Курс по STM32

Лекция #8:

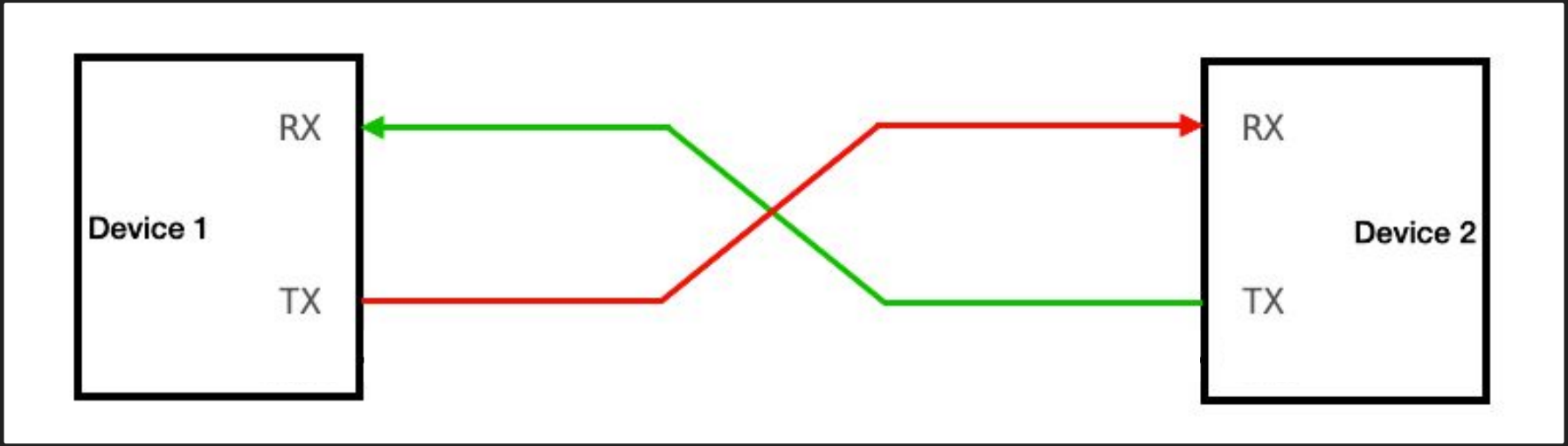
- Простейший протокол передачи данных: UART.
- Контроллер UART в STM32F051.
- Выдача ДЗ №5: `04_uart`.

01.04.2023 / 03.04.2023

Устройство протокола UART

(Universal Asynchronous Receiver-Transmitter)

Наивная передача данных



Наивная попытка передачи данных через GPIO с одного МК на другой:

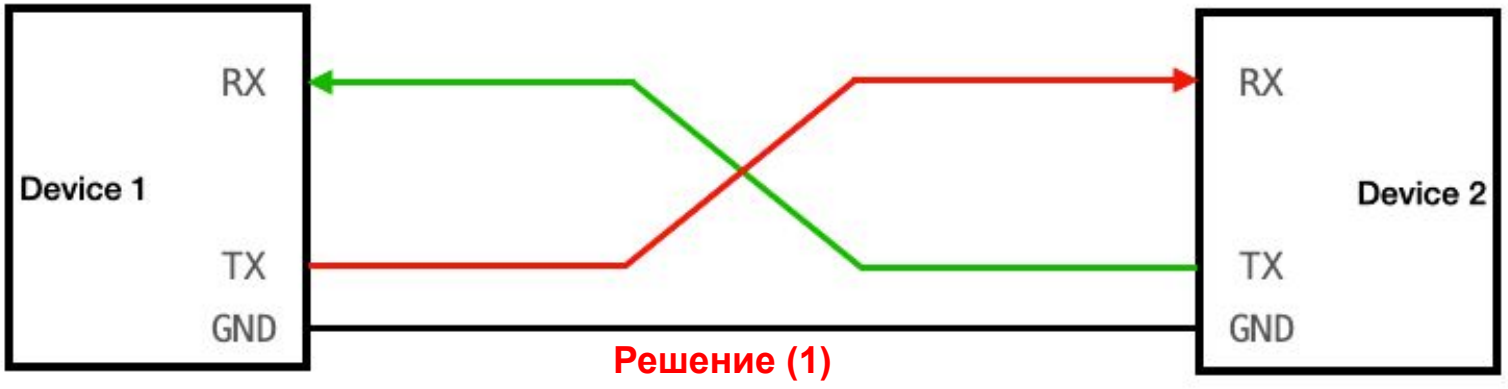
1) Есть ли электротехнические проблемы?
2) Что мешает передавать поток бит с заданной частотой?

Проблемы наивной передачи данных

Проблемы наивного подхода:

1) Несовпадение уровней “0” и “1”.
2) Различение состояний “передача идёт / не идёт”.
3) Проблемы рассинхронизации:
   3.1) Разброс параметров осцилляторов при производстве;
   3.2) Зависимости параметров осцилляторов от условий эксплуатации;
4) Переполнение буферов приёмника.
5) Наличие ошибок в среде передачи.

UART: решение проблем наивной реализации



Решение (1)

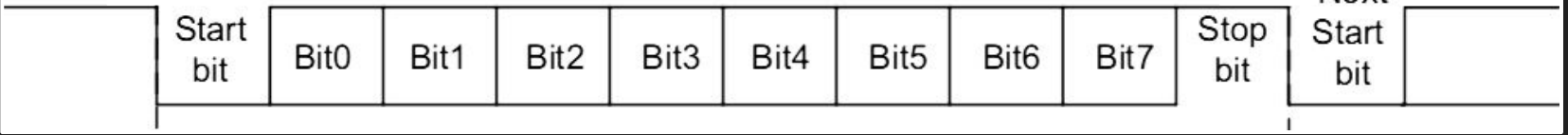
8-bit word length (M = 00), 1 Stop bit

Решение (2)

Data frame

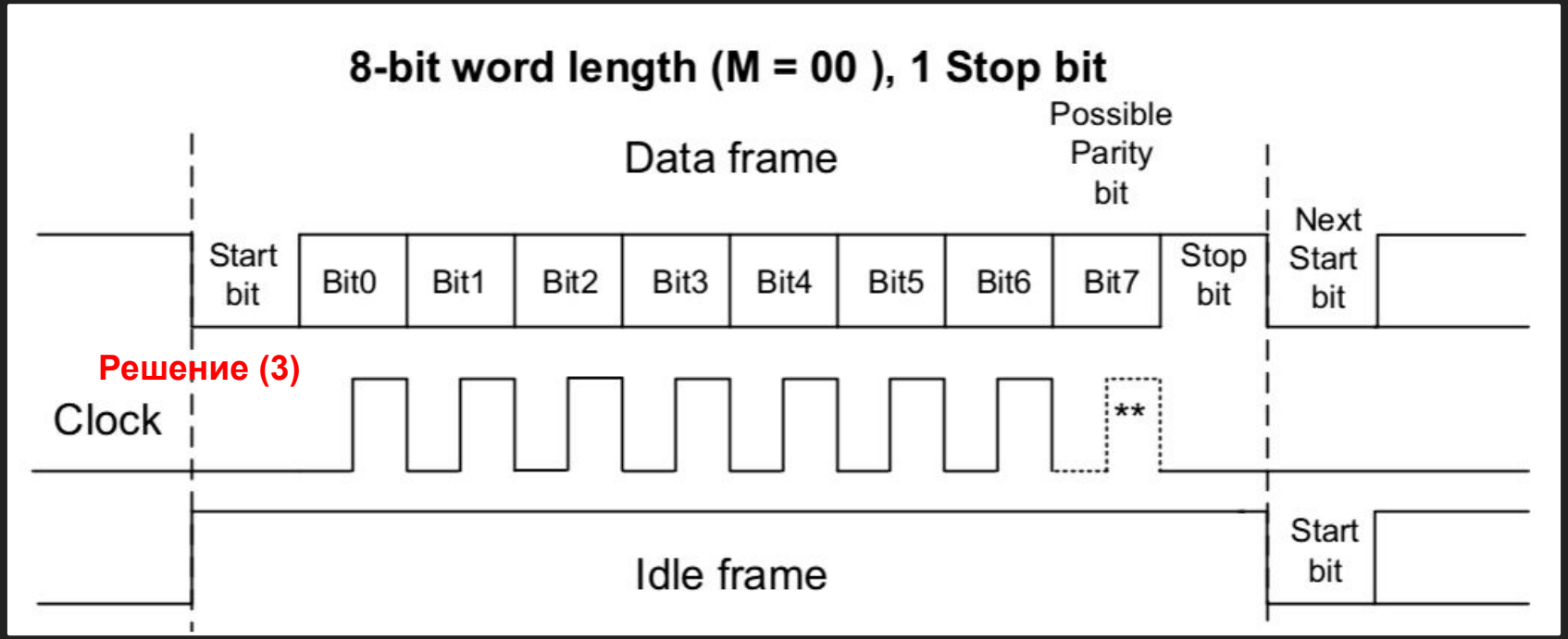
Possible Parity bit

Решение (5)



USART: решение проблем наивной реализации

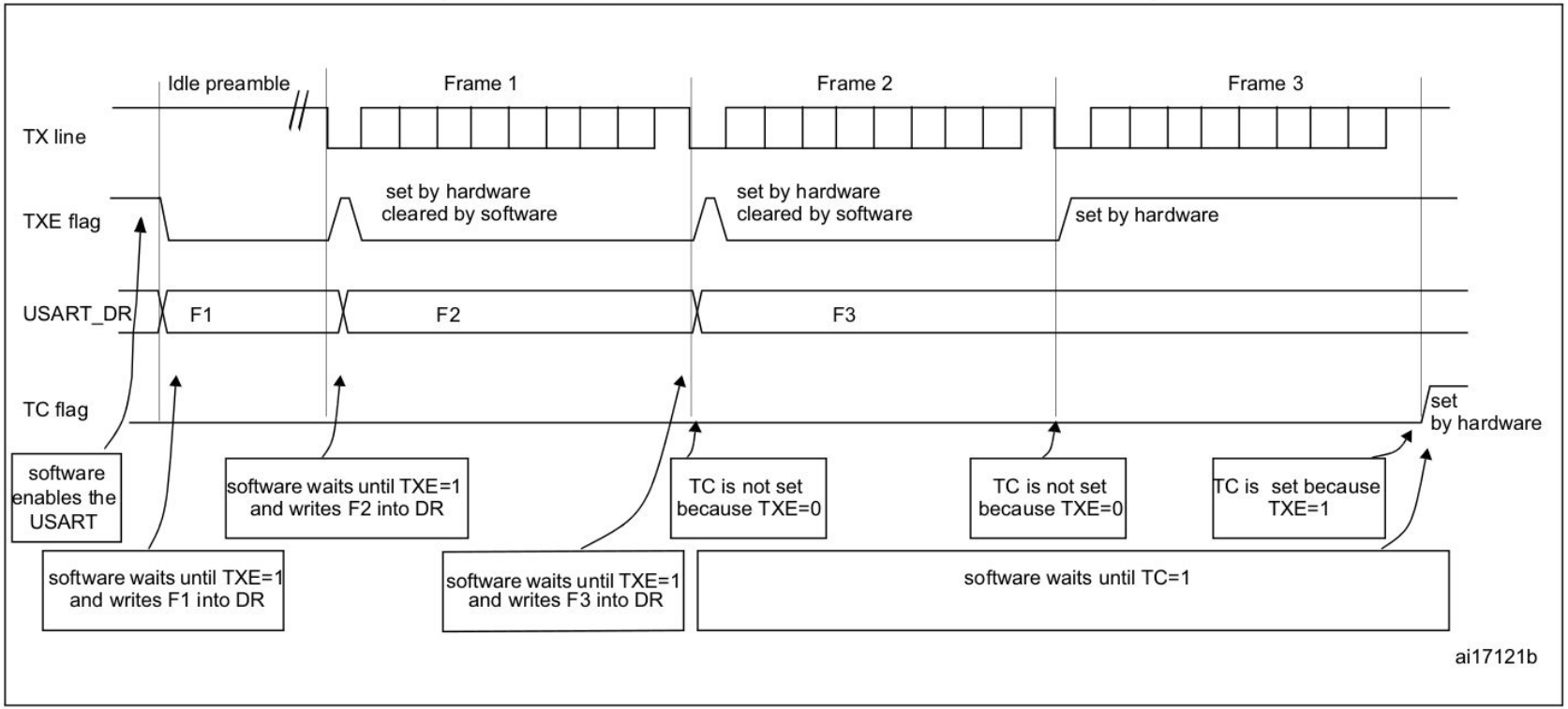
Протокол USART (universal synchronous/asynchronous receiver/transmitter):



Контроллер UART в STM32F051

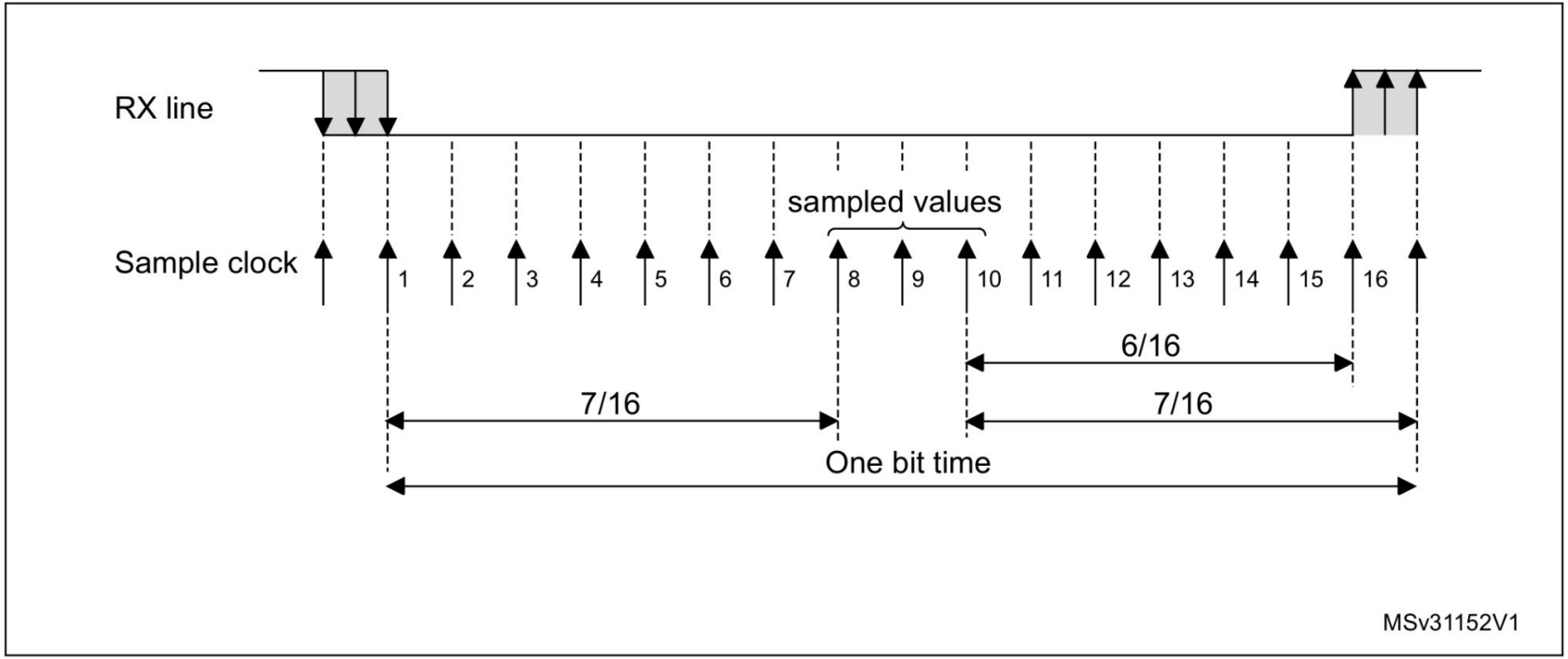
Контроллер UART: отправка символов

Figure 248. TC/TXE behavior when transmitting



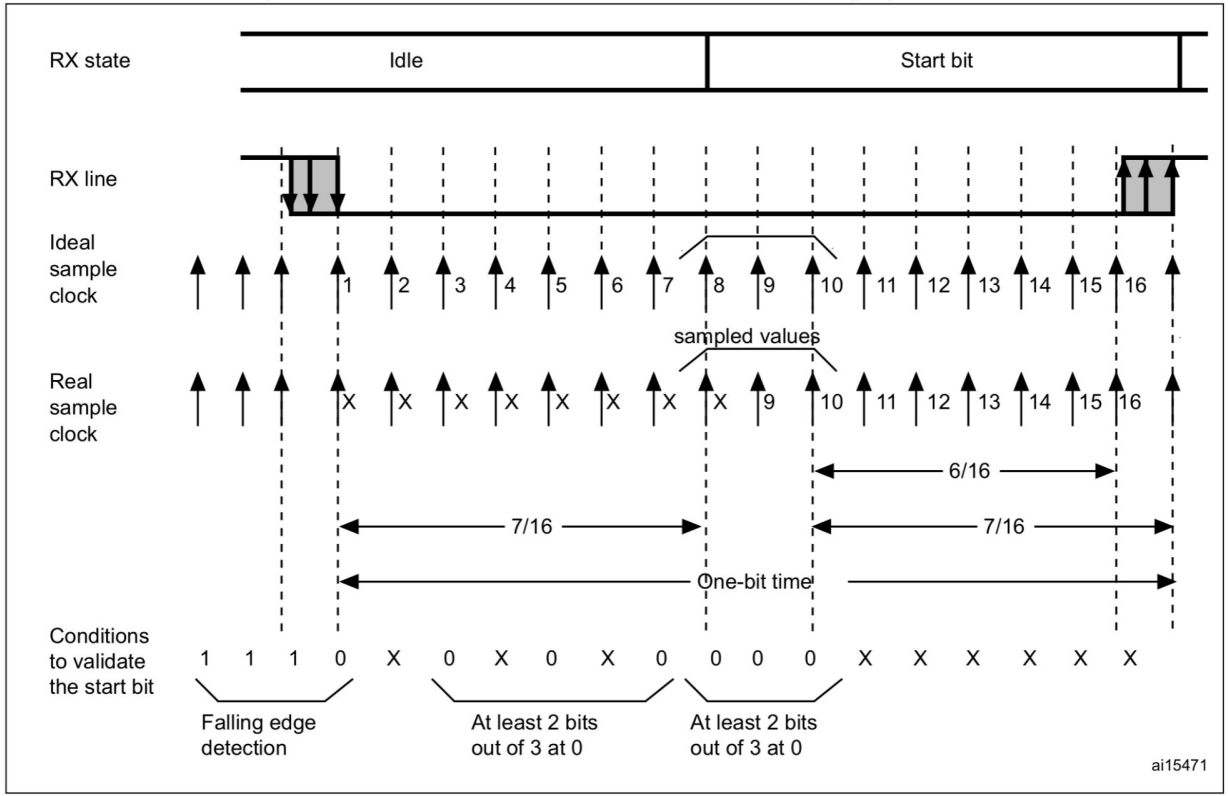
Контроллер UART: приём символов данных

Figure 250. Data sampling when oversampling by 16



Контроллер UART: приём start bit-a

Figure 249. Start bit detection when oversampling by 16 or 8



Контроллер UART: частота символов

Table 104. Error calculation for programmed baud rates at f_{CK} = 48 MHz in both cases of oversampling by 16 or by 8(1)

Baud rate		Oversampling by 16 (OVER8 = 0)			Oversampling by 8 (OVER8 = 1)		
S.No	Desired	Actual	BRR	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	BRR	% Error
2	2.4 KBps	2.4 KBps	0x4E20	0	2.4 KBps	0x9C40	0
3	9.6 KBps	9.6 KBps	0x1388	0	9.6 KBps	0x2710	0
4	19.2 KBps	19.2 KBps	0x9C4	0	19.2 KBps	0x1384	0
5	38.4 KBps	38.4 KBps	0x4E2	0	38.4 KBps	0x9C2	0
6	57.6 KBps	57.62 KBps	0x341	0.03	57.59 KBps	0x681	0.02
7	115.2 KBps	115.11 KBps	0x1A1	0.08	115.25 KBps	0x340	0.04
8	230.4 KBps	230.76KBps	0xD0	0.16	230.21 KBps	0x1A0	0.08
9	460.8 KBps	461.54KBps	0x68	0.16	461.54KBps	0xD0	0.16
10	921.6KBps	923.07KBps	0x34	0.16	923.07KBps	0x64	0.16

Контроллер UART: прерывания

Table 109. USART interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit data register empty	TXE	TXEIE
CTS interrupt	CTSIF	CTSIE
Transmission Complete	TC	TCIE
Receive data register not empty (data ready to be read)	RXNE	RXNEIE
Overrun error detected	ORE	
Idle line detected	IDLE	IDLEIE
Parity error	PE	PEIE
LIN break	LBDF	LBDIE
Noise Flag, Overrun error and Framing Error in multibuffer communication.	NF or ORE or FE	EIE
Character match	CMF	CMIE
Receiver timeout	RTOF	RTOIE
End of Block	EOBF	EOBIE
Wakeup from Stop mode	WUF(1)	WUFIE

Выдача ДЗ №5: 04_uart

UART в STM32F051: пример кода

Настройка UART (labs/04_uart):

```
// (1) Configure USART1 clocking:
*REG_RCC_APB2ENR |= (1U << 14U);
*REG_RCC_CFGR3   |= 0b00U;

// (2) Set USART1 parameters:
uint32_t reg_usart_cr1 = 0U;
uint32_t reg_usart_cr2 = 0U;

reg_usart_cr1 |= 0x00000000U;  // Data length: 8 bits
reg_usart_cr1 |=  (0U << 15U); // Use oversampling by 16
reg_usart_cr1 &= ~(1U << 10U); // Parity control: disabled
reg_usart_cr1 |=  (1U <<  3U); // Transmitter: enabled

reg_usart_cr2 |= (0U << 19U);    // Endianness: LSB first
reg_usart_cr2 |= (0b10U << 12U); // Stop bits: 2

*USART1_CR1 = reg_usart_cr1;
*USART1_CR2 = reg_usart_cr2;
```

```
// (3) Configure USART baud rate:
uint32_t usartdiv = (frequency + baudrate/2)/baudrate;

*USART1_BRR = usartdiv;

// (4) Enable UART:
*USART1_CR1 |= (1U << 0U);

// (5) Wait for TX to enable:
while ((*USART1_ISR & (1U << 21U)) == 0U);
```

```
void uart_send_byte(char sym)
{
    // Wait for TXE:
    while ((*USART1_ISR & (1U <<  7U)) == 0U);

    // Put data into DR:
    *USART1_TDR = sym;
}
```

UART на прерываниях – в репо Эдгара Казиахмедова (labs/09_usart_terminal).

UART в STM32F051: тонкие моменты

Частоты разных осцилляторов одного номинала – разные!

```
#define UART_BAUDRATE 9600
#define UART_BAUDRATE_FIX -300

int main()
{
    board_clocking_init();

    board_gpio_init();

    uart_init(UART_BAUDRATE + UART_BAUDRATE_FIX, CPU_FREQENCY);

    print_string("Hello, world!\r");
```

UART в STM32F051: тонкие моменты

Таблица допустимых отклонений отношения частот TX и RX осцилляторов:

Table 106. Tolerance of the USART receiver when BRR [3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Для нашего случая: 300 / 9600 = 3.1%

Утилита для связи по UART: minicom

```
~/path/to/stm32f051_rewind/labs/04_uart>
❯ make uart // Связь с МК по USB-UART-преобразователю
sudo cp minirc.stm32f051 /etc/minicom/
sudo minicom -D /dev/ttyUSB0 stm32f051

Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Dec 23 2019, 02:06:26.
Port /dev/ttyUSB0, 17:21:39

Press CTRL-A Z for help on special keys

Hello, world! // Данные, переданные с МК по UART!
```

Требования к ДЗ №05: 04_uart

```
[1] Отрефакторить код 04_uart и разметить все регистры:
    [-] Регистры используются только по их именам.
    [-] Используются биты регистров только по их именам.
[2] Реализовать “UartGPT”:
    [-] Сценарий работы:
        -  По UART-у через minicom отправляется строка “X”.
        -  Устройство принимает её и отвечает "no you X".
    [-] Пример работы:
        > make me a sandwich
        > no you make me a sandwich
        > go reboot yourself
        > no you go reboot yourself
[3] Подзадание со звездой -- появится на сл.лекции!
```

Спасибо за внимание!

Красивые иконки взяты с сайта handdrawngoods.com