line Options

Show Command Line

Interface

☒ SWD ☐ JTAG

☐ ST-LINK S/N Scan

Frequency (kHz): Auto

Access port: 0 - Cortex-M4

Reset behaviour

Type: Software system reset

Device settings

Debug in low power modes: Enable

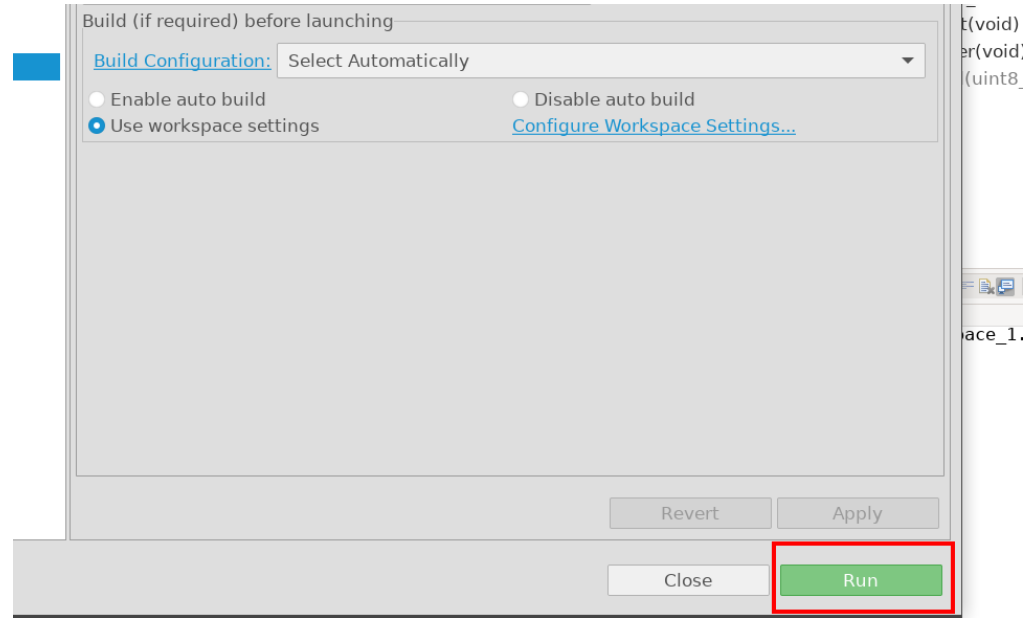
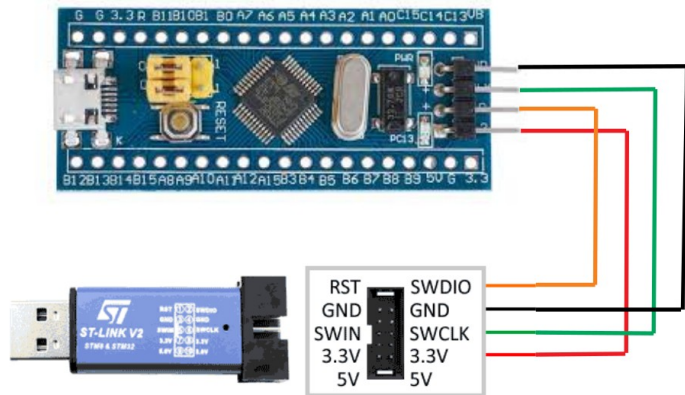
Suspend watchdog counters while halted: No configuration

Revert Apply

Close Run

Прошивка проекта

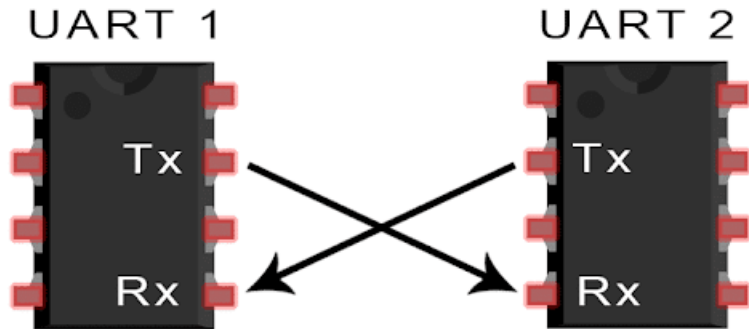
Подключение программатора и запуск



Делаем себе консоль

UART

Universal asynchronous receiver-transmitter



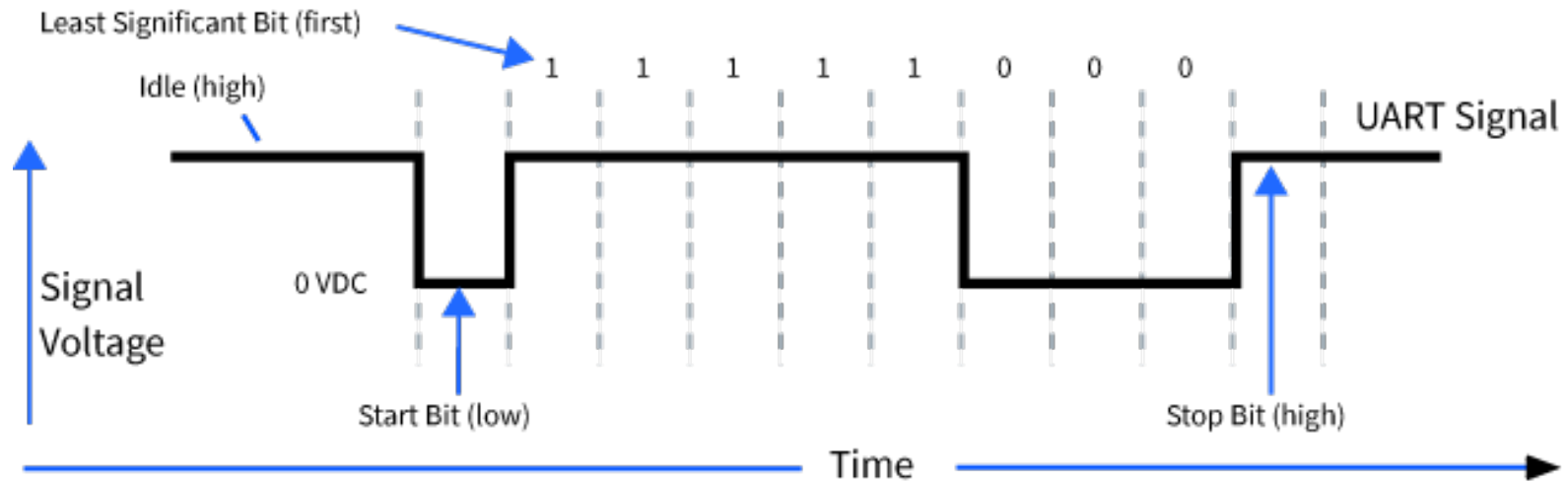
Асинхронный интерфейс

Позволяет связать **два равноправных** устройства

Достаточно быстрый

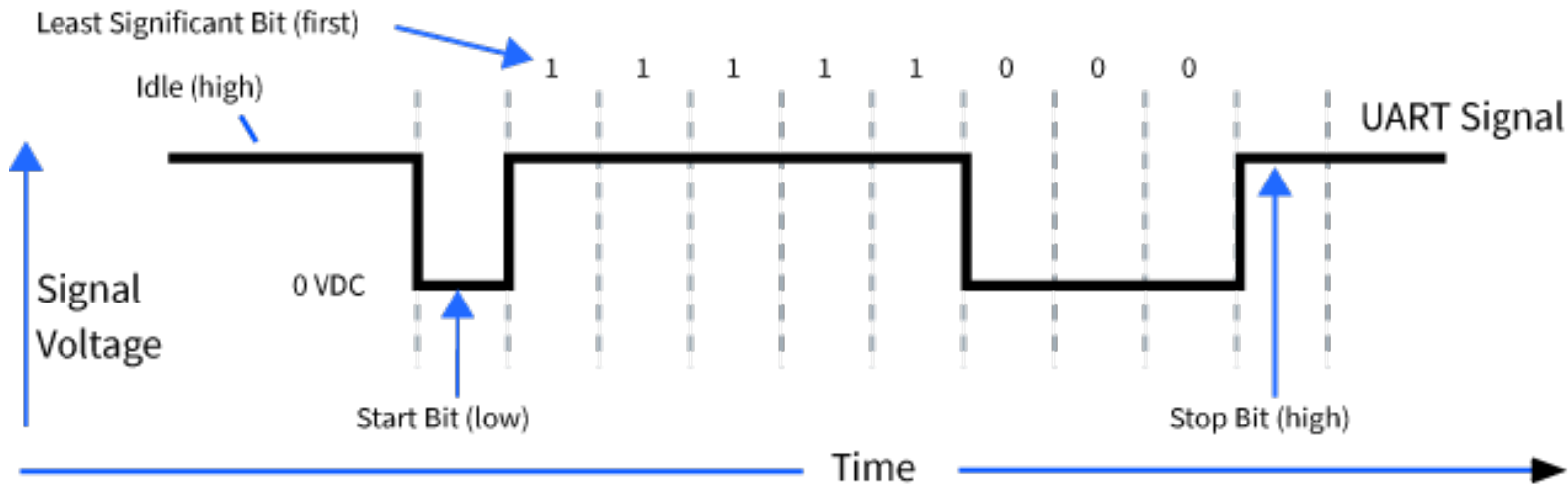
Очень распространённый

UART



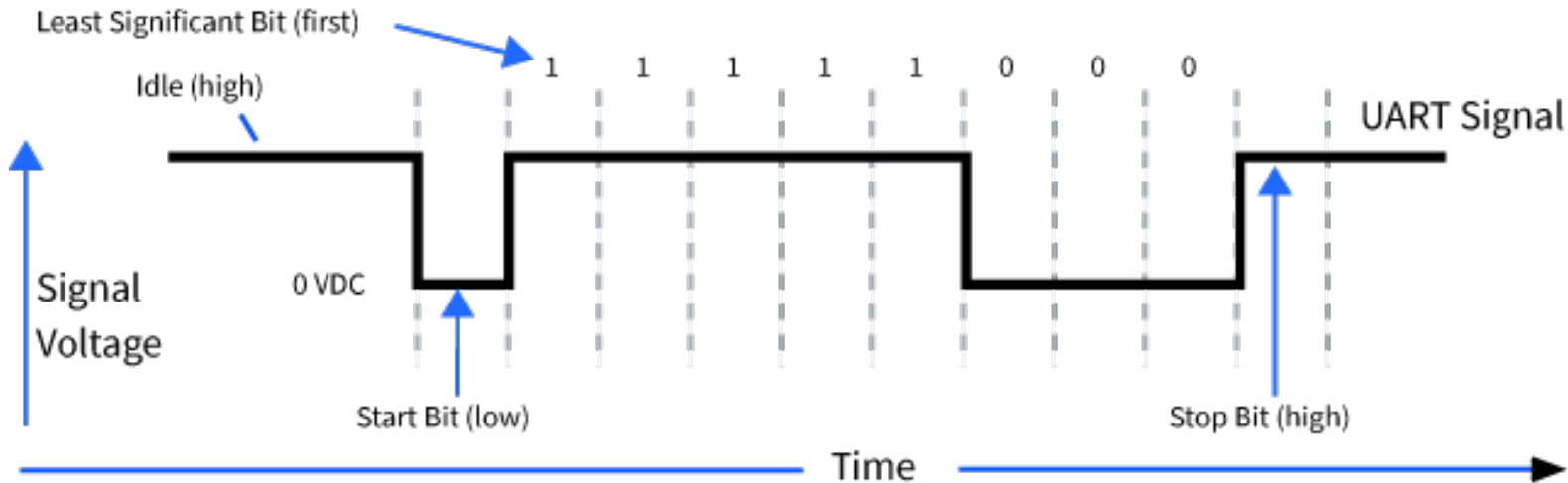
UART Baud Rate

Baud Rate это частота изменения сигнала. Показывает сколько «символов» в секунду передается посредством интерфейса.
Типовые значения: 110, 300, 600, 1200, 2400, 4800, **9600**, 14400, 19200, 38400, 57600, **115200**, 128000 и 256000



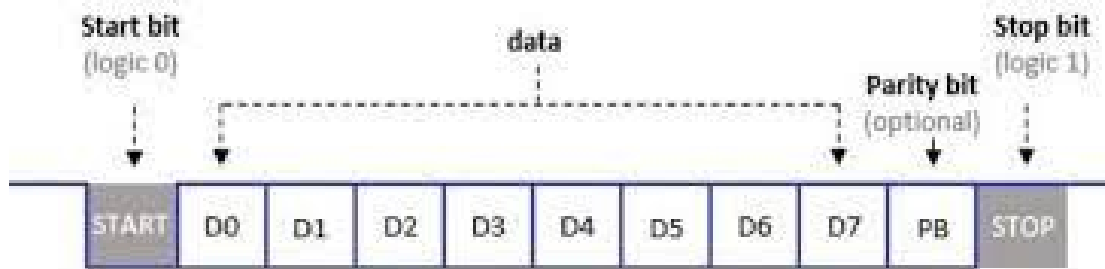
UART Word size

В этом примере передаётся 8 бит за один «фрейм». Некоторые устройства передают большее или меньшее количество бит за фрейм.



UART Parity

Возможно использование специального «бита чётности». Этот бит чётности вставляется между битами данных и стоп битом. Значение этого бита подбирается так, чтобы общее количество единиц в посылке было чётным (или не чётным, как настроить). Приёмник может обнаружить сбой в передаче при нарушении этого правила.

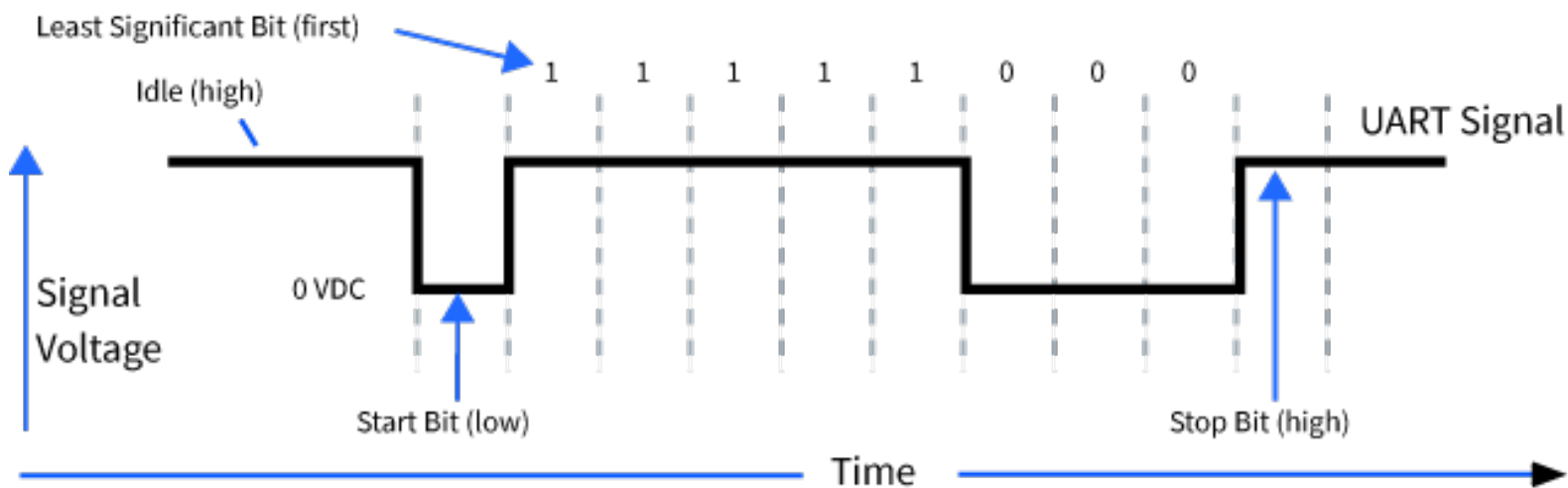


DIFFERENCE BETWEEN UART AND USART
(UART VS USART), YOU SHOULD KNOW

7 bits of data	(count of 1-bits)	8 bits including parity	
		even	odd
0000000	0	00000000	00000001
1010001	3	10100011	10100010
1101001	4	11010010	11010011
1111111	7	11111111	11111110

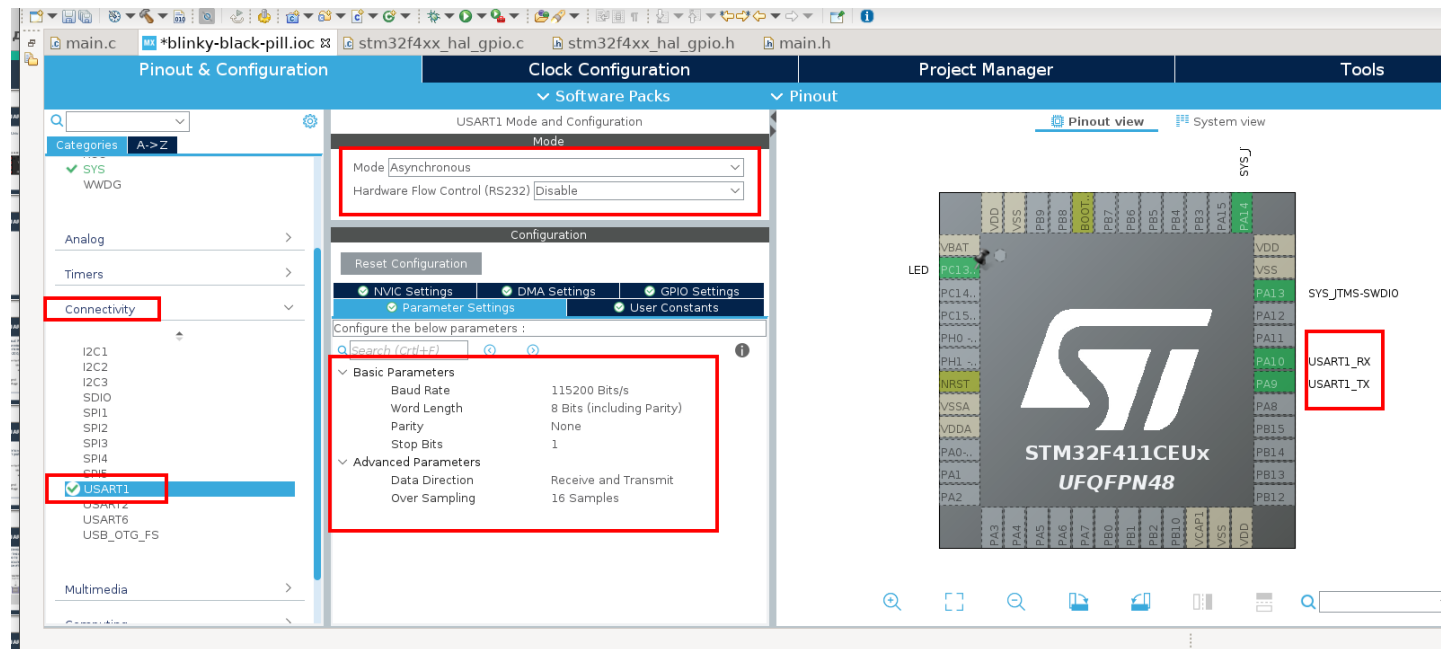
UART Stop Bits

В этом примере используется 1 стоп бит. Некоторые устройства используют несколько стоп битов



В нашем stm32 есть аж 3 U(S)ART модуля периферии

Включаем U(S)ART1



API для работы с U(S)ART

API для работы с U(S)ART

```
/* Initialization/de-initialization functions *****/
HAL_StatusTypeDef HAL_UART_Init(UART_HandleTypeDef *huart);

HAL_StatusTypeDef HAL_HalfDuplex_Init(UART_HandleTypeDef *huart);
HAL_StatusTypeDef HAL_LIN_Init(UART_HandleTypeDef *huart, uint32_t BreakDetectLength);
HAL_StatusTypeDef HAL_MultiProcessor_Init(UART_HandleTypeDef *huart, uint8_t Address, uint32_t WakeUpMethod);
HAL_StatusTypeDef HAL_UART_DeInit(UART_HandleTypeDef *huart);

/* IO operation functions *****/
HAL_StatusTypeDef HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout);
HAL_StatusTypeDef HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout);

HAL_StatusTypeDef HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_UART_Transmit_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_UART_Receive_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
```

Отправляем hello world

```
/* USER CODE BEGIN 2 */
const char string[] = "Hello world!\n";
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    // Зажигаем лампочку
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
    // Ждем 1000 миллисекунд
    HAL_Delay(1000);
    // Тушим лампочку
    HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
    // Ждем 500 миллисекунд
    HAL_Delay(500);

    // Отправляем хеллоу ворлд
    HAL_UART_Transmit(&uart1, string, sizeof(string)-1, HAL_MAX_DELAY);

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

Принимаем сообщение на ПК



RS-232 традиционно использовался на ПК для подключения различной периферии.

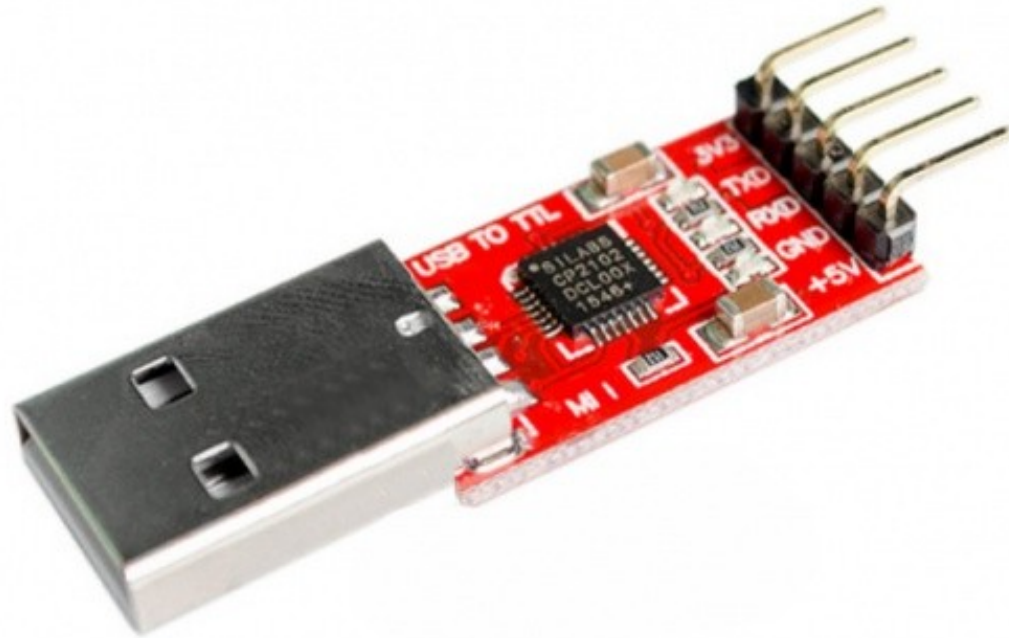
Сейчас он присутствует на многих материнских платах, но не выведен на корпус. На корпус он выведен в промышленных ПК (например в кассовых аппаратах)



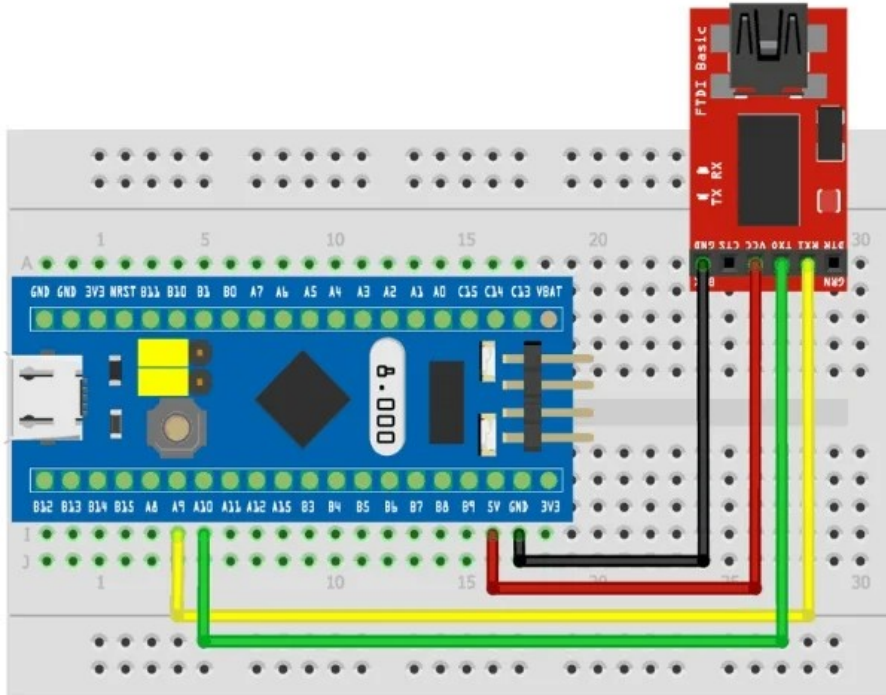
BADODU
SECURITY



USB-UART преобразователь



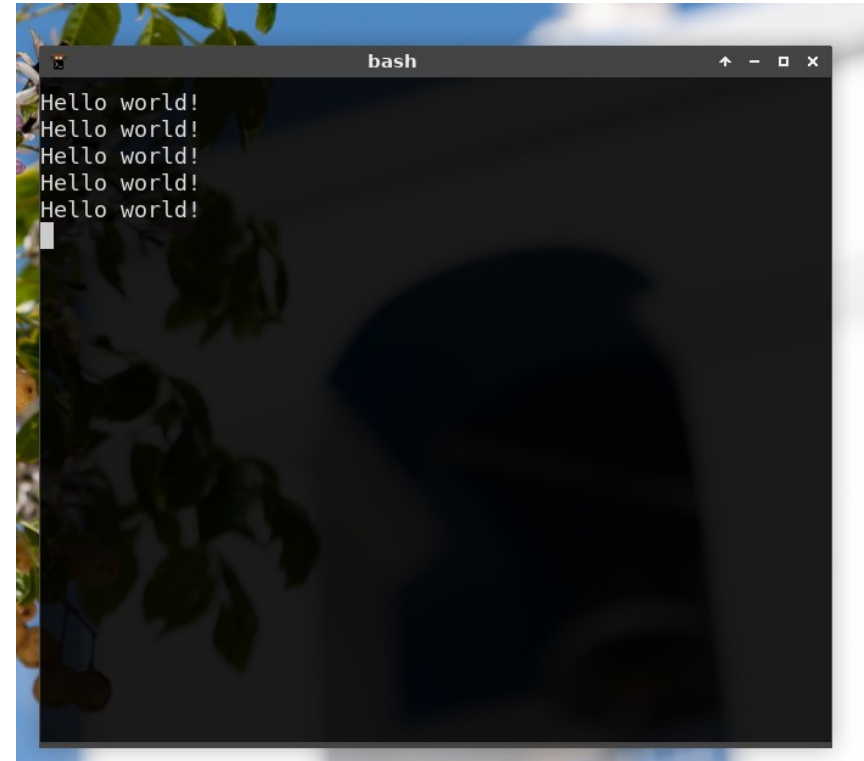
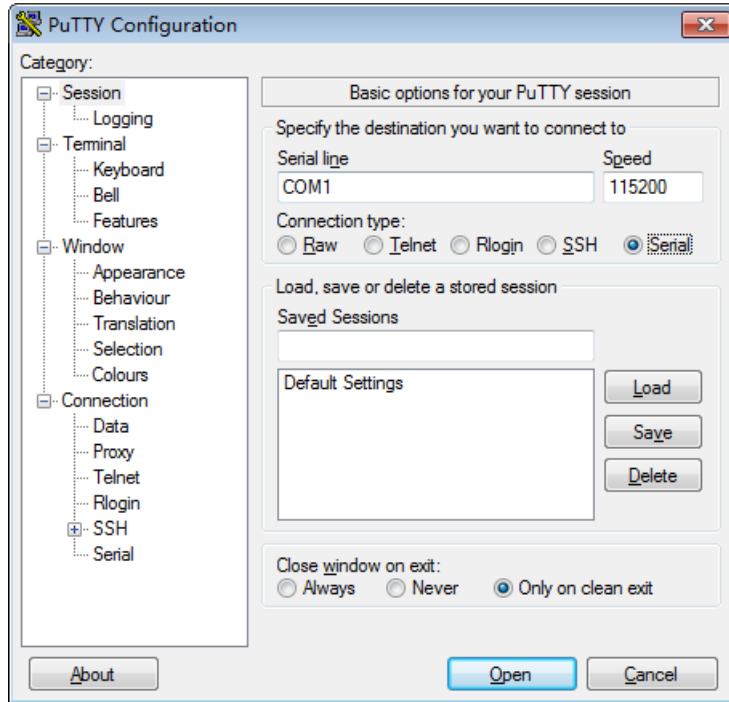
Подключение к STM32



FTDI >> STM32
Gnd >> Gnd
Vcc >> 5V
Rx >> A9
Tx >> A10

fritzing

Putty / minicom



Использование UART вместе с printf

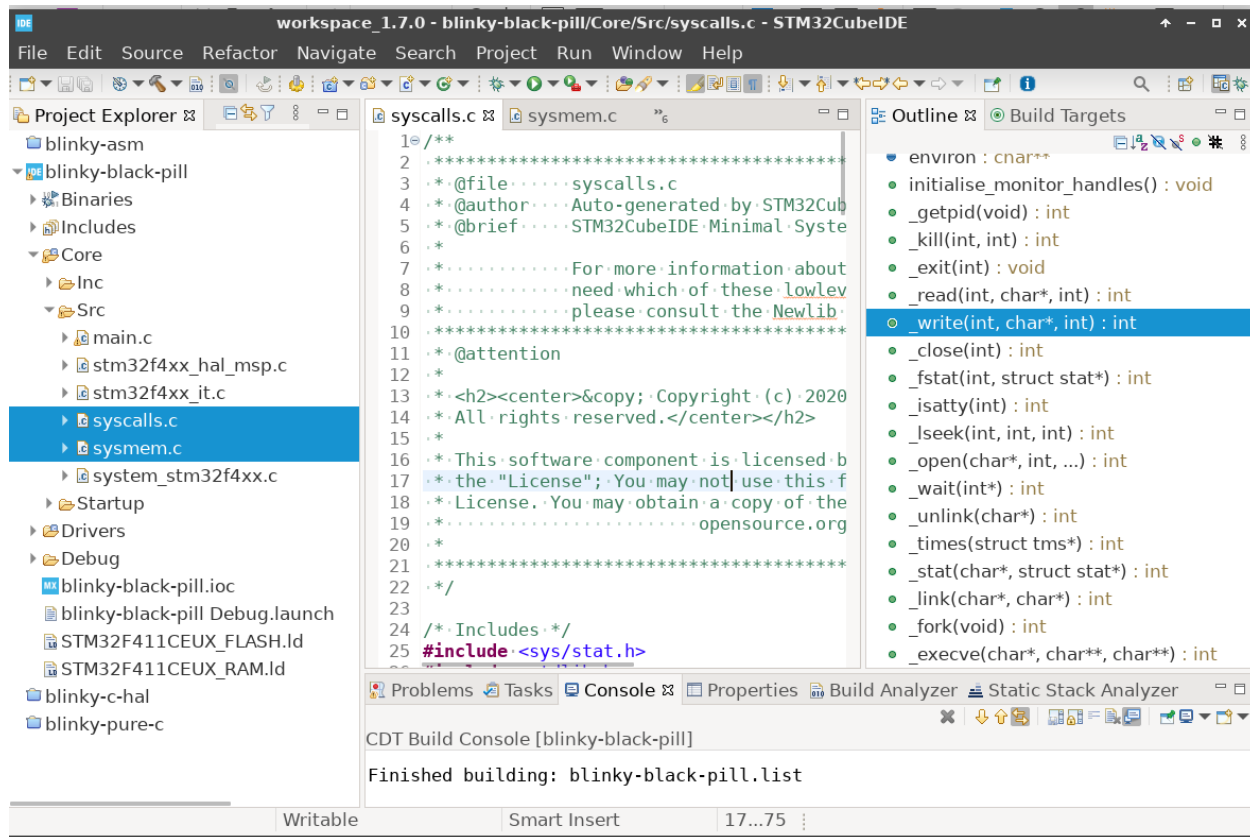
Пользовательское приложение

Стандартная библиотека Си (newlib)

Системное API

Железное «API»

Использование UART вместе с printf



```
int _write(int file, char *ptr, int len)
{
    /* ... */
}
```

```
#include <stm32f4xx_hal.h>
```

```
int _write(int file, char *ptr, int len)
{
    extern UART_HandleTypeDef huart1;

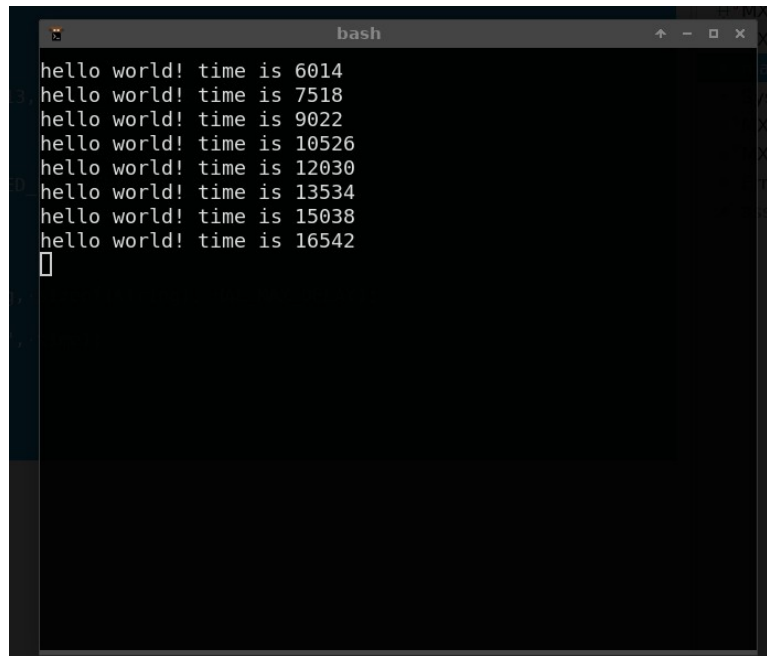
    HAL_UART_Transmit(
        &huart1,
        (uint8_t*)ptr, len,
        HAL_MAX_DELAY
    );

    return len;
}
```

Каноничный hello world

```
/* Private includes -----*/  
/* USER CODE BEGIN Includes */  
#include <stdio.h>  
/* USER CODE END Includes */
```

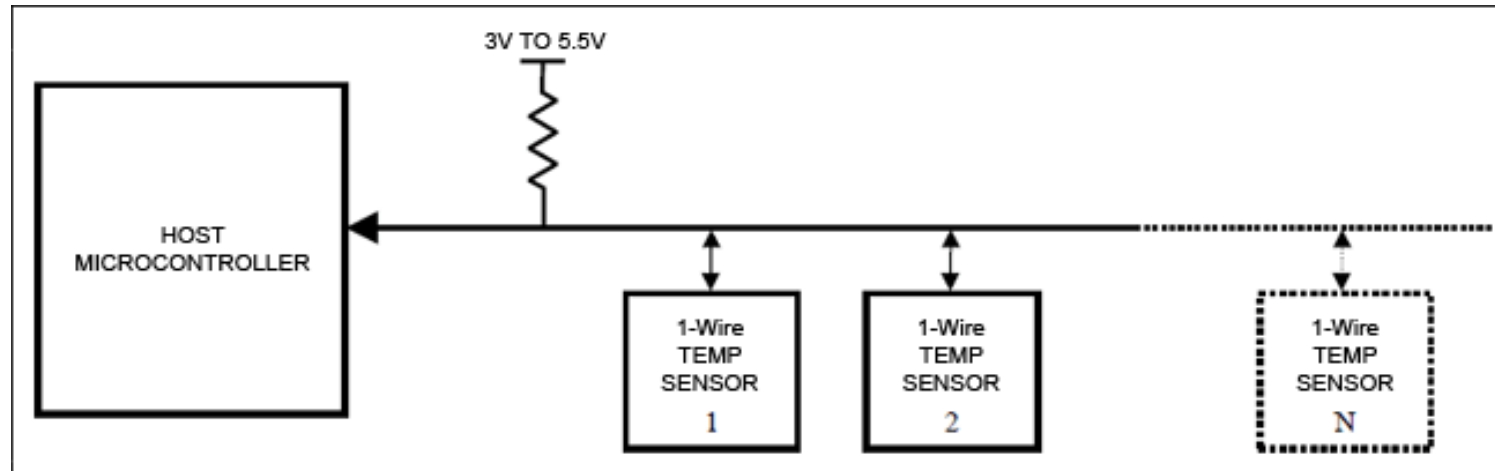
```
/* USER CODE BEGIN WHILE */  
while (1)  
{  
    // Зажигаем лампочку  
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);  
    // Ждем 1000 миллисекунд  
    HAL_Delay(1000);  
    // Тушим лампочку  
    HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);  
    // Ждем 500 миллисекунд  
    HAL_Delay(500);  
  
    // Отправляем хеллоу ворлд  
    //HAL_UART_Transmit(&huart1, string, sizeof(string), HAL_MAX_DELAY);  
    int32_t time = HAL_GetTick();  
    printf("hello world! time is %ld\n", time);  
  
    /* USER CODE END WHILE */  
  
    /* USER CODE BEGIN 3 */  
}  
/* USER CODE END 3 */
```



```
bash  
hello world! time is 6014  
hello world! time is 7518  
hello world! time is 9022  
hello world! time is 10526  
hello world! time is 12030  
hello world! time is 13534  
hello world! time is 15038  
hello world! time is 16542  
█
```

Шина OneWire

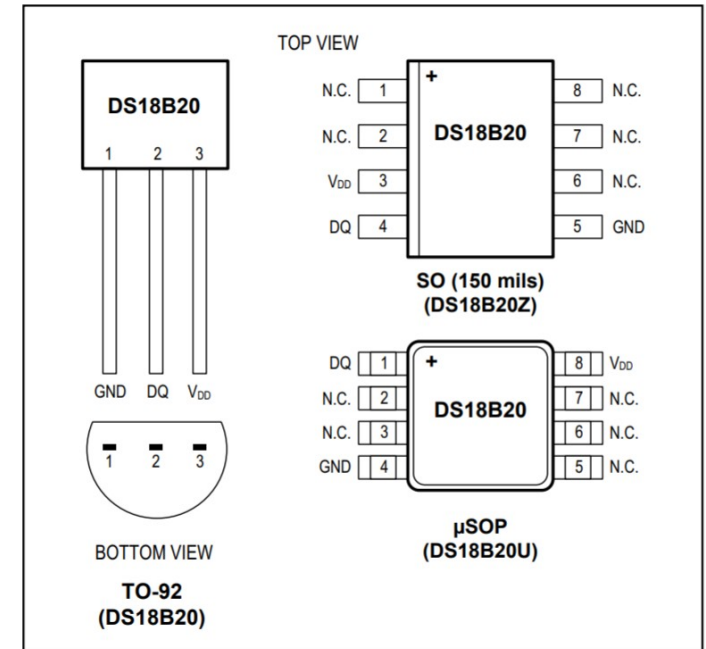
Шина 1-Wire



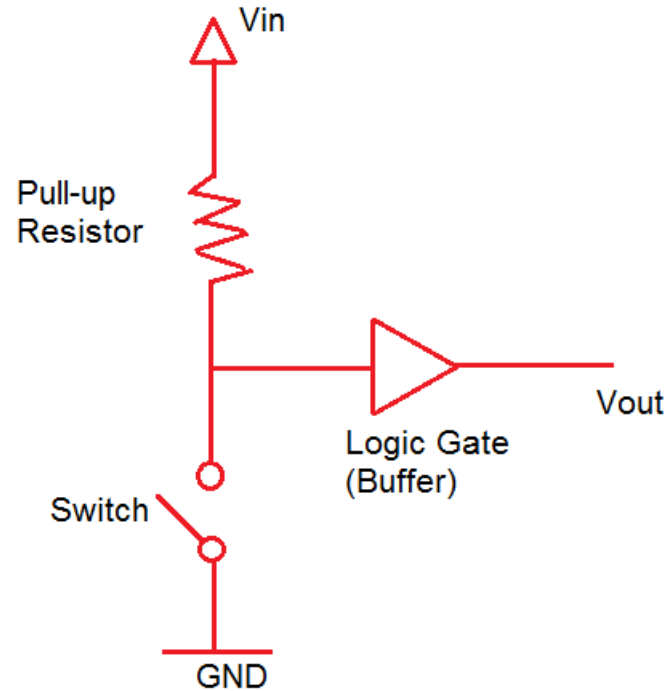
[Чумовой tutorial на русском](#)

DS18B20

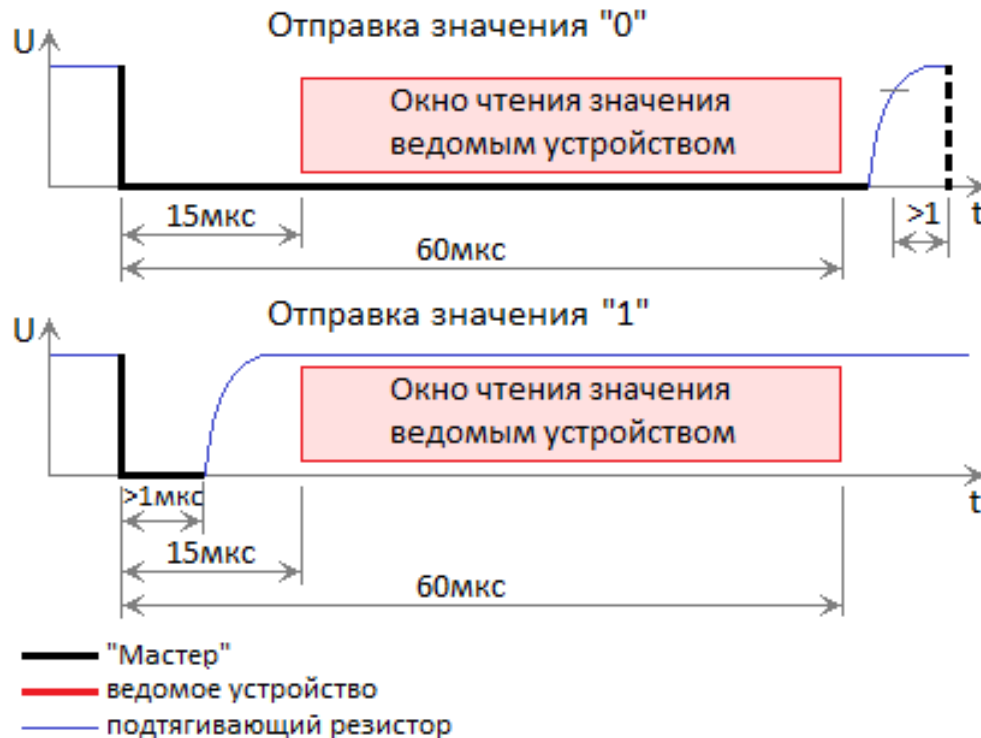
- Measures Temperatures from -55°C to $+125^{\circ}\text{C}$
- $\pm 0.5^{\circ}\text{C}$ Accuracy from -10°C to $+85^{\circ}\text{C}$
- Programmable Resolution from 9 Bits to 12 Bits
- No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for operation (DQ and GND)
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits



Резистор подтяжки. 0 и 1 на шине



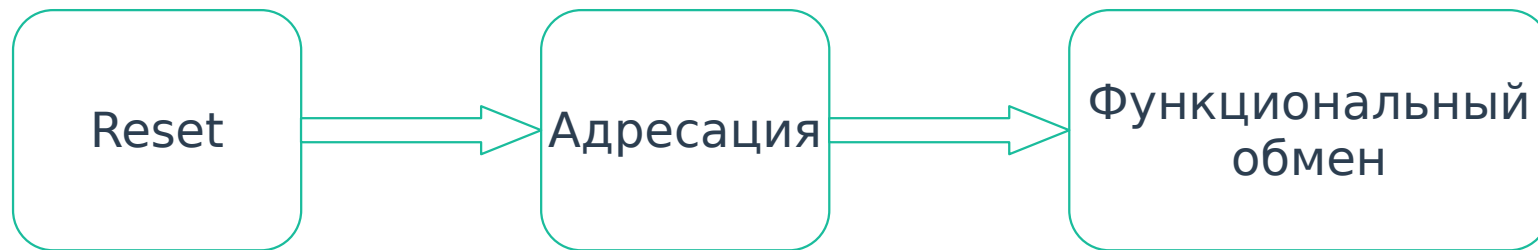
Передача битиков от ведущего



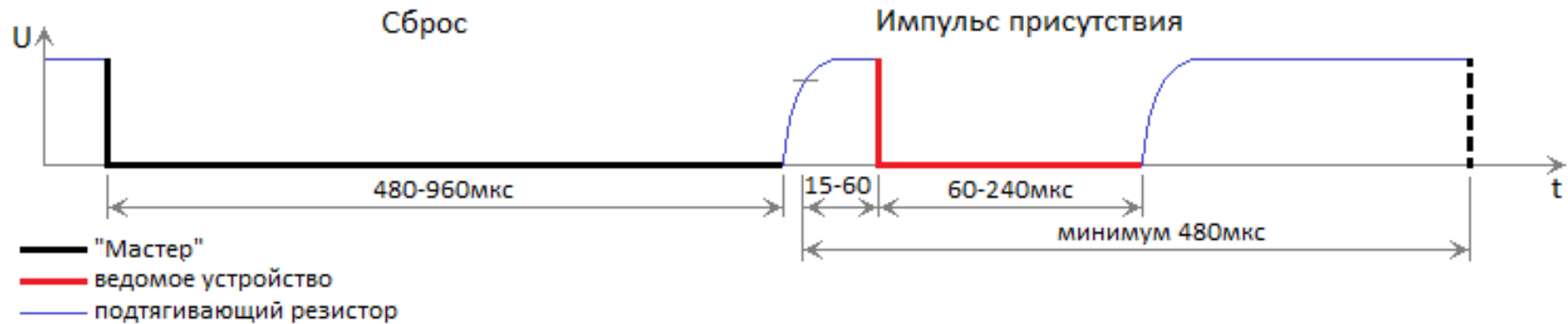
Передача битиков от ведомого



Этапы обмена



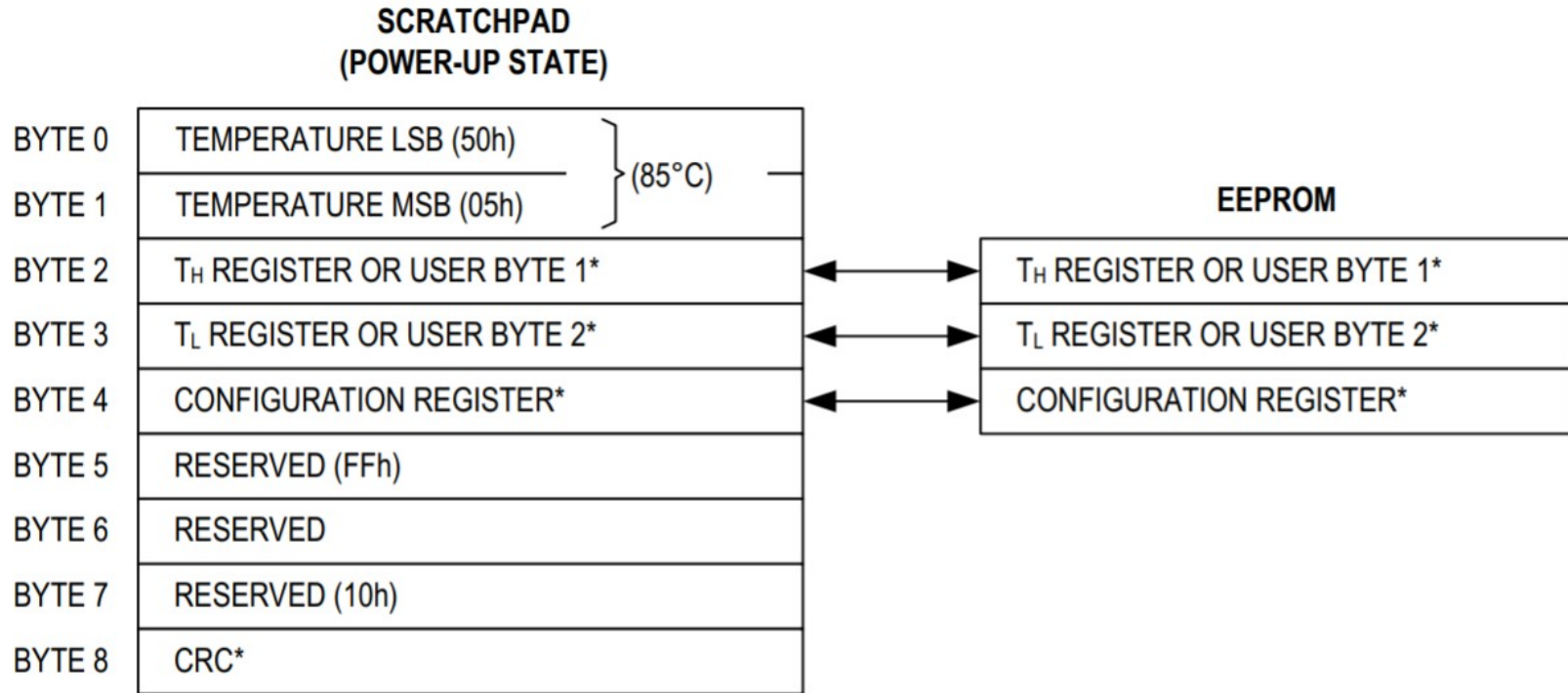
1-wire reset



Этап адресации

- У каждого устройства есть вшитый в него на фабрике 64-битный уникальный адрес
- После reset команды наступает этап адресации
- Команды этого этапа:
 - 0xCC — skip rom - пропуск этапа адресации
 - 0x33 — read rom - чтение адреса ведомого (если он один)
 - 0x55 — match rom - выбор ведомого по указанному адресу
 - 0xF0 — find rom - итеративное обнаружение всех ведомых на шине
 - [0xEC — alarm search — эксклюзивная команда ds18b20 для поиска ведомых в состоянии «тревоги»]

Scratchpad



*POWER-UP STATE DEPENDS ON VALUE(S) STORED IN EEPROM.

Команды DS18B20

0x44 — convert T — начать замер температуры

0xEH — write scratchpad — принять значения для scratchpad с 1-wire шины

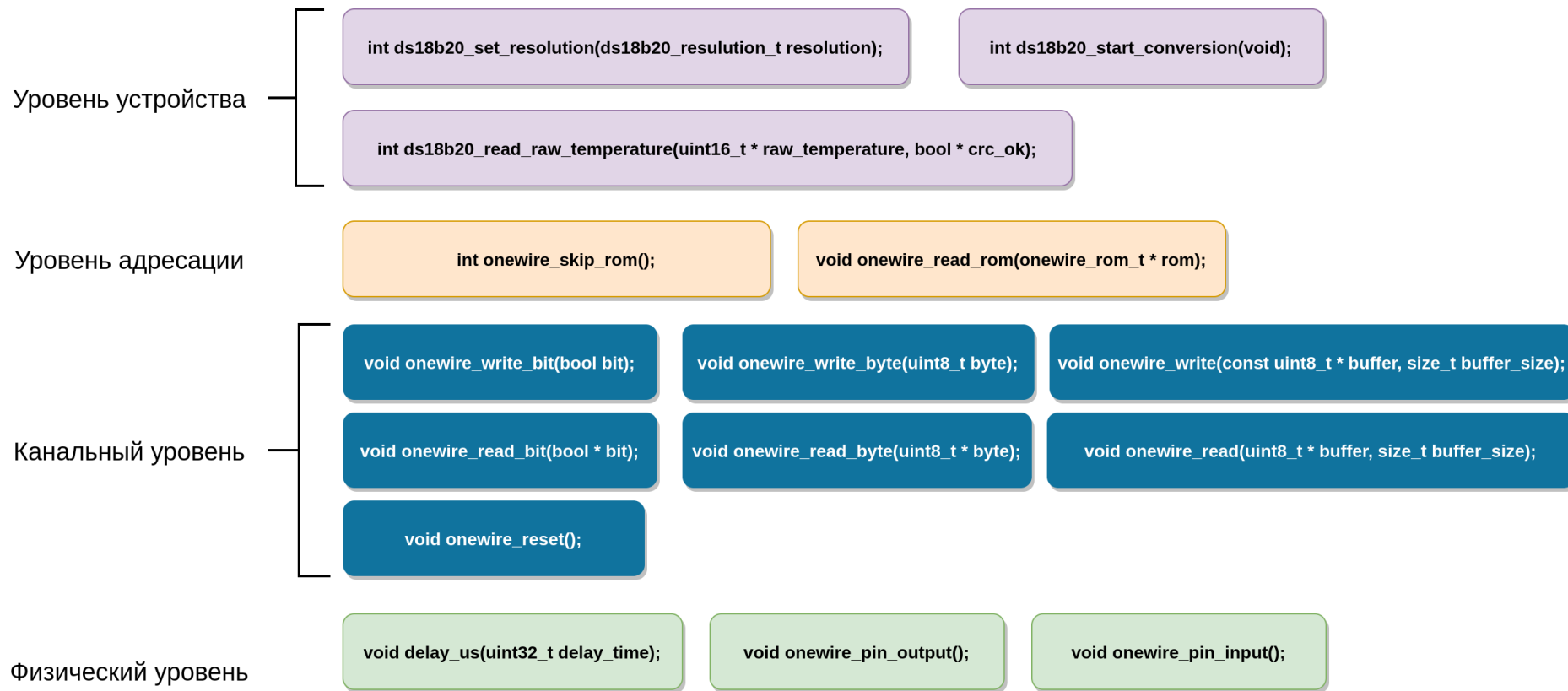
0xBE — read scratchpad — выдать значения из scratchpad на 1-wire шину

0x48 — copy scratchpad — сохранить значения из scratchpad в ПЗУ датчика

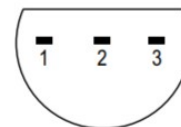
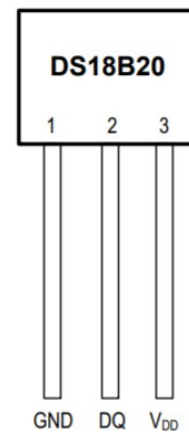
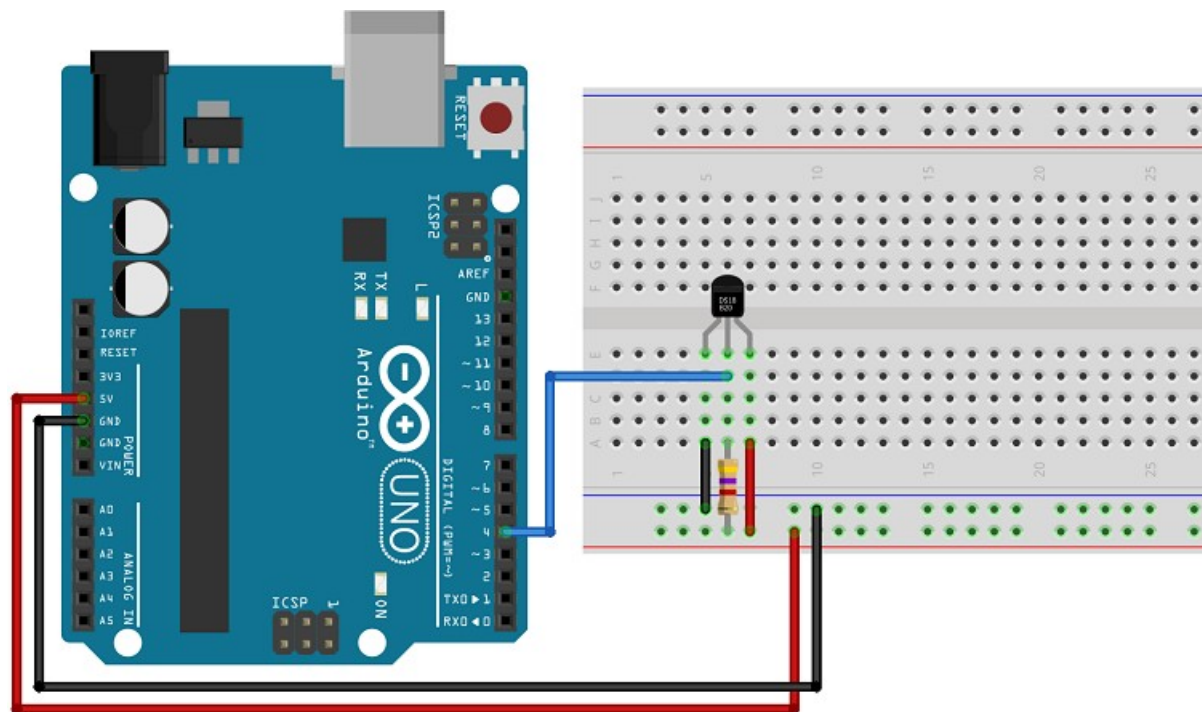
0x8H — recall — загрузить настройки в scratchpad из eeprom

0xB4 — read power supply — проверка режима питания

Декомпозиция драйвера



Подключение датчика



BOTTOM VIEW

TO-92
(DS18B20)