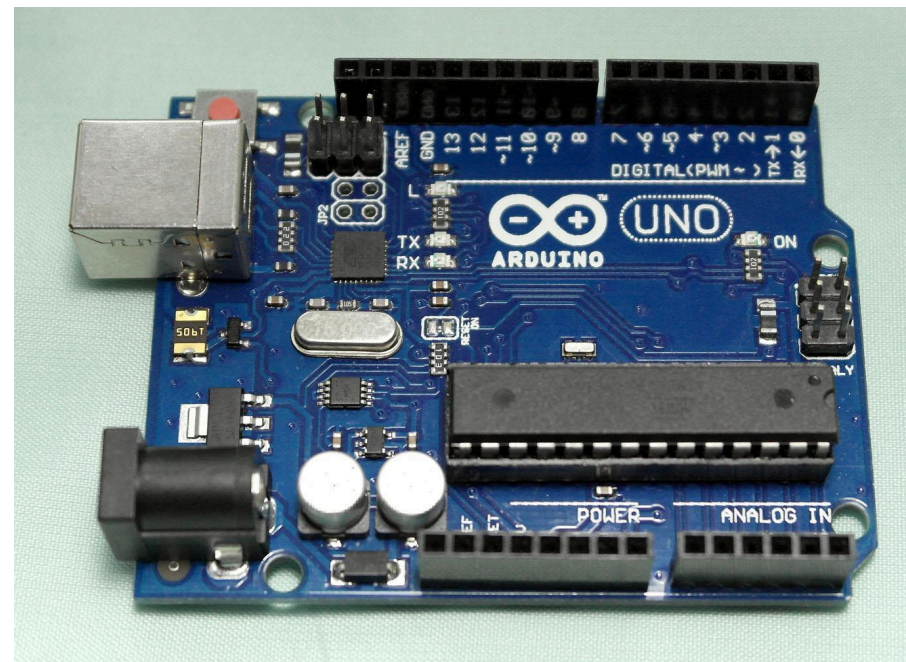
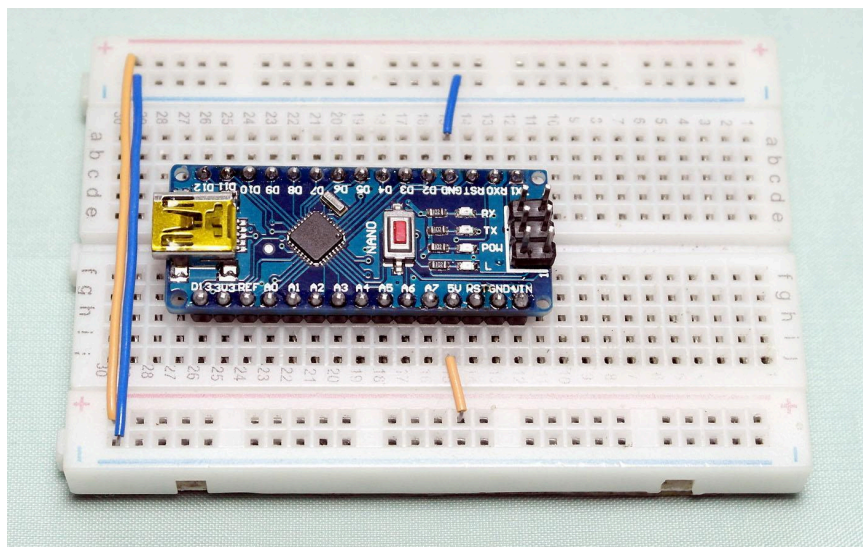
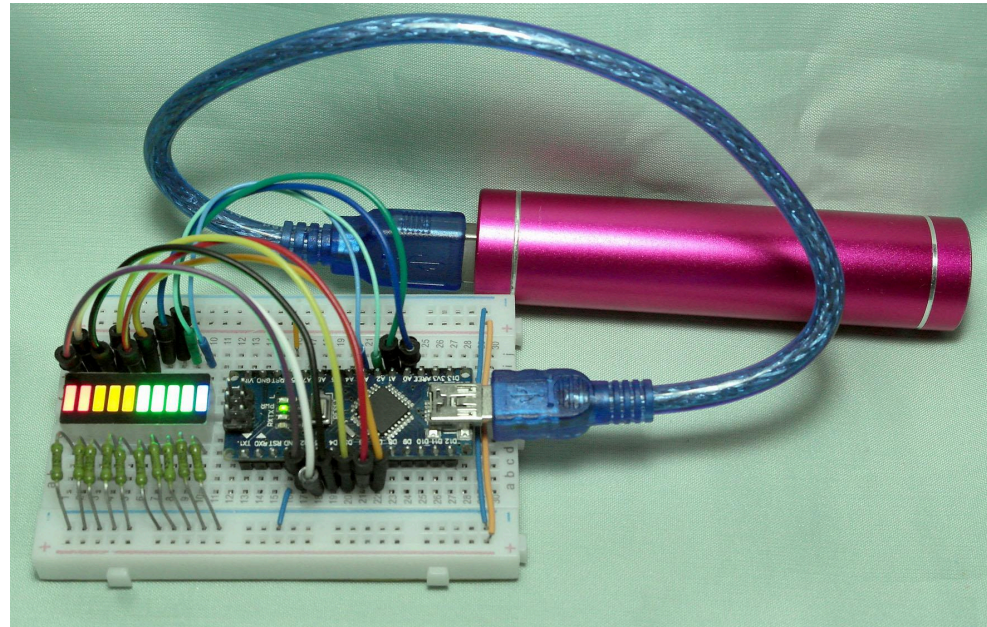
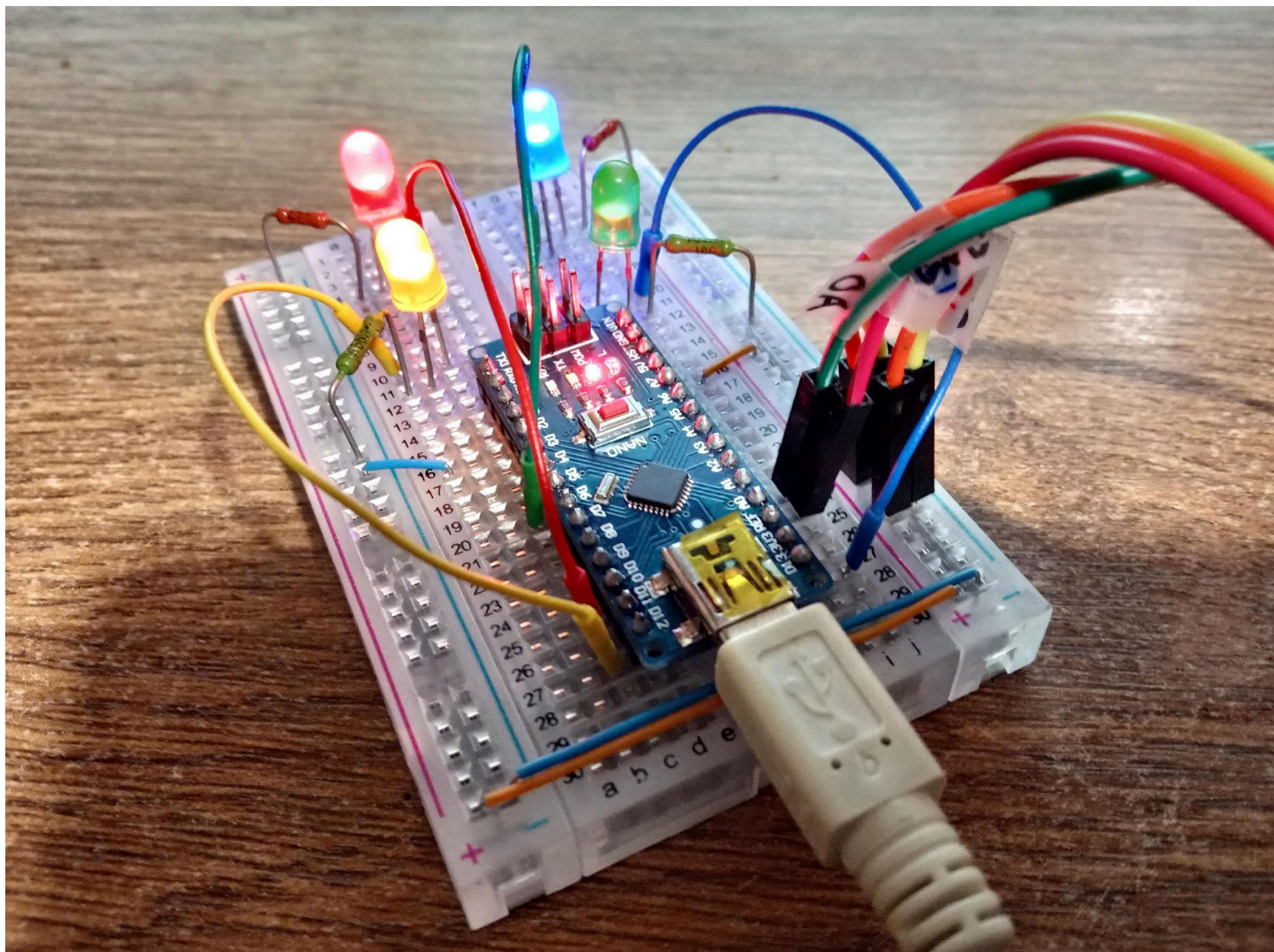


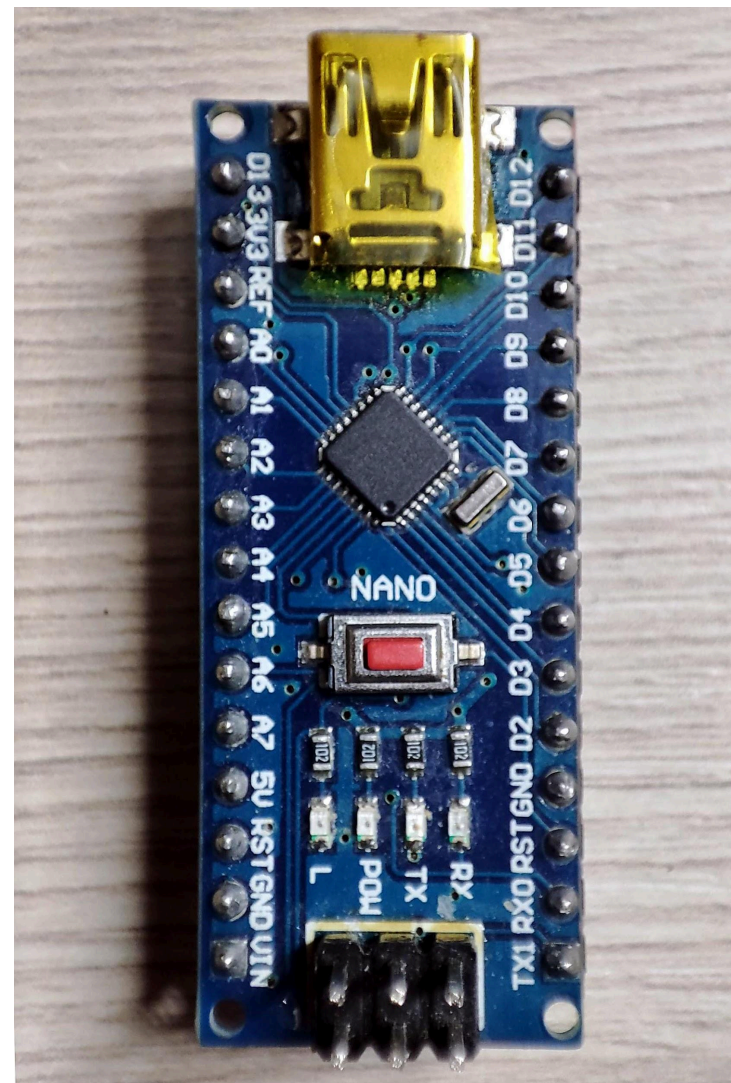
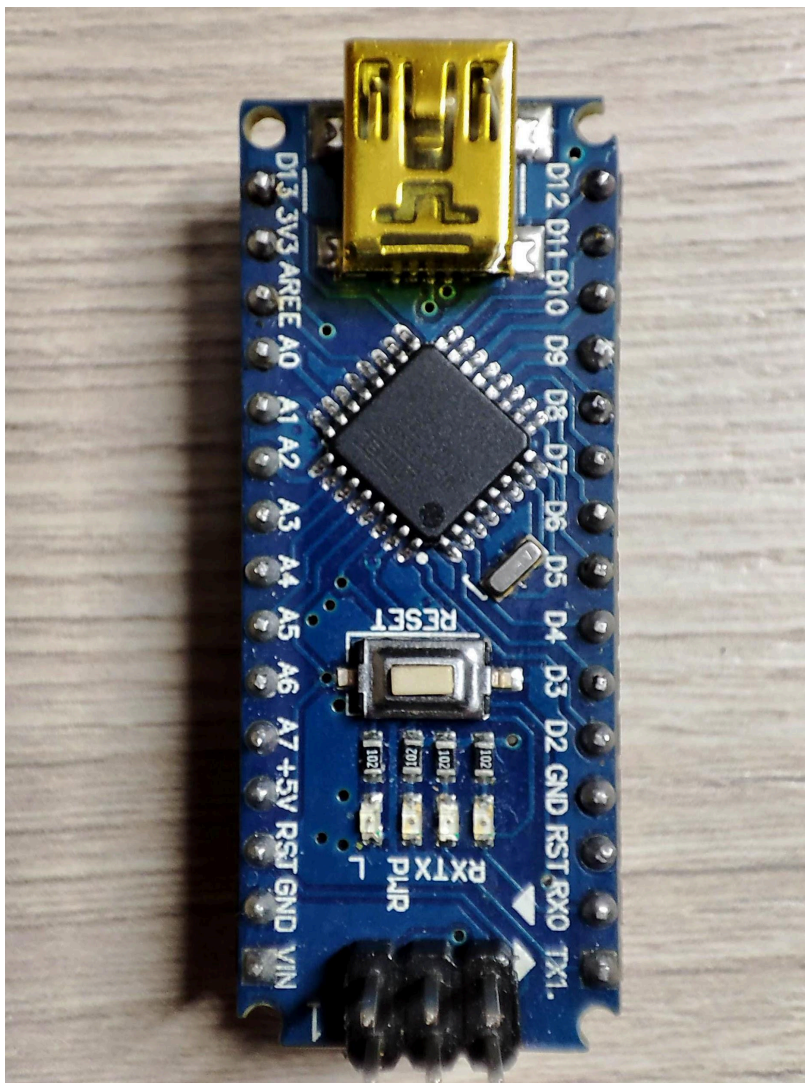
ПЛАТЫ «ARDUINO» (НА ОСНОВЕ МК ATMEGA328P)



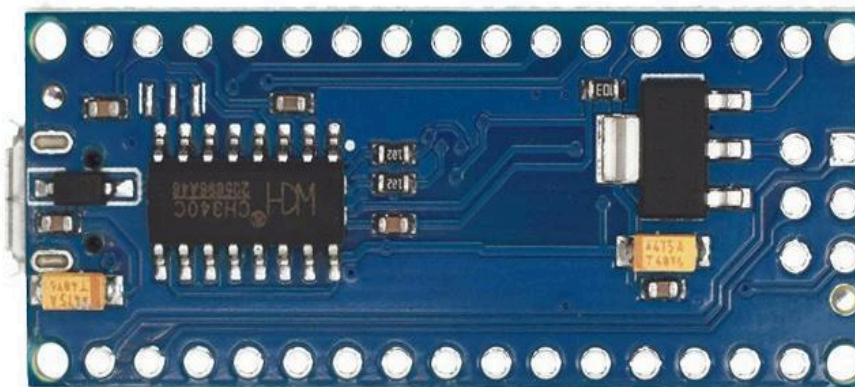
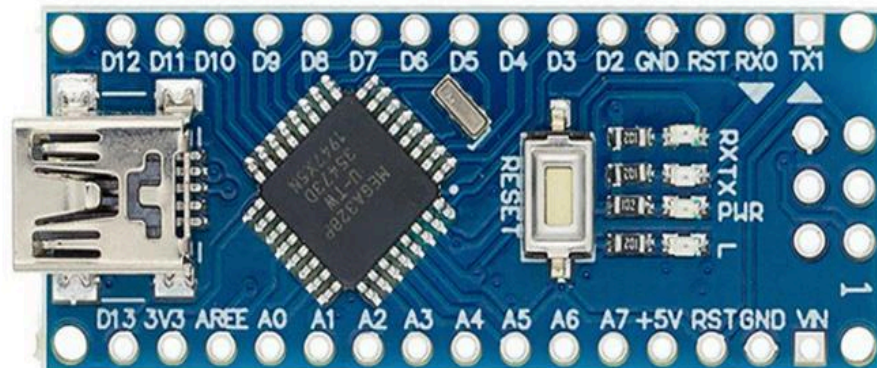
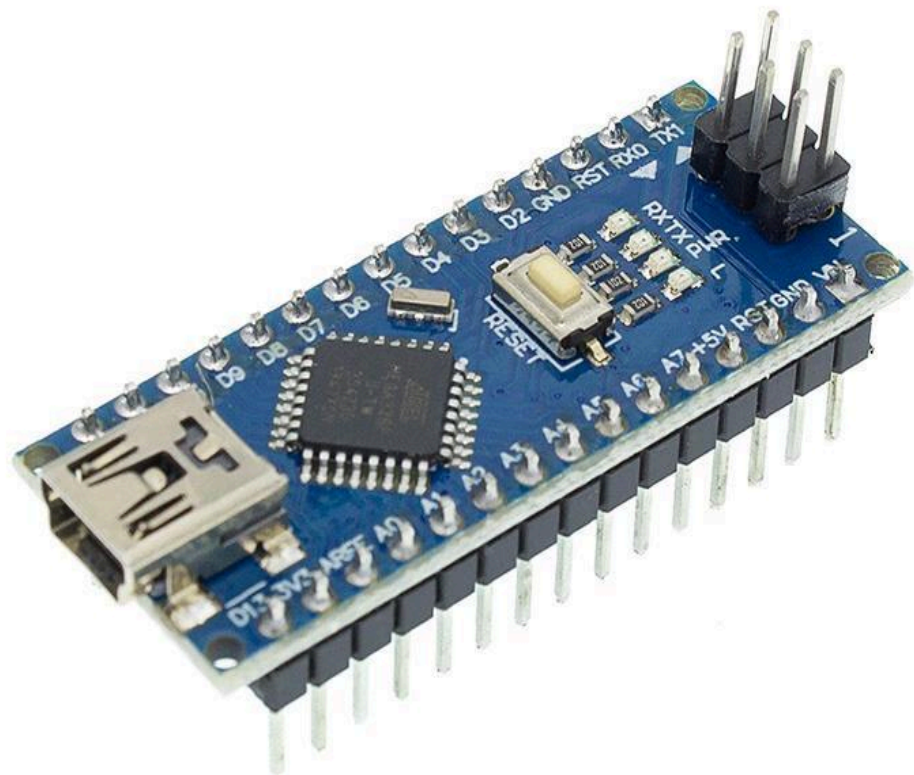




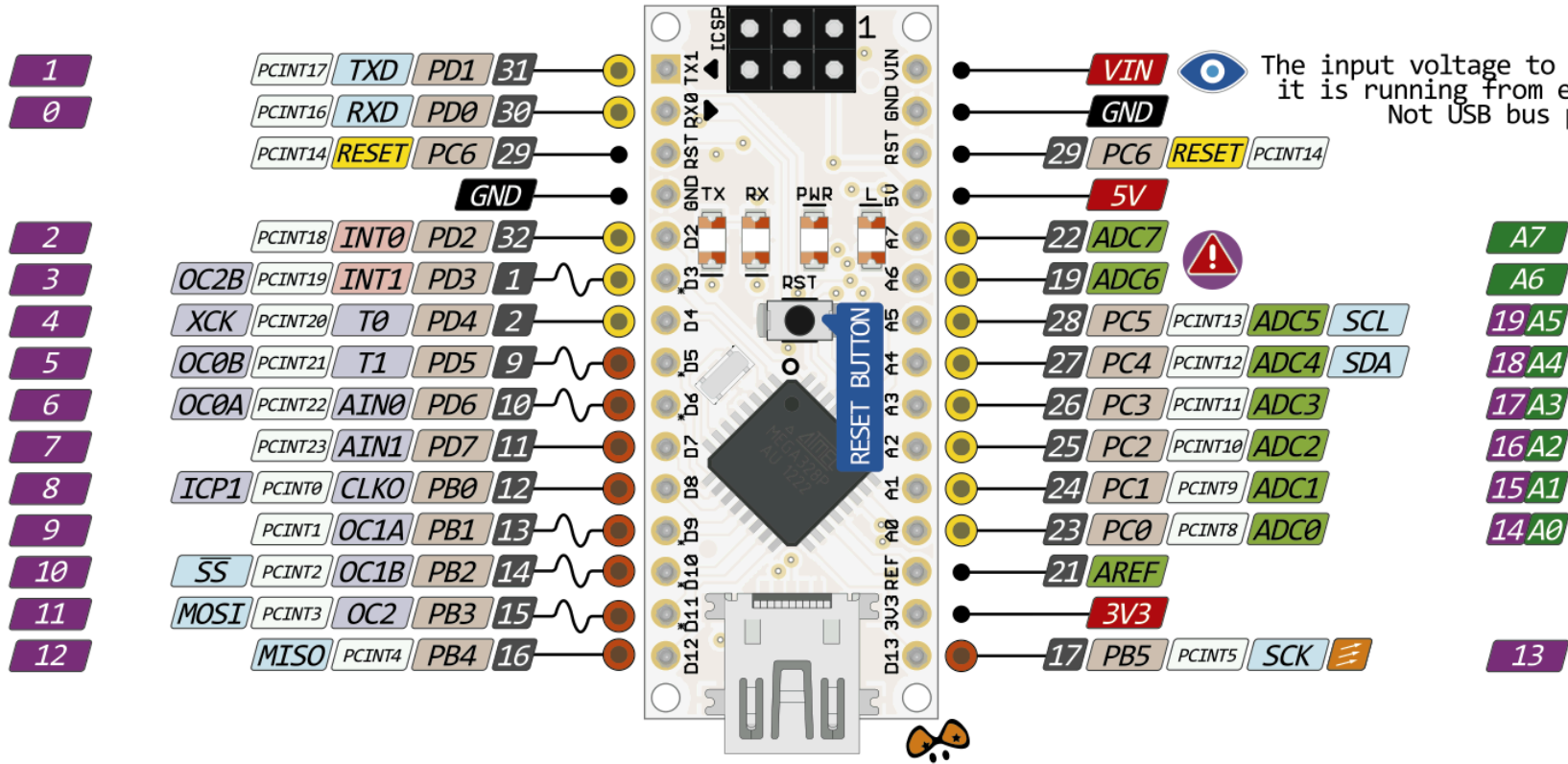
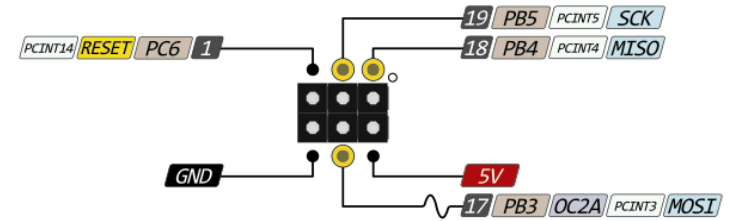
ARDUINO NANO (ARDUINO IDE, CODE VISION AVR, PROTEUS)



IDE — Integrated Development Environment



NANO PINOUT



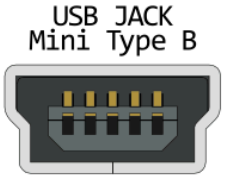
VIN The input voltage to the board when it is running from external power. Not USB bus power.

- Power
- GND
- Serial Pin
- Analog Pin
- Control
- INT
- Physical Pin
- Port Pin
- Pin function
- Interrupt Pin
- PWM Pin
- Port Power

The power sum for each pin's group should not exceed 100mA

Absolute MAX per pin 40mA recommended 20mA

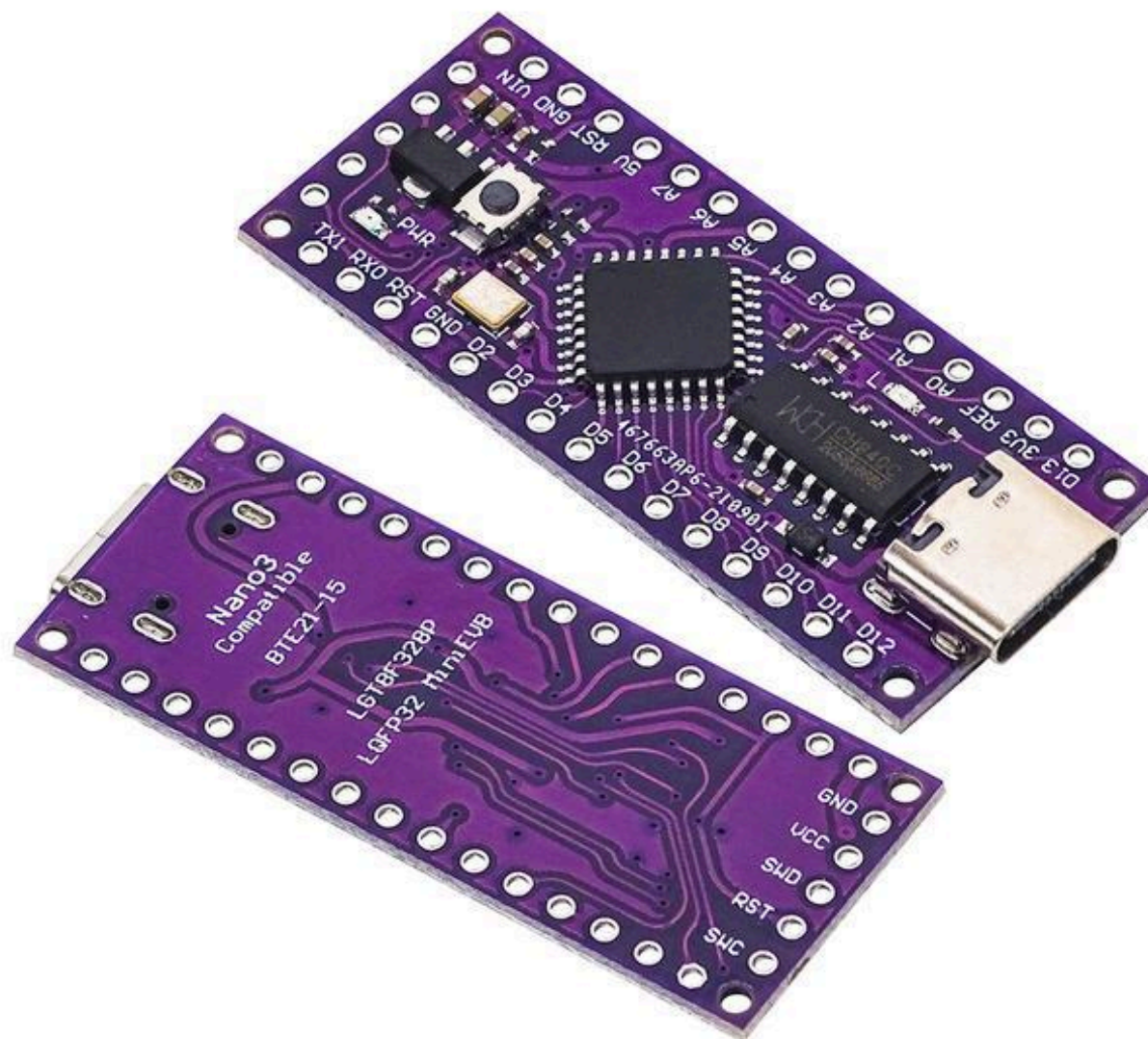
Absolute MAX 200mA for entire package



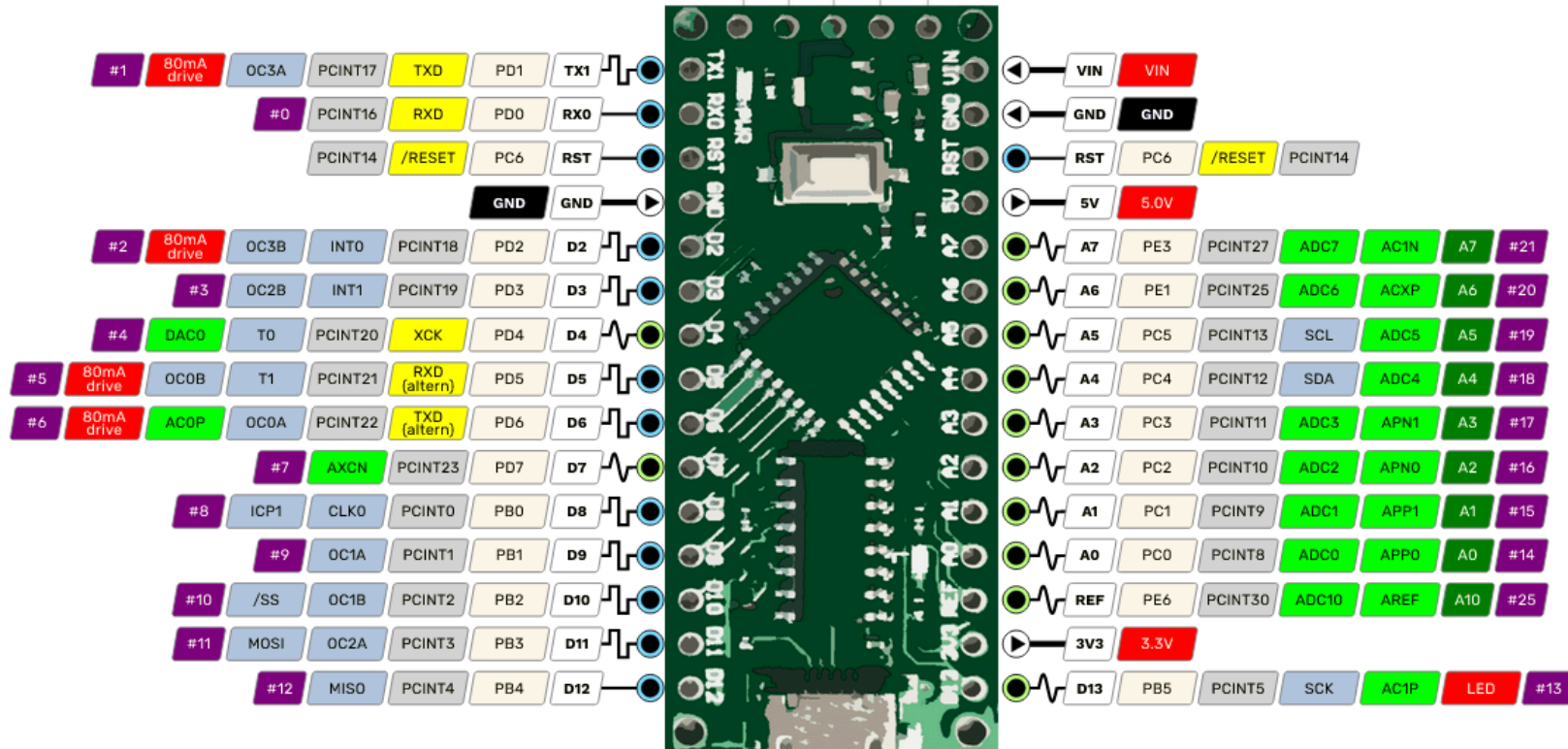
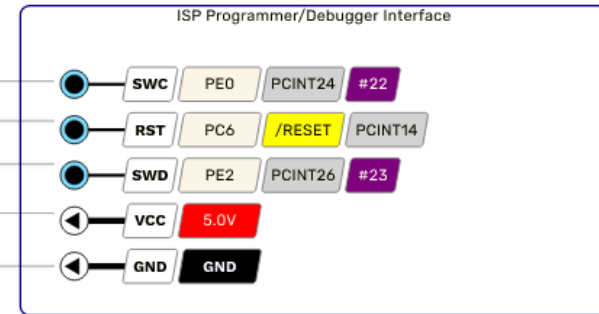
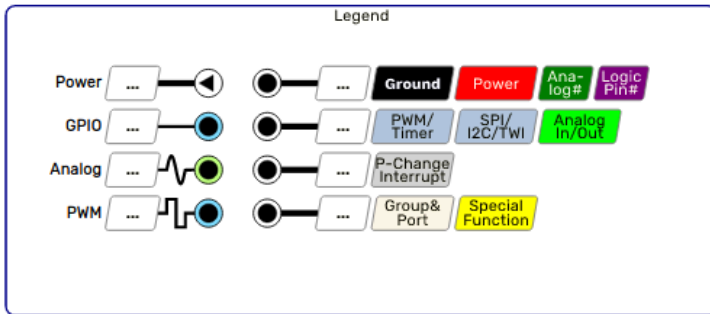
Analog exclusively Pins



LGT8F328P (ТОЛЬКО ARDUINO IDE)



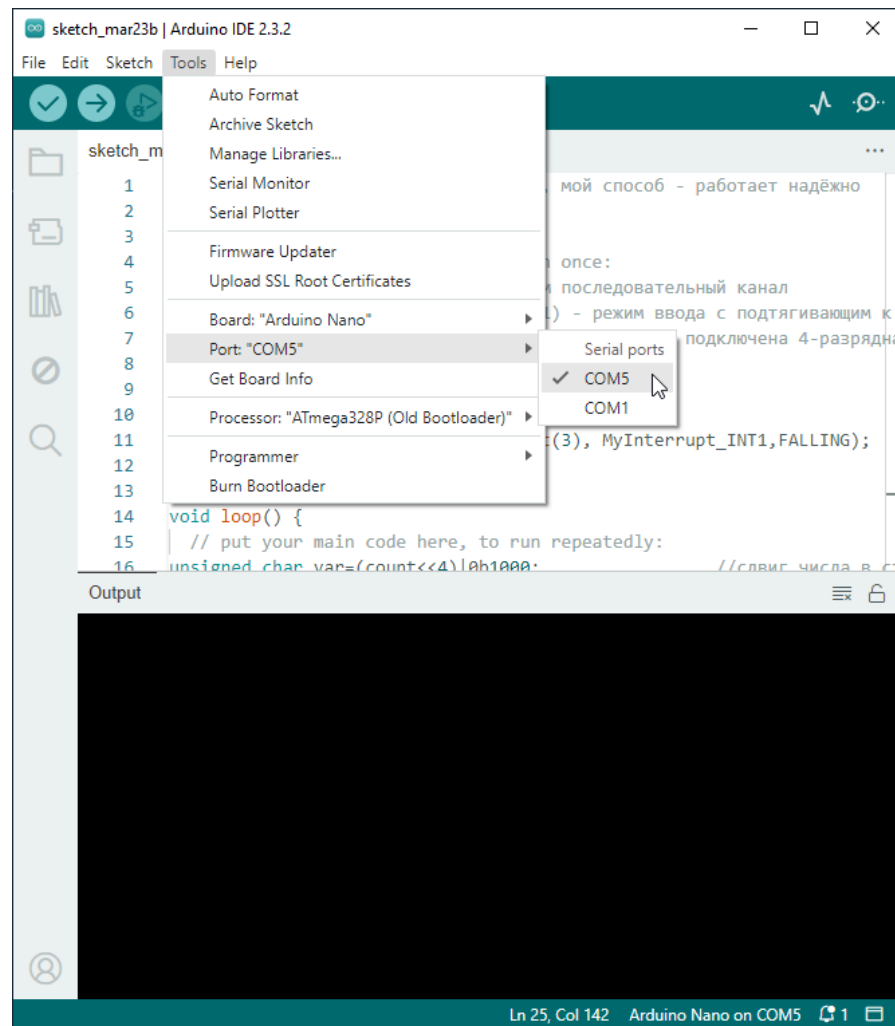
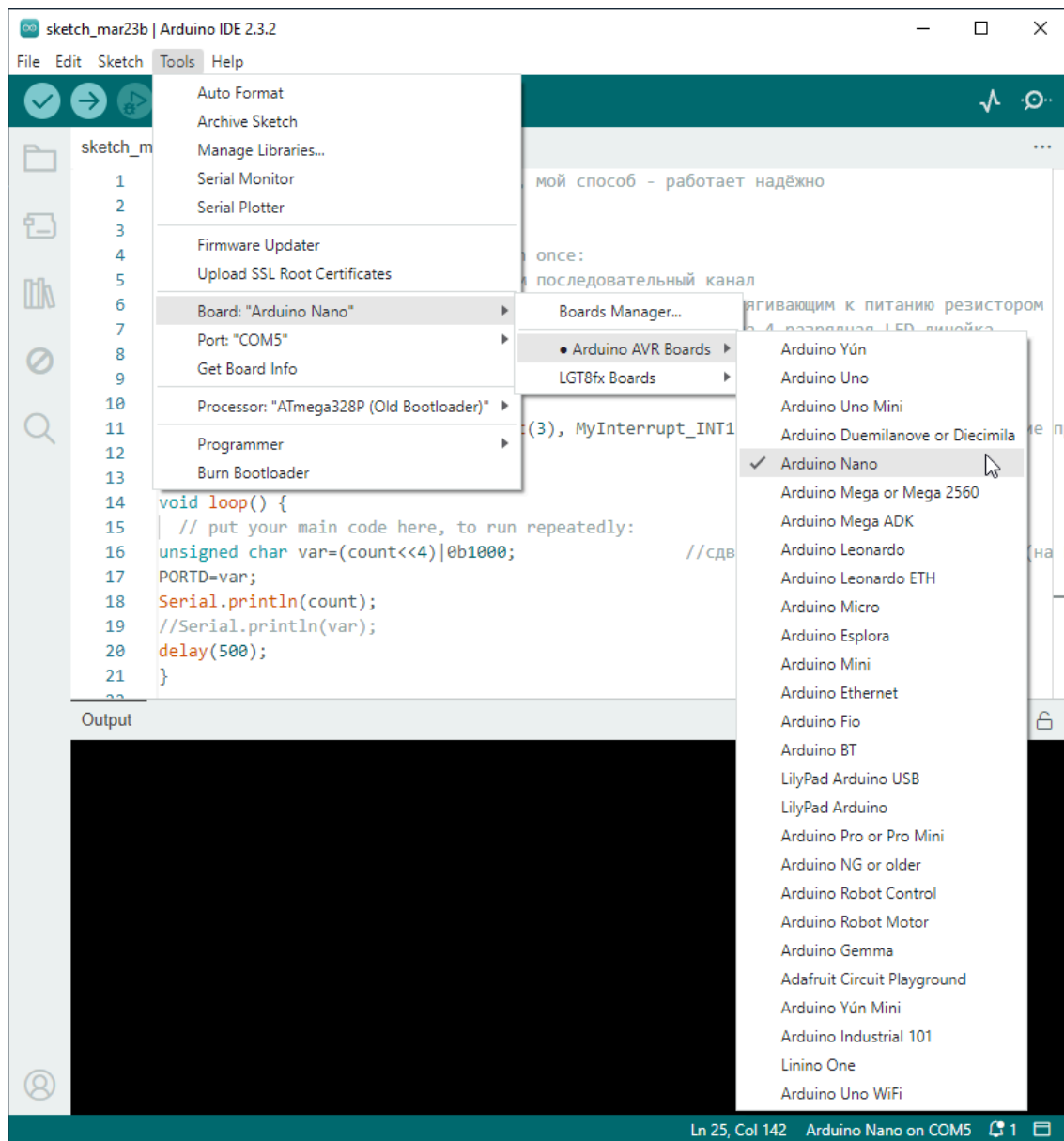
LGT8F328P Nano

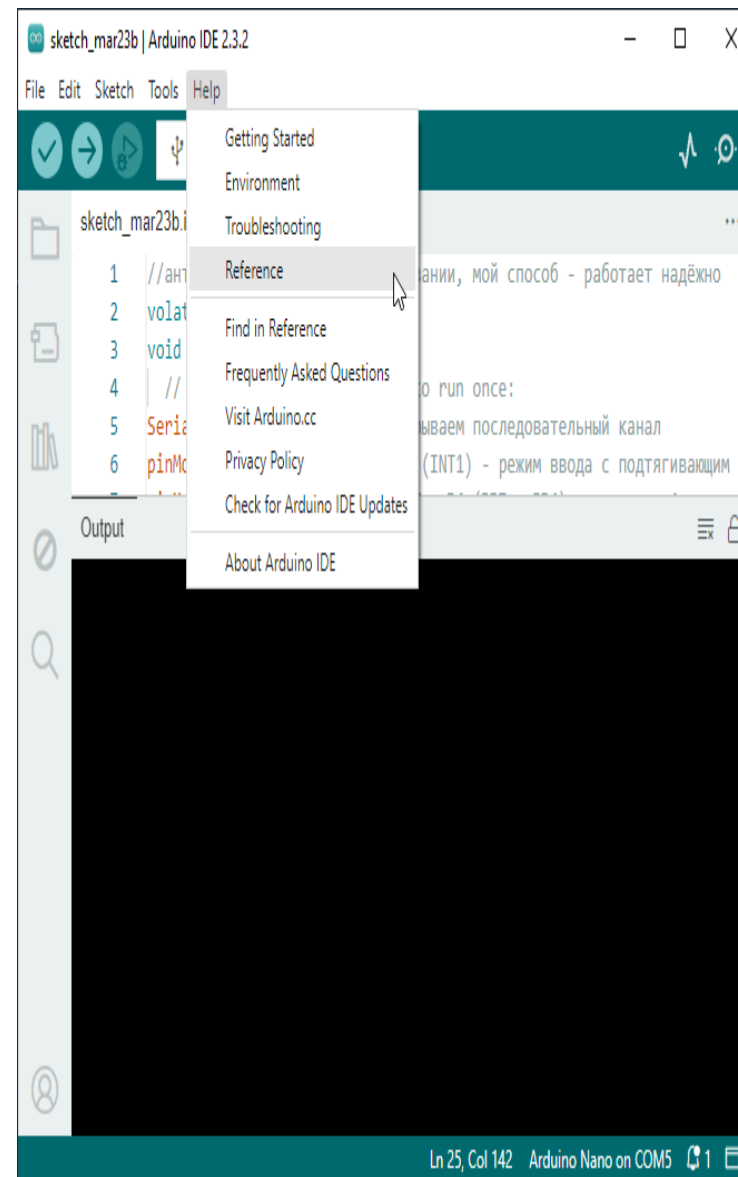
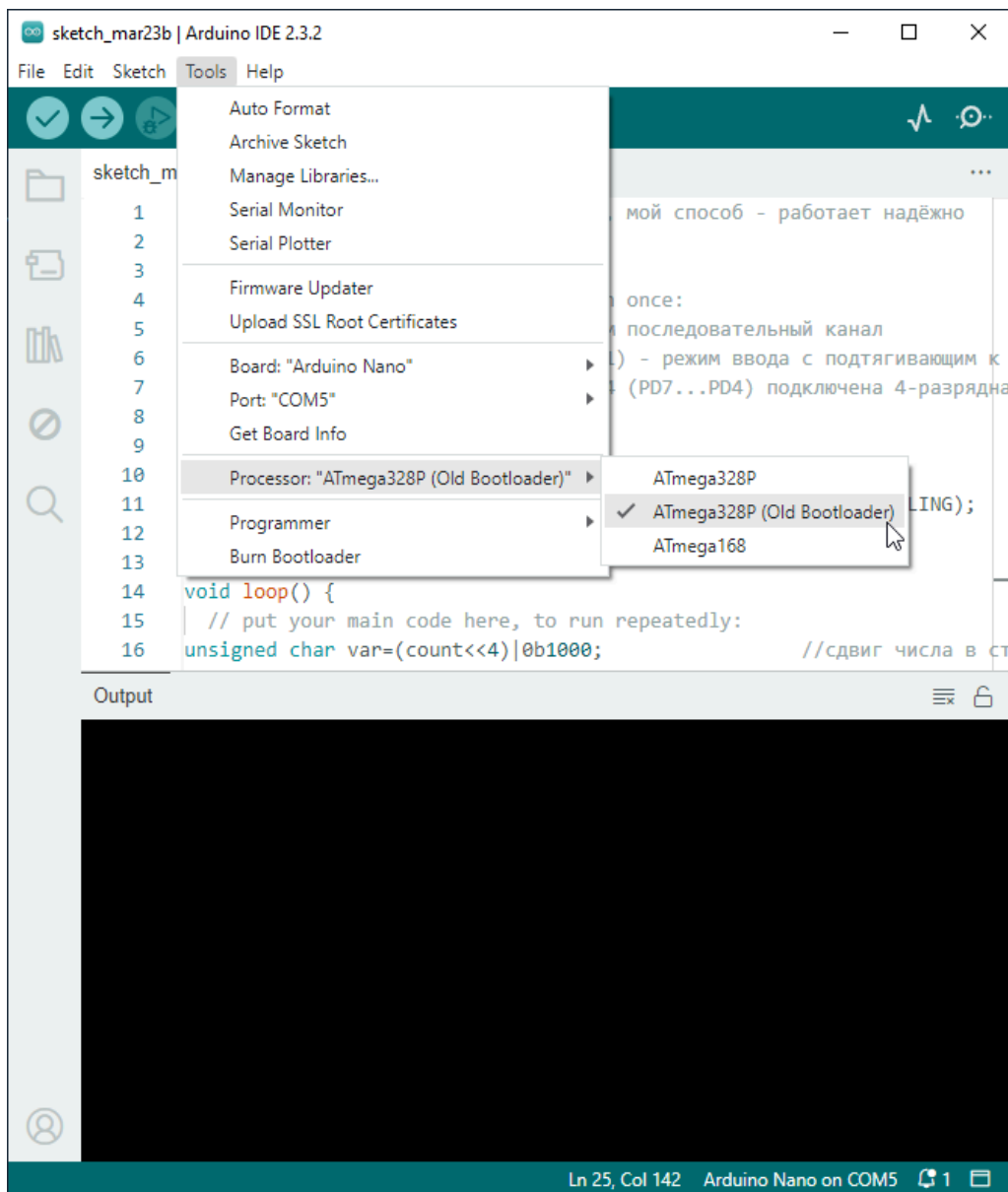


Arduino IDE <https://www.arduino.cc/en/software>

Arduino IDE 1.8.19 и старше для Windows 7 (русифицированные)

Arduino IDE 2.3.2 и новее для Windows 10 (англоязычная)





Новые платы *Nano* как правило поступают с новым загрузчиком (*Bootloader*).
Используется плата со старым загрузчиком (*Old Bootloader*) для возможности прошивки МК и из среды полнофункциональной CodeVisionAVR V3.12.
Работа в среде *CodeVisionAVR V3.12* подробно описана здесь [«Методические рекомендации по РГР по предмету «Основы микропроцессорной техники» \(приложение А\)](#).

Работа в *Proteus 7.10* (система моделирования электронных устройств) описана здесь [«Краткие сведения о модуле ISIS пакета программ PROTEUS»](#).

Кроме того, есть **видеолекции** (работа в Proteus 7, 8; программирование МК AVR в среде AVRstudio на ассемблере и в среде Code Vision AVR V3.12), ссылки на playlists:

[Часть 1 «Основы микропроцессорной техники»](#)

[Часть 2 «Аппаратные средства микроконтроллеров»](#)

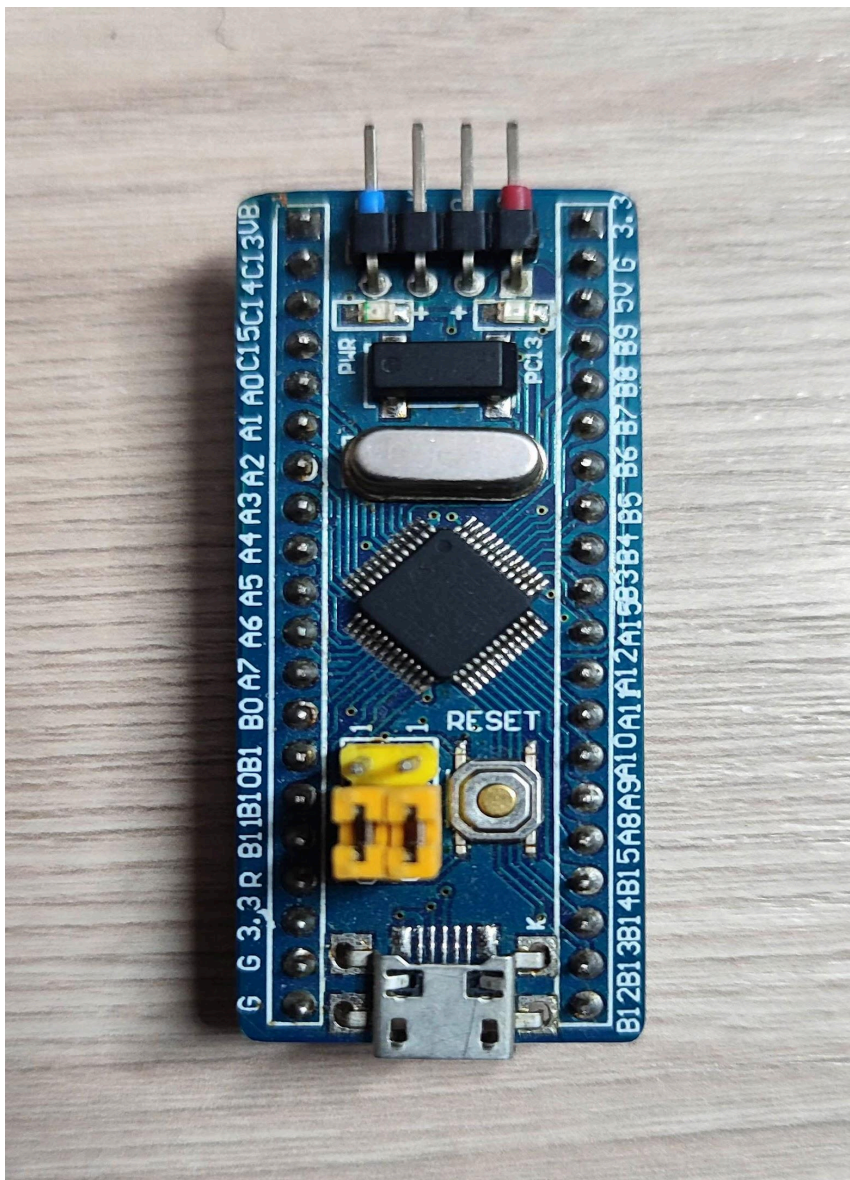
Дистрибутивы:

[Code Vision AVR V3.12](#) (полнофункциональная)

[Proteus 7.10](#), [Proteus 8.15](#), [Proteus 8.9](#).

Пароль архива (при необходимости) – **ЭиМТ**

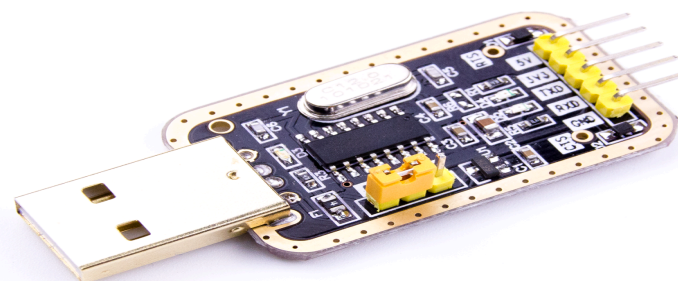
STM32F103C8T6 (ARDUINO IDE, KEIL + CUBEMX)



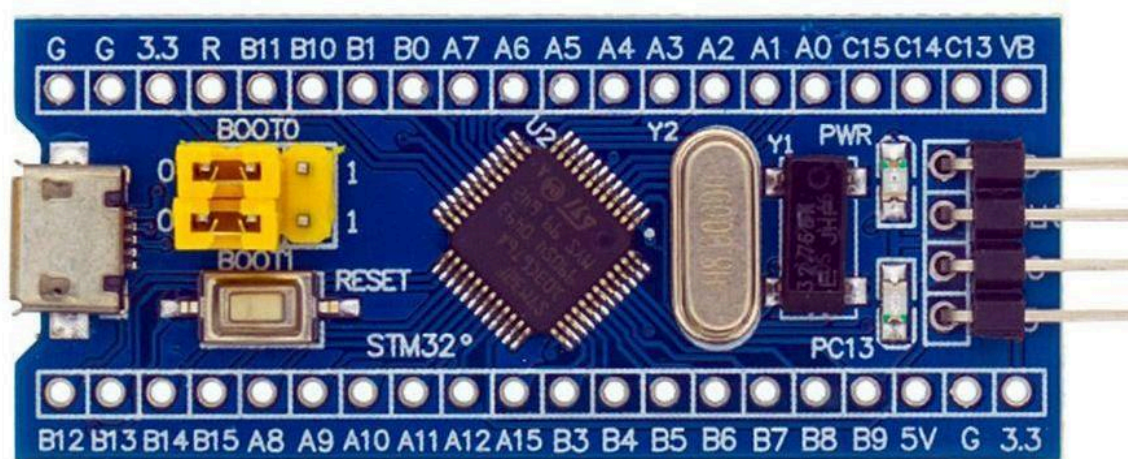
Программатор с отладочным интерфейсом

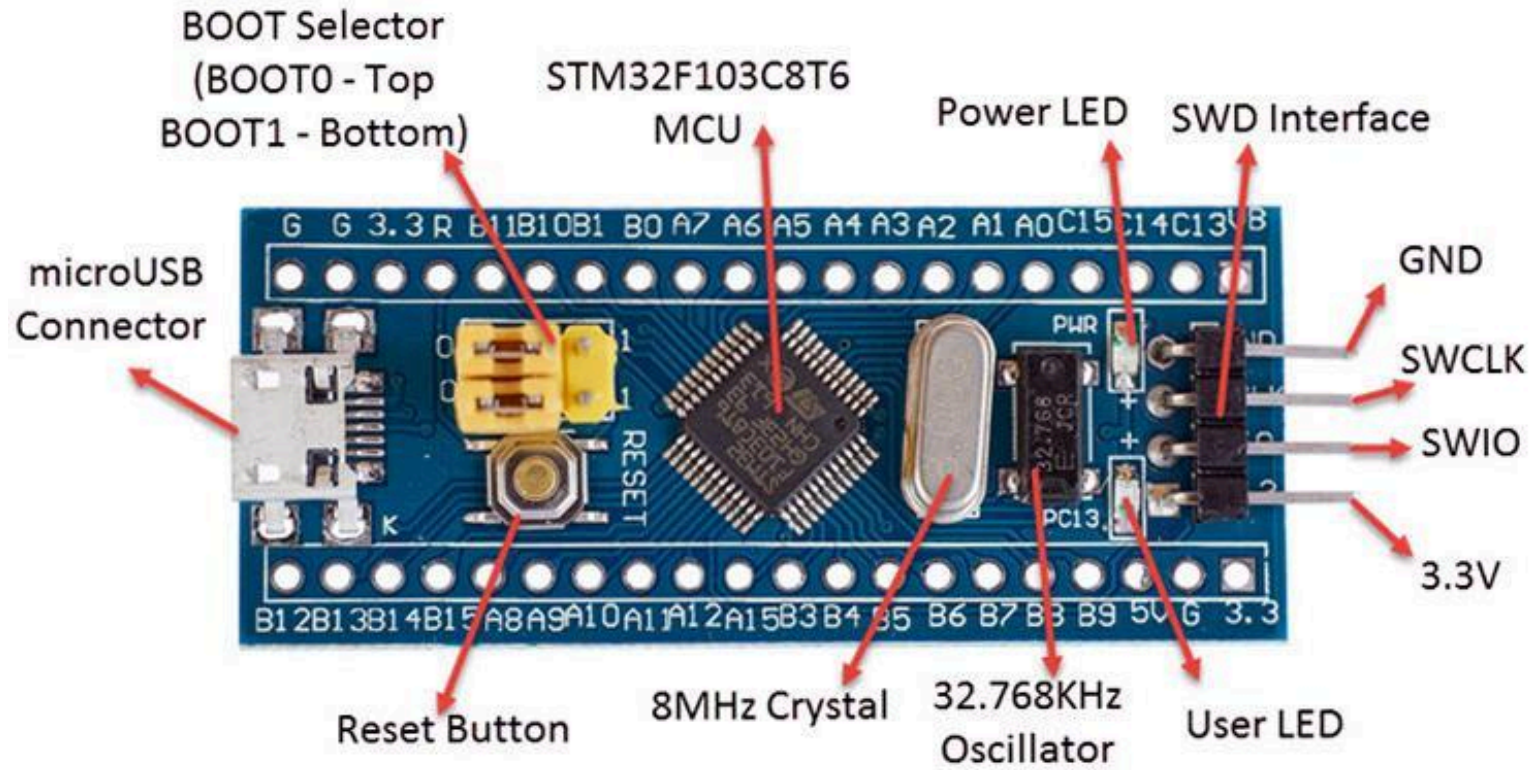


Переходник USB-UART



iarduino.ru





THE GENERIC STM32F103 PINOUT DIAGRAM

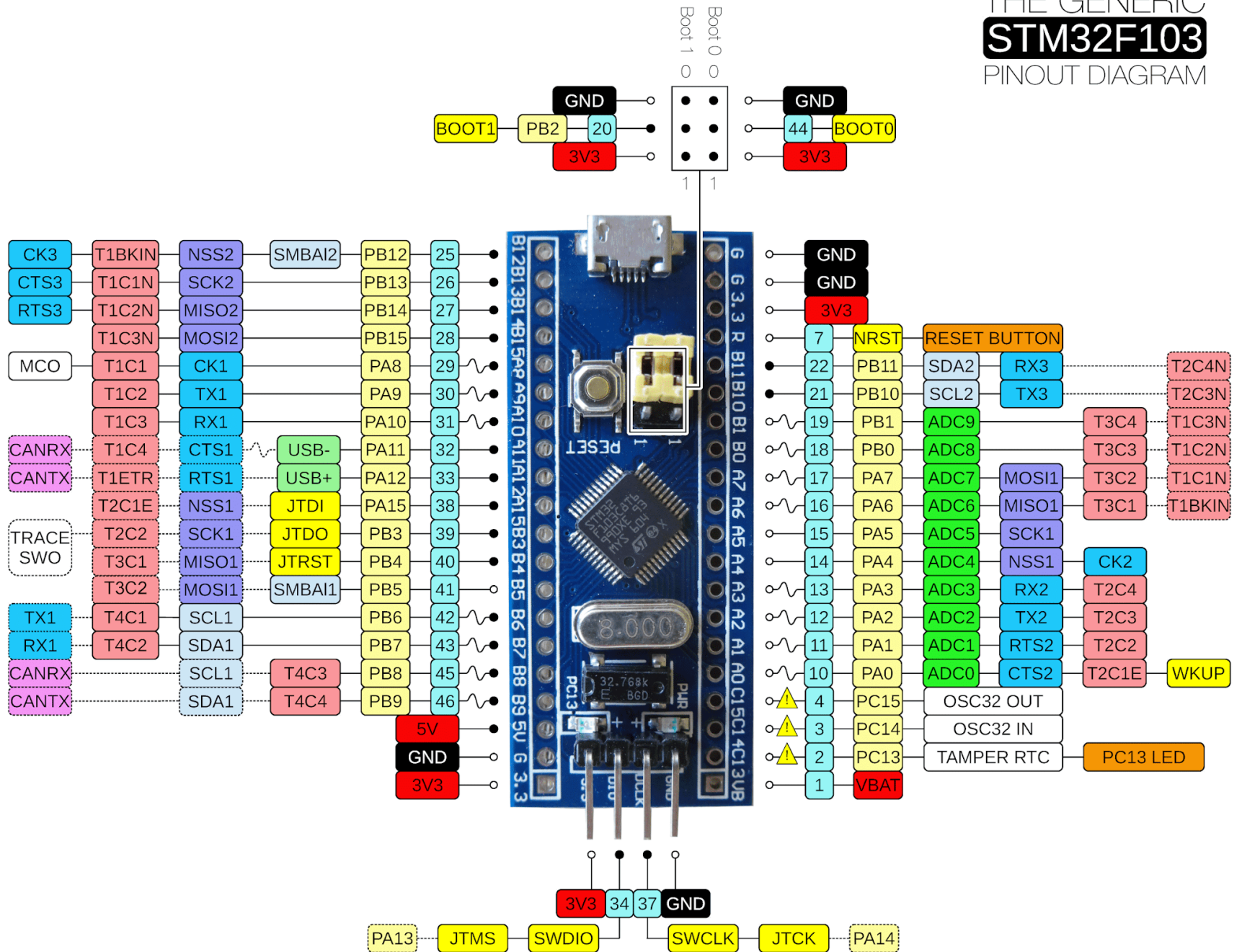
LEGEND

POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE

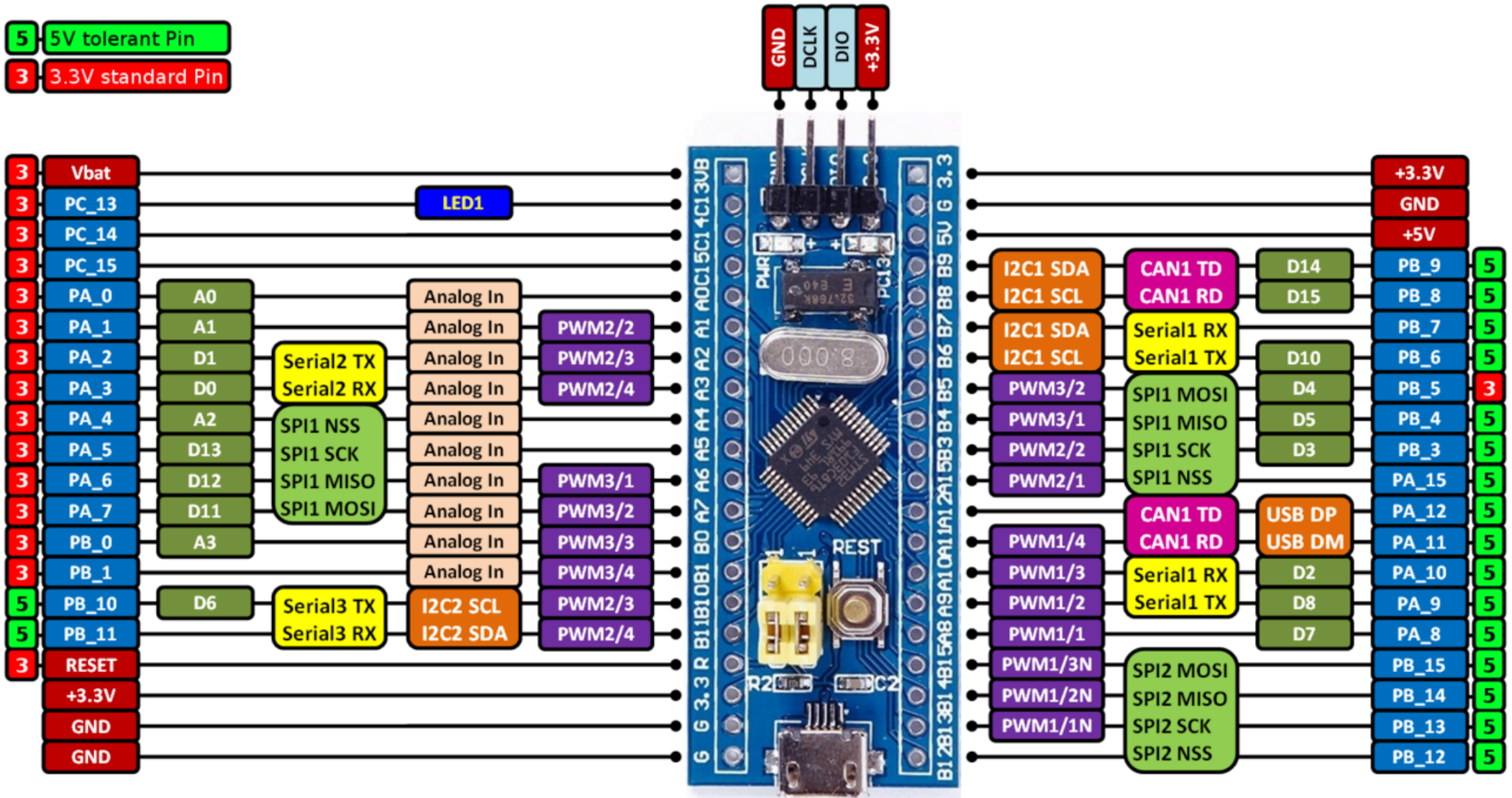
- 5V tolerant
- Not 5V tolerant
- ~ PWM pin
- ⋯ Alternate function
- ⚠ PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF

Absolute MAX 150mA total source/sink for entire CPU

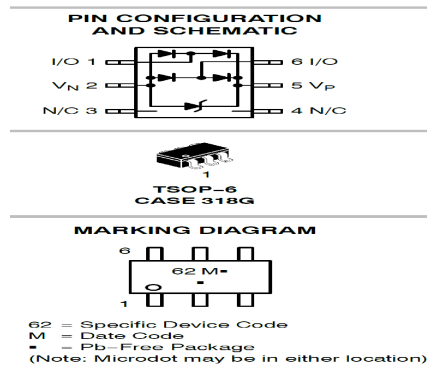
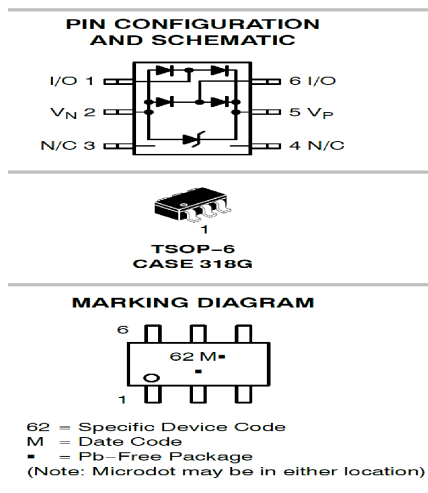
Max ±20mA per pin, ±8mA recommended



- 5** 5V tolerant Pin
- 3** 3.3V standard Pin

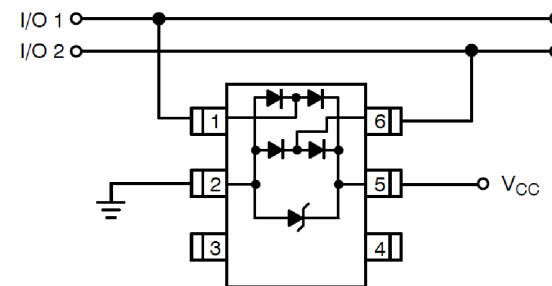


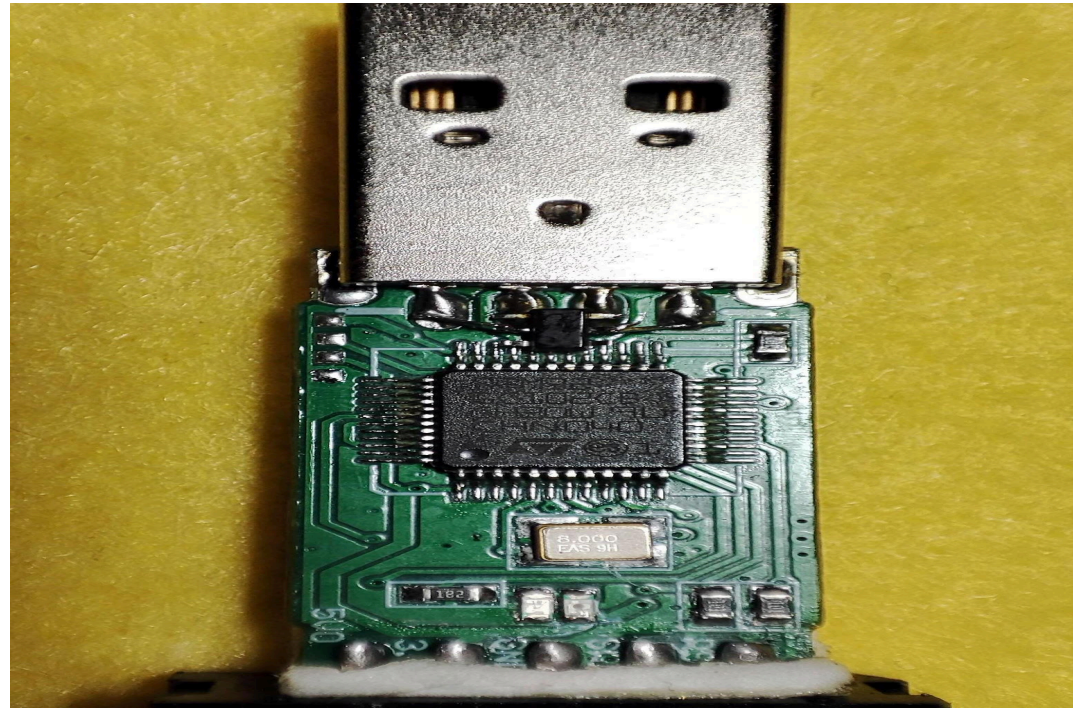
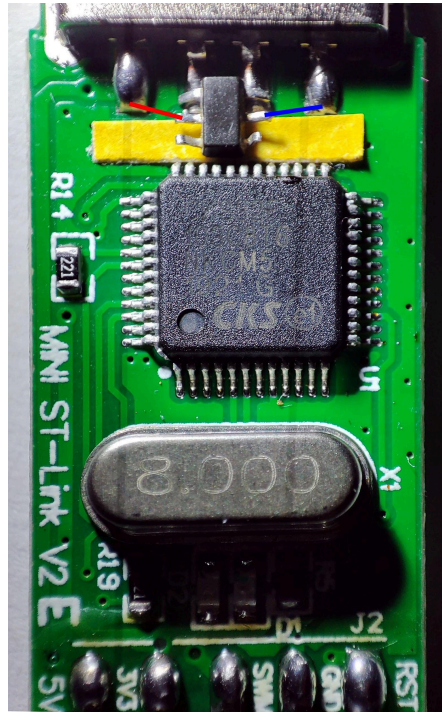
ДОРАБОТКА КИТАЙСКОГО ST-LINK (ПРЕДОТВРАЩЕНИЕ ПРОБОЯ СТАТИЧЕСКИМ ЭЛЕКТРИЧЕСТВОМ С ПОМОЩЬЮ САПРЕССОРА NUP2201MR6)



Option 1

Protection of two data lines and the power supply using V_{CC} as reference.





ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ РАБОТЫ С MCU STM32

Ссылка для скачивания генератора предварительного кода **Cube MX**:
<https://www.st.com/en/development-tools/stm32cubemx.html>

Ссылка для скачивания генератора предварительного кода **Cube MX V6.9.2**:
<https://cloud.mail.ru/public/Jhvx/kpokQhPVw>

Ссылка для скачивания последней *evaluation version* среды разработки (IDE) программ на языке C для микроконтроллеров с ядром ARM **Keil MDK**:

<https://www.keil.com/demo/eval/arm.htm>

Ссылка на **MDK538a** (Keil 5.38 – предыдущая версия, поддерживает китайский STlink):

<https://cloud.mail.ru/public/J3Fr/bA3wczkRK>

Программа для ST-LINK:

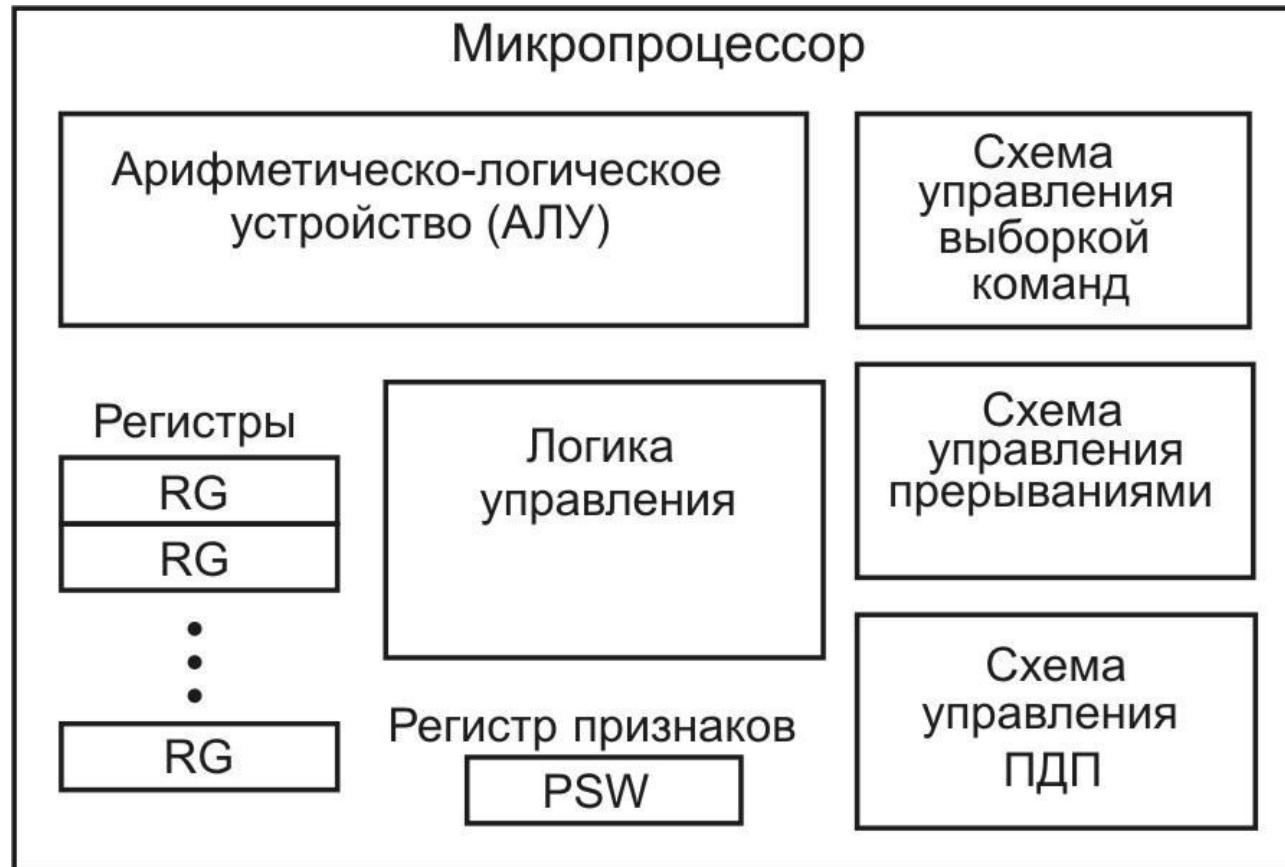
<https://cloud.mail.ru/public/m3qb/sA41rYePX>

Ссылки на *Flash Loader* – прошивка flash-памяти МК через переходник USB-UART

<https://www.st.com/en/development-tools/flasher-stm32.html>

<https://cloud.mail.ru/public/m3qb/sA41rYePX>

УПРОЩЕННАЯ СТРУКТУРА ПРОЦЕССОРА

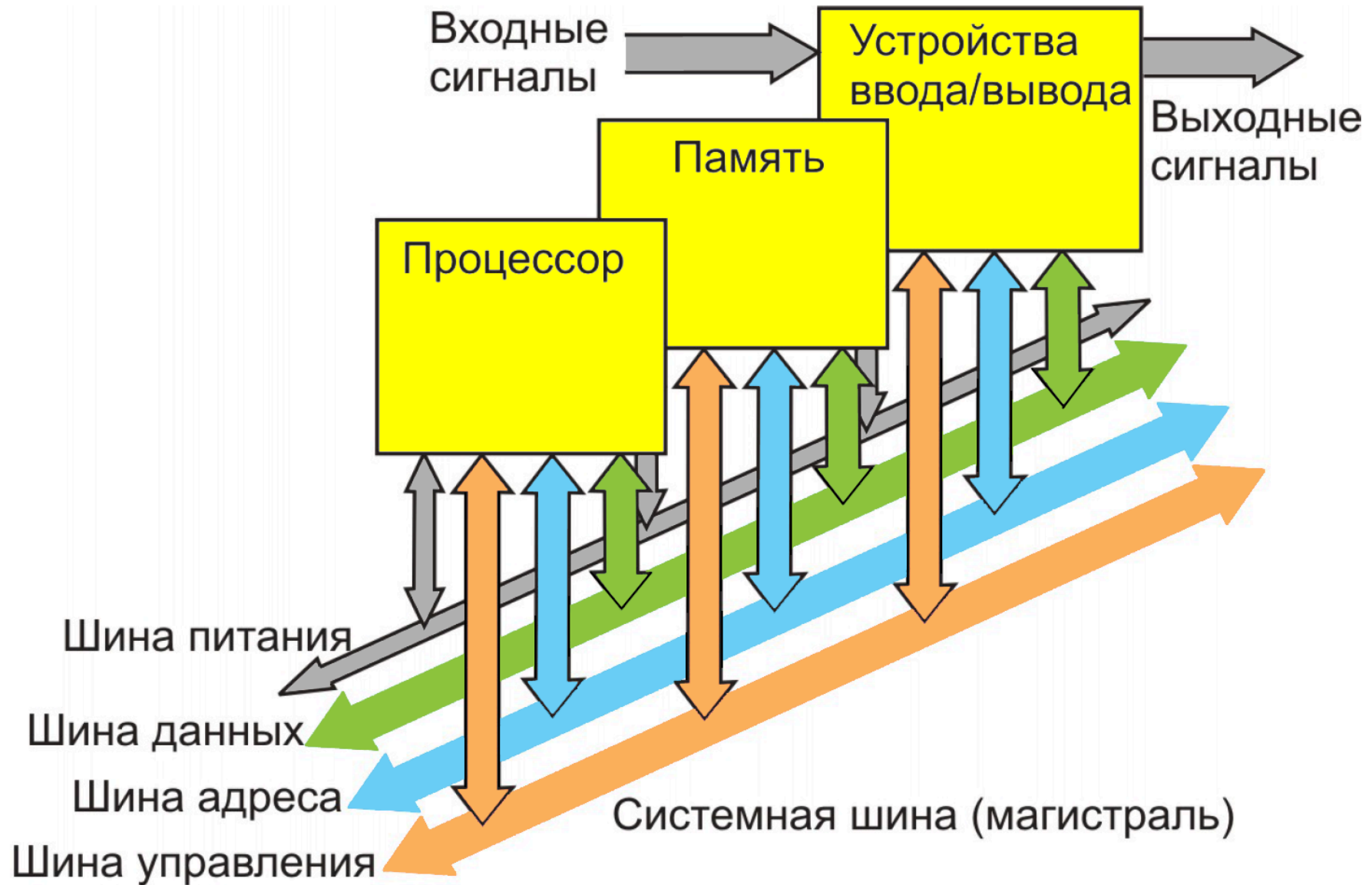


Основные функции ЦП, реализуемые при выполнении программы:

- 1) перенос данных (наиболее часто)
- 2) арифметические и логические операции над данными

3) управление последовательностью выполнения программы в зависимости от условий

ФУНКЦИОНИРОВАНИЕ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ



Работает микропроцессорная система следующим образом. После подачи питания или сигнала сброса RESET, процессор начинает последовательно

читать *программную память*, выполняя записанные в ней команды (коды операций). Это достигается с помощью специального устройства — *счетчика команд (Program Counter — PC)*, получающего в каждом новом машинном цикле приращение на 1. Начинается все после сброса с нуля (PC = 000). Таким образом, через программный счетчик PC (именно его содержимое выставляется на *шину адреса* в соответствующей фазе цикла) читается содержимое последовательных ячеек *программной памяти* (на *шину данных* в соответствующей фазе цикла выставляется их содержимое). После попадания через *шину данных* кода операции в процессор, машинная инструкция декодируется и выполняется. Машинные инструкции, которые выполняет процессор, предельно просты, например: прочитать ЯП с заданным адресом в регистр МП, записать содержимое регистра МП в ЯП с заданным адресом, прочитать данные из устройства ввода в регистр МП, записать данные из регистра МП в устройство вывода, элементарные арифметические, логические операции, операции сдвига над *операндами в регистрах МП*.

Часто в зависимости от результата операции при обработке данных, нарушается последовательное чтение программной памяти и происходит «прыжок» вперед или назад (условный переход). Т.е. реализуются две различные ветви алгоритма в зависимости от результата обработки. Создается иллюзия, что процессор принимает решение. При этом

программный счетчик нарушает свой привычный алгоритм работы (инкремент) и *загружается адресом перехода*. Нарушение обычного алгоритма работы *программного счетчика* происходит и при выполнении команд безусловных переходов, вызовов подпрограмм и возвратов из них, а также при реакции на аппаратные *запросы прерывания*.

Таким образом выполняется программа из элементарных машинных инструкций процессора, которые как правило помещаются в 1-2 слова программной памяти. В процессе выполнения такой программы процессор также обменивается данными с *внешними устройствами*: читает данные из устройства ввода, записывает данные в устройства вывода.

То есть процессор под управлением программы реализует функции управления системой — получает сигналы и коды из внешнего мира (с помощью устройств ввода), обрабатывает их (используя арифметико-логические функции) и выводит воздействия в исполнительные устройства, а обработанные данные в устройства отображения (устройства вывода).

ФУНКЦИИ СОСТАВНЫХ ЧАСТЕЙ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ

- **Процессор** — обработчик, выполняет пересылку данных и их обработку (арифметические и логические операции) в соответствии с программой; управляет выборкой команд, изменяет последовательность выполнения команд программы;
- **Память** — оперативная (RAM, ОЗУ) и постоянная (ПЗУ, ROM, FLASHROM,) хранит данные и программы. *Оперативная* — для временного хранения данных, *постоянная* — для постоянного хранения программ и реже данных (констант), главное — для программы начального запуска при включении питания;
- **Устройства ввода/вывода** (УВВ, I/O — Input/Output) — служат для обеспечения связи микропроцессорной системы с внешними устройствами и с пользователем (внешние интерфейсы и пользовательский интерфейс). Они же помогают процессору в пересылке данных и в реагировании на внешние события.

Функции шин микропроцессорной системы

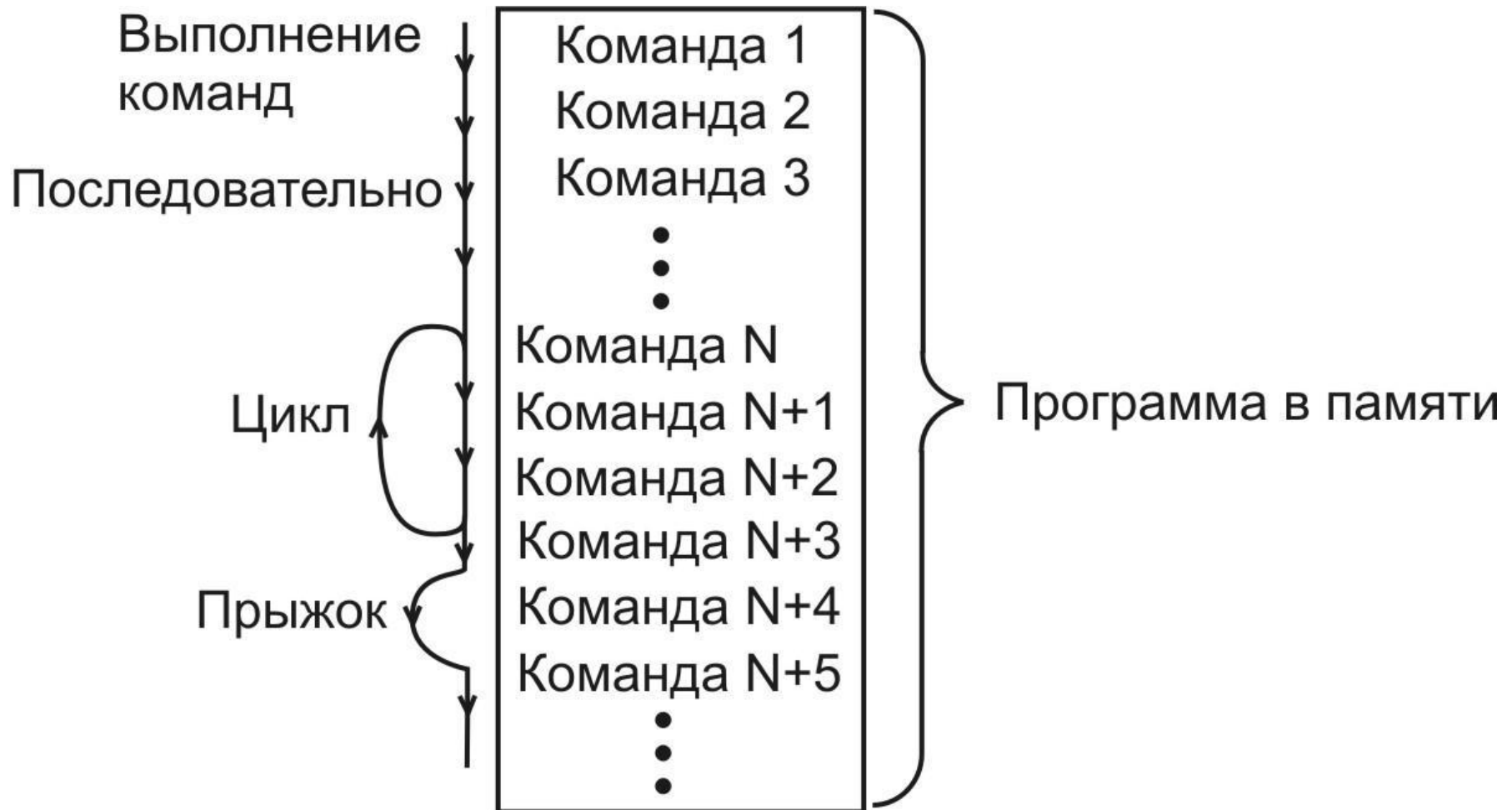
- **Шина адреса** (*Address Bus, AB, ША*) — для пересылки кода адреса (индивидуального номера устройства или ячейки памяти, участвующего в обмене в данный момент). Её разрядность определяет допустимый объем памяти и, следовательно, максимально возможный размер программы и максимально возможный объем запоминаемых данных. Количество адресов, обеспечиваемых шиной адреса, определяется как 2^k , где k — количество разрядов шины адреса.
- **Шина данных** (*Data Bus, DB, ШД*) — для пересылки данных между устройствами. Двухнаправленная шина, так как предполагает передачу информации в обоих направлениях, состоит из нескольких байтов (1, 2, 4, 8) в зависимости от разрядности системы 8, 16, 32, 64. Разрядность ШД (количество линий связи) определяет скорость и эффективность информационного обмена, а также максимально возможное количество команд процессора. Тип *выходного* каскада для линий этой шины — выход с тремя состояниями.
- **Шина управления** (*Control Bus, CB, ШУ*) — для пересылки отдельных управляющих сигналов: тактовых, стробирующих, подтверждающих, инициирующих (сигнал считывания \overline{RD} ; сигнал записи \overline{WR} , и др. сигналы специфичные для конкретного типа МП, МК).

- **Шина питания** (*Power Bus*) — для подведения к устройствам напряжений питания (положительных, отрицательных, общего провода).

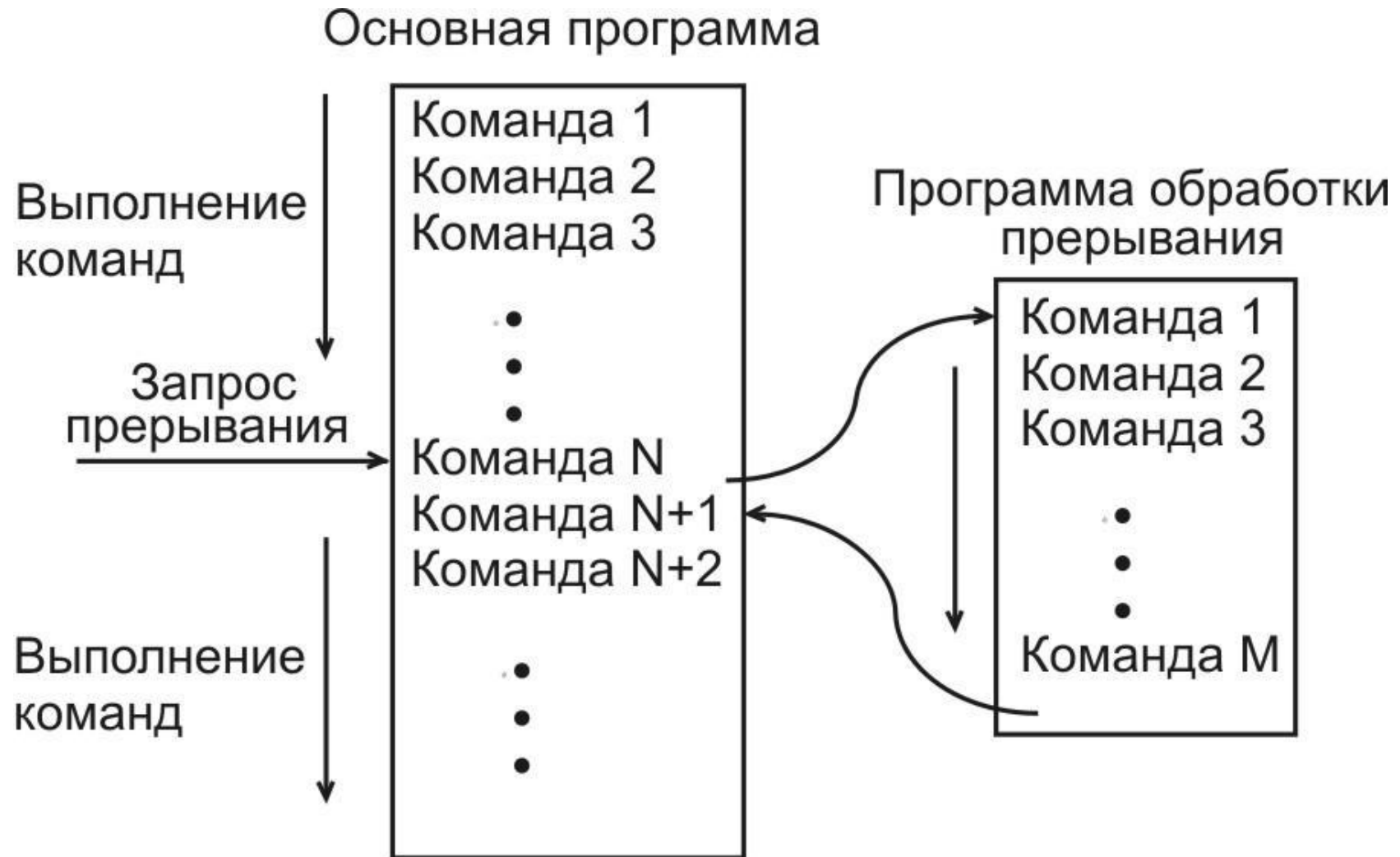
ЦИКЛЫ ОБМЕНА В МИКРОПРОЦЕССОРНОЙ СИСТЕМЕ

- **Программные** циклы обмена:
 - выборка команды из памяти (чтение, ввод);
 - чтение данных из памяти (чтение, ввод);
 - запись данных в память (запись, вывод);
 - приём данных из устройства ввода/вывода (чтение, ввод);
 - передача данных в устройство ввода/вывода (запись, вывод);
- Циклы обмена по **прерываниям** (*Interrupts*);
- Циклы обмена по **прямому доступу к памяти** (ПДП, DMA – *Direct Memory Access*).

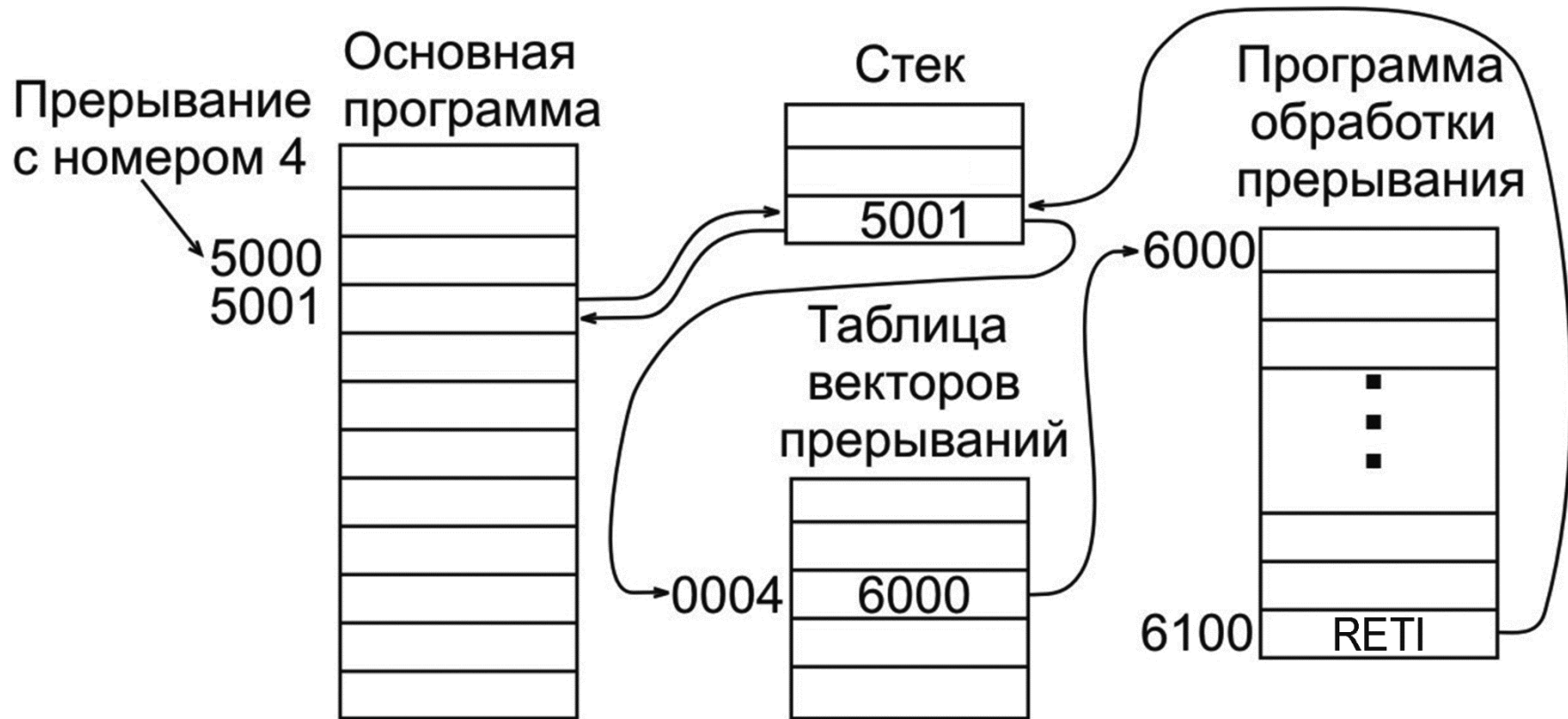
Выборка команд — выполнение программы (иллюстрация программного цикла)



Обслуживание прерывания (Interrupt)



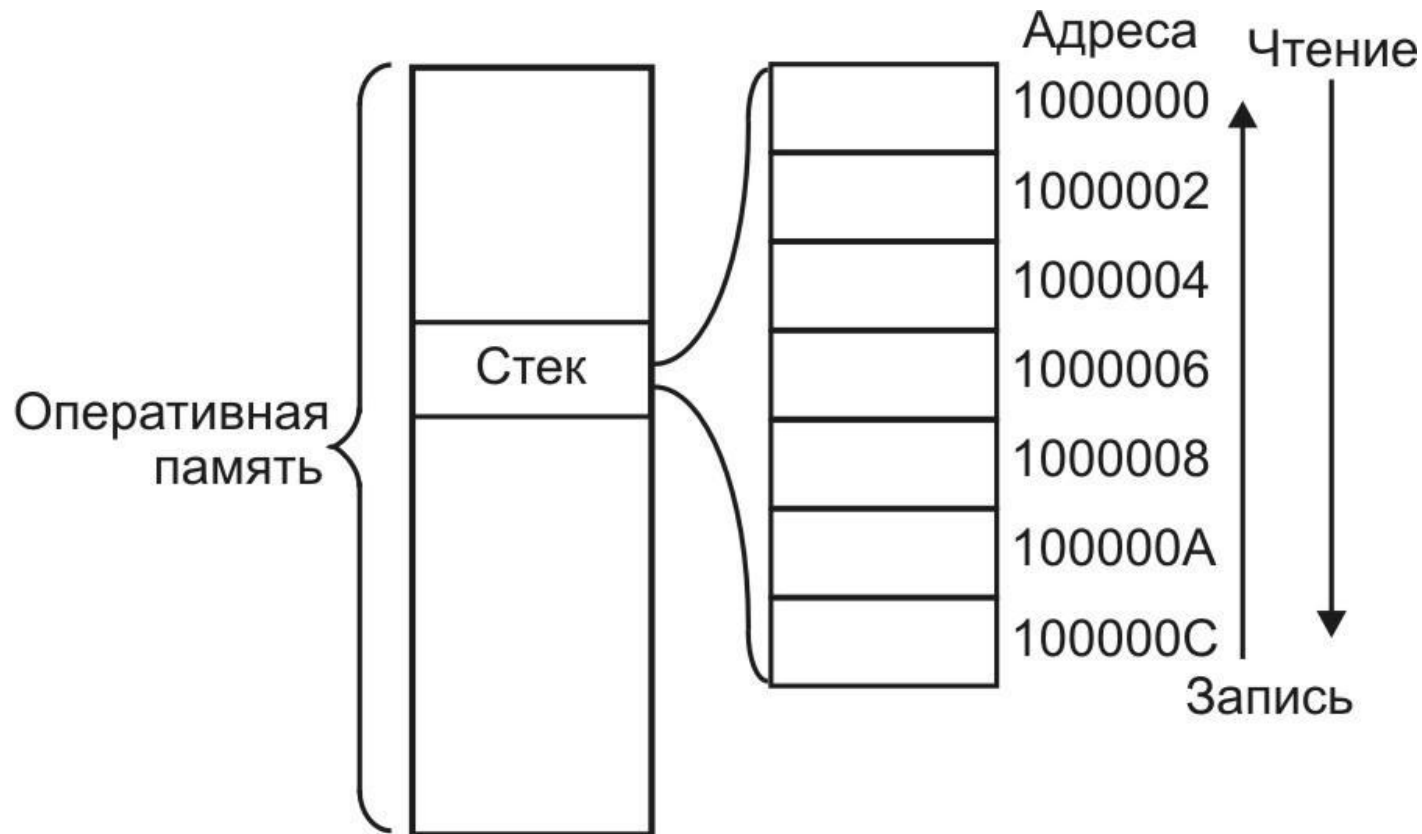
Механизм обработки прерывания (более подробно)



Часто при переходе к подпрограмме обработки прерывания (или просто подпрограмме) дополнительно программистом предусматривается сохранение в стеке слова состояния процессора PSW и важных РОН. При этом следует помнить, что перед возвратом восстановление регистров должно идти в порядке, обратном сохранению из-за принципа работы стека.

За счет использования стека реализуется и механизм вложенности подпрограмм (в том числе и прерываний).

Принцип работы стековой памяти (STACK)

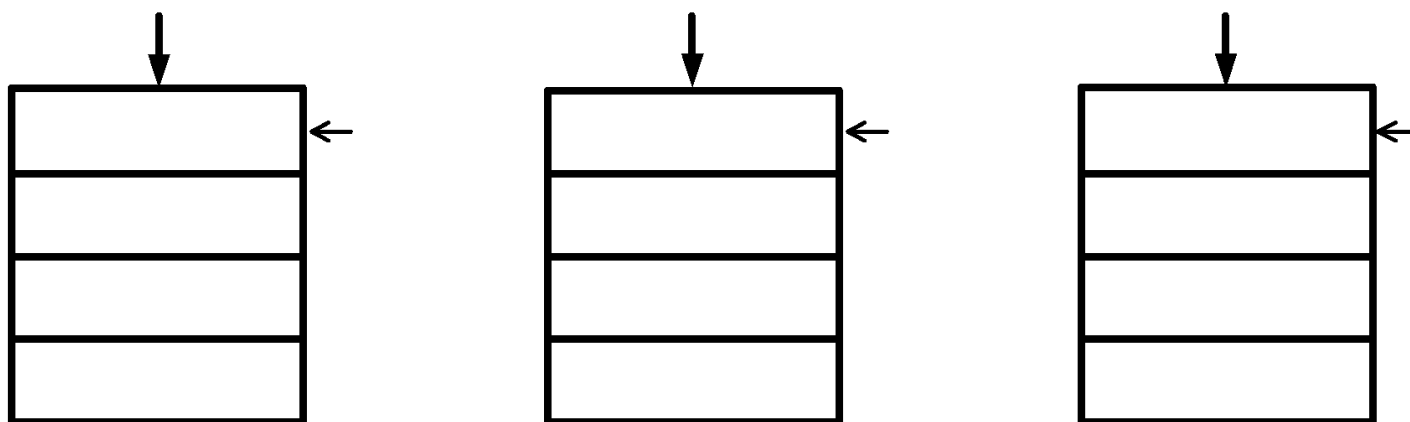


Основной принцип: последним вошел — первым вышел (**Last In First Out — LIFO**). Он реализуется с помощью специального устройства — указателя стека

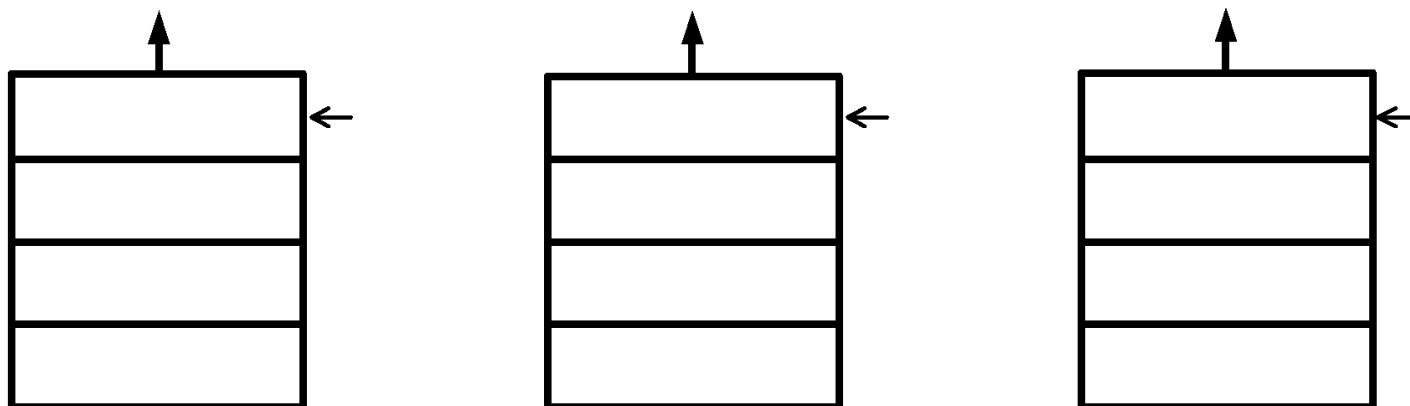
(SP). Основная функция этого устройства — формирование адресов ОЗУ при операциях записи и чтения. Здесь при записи адрес уменьшается $SP \downarrow$, при чтении адрес увеличивается $SP \uparrow$.

СТЕК при операциях записи и чтения

ЗАПИСЬ ПРОИЗВОДИТСЯ С ПОСТДЕКРЕМЕНТОМ УКАЗАТЕЛЯ СТЕКА SP

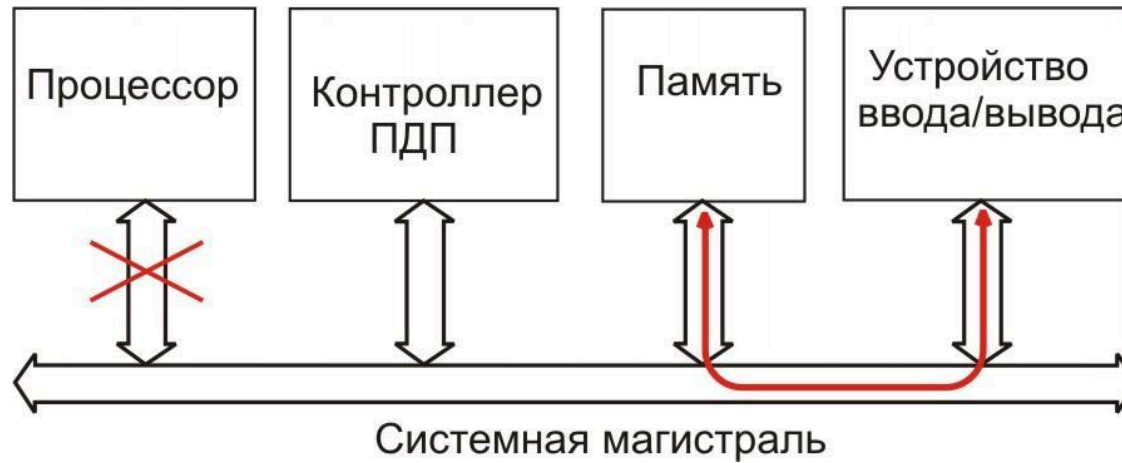
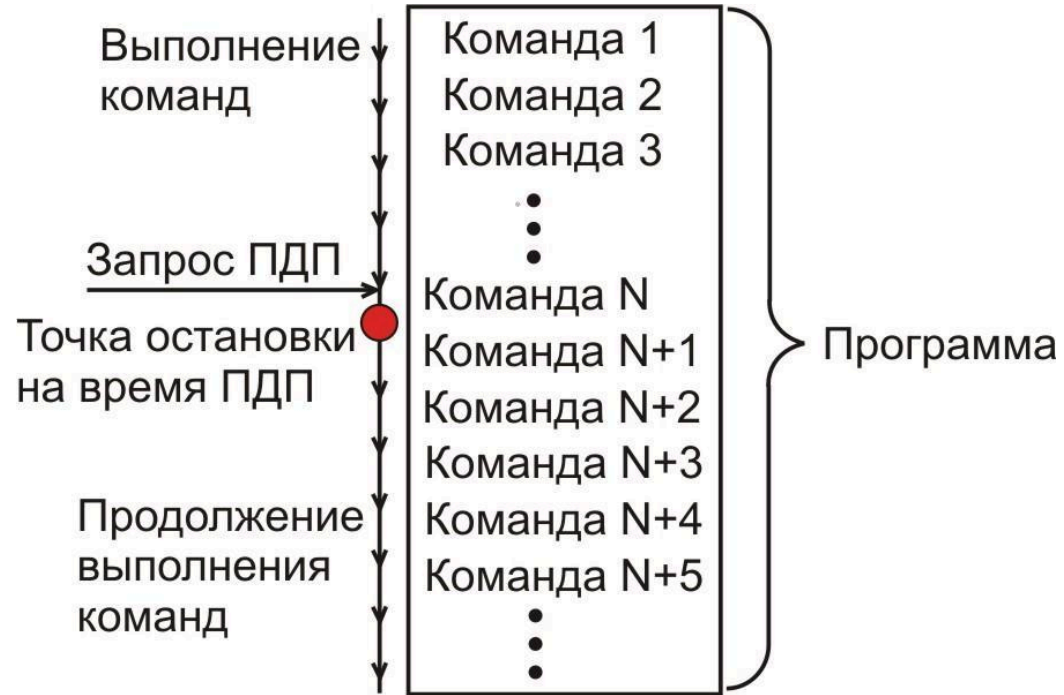


ЧТЕНИЕ ПРОИЗВОДИТСЯ С ПРЕДИНКРЕМЕНТОМ УКАЗАТЕЛЯ СТЕКА SP

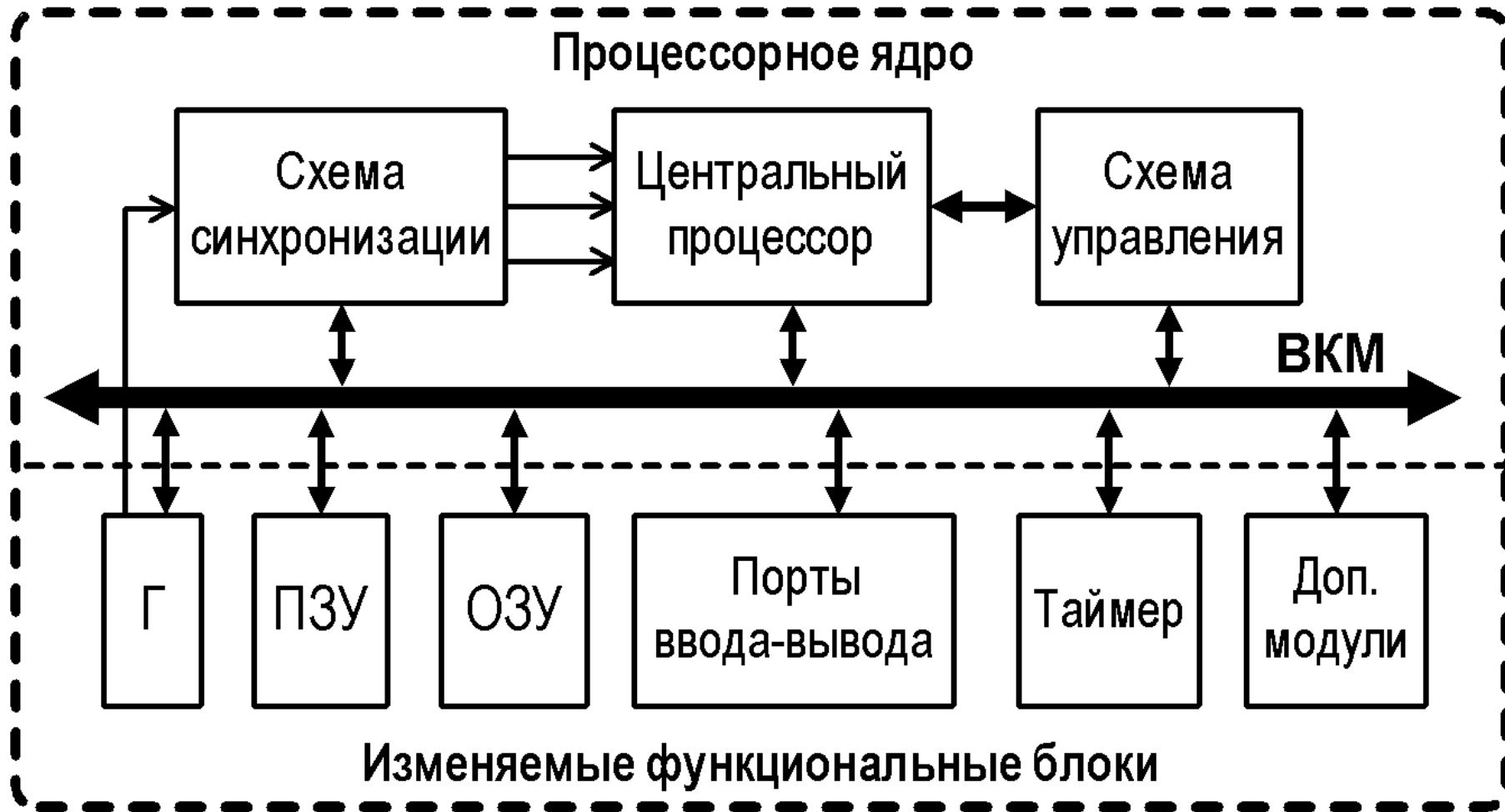


Результат работы со стеком не изменится, если запись будет происходить с *преддекрементом*, а чтение — с *постинкрементом*.

Прямой доступ к памяти – ПДП (Direct Memory Access – DMA)



СТРУКТУРНАЯ СХЕМА МИКРОКОНТРОЛЛЕРА

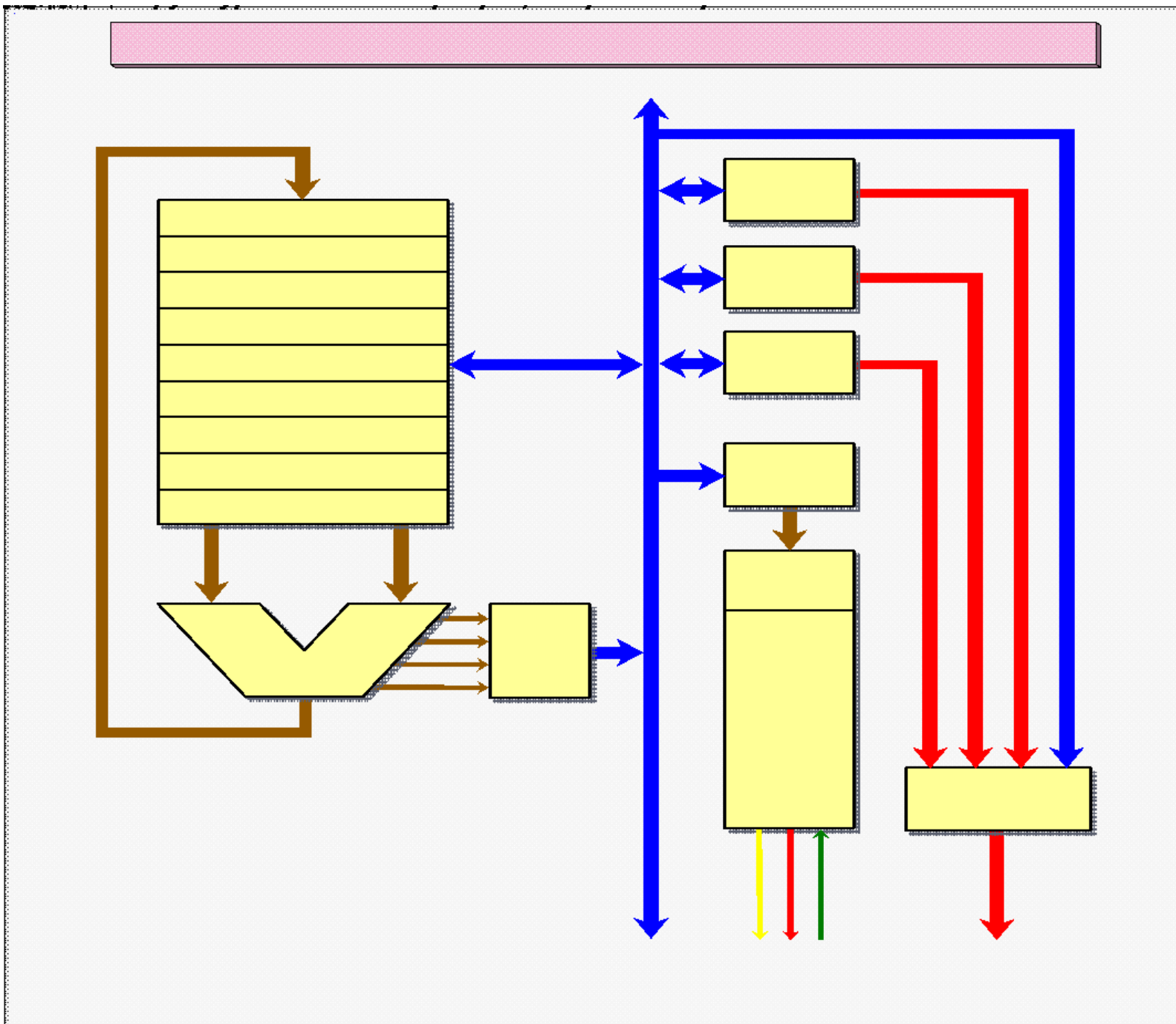


ВКМ — внутренняя контроллерная магистраль

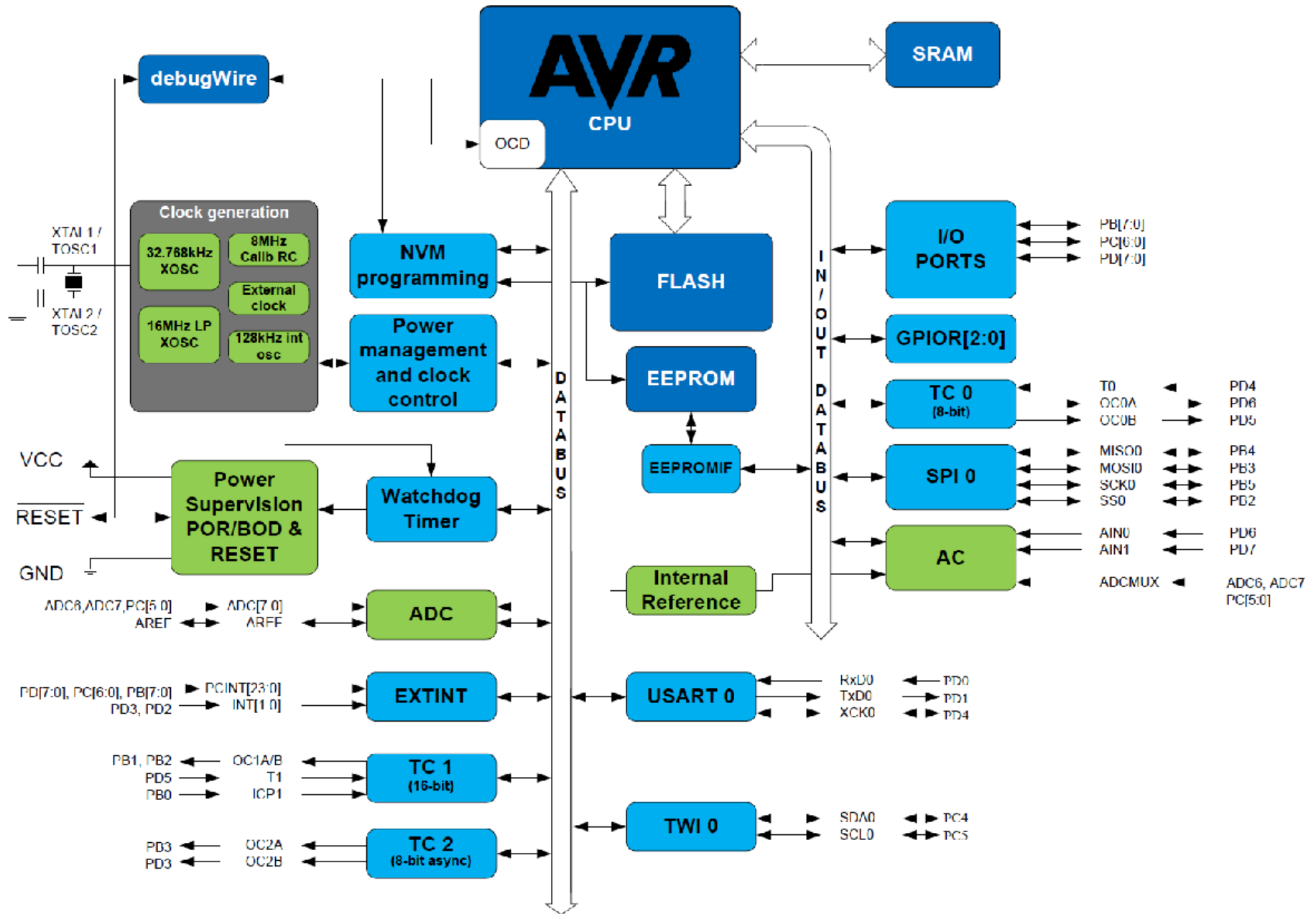
Г – тактовый генератор

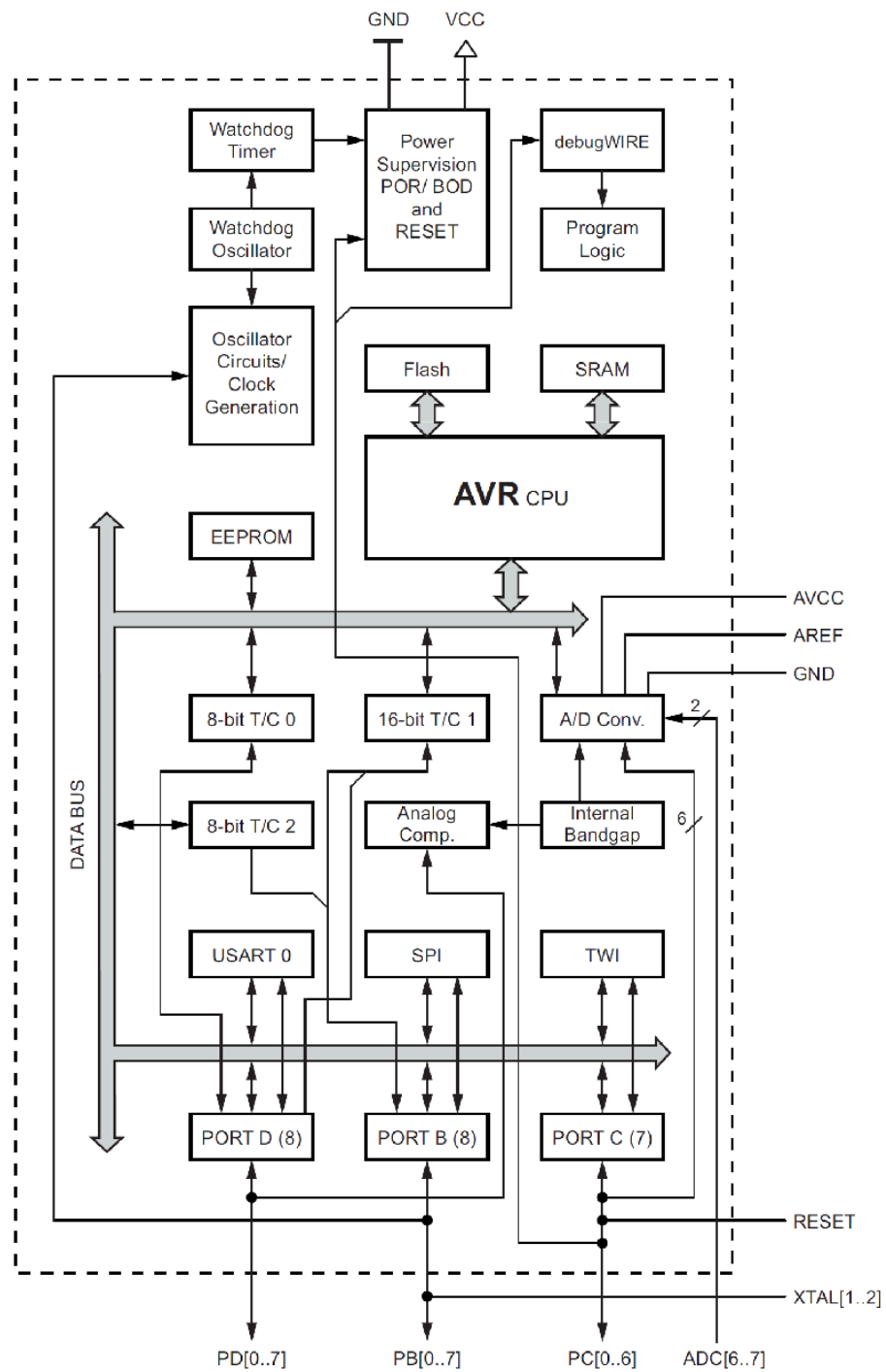
ПЗУ – постоянное запоминающее устройство (для хранения программ и констант)

ОЗУ – оперативное запоминающее устройство (для хранения переменных – изменяемых данных)



МИКРОКОНТРОЛЛЕР AVR (АТМЕГА328Р)





Конфигурация параллельных портов ввода/вывода (ATmega328p)

Состояние разряда $DDRX.n$ определяет направление передачи бита через вывод порта $PX.n$. При $DDRX.n=0$ вывод МК $PX.n$ работает в режиме входа, при $DDRX.n=1$ – в режиме выхода.

В режиме входа состояние разряда $PORTX.n$ определяет состояние вывода $PX.n$. При $PORTX.n=0$ вывод находится в высокоимпедансном состоянии (Zсостояние). При $PORTX.n=1$ к нему подключается внутренний резистор, соединенный с шиной питания.

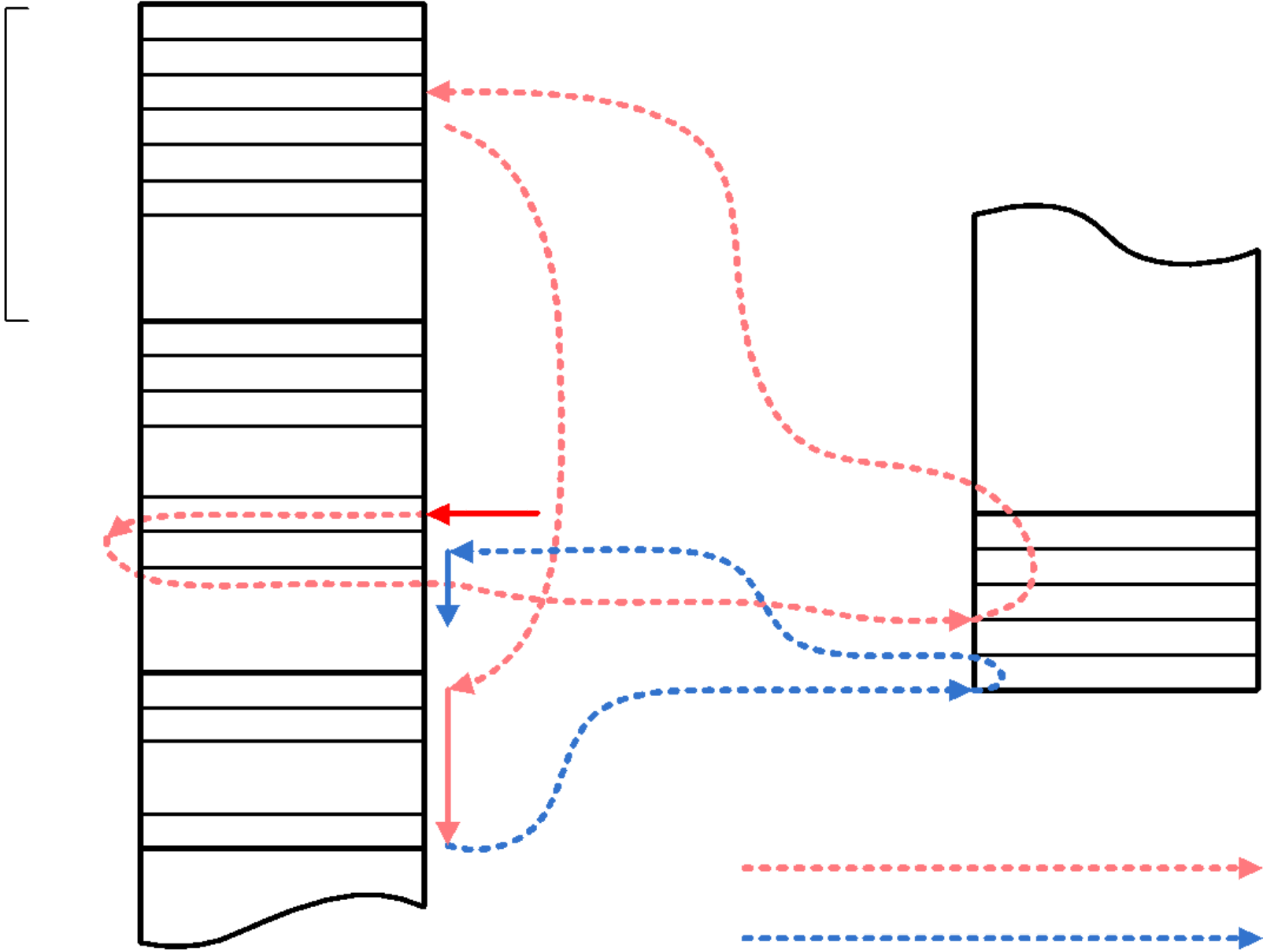
В режиме выхода разряд $PORTX.n$ определяет значение выходного сигнала на выводе $PX.n$. При $PORTX.n=0$ выходной сигнал имеет низкий уровень напряжения (U_{OL}), при $PORTX.n=1$ – высокий уровень напряжения (U_{OH}).

В таблице указано состояние вывода $PX.n$ при различных комбинациях состояний разрядов $DDRX.n$ и $PORTX.n$.

DDRX.n	PORTX.n	Вывод МК PX.n
0	0	Вход, (Z)
0	1	Вход, (Rpullup)

1	0	Выход, (U_{OL})
1	1	Выход, (U_{OH})

Обработка аппаратного прерывания и возврат из него (ATmega328p)



Битовые операции в языке C

Запись логической «1» в выбранные разряды (4, 2, 1) без изменения остальных разрядов:

```
n |= (1<<4) | (1<<2) | (1<<1);
```

Запись логического «0» в выбранные разряды (5, 3, 2) без изменения остальных разрядов:

```
n &= ~( (1<<5) | (1<<3) | (1<<2) ); или n &= ~(1<<5) & ~(1<<3) & ~(1<<2);
```

Инверсия значений выбранных разрядов (7, 4, 0) без изменения остальных разрядов:

```
n ^= (1<<7) | (1<<4) | (1<<0);
```

Ожидание появления логической 1 в заданном разряде (7) байта n :

```
while (~n & (1<<7)); while ((n & (1<<7)) == 0); while (!(n & (1<<7)));
```

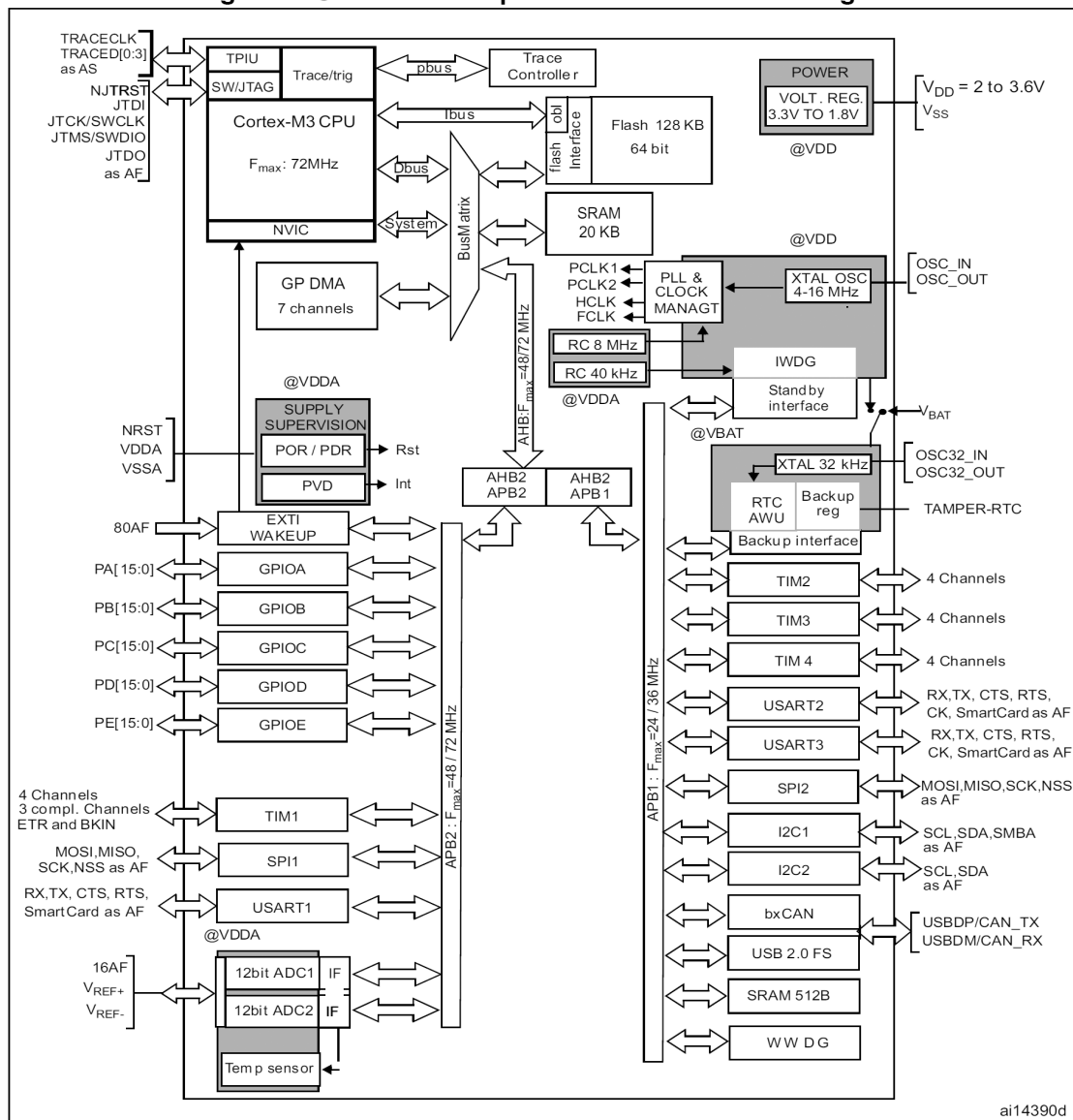
Ожидание появления логического 0 в заданном разряде (3):

```
while (n&(1<<3));
```

```
while ((n&(1<<3))!=0);
```

МИКРОКОНТРОЛЛЕР STM32F103XX

Figure 1. STM32F103xx performance line block diagram



1. T_A = -40°C to +105°C (junction temperature up to 125°C).
2. AF = alternate function on I/O port pin.

Информация по MCU STM32F103C8T6

The screenshot shows the STM32CubeMX software interface. The main window is titled "STM32CubeMX 02_Int ButtonBounced.ioc: STM32F103C8Tx". The menu bar includes "File", "Window", and "Help". The user is logged in as "Hello Marina". The breadcrumb navigation shows "Home > STM32F103C8Tx > 02_Int ButtonBounced.ioc - Pinout & Configuration". The "GENERATE CODE" button is visible. The main workspace is divided into four tabs: "Pinout & Configuration", "Clock Configuration", "Project Manager", and "Tools". The "Pinout & Configuration" tab is active, showing a "Pinout view" of the STM32F103C8Tx LQFP48 package. A tooltip for the GPIO pin is displayed, containing the following text:

GPIO:
General Purpose Input/Output

Summary:
GPIO library is used to work with physical pins on microcontroller. You can set pins to input or output, put them low (0 volts) or HIGH (3,3 volts), select pull resistors, choose output type and select clock speed.

Related documentation:

- [Datasheet](#)
- [Reference manual](#)
- [Tutorial Video](#)

[RM0008](#) – руководство по микроконтроллерам серии 10X

[UM1850](#) – Description of STM32F1 HAL and low-layer drivers. Описание функций HAL и LL библиотек.

Ввод-вывод в STM32 и некоторые функции библиотеки HAL

Вывод

```
/*РАБОТА С ФУНКЦИЕЙ HAL_GPIO_WritePin*/
HAL_GPIO_WritePin (GPIOB, GPIO_PIN_All, GPIO_PIN_SET); //установка всех
бит порта B
HAL_GPIO_WritePin (GPIOB, GPIO_PIN_All, GPIO_PIN_RESET); //сброс всех бит
порта B
HAL_GPIO_WritePin (GPIOB, GPIO_PIN_15|GPIO_PIN_13|GPIO_PIN_11,
GPIO_PIN_RESET); //сброс выбранных бит порта B – PB15, PB13, PB11
HAL_GPIO_WritePin (GPIOB, GPIO_PIN_13|GPIO_PIN_11|GPIO_PIN_9,
GPIO_PIN_SET); //установка выбранных бит порта B – PB13, PB11, PB9

/*РАБОТА С РЕГИСТРОМ ПОБИТОВОЙ УСТАНОВКИ-СБРОСА BSRR (ст. 16 бит – сброс
бит, мл. 16 бит – установка бит)*/
GPIOA->BSRR=0xCC000000; //сброс 15,14,11,10 бит порта A
GPIOA->BSRR=0x0000CC00; //установка 15,14,11,10 бит порта A
```

/*РАБОТА С РЕГИСТРОМ ПИНОВЫХ ДАННЫХ ODR (Output Data Register), запись параллельно */

```
GPIOB->ODR=0xE305; //запись 0b1110001100000101 в порт B
GPIOA->ODR=0x0007; //запись 0b0000000000000111 в порт A
```

Ввод

//ЧТЕНИЕ ЧЕРЕЗ РЕГИСТР IDR (Input Data Register)

```
GPIOB->ODR=(GPIOA->IDR&0x000F)<<8; //чт. 4 мл. бит PA и запись их в 4
мл. бита старшего байта PB
```

//ЧТЕНИЕ ЧЕРЕЗ функцию HAL_GPIO_ReadPin

```
a=HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_3)<<3 |
HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_2)<<2 |
HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_1)<<1 |
HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_0)<<0; //a <- число в 4-х мл. битах PA
a<<=8; //сдвиг a на 8 бит влево
GPIOB->ODR=a; //чтение 4 мл. бит порта A и запись их в 4 мл. бита
старшего байта порта B
```

Callback-функция для обработчика внешнего прерывания на выводе МК GPIO_PIN

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
.....//обработчик внешнего прерывания
}
```

АЦП

```
HAL_ADCEx_Calibration_Start(&hadc1); //калибровка АЦП1
HAL_ADC_Start(&hadc1); //запуск АЦП1 на преобразование
HAL_ADC_Start_IT(&hadc1); //запуск АЦП1 на преобразование с прерываниями
HAL_ADC_Start_DMA(&hadc1, (uint32_t*)&adc, 10); //старт АЦП1 в режиме DMA
с передачей в массив adc[10] 10 слов данных с АЦП
HAL_ADC_PollForConversion(&hadc1, 100); //ожидание готовности выходных
данных АЦП1
adc = HAL_ADC_GetValue(&hadc1); //чтение преобразованных данных АЦП1 в
переменную adc
HAL_ADC_Stop(&hadc1); //останов АЦП1

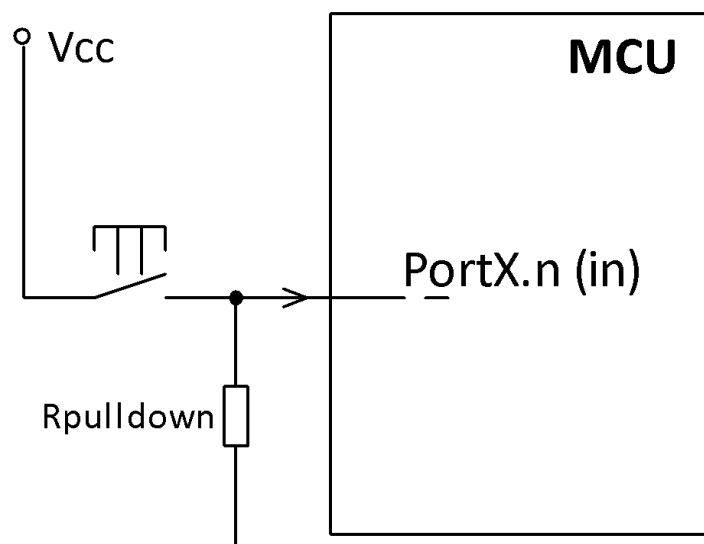
/*Callback-функция для обработчика прерывания от АЦП*/
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {.....}
```

UART

```
HAL_UART_Transmit(&huart1, (uint8_t*)uart_str, strlen(uart_str), 1000);  
//вывод сформированной строки массива uart_str[] в UART1
```

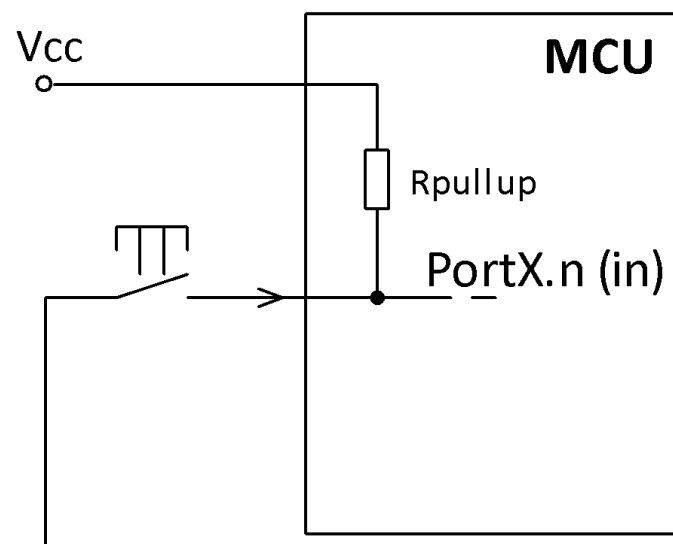
ПОДКЛЮЧЕНИЕ КНОПОК К ПОРТАМ ВВОДА

Клик кнопки формирует короткий импульс высокого уровня



Для формирования низкого уровня при отпущенной кнопке используется внешний подтягивающий к низкому уровню резистор ***Rpulldown***.

Клик кнопки формирует короткий импульс низкого уровня



Для формирования высокого уровня при отпущенной кнопке используется подтягивающий к высокому уровню резистор ***Rpullup***, в качестве которого может выступить

	и внутренний подтягивающий резистор линии порта ввода.
--	--

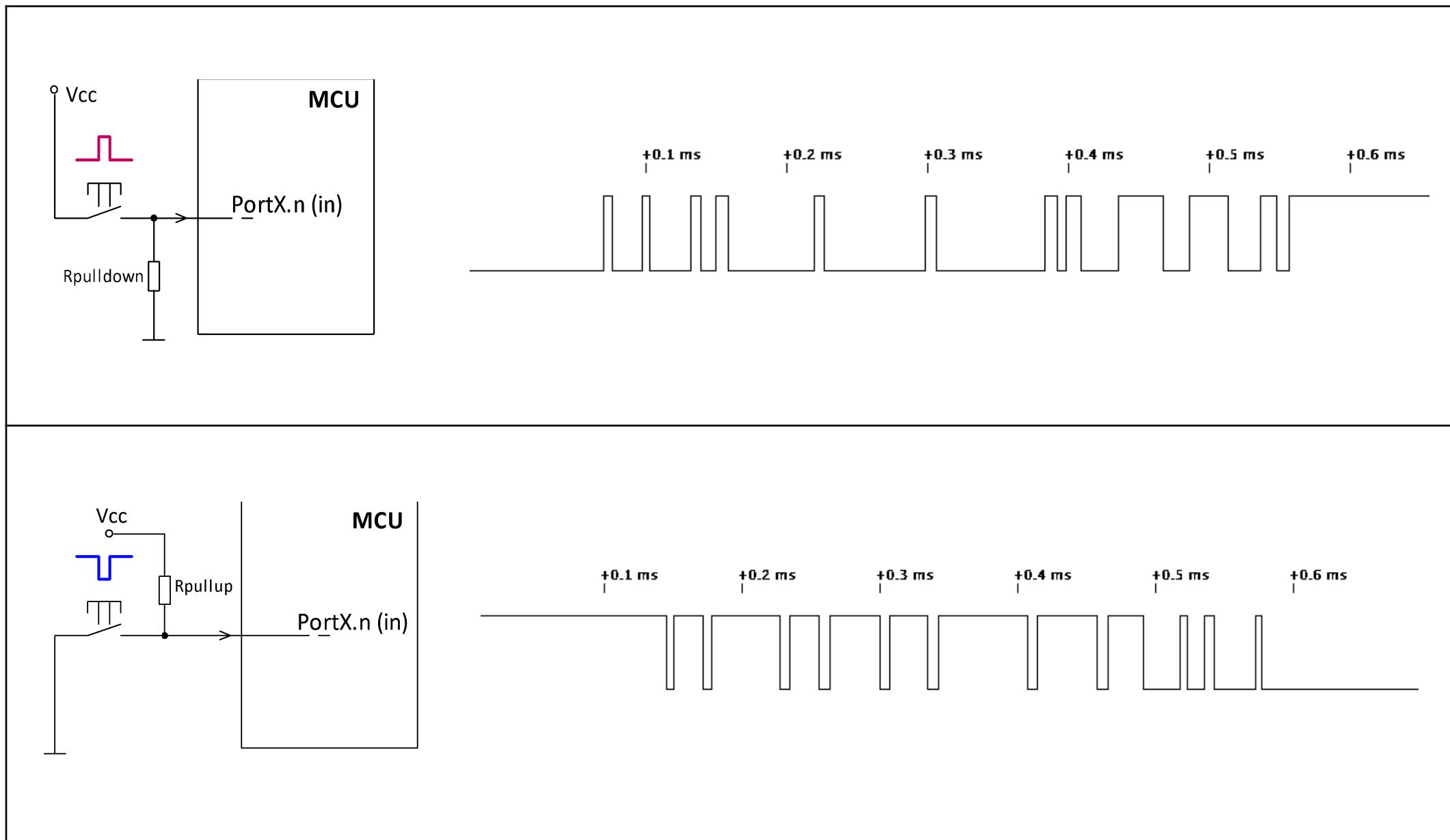
В микроконтроллерах **Atmega** (в том числе и Atmega328p) для портов ввода существует возможность подключения внутреннего подтягивающего к питанию резистора R_{pullup} (см. , справа).

В микроконтроллерах **STM32** для портов в режиме ввода существует возможность подключения внутренних подтягивающего как к + питания R_{pullup} , так и к выводу «общий» (земля) $R_{pulldown}$.

Схемотехническую реализацию различных вариантов подтягивающих резисторов см. на слайдах , .

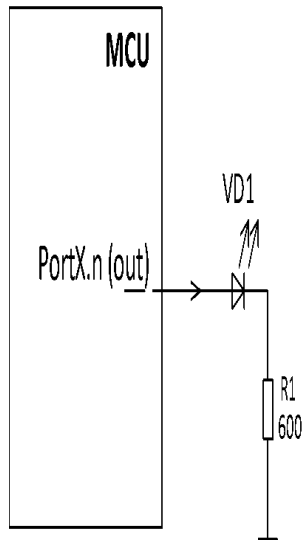
Таким образом в **МК STM32** при подключении кнопок к портам ввода всегда можно использовать внутренние подтягивающие (к земле или питанию) резисторы. В **МК Atmega** можно использовать только внутренний подтягивающий к питанию резистор.

ДРЕБЕЗГ КОНТАКТОВ ПРИ ЗАМЫКАНИИ КНОПКИ

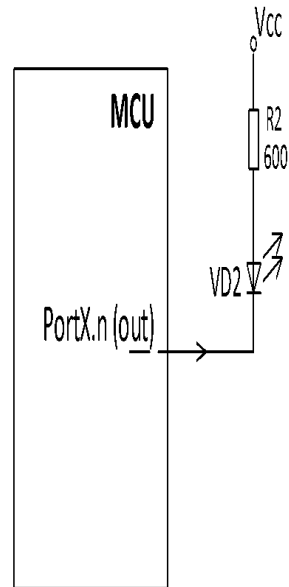


ПОДКЛЮЧЕНИЕ СВЕТОДИДОВ К ПОРТАМ ВЫВОДА

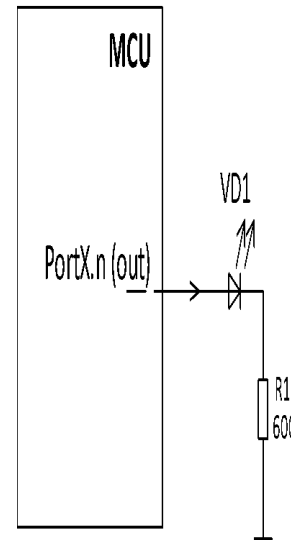
Светодиод светится
при лог. "1" на выходе



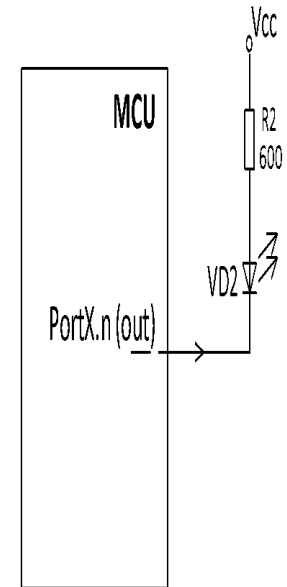
Светодиод светится
при лог. "0" на выходе



Светодиод светится
при лог. "1" на выходе



Светодиод светится
при лог. "0" на выходе



Внутри МК есть токоограничивающий резистор, однако добавочно включают внешний резистор R_1 для задания тока:

$$I_{LED} \approx \frac{V_{\log 1} - V_{LED}}{R_1}$$

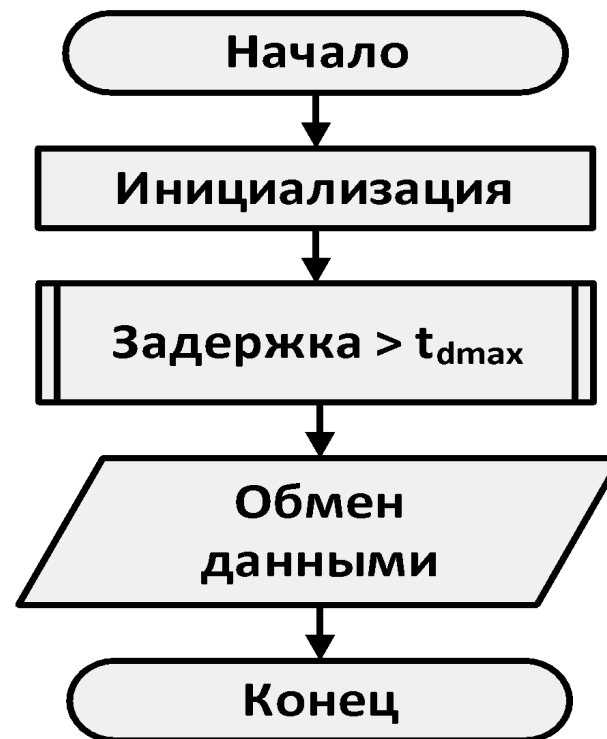
Резистор R_2 задает ток через светящийся светодиод:

$$I_{LED} \approx \frac{V_{CC} - V_{LED} - V_{\log 0}}{R_2} .$$

СПОСОБЫ ОБМЕНА ЦП С МЕДЛЕННЫМ УВВ

Синхронный способ обмена

В этом случае программа регулирует синхронный обмен с внешним устройством.



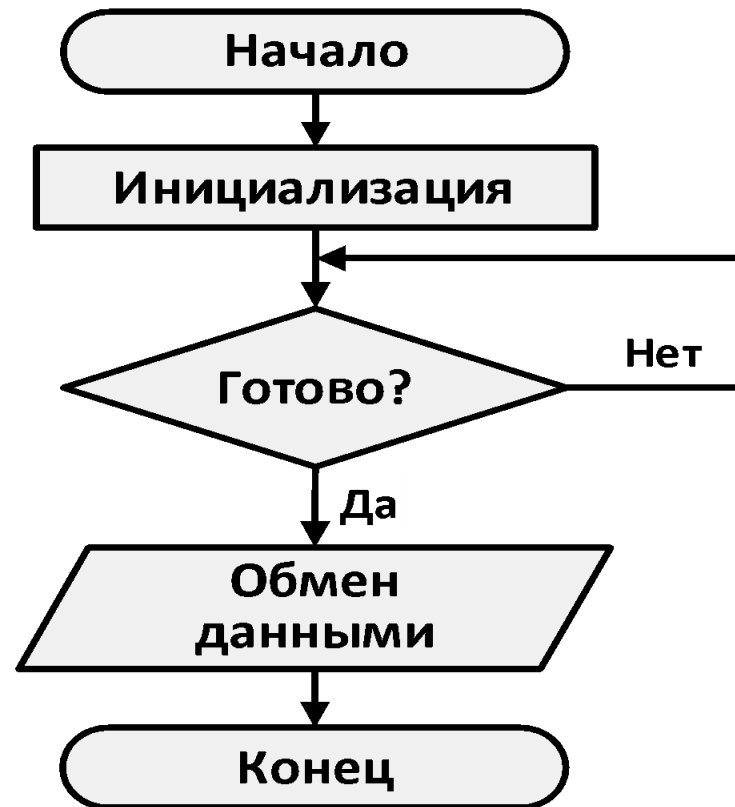
Недостатки:

- Привязка к паспортным данным периферийного устройства

- Потеря времени на создание задержек

АСИНХРОННЫЙ СПОСОБ ОБМЕНА

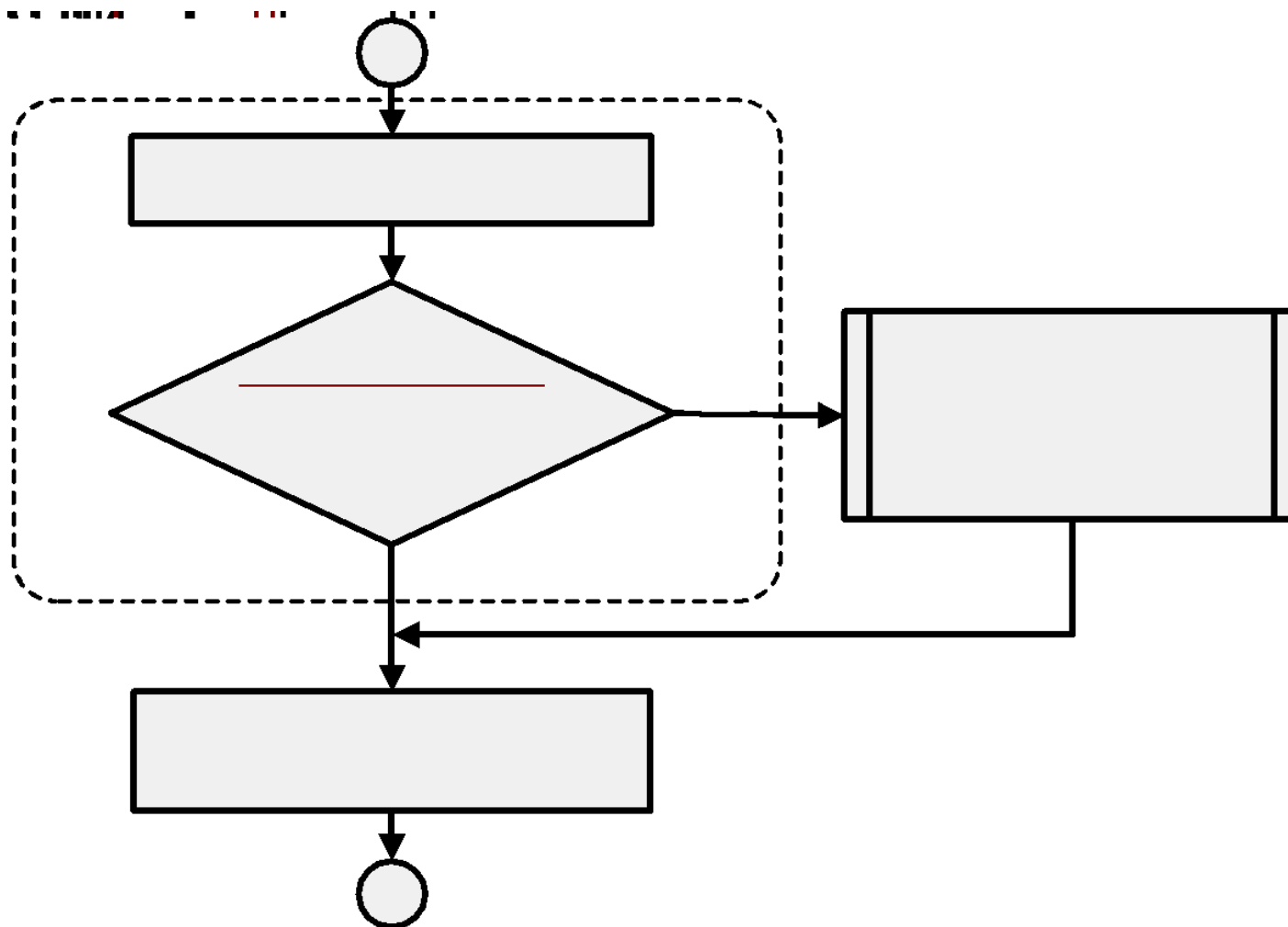
В этом случае ВУ формирует сигнал готовности, а процессор опрашивает наличие этого сигнала (программный поллинг).



Недостаток:

- В ожидании сигнала готовности процессор не может выполнять другие операции.

ОБМЕН ПО ПРЕРЫВАНИЮ



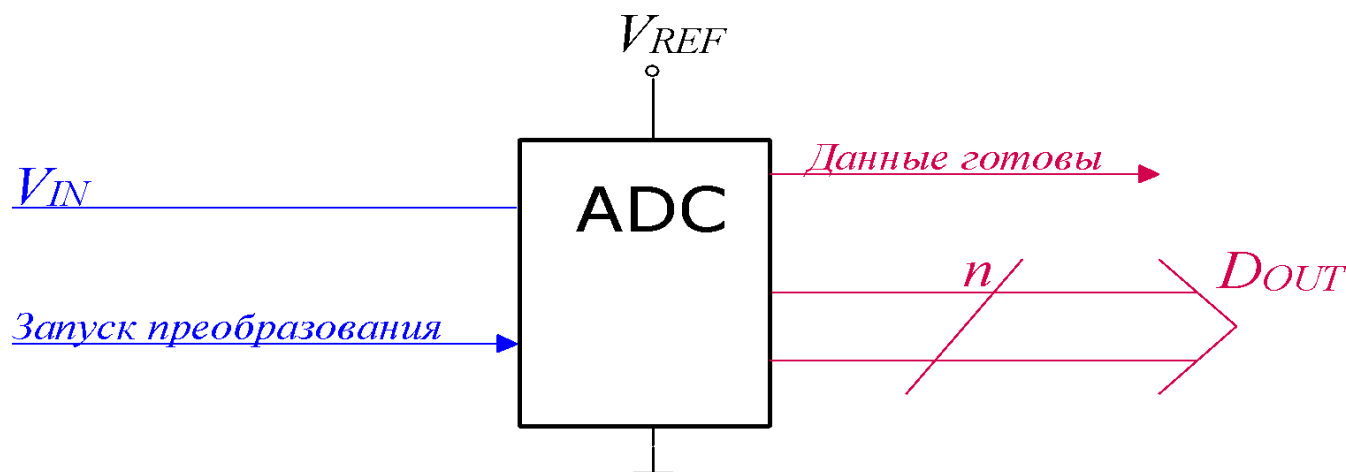
- При отсутствии запроса прерывания процессор выполняет основной алгоритм;

- При наличии запроса переходит на подпрограмму обслуживания прерывания, выполняет ее и возвращается к прерванной программе.

Достоинство: процессор не тратит время на создание необходимых временных задержек и на ожидание готовности внешнего устройства.

АНАЛОГО-ЦИФРОВОЙ ПРЕОБРАЗОВАТЕЛЬ (АЦП, ADC)

АЦП – устройство, осуществляющее автоматическое преобразование (измерение и кодирование) непрерывно изменяющихся во времени аналоговых значений (например, напряжения) в эквивалентные значения числовых кодов (как правило двоичных).



Преобразование обеспечивает соответствие дискретного отсчета $V_{IN}(t_i)$ значению выходного двоичного кода $D_{OUT\ t_i}$. Количественная связь для любого момента времени t_i , определяется соотношением:

$$D_{OUT} \approx \left\lfloor \frac{V_{IN}}{\Delta V} \right\rfloor = \left\lfloor 2^n \frac{V_{IN}}{V_{REF}} \right\rfloor$$

где n – количество выходных двоичных разрядов АЦП; V_{REF} – опорное напряжение АЦП; $\Delta V = V_{REF}/2^n$ – шаг квантования (или аналоговый эквивалент единицы младшего значащего разряда ЕМР).

АЦП часто входит в состав микроконтроллеров. Так АТmega328р имеет в своем составе 10-разрядный АЦП, а МК STM32F103C8T6 – два 12-разрядных АЦП.

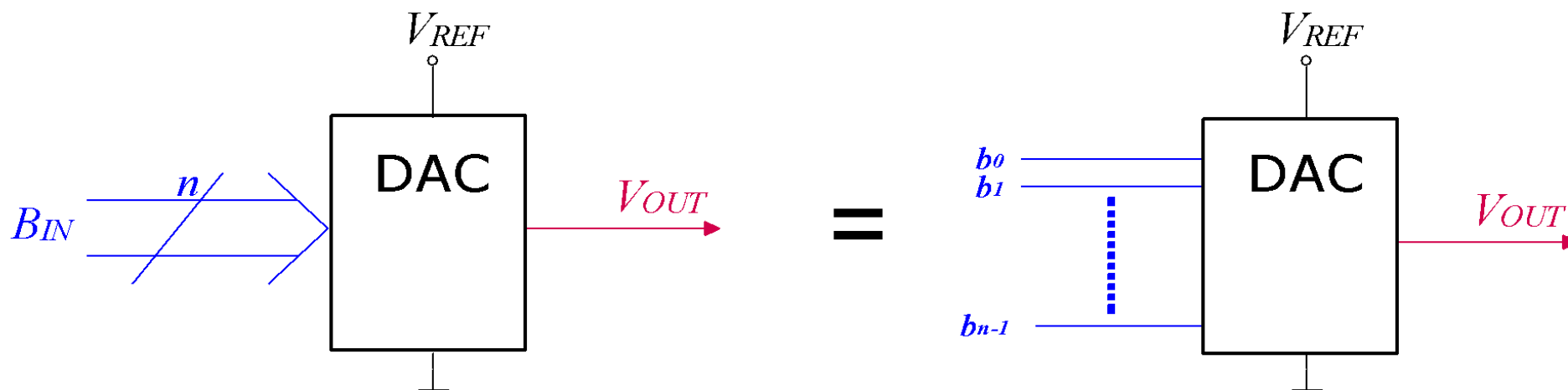
Взаимодействовать с АЦП программа микроконтроллера может двумя способами:

1. Поллинг. В этом случае после запуска АЦП периодически осуществляется опрос флага готовности, и при его появлении производится считывание результата преобразования — двоичного цифрового эквивалента (см.).

2. По прерыванию. В этом случае после запуска АЦП программа занимается другой задачей. При поступлении запроса прерывания от АЦП (данные готовы) задача прерывается и происходит переход к обработчику прерывания от АЦП, в котором считываются готовые данные (см.).

ЦИФРО-АНАЛОГОВЫЙ ПРЕОБРАЗОВАТЕЛЬ (ЦАП, DAC)

Устройство, осуществляющее автоматическое преобразование входных значений, представленных числовыми кодами, в эквивалентные им значения какой-нибудь физической величины (напряжения, тока и др.), называют цифроаналоговым преобразователем (ЦАП, DAC).



$$V_{OUT} = \Delta V \times B_{IN} = \frac{V_{REF} \cdot (b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + \dots + b_1 \cdot 2 + b_0)}{2^n},$$

где:

B_{IN} – входной n -разрядный двоичный код;

$b_0, b_1 \dots b_{n-1}$ – разряды (биты) двоичного кода, принимающие значение «0» или «1»

V_{REF} – опорное напряжение ЦАП.

ЦАП иногда входят в состав микроконтроллеров. Однако в МК ATmega328p и STM32F103C8T6 они отсутствуют.

Имеются они, например, в составе МК LGT8F328p (8-разрядный ЦАП на резистивной матрице R2R), STM32F407 (12-разрядный).

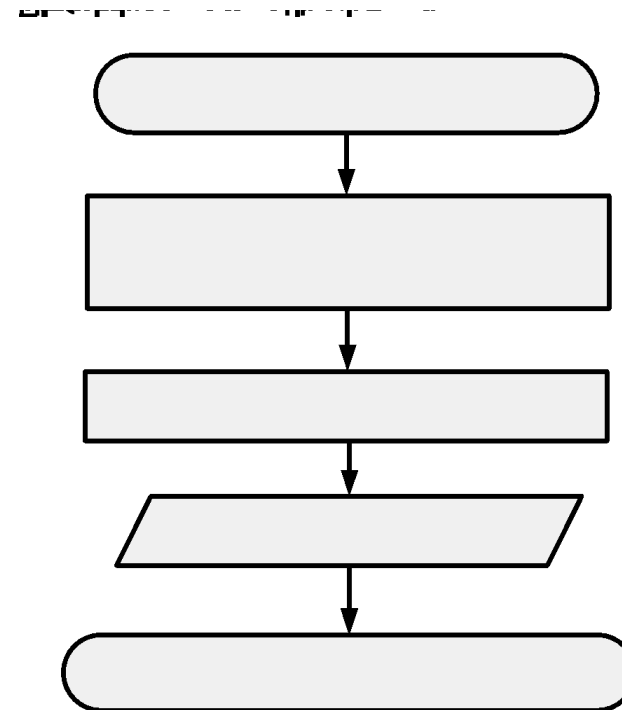
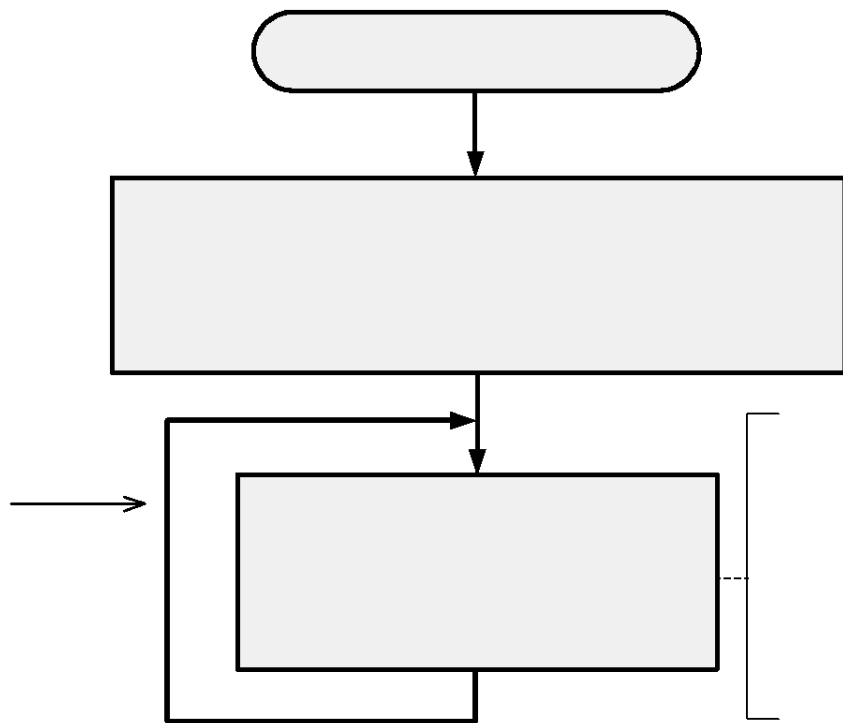
В случае необходимости подключаются внешние ЦАП, как, например, MCP4921 (12-разрядный ЦАП с SPI-интерфейсом), MCP4725 (12-разрядный ЦАП с I²Sинтерфейсом) и пр.

Кроме того можно самостоятельно собрать ЦАП невысокой разрядности (не более 8) на резистивной матрице R2R с резисторами с 1% допуском номиналами, например 5кОм-10кОм или 10кОм-20кОм. Такой ЦАП, являясь параллельным и по принципу действия и по интерфейсу связи обладает максимально возможным быстродействием. Принцип действия такого ЦАП рассмотрен [здесь](#) на стр. 33-35. Такой принцип действия имеет 8-разрядный ЦАП, встроенный в МК LGT8F328p. Ниже будут рассмотрены проекты цифрового синтеза периодических сигналов с использованием подобного ЦАП.

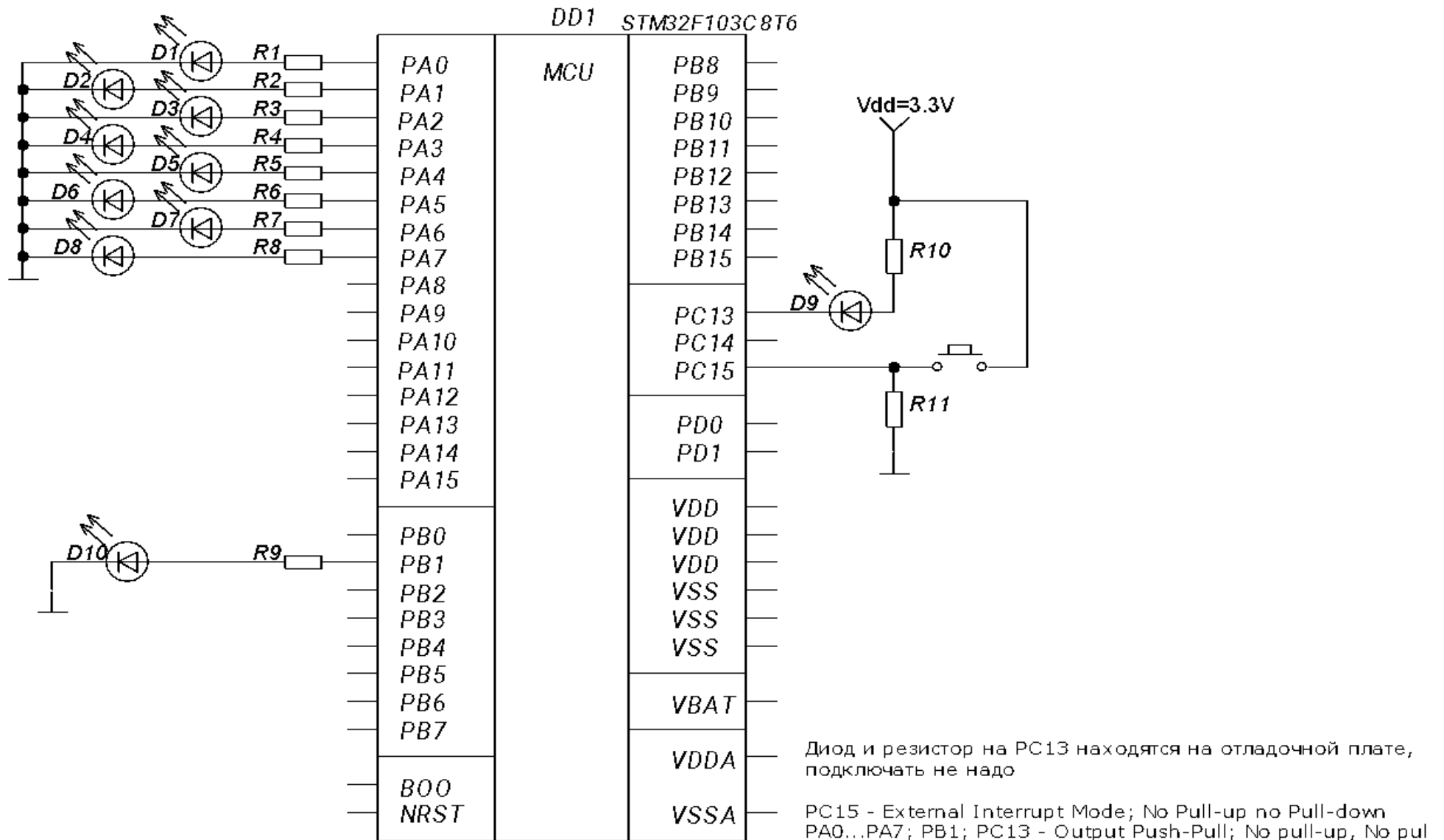
УЧЕБНЫЕ ПРОЕКТЫ ДЛЯ STM32F103C8T6, ATMEGA328P

Задача 1. Мигалка с подсчетом кликов кнопки с дребезгом

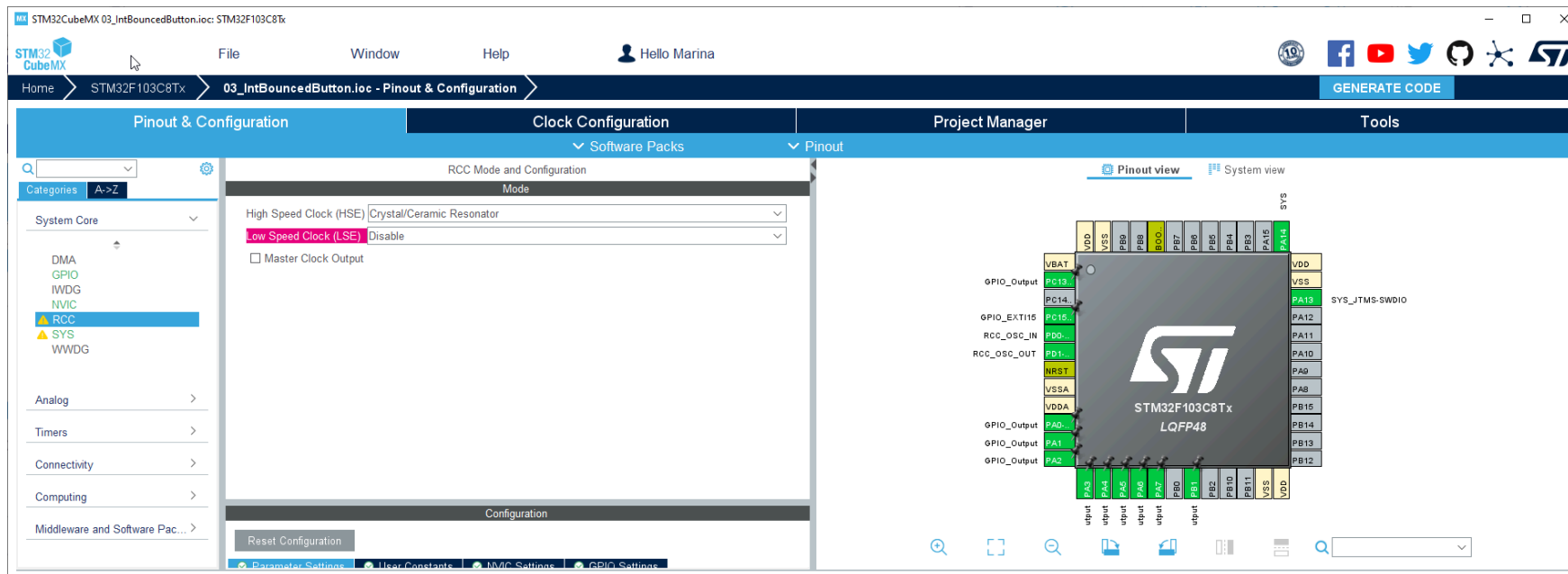
Организовать перемигивание светодиодов на выводах PC13 (светодиод на отладочной плате, включенный по схеме с общим анодом, см. , справа) и PB1 (включен по схеме с общим катодом, см. , слева) с периодом 0.5 с. Одновременно должен работать счетчик внешних событий от 0 до 255 (кликов кнопки на порте ввода PC16) и происходить вывод кликов кнопки в двоичном коде на светодиодную линейку, подключенную к младшему байту порта A (PA0...PA7). Светодиодная линейка включена по схеме с общим катодом.



Принципиальная схема проекта

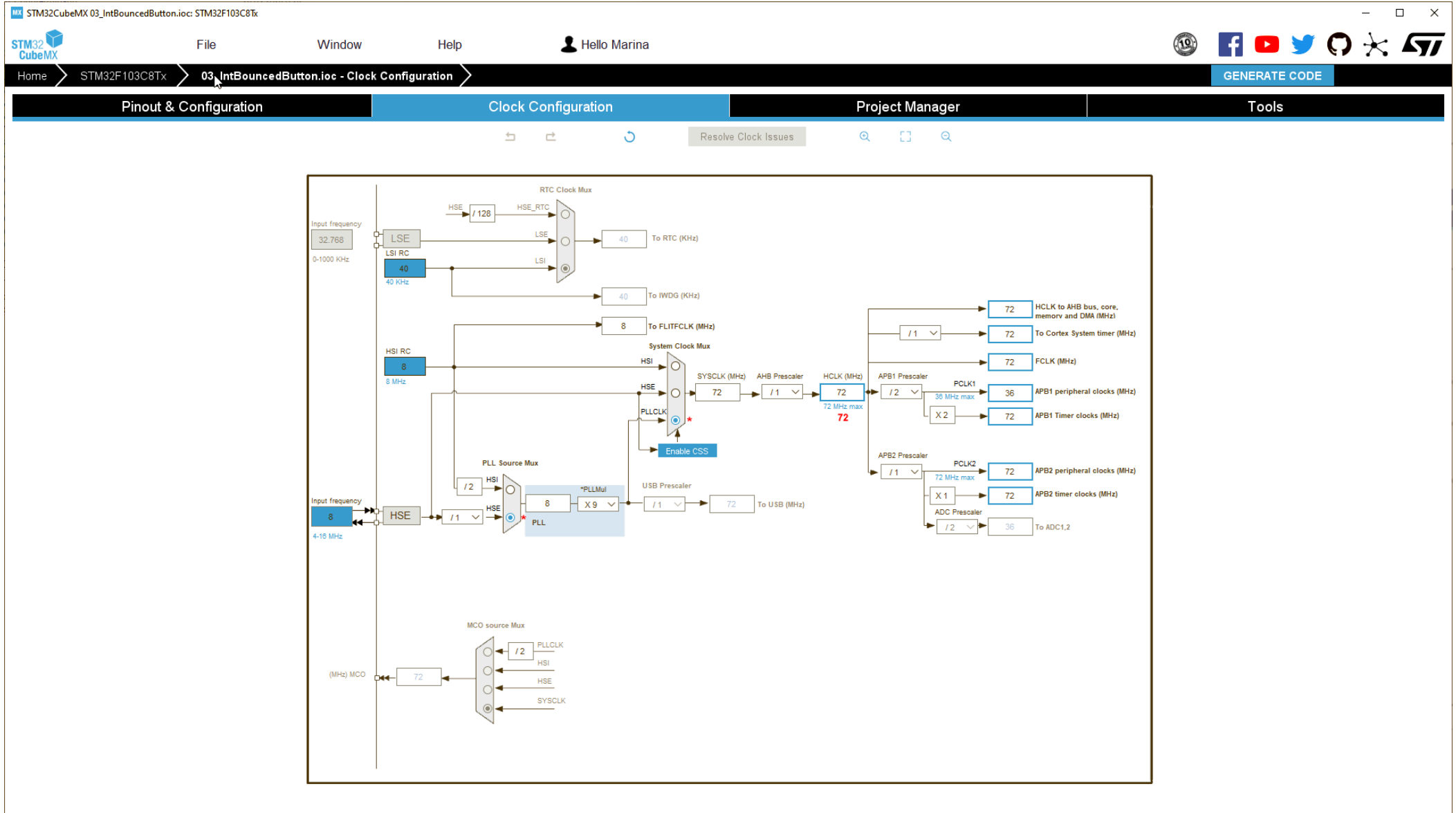


Настройка проекта в Cube MX



The screenshot displays the STM32CubeMX software interface. The main window is titled "Pinout & Configuration" for the project "03_IntBouncedButton.ioc". The left sidebar shows a tree view of system peripherals, with "SYS" selected. The central panel shows the configuration for the "SYS Mode and Configuration" peripheral, with "System Wake-Up" selected in the "Mode" dropdown and "SysTick" in the "Timebase Source" dropdown. A warning message at the bottom of this panel reads: "Warning: This peripheral has no parameters to be configured." The right side of the interface shows a "Pinout view" of the STM32F103C8Tx LQFP48 package. The pins are color-coded and labeled with their functions: VBAT, VSS, VDD, PA14, PA13, PA12, PA11, PA10, PA9, PA8, PA7, PA6, PA5, PA4, PA3, PA2, PA1, PA0, PB15, PB14, PB13, PB12, PB11, PB10, PB9, PB8, PB7, PB6, PB5, PB4, PB3, PB2, PB1, VSS, VDD, VSSA, VDDA, NRST, PD0, PD1, PC15, PC14, PC13, and VBAT. The functions are assigned as follows: PA14 (SYS), PA13 (SYS_JTMS-SWDIO), PA12 (GPIO_Output), PA11 (GPIO_Output), PA10 (GPIO_Output), PA9 (GPIO_Output), PA8 (GPIO_Output), PA7 (GPIO_Output), PA6 (GPIO_Output), PA5 (GPIO_Output), PA4 (GPIO_Output), PA3 (GPIO_Output), PA2 (GPIO_Output), PA1 (GPIO_Output), PA0 (GPIO_Output), PB15 (GPIO_Output), PB14 (GPIO_Output), PB13 (GPIO_Output), PB12 (GPIO_Output), PB11 (GPIO_Output), PB10 (GPIO_Output), PB9 (GPIO_Output), PB8 (GPIO_Output), PB7 (GPIO_Output), PB6 (GPIO_Output), PB5 (GPIO_Output), PB4 (GPIO_Output), PB3 (GPIO_Output), PB2 (GPIO_Output), PB1 (GPIO_Output), VSS (GPIO_Output), VDD (GPIO_Output), VSSA (GPIO_Output), VDDA (GPIO_Output), NRST (GPIO_Output), PD0 (GPIO_Output), PD1 (GPIO_Output), PC15 (GPIO_Output), PC14 (GPIO_Output), PC13 (GPIO_Output), and VBAT (GPIO_Output).

Настройка внутренней конфигурации МК



Настройка линий ввода-вывода

The screenshot displays the STM32CubeMX interface for configuring the STM32F103C8Tx microcontroller. The main window is titled "03_IntBouncedButton.ioc - Pinout & Configuration".

GPIO Mode and Configuration:

- Software Packs: GPIO, RCC, SYS, NVIC
- Search Signals: Search (Ctrl+F)
- Show only Modified Pins:

Pin Name	Signal on Pin	GPIO output	GPIO mode	GPIO Pull-up	Maximum ou...	User Label	Modified
PA0-WKUP	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA1	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA2	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA3	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA4	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA5	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA6	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA7	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PB1	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PC13-TAMP...	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PC15-OSC3...	n/a	n/a	External Inter...	No pull-up an...	n/a		<input type="checkbox"/>

PA5 Configuration:

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

Pinout Diagram:

The diagram shows the STM32F103C8Tx LQFP48 package with various pins labeled. A context menu is open over pin PA5, listing the following options:

- PA5
- Reset_State
- ADC1_IN5
- ADC2_IN5
- SPI1_SCK
- GPIO_Input
- GPIO_Output
- GPIO_Analog
- EVENTOUT
- GPIO_EXTI5

Настройка внешних прерываний (1)

The screenshot displays the STM32CubeMX software interface for configuring external interrupts. The main window is titled "03_IntBouncedButton.ioc - Pinout & Configuration".

GPIO Mode and Configuration Table:

Pin Name	Signal on Pin	GPIO output	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PA0-WKUP	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA1	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA2	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA3	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA4	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA5	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA6	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PA7	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PB1	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PC13-TAMP...	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PC15-OSC3...	n/a	n/a	External Inter...	No pull-up an...	n/a		<input type="checkbox"/>

PC15-OSC32_OUT Configuration:

- GPIO mode: External Interrupt Mode with Rising edge trigger detection
- GPIO Pull-up/Pull-down: No pull-up and no pull-down
- User Label: (empty)

Pinout Diagram: The diagram shows the STM32F103C8Tx microcontroller with pins PA0 through PA15 and PB0 through PB15. Pin PA15 is highlighted in green, indicating its configuration. A tooltip for PC15-OSC32_OUT shows its functions: Reset_State, ADC1_EXTI15, ADC2_EXTI15, RCC_OSC32_OUT, GPIO_Input, GPIO_Output, GPIO_Analog, and EVENTOUT. The pin is labeled as "GPIO_EXTI15" in the diagram.

Настройка внешних прерываний (2)

STM32CubeMX 03_IntBouncedButton.ioc: STM32F103C8Tx

File Window Help Hello Marina

Home STM32F103C8Tx 03_IntBouncedButton.ioc - Pinout & Configuration GENERATE CODE

Pinout & Configuration Clock Configuration Project Manager Tools

NVIC Mode and Configuration

Configuration

Priority Group: 4 bits for pre-em... Sort by Preemption Priority and Sub Priority Sort by interrupts names

Search: Search (...) Show available interrupts Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	15	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
EXTI line[15:10] interrupts	<input checked="" type="checkbox"/>	0	0

Enabled Preemption Priority Sub Priority

Pinout view System view

STM32F103C8Tx LQFP48

Настройки программного проекта и генерация кода

The screenshot displays the STM32CubeMX Project Manager interface. The window title is "STM32CubeMX 03_IntBouncedButton.ioc: STM32F103C8Tx". The menu bar includes "File", "Window", and "Help", along with a user profile "Hello Marina". The breadcrumb navigation shows "Home > STM32F103C8Tx > 03_IntBouncedButton.ioc - Project Manager". A "GENERATE CODE" button is visible in the top right corner.

The interface is divided into four main sections on the left: "Project", "Code Generator", "Advanced Settings", and "Linker Settings". The "Project Manager" tab is active, showing the following settings:

- Project Settings:**
 - Project Name: 03_IntBouncedButton
 - Project Location: D:\KEIL [Browse]
 - Application Structure: Advanced Do not generate the main()
 - Toolchain Folder Location: D:\KEIL\03_IntBouncedButton\
 - Toolchain / IDE: MDK-ARM Generate Under Root
 - Min Version: V5.32
- Linker Settings:**
 - Minimum Heap Size: 0x200
 - Minimum Stack Size: 0x400
- Thread-safe Settings:**
 - Cortex-M3NS: Enable multi-threaded support
 - Thread-safe Locking Strategy: Default - Mapping suitable strategy depending on RTOS selection.
- Mcu and Firmware Package:**
 - Mcu Reference: STM32F103C8Tx
 - Firmware Package Name and Version: STM32Cube_FW_F1 V1.8.5 Use latest available version
 - Use Default Firmware Location
 - Firmware Relative Path: C:/Users/ameli/STM32Cube/Repository/STM32Cube_FW_F1_V1.8.5 [Browse]

The screenshot shows the STM32CubeMX Project Manager interface for a project named '03_IntBouncedButton.ioc'. The interface is divided into several sections:

- Project:** Contains options for handling STM32Cube MCU packages and embedded software packs:
 - Copy all used libraries into the project folder
 - Copy only the necessary library files
 - Add necessary library files as reference in the toolchain project configuration file
- Code Generator:** Contains options for generated files:
 - Generate peripheral initialization as a pair of '.c/.h' files per peripheral
 - Backup previously generated files when re-generating
 - Keep User Code when re-generating
 - Delete previously generated files when not re-generated
- Advanced Settings:** Contains a search bar for needed files for the project, highlighted with a red box.
- HAL Settings:** Contains options for HAL settings:
 - Set all free pins as analog (to optimize the power consumption)
 - Enable Full Assert
- User Actions:** Contains two text input fields for user actions:
 - Before Code Generation: [Text Field] [Browse]
 - After Code Generation: [Text Field] [Browse]
- Template Settings:** Contains a text input field for selecting a template to generate customized code, with a [Settings...] button.

The screenshot displays the STM32CubeMX Project Manager interface. The main window is titled "STM32CubeMX 03_IntBouncedButton.ioc: STM32F103C8Tx". The interface includes a menu bar (File, Window, Help) and a user profile "Hello Marina". The breadcrumb navigation shows "Home > STM32F103C8Tx > 03_IntBouncedButton.ioc - Project Manager". A "GENERATE CODE" button is visible in the top right.

The main workspace is divided into several sections:

- Pinout & Configuration**: Contains a "Driver Selector" with a search bar and a list of drivers:

RCC	HAL
GPIO	HAL
- Code Generator**: Contains a "Generated Function Calls" table:

Generate Code	Rank	Function Name	Peripheral Instance Name	<input type="checkbox"/> Do Not Generate Function Call	<input type="checkbox"/> Visibility (Static)
<input checked="" type="checkbox"/>	1	SystemClock_Config	RCC	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	2	MX_GPIO_Init	GPIO	<input type="checkbox"/>	<input checked="" type="checkbox"/>
- Advanced Settings**: A section for configuring project settings.
- Tools**: Contains a "Register CallBack" list:

ADC	DISABLE
CAN	DISABLE
CEC	DISABLE
DAC	DISABLE
ETH	DISABLE
HCD	DISABLE
I2C	DISABLE
I2S	DISABLE
MMC	DISABLE
NAND	DISABLE
PCCARD	DISABLE
PCD	DISABLE
RTC	DISABLE
SD	DISABLE
SMARTCARD	DISABLE
IRDA	DISABLE
SRAM	DISABLE
SPI	DISABLE
TIM	DISABLE
UART	DISABLE
USART	DISABLE
WWDG	DISABLE

A "Code Generation" dialog box is open in the center, displaying the following information:

Code Generation

The Code is successfully generated under :
 D:/KEIL/03_IntBouncedButton
 Project language : C

Buttons: Open Folder, Open Project, Close

Программный проект в Keil μ vision

The screenshot shows the Keil uVision IDE interface. The main window displays a C source file named `main.c` with the following content:

```

1  /* USER CODE BEGIN Header */
2  /**
3   * *****
4   * @file      : main.c
5   * @brief     : Main program body
6   * *****
7   * @attention
8   *
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

```

An "Options for Target '03_IntBouncedButton'" dialog box is open, showing the "Code Generation" tab. The "ARM Compiler" dropdown menu is set to "Use default compiler version 6". Other settings include:

- Device: STMicroelectronics STM32F103C8
- Xtal (MHz): <undefined>
- Operating system: None
- System Viewer File: STM32F103xx.svd
- Use Custom File:
- Use MicroLIB: Use Cross-Module Optimization:
- Use Big Endian:

Memory areas configuration:

Read/Only Memory Areas					Read/Write Memory Areas				
default	off-chip	Start	Size	Startup	default	off-chip	Start	Size	NoInit
<input type="checkbox"/>	ROM1:			<input type="radio"/>	<input type="checkbox"/>	RAM1:			<input type="checkbox"/>
<input type="checkbox"/>	ROM2:			<input type="radio"/>	<input type="checkbox"/>	RAM2:			<input type="checkbox"/>
<input type="checkbox"/>	ROM3:			<input type="radio"/>	<input type="checkbox"/>	RAM3:			<input type="checkbox"/>
<input type="checkbox"/>	on-chip			<input type="radio"/>	<input type="checkbox"/>	on-chip			<input type="checkbox"/>
<input checked="" type="checkbox"/>	IRAM1:	0x8000000	0x10000	<input type="radio"/>	<input checked="" type="checkbox"/>	IRAM1:	0x20000000	0x5000	<input type="checkbox"/>
<input type="checkbox"/>	IRAM2:			<input type="radio"/>	<input type="checkbox"/>	IRAM2:			<input type="checkbox"/>

The dialog box also features "OK", "Cancel", "Defaults", and "Help" buttons at the bottom.

found in the LICENSE file
provided AS-IS.

-----*/

-----*/

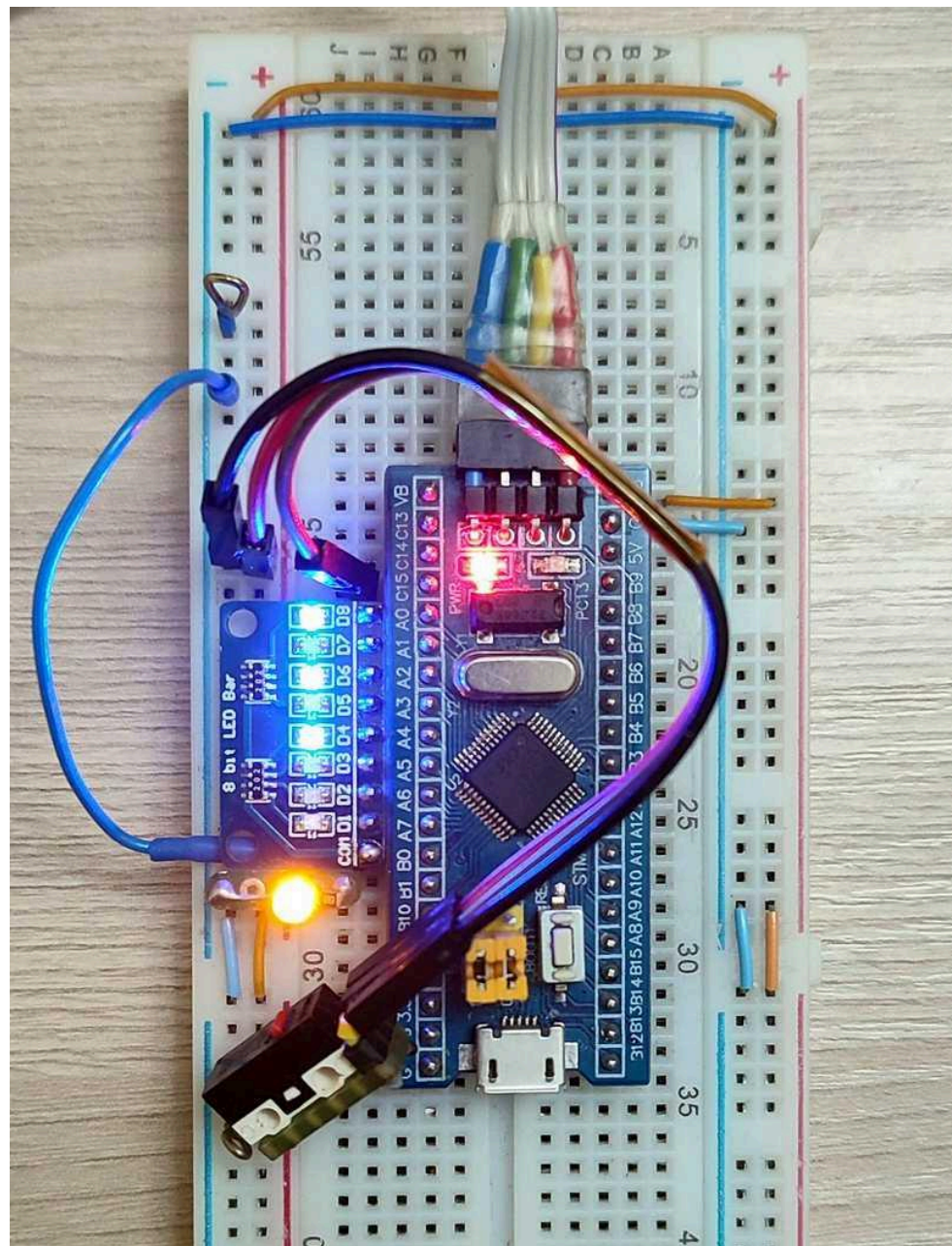
Фрагменты файла *main.c*

```

/* USER CODE BEGIN PV */
unsigned char counter=0;          //переменная для счета событий
/* USER CODE END PV */
//.....
/* Private user code -----*/
/* USER CODE BEGIN 0 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) //вызывается в обработчике прерывания
                                                //EXTI15_10_IRQHandler(void)
{
    if(GPIO_Pin==GPIO_PIN_15)                //если это используемое прерывание на линии PC15
        for (uint32_t i=0; i<=72000; i++) __NOP(); //делаем антидребезговую задержку
                                                //примерно 2 мс
    if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_15)==1) //если это не помеха-иголка,
                                                //выполняем необходимые действия
        {counter++; GPIOA->ODR=counter}; //инкремент счётчика событий и
                                                //его вывод на LED-линейку
    EXTI->PR|=1<<15; //Сброс флага прерывания, который из-за дребезга установится
                    //повторно и прерывание возникнет вновь
}
/* USER CODE END 0 */
int main(void)
{
//.....
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1) //перемигивание светодиодами PC13, PB1 (фоновая программа)
{
    HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13); //Изменение состояния PC13 на противополож.
    HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_1); //Изменение состояния PB1 на противополож.
    HAL_Delay(250); //Задержка 250 мс
/* USER CODE END WHILE */
}
}

```

Макет программно-аппаратного проекта



ТА ЖЕ ЗАДАЧА (СЧЕТЧИК КЛИКОВ С МИГАЛКОЙ) В СРЕДЕ ARDUINO ДЛЯ ПЛАТЫ ARDUINO NANO

```
volatile unsigned char count=0;
void setup() {
    // put your setup code here, to run once:
    pinMode(2, INPUT);          //I02 (INT0) - режим ввода с высокоимпедансным входом
    pinMode(13, OUTPUT);        //I013 (PB5) в режиме вывода, к нему подключен светодиод на плате
    digitalWrite(13,LOW);       //гашение светодиода на плате
    pinMode(7, OUTPUT);         //I07 (PD7) в режиме вывода, к нему подключен внешний светодиод
    digitalWrite(13,HIGH);      //включение светодиода на PD.7, чтобы LEDs работали в противофазе

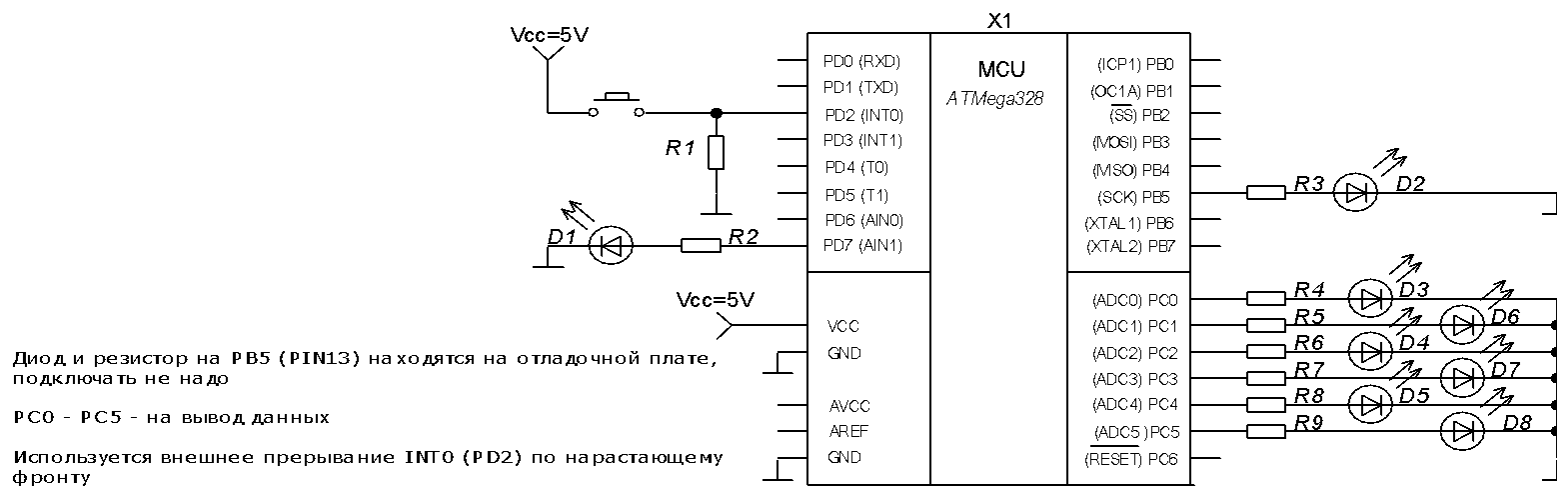
    DDRC=0b00111111;           //6 младших разрядов порта C - на вывод данных, на них LED-линейка
    attachInterrupt(digitalPinToInterrupt(2), MyInterrupt_INT0,RISING); //Подключение прерывания INT0 (I02) -
    срабатывание по нарастающему фронту
}

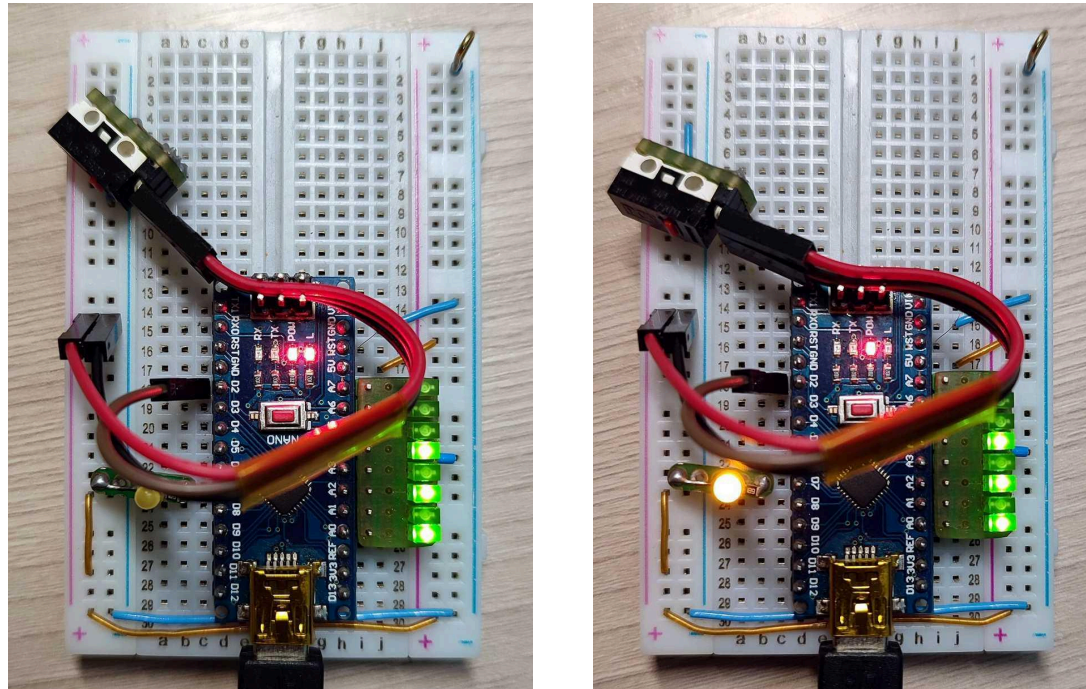
void loop() {
    // put your main code here, to run repeatedly:
    PORTB^=(0b00100000);        //изменение состояния светодиода на I013=PB5 на противоположное
    PORTD^=(0b10000000);        //изменение состояния светодиода на I07=PD7 на противоположное
    delay(250);
}

void MyInterrupt_INT0 () {
    for(unsigned int i=0; i<=32000; i++) {asm volatile ("NOP\n");}; //антидребезговая задержка на 2 мс, в прерывании
    функция delay() не работает
    if (digitalRead(2)==HIGH) { //если это не помеха-иголка, а нажатие кнопки - выполняем действие:
        count++;                //инкрементируем счетчик
        if (count>63) count=0;   //если счетчик вышел за пределы диапазона индикации, обнуляем его
        PORTC=count;            //выводим переменную counter в порт C
    };
    EIFR|=0b01;                 //Сброс флага прерывания INT0, который, возможно установится повторно из-за дребезга
```

};

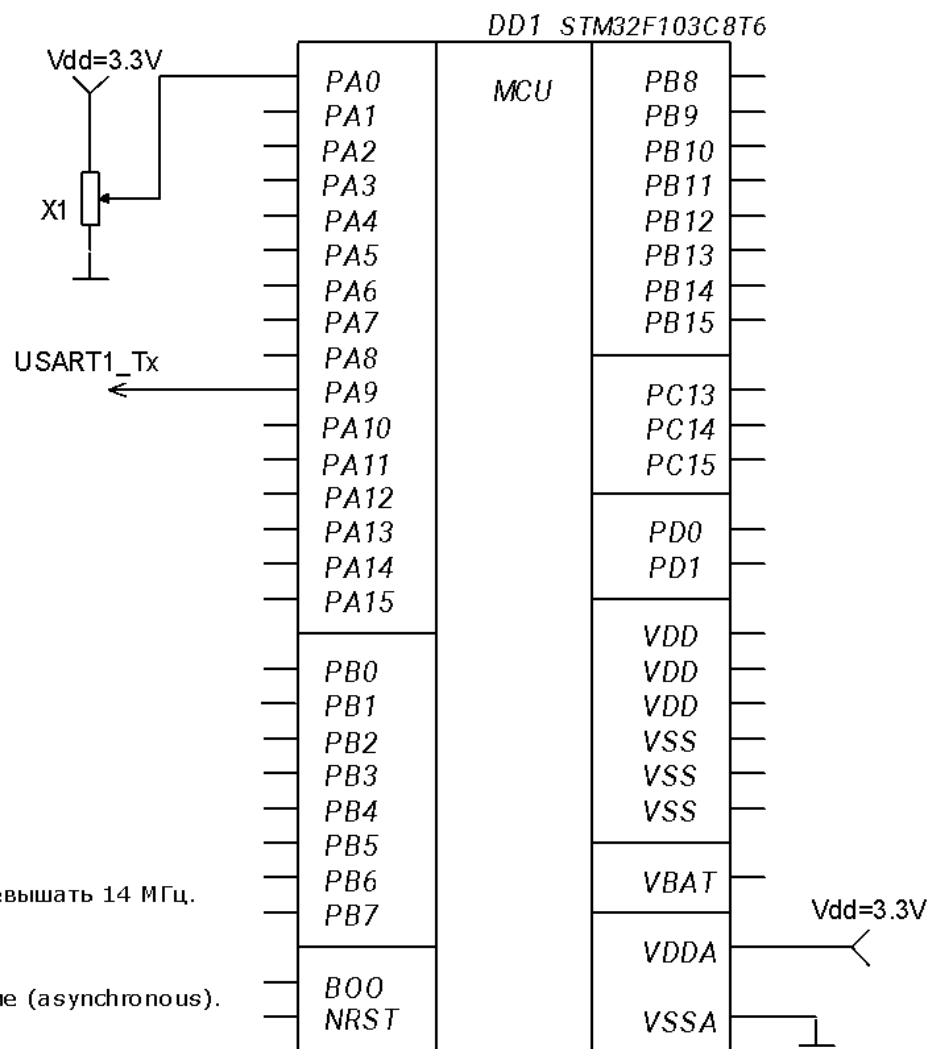
Принципиальная схема и макет проекта





Задача 2(а). АЦП с обменом по поллингу

Организовать аналого-цифровое преобразование напряжения на движке потенциометра. Выполнить циклический вывод результата в виде целого десятичного числа на последовательный терминал через интерфейс UART с интервалом в 1 с. Для определения момента готовности преобразованных данных АЦП использовать *поллинг*.



Частота тактирования АЦП не должна превышать 14 МГц.
Включается ADC1.

Включается USART1 в асинхронном режиме (asynchronous).
data direction - transmitt only

Настройка проекта в CubeMX

Настройка внутренней конфигурации МК

STM32CubeMX 01_ADC1-polling.ioc: STM32F103C8Tx

File Window Help Hello Marina

Home > STM32F103C8Tx > 01_ADC1-polling.ioc - Clock Configuration

GENERATE CODE

Pinout & Configuration Clock Configuration Project Manager Tools

Resolve Clock Issues

Input frequency 32.768 0-1000 KHz

LSE 40 KHz

LSI RC 40 KHz

HSE 8 MHz

HSI RC 8 MHz

Input frequency 8 4-16 MHz

HSE

PLL Source Mux

PLL *PLLMul X9

PLLCCLK

System Clock Mux

SYSCCLK (MHz) 72

AHB Prescaler /1

HCLK (MHz) 72 72 MHz max

APB1 Prescaler /2

PCLK1 36 36 MHz max

X2

APB1 peripheral clocks (MHz) 72

APB1 Timer clocks (MHz) 72

APB2 Prescaler /1

PCLK2 72 72 MHz max

X1

APB2 peripheral clocks (MHz) 72

APB2 timer clocks (MHz) 72

ADC Prescaler /6

To ADC1,2 12 *

RTC Clock Mux

HSE /128 HSE_RTC

LSE

LSI

To RTC (KHz) 40

To IWDG (KHz) 40

To FLITFCLK (MHz) 8

USB Prescaler /1

To USB (MHz) 72

MCO source Mux

PLLCCLK /2

HSI

HSE

SYSCCLK

(MHz) MCO 72

Enable CSS

Настройка ADC1

The screenshot displays the STM32CubeMX software interface for configuring the ADC1. The main window is titled "02_ADC1-interrupt.ioc - Pinout & Configuration". The left sidebar shows a tree view with "ADC1" selected under the "Analog" category. The central panel shows the "ADC1 Mode and Configuration" settings, where the "Mode" is set to "Independent mode" and "IN0" is selected. Below this, the "Configuration" section is visible, with "Parameter Settings" selected. The right panel shows the "Pinout view" of the STM32F103C8Tx LQFP48 package, with the ADC1_IN0 pin highlighted in green.

Настройка USART1

The screenshot displays the STM32CubeMX interface for configuring USART1 on an STM32F103C8Tx microcontroller. The main configuration window shows the following settings:

- Mode:** Asynchronous
- Hardware Flow Control (RS232):** Disable

The configuration parameters are as follows:

Category	Parameter	Value
Basic Parameters	Baud Rate	115200 Bits/s
	Word Length	8 Bits (including Parity)
	Parity	None
	Stop Bits	1
Advanced Parameters	Data Direction	Transmit Only
	Over Sampling	16 Samples

The Pinout view on the right shows the physical pin connections for USART1:

- PA9:** USART1_TX
- PA10:** USART1_RX
- PA13:** SYS_JTMS-SWDIO
- PA14:** SYS_JTCK-SWCLK

Other pins shown include VDD, VSS, VBAT, PC13-15, PD0-1, NRST, VSSA, VDDA, PA0-2, PA3-8, PB0-15, and PB10-11.

Фрагменты файла *main.c*

```

/* Private includes -----*/
/* USER CODE BEGIN Includes */
#include "string.h"      //заголовочный файл для строковых функций, напр. strlen()
#include "stdio.h"      //заголовочный файл для функций вв-выв на std
/* USER CODE END Includes */
//.....
..
/* USER CODE BEGIN PV */
char uart_str[64] = {0,}; //массив для строки, выводимой на терминал
uint16_t adc = 0;        //переменная для выходных данных АЦП
/* USER CODE END PV */
int main(void)
{
.....
/* USER CODE BEGIN 2 */
HAL_ADCEx_Calibration_Start(&hadc1); //калибровка АЦП1
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    HAL_ADC_Start(&hadc1); //запуск АЦП1 на преобразование
    HAL_ADC_PollForConversion(&hadc1, 100); //ожидание готовности выходных данных АЦП
    adc = HAL_ADC_GetValue(&hadc1); //чтение преобразованных данных в переменную adc
    HAL_ADC_Stop(&hadc1); //останов АЦП (не обязательно)
    snprintf(uart_str, 63, "ADC=%d\n\r", adc); //формирование строки вывода на терминал
    HAL_UART_Transmit(&huart1, (uint8_t*)uart_str, strlen(uart_str), 1000); //вывод сформ.строки
    HAL_Delay(1000); //задержка между циклами вывода
/* USER CODE END WHILE */

```

```
}  
} //конец модуля main
```

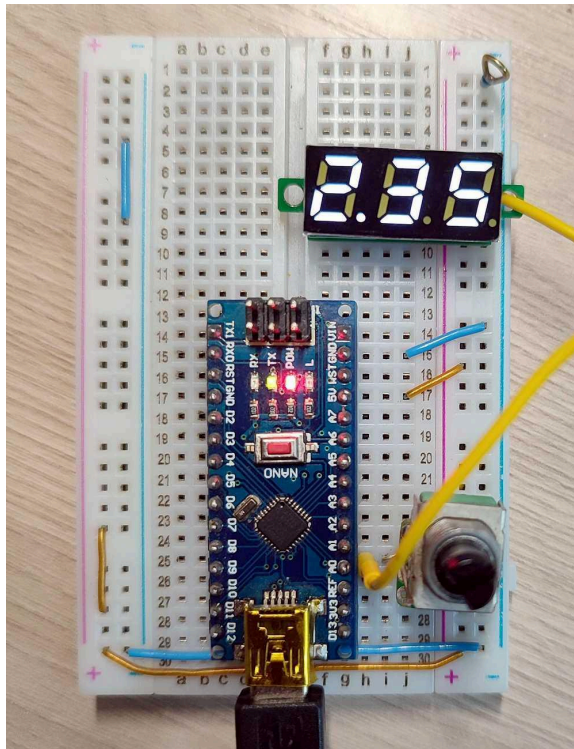
ТА ЖЕ ЗАДАЧА (АЦП С ОБМЕНОМ ПО ПОЛЛИНГУ) В СРЕДЕ ARDUINO ДЛЯ ПЛАТЫ ARDUINO NANO

/*Чтение напряжения на движке потенциометра на A0 (PC0), включенного между +питания и землей и посылка его на терминал*/

```
int PotPin=0; //выбираем аналоговый вход 0 (PC0)
```

```
void setup() {  
  analogReference(DEFAULT); //фактически устанавливается само (опорное напряжение по умолчанию - напряжение питания МК, здесь 4,67В)  
  Serial.begin(9600); //установка скорости обмена с терминалом  
}
```

```
void loop() {  
  Serial.print(F("Voltage on A0=")); //Печать заголовка  
  Serial.print(analogRead(PotPin)); //печать цифрового эквивалента напряжения на делителе  
  Serial.println(); //ВК, ПС - CR, LF. Переход в начало следующей строки  
  delay(500);  
}
```



Output Serial Monitor x

Message (Enter to send message to 'Arduino Nano' on 'COM5') No Line Ending 9600 baud

```
Voltage on A0=513
Voltage on A0=513
Voltage on A0=513
Voltage on A0=513
Voltage on A0=513
Voltage on A0=513
Voltage on A0=513
Voltage on A0=513
Voltage on A0=513
Voltage on A0=513
Voltage on A0=513
```

Ln 16, Col 1 Arduino Nano on COM5 2

Задача 2(б). АЦП с ОБМЕНОМ ПО ПРЕРЫВАНИЮ

Организовать аналого-цифровое преобразование напряжения на движке потенциометра. Выполнить циклический вывод результата в виде целого десятичного числа на последовательный терминал через интерфейс UART с интервалом в 1 с. Для определения момента готовности преобразованных данных АЦП использовать *прерывание*.

The screenshot displays the STM32CubeMX software interface for configuring the STM32F103C8Tx microcontroller. The main window is titled "02_ADC1-interrupt.ioc - Pinout & Configuration".

ADC1 Mode and Configuration:

- Mode:** IN0 is selected (checked).
- Configuration:** EXTI Conversion Trigger is set to "Disable".

Configuration Table:

Parameter Settings	User Constants	NVIC Settings	DMA Settings	GPIO Settings
NVIC Interrupt Table				
ADC1 and ADC2 global interrupts	<input checked="" type="checkbox"/>	0	0	

Pinout View:

The pinout view shows the STM32F103C8Tx LQFP48 package with various pins highlighted in green to indicate their configuration:

- GPIO_Output:** PC13
- RCC_OSC_IN:** PD0
- RCC_OSC_OUT:** PD1
- ADC1_IN0:** PA0
- Other highlighted pins:** PA9, PA10, PA11, PA12, PA13, PA14, NRST, BOOT, PA15, PA14 (SYS_JTCK-SWCLK), PA13 (SYS_JTMS-SWDIO).

Фрагменты файла *main.c*

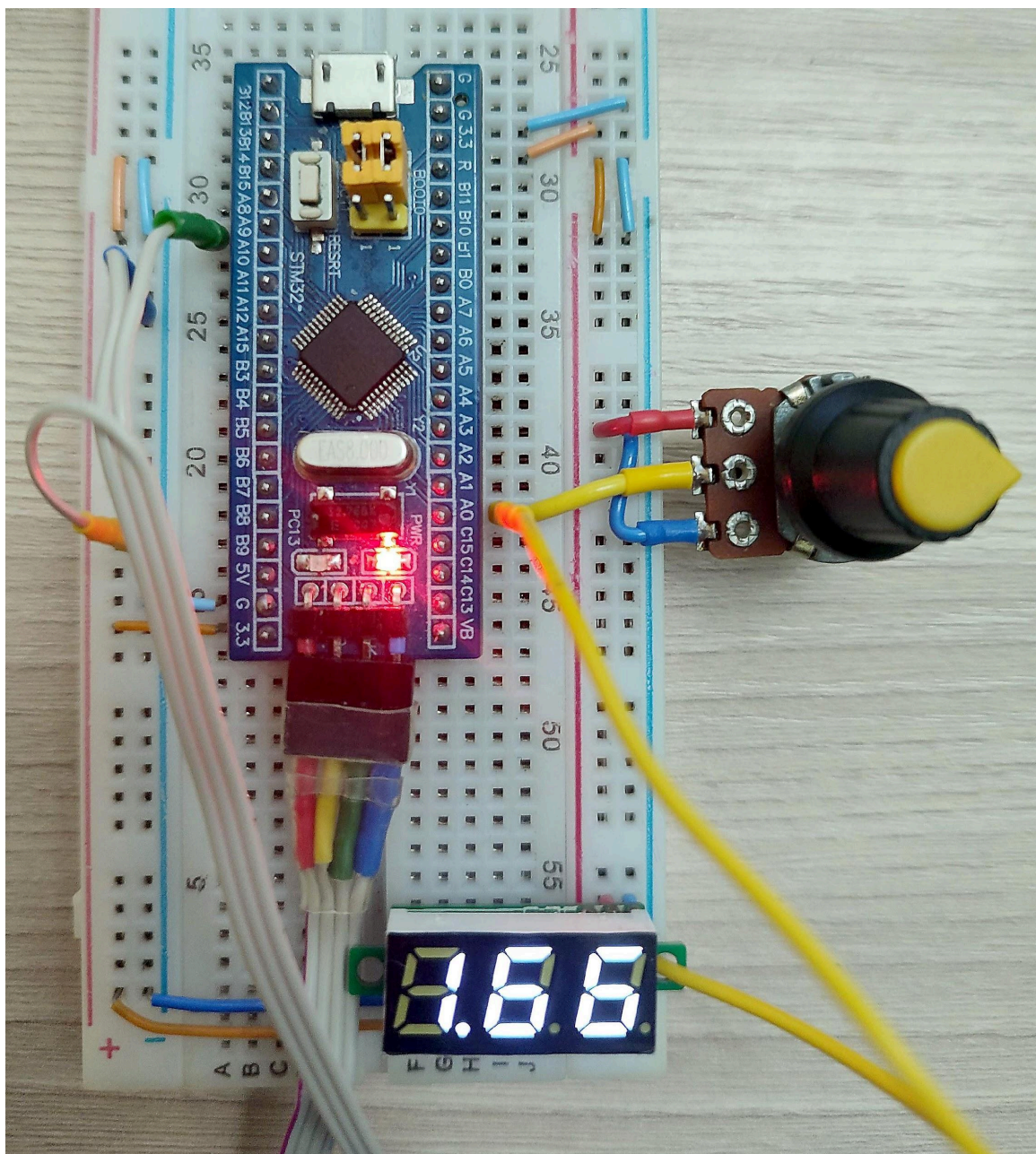
```

/* Private user code -----*/
/* USER CODE BEGIN 0 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)    //фрагмент обработчика прерывания ADC1
{
    if(hadc->Instance == ADC1)                            //проверка прихода прерывания от ACD1
    {
        adc = HAL_ADC_GetValue(&hadc1);                  //чтение результата в переменную adc
        HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13); //изменение состояния PC13 на противоположное
    }
}
/* USER CODE END 0 */
//.....
int main(void)
{
    .....
/* USER CODE BEGIN 2 */
HAL_ADCEx_Calibration_Start(&hadc1);                      //калибровка АЦП1
HAL_ADC_Start_IT(&hadc1);                                 //запуск АЦП1 в режиме с прерываниями
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    snprintf(uart_str,63,"ADC=%d\n\r",adc); //формирование строки вывода на терминал
    HAL_UART_Transmit(&huart1,(uint8_t*)uart_str, strlen(uart_str),1000); //вывод сформ. строки
    HAL_Delay(1000);                                     //задержка между циклами вывода
    HAL_ADC_Start_IT(&hadc1);                             //очередной запуск АЦП1 на преобразование
/* USER CODE END WHILE */
}

```

}

Макет проекта с АЦП



Терминал при медленном вращении ручки потенциометра по и против часовой стрелки

Terminal v1.93b - 20141030Я - by Br@y++

Disconnect	COM Port	Baud rate		
ReScan	COM6	<input type="radio"/> 600	<input type="radio"/> 14400	<input type="radio"/> 57600
Help		<input type="radio"/> 1200	<input type="radio"/> 19200	<input checked="" type="radio"/> 115200
About..	COMs	<input type="radio"/> 2400	<input type="radio"/> 28800	<input type="radio"/> 128000
Quit		<input type="radio"/> 4800	<input type="radio"/> 38400	<input type="radio"/> 256000
		<input type="radio"/> 9600	<input type="radio"/> 56000	<input type="radio"/> custom

Settings

Set font Auto Dis/Connect Time Stream log
 AutoStart Script CR=LF Stay on Top

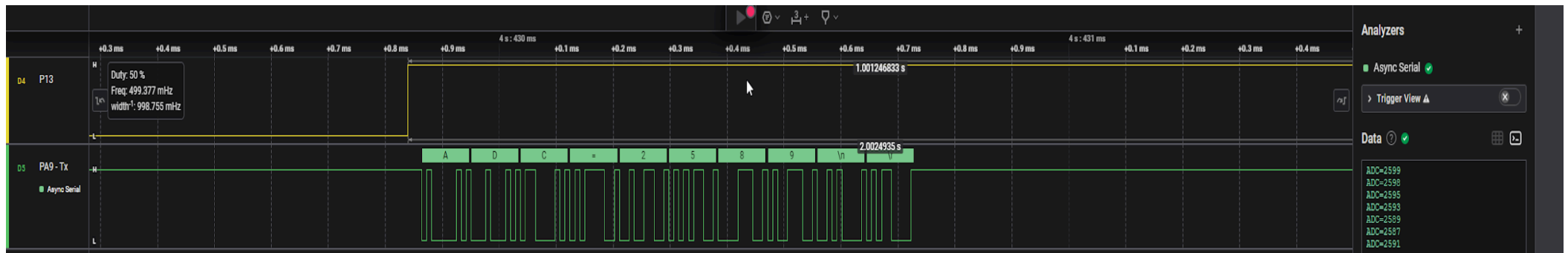
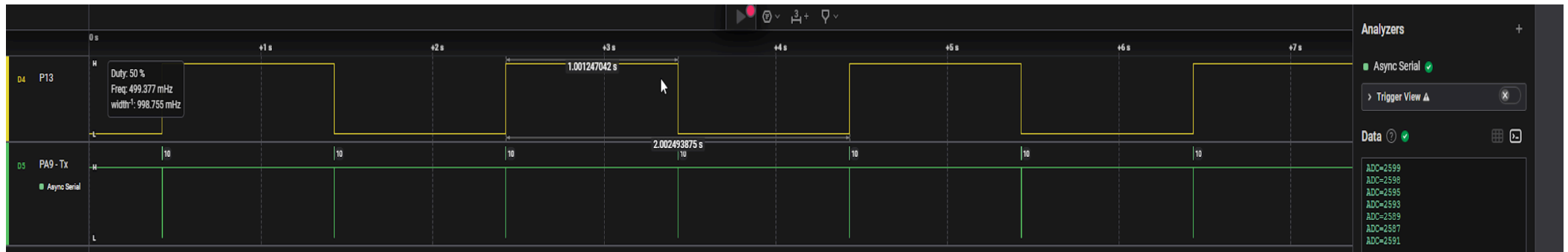
Receive

AutoScroll Cnt = 1070

```

ADC=0
ADC=10
ADC=377
ADC=900
ADC=1537
ADC=2189
ADC=2845
ADC=3474
ADC=4094
ADC=4095
ADC=3984
ADC=3320
ADC=2640
ADC=1895
ADC=1046
ADC=288
ADC=13
ADC=12
    
```

Сигналограммы логического анализатора



Светодиод на РС13 мигает с частотой 0.5 Гц (1 секунда светится, 1 секунда погашен). Смена состояний светодиода происходит в обработчике прерывания от АЦП.

АЦП в непрерывном режиме (не надо программно перезапускать)

The screenshot displays the STM32CubeMX interface for configuring the ADC1 on an STM32F103C8Tx microcontroller. The main window is titled "02_ADC1-interrupt.ioc - Pinout & Configuration".

ADC1 Mode and Configuration:

- Mode:** A list of channels (IN0 to IN9) is shown, with IN0 selected.
- Configuration:**
 - EXTI Conversion Trigger:** Set to "Disable".
 - Parameter Settings:**
 - Mode: Independent mode
 - Data Alignment: Right alignment
 - Scan Conversion Mode: Disabled
 - Continuous Conversion Mode: **Enabled**
 - Discontinuous Conversion Mode: Disabled
 - ADC_Regular_ConversionMode:
 - Enable Regular Conversions: Enable
 - Number Of Conversion: 1
 - External Trigger Conversion Source: Regular Conversion launched by software
 - Rank: 1
 - Channel: Channel 0
 - Sampling Time: 239.5 Cycles
 - ADC_Injected_ConversionMode:
 - Enable Injected Conversions: Disable
 - WatchDog:
 - Enable Analog WatchDog Mode:

Pinout View:

The pinout view shows the physical pins of the STM32F103C8Tx LQFP48 package. The selected configuration is highlighted in green:

- GPIO_Output: PC13-..
- RCC_OSC_IN: PD0-..
- RCC_OSC_OUT: PD1-..
- ADC1_IN0: PA0-..
- PA13: SYS_JTMS-SWDIO
- PA12
- PA11
- PA10: USART1_RX
- PA9: USART1_TX
- PA8
- PB15
- PB14
- PB13
- PB12

Other pins shown include VDD, VSS, PB9, PB8, BOOT, PB7, PB6, PB5, PB4, PB3, PA15, VBAT, NRST, VSSA, VDDA, PA3, PA4, PA5, PA6, PA7, PB0, PB1, PB2, PB10, PB11, and VSS.

Фрагменты файла *main.c*

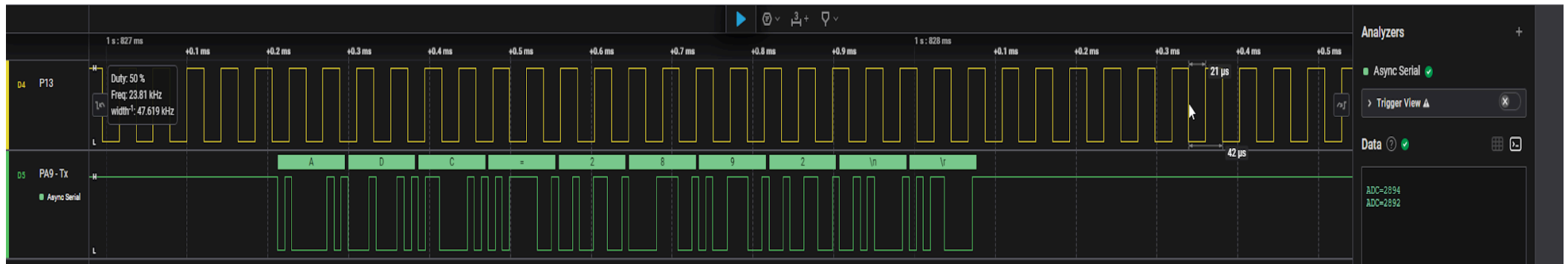
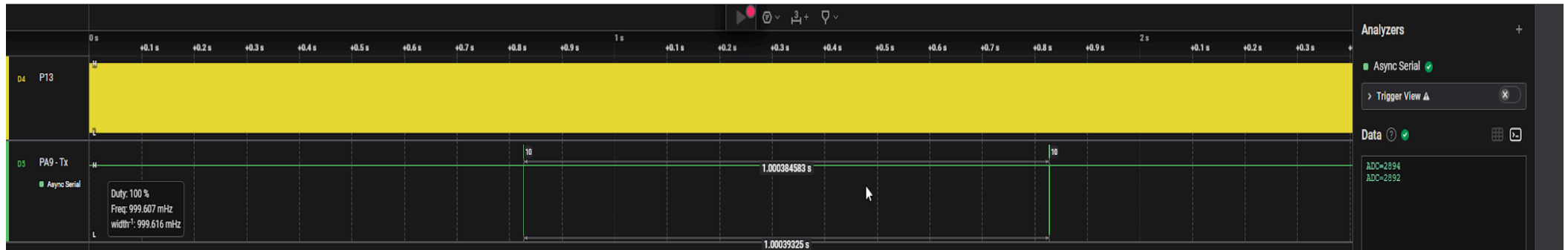
```

/* Private user code -----*/
/* USER CODE BEGIN 0 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)    //фрагмент обработчика прерывания ADC1
{
    if(hadc->Instance == ADC1)                            //проверка прихода прерывания от ACD1
    {
        adc = HAL_ADC_GetValue(&hadc1);                  //чтение результата в переменную adc
        HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13); //изменение состояния PC13 на противоположное
    }
}
/* USER CODE END 0 */
//.....
int main(void)
{
    .....
/* USER CODE BEGIN 2 */
HAL_ADCEx_Calibration_Start(&hadc1);                    //калибровка АЦП1
HAL_ADC_Start_IT(&hadc1);                               //запуск АЦП1 в режиме с прерываниями
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    snprintf(uart_str,63,"ADC=%d\n\r", adc); //формирование строки вывода на терминал
    HAL_UART_Transmit(&huart1,(uint8_t*)uart_str, strlen(uart_str),1000); //вывод сформ. строки
    HAL_Delay(1000);                               //задержка между циклами вывода
    //HAL_ADC_Start_IT(&hadc1);                    //АЦП перезапускать не надо, он перезапустится сам
/* USER CODE END WHILE */
}

```

}

Сигналограммы логического анализатора



Светодиод на PC13 (на отладочной плате) светится в полнакала из-за 50% коэффициента заполнения ШИМ-сигнала на нём.

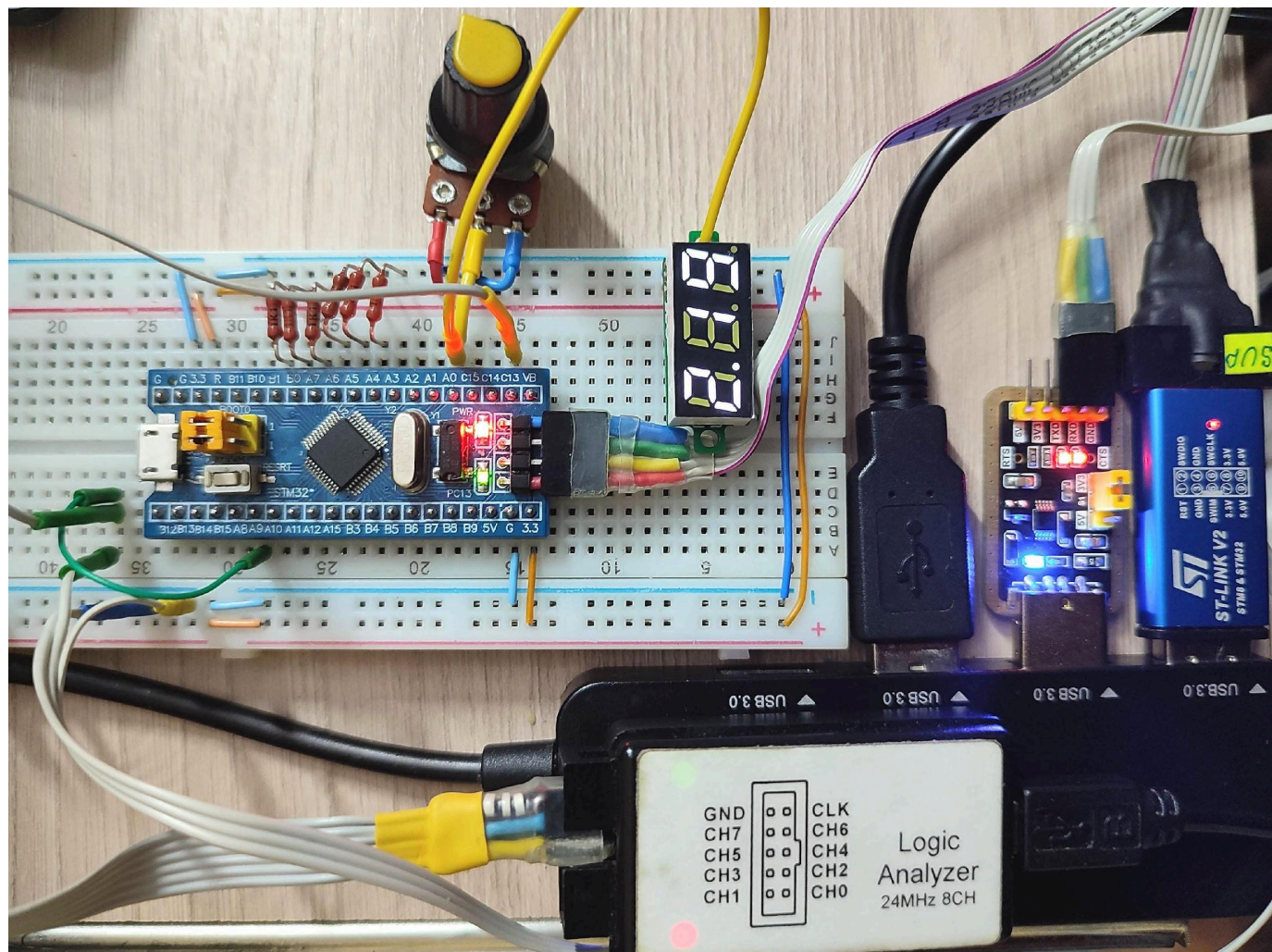
Время преобразования АЦП в тактах:

$T_{conv} = \text{SamplingTime} + 12.5 = 239.5 + 12.5 = 252$ такта.

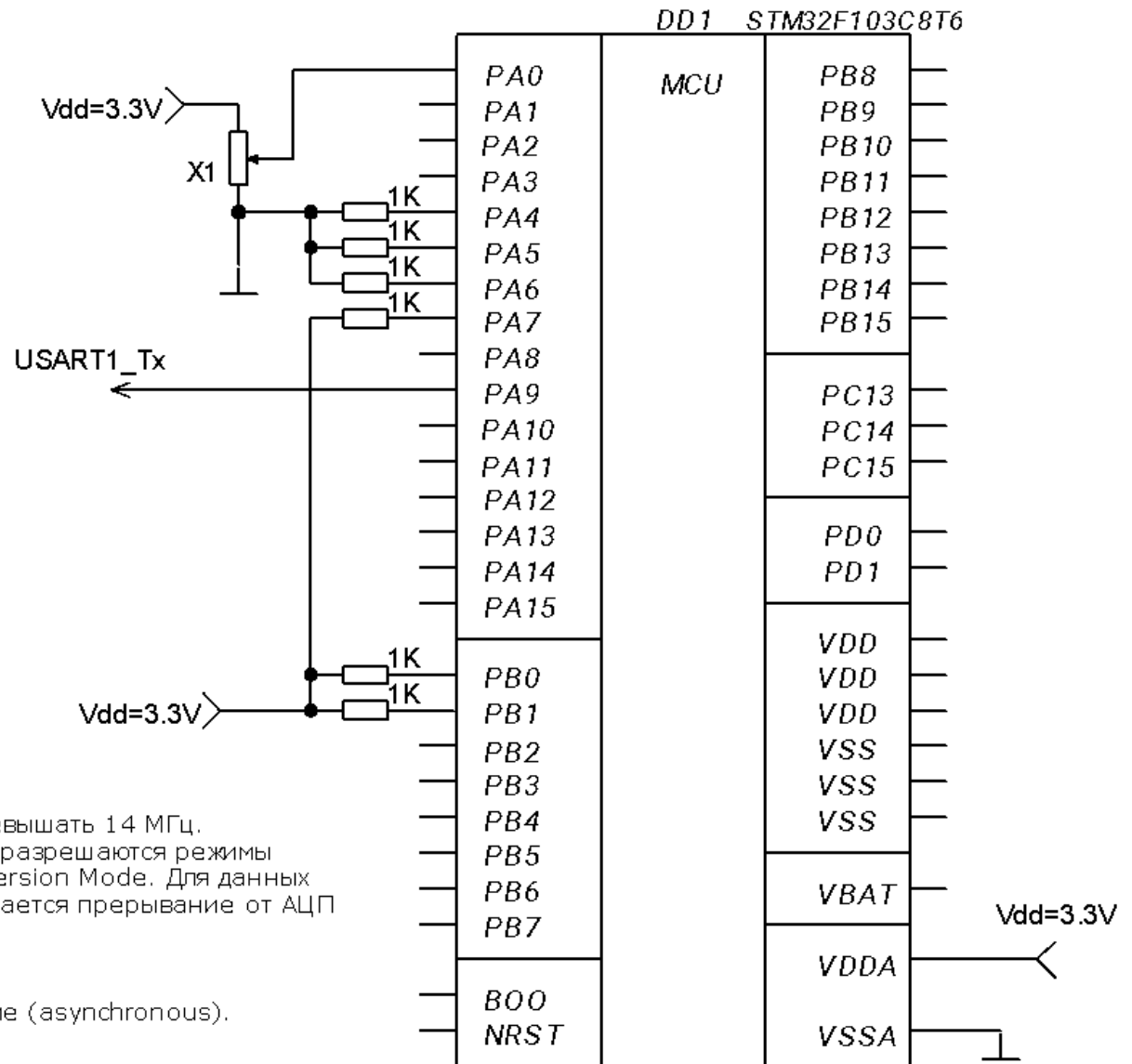
Время преобразования АЦП здесь в секундах: $252 * (1/12E6) = 21E-6$ с = 21 мкс — длительность интервала неизменного состояния светодиода на РС13 (см. нижнюю сигналограмму).

Задача 2(в). АЦП с обменом по DMA

Организовать циклическое аналого-цифровое преобразование напряжений на 10 аналоговых входах МК (РА0-РА7, РВ0, РВ1) с циклической записью в массив из 10 слов *uint_16t adc[10]*. Вывод РА0 подключен к движку потенциометра, выводы РА1РА3 висят неподключенные, РА4-РА6 подключены к «земле», РА7, РВ0, РВ1 подключены к питанию. Выполнить циклический вывод массива результатов на последовательный терминал через интерфейс UART. Для формирования массива в памяти использовать DMA, для определения момента готовности массива преобразованных данных АЦП использовать *прерывание*.



Принципиальная схема проекта



Частота тактирования АЦП не должна превышать 14 МГц.
 Включается ADC1 с 10 каналами In0-In9, разрешаются режимы Scan Conversion Mode и Continuous Conversion Mode. Для данных с АЦП используется DMA Circular и разрешается прерывание от АЦП в этом режиме.

Включается USART1 в асинхронном режиме (asynchronous).
 data direction - transmitt only

Настройка проекта в CubeMX

The screenshot displays the STM32CubeMX interface for configuring the ADC1 peripheral on an STM32F103C8Tx microcontroller. The left sidebar shows the configuration tree with 'ADC1' selected under the 'Analog' category. The main window is divided into two panes: 'ADC1 Mode and Configuration' and 'Pinout view'.

ADC1 Mode and Configuration:

- Mode:** IN0 through IN9 are checked. 'Temperature Sensor Channel' and 'Vrefint Channel' are unchecked. 'EXTI Conversion Trigger' is set to 'Disable'.
- Configuration:**
 - Parameter Settings:**
 - Mode: Independent mode
 - ADCs_Common_Settings:
 - Mode: Right alignment
 - Scan Conversion Mode: Enabled
 - Continuous Conversion Mode: Enabled
 - Discontinuous Conversion Mode: Disabled
 - ADC_Regular_ConversionMode:
 - Enable Regular Conversions: Enable
 - Number Of Conversion: 10
 - External Trigger Conversion Source: Regular Conversion launched by software
 - ADC_Injected_ConversionMode:
 - Enable Injected Conversions: Disable
 - WatchDog:
 - Enable Analog WatchDog Mode:

Pinout view:

The pinout diagram shows the STM32F103C8Tx LQFP48 package with the following connections:

- GPIO_Output:** PC13- (VBAT), PC14-..., PC15-...
- RCC_OSC_IN:** PD0-...
- RCC_OSC_OUT:** PD1-..., NRST
- ADC1_IN0:** PA0-...
- ADC1_IN1:** PA1
- ADC1_IN2:** PA2
- ADC1_IN3:** PA3
- ADC1_IN4:** PA4
- ADC1_IN5:** PA5
- ADC1_IN6:** PA6
- ADC1_IN7:** PA7
- ADC1_IN8:** PB0
- ADC1_IN9:** PB1
- Other pins:** VDD, VSS, PB9, PB8, BOOT0, PB7, PB6, PB5, PB4, PB3, PA15, PA14, PA13, PA12, PA11, PA10, PA9, PA8, PB15, PB14, PB13, PB12, VSSA, VDDA, SYS_JTCK-SWCLK, SYS_JTMS-SWDIO, USART1_RX, USART1_TX.

The screenshot displays the STM32CubeMX interface for configuring the ADC1 peripheral on an STM32F103C8Tx microcontroller. The software is in the 'Pinout & Configuration' tab, showing the 'ADC1 Mode and Configuration' settings.

ADC1 Mode and Configuration:

- Mode:** IN0 through IN9 are selected (checked).
- Configuration:** Temperature Sensor Channel and Vrefint Channel are unchecked. EXTI Conversion Trigger is set to 'Disable'.
- Configuration Tabs:** Parameter Settings, User Constants, NVIC Settings, DMA Settings, and GPIO Settings are active.
- DMA Request Table:**

DMA Request	Channel	Direction	Priority
ADC1	DMA1 Channel 1	Peripheral To Memory	Low
- DMA Request Settings:**
 - Mode: Circular
 - Increment Address:
 - Memory:
 - Data Width: Half Word

Pinout View:

The pinout view shows the physical pins of the STM32F103C8Tx LQFP48 package. The ADC1 channels are mapped to the following pins:

- ADC1_IN0: PA0
- ADC1_IN1: PA1
- ADC1_IN2: PA2
- ADC1_IN3: PA3
- ADC1_IN4: PA4
- ADC1_IN5: PA5
- ADC1_IN6: PA6
- ADC1_IN7: PA7
- ADC1_IN8: PB0
- ADC1_IN9: PB1

Other pins shown include VDD, VSS, VSSA, VDDA, NRST, PD0, PD1, PC13, PC14, PC15, VBAT, BOOT0, PB7, PB6, PB5, PB4, PB3, PB2, PB10, PB11, PA14, PA13, PA12, PA11, PA10, PA9, PA8, PB15, PB14, PB13, PB12, and SYS_JTCK-SWCLK.

The screenshot displays the STM32CubeMX interface for configuring the STM32F103C8Tx microcontroller. The main window is titled "04_ADC1-DMAblock10-circular.ioc - Pinout & Configuration".

ADC1 Mode and Configuration:

- Mode:** IN0 through IN9 are checked. "Temperature Sensor Channel" and "Vrefint Channel" are unchecked. "EXTI Conversion Trigger" is set to "Disable".
- Configuration:** "Reset Configuration" button is visible.
- Parameter Settings:**

Parameter	Enabled	Preemption Priority	Sub Priority
DMA1 channel1 global interrupt	<input checked="" type="checkbox"/>	0	0
ADC1 and ADC2 global interrupts	<input type="checkbox"/>	0	0

Pinout View:

The pinout view shows the STM32F103C8Tx LQFP48 package with various pins assigned to functions:

- GPIO Output:** PC13, PC14, PC15
- RCC_OSC_IN:** PD0, PD1
- RCC_OSC_OUT:** NRST
- VSSA:** VSSA
- VDDA:** VDDA
- ADC1_IN0:** PA0
- ADC1_IN1:** PA1
- ADC1_IN2:** PA2
- Other pins:** VDD, VSS, VBAT, PA3-PA9, PB0-PB15, PA10-PA15, PA13, PA12, PA11, PA9, PA8, PB15, PB14, PB13, PB12, SYS_JTCK-SWCLK, SYS_JTMS-SWDIO, USART1_RX, USART1_TX.

Фрагменты файла *main.c*

```

/* Private includes -----*/
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "string.h"
/* USER CODE END Includes */
//.....
..
/* USER CODE BEGIN PV */
char trans_str[64] = {0,};           //массив для строки, выводимой на терминал
uint16_t adc[10] = {0,};           //у нас 10 каналов, массив для DMA - для 1 набора данных
volatile uint8_t flag = 0;         //флаг, изменяемый в разных модулях
/* USER CODE END PV */
/* Private user code -----*/
/* USER CODE BEGIN 0 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    if(hadc->Instance == ADC1)
        {flag = 1;                   //установка флага
        HAL_GPIO_TogglePin(GPIOC,GPIO_PIN_13);} //Изменение состояния PC13 на противоположное
/* USER CODE END 0
//.....
..
int main(void)
{
.....
/* USER CODE BEGIN 2 */
HAL_ADCEx_Calibration_Start(&hadc1); //калибровка АЦП

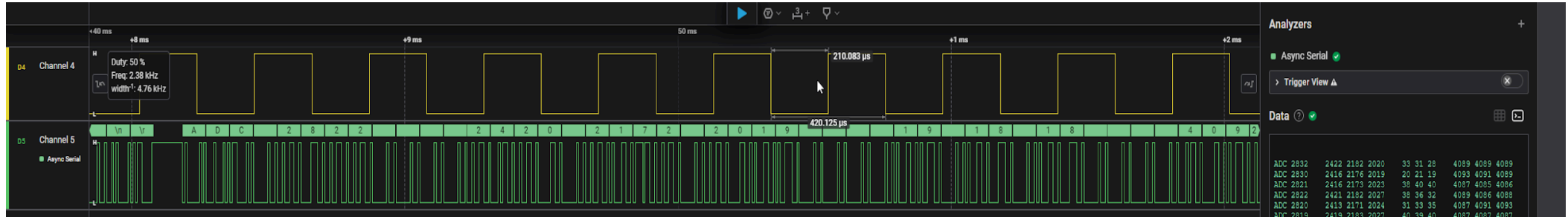
```

```
HAL_ADC_Start_DMA(&hadc1, (uint32_t*) &adc, 10); //старт АЦП в режиме DMA с передачей в память
//10 слов данных с АЦП

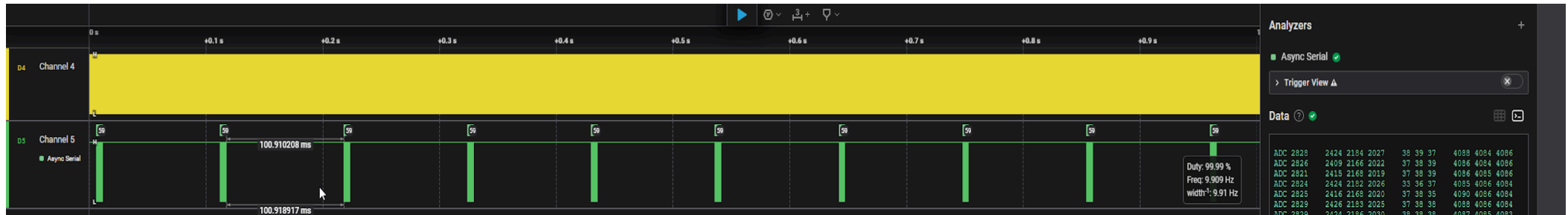
/* USER CODE END 2 */
```

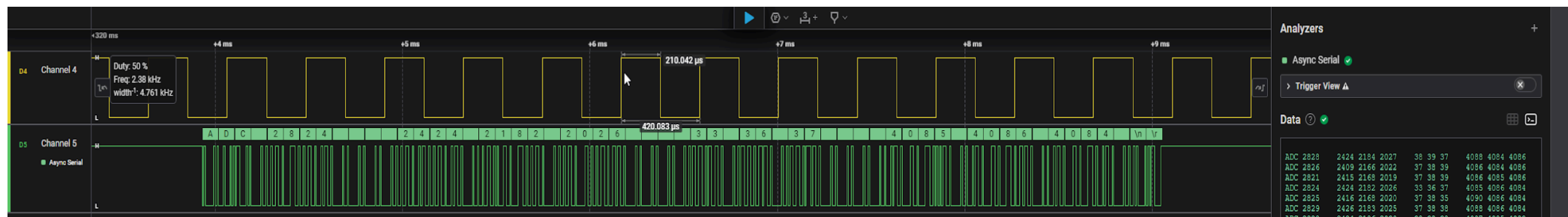
```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    if(flag)
    {
        flag = 0;    //сброс флага
        //формирование строки вывода на терминал для 10 слов данных с АЦП1 (для каналов
0-9)
        snprintf(trans_str, 63, "ADC %d    %d %d %d    %d %d %d    %d %d %d \n\r",
                adc[0],adc[1],adc[2],adc[3],adc[4],adc[5],adc[6],adc[7],adc[8],adc[9]);
        HAL_UART_Transmit(&huart1, (uint8_t*)trans_str, strlen(trans_str), 1000);
        //т.к. DMA циклический, то его перезапускать не надо
        //HAL_Delay(100);    //задержка между циклами вывода на терминал
    }
    // конец if
/* USER CODE END WHILE */
}
/* USER CODE END 3 */
}    //конец main
```

Сигналограммы логического анализатора Без временной задержки между пакетами вывода



С временной задержкой между пакетами вывода





Задача 3. Генератор синуса на основе ЦАП

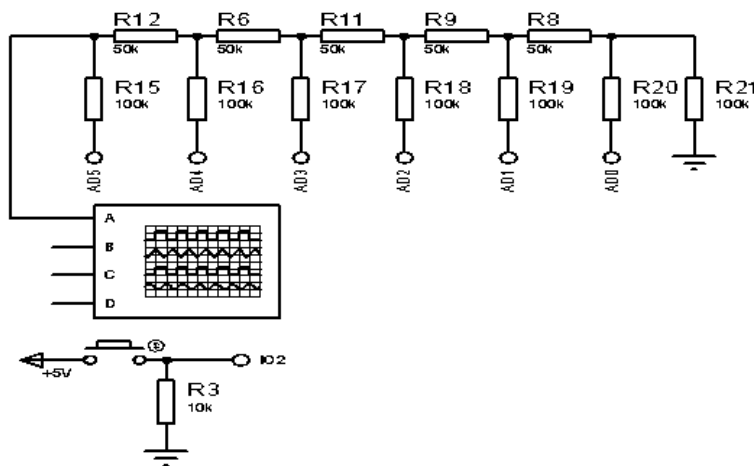
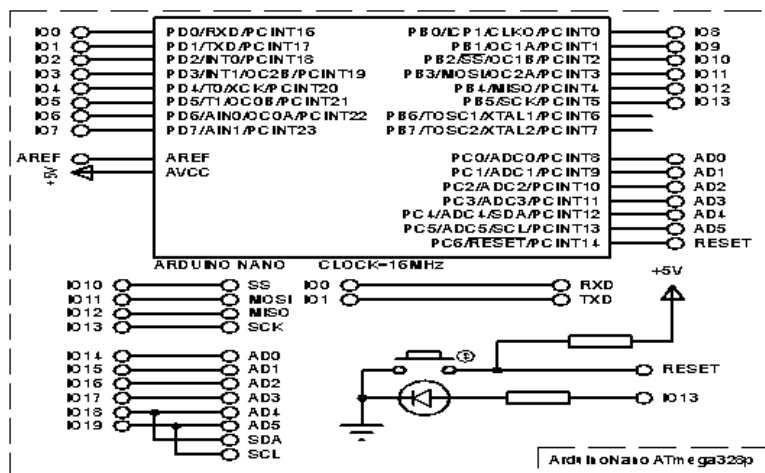
Построить генератор периодического синусоидального сигнала, частота которого увеличивается в 2 раза по каждому клику кнопки. После достижения максимальной частоты (при минимальном количестве отсчетов сигнала за период) клик кнопки приводит к установке минимальной частоты (при максимальном количестве отсчетов за период).

При решении задачи используется принцип циклического считывания цифровых отсчётов синуса за период и послыки их на ЦАП. При этом для формирования минимальной частоты используется максимальное количество отсчётов за период (например, $SAMPLES=1024=2^{10}$). Для формирования максимальной частоты используется минимальное количество отсчетов за период, при котором с помощью фильтра еще можно восстановить непрерывный синус, например, 8. Цифровые отсчеты посылаются на ЦАП, в качестве которого в ATmega328p выступает резистивная матрица R2R, подключенная к 6 младшим разрядам порта C, а в LGT8F328p – внутренний регистр ЦАП DAC0.

После клика кнопки для увеличения частоты в 2 раза, в 2 раза увеличивается шаг считывания табличных данных $step$, т.е. $step \square step*2$. Для этого сигнальный вывод кнопки заводится на линию внешнего запроса прерывания INT0, внешнее прерывание INT0 разрешается и настраивается, а в его обработчике осуществляется увеличение в 2 раза шага считывания данных $step$ из таблицы отсчётов периодической функции $table[SAMPLES]$.

Принципиальные схемы проектов с DAC, программы, макеты

На Arduino nano



```
#define SAMPLES 1024 //количество отсчётов функции за период
#define nDAC 6 //разрядность используемого внешнего ЦАП на резистивной матрице R-2R
#define TWO_PI 6.283185307179586476925286766559

unsigned char table[SAMPLES]; //массив отсчётов периодической функции, тип зависит от разрядности ЦАП
unsigned int dig_max; //максимальный входной цифровой код для ЦАП
unsigned int step=1; //начальный шаг считывания данных в таблице отсчетов

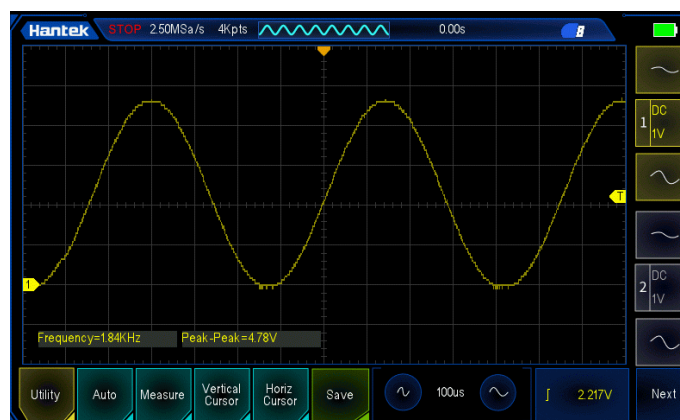
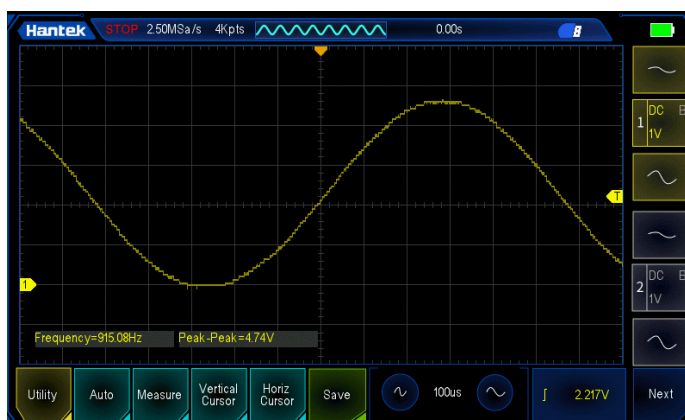
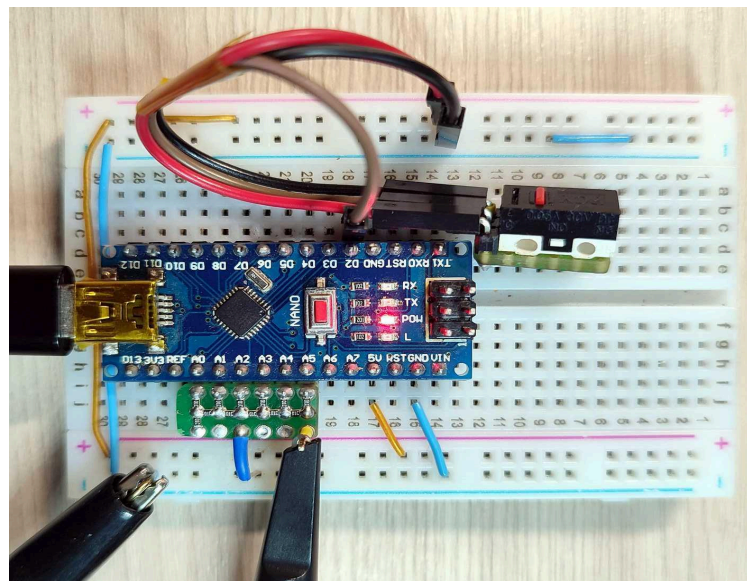
void setup() {
    // put your setup code here, to run once:
    dig_max=(unsigned int)(pow(2,nDAC))-1; //расчёт максимального входного кода для ЦАП
    for (int i = 0; i < SAMPLES; i++) {
        table[i] = (1 + sin((float)i * TWO_PI / SAMPLES))/2*dig_max; //заполнение массива table[SAMPLES] отсчетами синуса за период
    }
    DDRC=0x3F; //6 младших бит порта C - на вывод, к нему подключен ЦАП на резистивной матрице R2R
    PORTC=0x00;
    attachInterrupt(digitalPinToInterrupt(2), ButtonInterrupt_INT0,RISING); //Подключение прерывания INT0 (I02) - срабатывание по
    нарастающему фронту
}

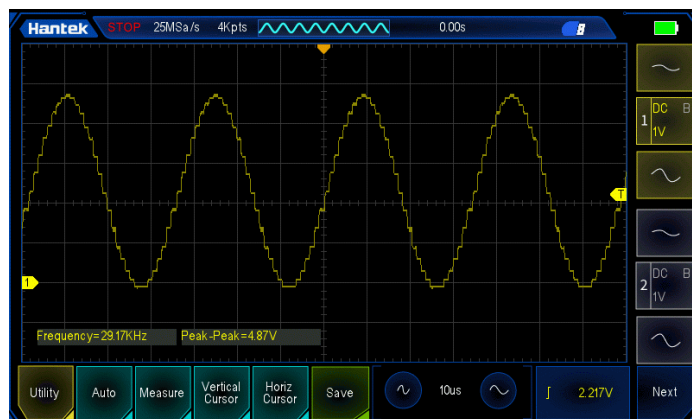
void loop() {
    // put your main code here, to run repeatedly:
    for (unsigned int i = 0; i < SAMPLES; i=i+step) PORTC=table[i]; //вывод очередного цифрового отсчета на ЦАП
}

void ButtonInterrupt_INT0 () {
    for(unsigned int i=0; i<=32000; i++) {asm volatile ("NOP\n");}; //антидребезговая задержка на 2 мс, в прерывании функция delay()
    не работает
    if (digitalRead(2)==HIGH) { //если это не помеха-иголка, а нажатие кнопки - выполняем действие:
        step=step*2; //увеличиваем шаг считывания таблицы отсчетов в 2 раза, частота увеличивается в 2 раза
    }
}
```

```

if (step>(SAMPLES/8)) step=1; //если менее 8 отсчётов за период, возвращаемся к исходному шагу для минимальной частоты
};
EIFR|=0b01; //Сброс флага прерывания INT0, который, возможно установится повторно из-за дребезга
}
    
```






```
#define SAMPLES 1024 //количество отсчётов функции за период
#define nDAC 8 //разрядность внутреннего ЦАП LGT8F328
#define TWO_PI 6.283185307179586476925286766559

unsigned char table[SAMPLES]; //массив отсчётов периодической функции, тип зависит от разрядности ЦАП
unsigned int dig_max; //максимальный входной цифровой код для ЦАП
unsigned int step=1; //исходный шаг считывания данных в таблице отсчетов (для минимальной частоты)

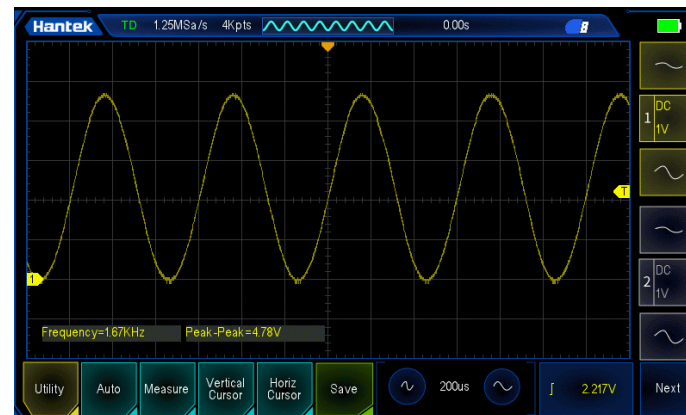
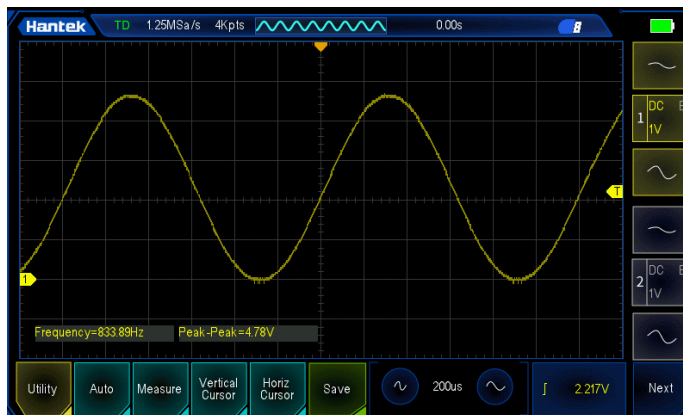
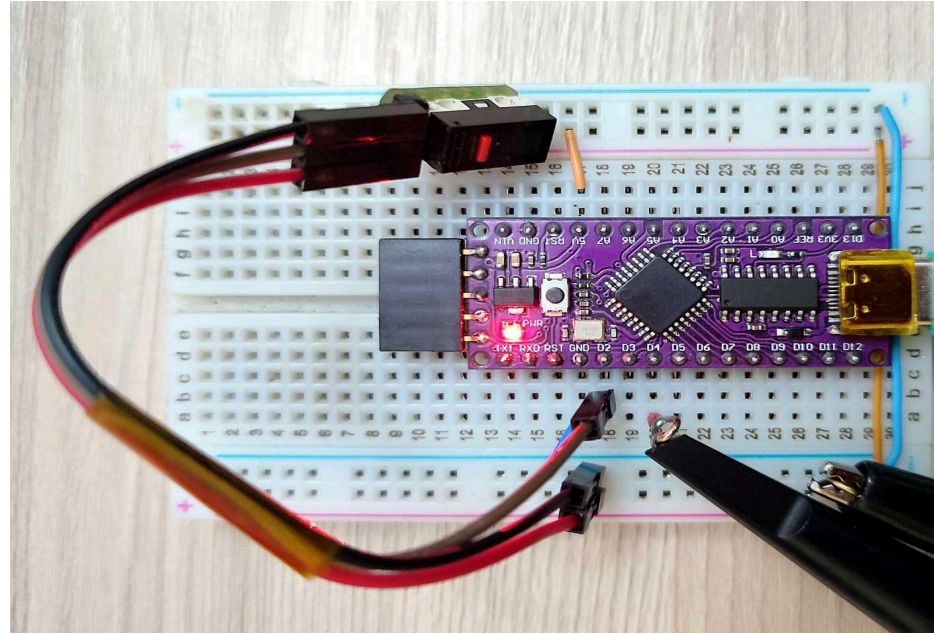
void setup() {
    // put your setup code here, to run once:
    dig_max=(unsigned int)(pow(2,nDAC))-1; //расчёт максимального входного кода для ЦАП
    for (int i = 0; i < SAMPLES; i++) {
        table[i] = (1 + sin((float)i * TWO_PI / SAMPLES))/2*dig_max; //заполнение массива table[SAMPLES] отсчетами синуса за период
    }
    analogReference(DEFAULT); // 5v Установка опорного напряжения ЦАП равного напряжению питания платы
    // analogReference(EXTERNAL); // REF PIN Voltage
    // analogReference(INTERNAL4V096); // 4.096V
    // analogReference(INTERNAL2V048); // 2.048v
    // analogReference(INTERNAL1V024); // 1.024v
    pinMode(DAC0, ANALOG);
    attachInterrupt(digitalPinToInterrupt(2), ButtonInterrupt_INT0,RISING); //Подключение прерывания INT0 (I02) - срабатывание по
    нарастающему фронту
}

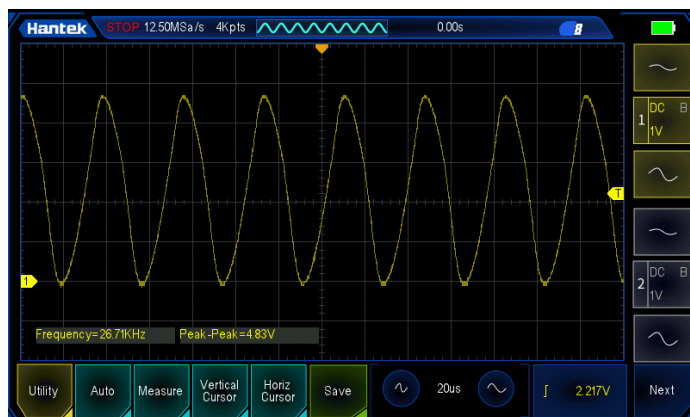
void loop() {
    // put your main code here, to run repeatedly:
    for (unsigned int i = 0; i < SAMPLES; i=i+step) analogWrite(DAC0, table[i]); // вывод очередного цифрового отсчета на ЦАП
}

void ButtonInterrupt_INT0 () {
    for(unsigned int i=0; i<=32000; i++) {asm volatile ("NOP\n");}; //антидребезговая задержка на 2 мс, в прерывании функция delay() не
    работает
    if (digitalRead(2)==HIGH) { //если это не помеха-иголка, а нажатие кнопки - выполняем действие:
        step=step*2; //увеличиваем шаг прореживания таблицы отсчетов в 2 раза, частота увеличивается в 2 раза
        if (step>(SAMPLES/8)) step=1; //если менее 8 отсчётов за период, возвращаемся к исходному шагу для минимальной частоты
    };
}
```

```
EIFR|=0b01;  
}
```

```
//Сброс флага прерывания INT0, который, возможно установится повторно из-за дребезга
```





ПОЛЕЗНЫЕ ССЫЛКИ ДЛЯ ИЗУЧЕНИЯ МК STM32

[Таймеры часть 1](#)

[Таймеры часть 2](#)

[АЦП](#)

[DMA часть 1](#)

[DMA часть 2](#)

[Уроки STM32 \(narod\)](#)

[Уроки STM32 \(narod\) YOUTUBE](#)

[А. Рожков – Уроки по STM32](#)

[Уроки STM32](#)

[DIMoonElectronics-блог STM32F1](#)