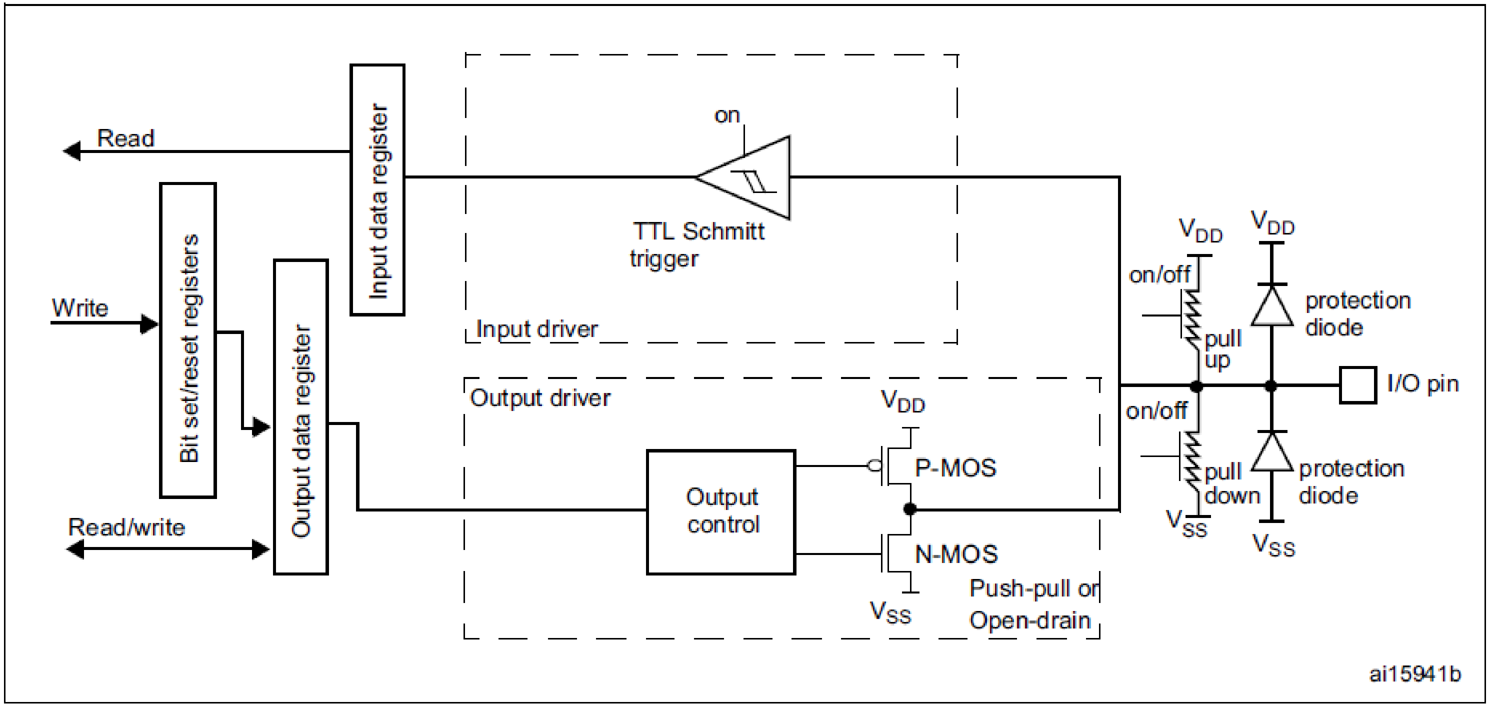
МК и взаимодействие с внешним миром

GPIO


Read
Write
Read/write
Bit set/reset registers
Input data register
Output data register
on
TTL Schmitt trigger
Input driver
Output driver
Output control
V_{DD}
P-MOS
N-MOS
V_{SS}
Push-pull or Open-drain
on/off
pull up
pull down
V_{DD}
V_{SS}
protection diode
protection diode
I/O pin
ai15941b

Характеристики GPIO

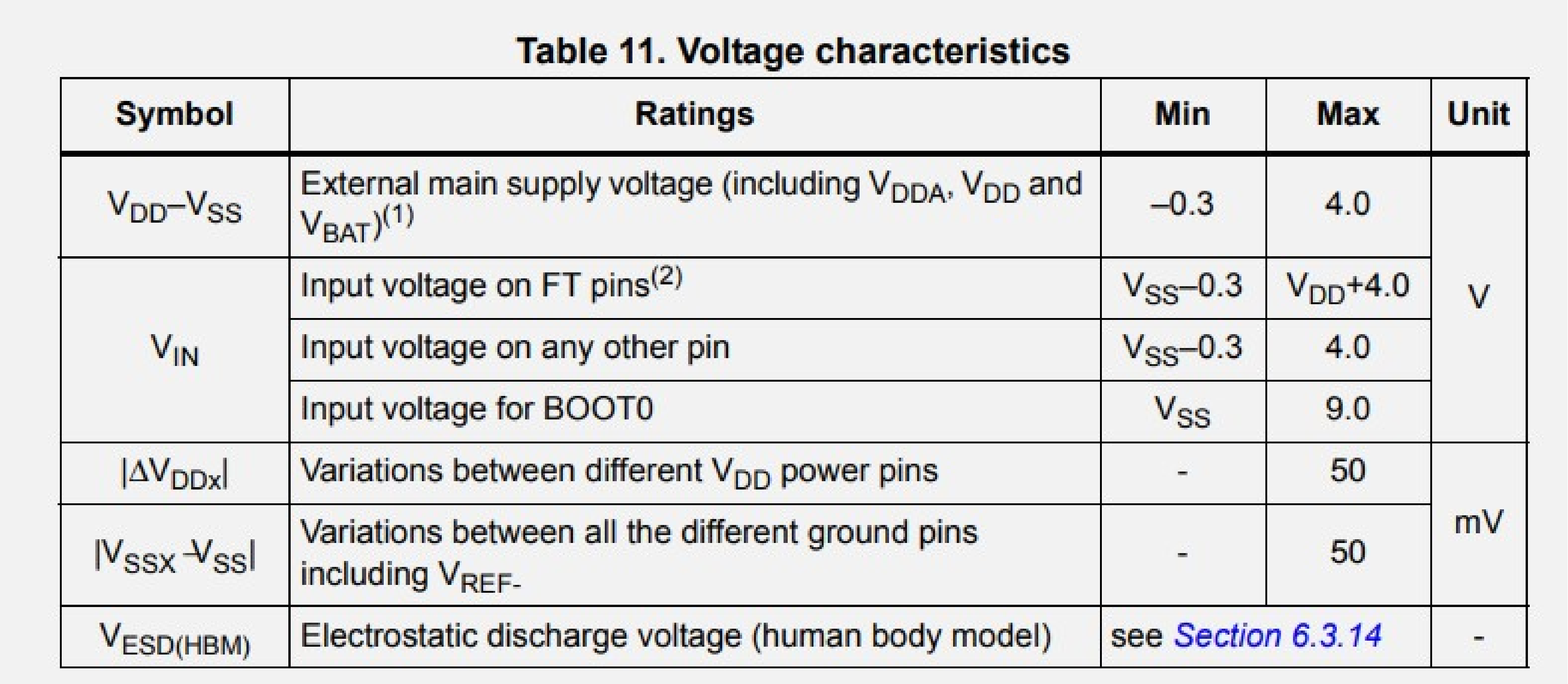


Table 11. Voltage characteristics

Symbol	Ratings	Min	Max	Unit
$V_{DD}–V_{SS}$	External main supply voltage (including V_{DDA}, V_{DD} and V_{BAT})(1)	–0.3	4.0	V
V_{IN}	Input voltage on FT pins(2)	V_{SS}–0.3	V_{DD}+4.0	
	Input voltage on any other pin	V_{SS}–0.3	4.0	
	Input voltage for BOOT0	V_{SS}	9.0	
$\|\Delta V_{DDx}\|$	Variations between different V_{DD} power pins	-	50	mV
$\|V_{SSX} - V_{SS}\|$	Variations between all the different ground pins including V_{REF-}	-	50	
$V_{ESD(HBM)}$	Electrostatic discharge voltage (human body model)	see *Section 6.3.14*		-

Характеристики GPIO

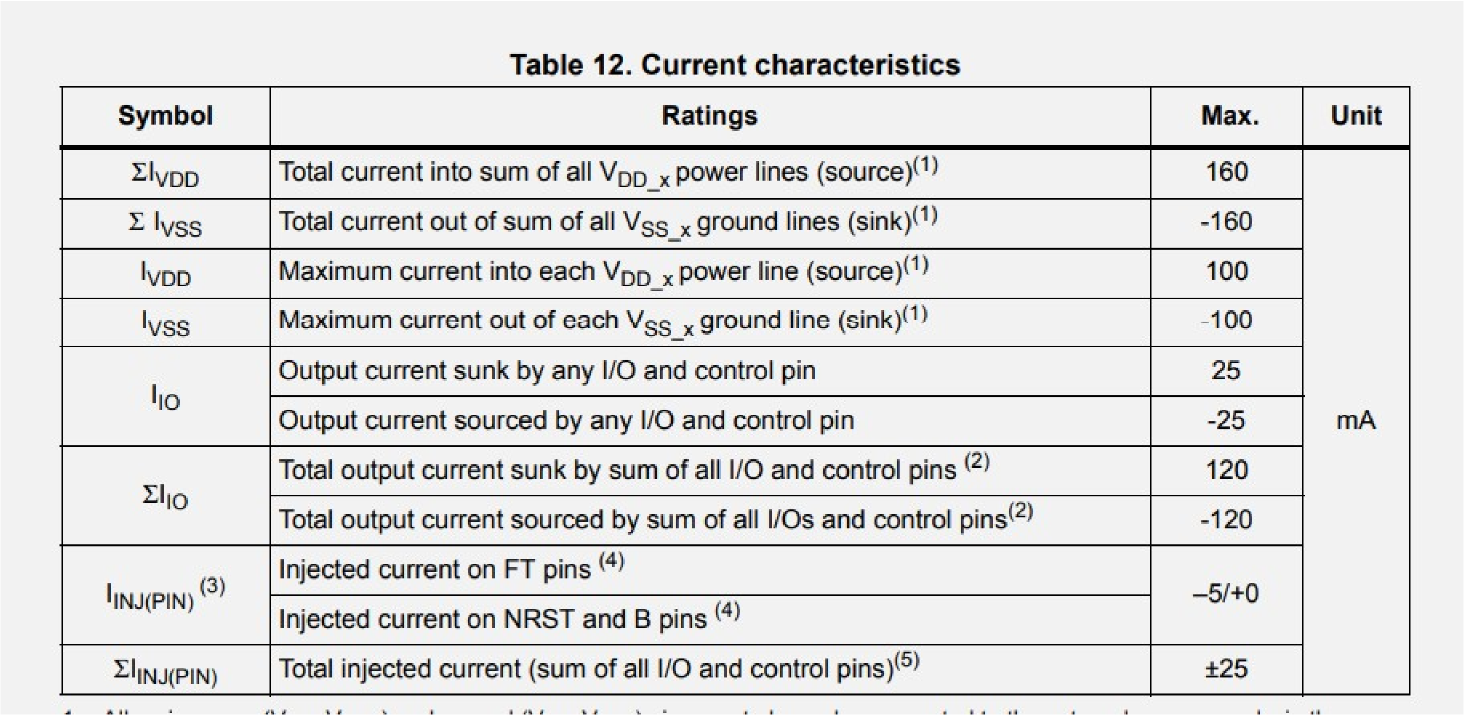
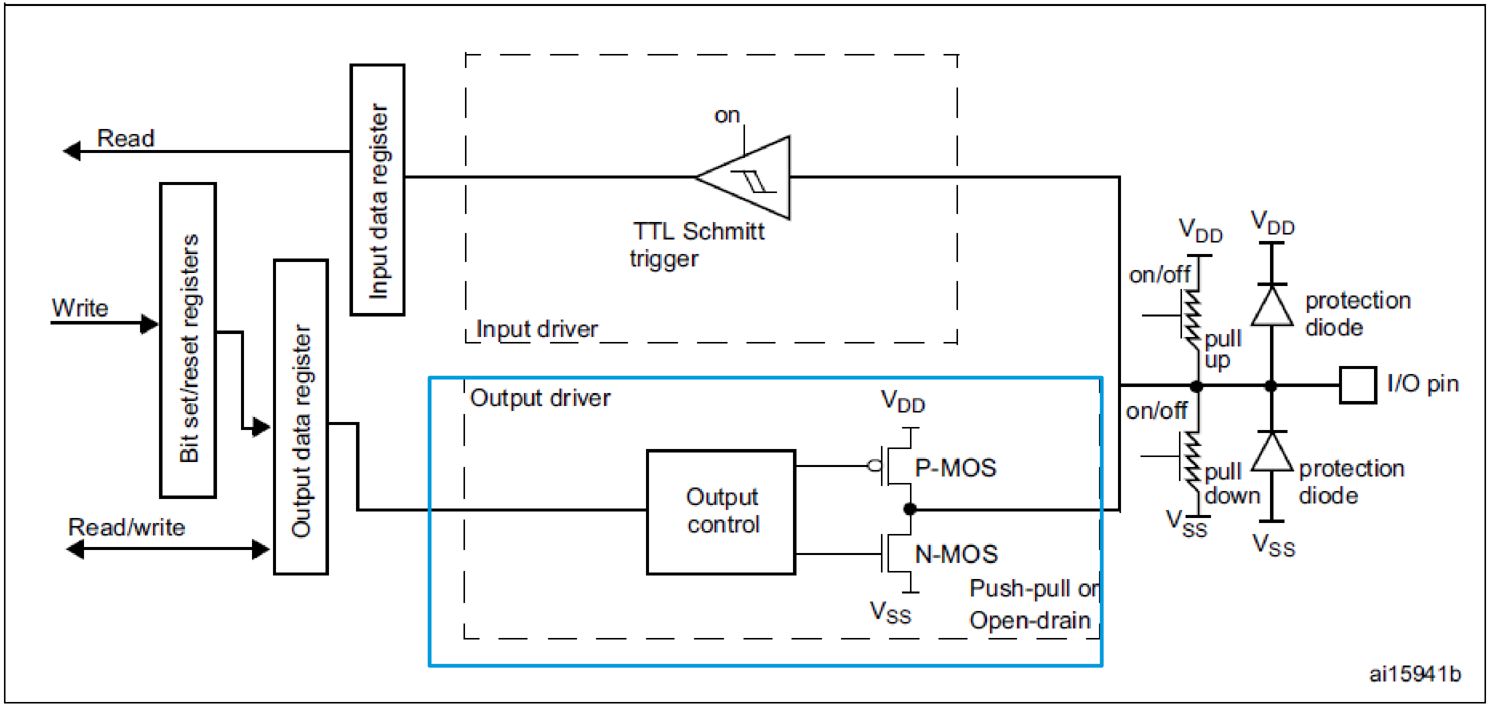


Table 12. Current characteristics

Symbol	Ratings	Max.	Unit
ΣI_{VDD}	Total current into sum of all V_{DD_x} power lines (source)(1)	160	mA
$\Sigma\ I_{VSS}$	Total current out of sum of all V_{SS_x} ground lines (sink)(1)	-160	
I_{VDD}	Maximum current into each V_{DD_x} power line (source)(1)	100	
I_{VSS}	Maximum current out of each V_{SS_x} ground line (sink)(1)	-100	
I_{IO}	Output current sunk by any I/O and control pin	25	
	Output current sourced by any I/O and control pin	-25	
ΣI_{IO}	Total output current sunk by sum of all I/O and control pins (2)	120	
	Total output current sourced by sum of all I/Os and control pins(2)	-120	
$I_{INJ(PIN)}$ (3)	Injected current on FT pins (4)	–5/+0	
	Injected current on NRST and B pins (4)		
$\Sigma I_{INJ(PIN)}$	Total injected current (sum of all I/O and control pins)(5)	±25	

Запись в GPIO



Регистры управления GPIO

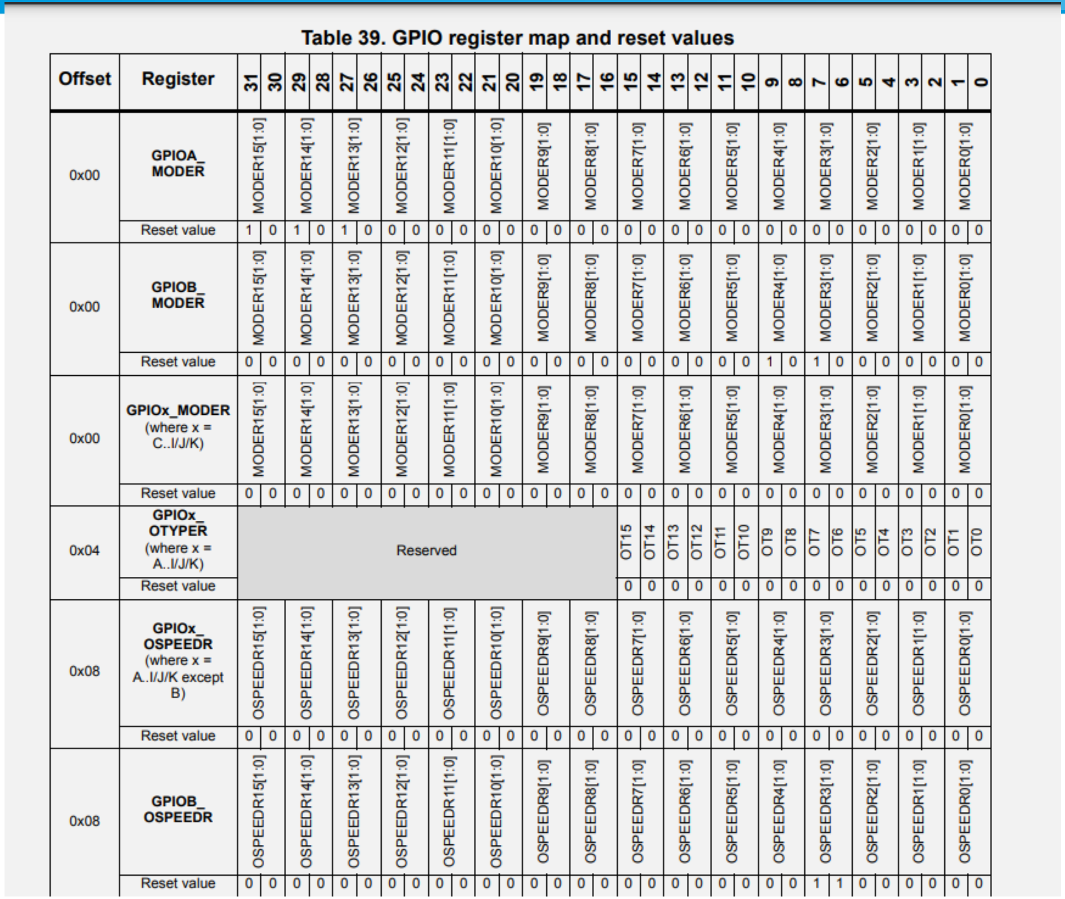
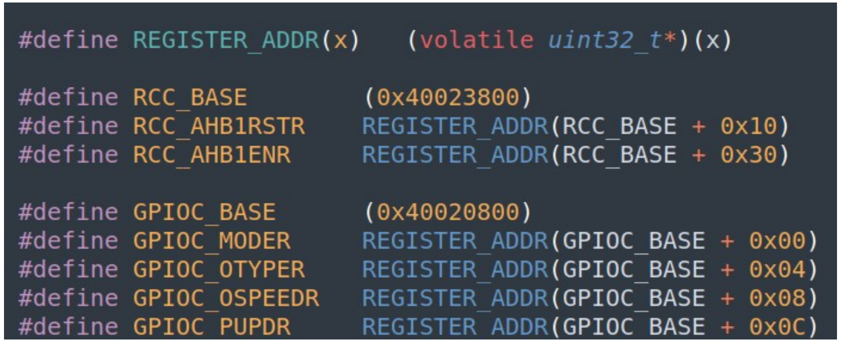
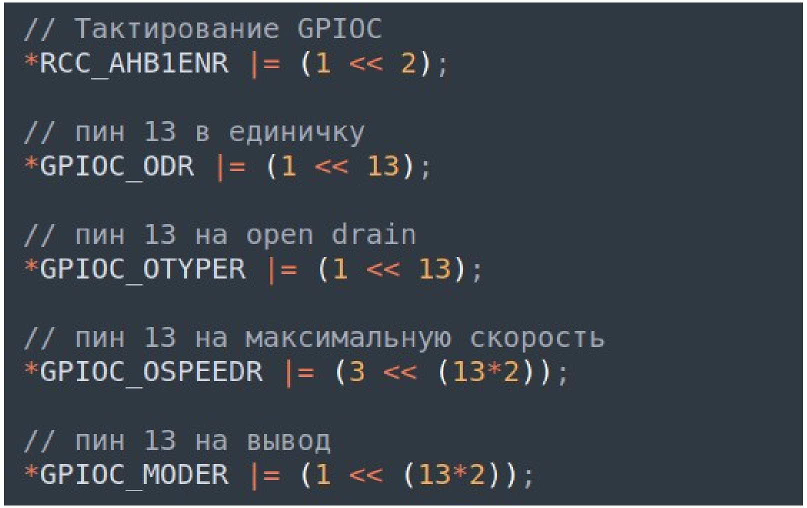


Table 39. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOA_ MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	GPIOB_ MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x00	GPIOx_MODER (where x = C..I/J/K)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	GPIOx_ OTYPER (where x = A..I/J/K)	Reserved																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOx_ OSPEEDR (where x = A..I/J/K except B)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOB_ OSPEEDR	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

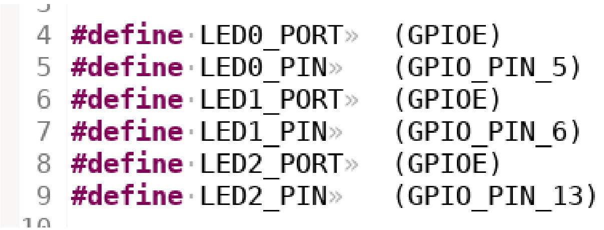


```c
#define REGISTER_ADDR(x)   (volatile uint32_t*)(x)

#define RCC_BASE          (0x40023800)
#define RCC_AHB1RSTR      REGISTER_ADDR(RCC_BASE + 0x10)
#define RCC_AHB1ENR       REGISTER_ADDR(RCC_BASE + 0x30)

#define GPIOC_BASE        (0x40020800)
#define GPIOC_MODER       REGISTER_ADDR(GPIOC_BASE + 0x00)
#define GPIOC_OTYPER      REGISTER_ADDR(GPIOC_BASE + 0x04)
#define GPIOC_OSPEEDR     REGISTER_ADDR(GPIOC_BASE + 0x08)
#define GPIOC_PUPDR       REGISTER_ADDR(GPIOC_BASE + 0x0C)
```

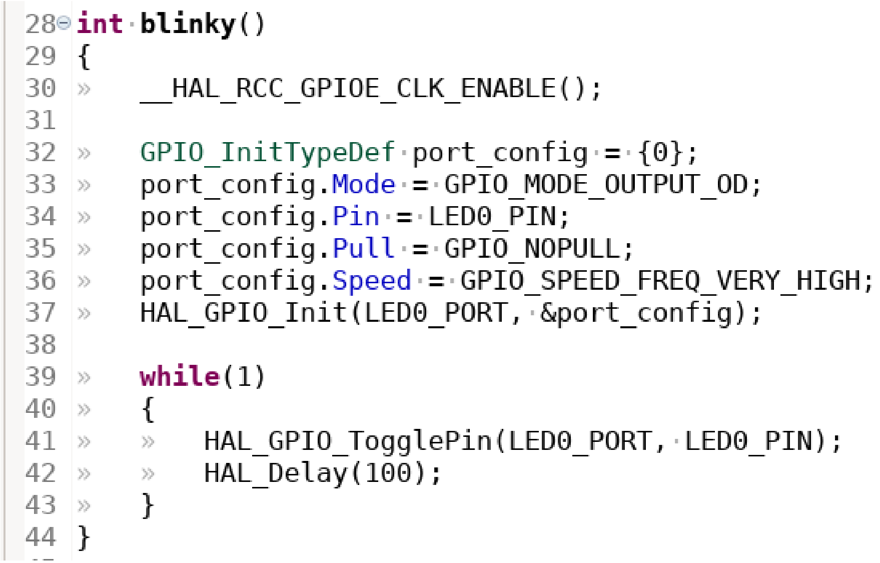


```c
// Тактирование GPIOC
*RCC_AHB1ENR |= (1 << 2);

// пин 13 в единичку
*GPIOC_ODR |= (1 << 13);

// пин 13 на open drain
*GPIOC_OTYPER |= (1 << 13);

// пин 13 на максимальную скорость
*GPIOC_OSPEEDR |= (3 << (13*2));

// пин 13 на вывод
*GPIOC_MODER |= (1 << (13*2));
```

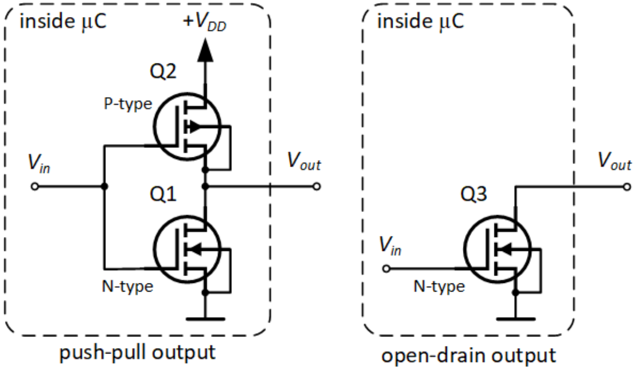
Blinky (Hello world)



```c
#define LED0_PORT   (GPIOE)
#define LED0_PIN    (GPIO_PIN_5)
#define LED1_PORT   (GPIOE)
#define LED1_PIN    (GPIO_PIN_6)
#define LED2_PORT   (GPIOE)
#define LED2_PIN    (GPIO_PIN_13)
```

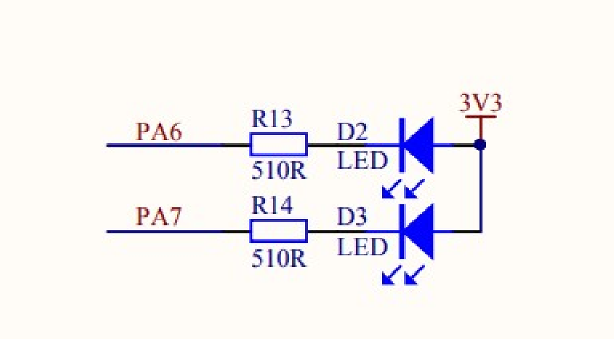


```c
int blinky()
{
    __HAL_RCC_GPIOE_CLK_ENABLE();

    GPIO_InitTypeDef port_config = {0};
    port_config.Mode = GPIO_MODE_OUTPUT_OD;
    port_config.Pin = LED0_PIN;
    port_config.Pull = GPIO_NOPULL;
    port_config.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    HAL_GPIO_Init(LED0_PORT, &port_config);

    while(1)
    {
        HAL_GPIO_TogglePin(LED0_PORT, LED0_PIN);
        HAL_Delay(100);
    }
}
```

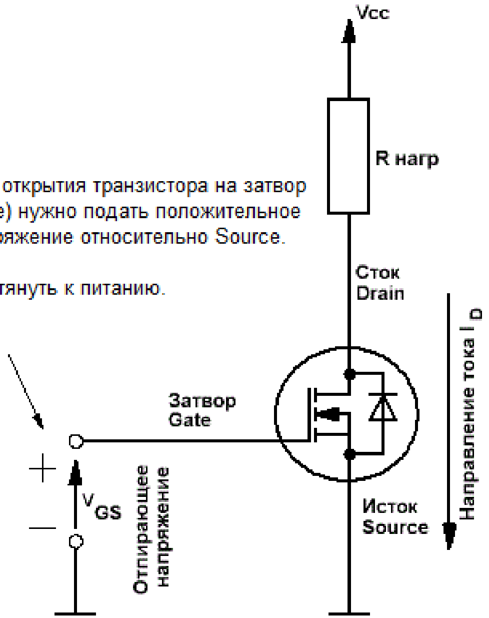


push-pull output

open-drain output



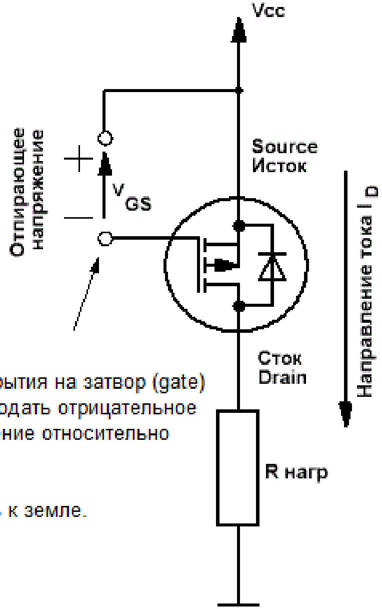
Управление мощной нагрузкой



Для открытия транзистора на затвор (gate) нужно подать положительное напряжение относительно Source.

Подтянуть к питанию.

N-тип

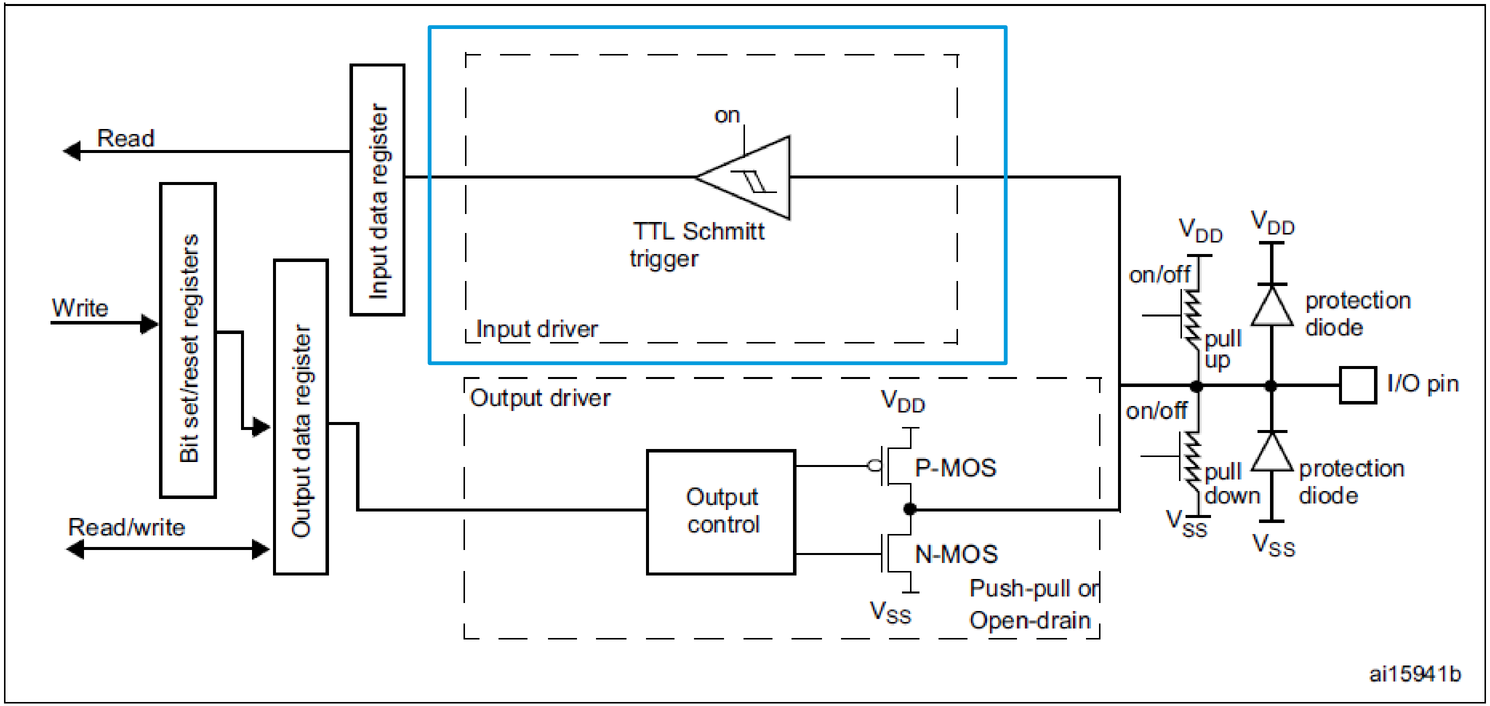


Для открытия на затвор (gate) нужно подать отрицательное напряжение относительно Source.

Прижать к земле.

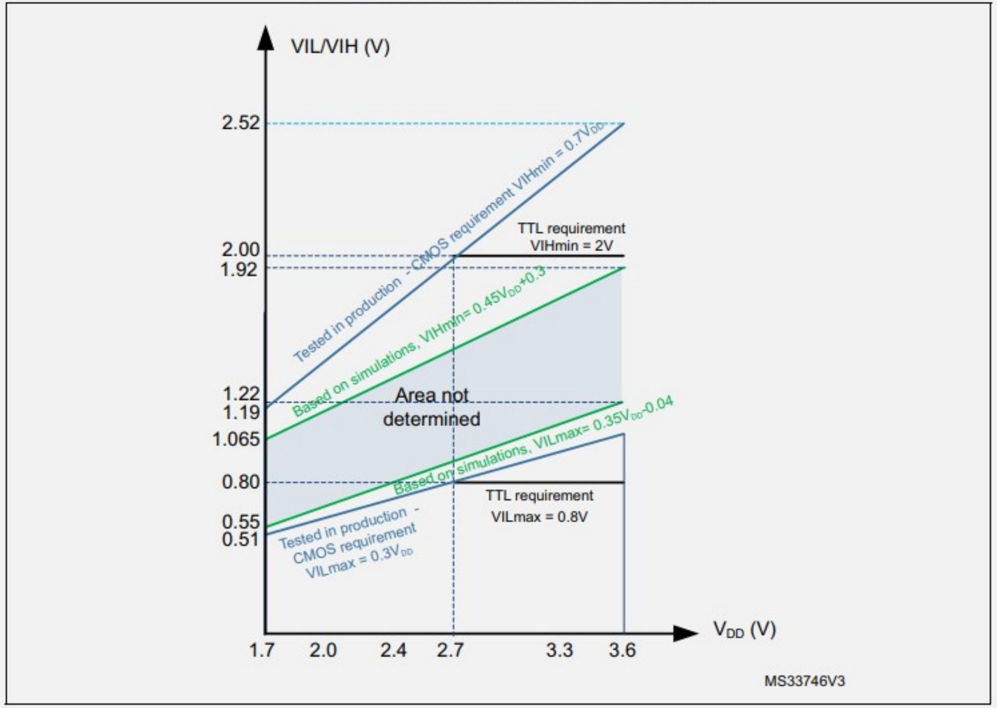
P-тип

Чтение GPIO

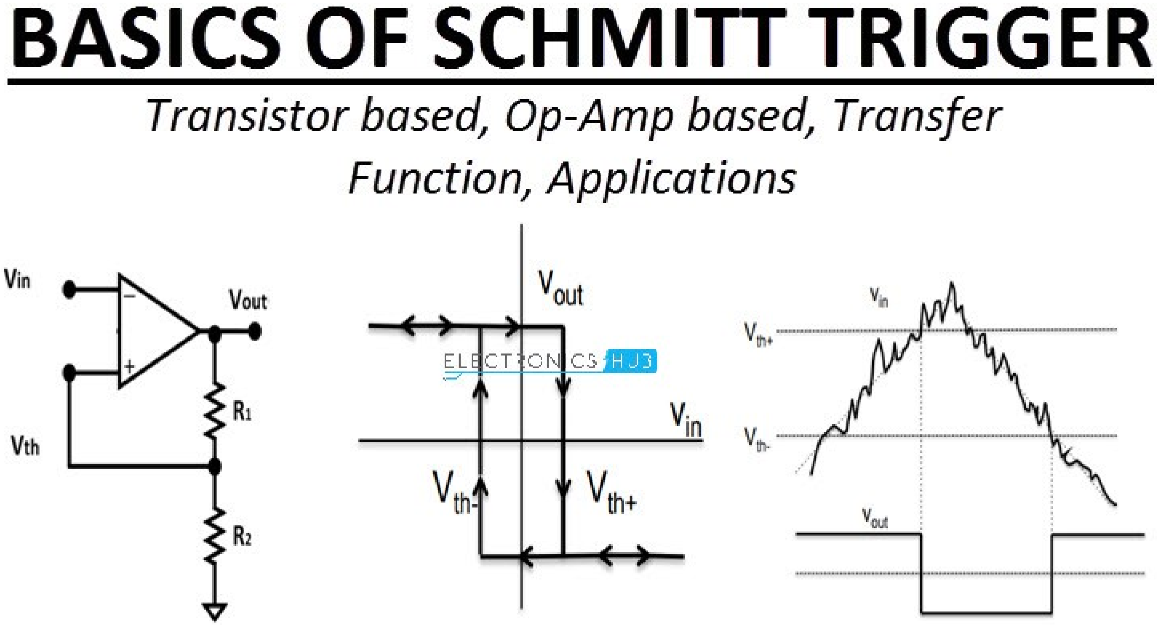

Read
Write
Read/write
Bit set/reset registers
Input data register
Output data register
on
TTL Schmitt trigger
Input driver
Output driver
Output control
V_{DD}
P-MOS
N-MOS
V_{SS}
Push-pull or Open-drain
V_{DD}
on/off
pull up
on/off
pull down
V_{SS}
V_{DD}
protection diode
protection diode
V_{SS}
I/O pin
ai15941b

Чтение напряжения с GPIO

Figure 30. FT I/O input characteristics



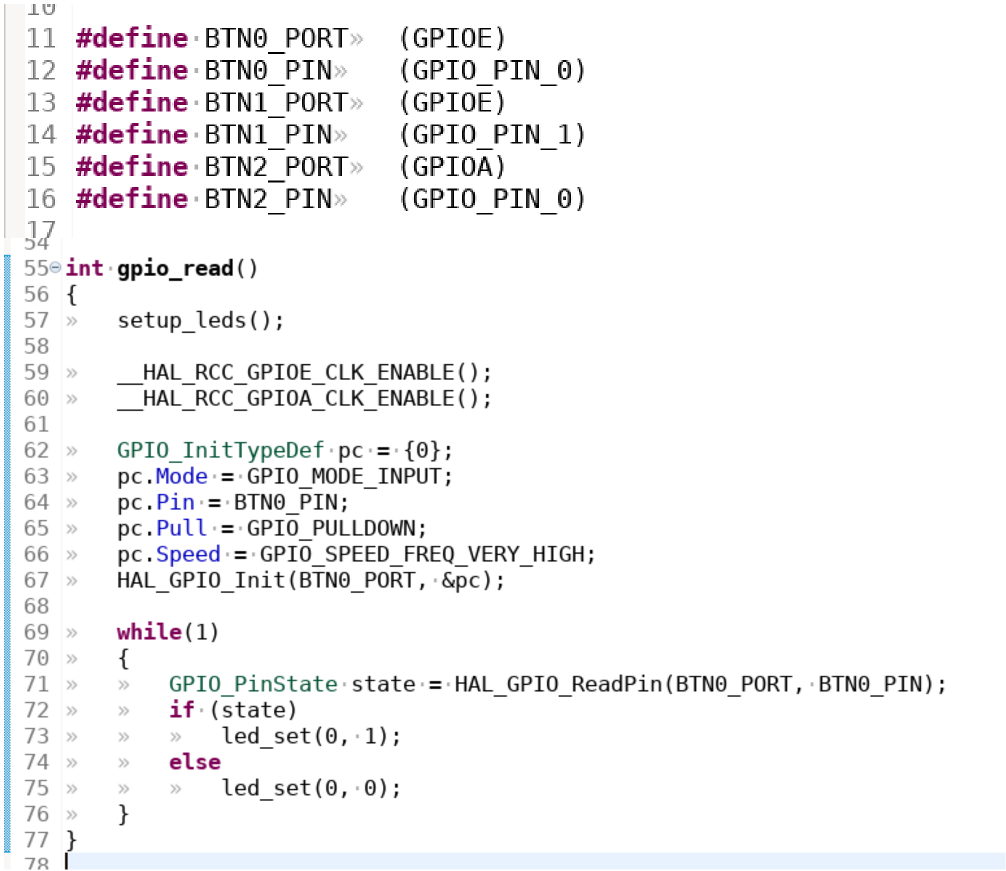
Триггер Шмидта



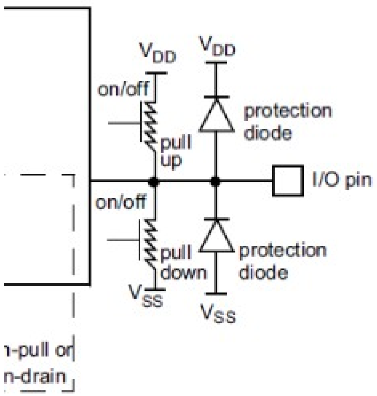
BASICS OF SCHMITT TRIGGER

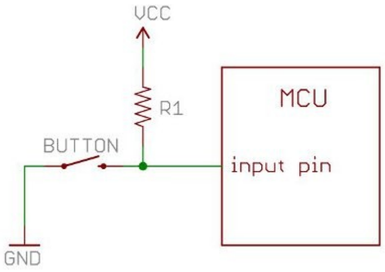
Transistor based, Op-Amp based, Transfer Function, Applications

Pull-up или pull-down. Подтяжки



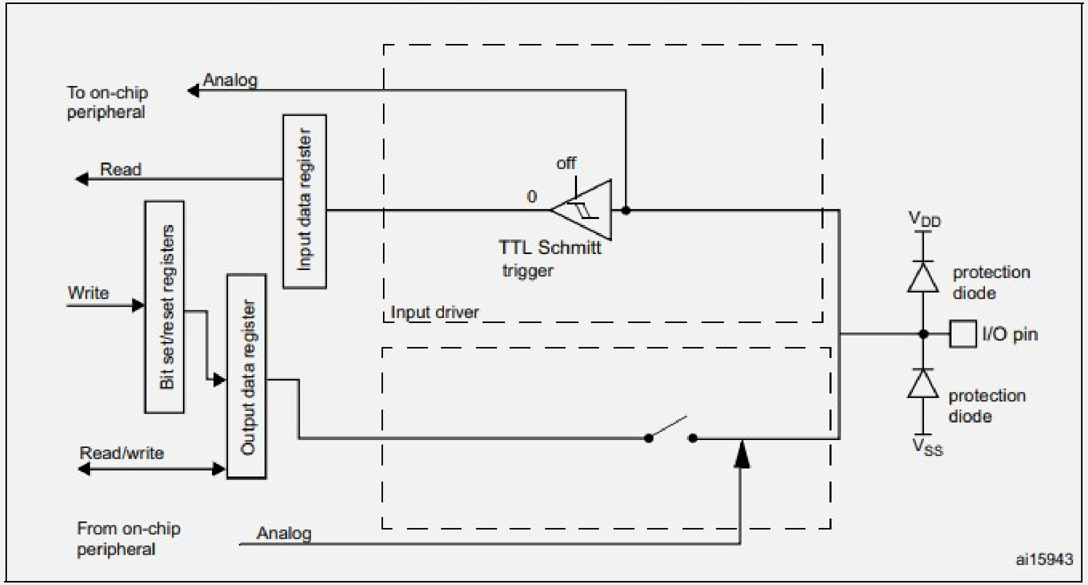
```c
#define BTN0_PORT   (GPIOE)
#define BTN0_PIN    (GPIO_PIN_0)
#define BTN1_PORT   (GPIOE)
#define BTN1_PIN    (GPIO_PIN_1)
#define BTN2_PORT   (GPIOA)
#define BTN2_PIN    (GPIO_PIN_0)

int gpio_read()
{
    setup_leds();

    __HAL_RCC_GPIOE_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();

    GPIO_InitTypeDef pc = {0};
    pc.Mode = GPIO_MODE_INPUT;
    pc.Pin = BTN0_PIN;
    pc.Pull = GPIO_PULLDOWN;
    pc.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    HAL_GPIO_Init(BTN0_PORT, &pc);

    while(1)
    {
        GPIO_PinState state = HAL_GPIO_ReadPin(BTN0_PORT, BTN0_PIN);
        if (state)
            led_set(0, 1);
        else
            led_set(0, 0);
    }
}
```

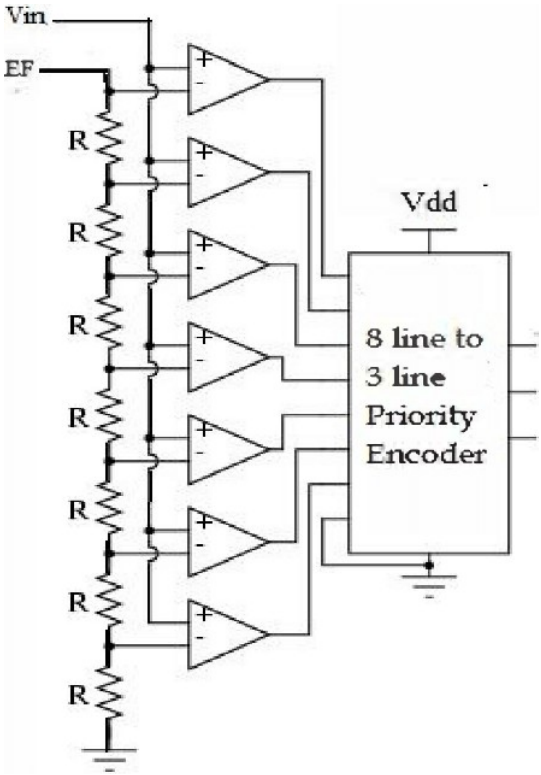




Аналого-цифровой преобразователь (АЦП)

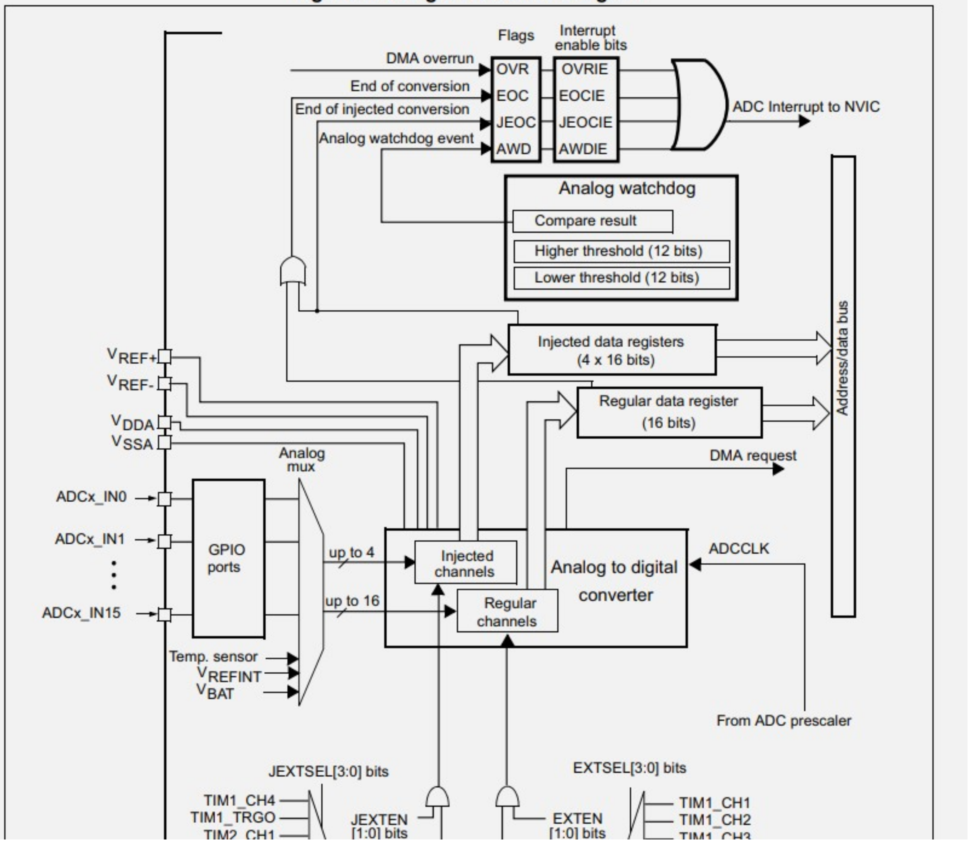
Figure 31. High impedance-analog configuration



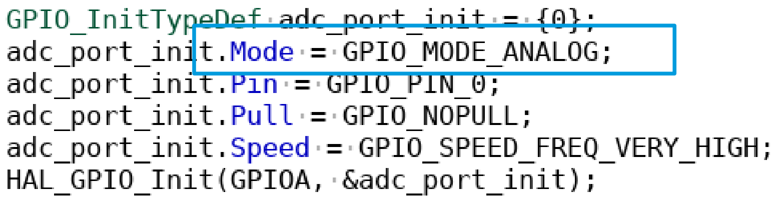


Аналого-цифровой преобразователь (АЦП)

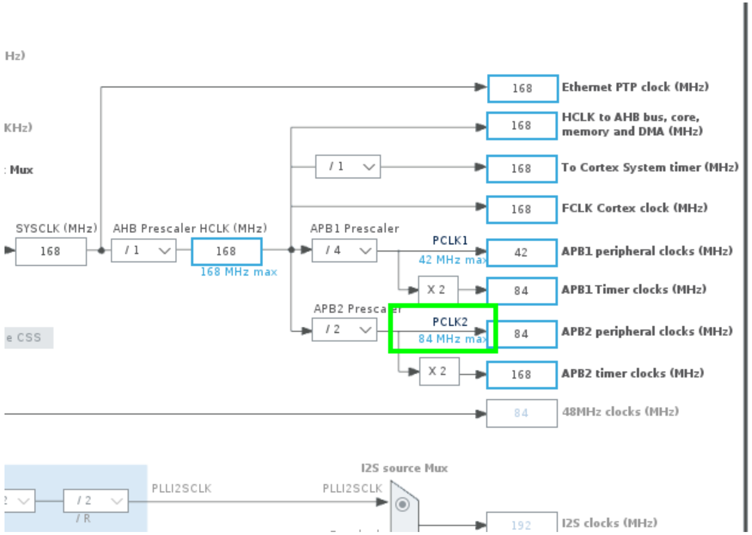
Figure 44. Single ADC block diagram

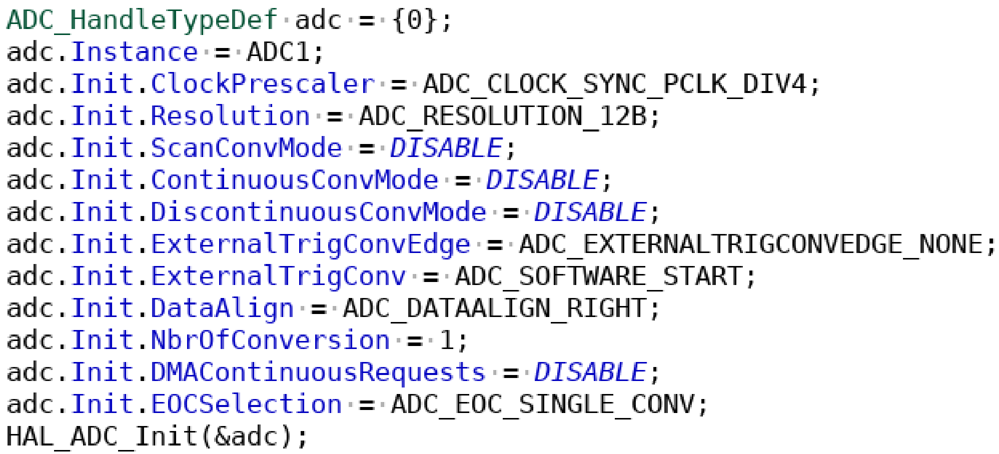


Аналого-цифровой преобразователь (АЦП)

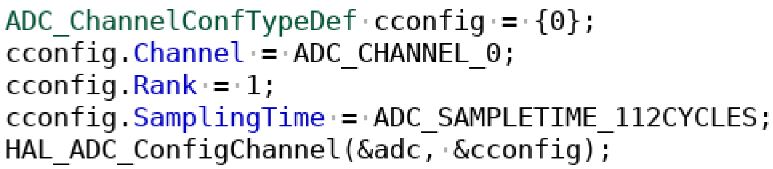


```
GPIO_InitTypeDef adc_port_init = {0};
adc_port_init.Mode = GPIO_MODE_ANALOG;
adc_port_init.Pin = GPIO_PIN_0;
adc_port_init.Pull = GPIO_NOPULL;
adc_port_init.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
HAL_GPIO_Init(GPIOA, &adc_port_init);
```





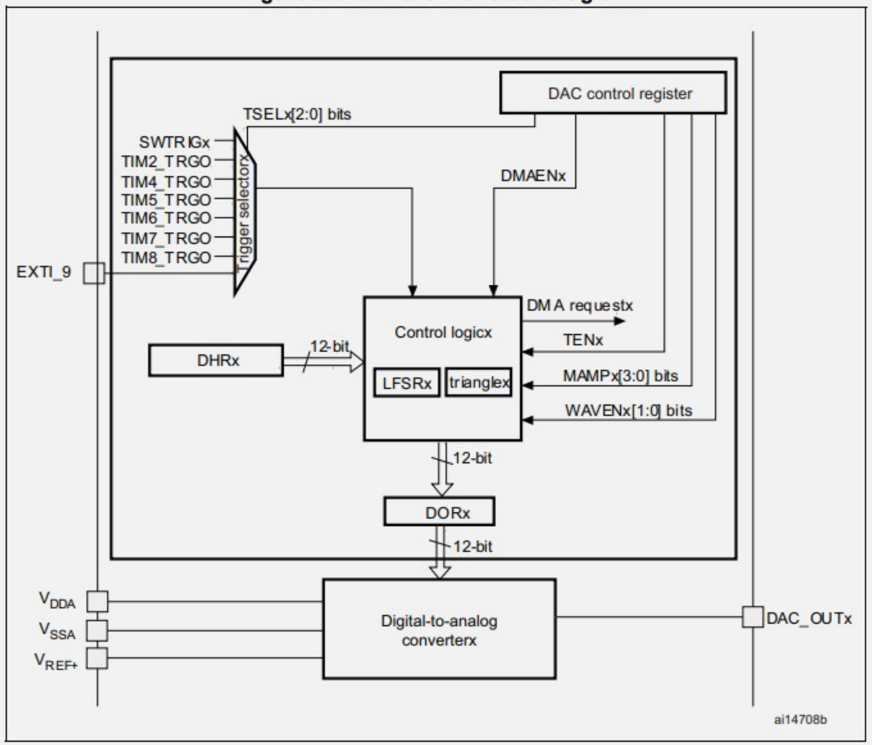
```
ADC_HandleTypeDef adc = {0};
adc.Instance = ADC1;
adc.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
adc.Init.Resolution = ADC_RESOLUTION_12B;
adc.Init.ScanConvMode = DISABLE;
adc.Init.ContinuousConvMode = DISABLE;
adc.Init.DiscontinuousConvMode = DISABLE;
adc.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
adc.Init.ExternalTrigConv = ADC_SOFTWARE_START;
adc.Init.DataAlign = ADC_DATAALIGN_RIGHT;
adc.Init.NbrOfConversion = 1;
adc.Init.DMAContinuousRequests = DISABLE;
adc.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
HAL_ADC_Init(&adc);
```

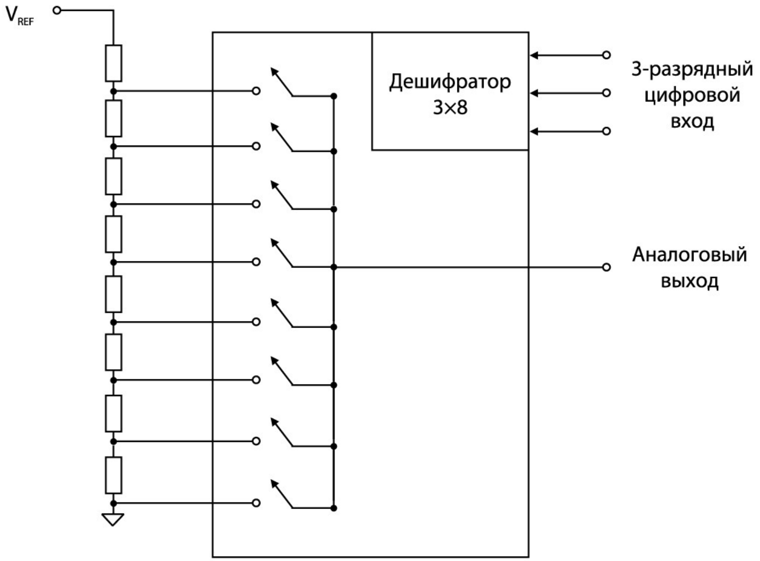


```
ADC_ChannelConfTypeDef cconfig = {0};
cconfig.Channel = ADC_CHANNEL_0;
cconfig.Rank = 1;
cconfig.SamplingTime = ADC_SAMPLETIME_112CYCLES;
HAL_ADC_ConfigChannel(&adc, &cconfig);
```

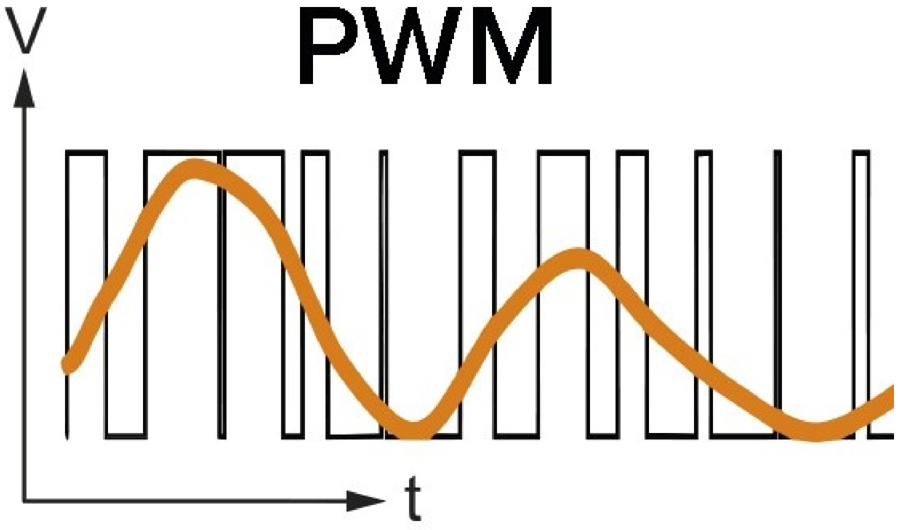
Цифро-аналоговый преобразователь (ЦАП)

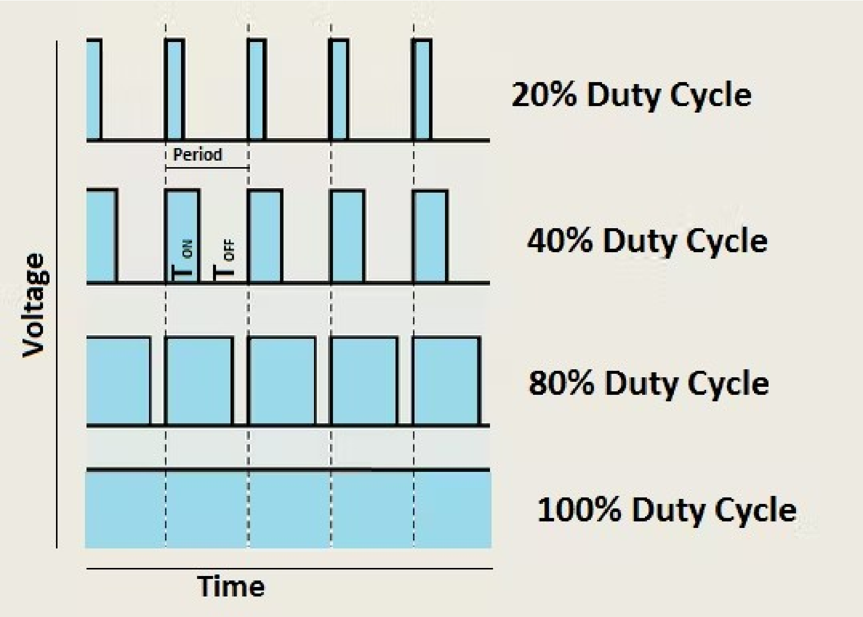
Figure 64. DAC channel block diagram



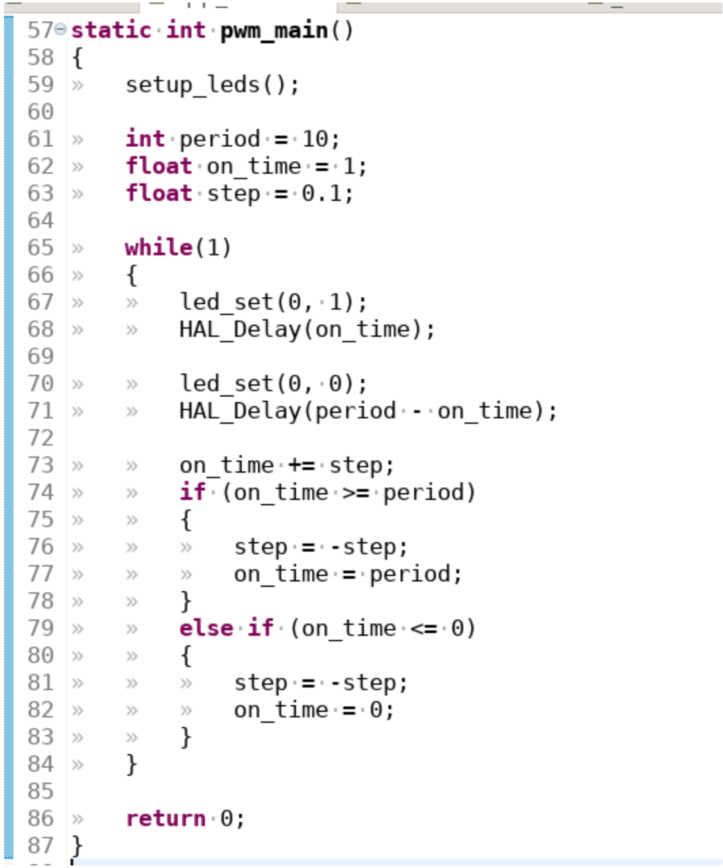


Широтно-имульсная модуляция (ШИМ/PWM)

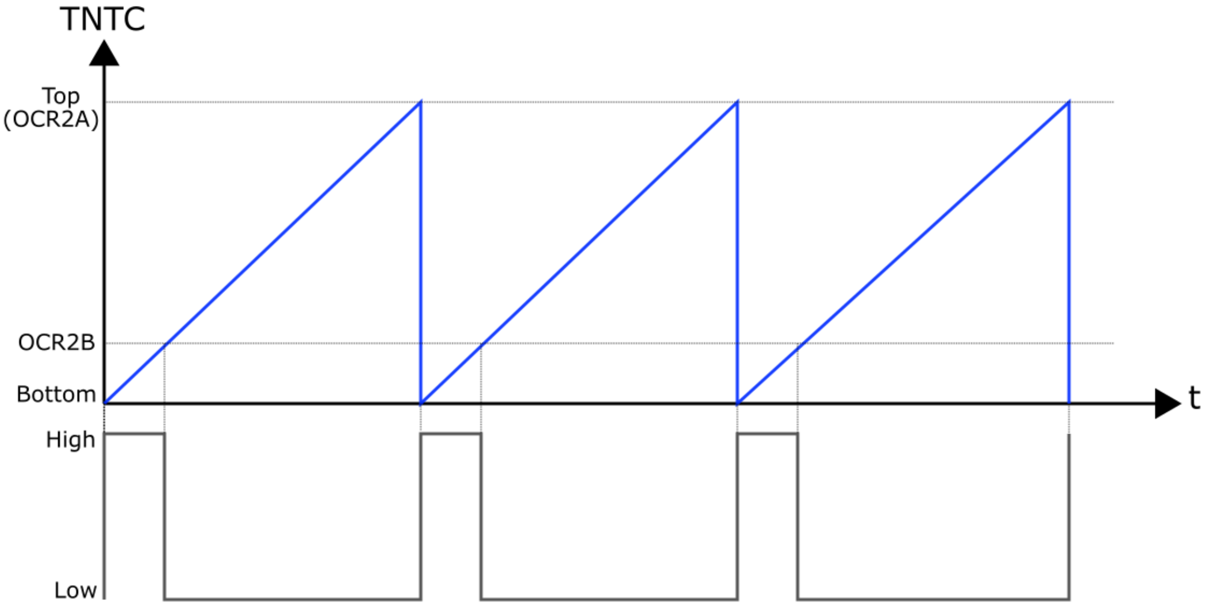




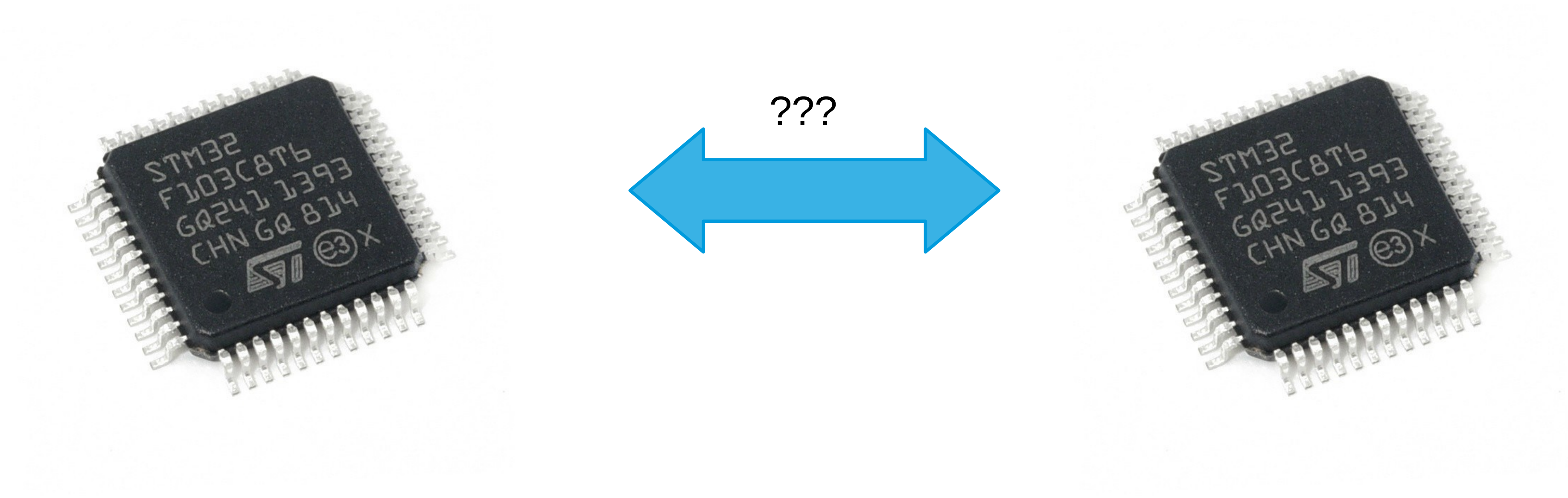
Широтно-имульсная модуляция (ШИМ/PWM)



```c
static int pwm_main()
{
    setup_leds();

    int period = 10;
    float on_time = 1;
    float step = 0.1;

    while(1)
    {
        led_set(0, 1);
        HAL_Delay(on_time);

        led_set(0, 0);
        HAL_Delay(period - on_time);

        on_time += step;
        if (on_time >= period)
        {
            step = -step;
            on_time = period;
        }
        else if (on_time <= 0)
        {
            step = -step;
            on_time = 0;
        }
    }

    return 0;
}
```

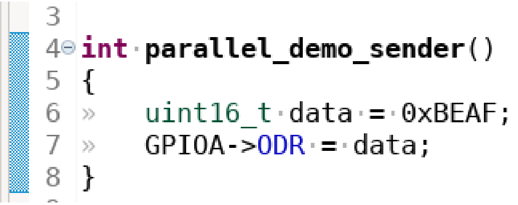


Обмен цифровой информацией

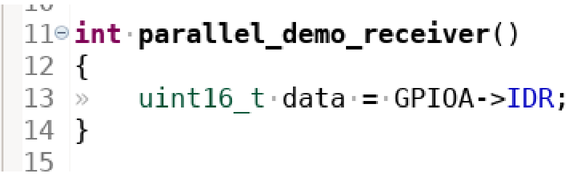


???

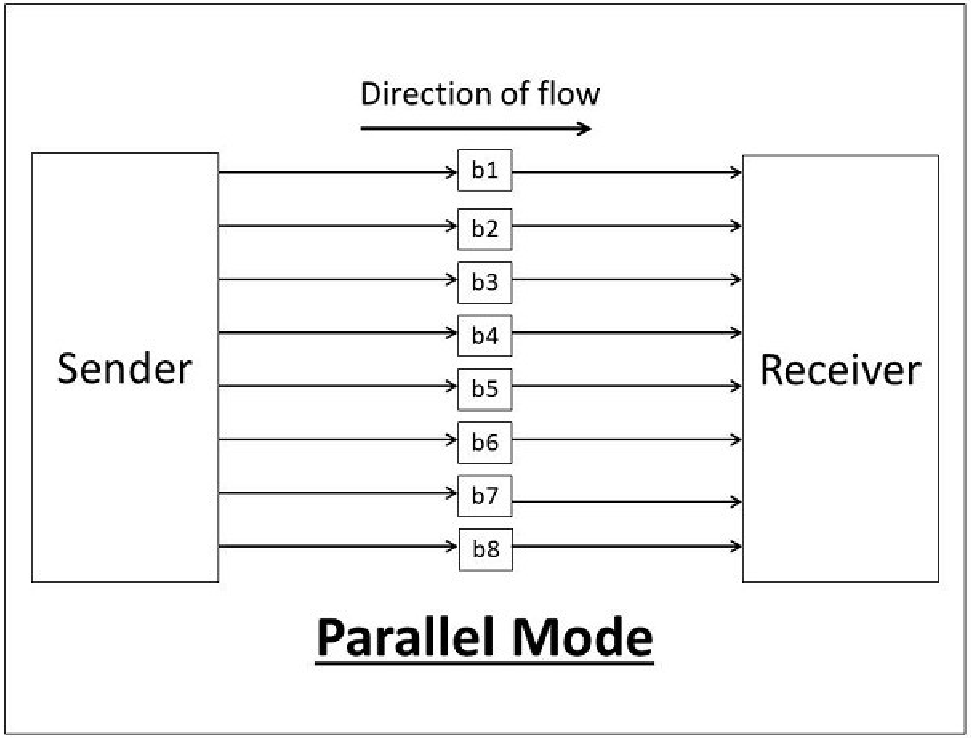
Параллельный порт



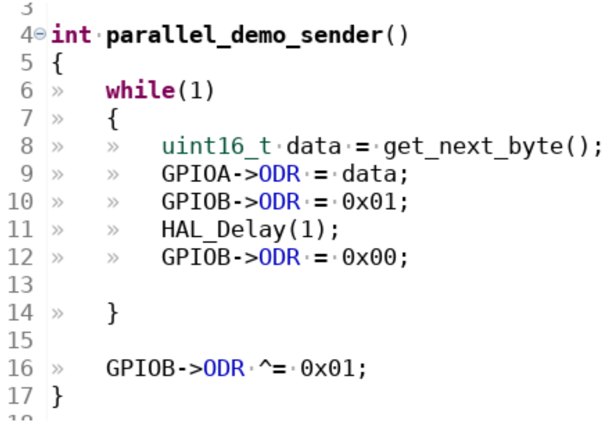
```
int parallel_demo_sender()
{
    uint16_t data = 0xBEAF;
    GPIOA->ODR = data;
}
```



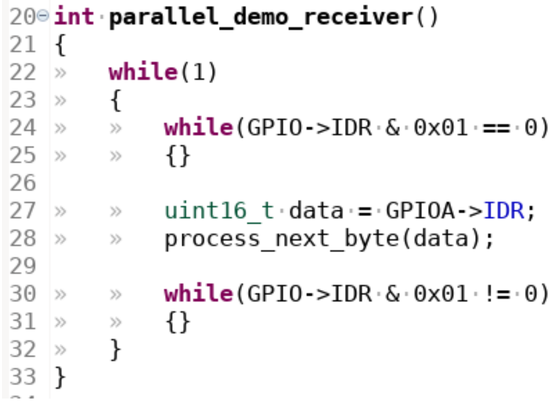
```
int parallel_demo_receiver()
{
    uint16_t data = GPIOA->IDR;
}
```



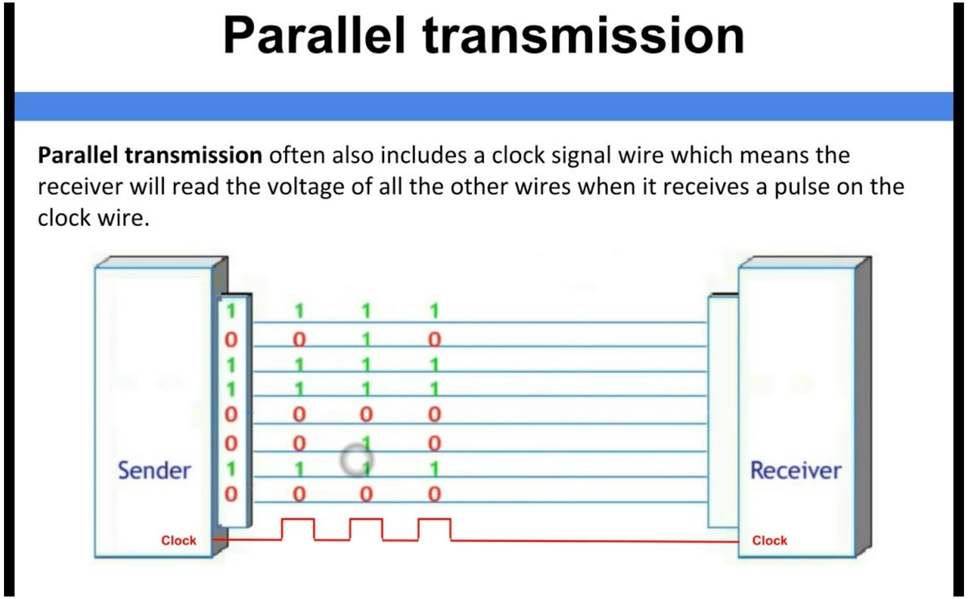
Параллельный порт с синхронизацией



```c
int parallel_demo_sender()
{
	while(1)
	{
		uint16_t data = get_next_byte();
		GPIOA->ODR = data;
		GPIOB->ODR = 0x01;
		HAL_Delay(1);
		GPIOB->ODR = 0x00;

	}

	GPIOB->ODR ^= 0x01;
}
```



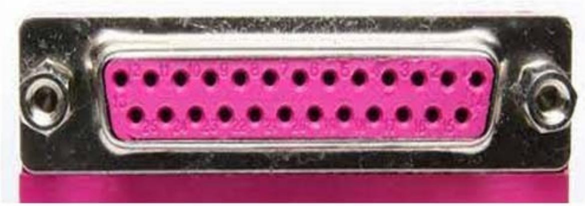
```c
int parallel_demo_receiver()
{
	while(1)
	{
		while(GPIO->IDR & 0x01 == 0)
		{}

		uint16_t data = GPIOA->IDR;
		process_next_byte(data);

		while(GPIO->IDR & 0x01 != 0)
		{}
	}
}
```

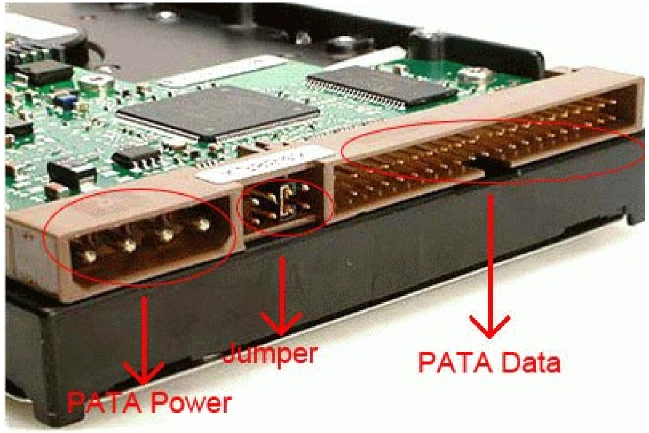


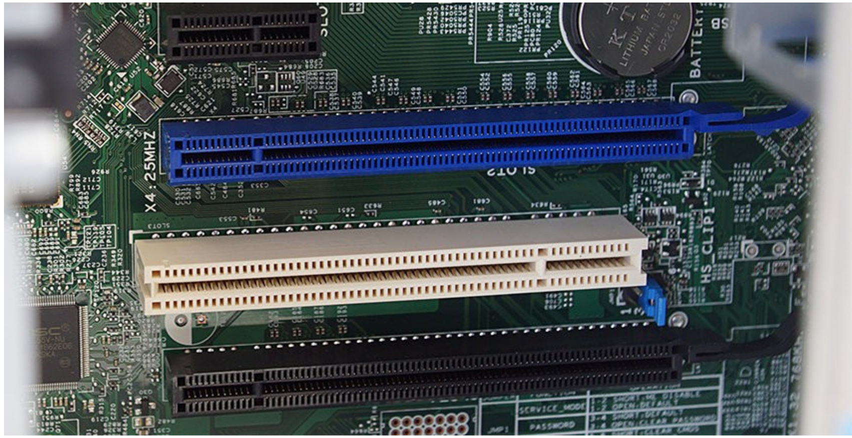
Parallel transmission

Parallel transmission often also includes a clock signal wire which means the receiver will read the voltage of all the other wires when it receives a pulse on the clock wire.

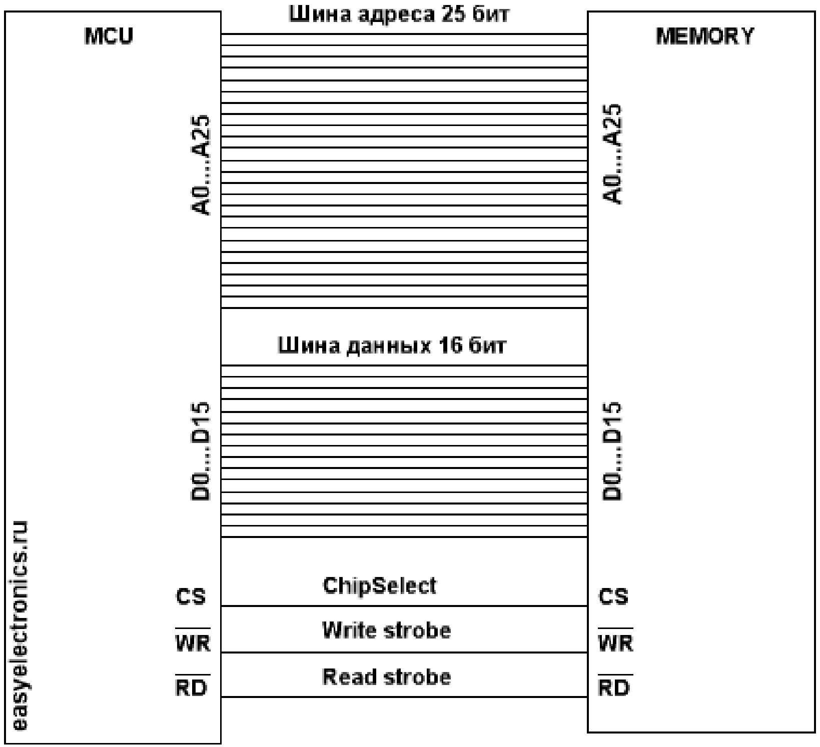
Параллельный порт



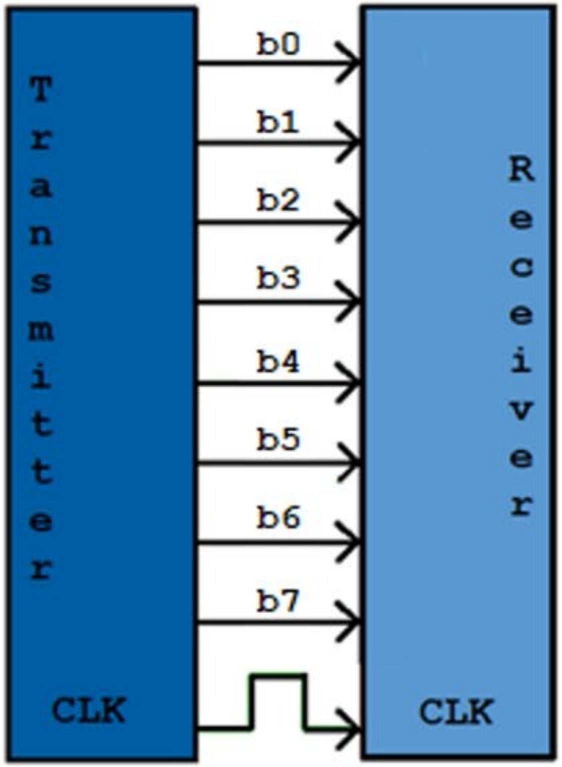




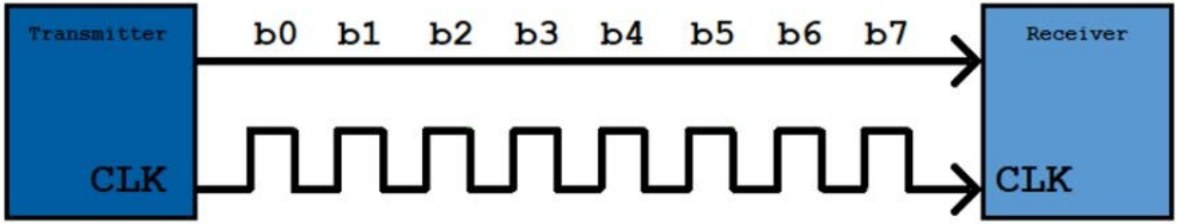
Параллельный порт и FSMC



Последовательный порт

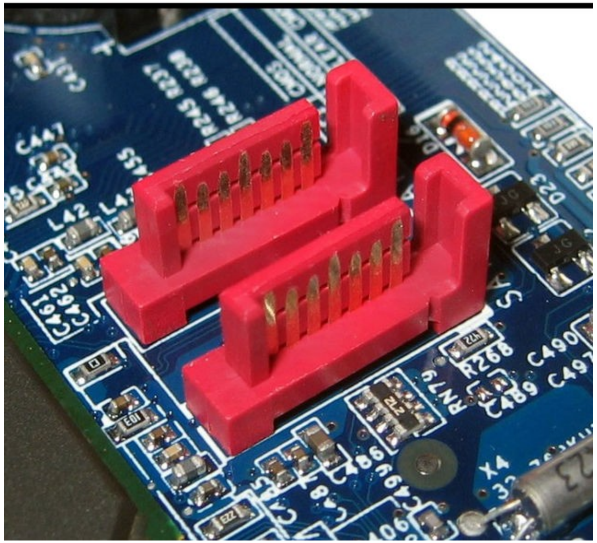


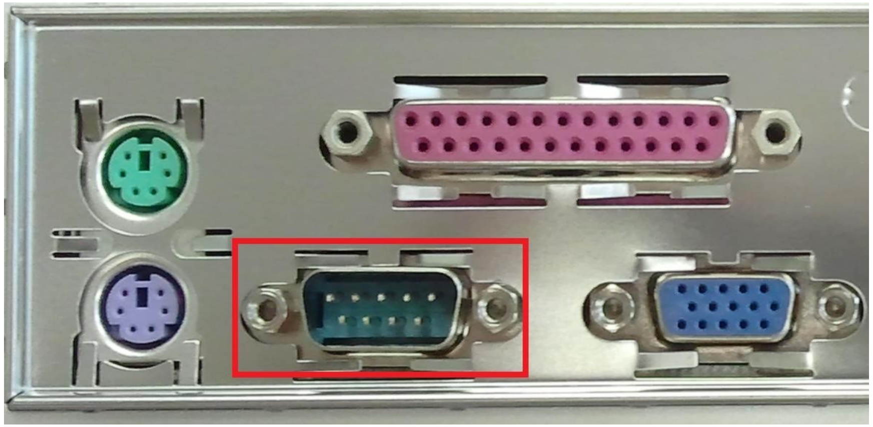
8-bit Parallel Interface



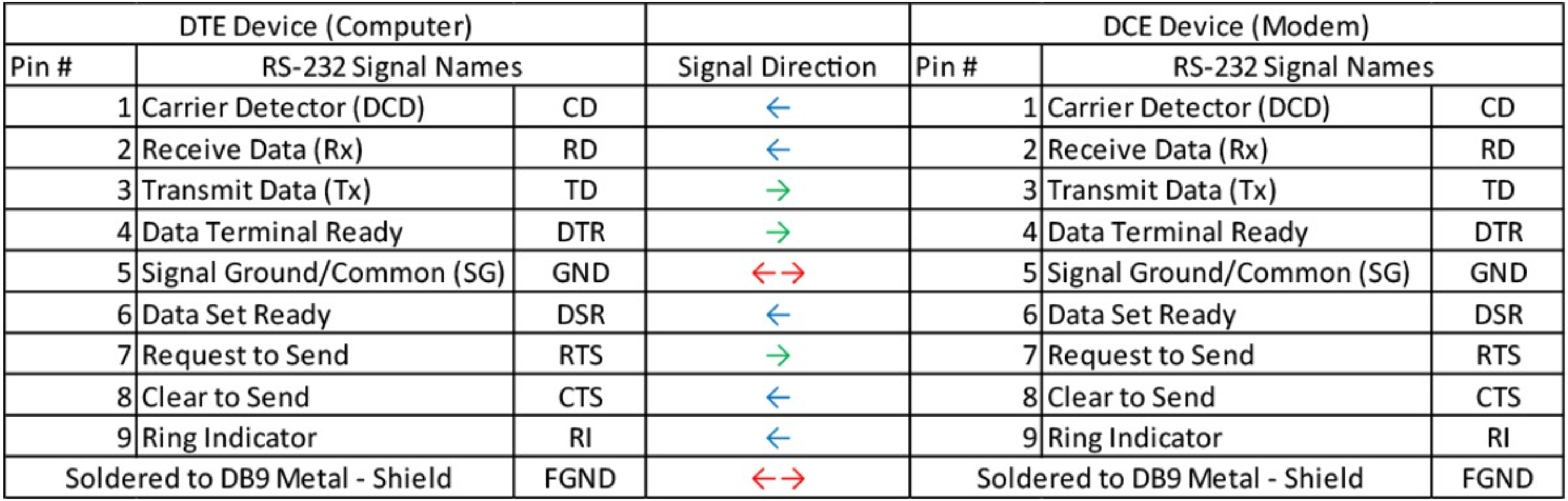
8-bit Serial Interface

Последовательный порт



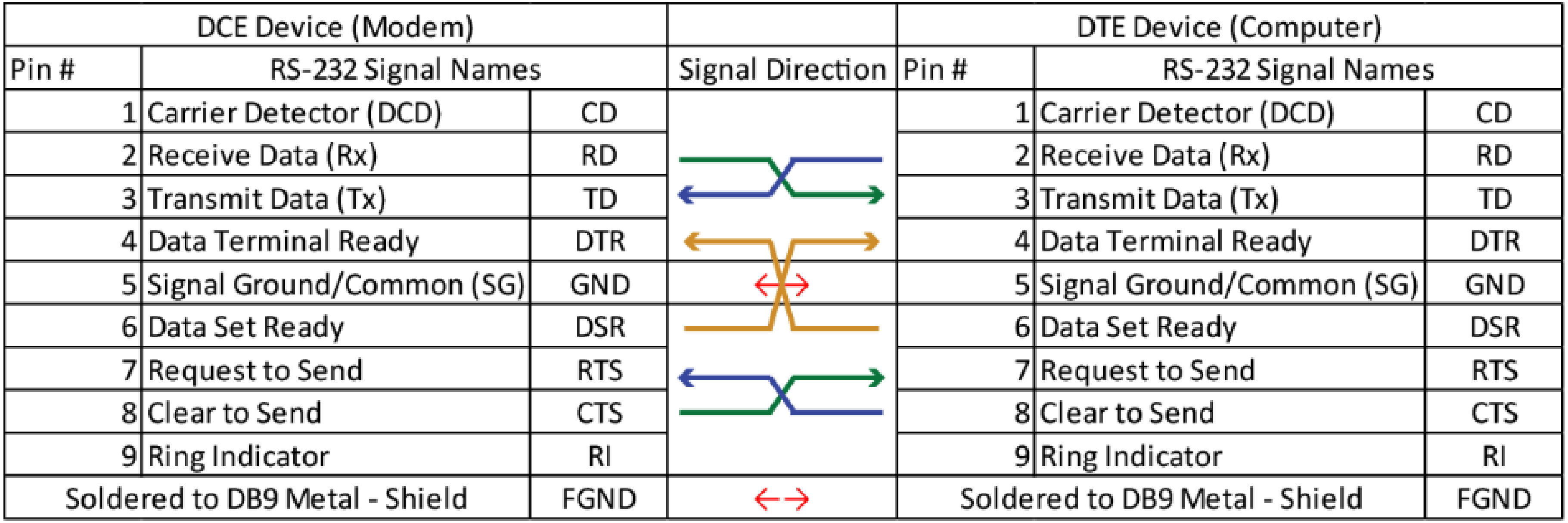


RS232



DTE Device (Computer)				DCE Device (Modem)		
Pin #	RS-232 Signal Names		Signal Direction	Pin #	RS-232 Signal Names	
1	Carrier Detector (DCD)	CD	←	1	Carrier Detector (DCD)	CD
2	Receive Data (Rx)	RD	←	2	Receive Data (Rx)	RD
3	Transmit Data (Tx)	TD	→	3	Transmit Data (Tx)	TD
4	Data Terminal Ready	DTR	→	4	Data Terminal Ready	DTR
5	Signal Ground/Common (SG)	GND	←→	5	Signal Ground/Common (SG)	GND
6	Data Set Ready	DSR	←	6	Data Set Ready	DSR
7	Request to Send	RTS	→	7	Request to Send	RTS
8	Clear to Send	CTS	←	8	Clear to Send	CTS
9	Ring Indicator	RI	←	9	Ring Indicator	RI
Soldered to DB9 Metal - Shield		FGND	←→	Soldered to DB9 Metal - Shield		FGND

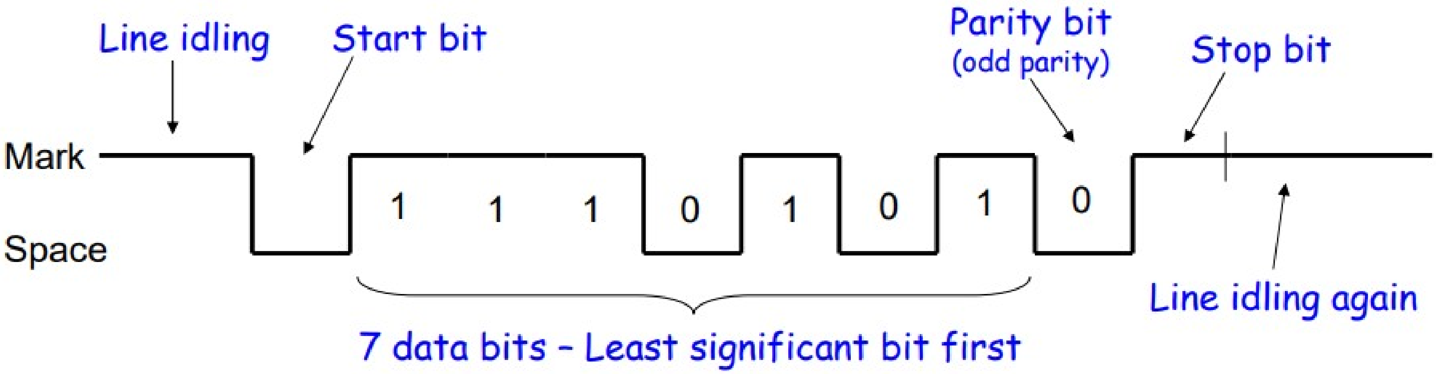
RS232



DCE Device (Modem)				DTE Device (Computer)		
Pin #	RS-232 Signal Names		Signal Direction	Pin #	RS-232 Signal Names	
1	Carrier Detector (DCD)	CD		1	Carrier Detector (DCD)	CD
2	Receive Data (Rx)	RD		2	Receive Data (Rx)	RD
3	Transmit Data (Tx)	TD		3	Transmit Data (Tx)	TD
4	Data Terminal Ready	DTR		4	Data Terminal Ready	DTR
5	Signal Ground/Common (SG)	GND	←→	5	Signal Ground/Common (SG)	GND
6	Data Set Ready	DSR		6	Data Set Ready	DSR
7	Request to Send	RTS		7	Request to Send	RTS
8	Clear to Send	CTS		8	Clear to Send	CTS
9	Ring Indicator	RI		9	Ring Indicator	RI
Soldered to DB9 Metal - Shield		FGND	←→	Soldered to DB9 Metal - Shield		FGND

U(S)ART

- Send the ASCII letter 'W' (1010111)



U(S)ART Pinout

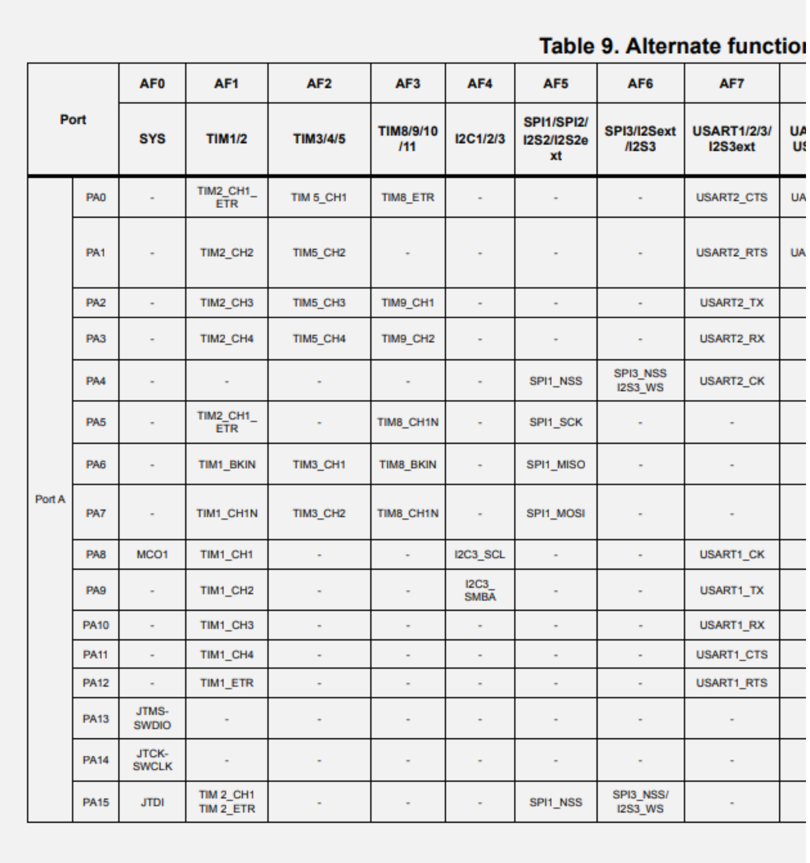
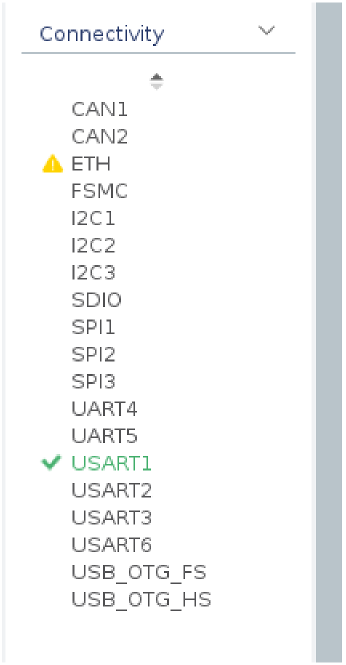


Table 9. Alternate functio

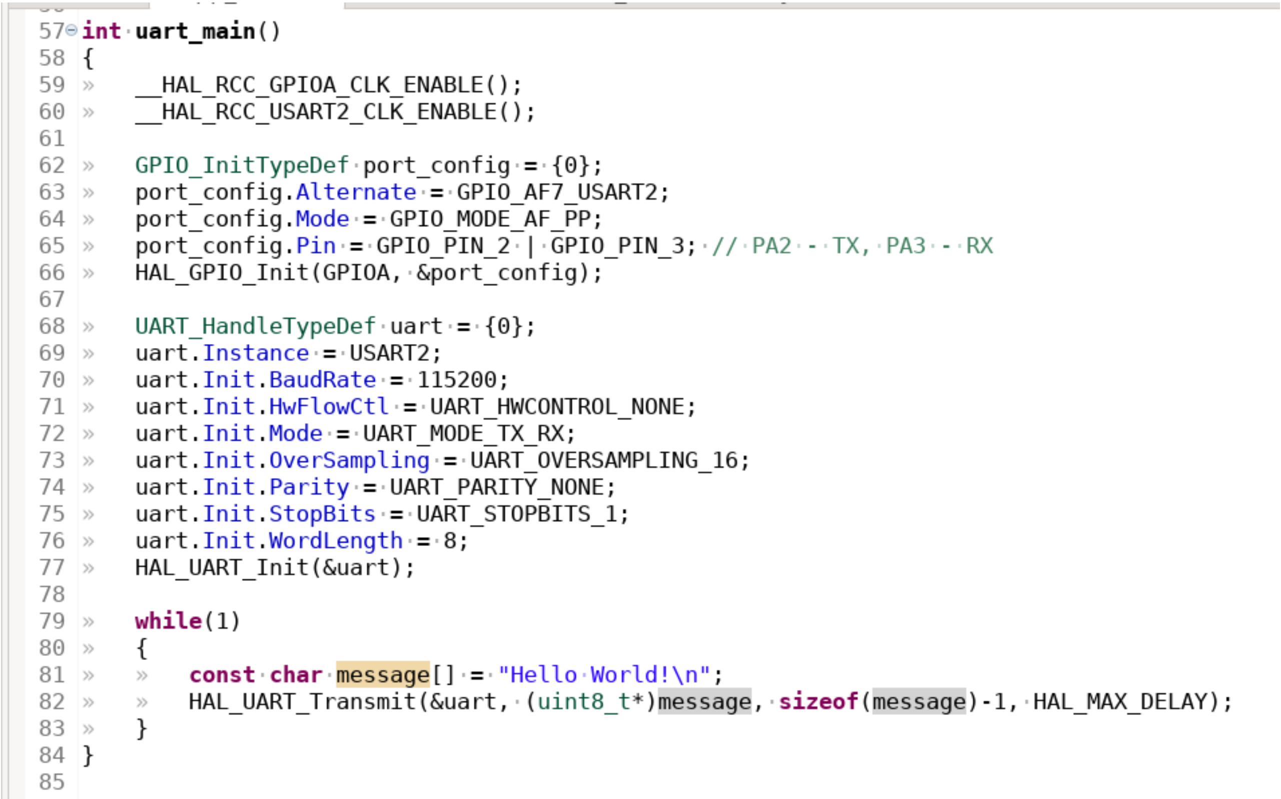
Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	
		SYS	TIM1/2	TIM3/4/5	TIM8/9/10 /11	I2C1/2/3	SPI1/SPI2/ I2S2/I2S2ext	SPI3/I2Sext /I2S3	USART1/2/3/ I2S3ext	UA US
Port A	PA0	-	TIM2_CH1_ ETR	TIM 5_CH1	TIM8_ETR	-	-	-	USART2_CTS	UA
	PA1	-	TIM2_CH2	TIM5_CH2	-	-	-	-	USART2_RTS	UA
	PA2	-	TIM2_CH3	TIM5_CH3	TIM9_CH1	-	-	-	USART2_TX	
	PA3	-	TIM2_CH4	TIM5_CH4	TIM9_CH2	-	-	-	USART2_RX	
	PA4	-	-	-	-	-	SPI1_NSS	SPI3_NSS I2S3_WS	USART2_CK	
	PA5	-	TIM2_CH1_ ETR	-	TIM8_CH1N	-	SPI1_SCK	-	-	
	PA6	-	TIM1_BKIN	TIM3_CH1	TIM8_BKIN	-	SPI1_MISO	-	-	
	PA7	-	TIM1_CH1N	TIM3_CH2	TIM8_CH1N	-	SPI1_MOSI	-	-	
	PA8	MCO1	TIM1_CH1	-	-	I2C3_SCL	-	-	USART1_CK	
	PA9	-	TIM1_CH2	-	-	I2C3_ SMBA	-	-	USART1_TX	
	PA10	-	TIM1_CH3	-	-	-	-	-	USART1_RX	
	PA11	-	TIM1_CH4	-	-	-	-	-	USART1_CTS	
	PA12	-	TIM1_ETR	-	-	-	-	-	USART1_RTS	
	PA13	JTMS- SWDIO	-	-	-	-	-	-	-	
	PA14	JTCK- SWCLK	-	-	-	-	-	-	-	
	PA15	JTDI	TIM 2_CH1 TIM 2_ETR	-	-	-	SPI1_NSS	SPI3_NSS/ I2S3_WS	-	



Connectivity

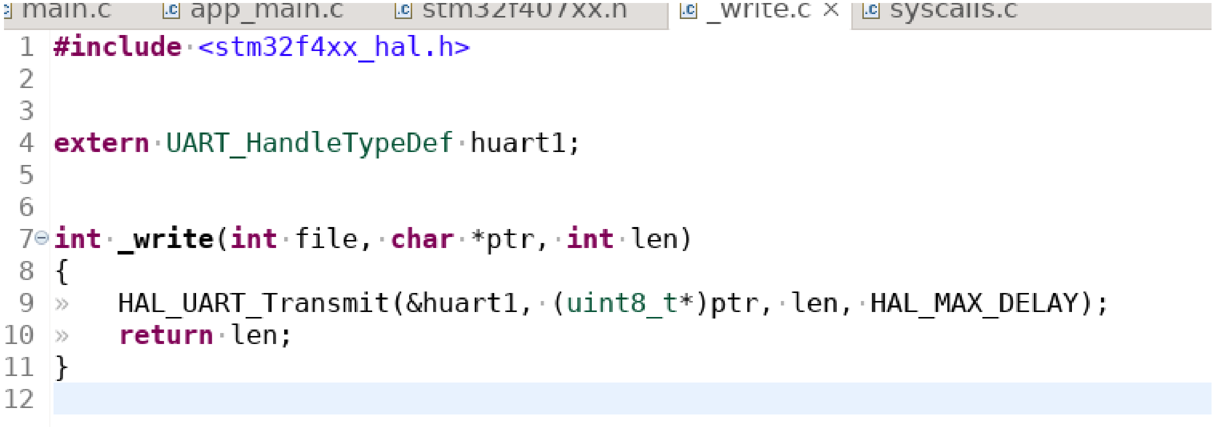
- CAN1
- CAN2
- ⚠ ETH
- FSMC
- I2C1
- I2C2
- I2C3
- SDIO
- SPI1
- SPI2
- SPI3
- UART4
- UART5
- ✓ USART1
- USART2
- USART3
- USART6
- USB_OTG_FS
- USB_OTG_HS

USART Конфигурация



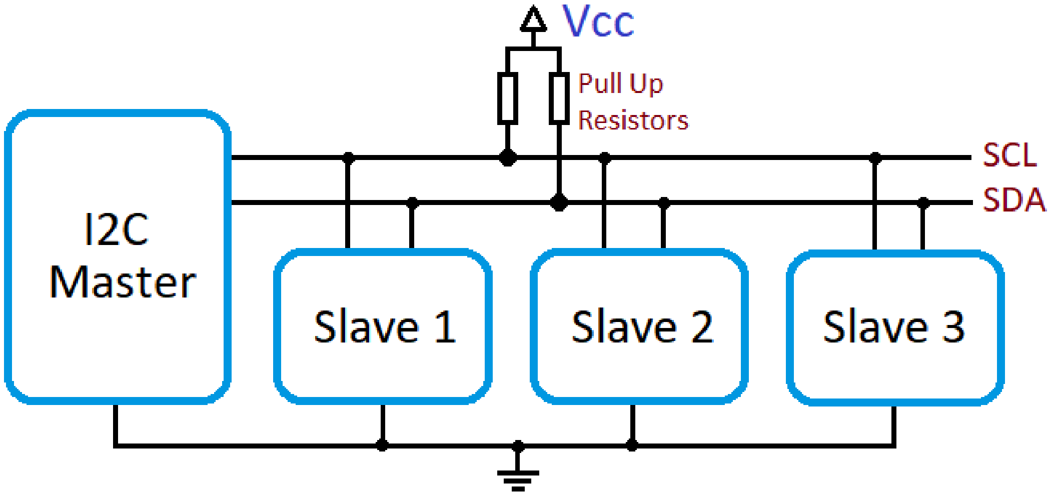
```c
int uart_main()
{
	__HAL_RCC_GPIOA_CLK_ENABLE();
	__HAL_RCC_USART2_CLK_ENABLE();

	GPIO_InitTypeDef port_config = {0};
	port_config.Alternate = GPIO_AF7_USART2;
	port_config.Mode = GPIO_MODE_AF_PP;
	port_config.Pin = GPIO_PIN_2 | GPIO_PIN_3; // PA2 - TX, PA3 - RX
	HAL_GPIO_Init(GPIOA, &port_config);

	UART_HandleTypeDef uart = {0};
	uart.Instance = USART2;
	uart.Init.BaudRate = 115200;
	uart.Init.HwFlowCtl = UART_HWCONTROL_NONE;
	uart.Init.Mode = UART_MODE_TX_RX;
	uart.Init.OverSampling = UART_OVERSAMPLING_16;
	uart.Init.Parity = UART_PARITY_NONE;
	uart.Init.StopBits = UART_STOPBITS_1;
	uart.Init.WordLength = 8;
	HAL_UART_Init(&uart);

	while(1)
	{
		const char message[] = "Hello World!\n";
		HAL_UART_Transmit(&uart, (uint8_t*)message, sizeof(message)-1, HAL_MAX_DELAY);
	}
}
```

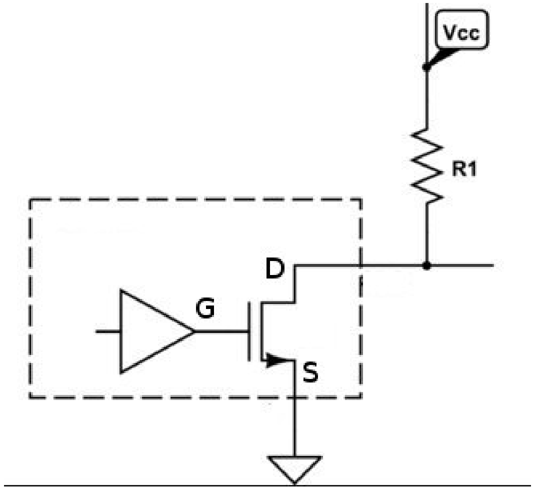
USART printf



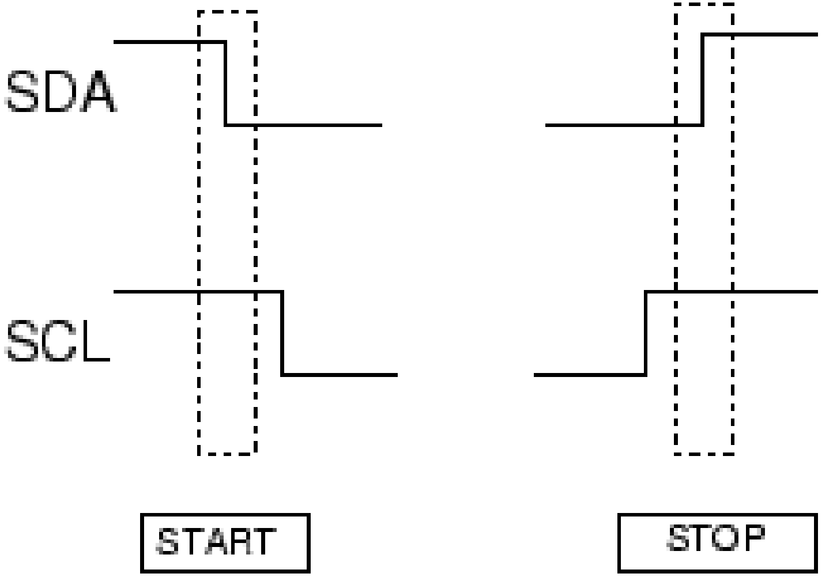
```c
#include <stm32f4xx_hal.h>


extern UART_HandleTypeDef huart1;


int _write(int file, char *ptr, int len)
{
	HAL_UART_Transmit(&huart1, (uint8_t*)ptr, len, HAL_MAX_DELAY);
	return len;
}

```

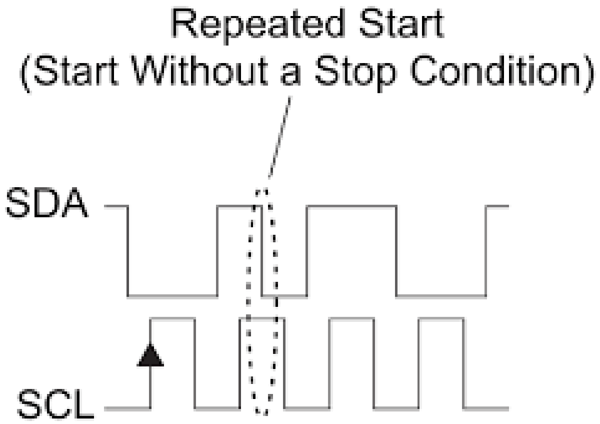
Inter-Integrated Circuit (IIC, I2C)



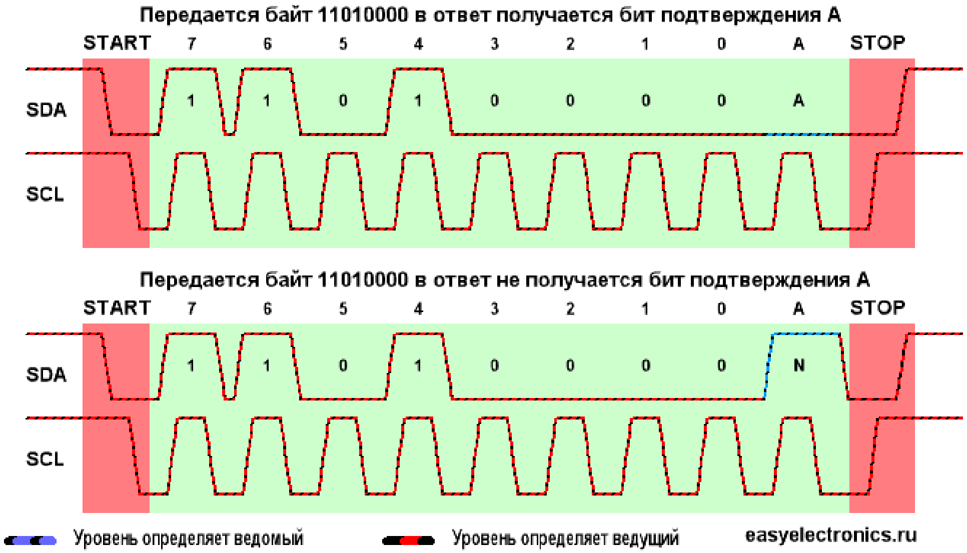


Inter-Integrated Circuit (IIC, I2C)

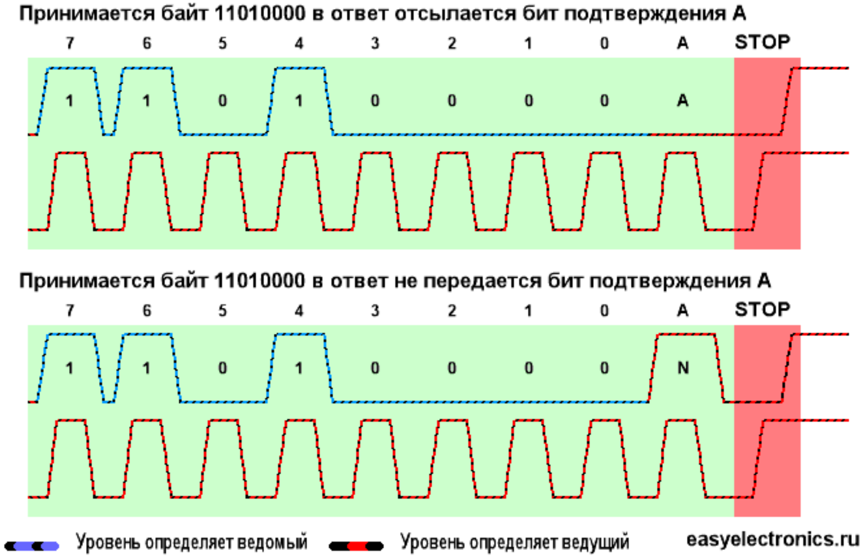




Inter-Integrated Circuit (IIC, I2C)

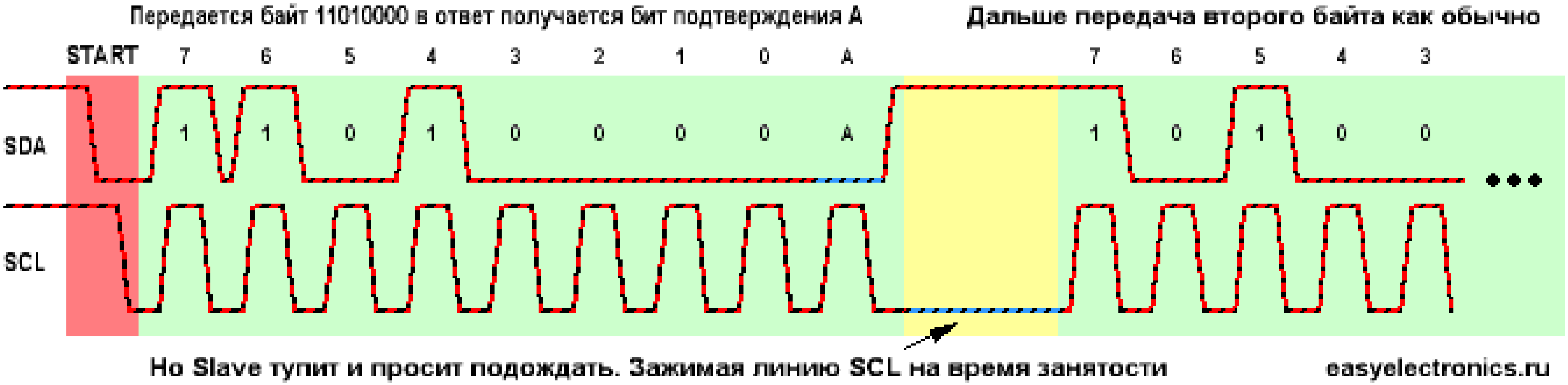

Передается байт 11010000 в ответ получается бит подтверждения А

Передается байт 11010000 в ответ не получается бит подтверждения А

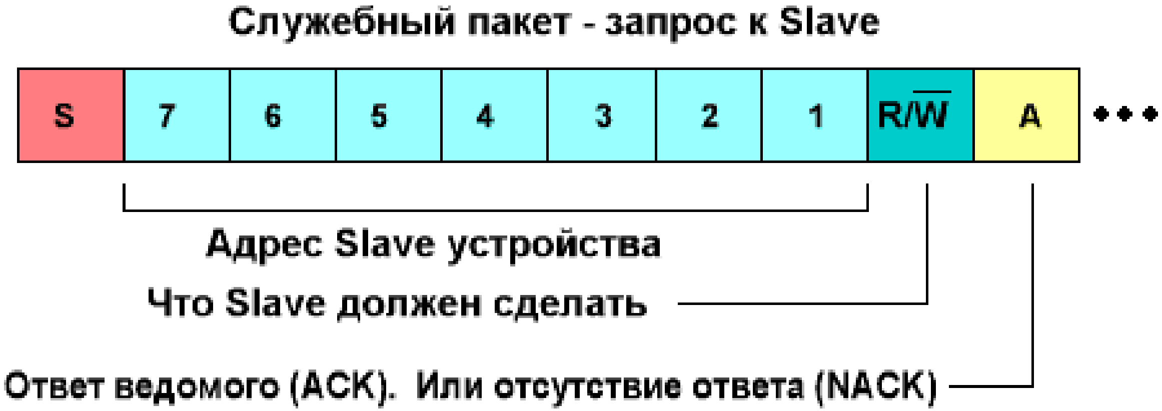

Принимается байт 11010000 в ответ отсылается бит подтверждения А

Принимается байт 11010000 в ответ не передается бит подтверждения А

Inter-Integrated Circuit (IIC, I2C)


Передается байт 11010000 в ответ получается бит подтверждения A
Дальше передача второго байта как обычно
START
7
6
5
4
3
2
1
0
A
7
6
5
4
3
SDA
1
1
0
1
0
0
0
0
A
1
0
1
0
0
SCL
Но Slave тупит и просит подождать. Зажимая линию SCL на время занятости
easyelectronics.ru

Inter-Integrated Circuit (IIC, I2C)



Служебный пакет - запрос к Slave

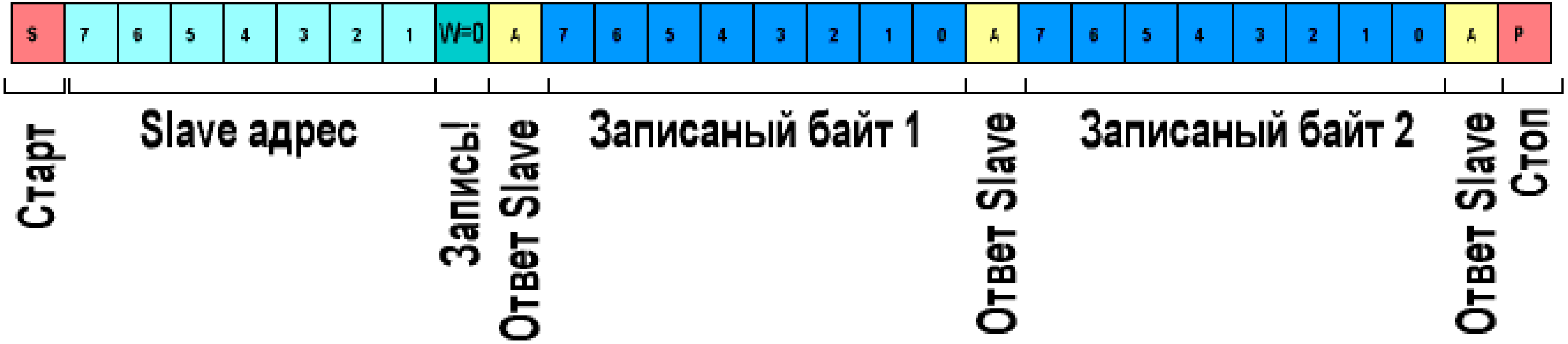
S	7	6	5	4	3	2	1	$R/\overline{W}$	A	• • •

Адрес Slave устройства

Что Slave должен сделать

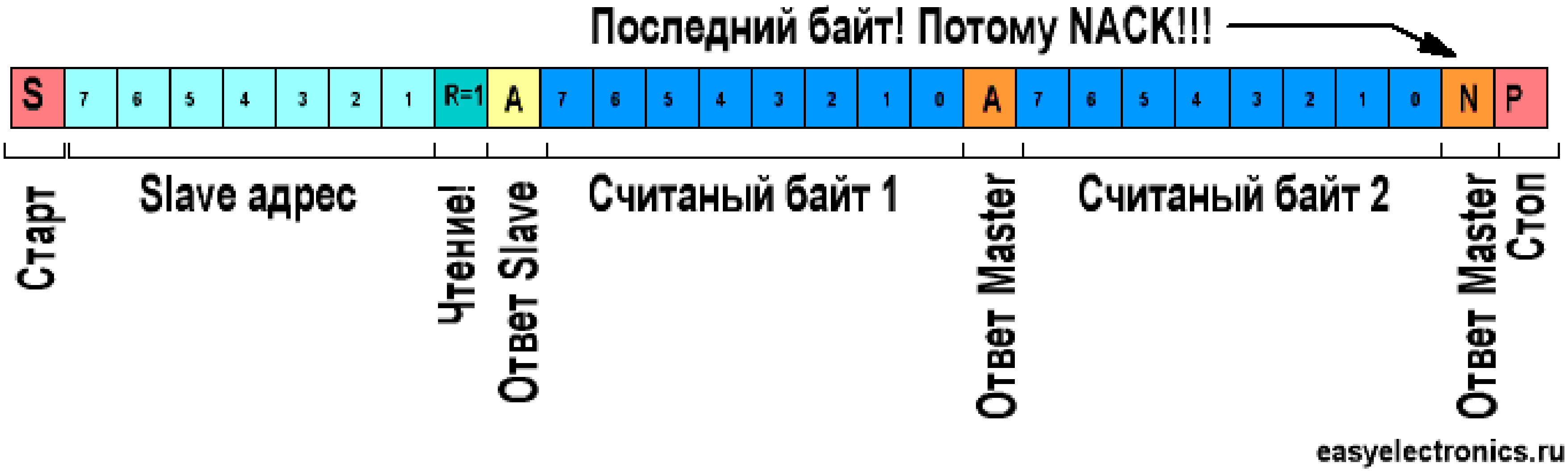
Ответ ведомого (ACK). Или отсутствие ответа (NACK)

Inter-Integrated Circuit (IIC, I2C)



easyelectronics.ru

Inter-Integrated Circuit (IIC, I2C)

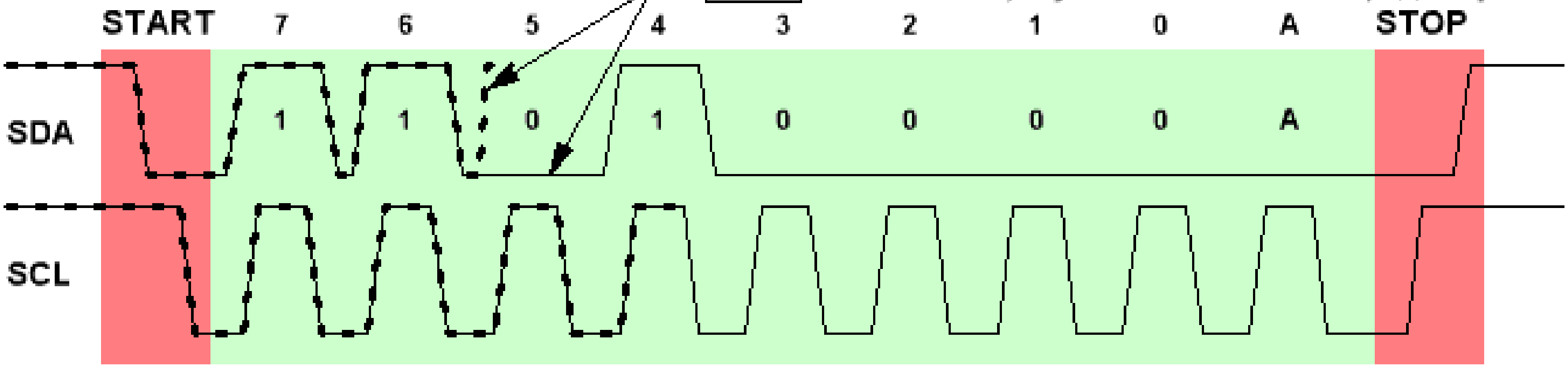


easyelectronics.ru

Арбитраж I2C

- - - - - пытался поставить 1, но обломался, т.к.

—— выставил 0, а у нас монтажное И, где 0 решает!

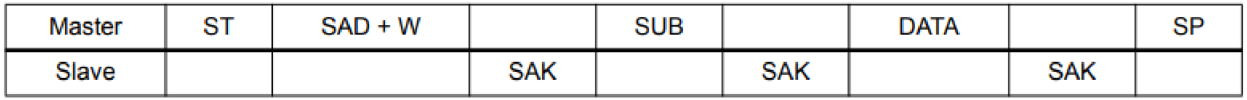


—— Пытается передать 11010000

- - - - Пытается передать 11110000

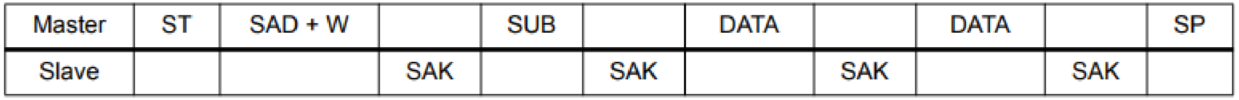
I2C Регистровый доступ

Table 14. Transfer when master is writing one byte to slave



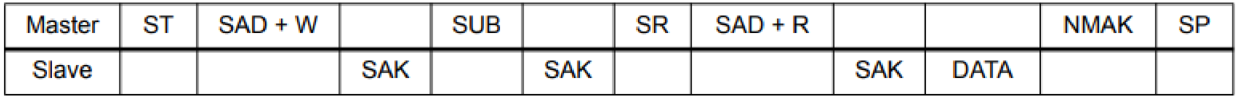
Master	ST	SAD + W		SUB		DATA		SP
Slave			SAK		SAK		SAK	

Table 15. Transfer when master is writing multiple bytes to slave



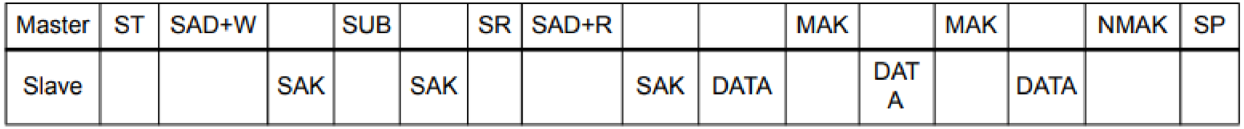
Master	ST	SAD + W		SUB		DATA		DATA		SP
Slave			SAK		SAK		SAK		SAK	

Table 16. Transfer when master is receiving (reading) one byte of data from slave



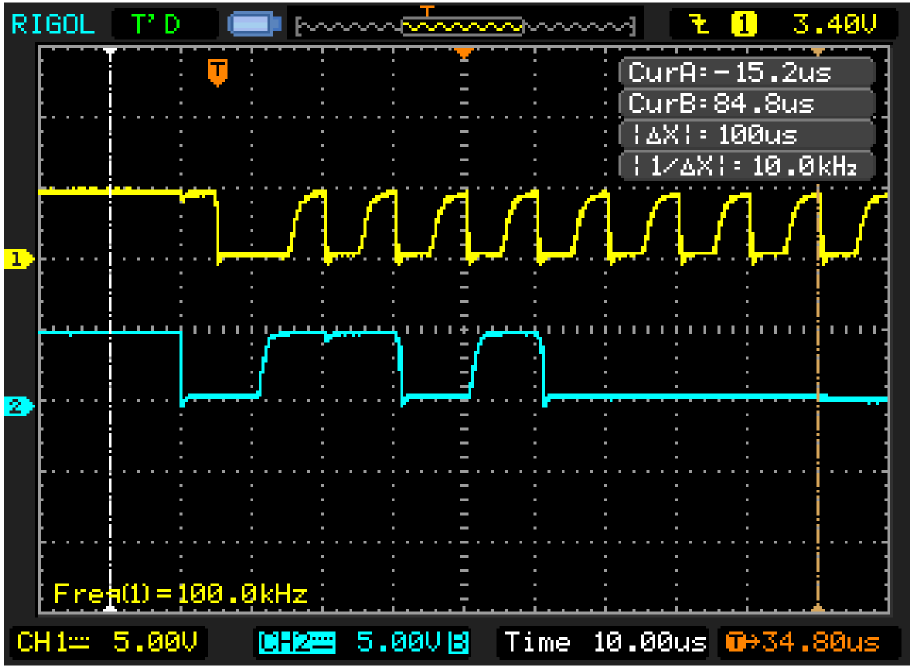
Master	ST	SAD + W		SUB		SR	SAD + R			NMAK	SP
Slave			SAK		SAK			SAK	DATA		

Table 17. Transfer when master is receiving (reading) multiple bytes of data from slave



Master	ST	SAD+W		SUB		SR	SAD+R			MAK		MAK		NMAK	SP
Slave			SAK		SAK			SAK	DATA		DATA		DATA		

Скорость I2C



RIGOL T'D 3.40V

CurA:-15.2us

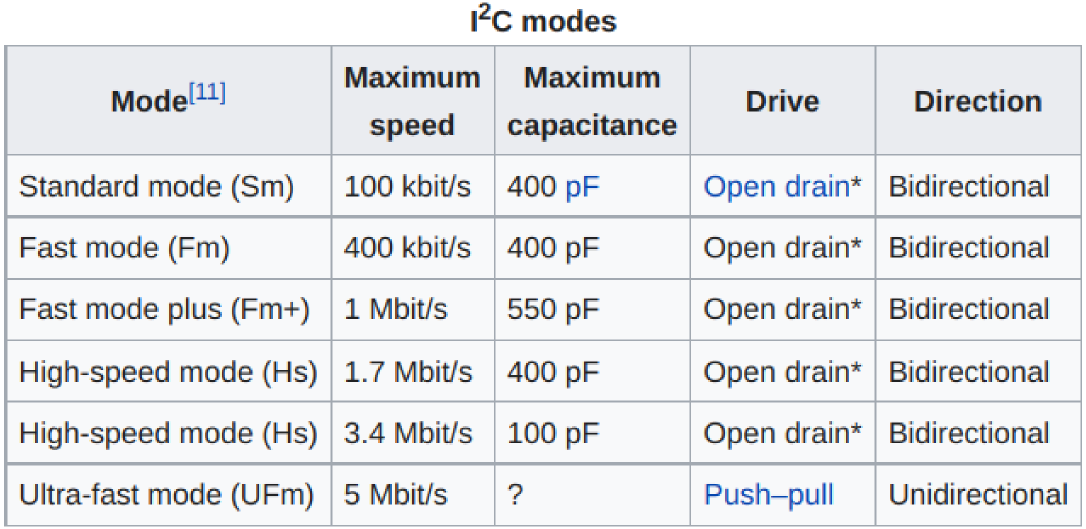
CurB:84.8us

|ΔX|:100us

|1/ΔX|:10.0kHz

Freq(1)=100.0kHz

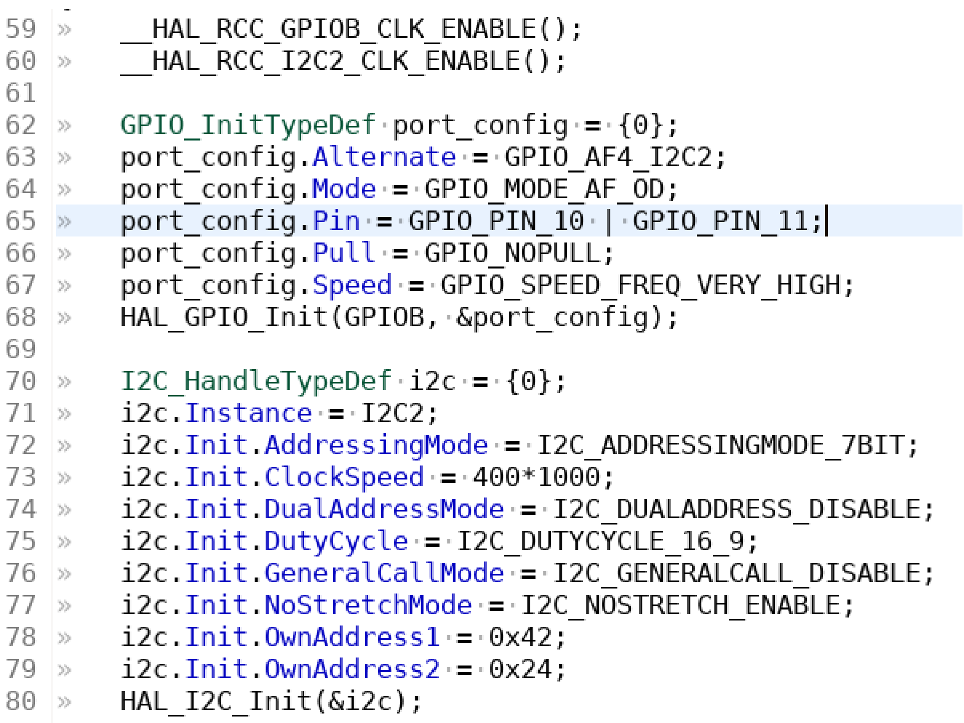
CH1 5.00V CH2 5.00V Time 10.00us T→34.80us



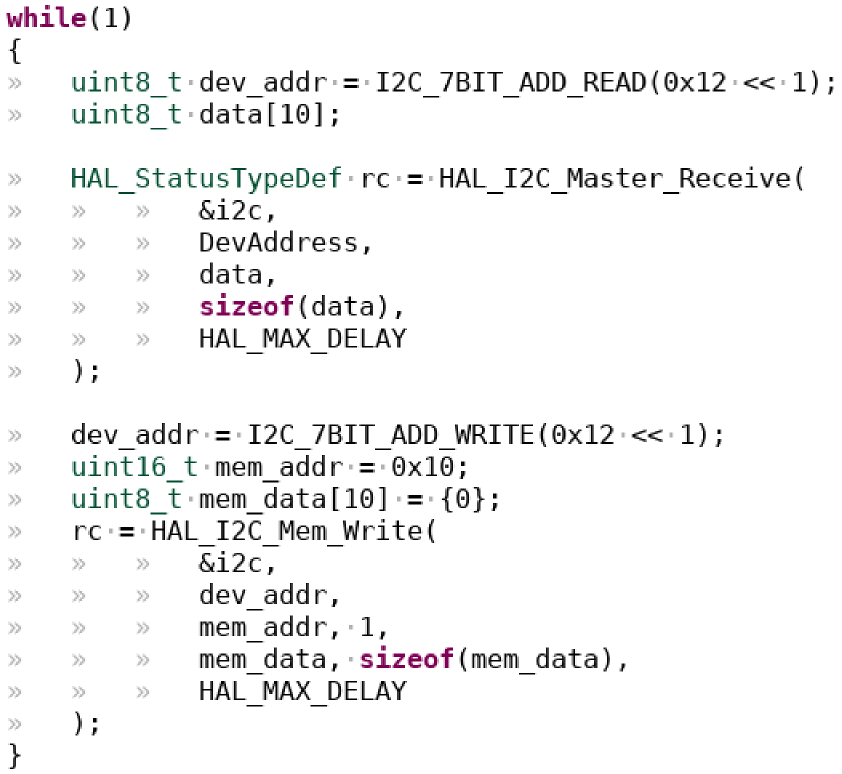
I²C modes

Mode[11]	Maximum speed	Maximum capacitance	Drive	Direction
Standard mode (Sm)	100 kbit/s	400 pF	Open drain*	Bidirectional
Fast mode (Fm)	400 kbit/s	400 pF	Open drain*	Bidirectional
Fast mode plus (Fm+)	1 Mbit/s	550 pF	Open drain*	Bidirectional
High-speed mode (Hs)	1.7 Mbit/s	400 pF	Open drain*	Bidirectional
High-speed mode (Hs)	3.4 Mbit/s	100 pF	Open drain*	Bidirectional
Ultra-fast mode (UFm)	5 Mbit/s	?	Push–pull	Unidirectional

Настройка I2C для STM32 HAL

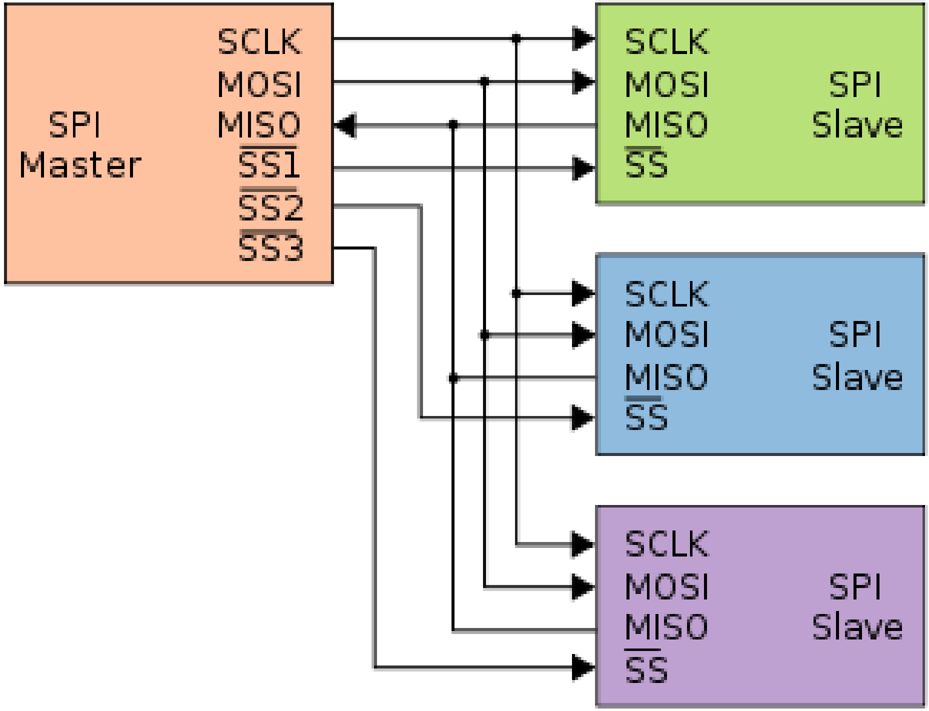


```c
    __HAL_RCC_GPIOB_CLK_ENABLE();
    __HAL_RCC_I2C2_CLK_ENABLE();

    GPIO_InitTypeDef port_config = {0};
    port_config.Alternate = GPIO_AF4_I2C2;
    port_config.Mode = GPIO_MODE_AF_OD;
    port_config.Pin = GPIO_PIN_10 | GPIO_PIN_11;
    port_config.Pull = GPIO_NOPULL;
    port_config.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    HAL_GPIO_Init(GPIOB, &port_config);

    I2C_HandleTypeDef i2c = {0};
    i2c.Instance = I2C2;
    i2c.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
    i2c.Init.ClockSpeed = 400*1000;
    i2c.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
    i2c.Init.DutyCycle = I2C_DUTYCYCLE_16_9;
    i2c.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
    i2c.Init.NoStretchMode = I2C_NOSTRETCH_ENABLE;
    i2c.Init.OwnAddress1 = 0x42;
    i2c.Init.OwnAddress2 = 0x24;
    HAL_I2C_Init(&i2c);
```

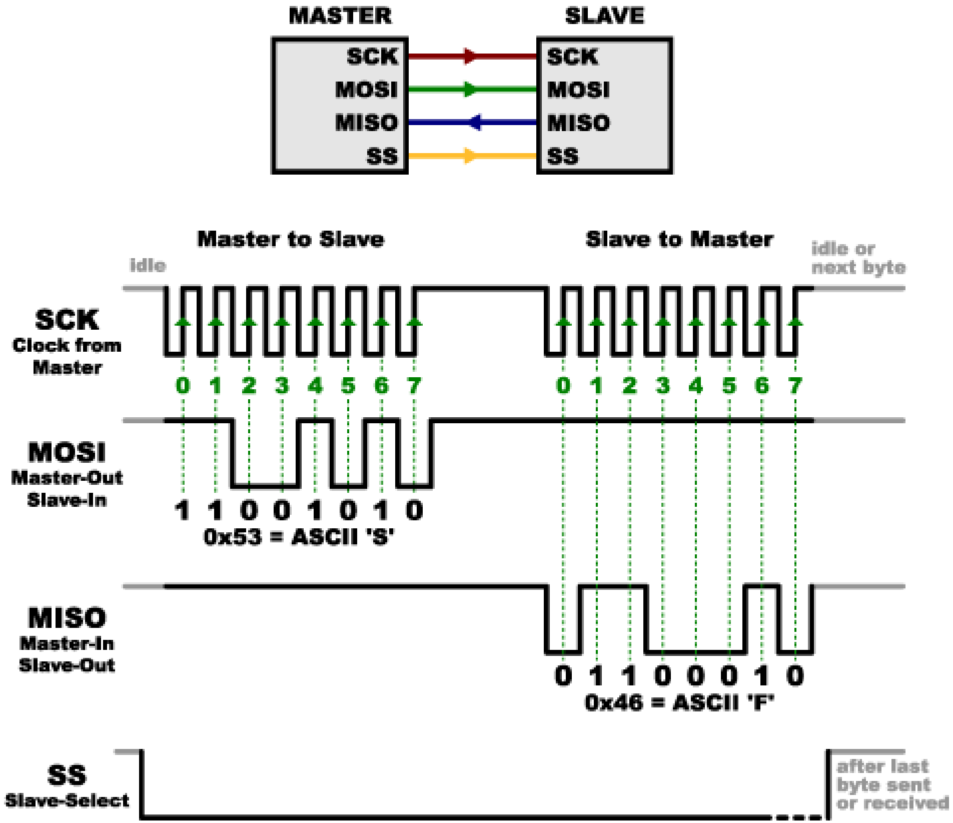


```c
while(1)
{
    uint8_t dev_addr = I2C_7BIT_ADD_READ(0x12 << 1);
    uint8_t data[10];

    HAL_StatusTypeDef rc = HAL_I2C_Master_Receive(
                &i2c,
                DevAddress,
                data,
                sizeof(data),
                HAL_MAX_DELAY
    );

    dev_addr = I2C_7BIT_ADD_WRITE(0x12 << 1);
    uint16_t mem_addr = 0x10;
    uint8_t mem_data[10] = {0};
    rc = HAL_I2C_Mem_Write(
                &i2c,
                dev_addr,
                mem_addr, 1,
                mem_data, sizeof(mem_data),
                HAL_MAX_DELAY
    );
}
```

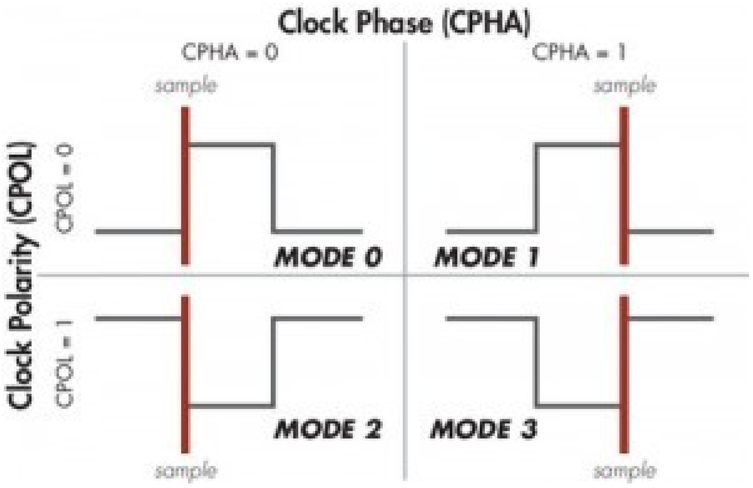
Serial Peripheral Interface

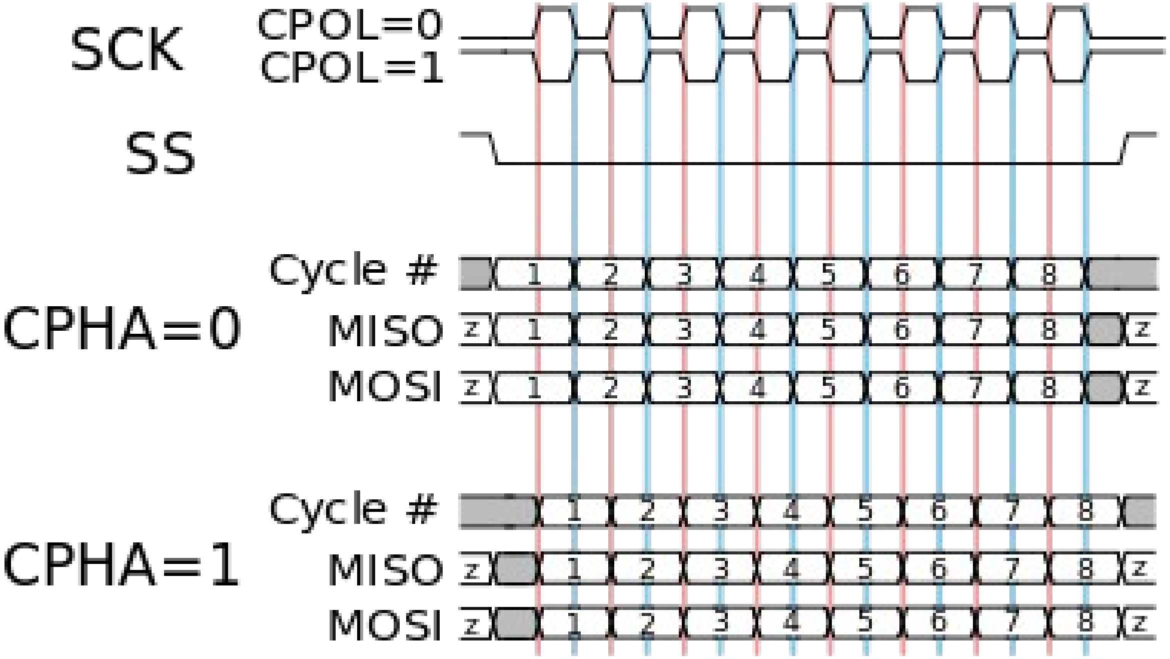


Serial Peripheral Interface



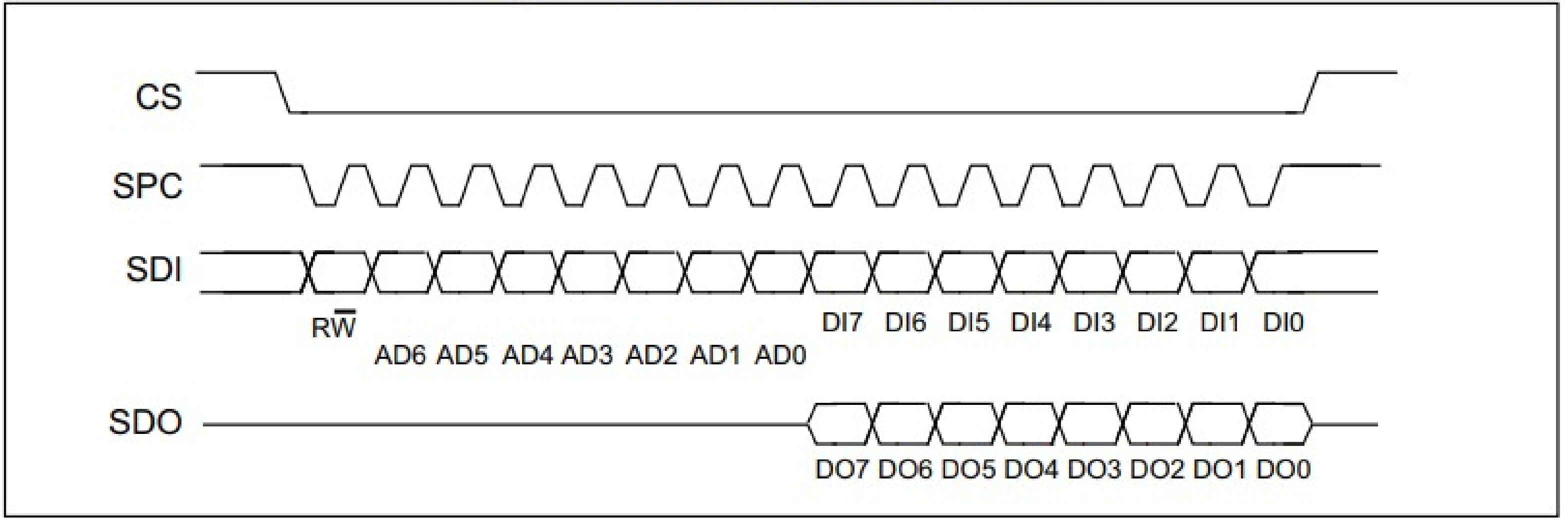
Serial Peripheral Interface





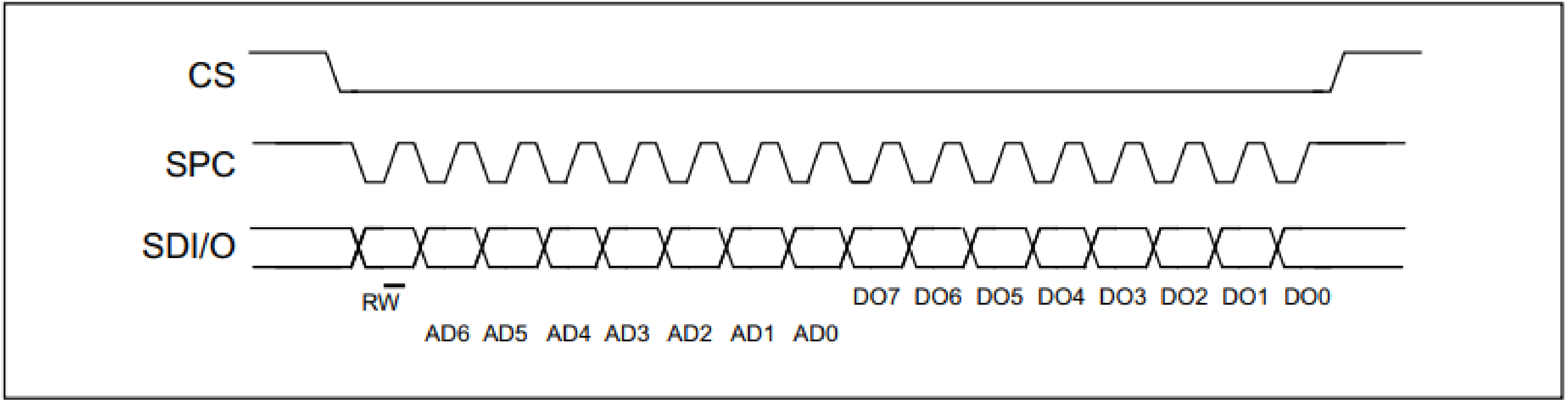
Serial Peripheral Interface

Figure 9. Read and write protocol

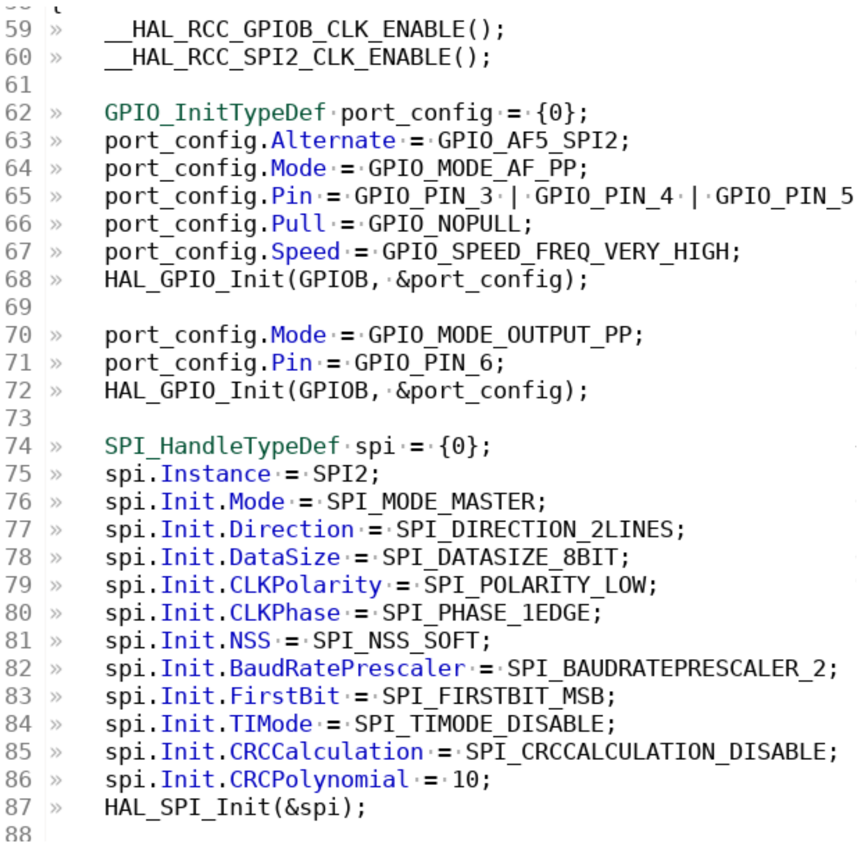

CS
SPC
SDI
RW̄
AD6 AD5 AD4 AD3 AD2 AD1 AD0
DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0
SDO
DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0

Serial Peripheral Interface

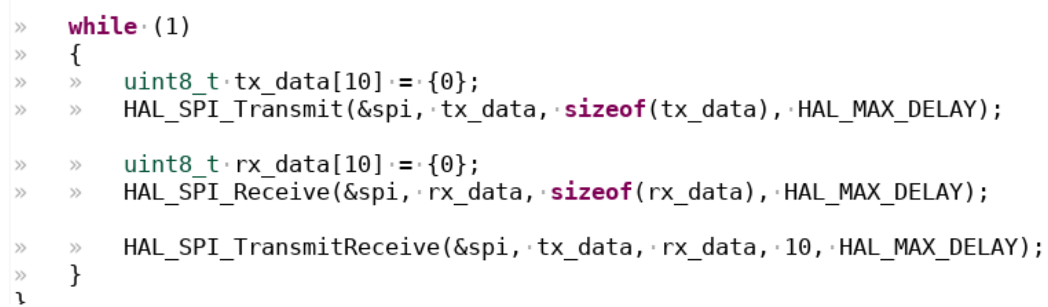
Figure 14. SPI read protocol in 3-wire mode



Serial Peripheral Interface

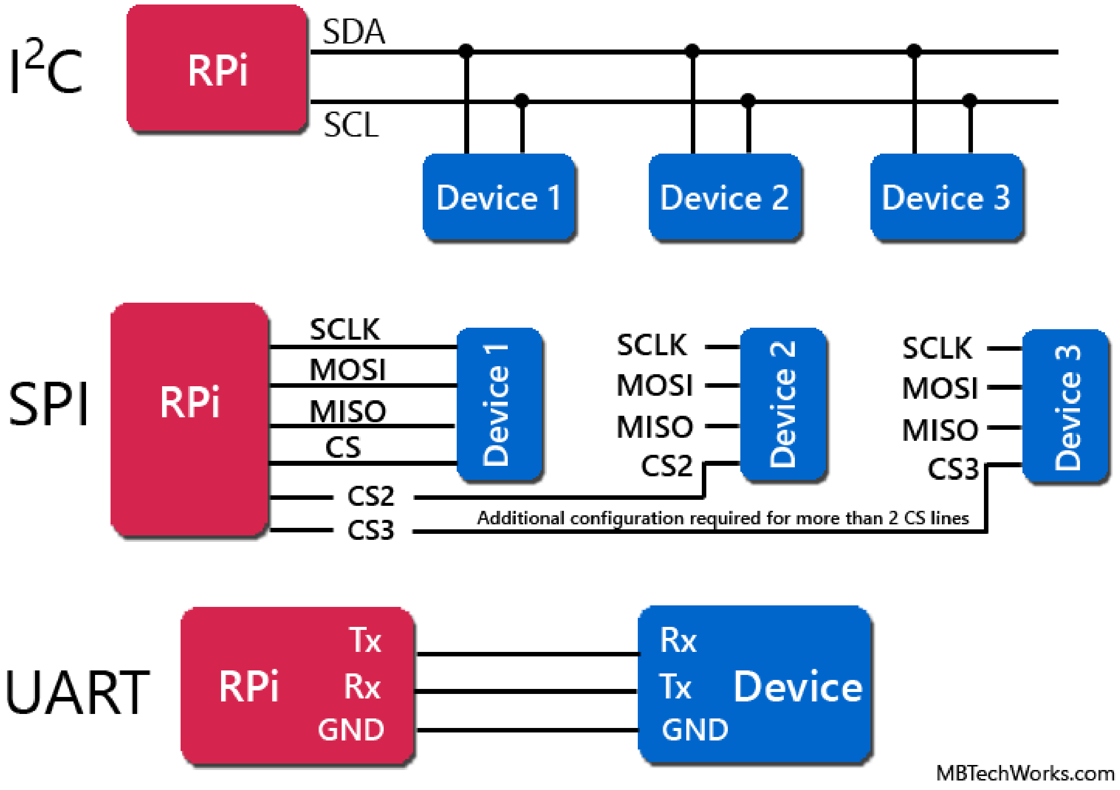


```c
{
    __HAL_RCC_GPIOB_CLK_ENABLE();
    __HAL_RCC_SPI2_CLK_ENABLE();

    GPIO_InitTypeDef port_config = {0};
    port_config.Alternate = GPIO_AF5_SPI2;
    port_config.Mode = GPIO_MODE_AF_PP;
    port_config.Pin = GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5;
    port_config.Pull = GPIO_NOPULL;
    port_config.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    HAL_GPIO_Init(GPIOB, &port_config);

    port_config.Mode = GPIO_MODE_OUTPUT_PP;
    port_config.Pin = GPIO_PIN_6;
    HAL_GPIO_Init(GPIOB, &port_config);

    SPI_HandleTypeDef spi = {0};
    spi.Instance = SPI2;
    spi.Init.Mode = SPI_MODE_MASTER;
    spi.Init.Direction = SPI_DIRECTION_2LINES;
    spi.Init.DataSize = SPI_DATASIZE_8BIT;
    spi.Init.CLKPolarity = SPI_POLARITY_LOW;
    spi.Init.CLKPhase = SPI_PHASE_1EDGE;
    spi.Init.NSS = SPI_NSS_SOFT;
    spi.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
    spi.Init.FirstBit = SPI_FIRSTBIT_MSB;
    spi.Init.TIMode = SPI_TIMODE_DISABLE;
    spi.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
    spi.Init.CRCPolynomial = 10;
    HAL_SPI_Init(&spi);

```



```c
    while (1)
    {
        uint8_t tx_data[10] = {0};
        HAL_SPI_Transmit(&spi, tx_data, sizeof(tx_data), HAL_MAX_DELAY);

        uint8_t rx_data[10] = {0};
        HAL_SPI_Receive(&spi, rx_data, sizeof(rx_data), HAL_MAX_DELAY);

        HAL_SPI_TransmitReceive(&spi, tx_data, rx_data, 10, HAL_MAX_DELAY);
    }
}
```

Все вместе



I²C
RPi
SDA
SCL
Device 1
Device 2
Device 3
SPI
RPi
SCLK
MOSI
MISO
CS
CS2
CS3
Device 1
SCLK
MOSI
MISO
CS2
Device 2
SCLK
MOSI
MISO
CS3
Device 3
Additional configuration required for more than 2 CS lines
UART
RPi
Tx
Rx
GND
Rx
Tx
GND
Device
MBTechWorks.com