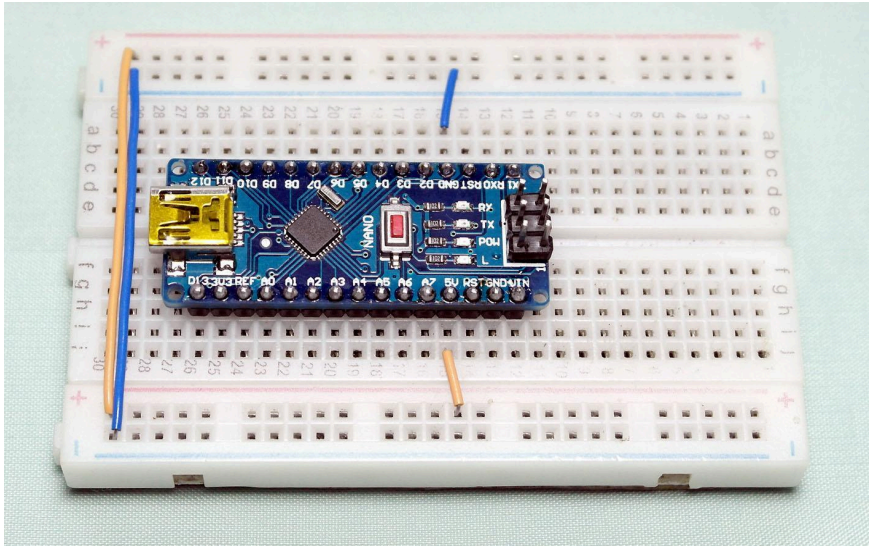
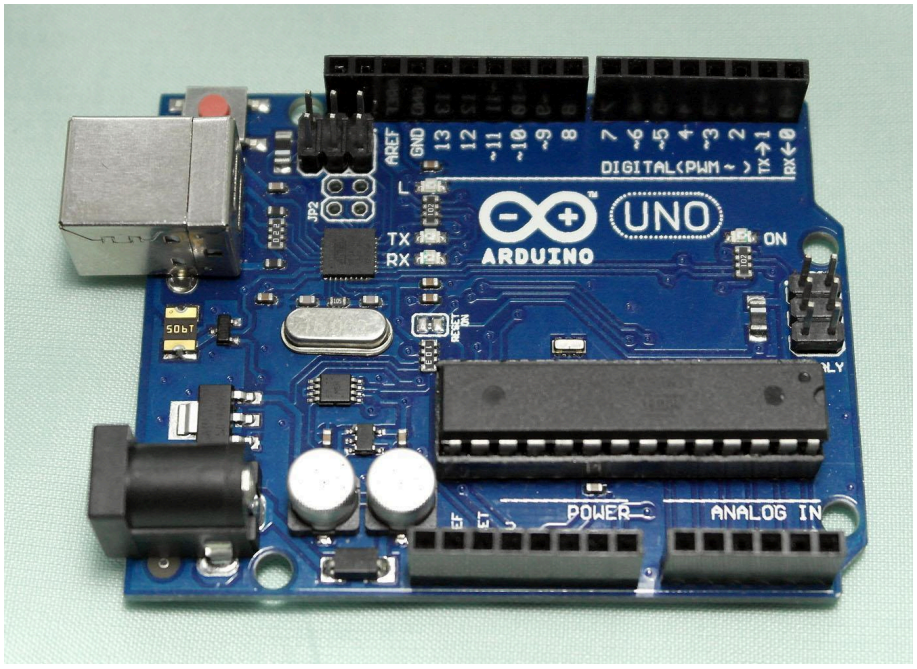
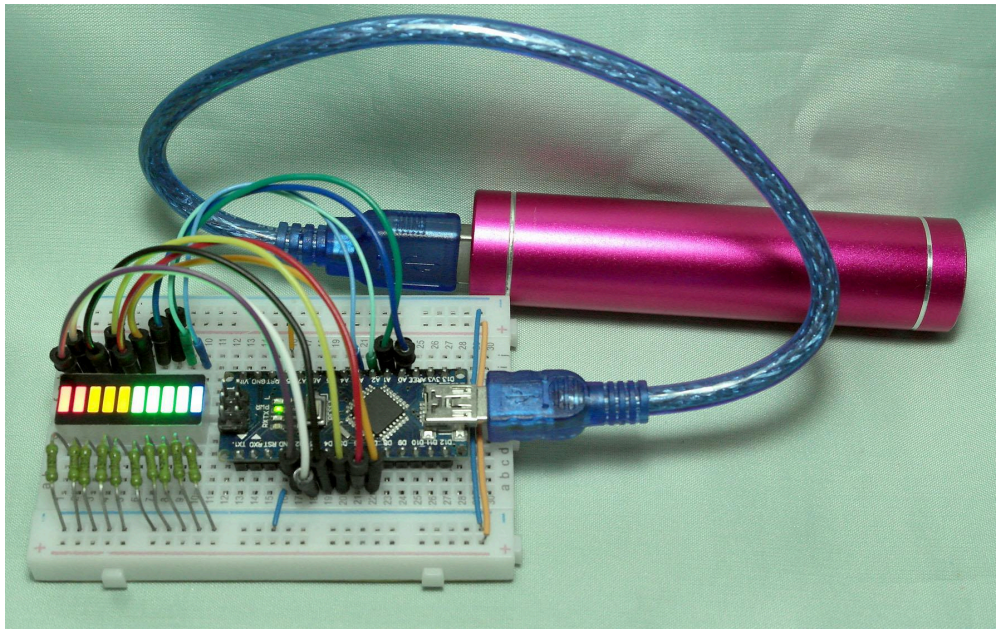
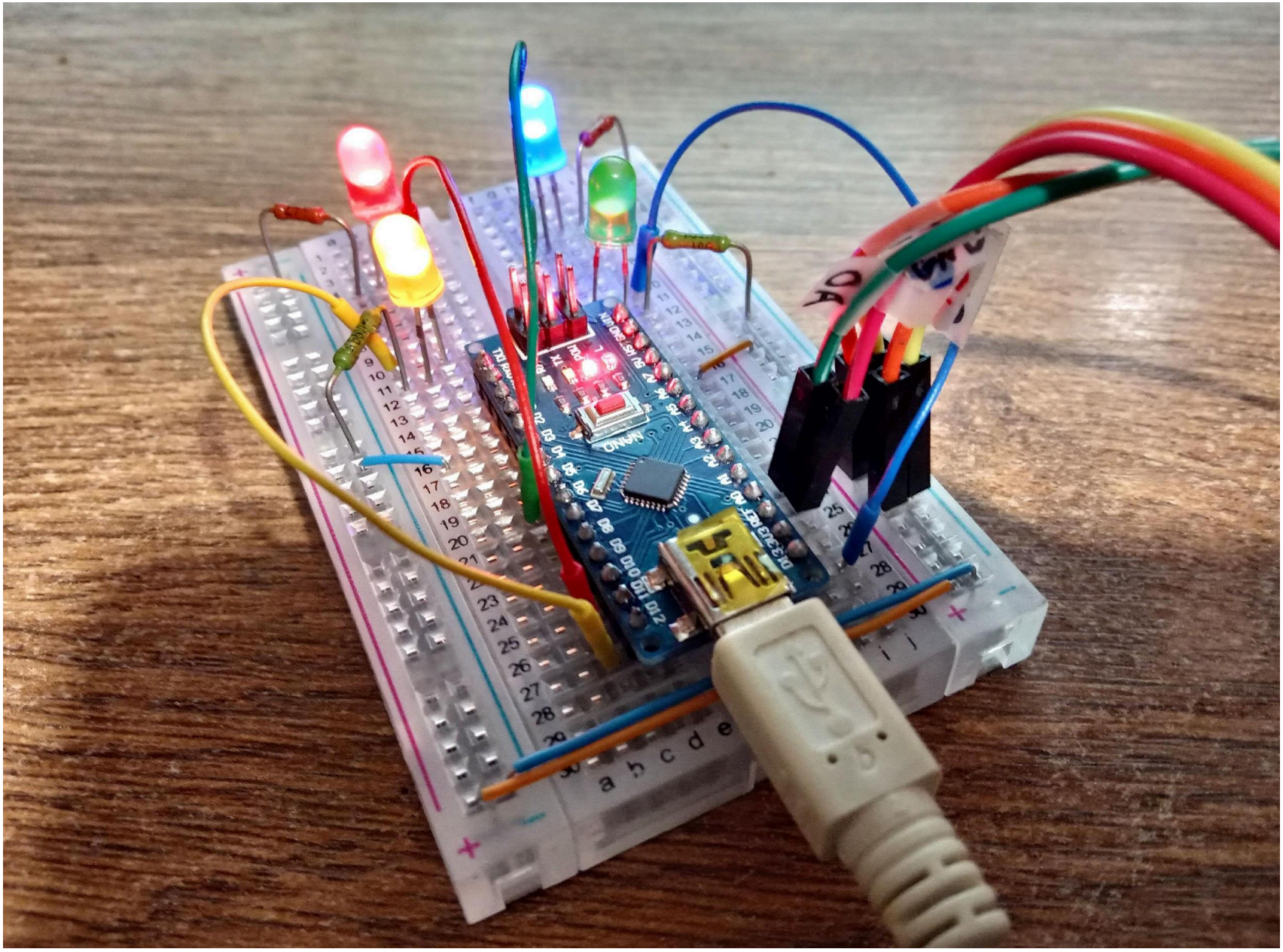
ПЛАТЫ «ARDUINO» (НА ОСНОВЕ МК ATMEGA328P)

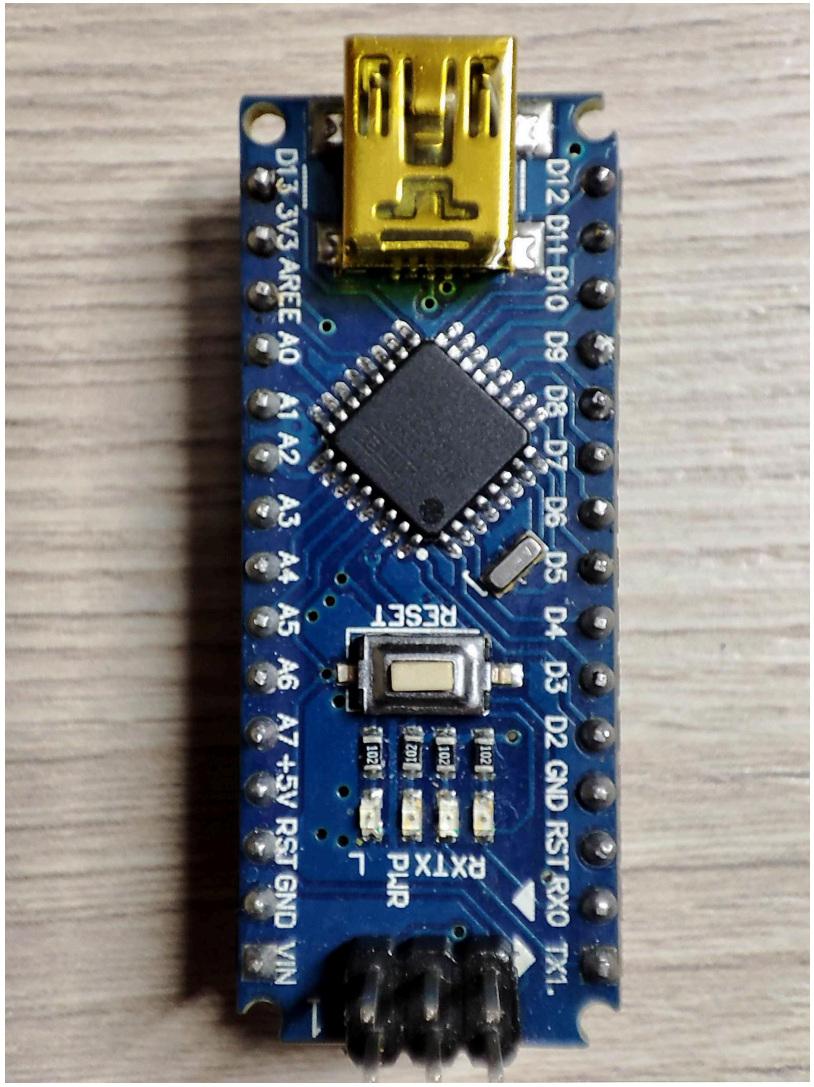


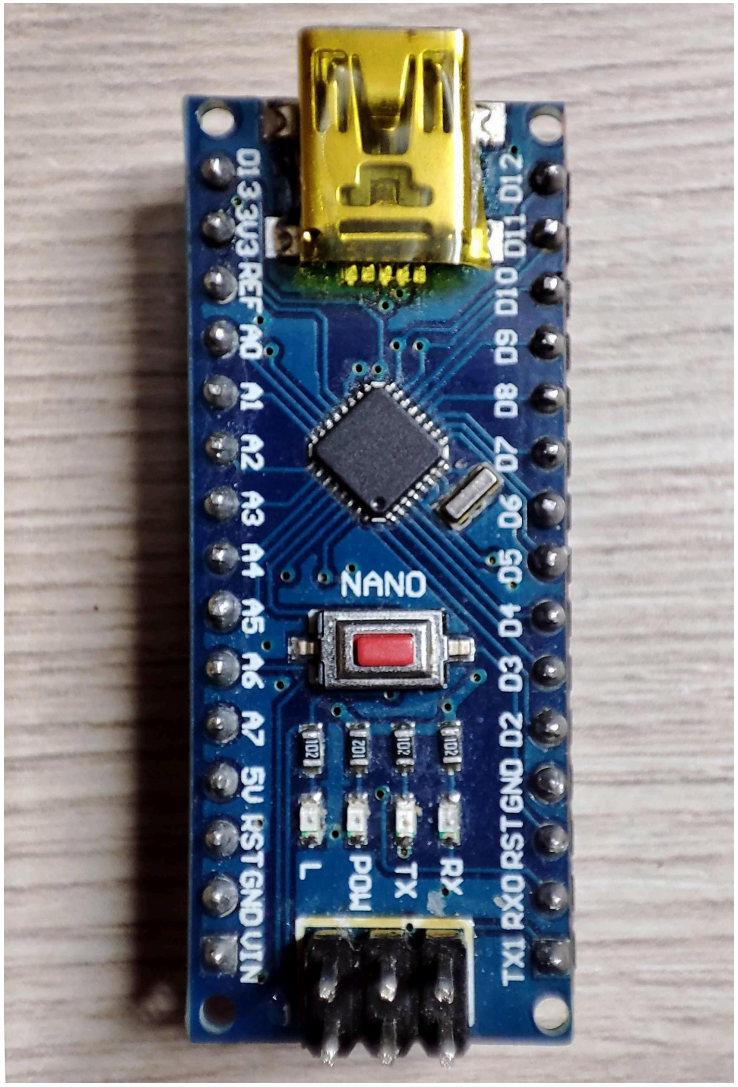




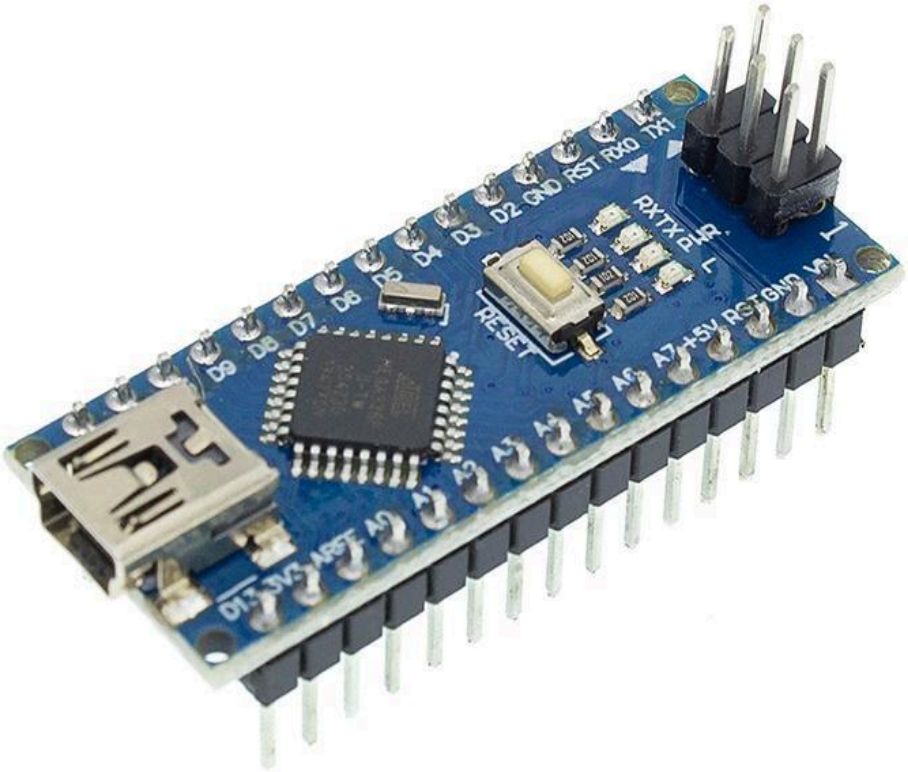


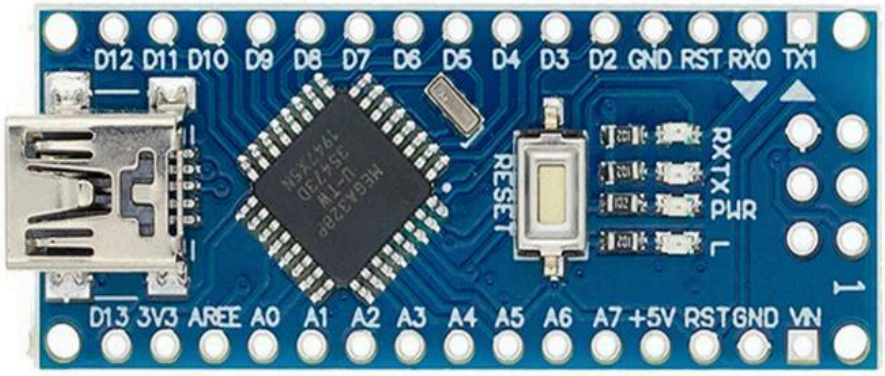
Arduino Nano (Arduino IDE, Code Vision AVR, Proteus)

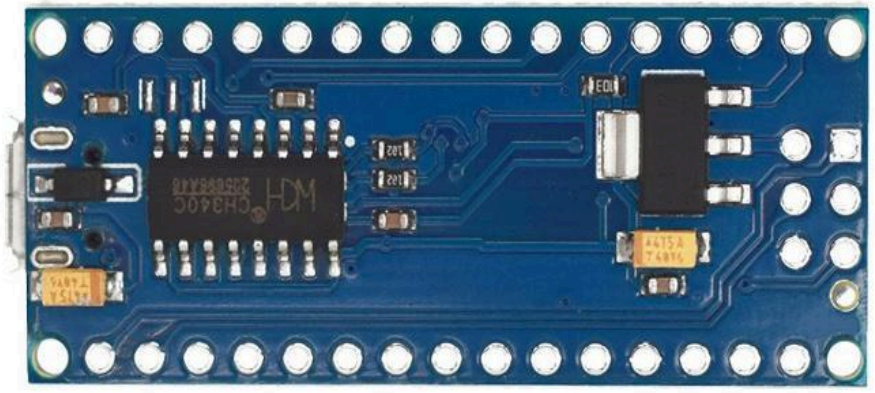




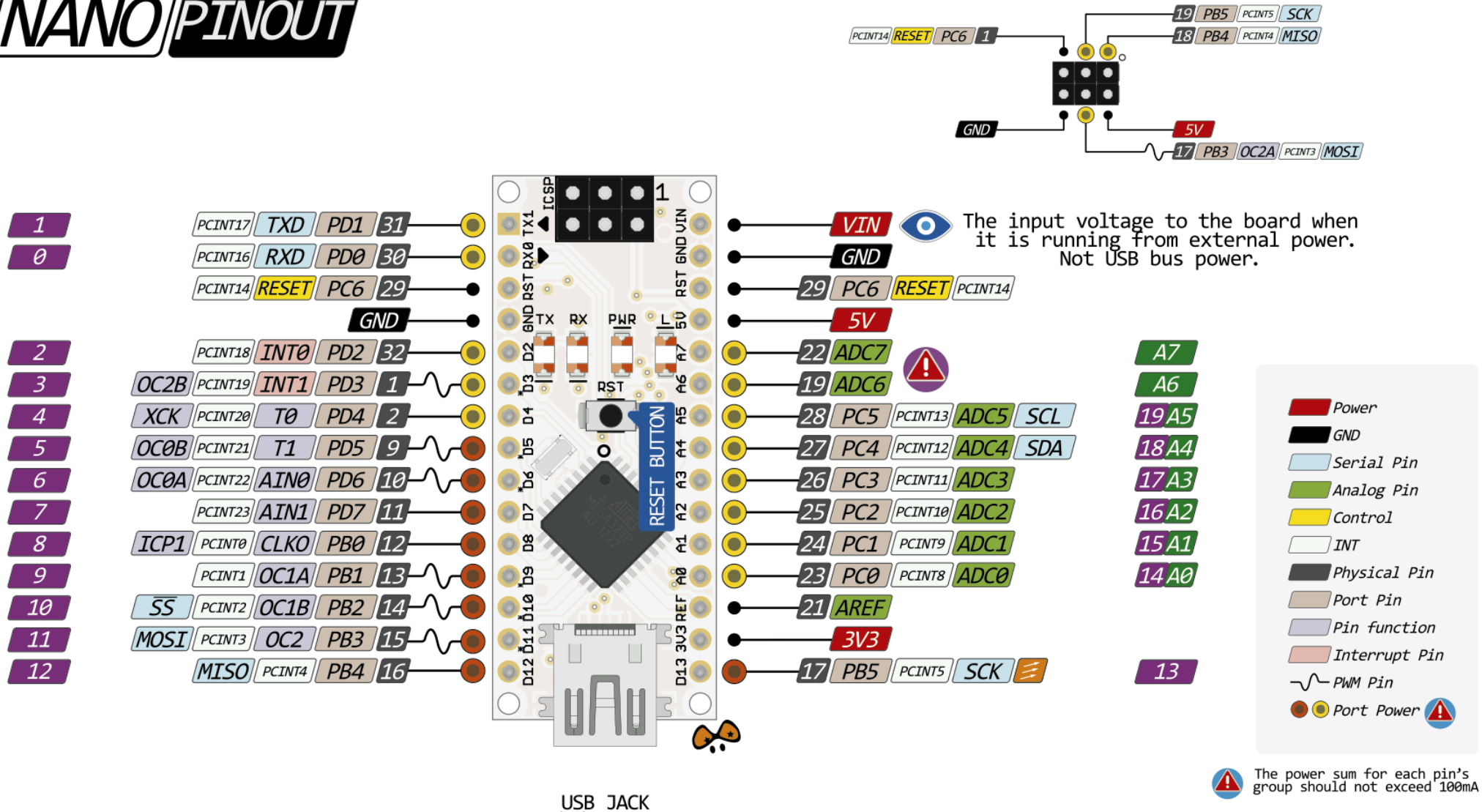
IDE — Integrated Development Environment


D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
RXTX PWR L
RESET
D13 3V3 AREF A0 A1 A2 A3 A4 A5 A6 A7 +5V RST GND VIN

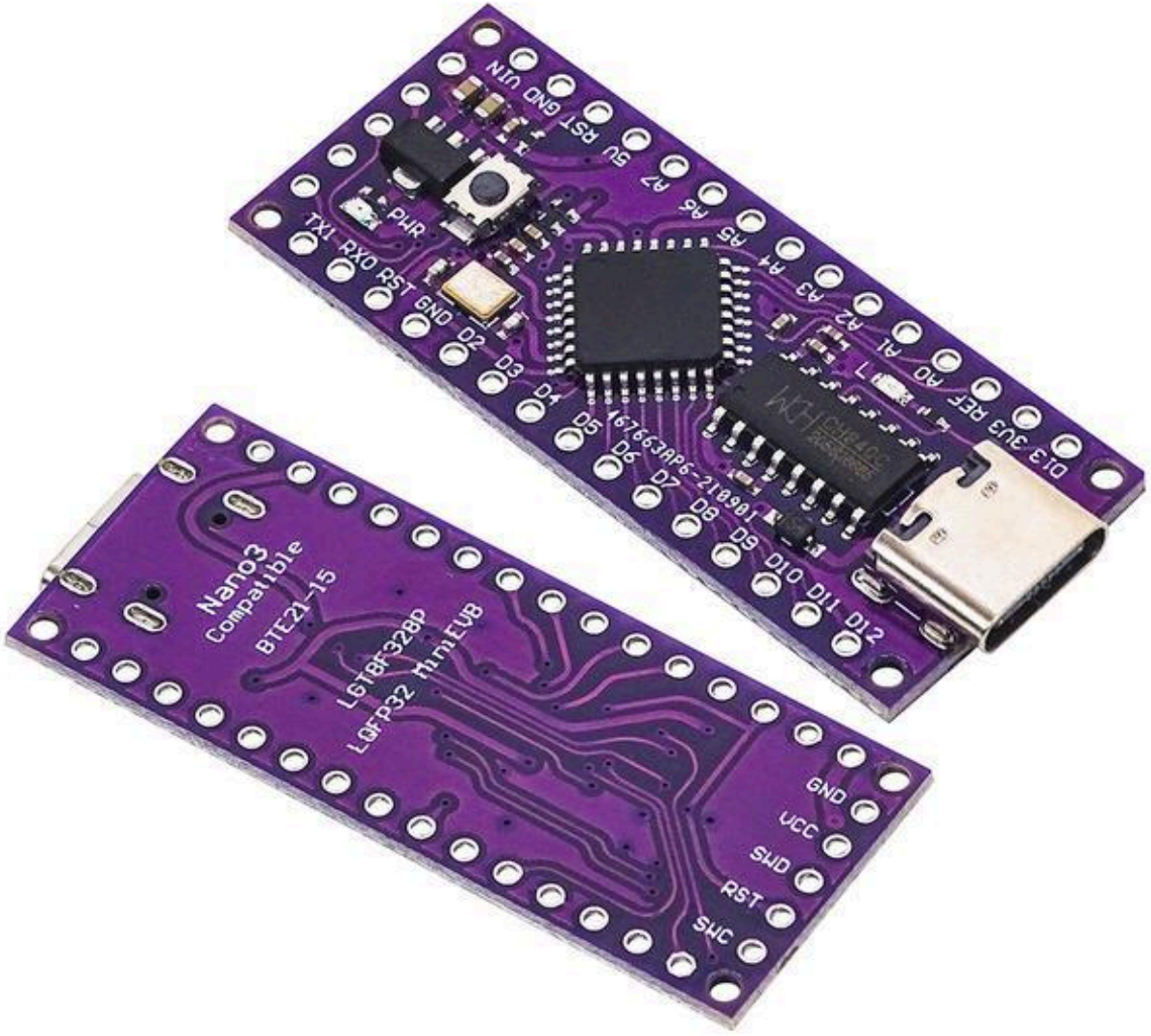

D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
RXTX PWR L
RESET
D13 3V3 AREF A0 A1 A2 A3 A4 A5 A6 A7 +5V RST GND VIN



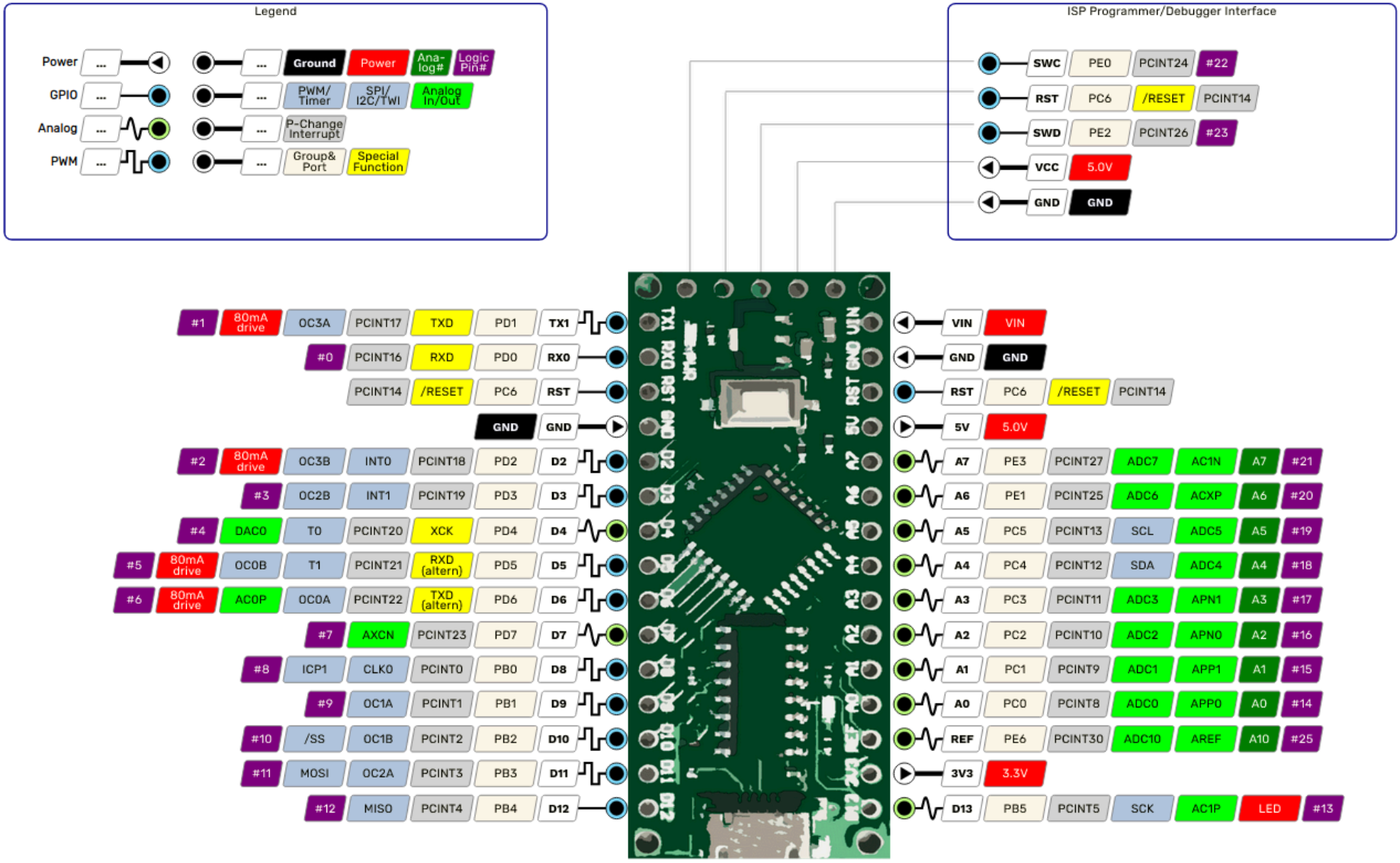
NANO PINOUT


PCINT14 RESET PC6 1
19 PB5 PCINT5 SCK
18 PB4 PCINT4 MISO
GND
5V
17 PB3 OC2A PCINT3 MOSI
1 PCINT17 TXD PD1 31
0 PCINT16 RXD PD0 30
PCINT14 RESET PC6 29
GND
2 PCINT18 INT0 PD2 32
3 OC2B PCINT19 INT1 PD3 1
4 XCK PCINT20 T0 PD4 2
5 OC0B PCINT21 T1 PD5 9
6 OC0A PCINT22 AIN0 PD6 10
7 PCINT23 AIN1 PD7 11
8 ICP1 PCINT0 CLKO PB0 12
9 PCINT1 OC1A PB1 13
10 SS PCINT2 OC1B PB2 14
11 MOSI PCINT3 OC2 PB3 15
12 MISO PCINT4 PB4 16
ICSP
TX RX PWR L
RST
RESET BUTTON
VIN
GND
29 PC6 RESET PCINT14
5V
22 ADC7
19 ADC6
28 PC5 PCINT13 ADC5 SCL
27 PC4 PCINT12 ADC4 SDA
26 PC3 PCINT11 ADC3
25 PC2 PCINT10 ADC2
24 PC1 PCINT9 ADC1
23 PC0 PCINT8 ADC0
21 AREF
3V3
17 PB5 PCINT5 SCK
A7
A6
19 A5
18 A4
17 A3
16 A2
15 A1
14 A0
13
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
USB JACK
Mini Type B
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 старее для Windows 7 (русифицированные)
Arduino IDE 2.3.2 и новее для Windows 10 (англоязычная)

sketch_mar23b | Arduino IDE 2.3.2

File Edit Sketch Tools Help

Auto Format
Archive Sketch
Manage Libraries...
Serial Monitor
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "Arduino Nano"
Port: "COM5"
Get Board Info
Processor: "ATmega328P (Old Bootloader)"
Programmer
Burn Bootloader

Boards Manager...
• Arduino AVR Boards
LGT8fx Boards

Arduino Yún
Arduino Uno
Arduino Uno Mini
Arduino Duemilanove or Diecimila
✓ Arduino Nano
Arduino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Leonardo ETH
Arduino Micro
Arduino Esplora
Arduino Mini
Arduino Ethernet
Arduino Fio
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini
Arduino NG or older
Arduino Robot Control
Arduino Robot Motor
Arduino Gemma
Adafruit Circuit Playground
Arduino Yún Mini
Arduino Industrial 101
Linino One
Arduino Uno WiFi

```
мой способ - работает надёжно

once:
последовательный канал
ягивающим к питанию резистором

(3), MyInterrupt_INT1

void loop() {
  // put your main code here, to run repeatedly:
unsigned char var=(count<<4)|0b1000;          //сдв
PORTD=var;
Serial.println(count);
//Serial.println(var);
delay(500);
}
```

Output

Ln 25, Col 142 Arduino Nano on COM5 1

sketch_mar23b | Arduino IDE 2.3.2

File Edit Sketch Tools Help

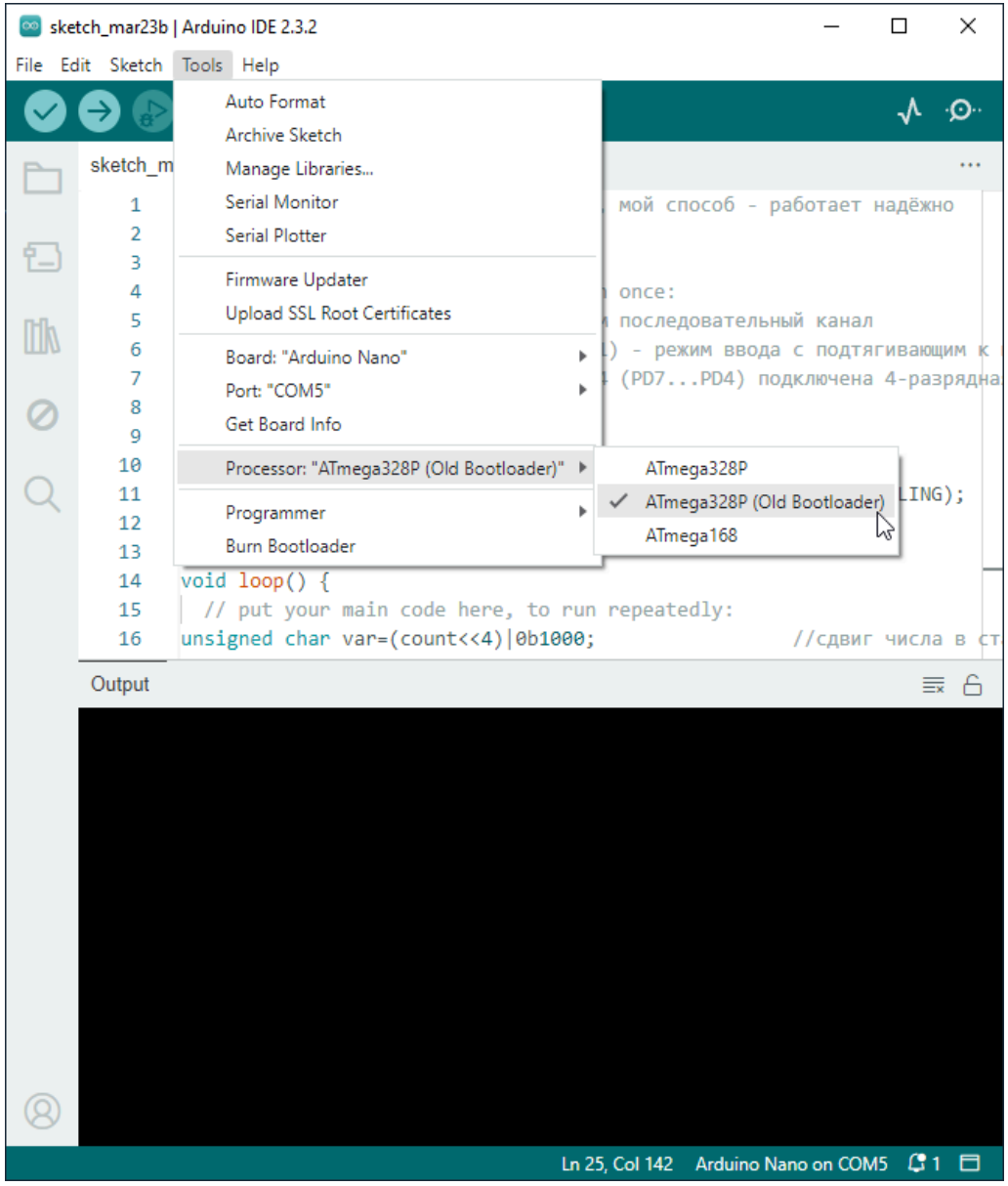
Auto Format
Archive Sketch
Manage Libraries...
Serial Monitor
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "Arduino Nano"
Port: "COM5"
Get Board Info
Processor: "ATmega328P (Old Bootloader)"
Programmer
Burn Bootloader

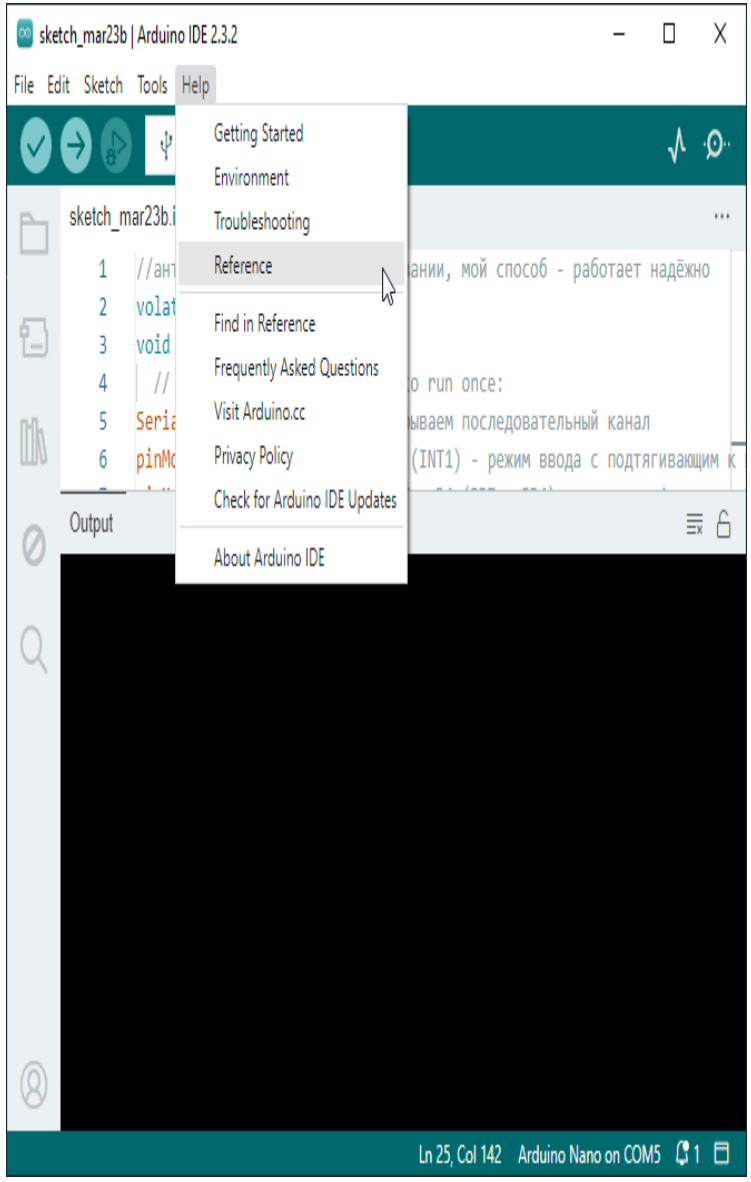
Serial ports
✓ COM5
COM1

```
мой способ - работает надёжно

once:
последовательный канал
) - режим ввода с подтягивающим к
подключена 4-разрядна

(3), MyInterrupt_INT1,FALLING);

void loop() {
  // put your main code here, to run repeatedly:
unsigned char var=(count<<4)|0b1000;          //сдвиг числа в ст
```

Output

Ln 25, Col 142 Arduino Nano on COM5 1


sketch_mar23b | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Auto Format
Archive Sketch
Manage Libraries...
Serial Monitor
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "Arduino Nano"
Port: "COM5"
Get Board Info
Processor: "ATmega328P (Old Bootloader)"
Programmer
Burn Bootloader
ATmega328P
ATmega328P (Old Bootloader)
ATmega168
void loop() {
// put your main code here, to run repeatedly:
unsigned char var=(count<<4)|0b1000;
Output
Ln 25, Col 142 Arduino Nano on COM5


sketch_mar23b | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Getting Started
Environment
Troubleshooting
Reference
Find in Reference
Frequently Asked Questions
Visit Arduino.cc
Privacy Policy
Check for Arduino IDE Updates
About Arduino IDE
Output
Ln 25, Col 142 Arduino Nano on COM5

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»

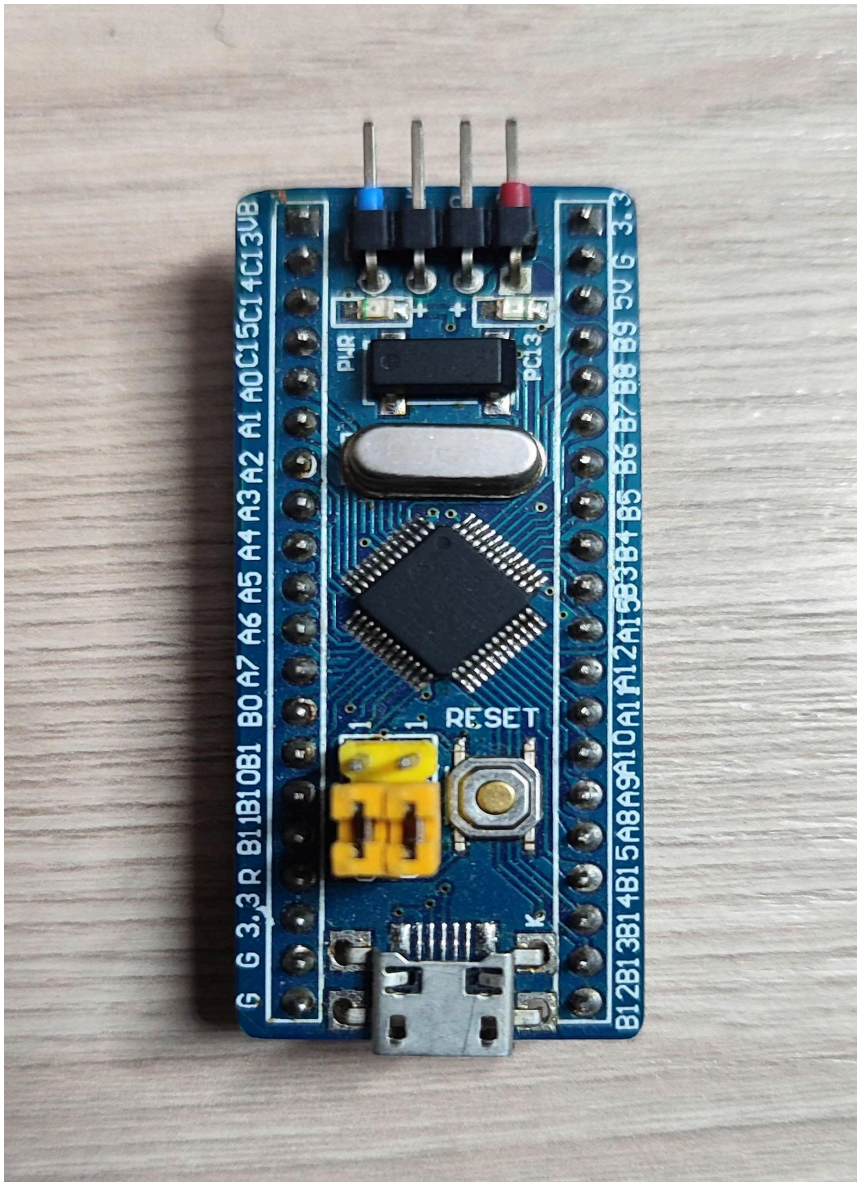
Дистрибутивы:

Code Vision AVR V3.12 (полнофункциональная)

Proteus 7.10, Proteus 8.15, Proteus 8.9.

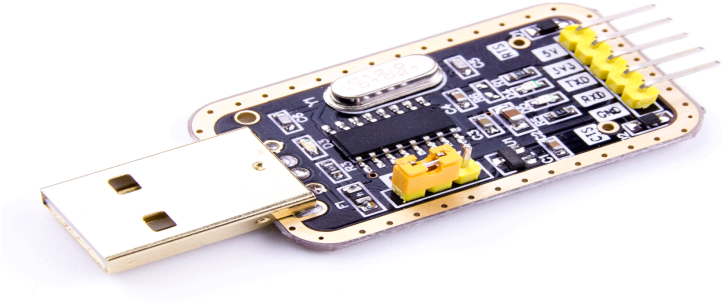
Пароль архива (при необходимости) – **ЭиМТ**

STM32F103C8T6 (ARDUINO IDE, KEIL + CUBEMX)

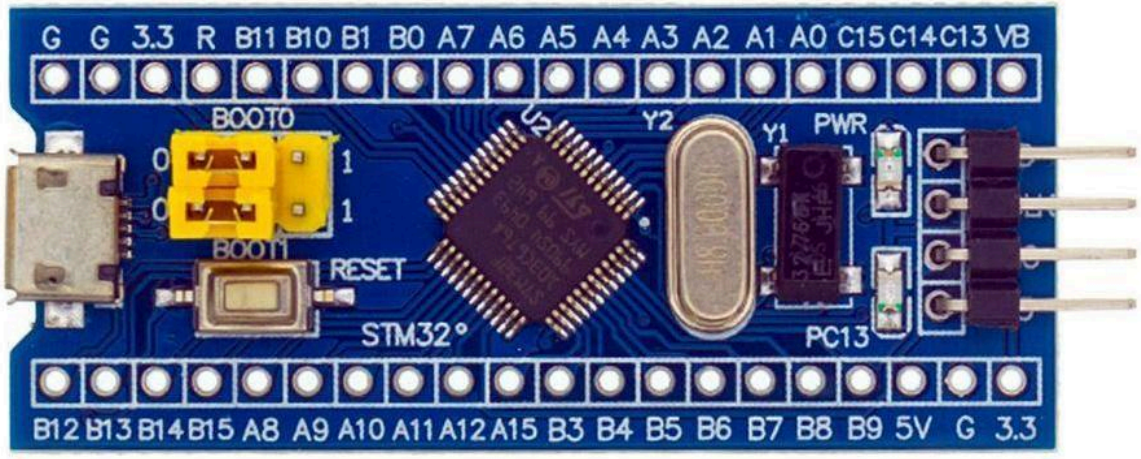


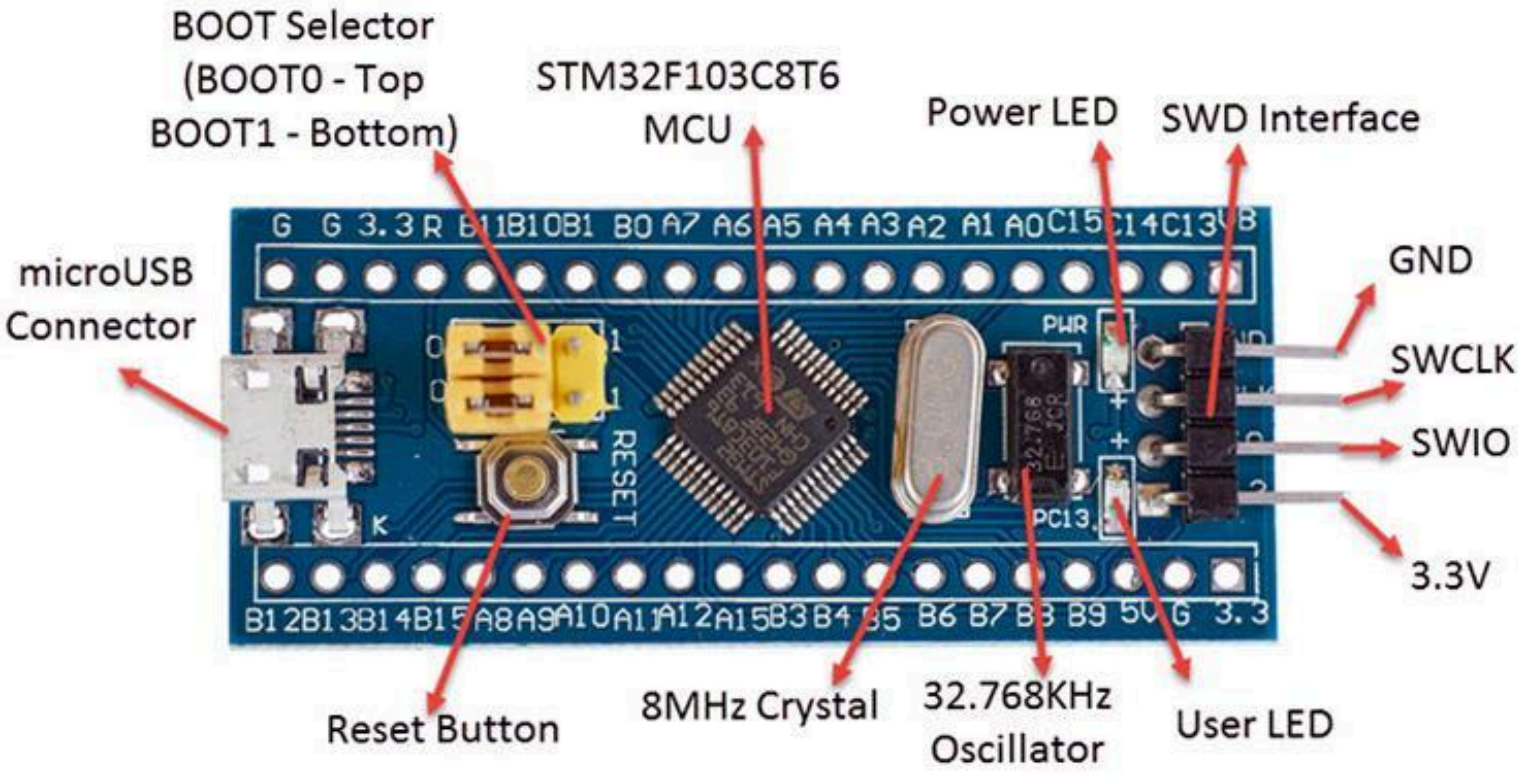
Программатор с отладочным интерфейсом

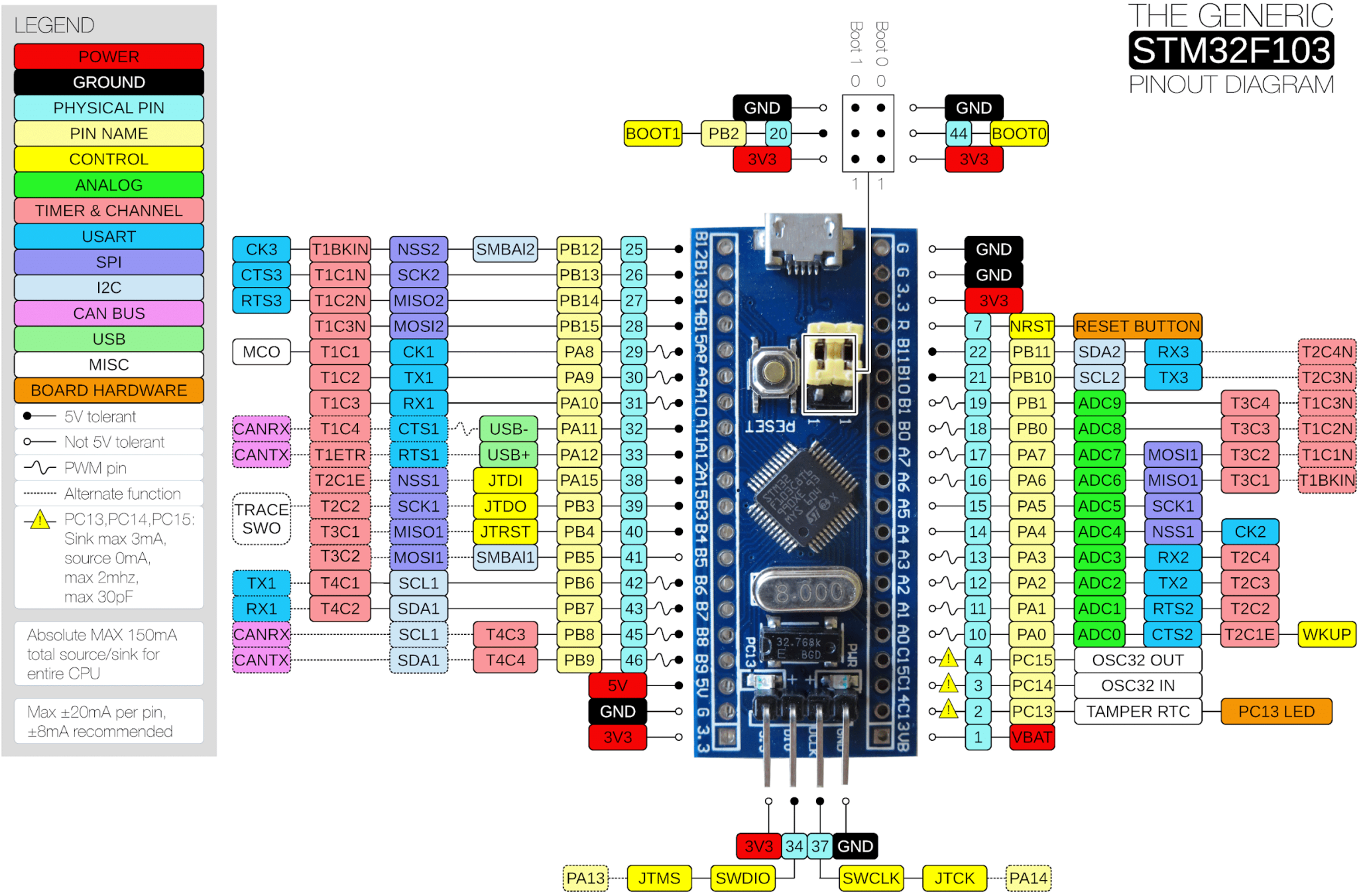
Переходник USB-UART

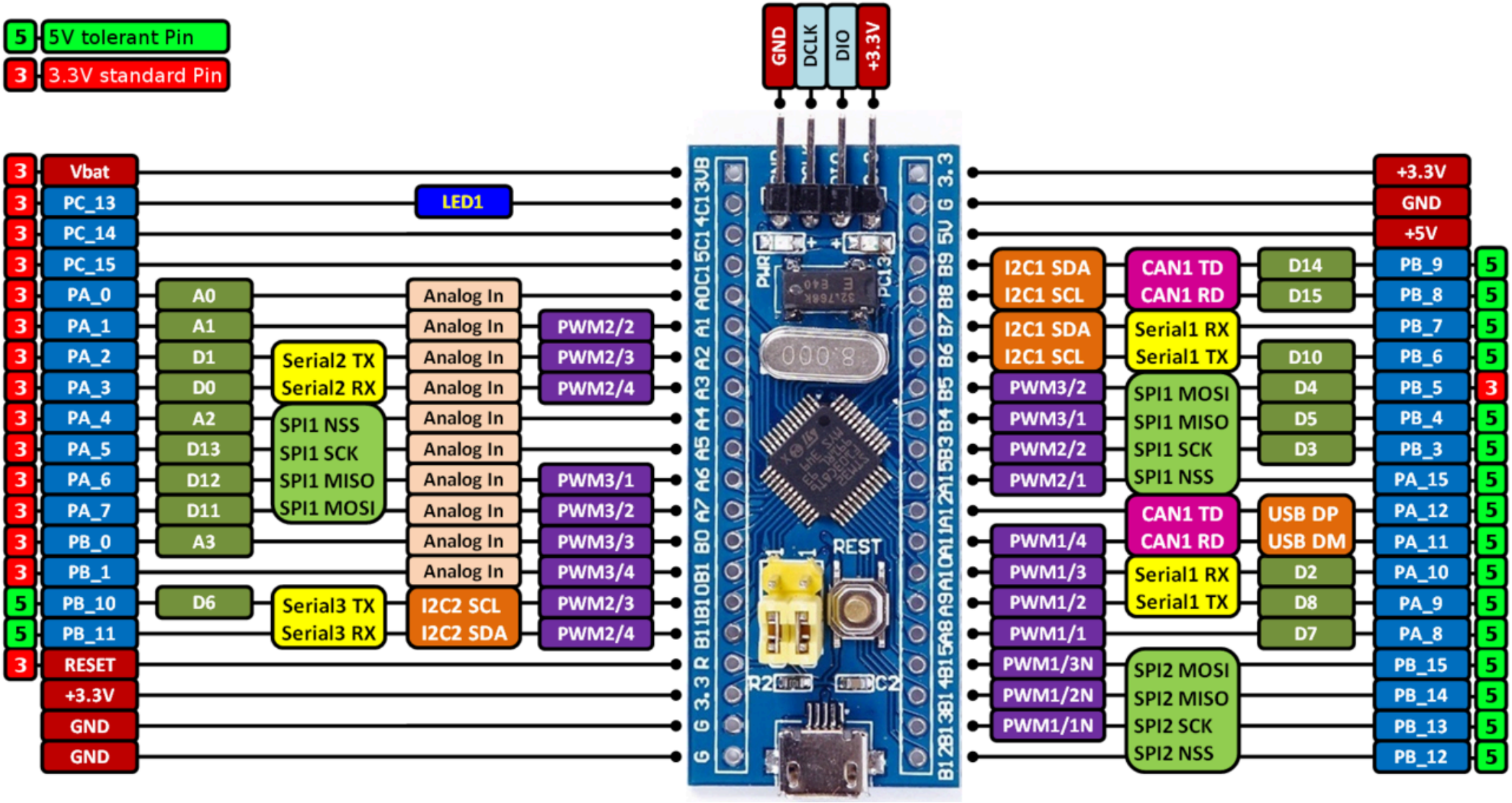


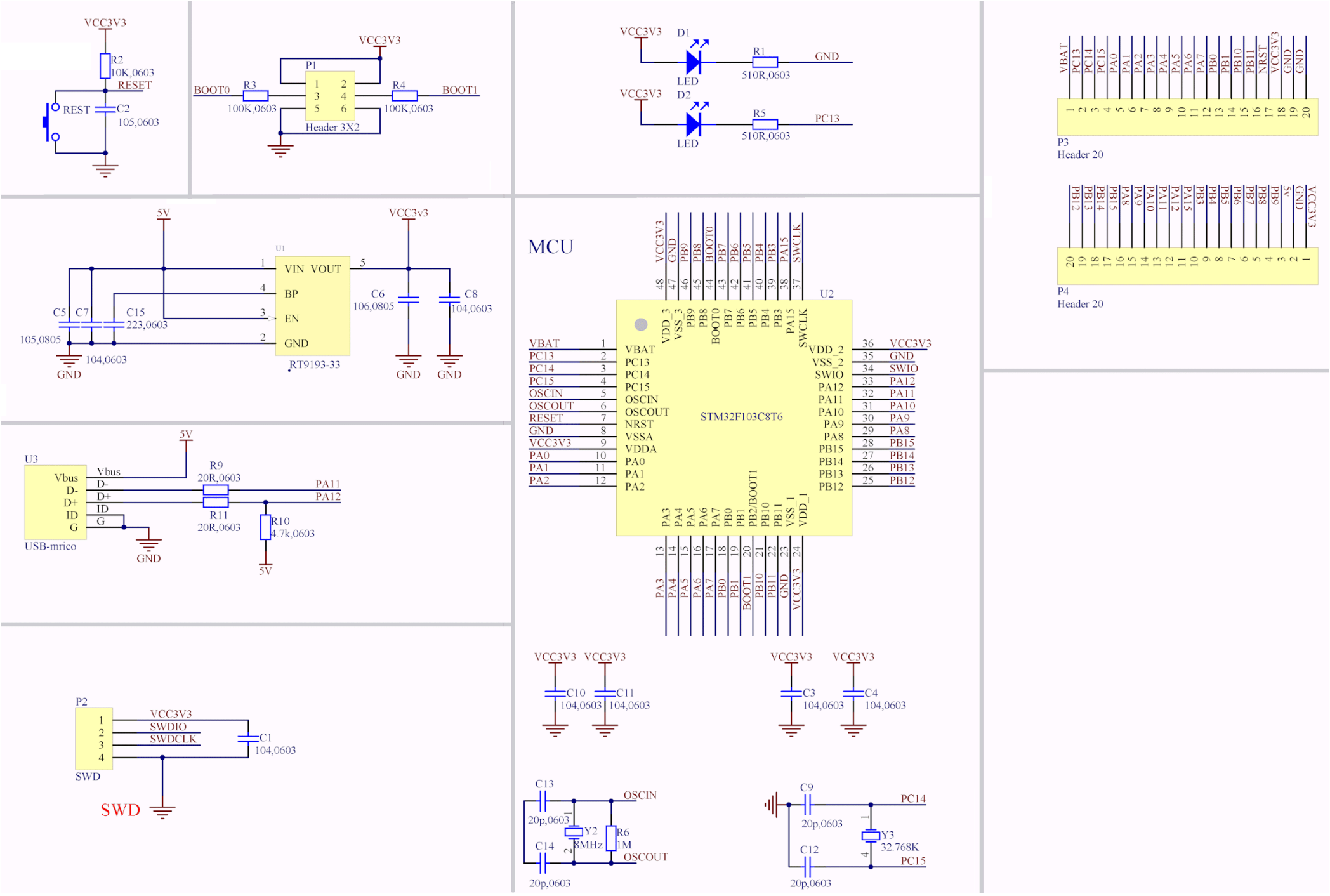
iarduino.ru



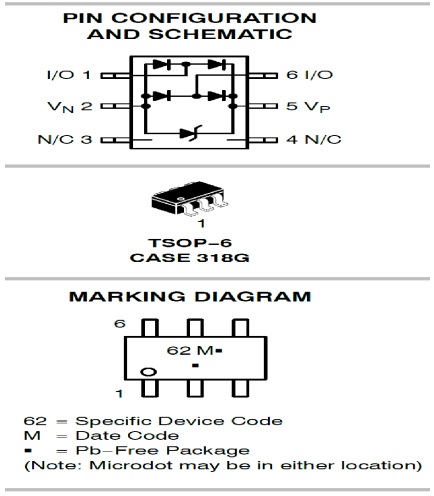

BOOT Selector
(BOOT0 - Top
BOOT1 - Bottom)
STM32F103C8T6
MCU
Power LED
SWD Interface
microUSB
Connector
GND
SWCLK
SWIO
3.3V
Reset Button
8MHz Crystal
32.768KHz
Oscillator
User LED

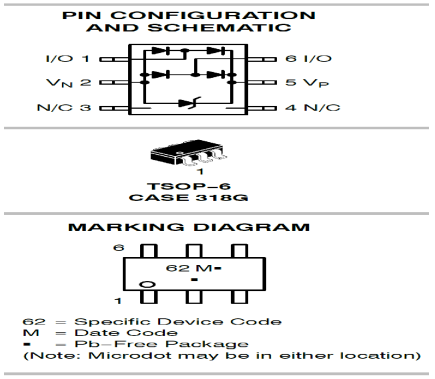

THE GENERIC
STM32F103
PINOUT DIAGRAM
LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
Boot 1
Boot 0
BOOT1
PB2
20
GND
3V3
44
BOOT0
CK3
CTS3
RTS3
MCO
CANRX
CANTX
TRACE SWO
TX1
RX1
T1BKIN
T1C1N
T1C2N
T1C3N
T1C1
T1C2
T1C3
T1C4
T1ETR
T2C1E
T2C2
T3C1
T3C2
T4C1
T4C2
NSS2
SCK2
MISO2
MOSI2
CK1
TX1
RX1
CTS1
RTS1
NSS1
SCK1
MISO1
MOSI1
SCL1
SDA1
SMBAI2
USB-
USB+
JTDI
JTDO
JTRST
SMBAI1
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PA11
PA12
PA15
PB3
PB4
PB5
PB6
PB7
PB8
PB9
25
26
27
28
29
30
31
32
33
38
39
40
41
42
43
45
46
5V
7
22
21
19
18
17
16
15
14
13
12
11
10
4
3
2
1
NRST
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
VBAT
RESET BUTTON
SDA2
SCL2
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
OSC32 OUT
OSC32 IN
TAMPER RTC
RX3
TX3
MOSI1
MISO1
SCK1
NSS1
RX2
TX2
RTS2
CTS2
T3C4
T3C3
T3C2
T3C1
CK2
T2C4
T2C3
T2C2
T2C1E
WKUP
PC13 LED
T2C4N
T2C3N
T1C3N
T1C2N
T1C1N
T1BKIN
34
37
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14


5 5V tolerant Pin
3 3.3V standard Pin
GND
DCLK
DIO
+3.3V
3 Vbat
3 PC_13
3 PC_14
3 PC_15
3 PA_0
3 PA_1
3 PA_2
3 PA_3
3 PA_4
3 PA_5
3 PA_6
3 PA_7
3 PB_0
3 PB_1
5 PB_10
5 PB_11
3 RESET
+3.3V
GND
GND
LED1
A0
A1
D1
D0
A2
D13
D12
D11
A3
D6
Serial2 TX
Serial2 RX
SPI1 NSS
SPI1 SCK
SPI1 MISO
SPI1 MOSI
Serial3 TX
Serial3 RX
Analog In
I2C2 SCL
I2C2 SDA
PWM2/2
PWM2/3
PWM2/4
PWM3/1
PWM3/2
PWM3/3
PWM3/4
PWM2/3
PWM2/4
REST
+3.3V
GND
+5V
I2C1 SDA
I2C1 SCL
CAN1 TD
CAN1 RD
D14
D15
PB_9 5
PB_8 5
I2C1 SDA
I2C1 SCL
Serial1 RX
Serial1 TX
D10
PB_7 5
PB_6 5
PWM3/2
PWM3/1
PWM2/2
PWM2/1
SPI1 MOSI
SPI1 MISO
SPI1 SCK
SPI1 NSS
D4
D5
D3
PB_5 3
PB_4 5
PB_3 5
PA_15 5
CAN1 TD
CAN1 RD
USB DP
USB DM
PA_12 5
PA_11 5
PWM1/4
PWM1/3
PWM1/2
PWM1/1
PWM1/3N
PWM1/2N
PWM1/1N
Serial1 RX
Serial1 TX
D2
D8
D7
PA_10 5
PA_9 5
PA_8 5
SPI2 MOSI
SPI2 MISO
SPI2 SCK
SPI2 NSS
PB_15 5
PB_14 5
PB_13 5
PB_12 5


VCC3V3
R2
10K,0603
RESET
REST
C2
105,0603
P1
BOOT0
R3
100K,0603
R4
100K,0603
BOOT1
Header 3X2
D1
LED
R1
510R,0603
GND
D2
R5
PC13
VBAT PC13 PC14 PC15 PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7 PB0 PB1 PB10 PB11 NRST VCC3V3 GND GND
P3
Header 20
PB12 PB13 PB14 PB15 PA8 PA9 PA10 PA11 PA12 PA15 PB3 PB4 PB5 PB6 PB7 PB8 PB9 5v GND VCC3V3
P4
Header 20
5V
U1
VIN VOUT
BP
EN
GND
RT9193-33
C5
C7
C15
223,0603
105,0805
104,0603
C6
106,0805
C8
104,0603
VCC3v3
MCU
U2
STM32F103C8T6
VBAT
PC13
PC14
PC15
OSCIN
OSCOUT
RESET
GND
VCC3V3
PA0
PA1
PA2
VDD_2
VSS_2
SWIO
PA12
PA11
PA10
PA9
PA8
PB15
PB14
PB13
PB12
VDD_3
VSS_3
PB9
PB8
BOOT0
PB7
PB6
PB5
PB4
PB3
PA15
SWCLK
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2/BOOT1
PB10
PB11
VSS_1
VDD_1
U3
USB-mrico
Vbus
D-
D+
ID
G
R9
20R,0603
R11
20R,0603
R10
4.7k,0603
PA11
PA12
P2
SWD
SWDIO
SWDCLK
C1
104,0603
C10
C11
C3
C4
C13
20p,0603
Y2
8MHz
R6
1M
C14
OSCIN
OSCOUT
C9
C12
Y3
32.768K
PC14
PC15

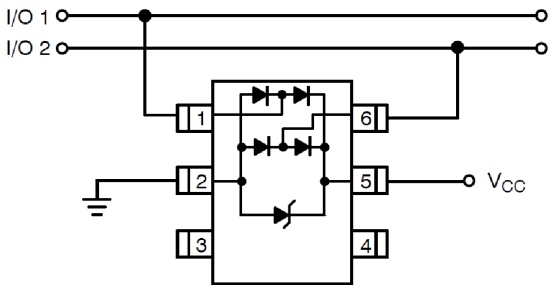
Доработка китайского ST-LINK (Предотвращение пробоя статическим электричеством с помощью саппрессора NUP2201MR6)

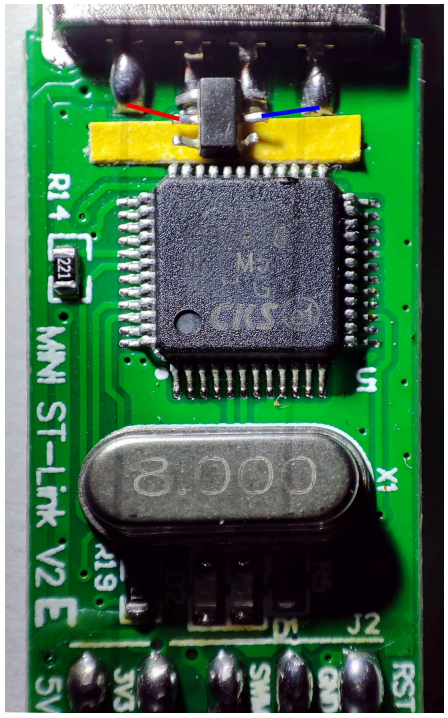


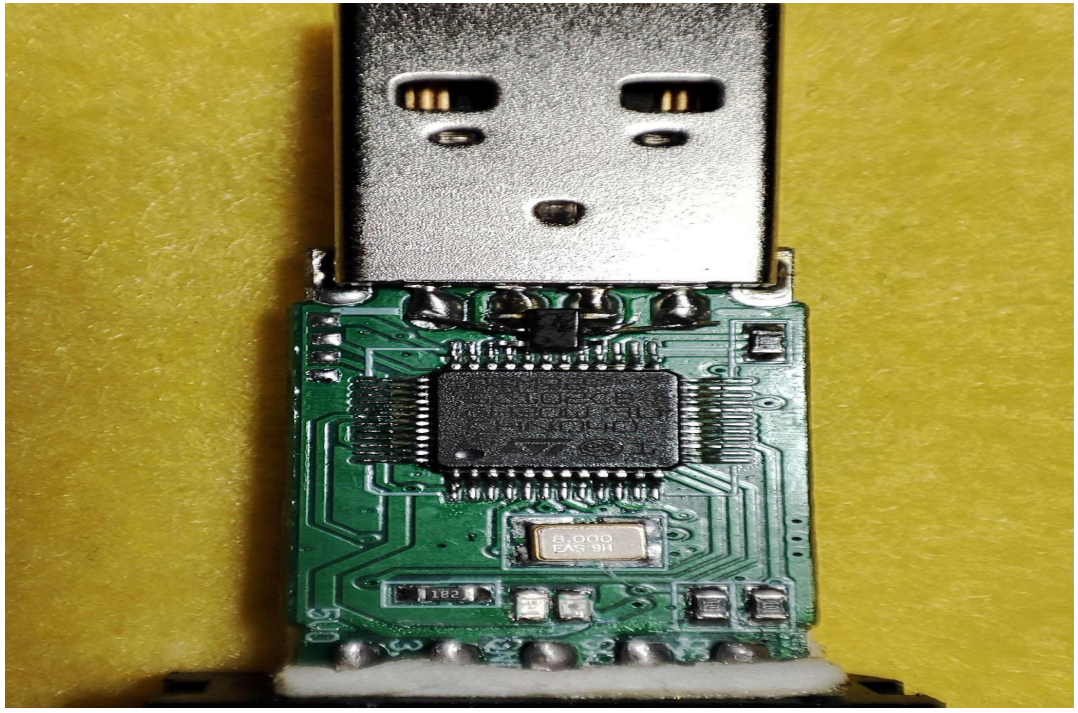


Option 1

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

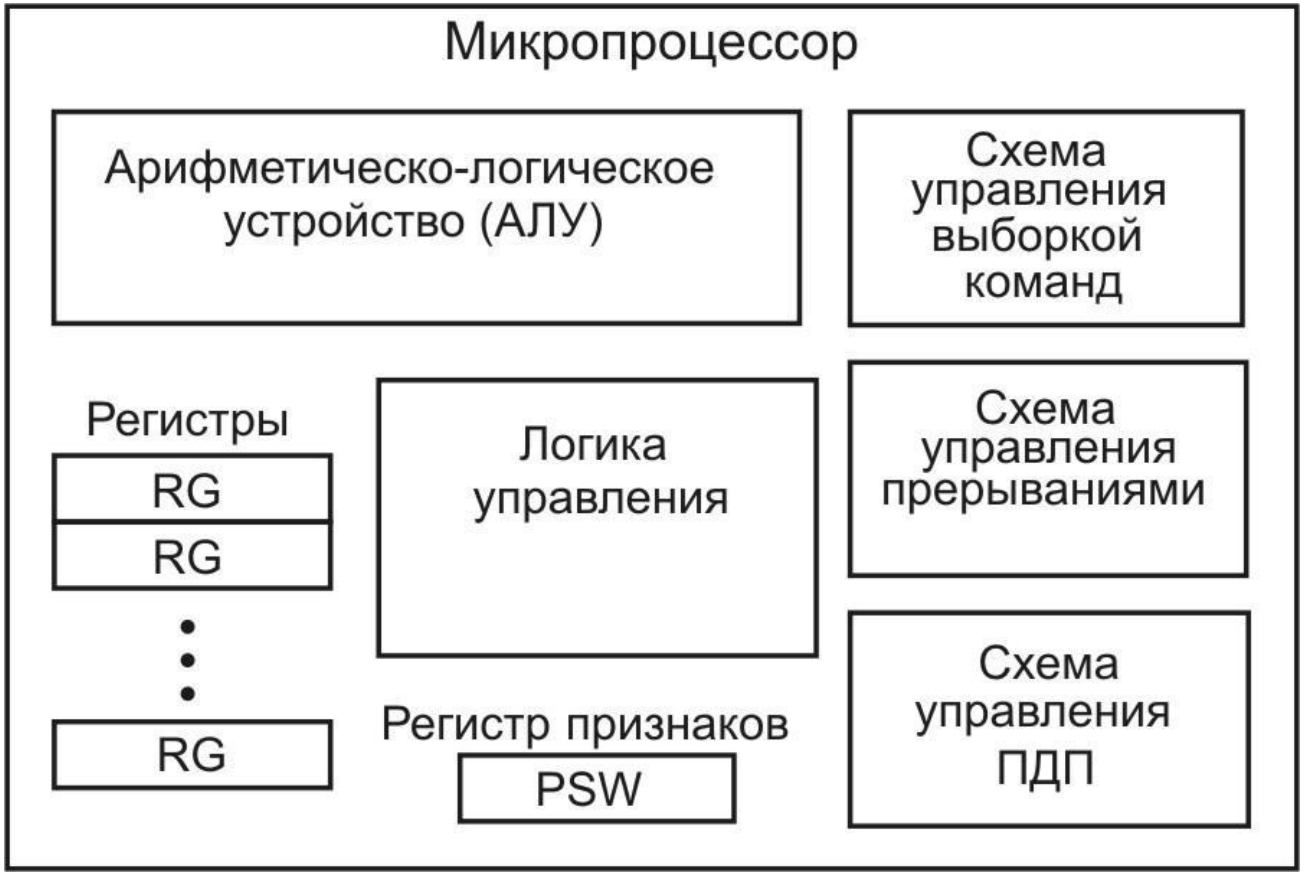
Ссылка для скачивания последней *evaluation version* среды разработки (IDE) программ на языке С для микроконтроллеров с ядром ARM ***Keil MDK***:
https://www.keil.com/demo/eval/arm.htm

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

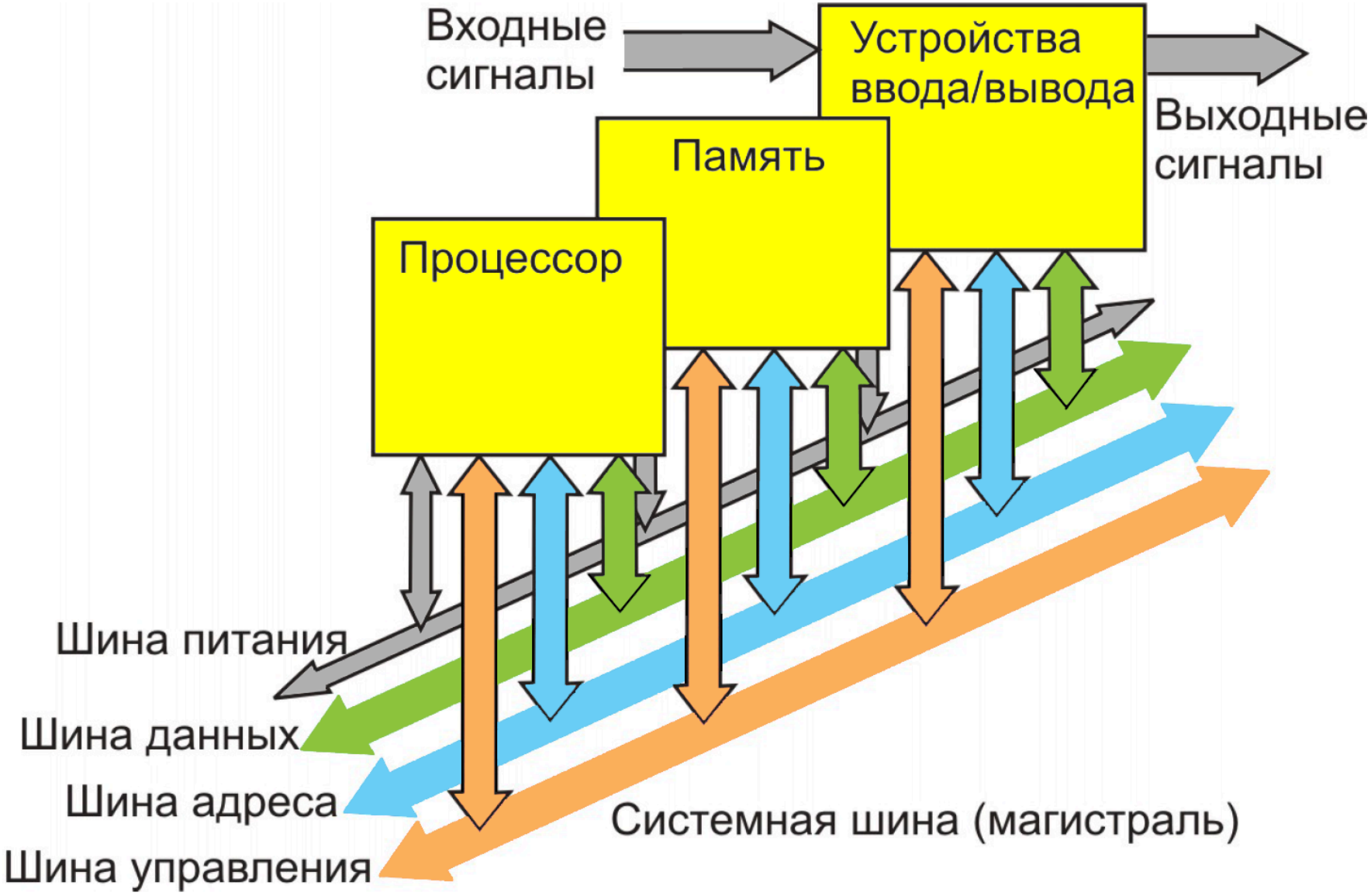


Основные функции ЦП, реализуемые при выполнении программы:

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

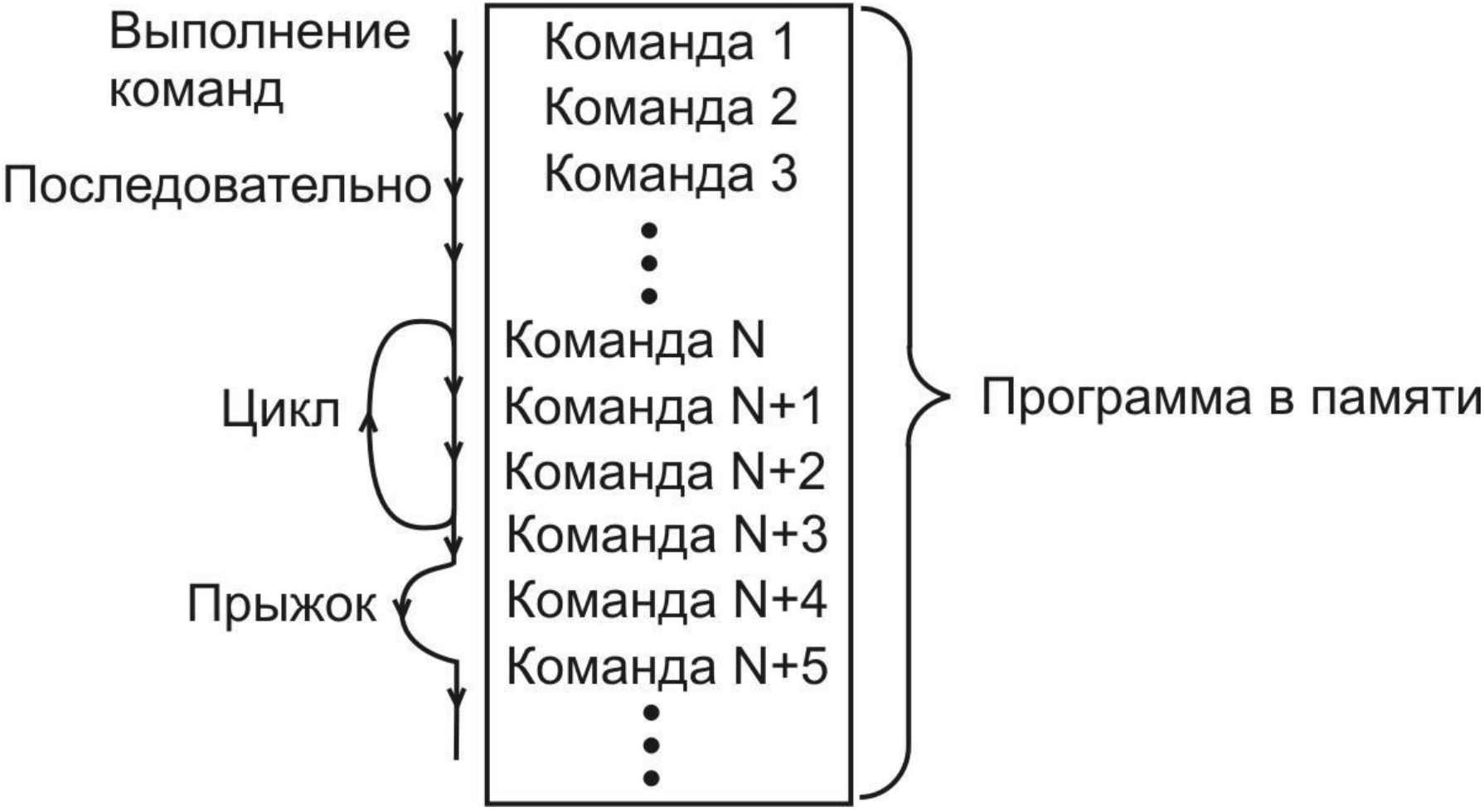
- **Шина адреса** (*Address Bus, AB, ША*) — для пересылки кода адреса (индивидуального номера устройства или ячейки памяти, участвующего в обмене в данный момент). Её разрядность определяет допустимый объем памяти и, следовательно, максимально возможный размер программы и максимально возможный объем запоминаемых данных. Количество адресов, обеспечиваемых шиной адреса, определяется как $\mathbf{2^k}$, где **k** — количество разрядов шины адреса.
- **Шина данных** (*Data Bus, DB, ШД*) — для пересылки данных между устройствами. Двунаправленная 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).

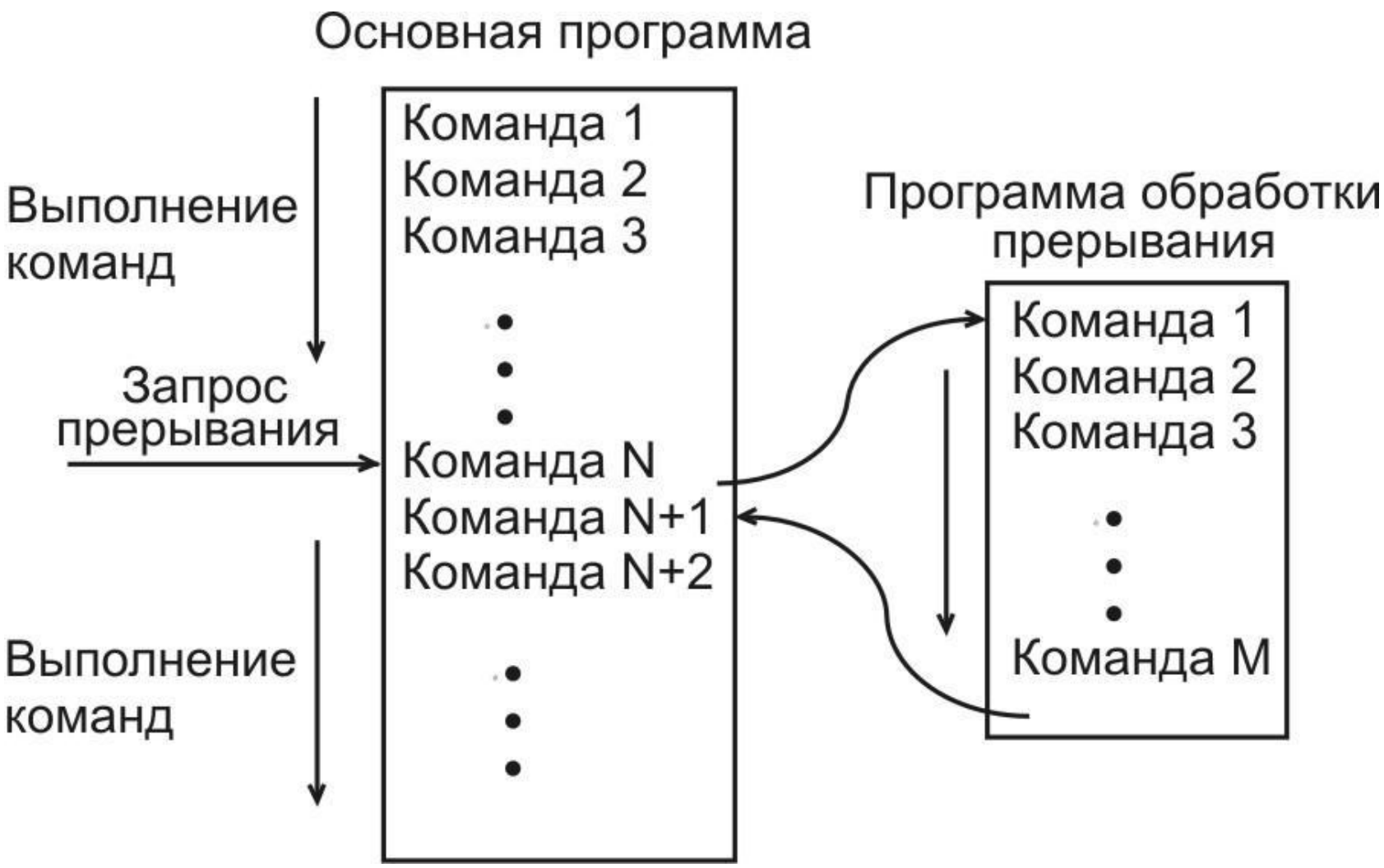
Циклы обмена в микропроцессорной системе

- **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*).

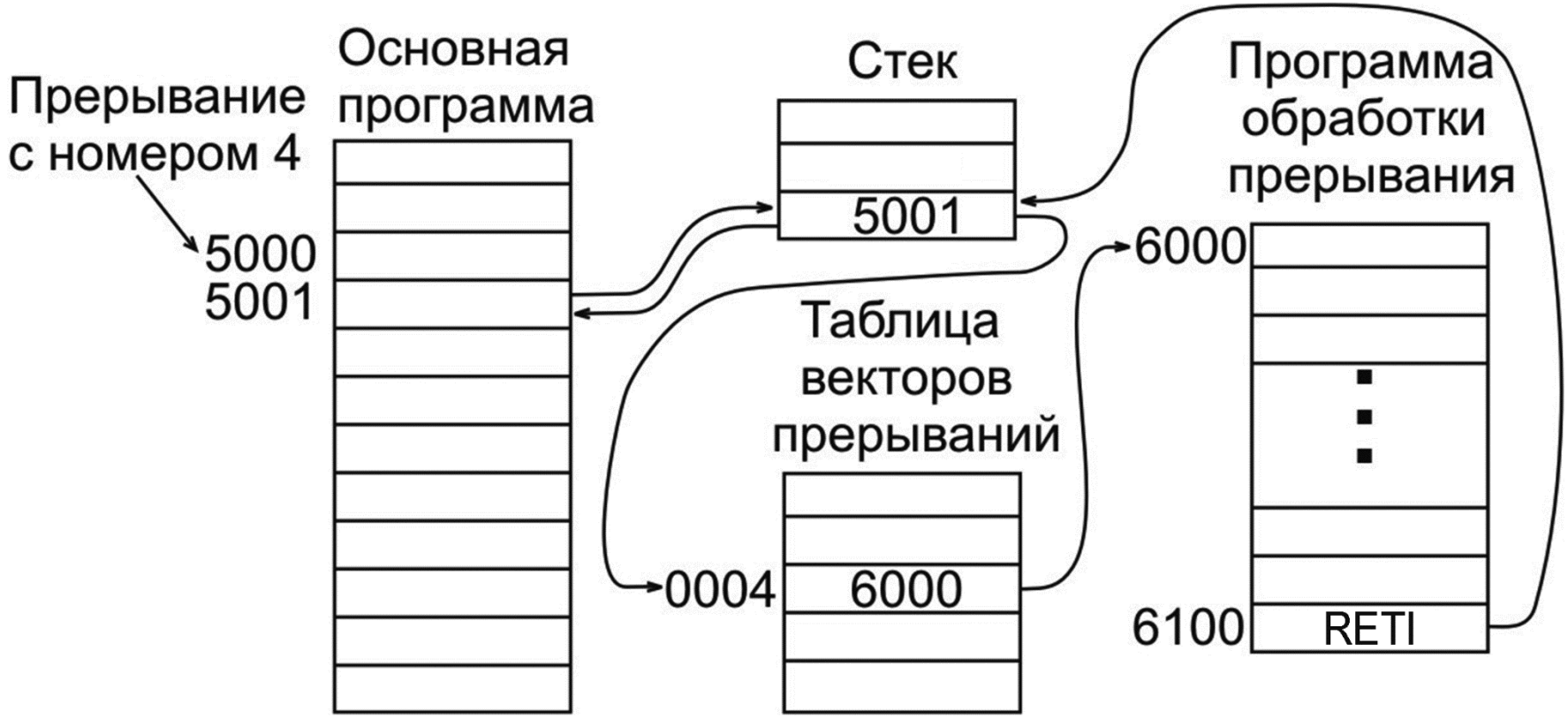
Выборка команд — выполнение программы (иллюстрация программного цикла)



Обслуживание прерывания (Interrupt)



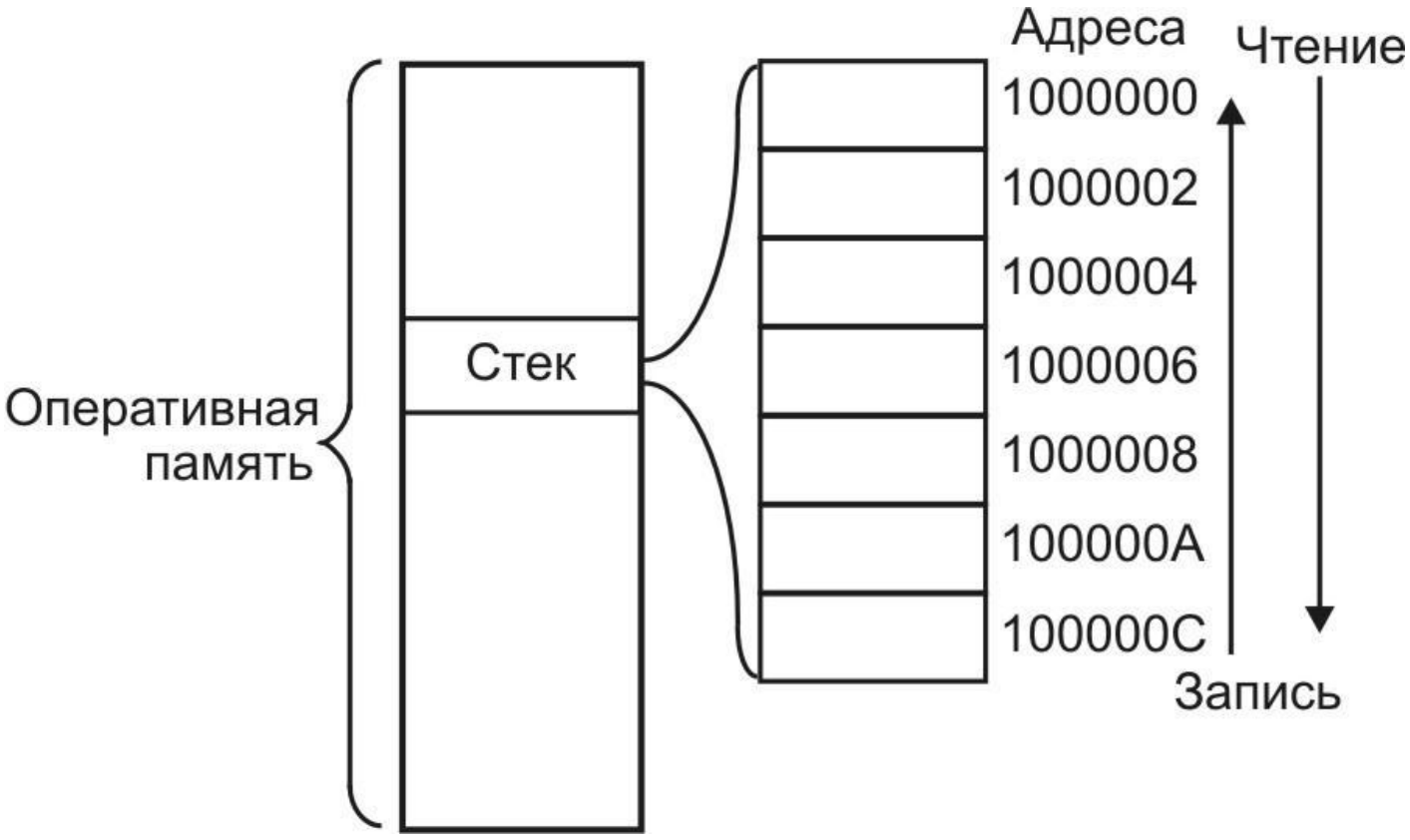
Механизм обработки прерывания (более подробно)



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)

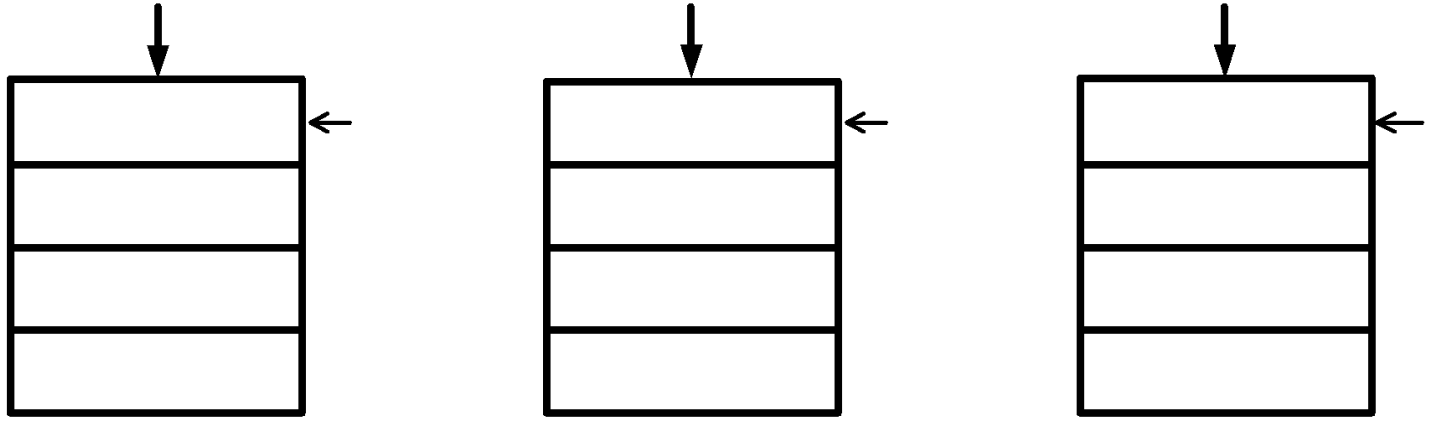


Основной принцип: последним вошел — первым вышел (**Last In First Out — LIFO**. Он реализуется с помощью специального устройства — указателя стека

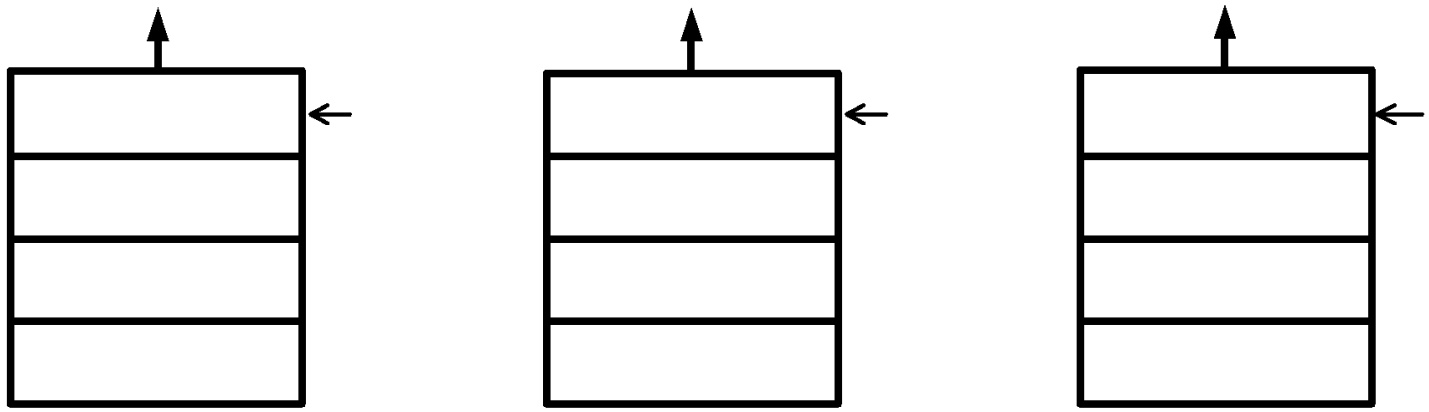
(SP). Основная функция этого устройства — формирование адресов ОЗУ при операциях записи и чтения. Здесь при записи адрес уменьшается SP↓, при чтении адрес увеличивается SP↑.

СТЕК при операциях записи и чтения

Запись производится с *постдекрементом* указателя стека SP

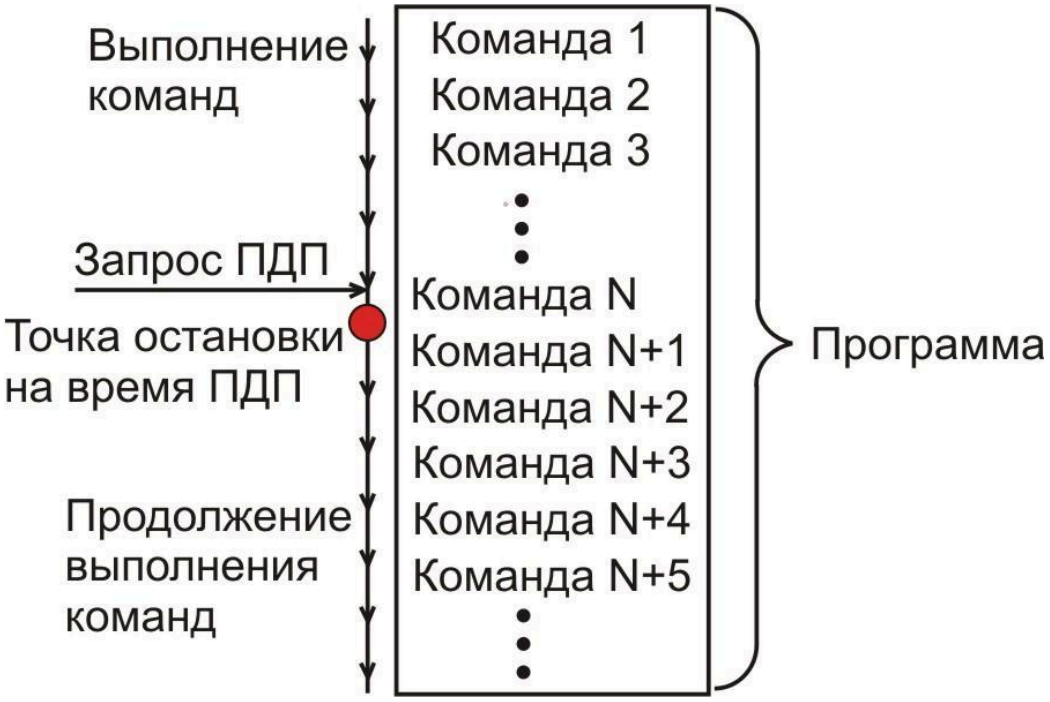


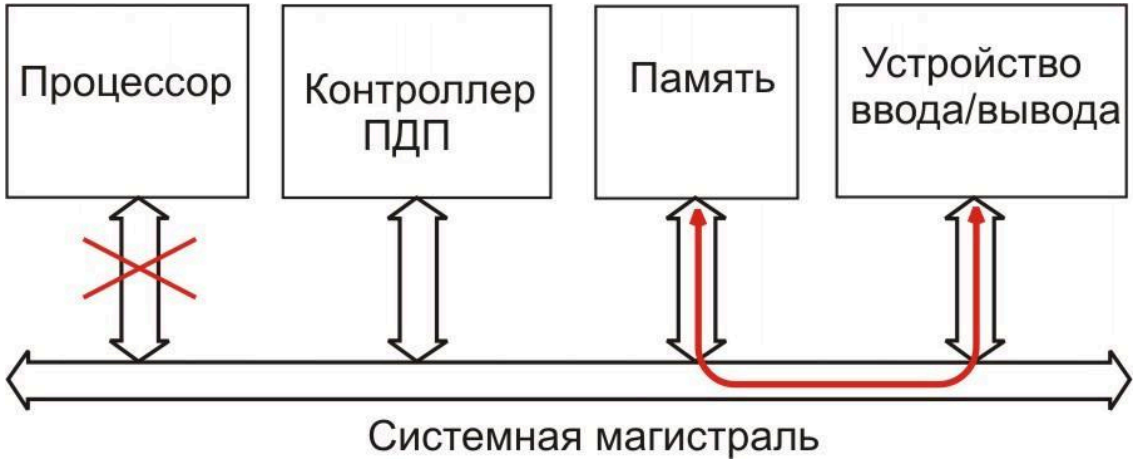
Чтение производится с *прединкрементом* указателя стека SP



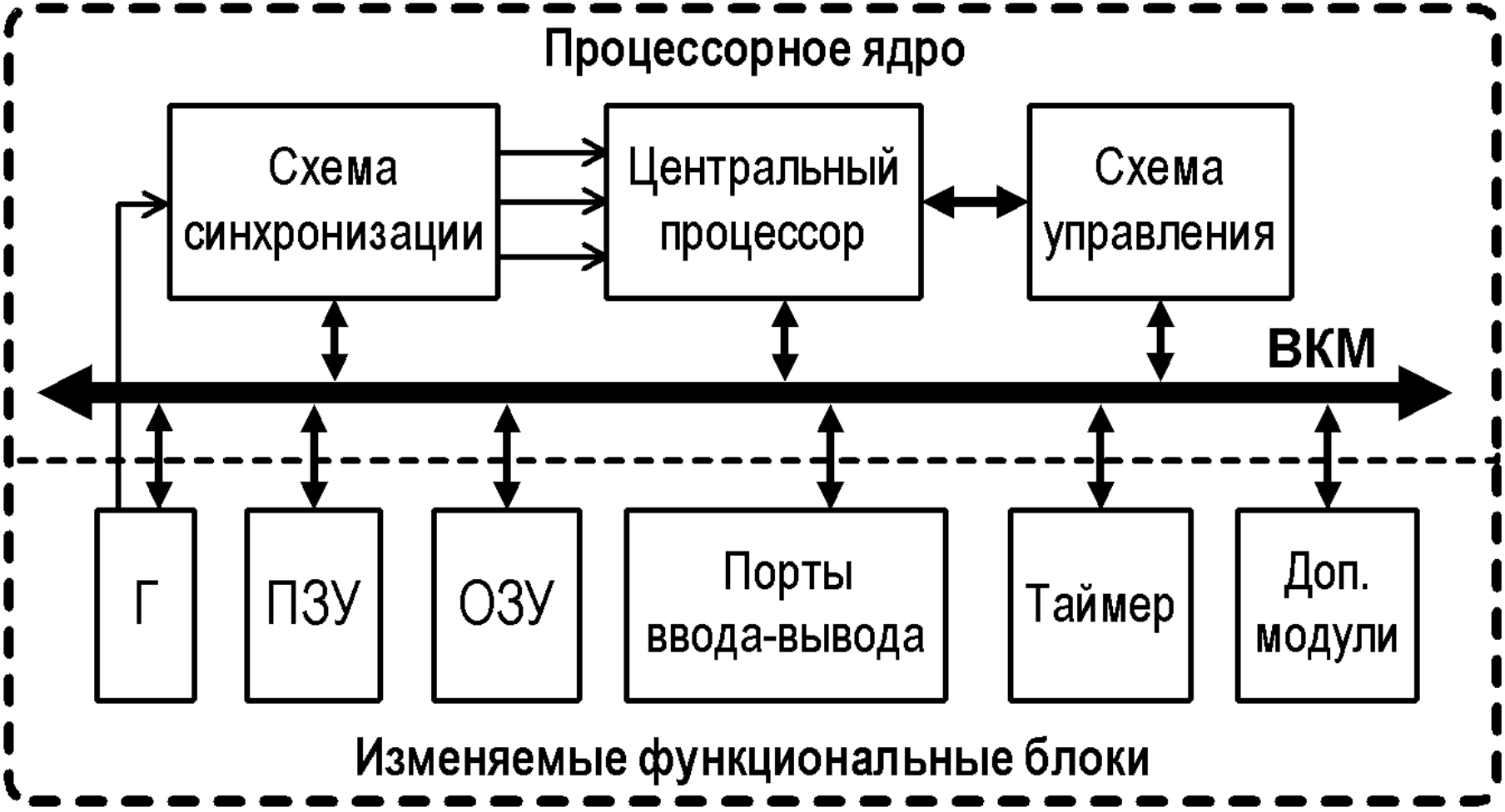
Результат работы со стеком не изменится, если запись будет происходить с *преддекрементом*, а чтение — с *постинкрементом*.

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


Выполнение команд
Запрос ПДП
Точка остановки на время ПДП
Продолжение выполнения команд
Команда 1
Команда 2
Команда 3
Команда N
Команда N+1
Команда N+2
Команда N+3
Команда N+4
Команда N+5
Программа

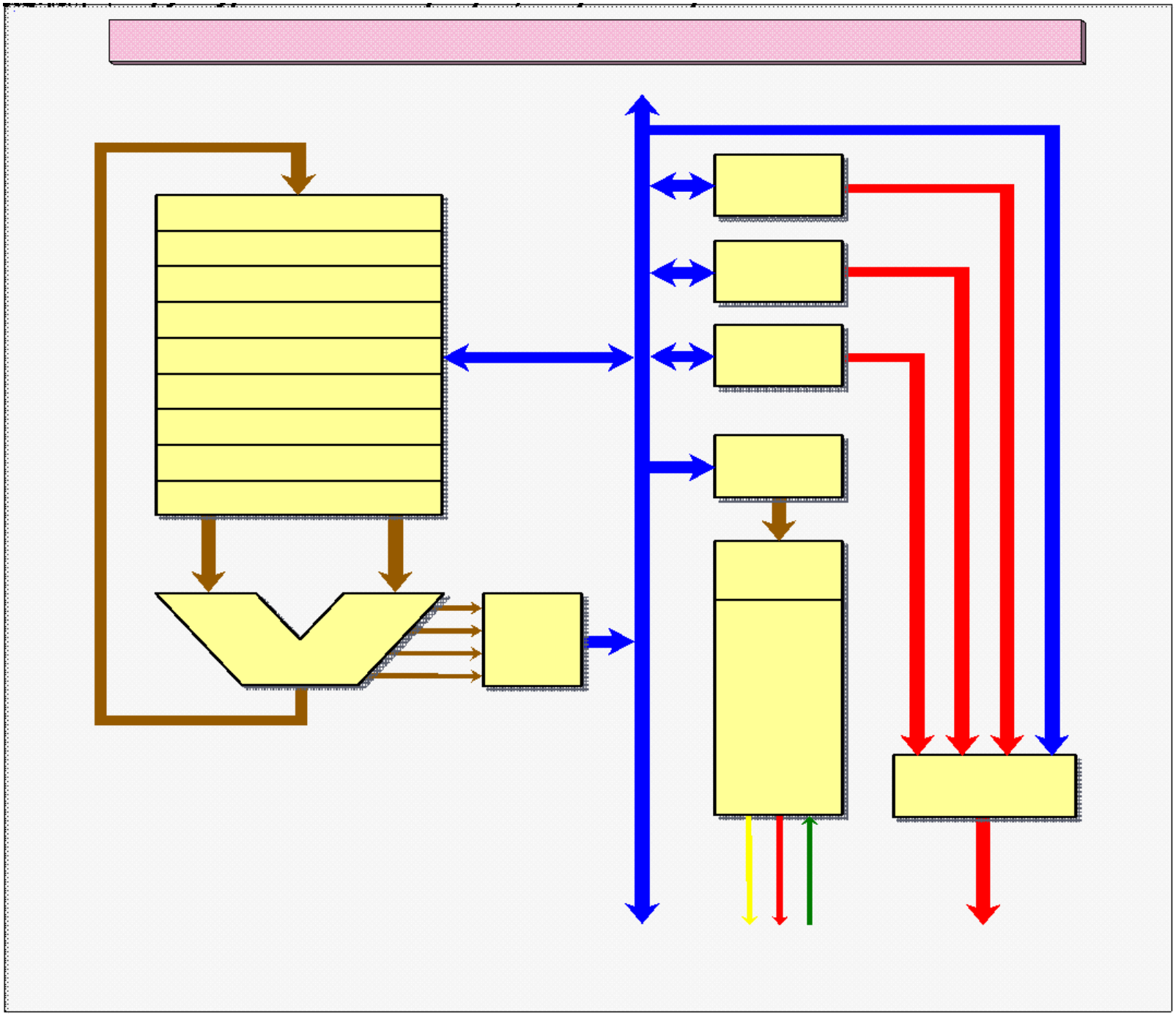

Процессор
Контроллер ПДП
Память
Устройство ввода/вывода
Системная магистраль

СТРУКТУРНАЯ СХЕМА МИКРОКОНТРОЛЛЕРА

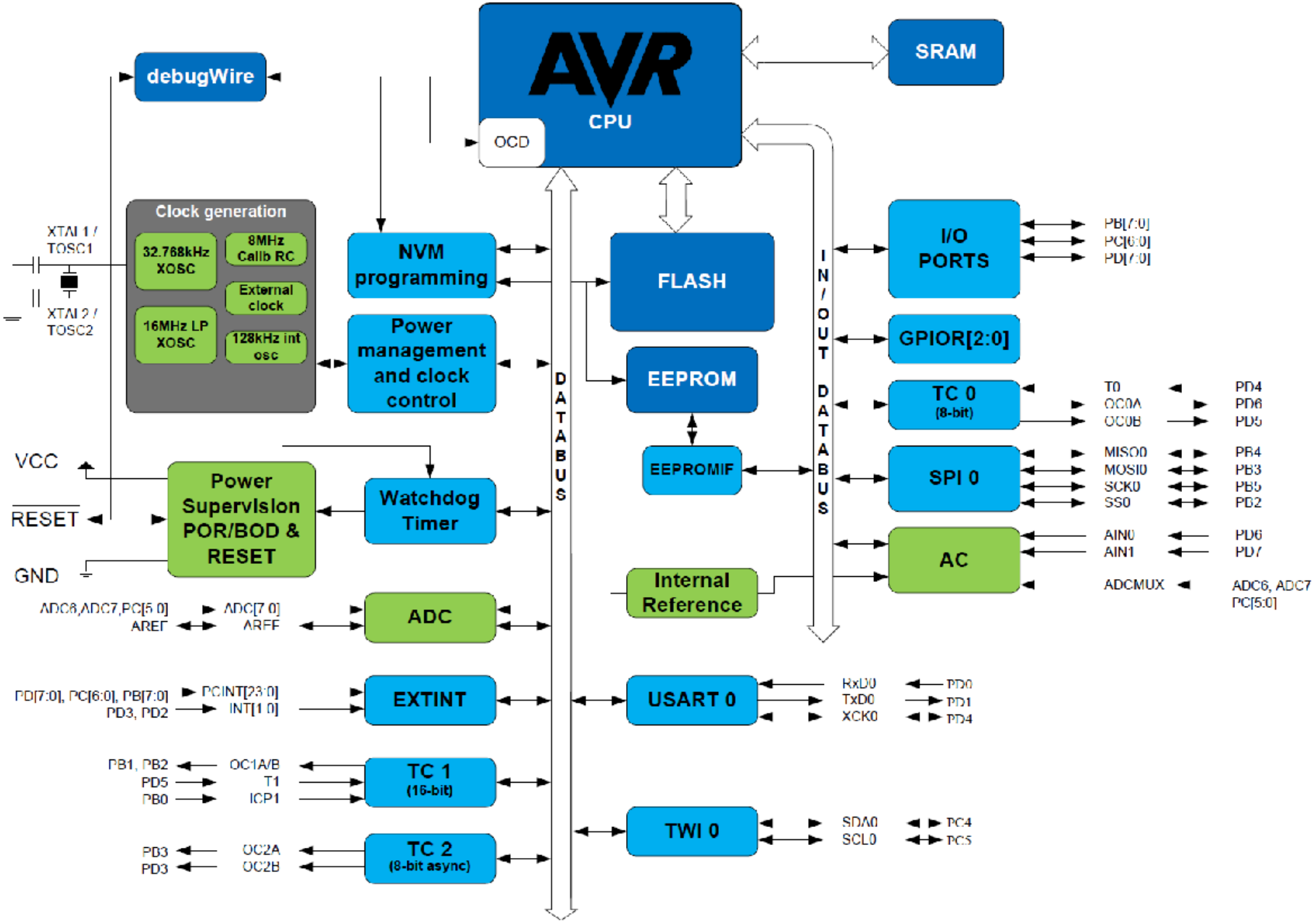


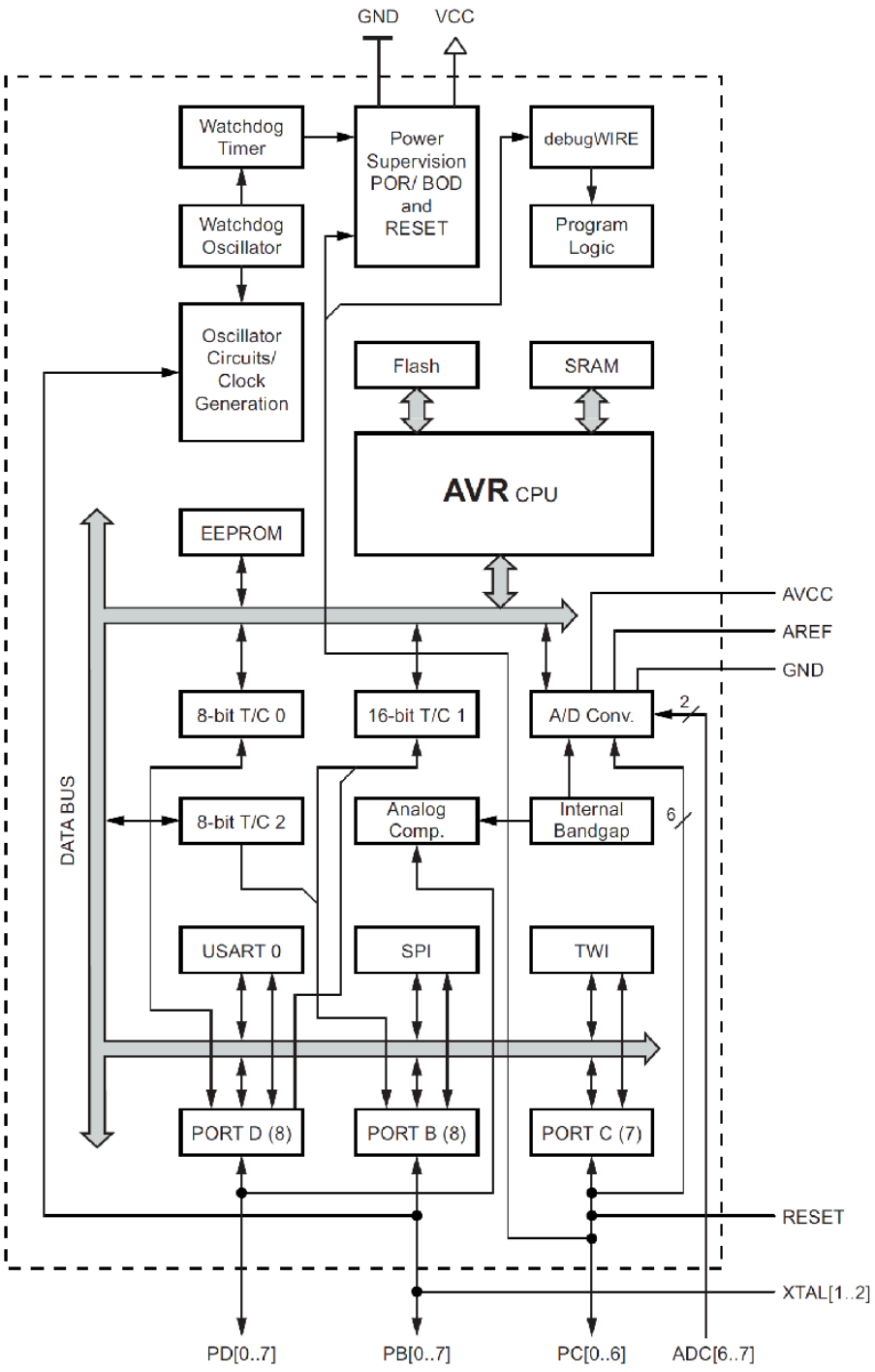
ВКМ — внутренняя контроллерная магистраль
Г – тактовый генератор
ПЗУ – постоянное запоминающее устройство (для хранения программ и констант)

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



Микроконтроллер AVR (Atmega328p)


AVR
CPU
OCD
SRAM
debugWire
Clock generation
32.768kHz XOSC
8MHz Calib RC
External clock
16MHz LP XOSC
128kHz int osc
XTAL1 / TOSC1
XTAL2 / TOSC2
NVM programming
Power management and clock control
FLASH
EEPROM
EEPROMIF
Internal Reference
I/O PORTS
PB[7:0]
PC[6:0]
PD[7:0]
GPIOR[2:0]
TC 0 (8-bit)
T0
OC0A
OC0B
PD4
PD6
PD5
SPI 0
MISO0
MOSI0
SCK0
SS0
PB4
PB3
PB5
PB2
AC
AIN0
AIN1
PD6
PD7
ADCMUX
ADC6, ADC7
PC[5:0]
I/O DATABUS
DATABUS
VCC
RESET
GND
Power Supervision POR/BOD & RESET
Watchdog Timer
ADC
ADC6,ADC7,PC[5:0]
AREF
ADC[7:0]
AREF
EXTINT
PD[7:0], PC[6:0], PB[7:0]
PD3, PD2
PCINT[23:0]
INT[1:0]
USART 0
RxD0
TxD0
XCK0
PD0
PD1
PD4
TC 1 (16-bit)
PB1, PB2
PD5
PB0
OC1A/B
T1
ICP1
TWI 0
SDA0
SCL0
PC4
PC5
TC 2 (8-bit async)
PD3
PD3
OC2A
OC2B


GND
VCC
Watchdog Timer
Power Supervision POR/ BOD and RESET
debugWIRE
Watchdog Oscillator
Program Logic
Oscillator Circuits/ Clock Generation
Flash
SRAM
AVR CPU
EEPROM
AVCC
AREF
GND
8-bit T/C 0
16-bit T/C 1
A/D Conv.
2
DATA BUS
8-bit T/C 2
Analog Comp.
Internal Bandgap
6
USART 0
SPI
TWI
PORT D (8)
PORT B (8)
PORT C (7)
RESET
XTAL[1..2]
PD[0..7]
PB[0..7]
PC[0..6]
ADC[6..7]

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.

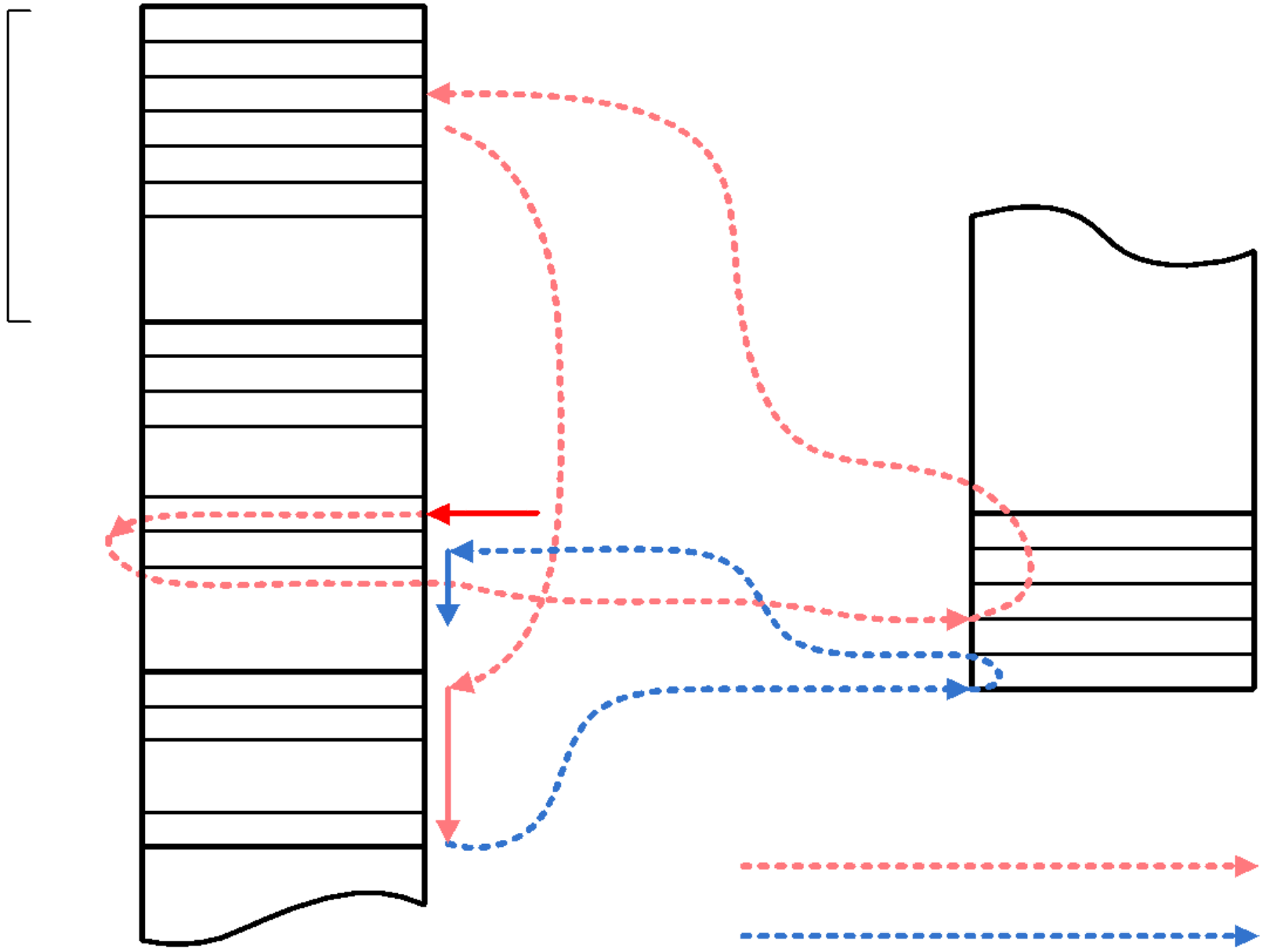
В режиме выхода разряд *PORTX.n* определяет значение выходного сигнала на выводе *PX.n*. При *PORTX.n*=0 выходной сигнал имеет низкий уровень напряжения (U_{OL}), при *PORTX.n*=1 – высокий уровень напряжения (U_{OH}).

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 С

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*:

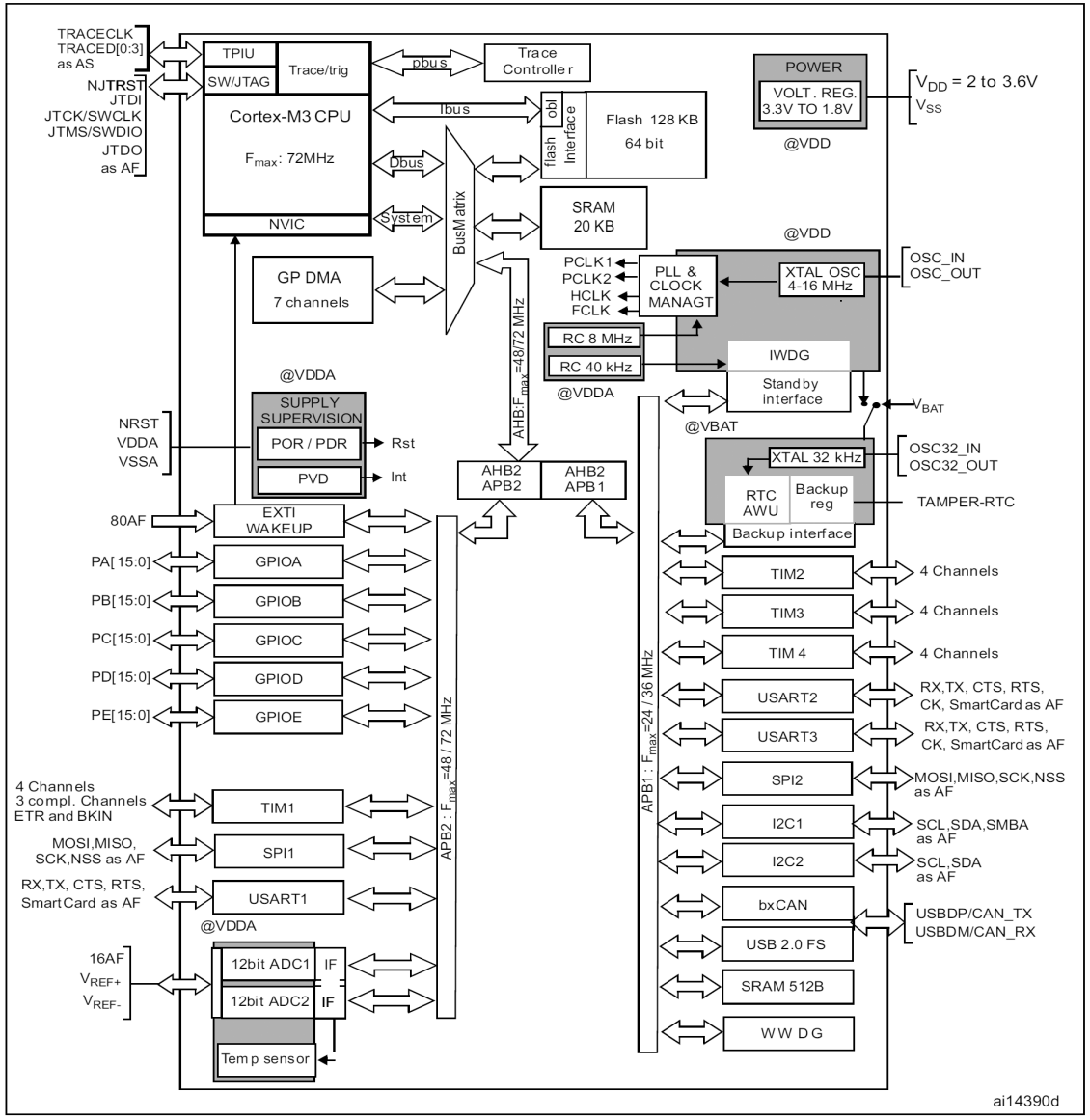
```
while (~n&(1<<7)); while ((n&(1<<7))==0); while (!(n&(1<<7)));
```

Ожидание появления логического 0 в заданном разряде (3):

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

Микроконтроллер STM32F103XX

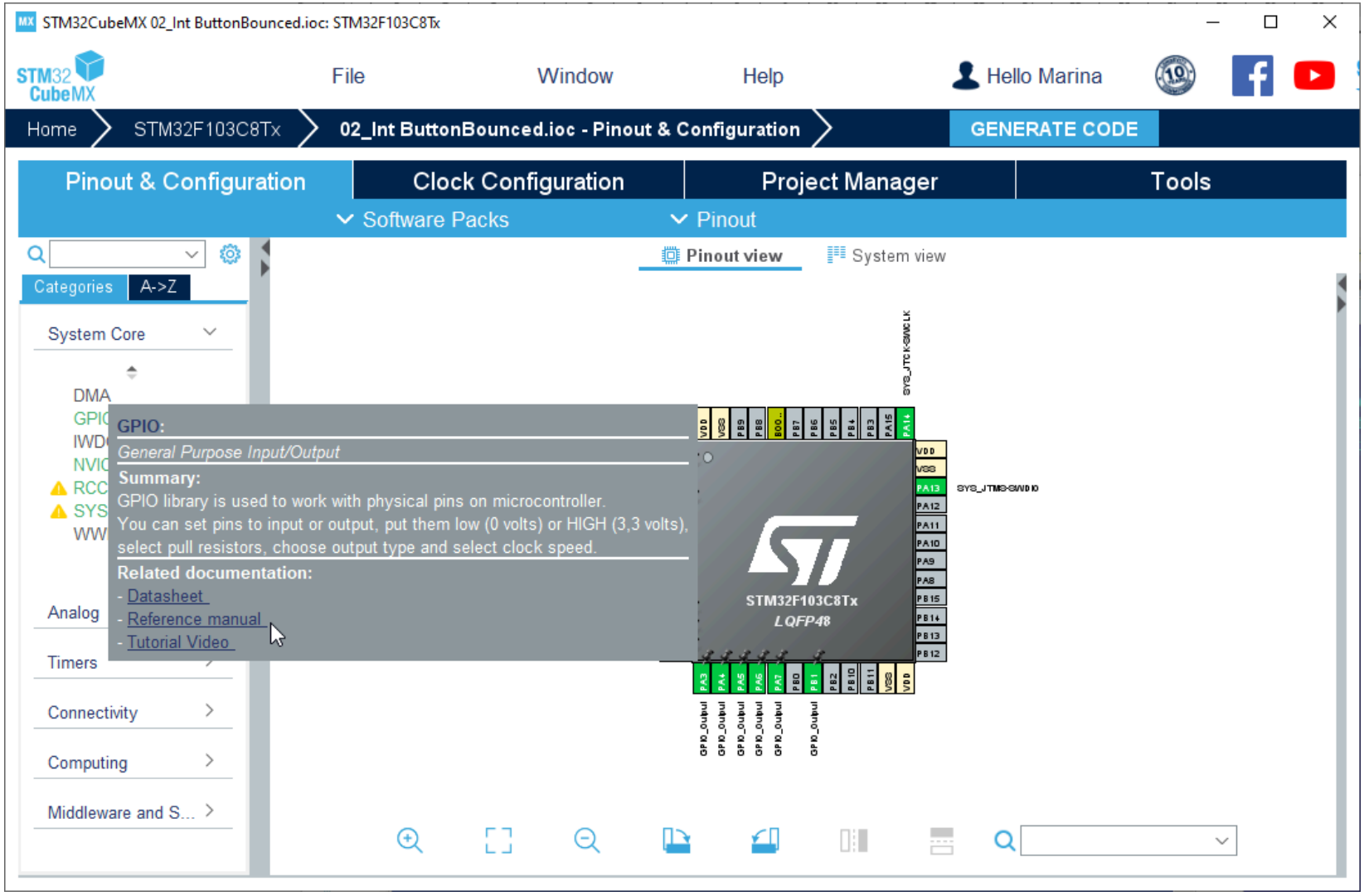
Figure 1. STM32F103xx performance line block diagram



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

AF – альтернативная функция линии ввода-вывода

Информация по MCU STM32F103C8T6



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

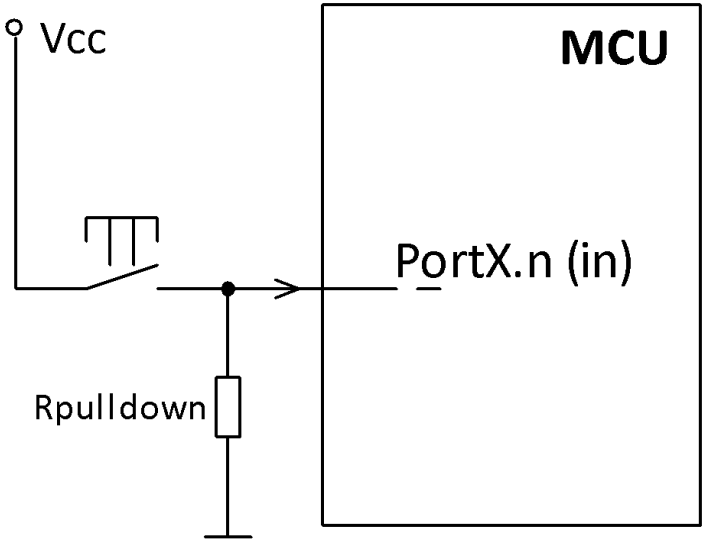
```
HAL_ADCEx_Calibration_Start(&hadc1);    //калибровка АЦП1
HAL_ADC_Start(&hadc1); //запуск АЦП1 на преобразование
HAL_ADC_Start_IT(&hadc1);  //запуск АЦП1 на преобразование с прерываниями
HAL_ADC_Start_DMA(&hadc1,(uint32_t*)&adc,10);  //старт АЦП1 в режиме DMA
с передачей в массив adc[10] 10 слов 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
```

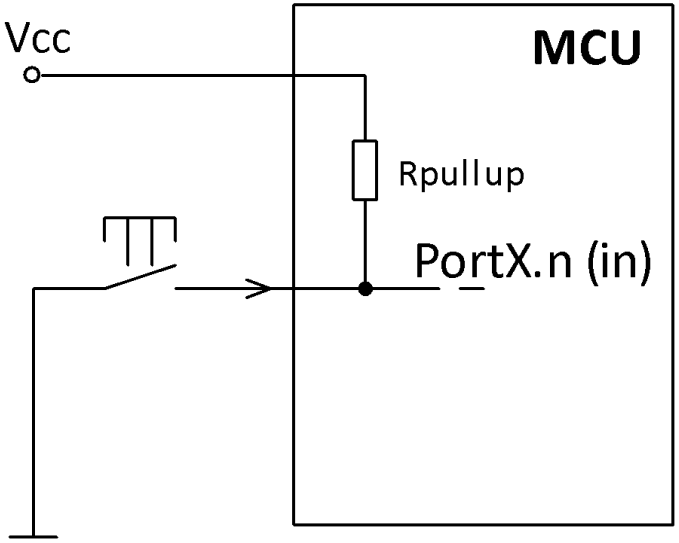
ПОДКЛЮЧЕНИЕ КНОПОК К ПОРТАМ ВВОДА

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

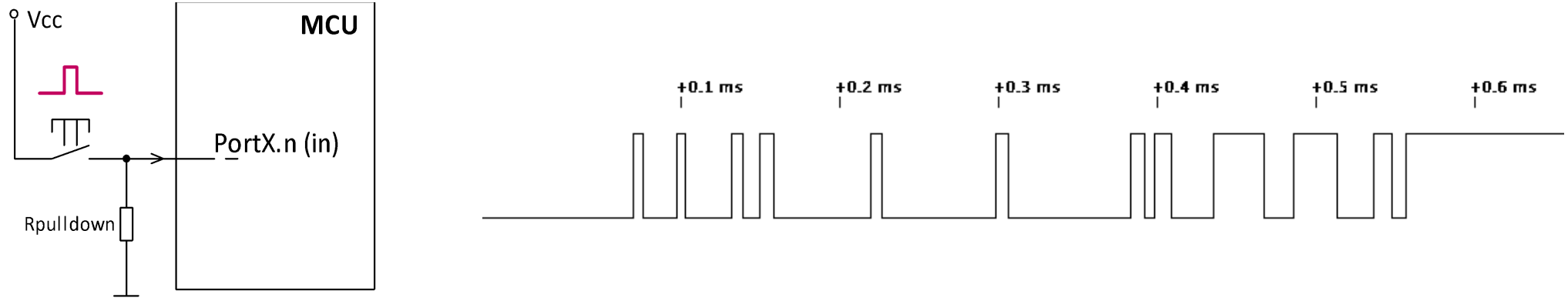


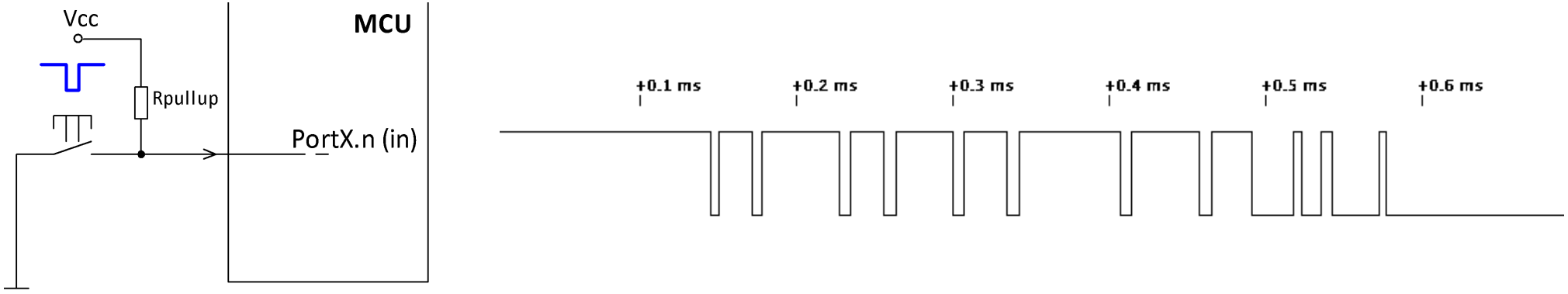
Для формирования высокого уровня при отпущенной кнопке используется подтягивающий к высокому уровню резистор ***Rpullup***, в качестве которого может выступать

	и внутренний подтягивающий резистор линии порта ввода.

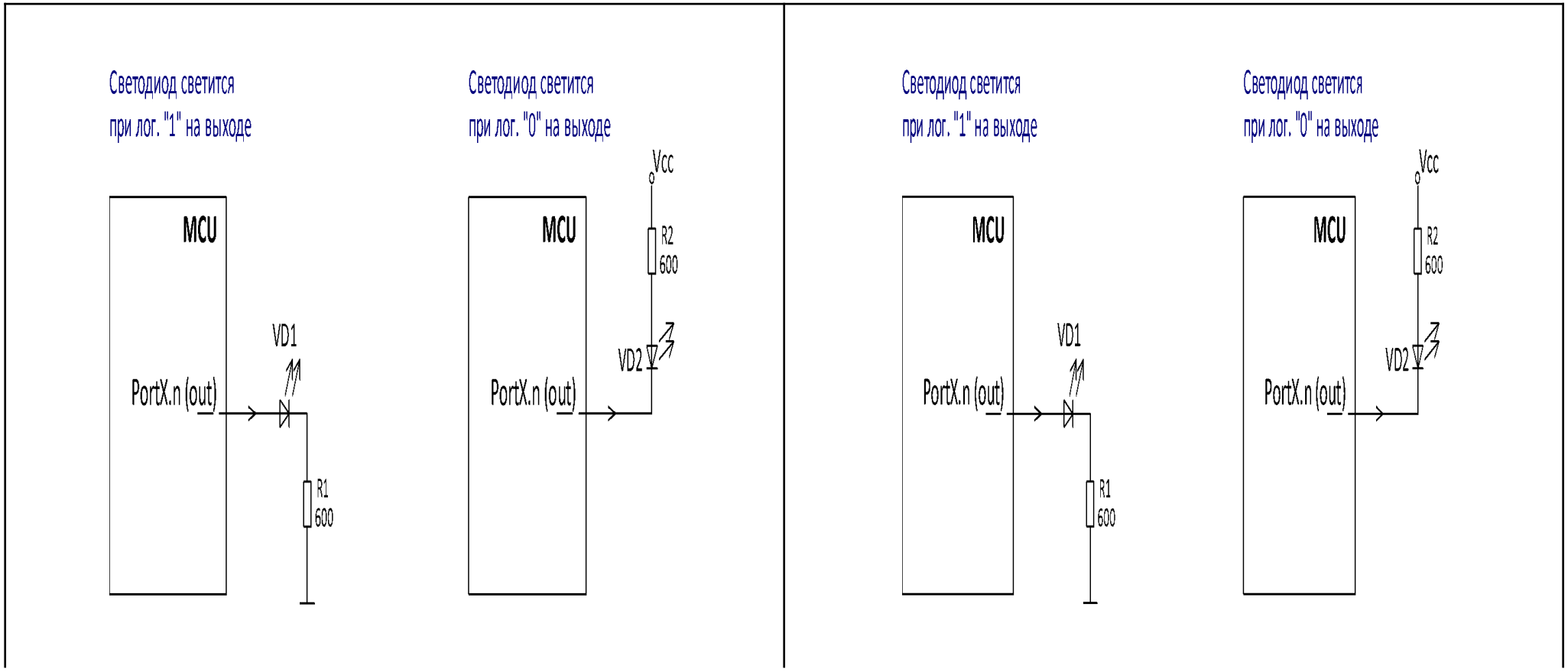
В микроконтроллерах **Atmega** (в том числе и Atmega328p) для портов ввода существует возможность подключения внутреннего подтягивающего к питанию резистора *Rpullup* (см. , справа).
В микроконтроллерах **STM32** для портов в режиме ввода существует возможность подключения внутренних подтягивающего как к + питания *Rpullup*, так и к выводу «общий» (земля) *Rpulldown*.
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 СВЕТОДИОДОВ К ПОРТАМ ВЫВОДА

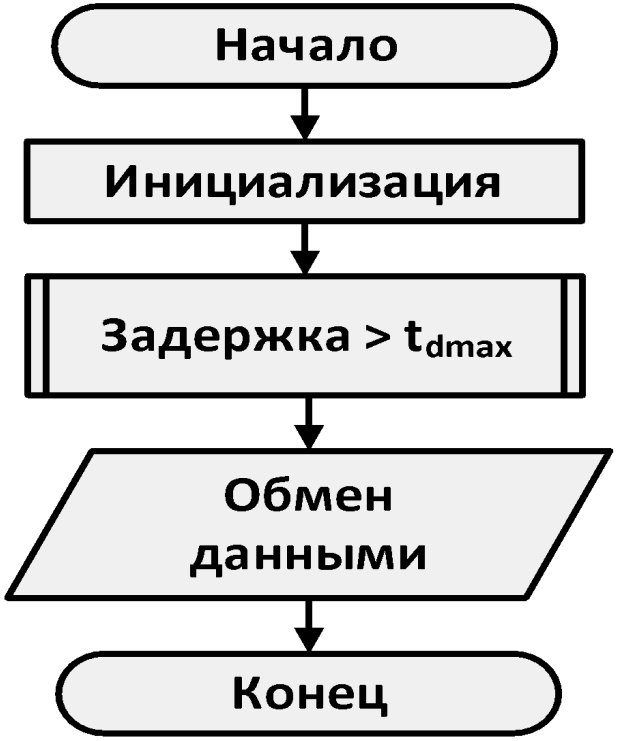


Внутри МК есть токоограничивающий резистор, однако добавочно включают внешний резистор R1 для задания тока: $$I_{LED} \approx \frac{V_{\log 1} - V_{LED}}{R_1}$$	Резистор R2 задает ток через светящийся светодиод: $$I_{LED} \approx \frac{V_{CC} - V_{LED} - V_{\log 0}}{R_2}.$$

СПОСОБЫ ОБМЕНА ЦП С МЕДЛЕННЫМ УВВ

Синхронный способ обмена

В этом случае программа регулирует синхронный обмен с внешним устройством.



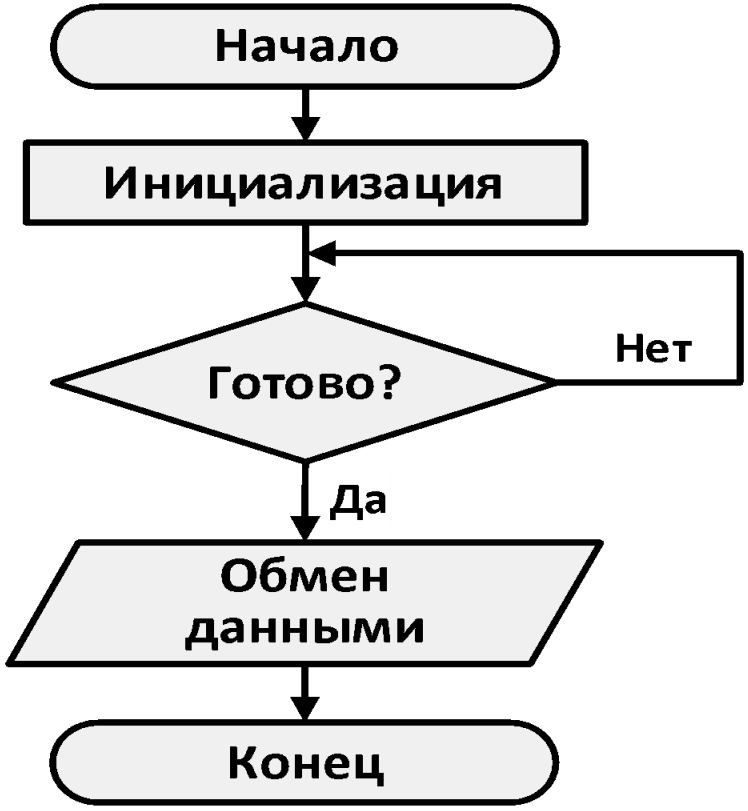
Недостатки:

- Привязка к паспортным данным периферийного устройства

- Потеря времени на создание задержек

Асинхронный способ обмена

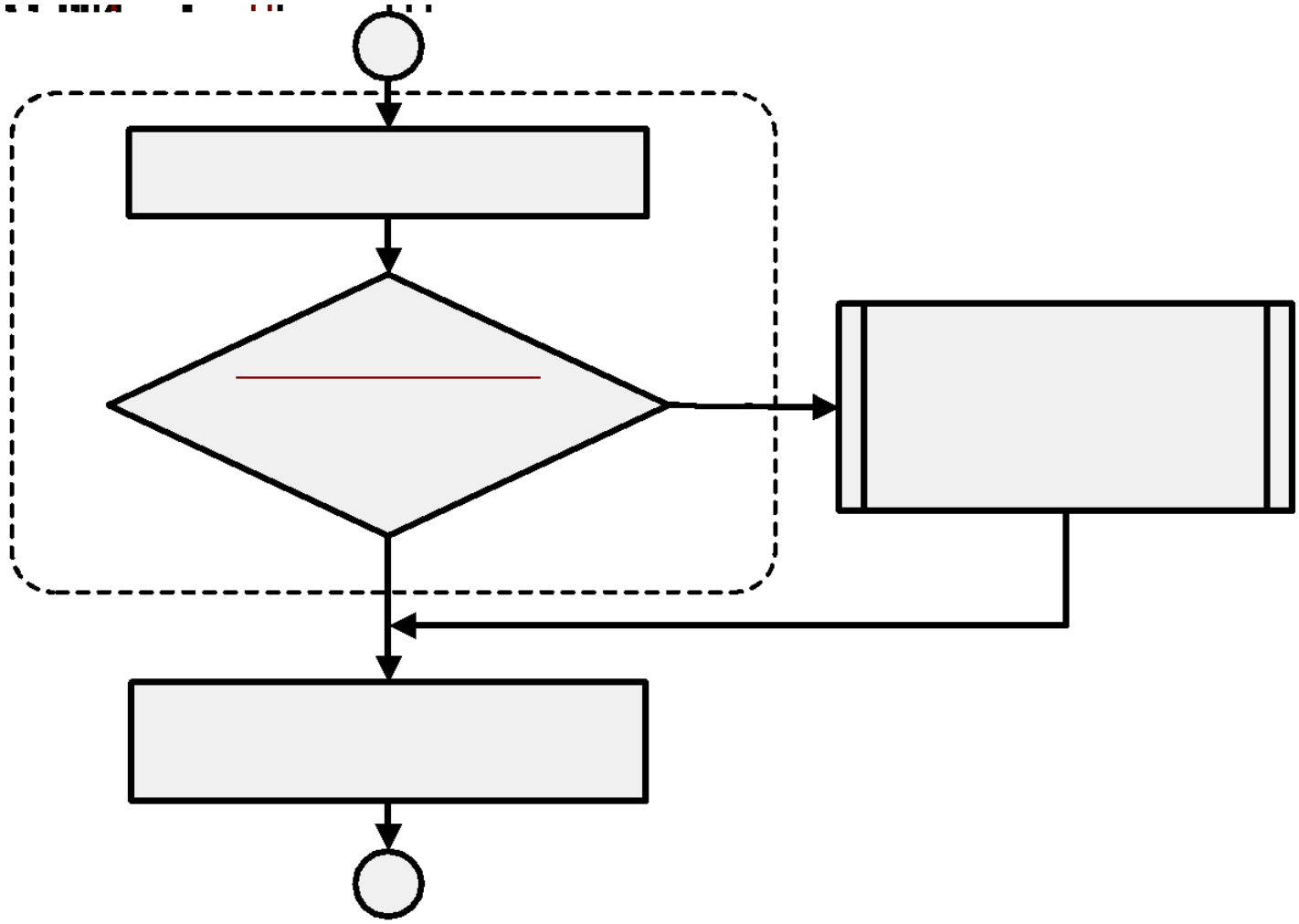
В этом случае ВУ формирует сигнал готовности, а процессор опрашивает наличие этого сигнала (программный поллинг).



Недостаток:

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

Обмен по прерыванию



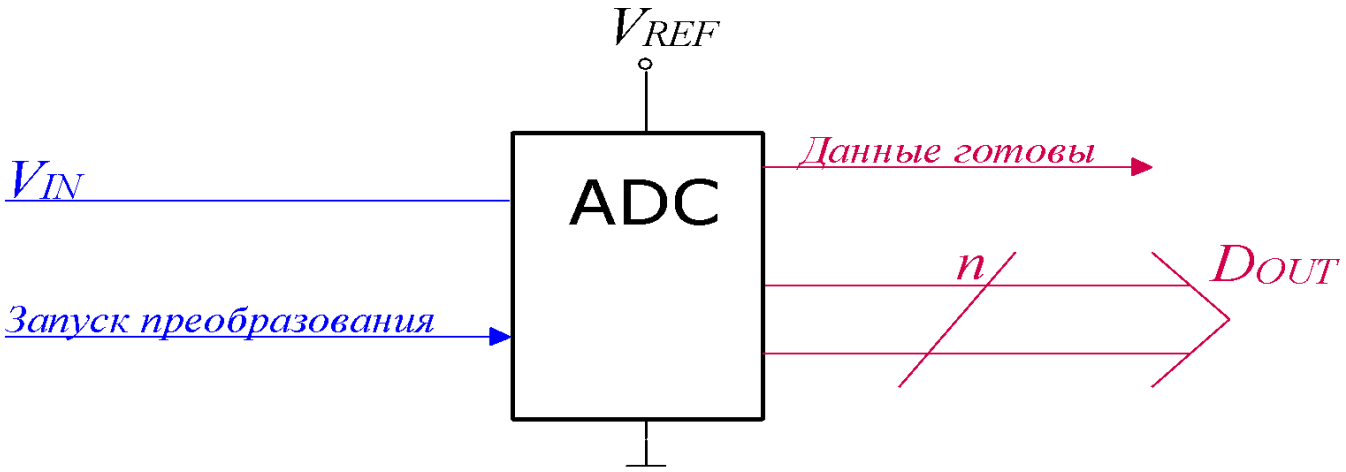
- При отсутствии запроса прерывания процессор выполняет основной алгоритм;

- 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 $D_{OUT\,ti}$. 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 ATmega328p имеет в своем составе 10-разрядный АЦП, а МК STM32F103C8T6 – два 12-разрядных АЦП.

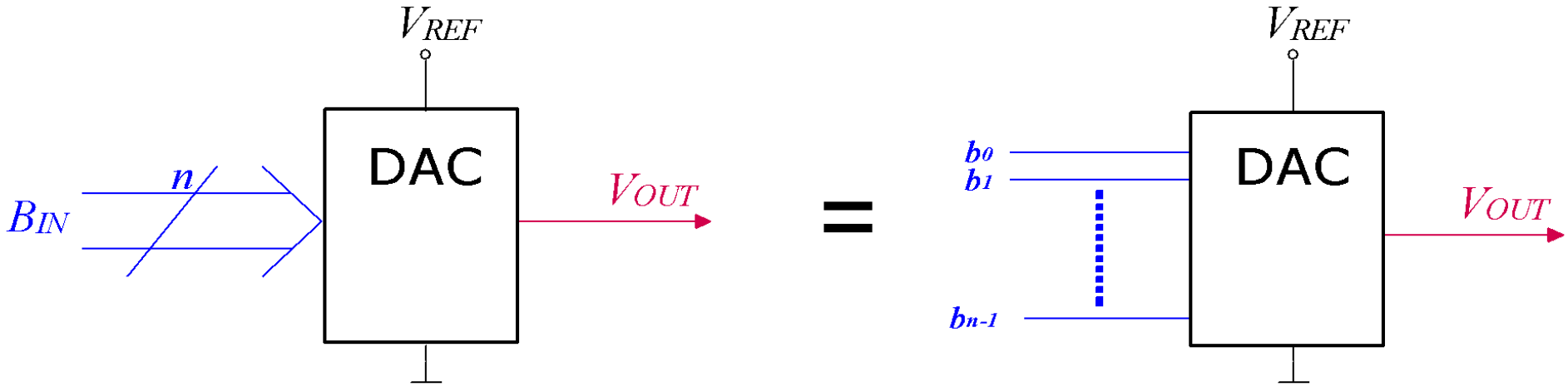
Взаимодействовать с АЦП программа микроконтроллера может двумя способами:

1. Поллингом. 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).



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

где:

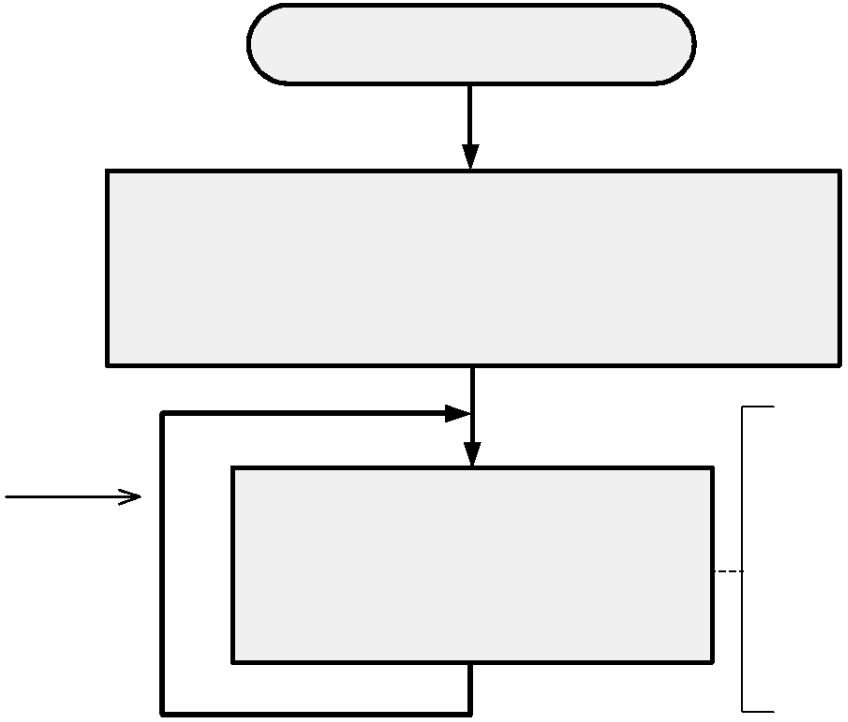
B_{IN} – входной n-разрядный двоичный код;

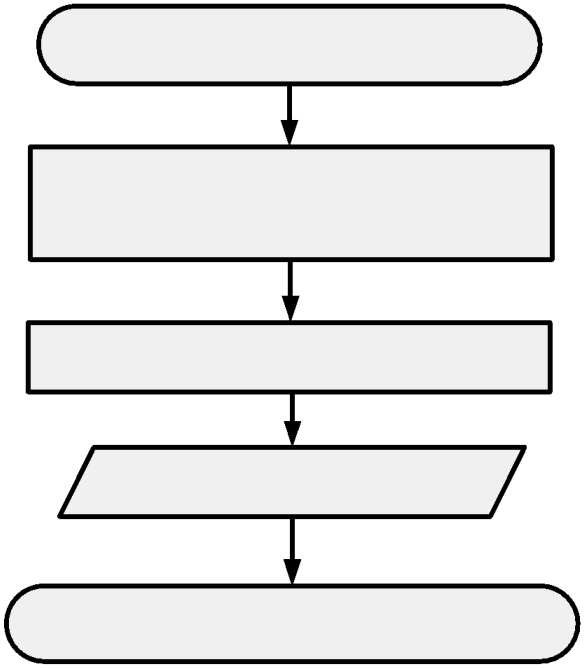
$b_0, b_1 \ldots b_{n-1}$ – разряды (биты) двоичного кода, принимающие значение «0» или «1»
V_{REF} – опорное напряжение ЦАП.
ЦАП иногда входят в состав микроконтроллеров. Однако в МК ATmega328p и STM32F103C8T6 они отсутствуют.
Имеются они, например, в составе МК LGT8F328p (8-разрядный ЦАП на резистивной 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 I^2Cинтерфейсом) и пр.
Кроме того можно самостоятельно собрать ЦАП невысокой разрядности (не более 8) на резистивной матрица 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-разрядныйЦАП, встроенный в МК LGT8F328p. Ниже будут рассмотрены проекты цифрового синтеза периодических сигналов с использованием подобного ЦАП.

УЧЕБНЫЕ ПРОЕКТЫ ДЛЯ STM32F103C8T6, ATMEGA328P

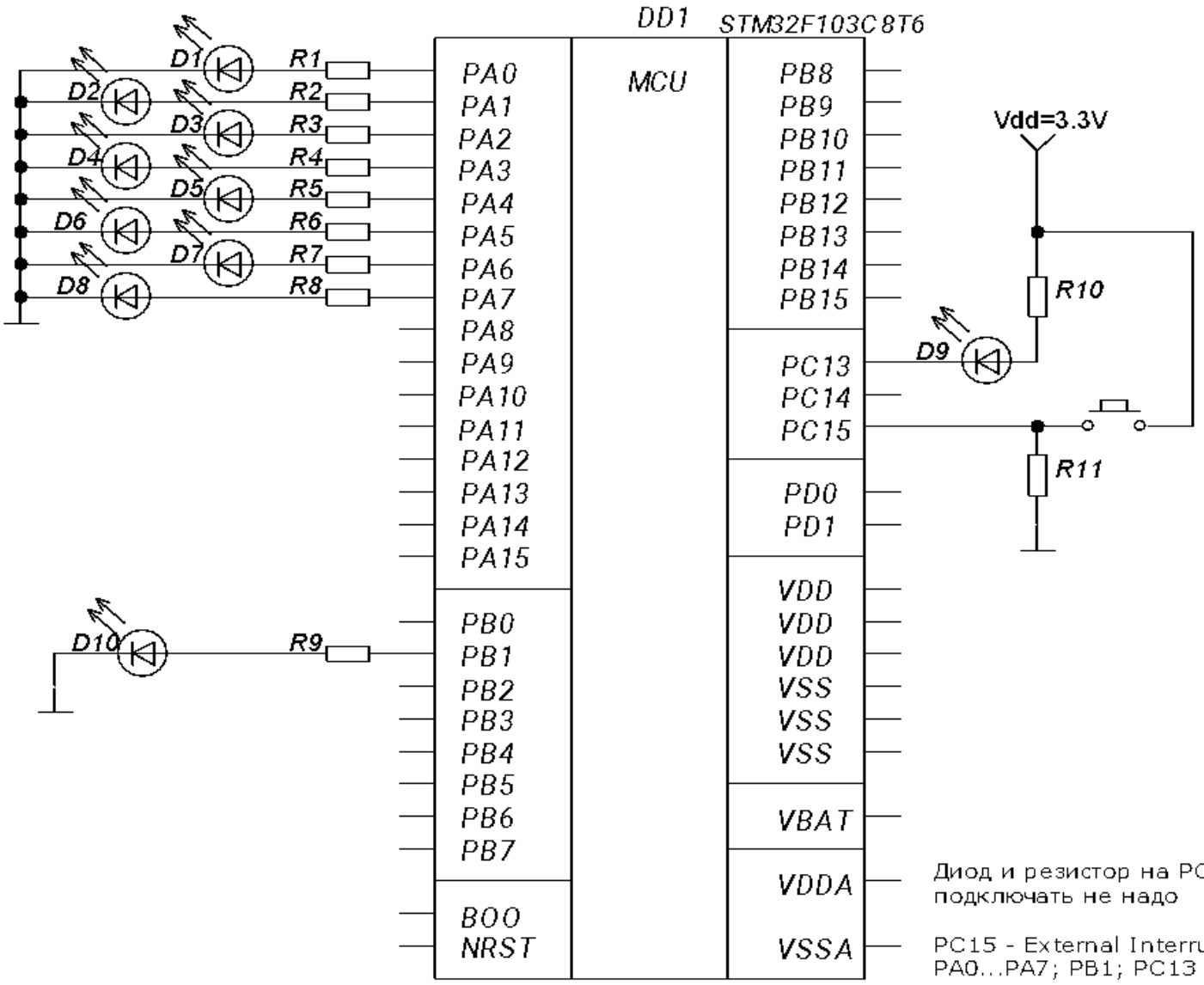
Задача 1. Мигалка с подсчетом кликов кнопки с дребезгом

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.





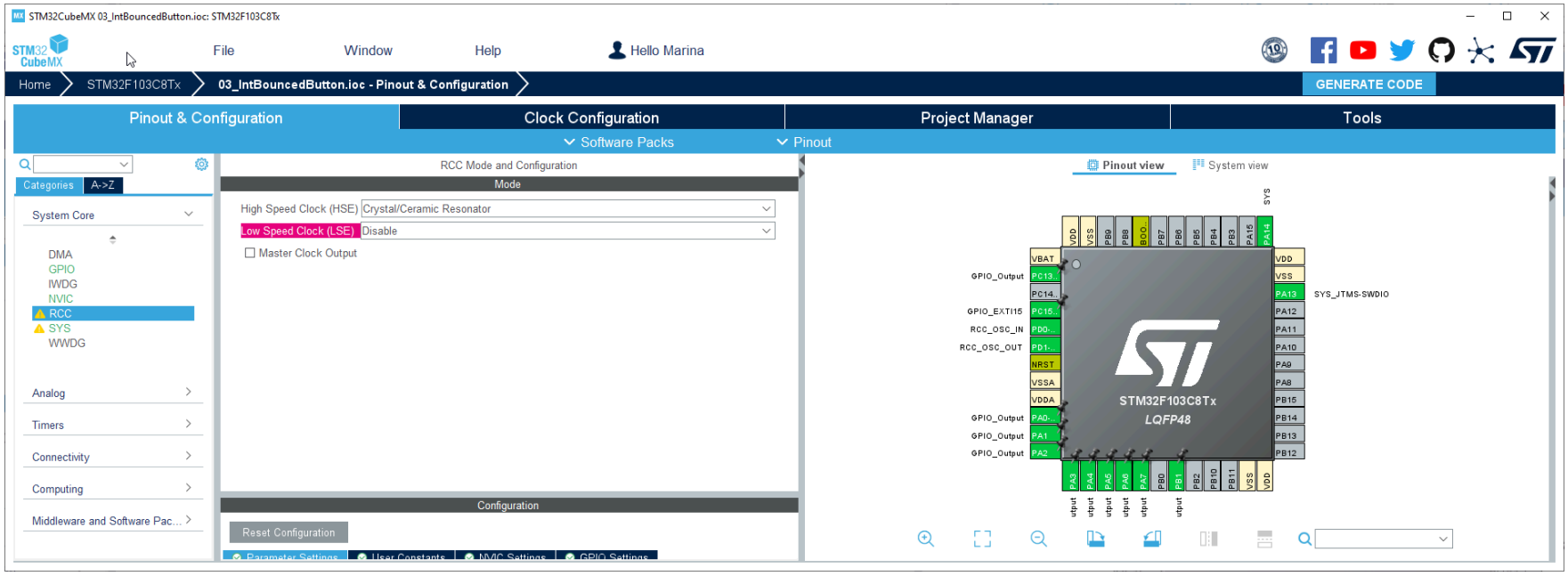
Принципиальная схема проекта

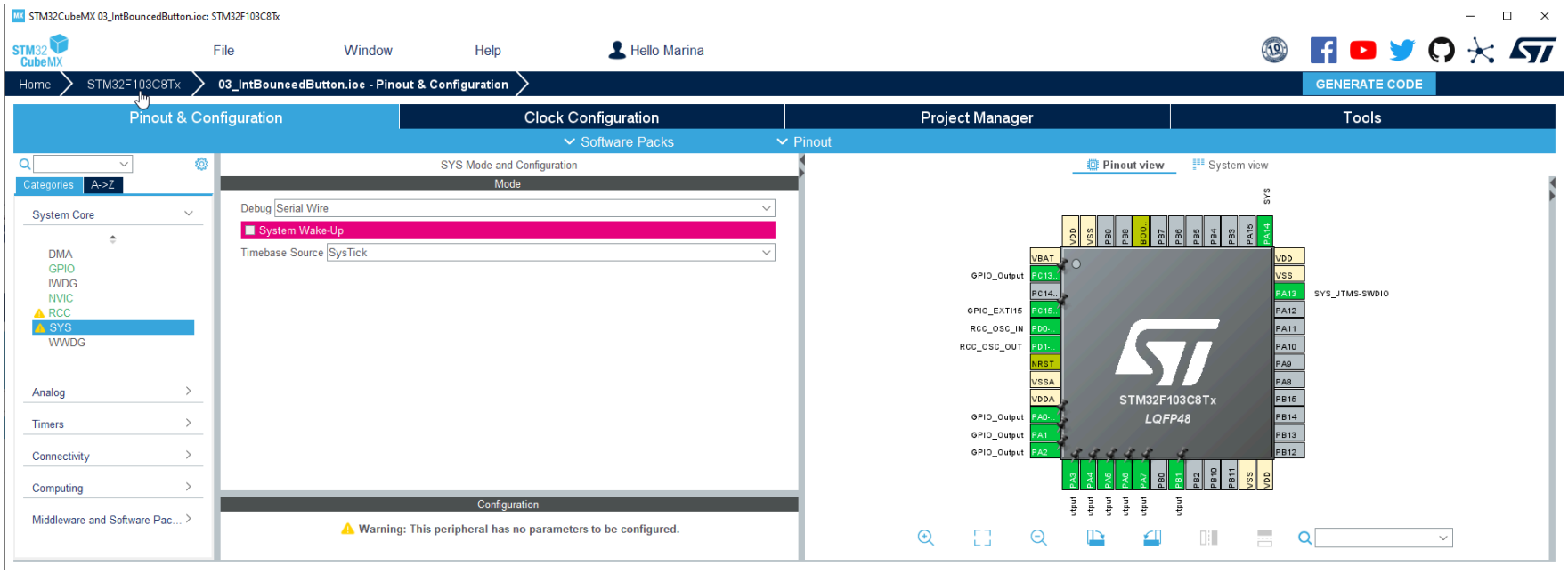


Диод и резистор на PC13 находятся на отладочной плате, подключать не надо

PC15 - External Interrupt Mode; No Pull-up no Pull-down
PA0...PA7; PB1; PC13 - Output Push-Pull; No pull-up, No pul

Настройка проекта в Cube MX




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
Software Packs
Pinout
Categories
A->Z
System Core
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG
Analog
Timers
Connectivity
Computing
Middleware and Software Pac...
SYS Mode and Configuration
Mode
Debug
Serial Wire
System Wake-Up
Timebase Source
SysTick
Configuration
Warning: This peripheral has no parameters to be configured.
Pinout view
System view
STM32F103C8Tx
LQFP48
SYS_JTMS-SWDIO
GPIO_Output
GPIO_EXTI15
RCC_OSC_IN
RCC_OSC_OUT

Настройка внутренней конфигурации МК

STM32CubeMX 03_IntBouncedButton.ioc: STM32F103C8Tx

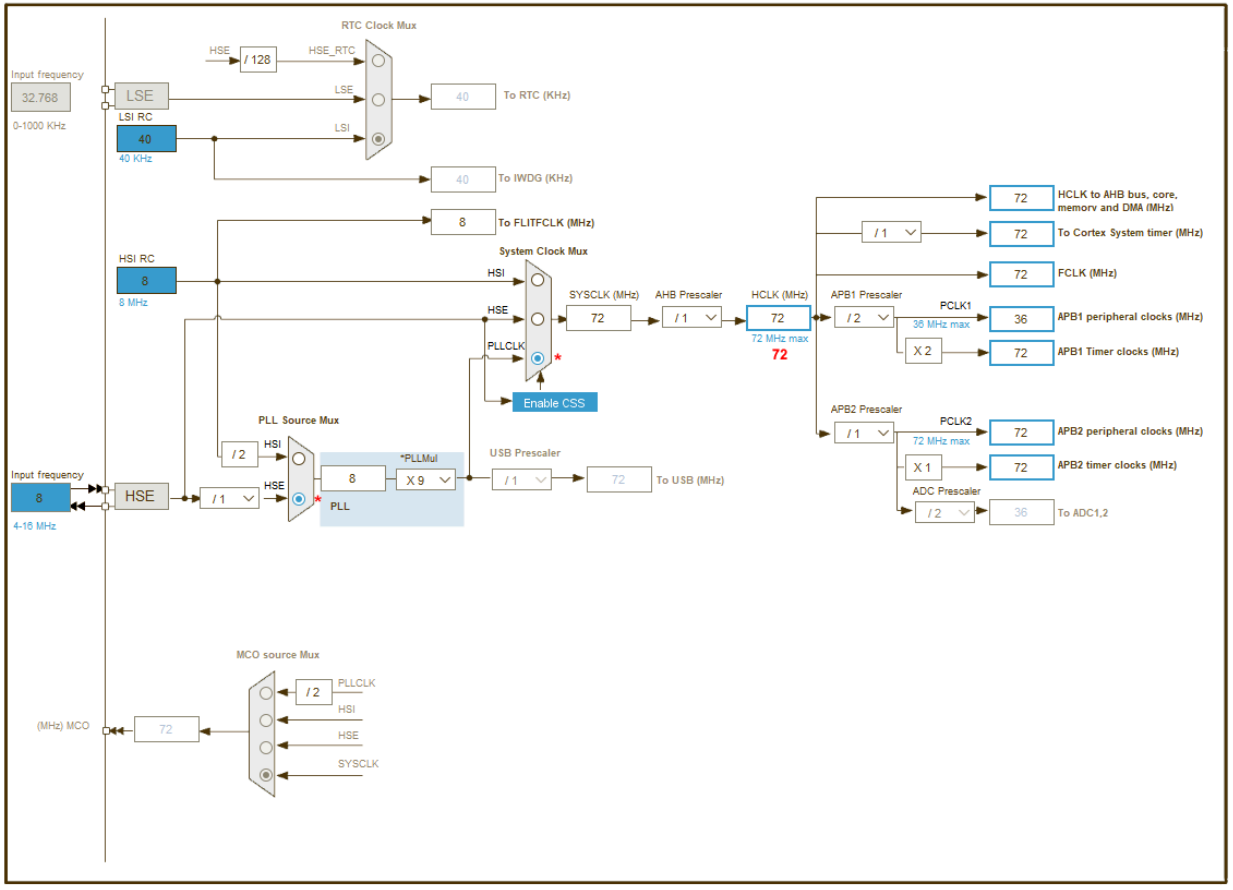
File Window Help Hello Marina

Home > STM32F103C8Tx > 03_IntBouncedButton.ioc - Clock Configuration

GENERATE CODE

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Resolve Clock Issues



Настройка линий ввода-вывода

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

Software Packs | Pinout

Categories | A->Z

System Core
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog

Timers

Connectivity

Computing

Middleware and Software Packs

GPIO Mode and Configuration

Configuration

Group By Peripherals

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		☐
PA1	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PA2	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PA3	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PA4	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PA5	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PA6	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PA7	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PB1	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PC13-TAMP...	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PC15-OSC3...	n/a	n/a	External Inter...	No pull-up an...	n/a		☐

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 view | System view

STM32F103C8Tx
LQFP48

SYS_JTCK-SWCLK

SYS_JTMS-SWDIO

GPIO_Output
GPIO_EXTI15
RCC_OSC_IN
RCC_OSC_OUT

PA5
Reset_State
ADC1_IN5
ADC2_IN5
SPI1_SCK
GPIO_Input
GPIO_Output
GPIO_Analog
EVENTOUT
GPIO_EXTI5

Настройка внешних прерываний (1)

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

Software Packs | Pinout

Categories | A->Z

System Core
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog

Timers

Connectivity

Computing

Middleware and Software Packs

GPIO Mode and Configuration

Configuration

Group By Peripherals

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		☐
PA1	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PA2	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PA3	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PA4	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PA5	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PA6	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PA7	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PB1	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PC13-TAMP...	n/a	Low	Output Push ...	No pull-up an...	Low		☐
PC15-OSC3...	n/a	n/a	External Inter...	No pull-up an...	n/a		☐

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:

Pinout view | System view

SYS_JTCK-SWCLK

VDD VSS PB9 PB8 BOOT0 PB7 PB6 PB5 PB4 PB3 PA15 PA14

VBAT
GPIO_Output PC13-...
PC14-...
GPIO_EXTI15 PC15-...
RCC_OSC_IN PD0-...
RCC_OSC_OUT PD1-...
NRST
VSSA
VDDA
GPIO_Output PA0-...
GPIO_Output PA1
GPIO_Output PA2

PC15-OSC32_OUT
Reset_State
ADC1_EXTI15
ADC2_EXTI15
RCC_OSC32_OUT
GPIO_Input
GPIO_Output
GPIO_Analog
EVENTOUT
GPIO_EXTI15

103C8Tx
LQFP48

VDD
VSS
PA13 SYS_JTMS-SWDIO
PA12
PA11
PA10
PA9
PA8
PB15
PB14
PB13
PB12

PA3 PA4 PA5 PA6 PA7 PB0 PB1 PB2 PB10 PB11 VSS VDD

GPIO_Output GPIO_Output GPIO_Output GPIO_Output GPIO_Output GPIO_Output

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

Software Packs | Pinout

Categories | A->Z

System Core
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog

Timers

Connectivity

Computing

Middleware and Software Packs

NVIC Mode and Configuration

Configuration

NVIC | Code generation

Priority Group: 4 bits for pre-em... ☐ Sort by Premption 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	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Prefetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	15	0
PVD interrupt through EXTI line 16	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
EXTI line[15:10] interrupts	☑	0	0

☐ Enabled Preemption Priority Sub Priority

Pinout view | System view

STM32F103C8Tx
LQFP48

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

Software Packs | Pinout

Categories | A->Z

System Core
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog
Timers
Connectivity
Computing
Middleware and Software Packs

NVIC Mode and Configuration

Configuration

NVIC | Code generation

Enabled interrupt table	Select for init...	Generate Enable in...	Generate IRQ handler	Call HAL handler
System service call via SWI inst...	☐	☐	☑	☐
Debug monitor	☐	☐	☑	☐
Pendable request for system ser...	☐	☐	☑	☐
Time base: System tick timer	☐	☐	☑	☑
EXTI line[15:10] interrupts	☐	☑	☑	☑

Interrupt unmasking ordering table (interrupt init code is moved after all the peripheral init code)

Rank	Interrupt name

Pinout view | System view

STM32F103C8Tx
LQFP48

Настройки программного проекта и генерация кода

STM32CubeMX 03_IntBouncedButton.ioc: STM32F103C8Tx

STM32 CubeMX | File | Window | Help | Hello Marina

Home > STM32F103C8Tx > 03_IntBouncedButton.ioc - Project Manager | GENERATE CODE

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Project | Code Generator | Advanced 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 Min Version: V5.32 [ ] Generate Under Root

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 [x] Use latest available version

[x] Use Default Firmware Location

Firmware Relative Path: C:/Users/ameli/STM32Cube/Repository/STM32Cube_FW_F1_V1.8.5 [Browse]

STM32CubeMX 03_IntBouncedButton.ioc: STM32F103C8Tx

STM32 CubeMX | File | Window | Help | Hello Marina

Home > STM32F103C8Tx > 03_IntBouncedButton.ioc - Project Manager | GENERATE CODE

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Project

Code Generator

Advanced Settings

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

Generated files

- [ ] Generate peripheral initialization as a pair of '.c/.h' files per peripheral
- [ ] Backup previously generated files when re-generating
- [x] Keep User Code when re-generating
- [x] Delete previously generated files when not re-generated

Search needed files for the project...

HAL Settings

- [ ] Set all free pins as analog (to optimize the power consumption)
- [ ] Enable Full Assert

User Actions

Before Code Generation [ ] Browse

After Code Generation [ ] Browse

Template Settings

Select a template to generate customized code Settings...

STM32CubeMX 03_IntBouncedButton.ioc: STM32F103C8Tx

STM32 CubeMX | File | Window | Help | Hello Marina

Home > STM32F103C8Tx > 03_IntBouncedButton.ioc - Project Manager | GENERATE CODE

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Project | Code Generator | Advanced Settings

Driver Selector

Search (Ctrl+F)

RCC	HAL
GPIO	HAL

Generated Function Calls

Generate Code	Rank	Function Name	Peripheral Instance Name	Do Not Generate Function Call	Visibility (Static)
☑	1	SystemClock_Config	RCC	☐	☐
☑	2	MX_GPIO_Init	GPIO	☐	☑

Code Generation

The Code is successfully generated under :

D:/KEIL/03_IntBouncedButton

Project language : C

Open Folder | Open Project | Close

Register CallBack

Search (Ctrl+F)

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

Программный проект в Keil µ vision

D:\KEIL\03_IntBouncedButton\MDK-ARM\03_IntBouncedButton.uvprojx - µVision [Non-Commercial Use License]

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

EXTI_PR_PR15

03_IntBouncedButton

Project

- Project: 03_IntBouncedButton
  - 03_IntBouncedButton
    - Application/MDK-ARM
    - Application/User/Core
      - main.c
      - stm32f1xx_it.c
      - stm32f1xx_hal_msp.c
    - Drivers/STM32F1xx_HAL_Driver
    - Drivers/CMSIS
    - CMSIS

Project | Books | Functions | Templates

main.c

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

Options for Target '03_IntBouncedButton'

Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities

STMicroelectronics STM32F103C8

Xtal (MHz): <undefined>

Operating system: None

System Viewer File: STM32F103xx.svd

Use Custom File

Code Generation

ARM Compiler: Use default compiler version 6

Use MicroLIB Big Endian

Use Cross-Module Optimization

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
☑	IROM1:	0x8000000	0x10000	◉
☐	IROM2:			○

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
☑	IRAM1:	0x20000000	0x5000	☐
☐	IRAM2:			☐

OK | Cancel | Defaults | Help

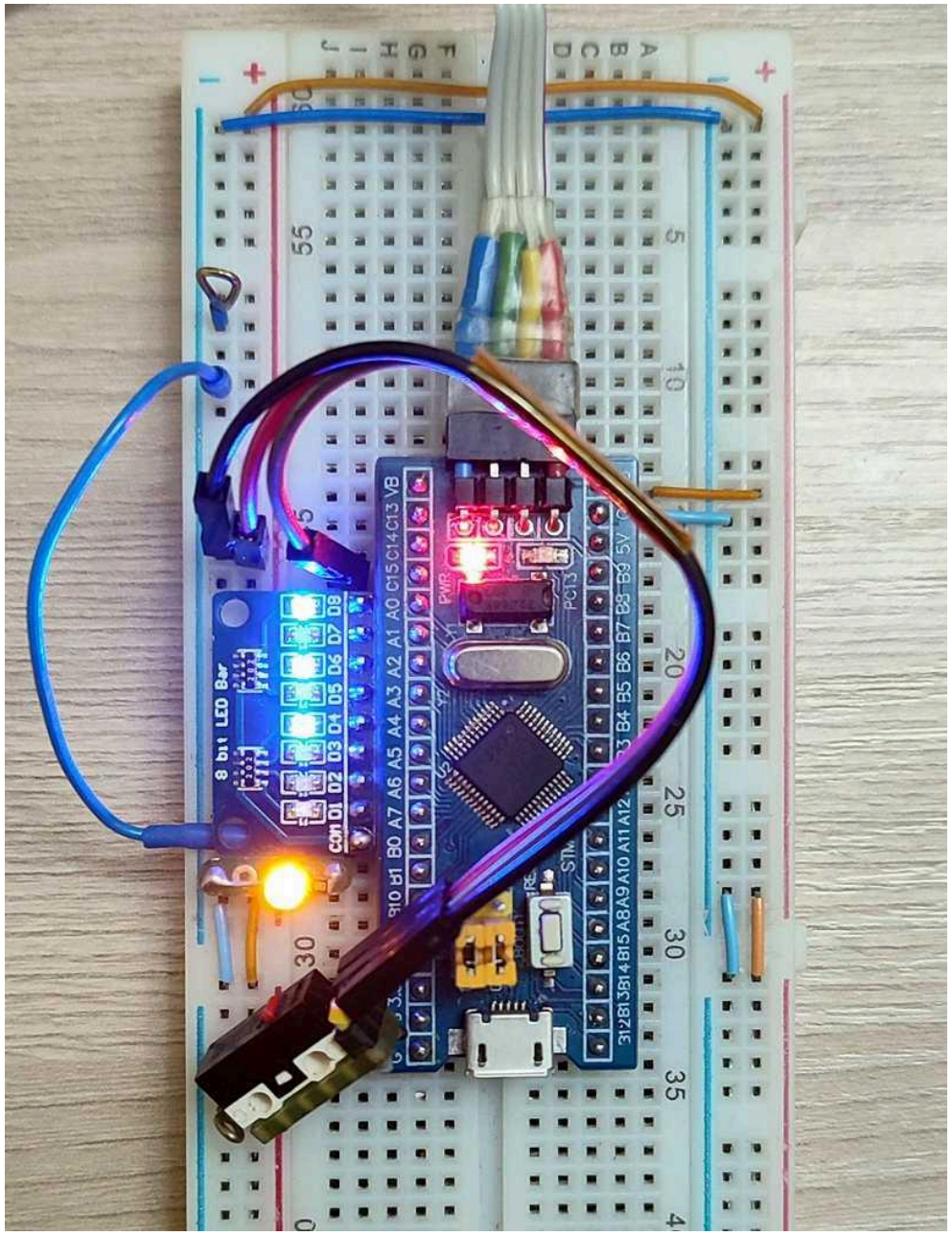
Build Output

ST-Link Debugger L:1 C:1

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 противопол.
        HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_1); //Изменение состояния PB1 на противопол.
        HAL_Delay(250);                            //Задержка 250 мс
  /* USER CODE END WHILE */
  }
}
```

Макет программно-аппаратного проекта

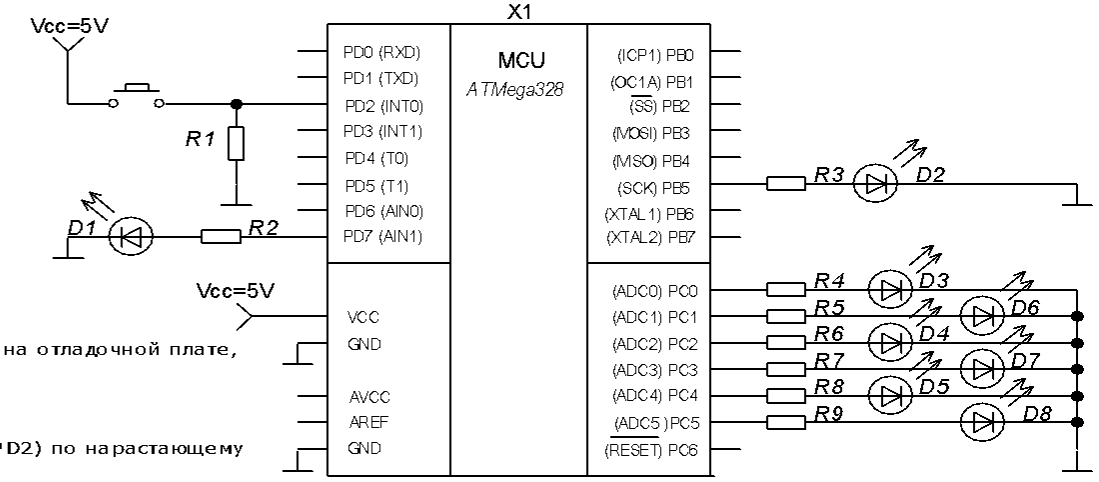

8 bit LED Bar
PWR
PC13

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);          //IO2 (INT0) - режим ввода с высокоимпедансным входом
pinMode(13, OUTPUT);        //IO13 (PB5) в режиме вывода, к нему подключен светодиод на плате
digitalWrite(13,LOW);       //гашение светодиода на плате
pinMode(7, OUTPUT);         //IO7 (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 (IO2) -
срабатывание по нарастающему фронту
}

void loop() {
  // put your main code here, to run repeatedly:
PORTB^=(0b00100000);         //изменение состояния светодиода на IO13=PB5 на противоположное
PORTD^=(0b10000000);         //изменение состояния светодиода на IO7=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
```

```
};
```

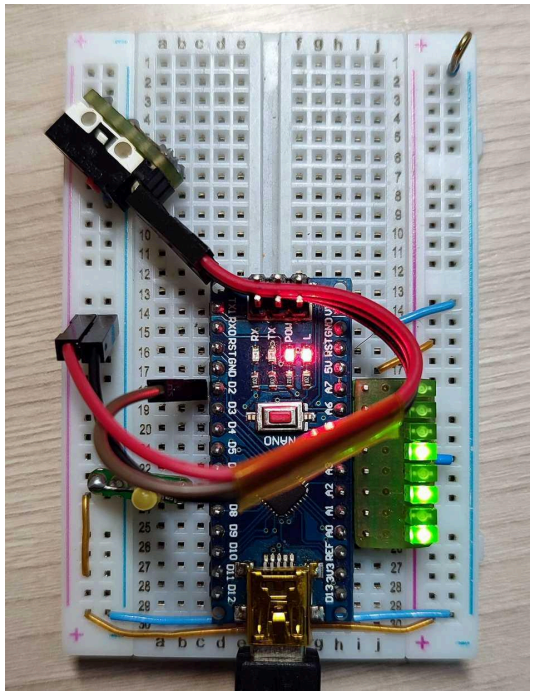
Принципиальная схема и макет проекта

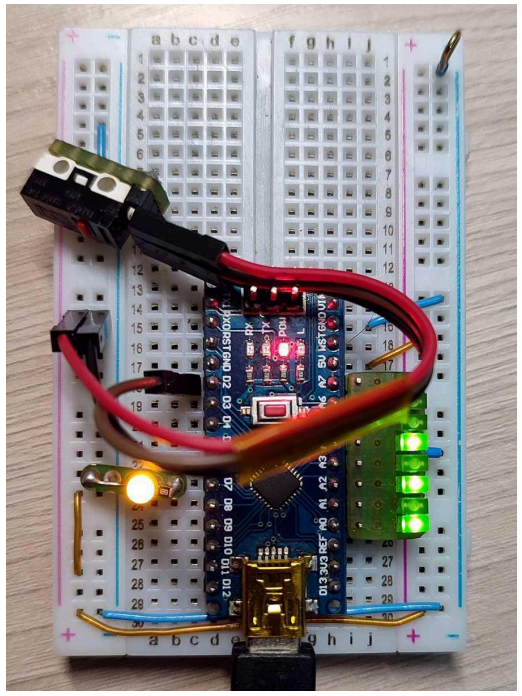


Диод и резистор на PB5 (PIN13) находятся на отладочной плате, подключать не надо

PC0 - PC5 - на вывод данных

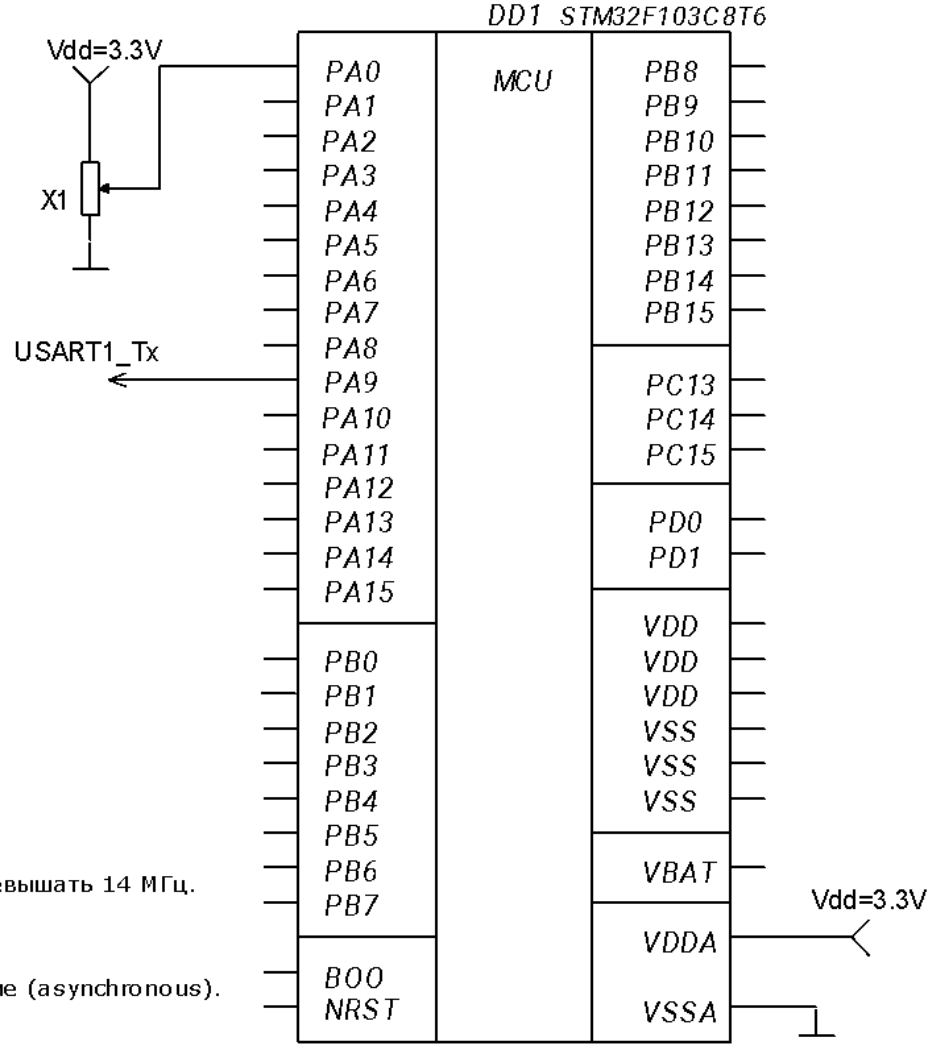
Используется внешнее прерывание INT0 (PD2) по нарастающему фронту





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

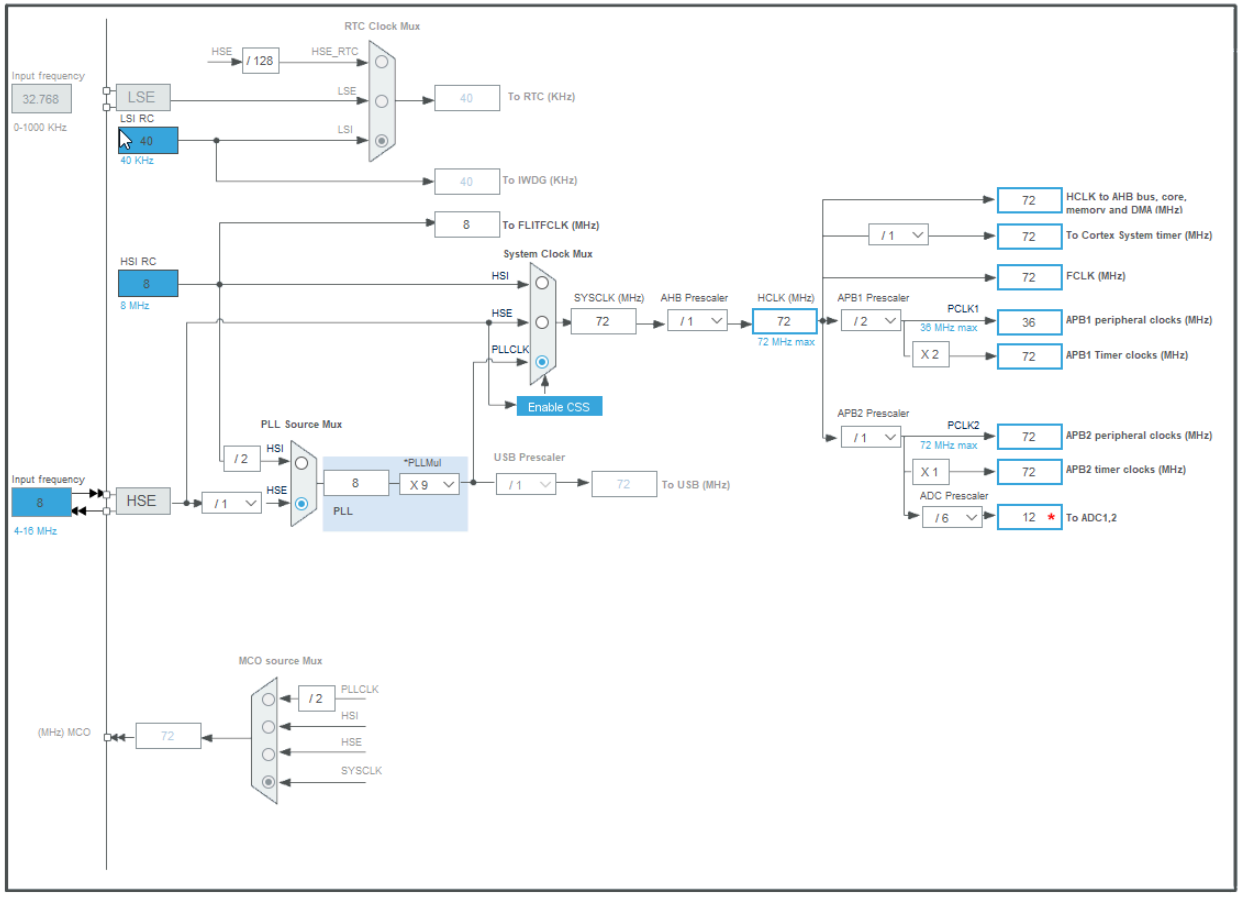
File Window Help Hello Marina

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

GENERATE CODE

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Resolve Clock Issues



Настройка ADC1

STM32CubeMX 02_ADC1-interrupt.ioc*: STM32F103C8Tx

File Window Help Hello Marina

Home > STM32F103C8Tx > 02_ADC1-interrupt.ioc - Pinout & Configuration

GENERATE CODE

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

- System Core
- Analog
  - ADC1
  - ADC2
- Timers
- Connectivity
- Computing
- Middleware and Software Pac...

ADC1 Mode and Configuration

Mode

- [x] IN0
- [ ] IN1
- [ ] IN2
- [ ] IN3
- [ ] IN4
- [ ] IN5
- [ ] IN6
- [ ] IN7
- [ ] IN8
- [ ] IN9
- [ ] Temperature Sensor Channel
- [ ] Vrefint Channel

EXTI Conversion Trigger: Disable

Configuration

Reset Configuration

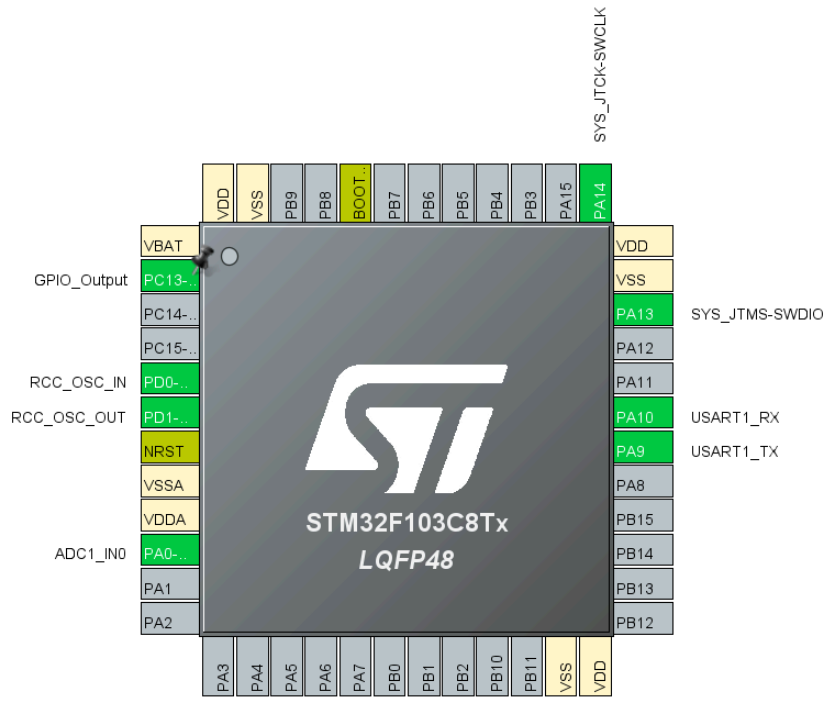
Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Data Alignment	Right alignment
Scan Conversion Mode	Disabled
Continuous Conversion Mode	Disabled
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	1.5 Cycles
ADC_Injected_ConversionMode	
Enable Injected Conversions	Disable
WatchDog	
Enable Analog WatchDog Mode	☐

Pinout view | System view



Настройка USART1

STM32CubeMX 01_ADC1-polling.ioc: STM32F103C8Tx

File Window Help Hello Marina

Home > STM32F103C8Tx > 01_ADC1-polling.ioc - Pinout & Configuration

GENERATE CODE

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

System Core

Analog
- ADC1
- ADC2

Timers
- RTC
- TIM1
- TIM2
- TIM3
- TIM4

Connectivity
- CAN
- I2C1
- I2C2
- SPI1
- SPI2
- USART1
- USART2
- USART3
- USB

Computing

Middleware and Software Pac...

USART1 Mode and Configuration

Mode

Mode: Asynchronous

Hardware Flow Control (RS232): Disable

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

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

Pinout view | System view

SYS_JTCK-SWCLK

VDD VSS PB9 PB8 BOOT PB7 PB6 PB5 PB4 PB3 PA15 PA14

VBAT
PC13-..
PC14-..
PC15-..
RCC_OSC_IN PD0-..
RCC_OSC_OUT PD1-..
NRST
VSSA
VDDA
ADC1_IN0 PA0-..
PA1
PA2

VDD
VSS
PA13 SYS_JTMS-SWDIO
PA12
PA11
PA10 USART1_RX
PA9 USART1_TX
PA8
PB15
PB14
PB13
PB12

PA3 PA4 PA5 PA6 PA7 PB0 PB1 PB2 PB10 PB11 VSS VDD

STM32F103C8Tx
LQFP48

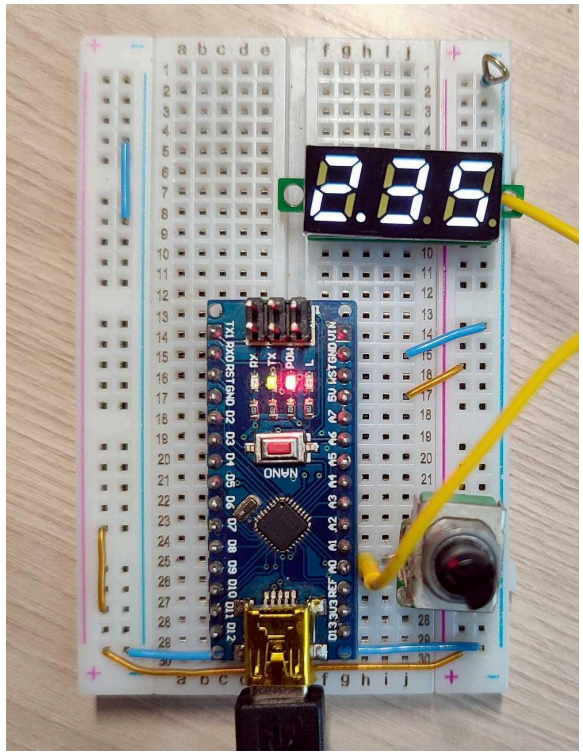
Фрагменты файла *main.c*

```
/* 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
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
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*.

STM32CubeMX 02_ADC1-interrupt.ioc*: STM32F103C8Tx

File Window Help Hello Marina

Home > STM32F103C8Tx > 02_ADC1-interrupt.ioc - Pinout & Configuration

GENERATE CODE

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

System Core
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog
- ADC1
- ADC2

Timers

Connectivity

Computing

Middleware and Software Pac...

ADC1 Mode and Configuration

Mode

- IN0
- IN1
- IN2
- IN3
- IN4
- IN5
- IN6
- IN7
- IN8
- IN9
- Temperature Sensor Channel
- Vrefint Channel

EXTI Conversion Trigger: Disable

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
ADC1 and ADC2 global interrupts	☑	0	0

Pinout view | System view

SYS_JTCK-SWCLK

VDD VSS PB9 PB8 BOOT PB7 PB6 PB5 PB4 PB3 PA15 PA14

VBAT
GPIO_Output PC13-..
PC14-..
PC15-..
RCC_OSC_IN PD0-..
RCC_OSC_OUT PD1-..
NRST
VSSA
VDDA
ADC1_IN0 PA0-..
PA1
PA2

VDD
VSS
PA13 SYS_JTMS-SWDIO
PA12
PA11
PA10 USART1_RX
PA9 USART1_TX
PA8
PB15
PB14
PB13
PB12

STM32F103C8Tx
LQFP48

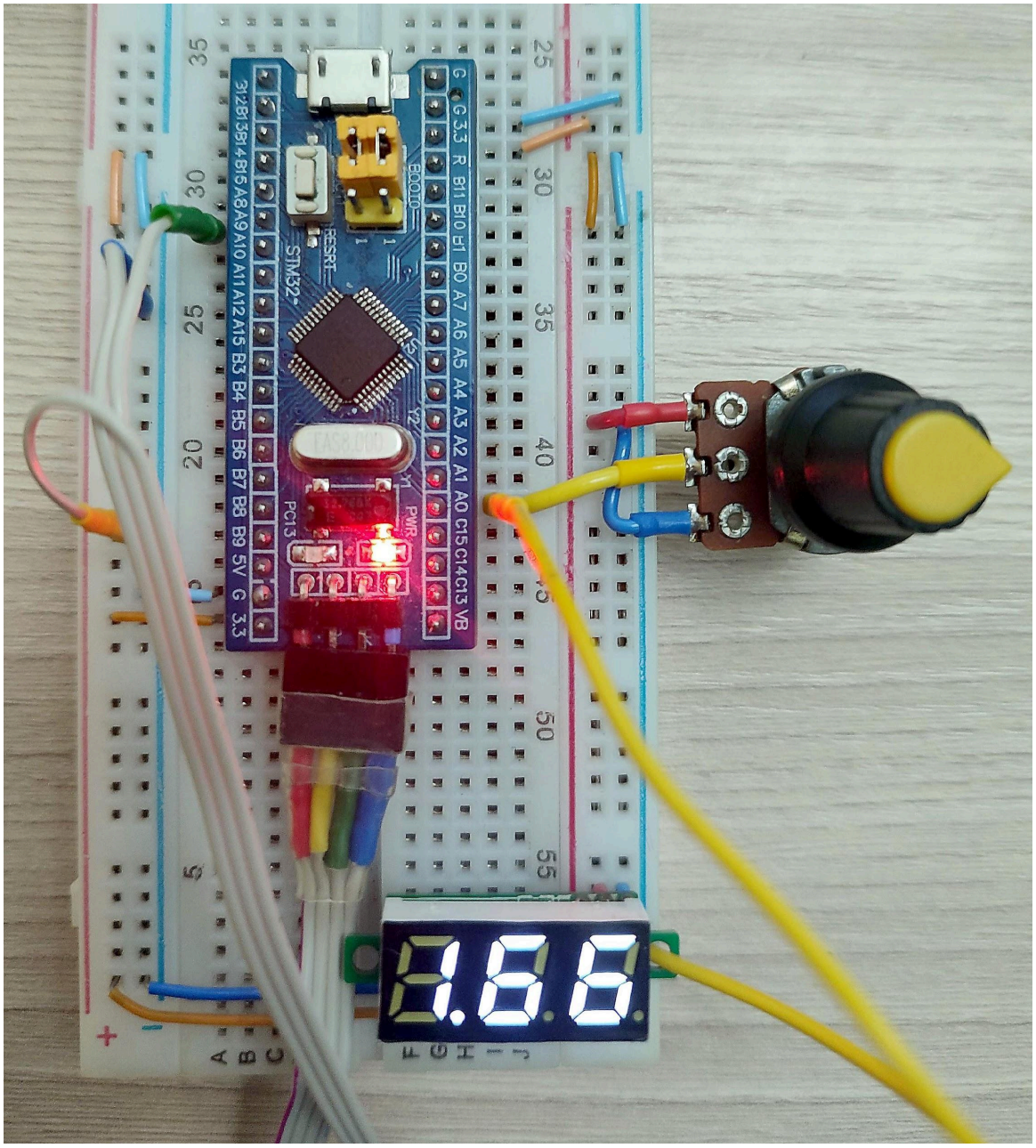
PA3 PA4 PA5 PA6 PA7 PB0 PB1 PB2 PB10 PB11 VSS VDD

Фрагменты файла *main.c*

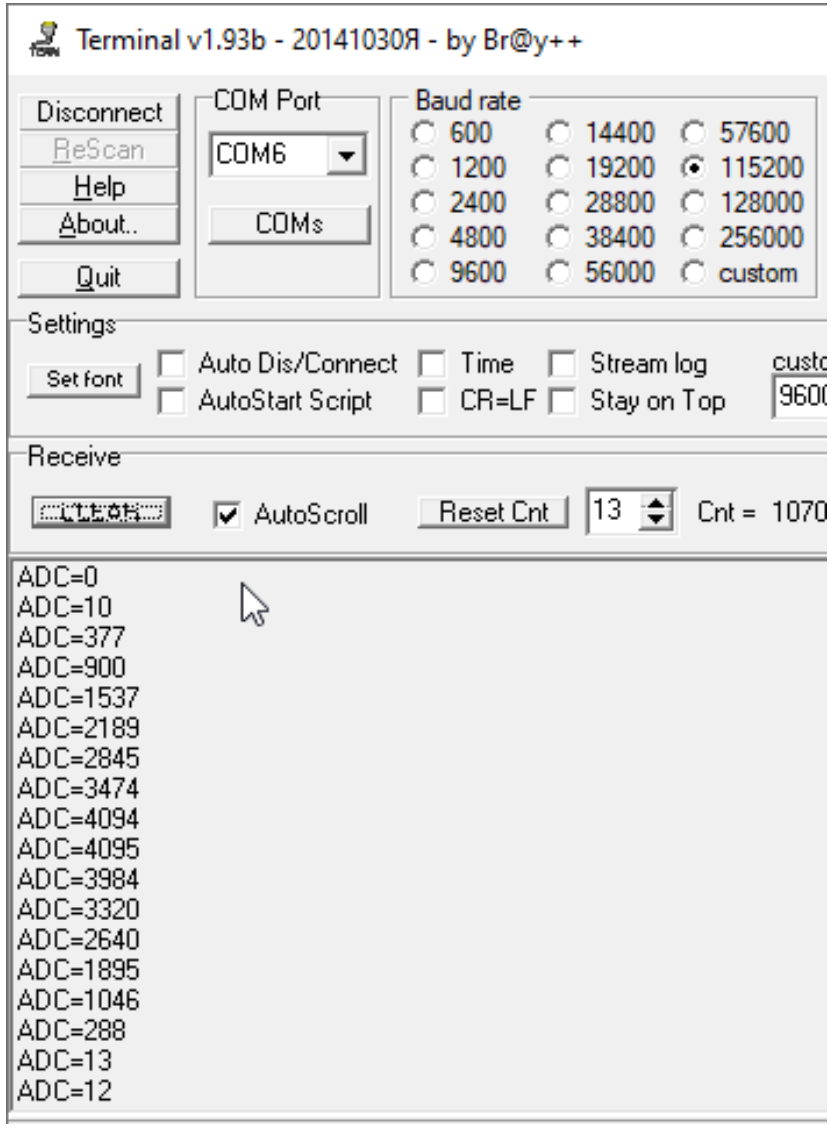
```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)    //фрагмент обработчика прерывания ADC1
{
  if(hadc->Instance == ADC1)                          //проверка прихода прерывания от ACD1
   {
     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 */
  }
```

```
}
```

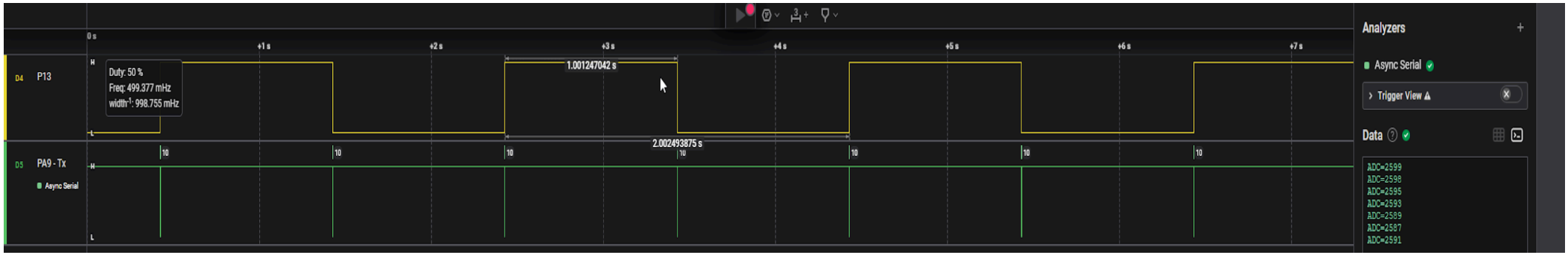
Макет проекта с АЦП



Терминал при медленном вращении ручки потенциометра по и против часовой стрелки



Сигналограммы логического анализатора



Светодиод на PC13 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)

STM32CubeMX 02_ADC1-interrupt.ioc*: STM32F103C8Tx

File Window Help Hello Marina

Home > STM32F103C8Tx > 02_ADC1-interrupt.ioc - Pinout & Configuration

GENERATE CODE

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

- System Core
- Analog
  - ADC1
  - ADC2
- Timers
- Connectivity
- Computing
- Middleware and Software Pac...

ADC1 Mode and Configuration

Mode

- IN0
- IN1
- IN2
- IN3
- IN4
- IN5
- IN6
- IN7
- IN8
- IN9
- Temperature Sensor Channel
- Vrefint Channel

EXTI Conversion Trigger: Disable

Configuration

Reset Configuration

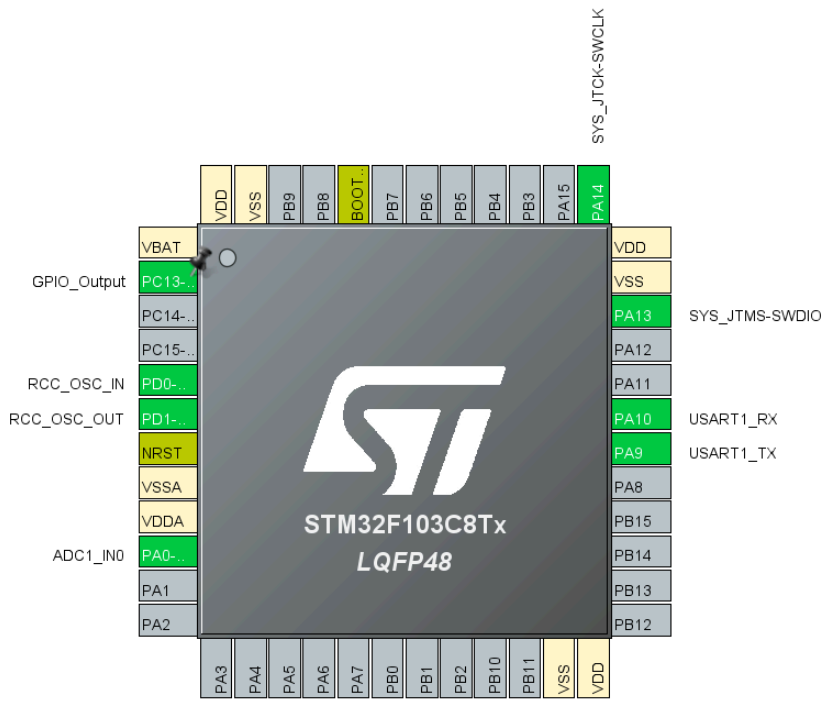
Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
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 | System view

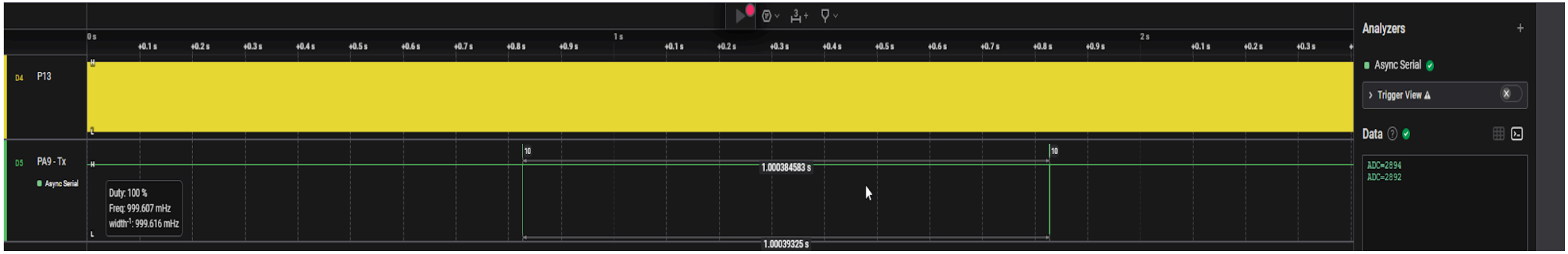


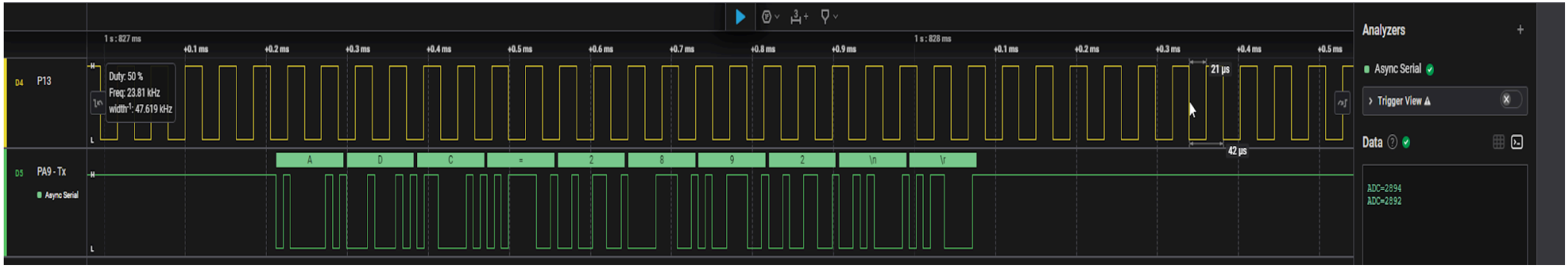
Фрагменты файла *main.c*

```
/* 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





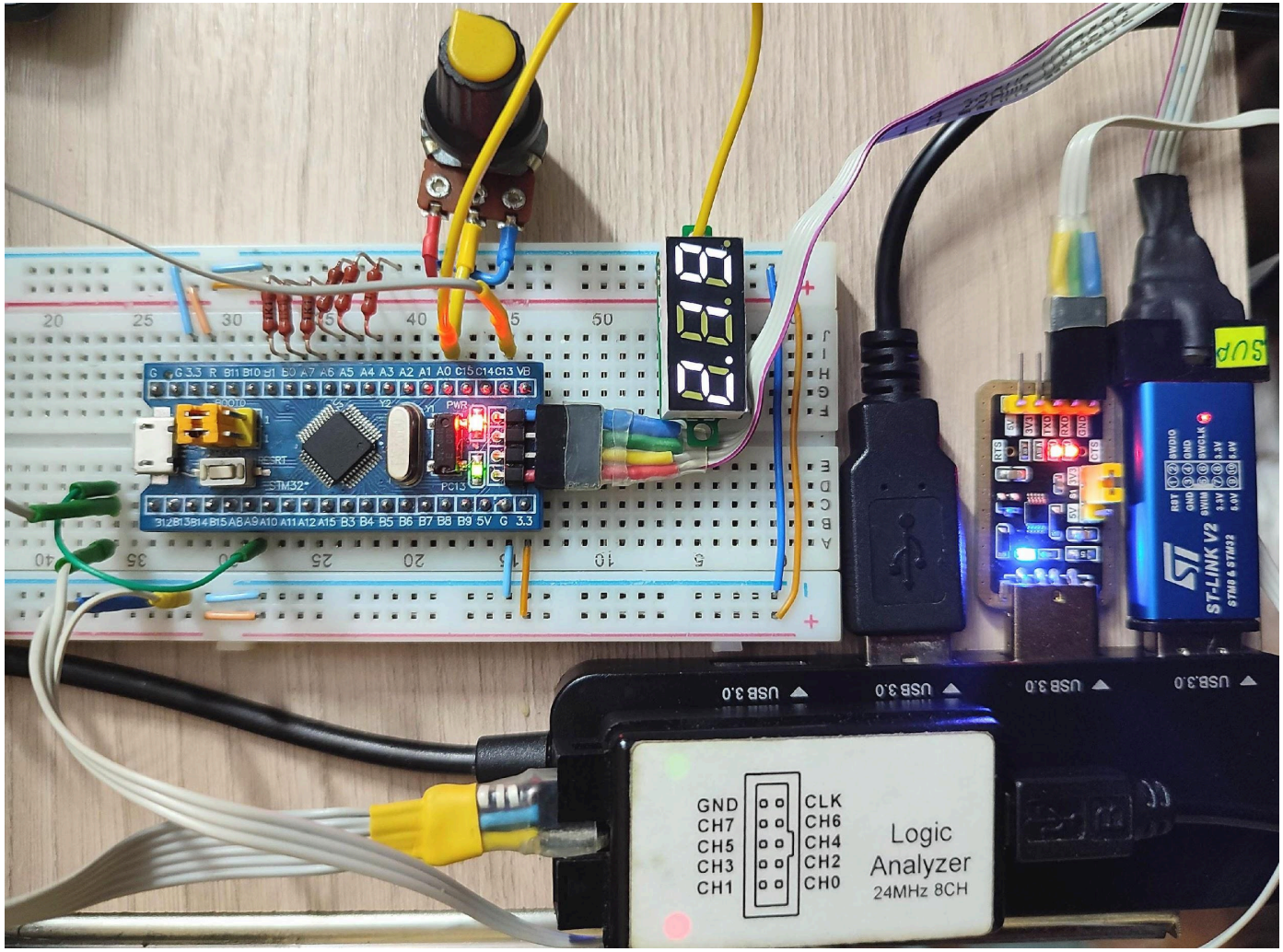
Светодиод на PC13 (на отладочной плате) светится в полнакала из-за 50% коэффициента заполнения ШИМ-сигнала на нём.

Время преобразования АЦП в тактах:
Tconv=SamplingTime+12.5=239.5+12.5=252 такта.

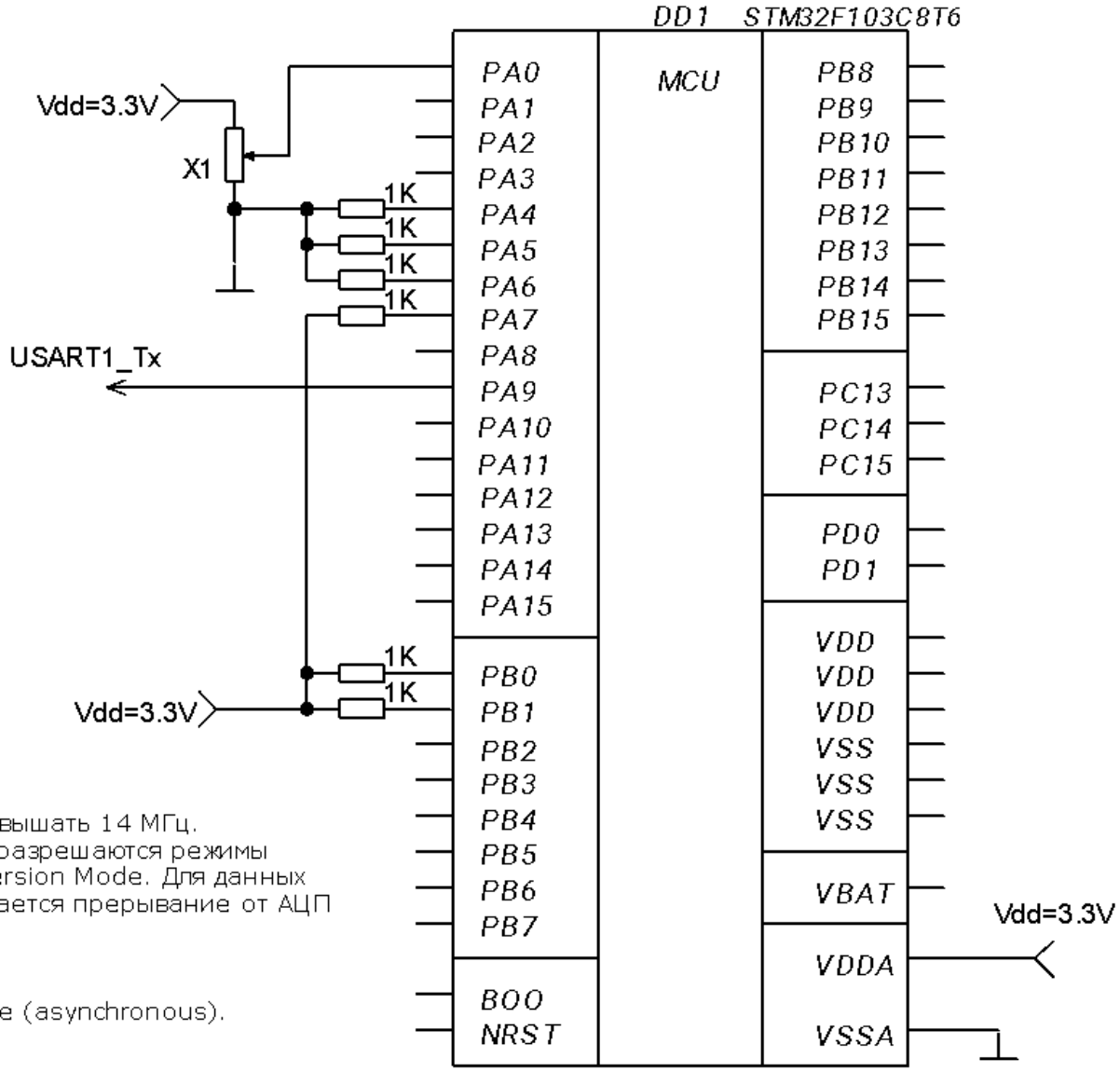
Время преобразования АЦП здесь в секундах: 252*(1/12E6)=21E-6 с =21 мкс — длительность интервала неизменного состояния светодиода на PC13 (см. нижнюю сигналограмму).

Задача 2(в). АЦП с обменом по DMA

Организовать циклическое аналого-цифровое преобразование напряжений на 10 аналоговых входах МК (PA0-PA7, PB0, PB1) с циклической записью в массив из 10 слов ***uint_16t adc[10]***. Вывод PA0 подключен к движку потенциометра, выводы PA1PA3 висят неподключенные, PA4-PA6 подключены к «земле», PA7, PB0, PB1 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*.


ST-LINK V2
USB 3.0
GND
CH7
CH5
CH3
CH1
CLK
CH6
CH4
CH2
CH0
Logic
Analyzer
24MHz 8CH

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

STM32CubeMX 04_ADC1-DMAblock10-circular.ioc: STM32F103C8Tx

File Window Help Hello Marina

Home > STM32F103C8Tx > 04_ADC1-DMAblock10-circular.ioc - Pinout & Configuration

GENERATE CODE

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

- System Core
- Analog
  - ADC1
  - ADC2
- Timers
- Connectivity
- Computing
- Middleware and Software Pac...

ADC1 Mode and Configuration

Mode

- IN0
- IN1
- IN2
- IN3
- IN4
- IN5
- IN6
- IN7
- IN8
- IN9
- Temperature Sensor Channel
- Vrefint Channel

EXTI Conversion Trigger: Disable

Configuration

Reset Configuration

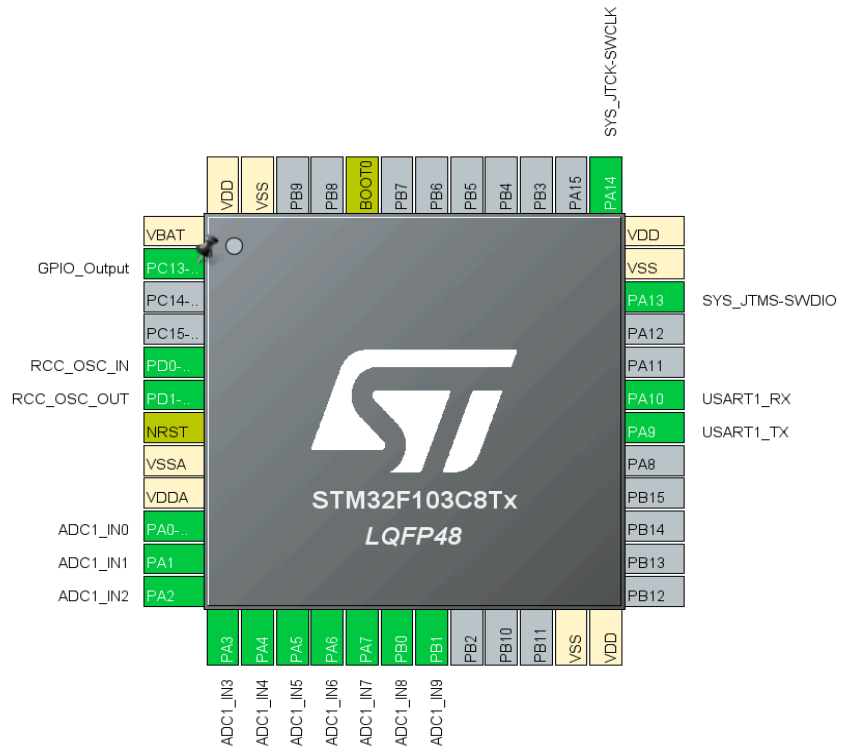
Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Data Alignment	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
Rank	1
Channel	Channel 0
Sampling Time	239.5 Cycles
Rank	2
Channel	Channel 1
Sampling Time	239.5 Cycles
Rank	3
Rank	4
Rank	5
Rank	6
Rank	7
Rank	8
Rank	9
Rank	10
Channel	Channel 9
Sampling Time	239.5 Cycles
ADC_Injected_ConversionMode	
Enable Injected Conversions	Disable
WatchDog	
Enable Analog WatchDog Mode	☐

Pinout view | System view



STM32CubeMX 04_ADC1-DMAblock10-circular.ioc: STM32F103C8Tx

File Window Help Hello Marina

Home > STM32F103C8Tx > 04_ADC1-DMAblock10-circular.ioc - Pinout & Configuration

GENERATE CODE

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

System Core
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog
- ADC1
- ADC2

Timers

Connectivity

Computing

Middleware and Software Pac...

ADC1 Mode and Configuration

Mode

- IN0
- IN1
- IN2
- IN3
- IN4
- IN5
- IN6
- IN7
- IN8
- IN9
- Temperature Sensor Channel
- Vrefint Channel

EXTI Conversion Trigger: Disable

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

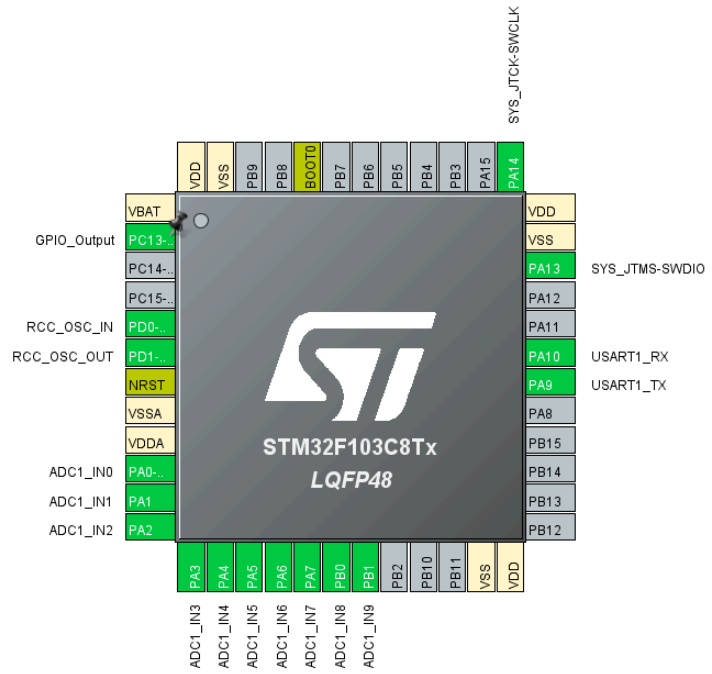
DMA Request	Channel	Direction	Priority
ADC1	DMA1 Channel 1	Peripheral To Memory	Low

Add | Delete

DMA Request Settings

		Peripheral	Memory
Mode	Circular	Increment Address	☐	☑
		Data Width	Half Word	Half Word

Pinout view | System view



STM32CubeMX 04_ADC1-DMAblock10-circular.ioc: STM32F103C8Tx

File Window Help Hello Marina

Home > STM32F103C8Tx > 04_ADC1-DMAblock10-circular.ioc - Pinout & Configuration

GENERATE CODE

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

System Core
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog
- ADC1
- ADC2

Timers

Connectivity

Computing

Middleware and Software Pac...

ADC1 Mode and Configuration

Mode

- IN0
- IN1
- IN2
- IN3
- IN4
- IN5
- IN6
- IN7
- IN8
- IN9
- Temperature Sensor Channel
- Vrefint Channel

EXTI Conversion Trigger: Disable

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
DMA1 channel1 global interrupt	☑	0	0
ADC1 and ADC2 global interrupts	☐	0	0

Pinout view | System view

STM32F103C8Tx
LQFP48

SYS_JTCK-SWCLK

GPIO_Output
RCC_OSC_IN
RCC_OSC_OUT
ADC1_IN0
ADC1_IN1
ADC1_IN2

SYS_JTMS-SWDIO
USART1_RX
USART1_TX

ADC1_IN3 ADC1_IN4 ADC1_IN5 ADC1_IN6 ADC1_IN7 ADC1_IN8 ADC1_IN9

Фрагменты файла *main.c*

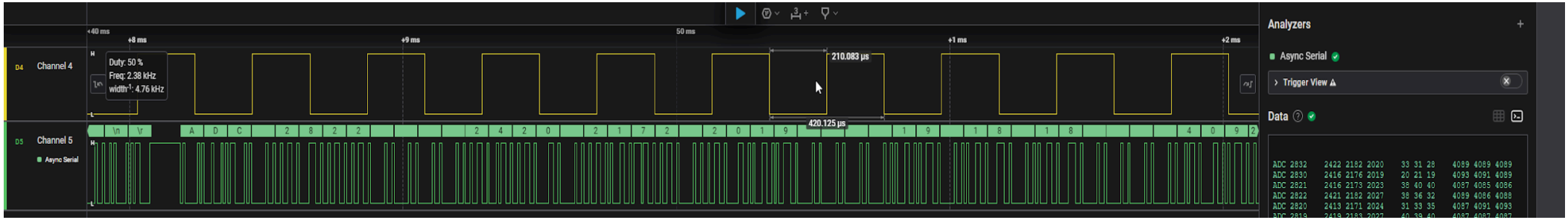
```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "string.h"
/* USER CODE END Includes */
//.....................................................................................
..
/* USER CODE BEGIN PV */
char 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 */
```

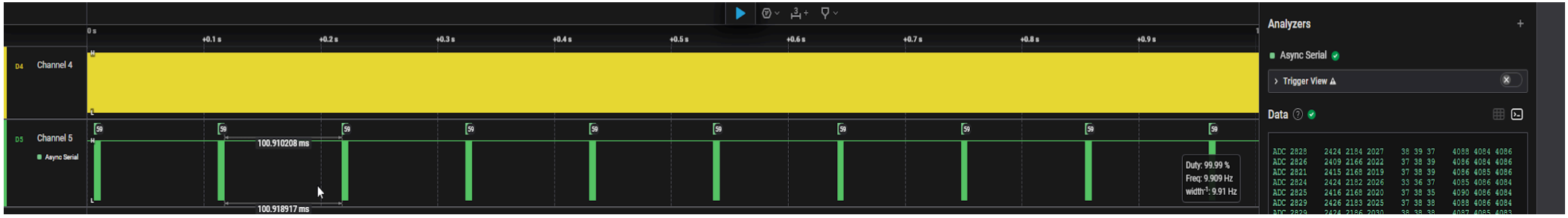
```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    if(flag)
      {
        flag = 0;     //сброс флага
        //формирование строки вывода на терминал для 10 слов данных с АЦП1 (для каналов
0-9)
        snprintf(trans_str, 63, "ADC %d    %d %d %d    %d %d %d    %d %d %d \n\r",
                 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
```

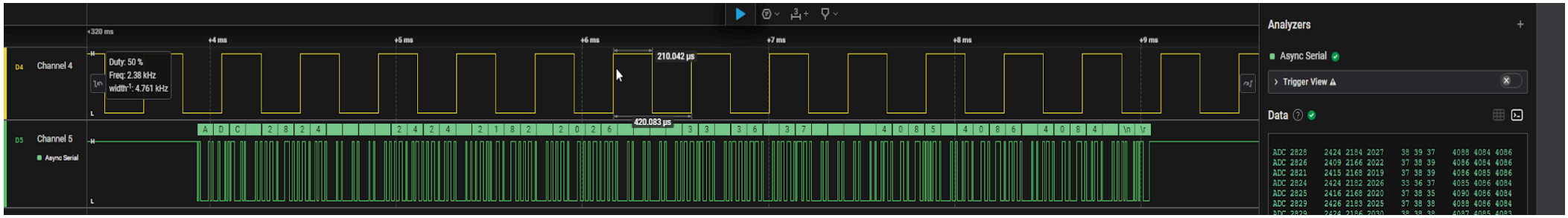
Сигналограммы логического анализатора

Без временной задержки между пакетами вывода



С временной задержкой между пакетами вывода





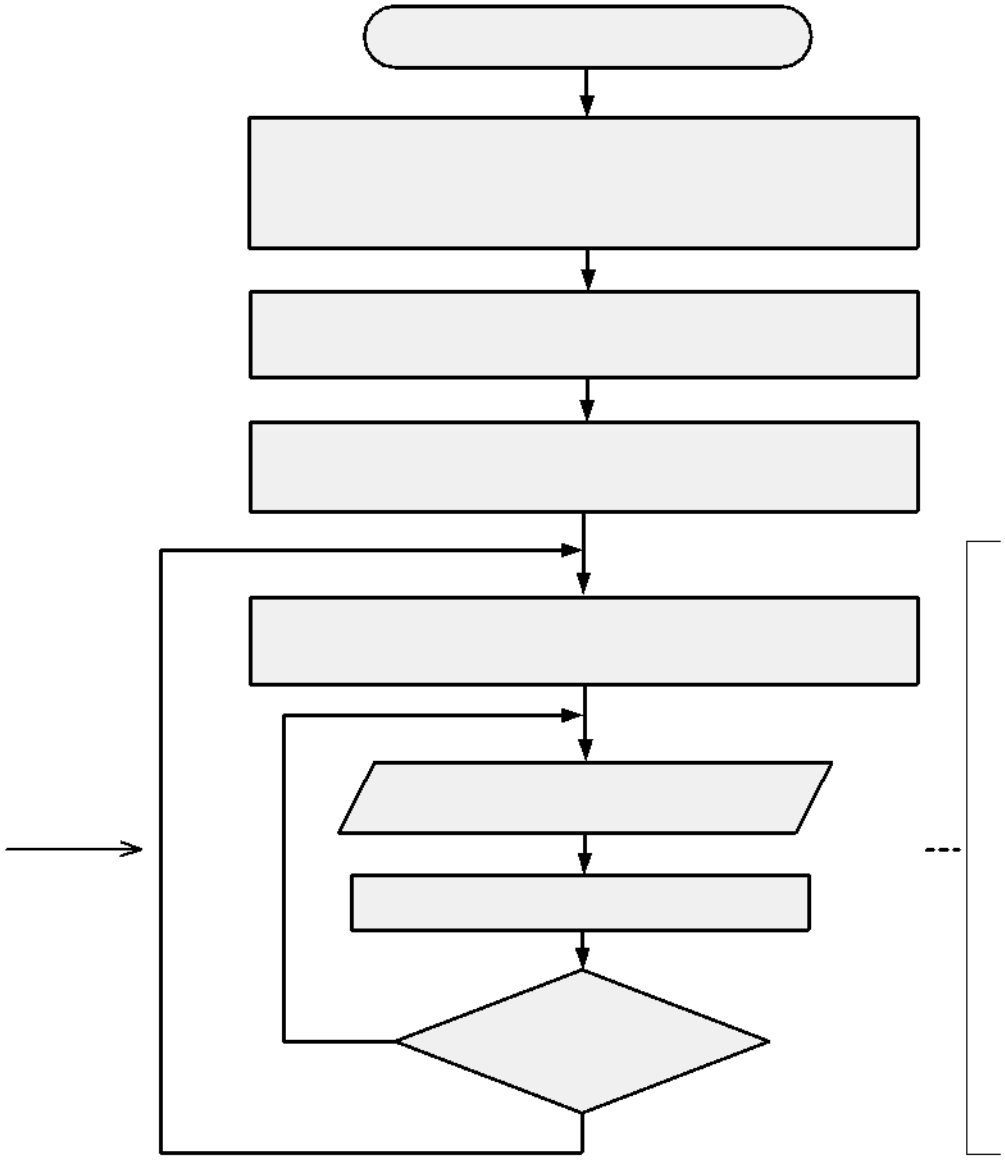
Задача 3. Генератор синуса на основе ЦАП

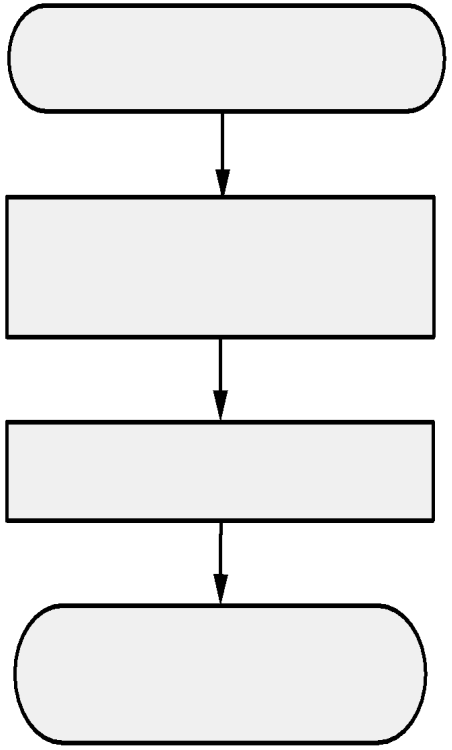
Построить генератор периодического синусоидального сигнала, частота которого увеличивается в 2 раза по каждому клику кнопки. После достижения максимальной частоты (при минимальном количестве отсчетов сигнала за период) клик кнопки приводит к установке минимальной частоты (при максимальном количестве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

При решении задачи используется принцип циклического считывания цифровых отсчётов синуса за период и посылки 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* *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]*.

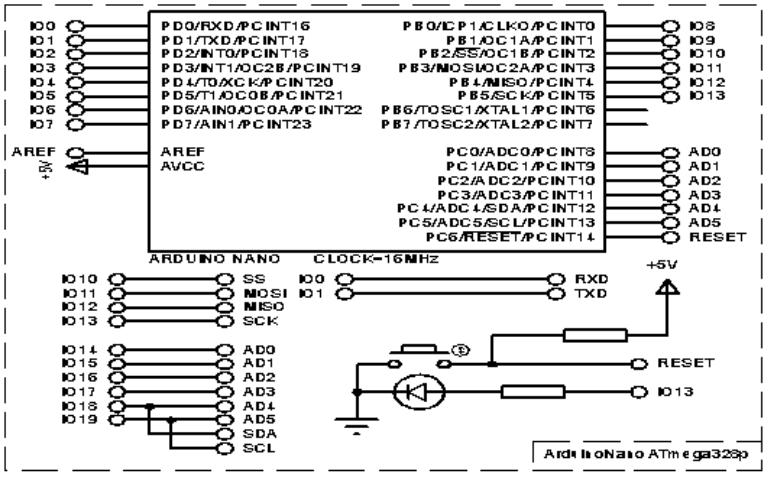
Блок-схема алгоритма

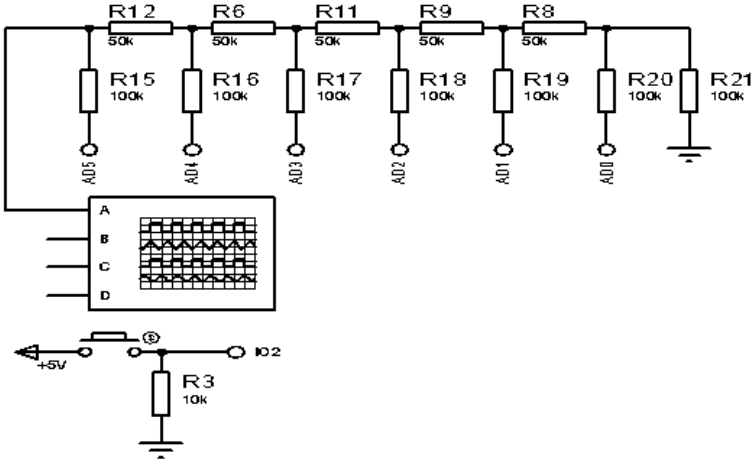




Принципиальные схемы проектов с DAC, программы, макеты

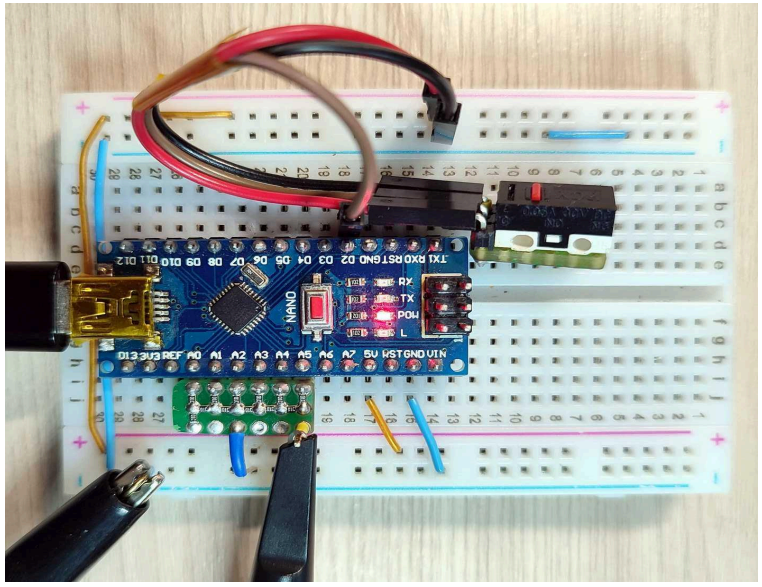
На Arduino nano

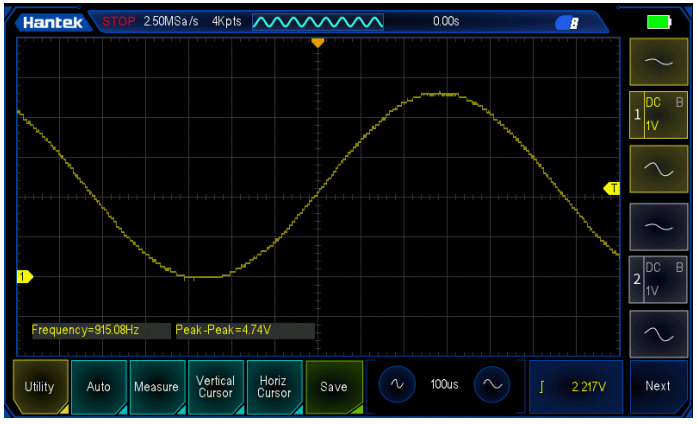


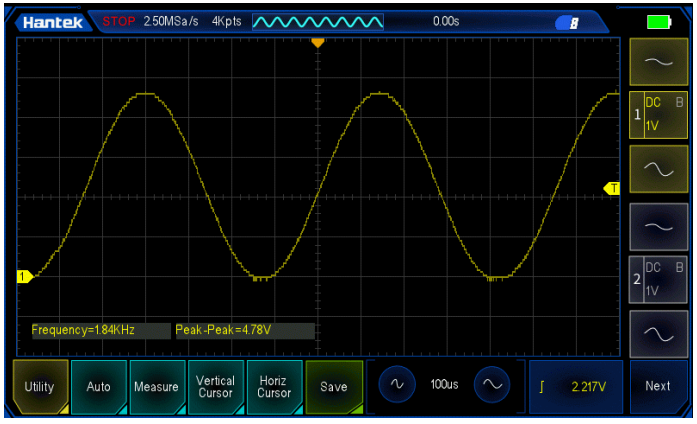


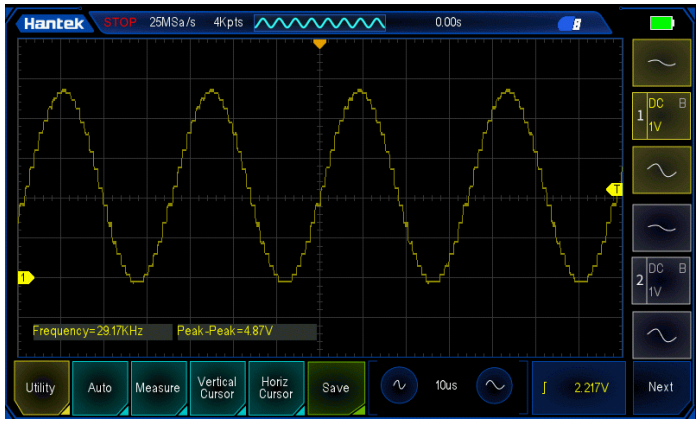
```
#define SAMPLES 1024  //количество отсчётов функции за период
#define nDAC 6        //разрядность используемого внешнего ЦАП на резистивной матрице R-2R
#define TWO_PI 6.28318530717958647692528676655 9

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 (IO2) - 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
}
```

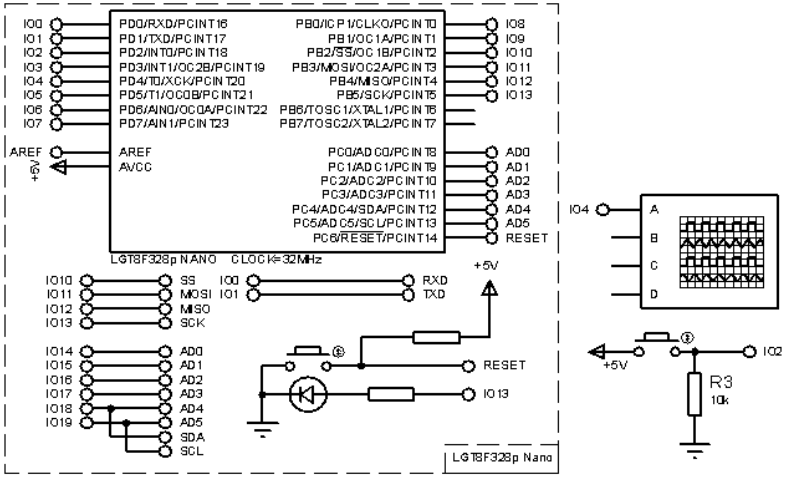






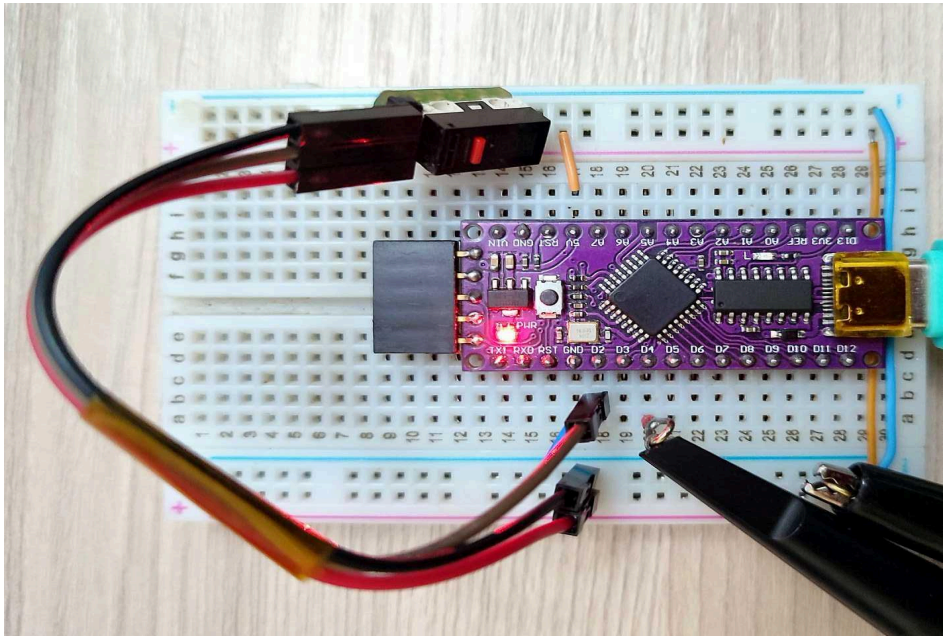

Hantek
STOP
25MSa/s
4Kpts
0.00s
DC
1V
Frequency=29.17KHz
Peak-Peak=4.87V
Utility
Auto
Measure
Vertical Cursor
Horiz Cursor
Save
10us
2.217V
Next

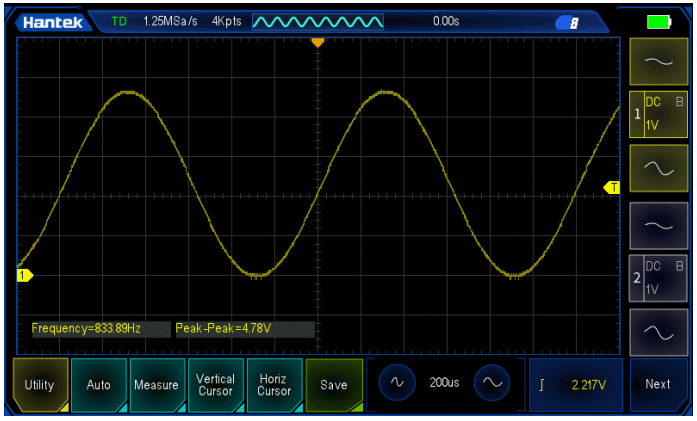
На LGT8F328p nano

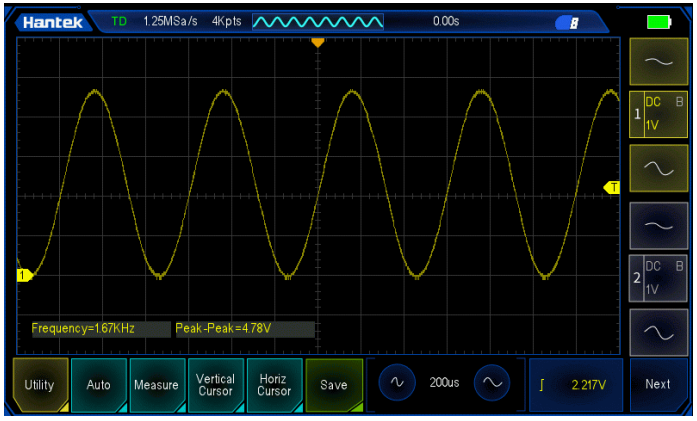


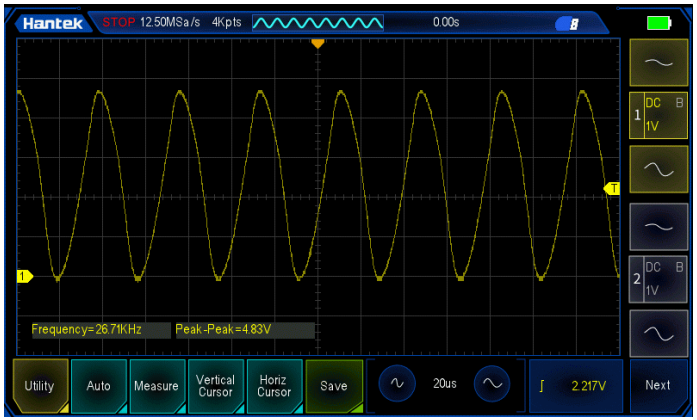
```
#define SAMPLES 1024  //количество отсчётов функции за период
#define nDAC 8        //разрядность внутреннего ЦАП LGT8F328
#define TWO_PI 6.28318530717958647692528676655 9

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 (IO2) - 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
}
```








Hantek
STOP 12.50MSa/s 4Kpts
0.00s
DC
1V
Frequency=26.71KHz
Peak-Peak=4.83V
Utility
Auto
Measure
Vertical Cursor
Horiz Cursor
Save
20us
2.217V
Next

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