UART (*Universal Asynchronous Receiver-Transmitter,* Универсальный асинхронный приёмопередатчик) — узел вычислительных устройств, предназначенный для организации связи с другими цифровыми устройствами.

Для работы с UART необходимо при создании проекта, во вкладке Repository-Peripherals выбрать следующие компоненты:

* M4 CMSIS Core
* CMSIS BOOT
* RCC. Отвечает за тактирование
* GPIO
* USART

Для начала рассмотрим подробнее отправление данных.

Инициализация структуры, содержащая настройки порта:

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE); //

GPIO\_InitTypeDef PORT\_SETUP; *//структура настройки ножки*

PORT\_SETUP.GPIO\_Mode = GPIO\_Mode\_AF; *//альтернативная функция ножки*

PORT\_SETUP.GPIO\_OType = GPIO\_OType\_PP; *//ножка пуш-пул*

PORT\_SETUP.GPIO\_Pin = GPIO\_Pin\_2; *//2 ножка*

PORT\_SETUP.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOA, &PORT\_SETUP);

GPIO\_PinAFConfig(GPIOA,GPIO\_PinSource2,GPIO\_AF\_USART2);

Инициализация структуры, содержащая настройки UART.

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART2, ENABLE);

USART\_InitTypeDef uart\_setup;

uart\_setup.USART\_BaudRate = 9600; *//скорость 9600*

uart\_setup.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None; *//нет контроля потока*

uart\_setup.USART\_Mode = USART\_Mode\_Tx; *//только передатчик*

uart\_setup.USART\_Parity = USART\_Parity\_No; *//без контроля четности*

uart\_setup.USART\_StopBits = USART\_StopBits\_1; *//1 стоп бит*

uart\_setup.USART\_WordLength = USART\_WordLength\_8b; *//8 бит данных*

USART\_Init(USART2, &uart\_setup); *//запись структуры*

USART\_Cmd(USART2, ENABLE); *//включить юарт2*

Виды режимов (.USART\_Mode):

USART\_Mode\_Rx - приём

USART\_Mode\_Tx - передача

USART\_Mode\_Rx|USART\_Mode\_Tx – приём\передача

Инициализация окончена.

Для отправки данных можно составить следующую функцию:

int putcharx(uint8\_t ch)

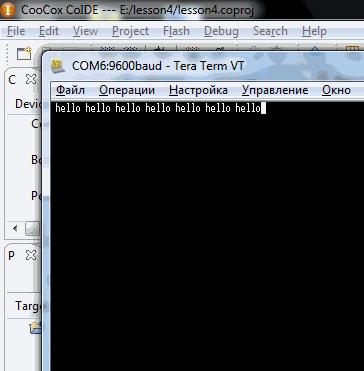
{

while (USART\_GetFlagStatus(USART2, USART\_FLAG\_TXE) == RESET);

USART\_SendData(USART2, (uint8\_t)ch);

return ch;

}

Пример работы с ней:

putcharx('h');

putcharx('e');

putcharx('l');

putcharx('l');

putcharx('o');

Можно составить следующую. Её можно использовать также для отправки целой строки данных.

//Функция отправляет байт в UART

void send\_to\_uart(uint8\_t data)

{

 while(!(USART2->SR & USART\_SR\_TC));

 USART2->DR=data;

}

//Функция отправляет строку в UART, по сути пересылая по байту в send\_to\_uart

void send\_str(char \* string)

{

 uint8\_t i=0;

 while(string[i])

{

  send\_to\_uart(string[i]);

  i++;

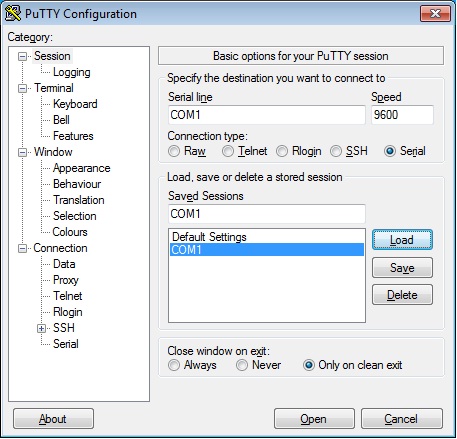
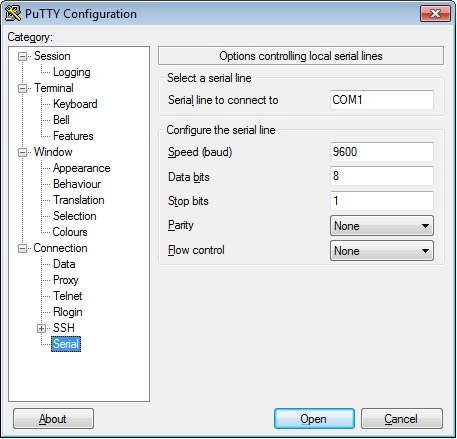
 }

}

Для приёма данных на компьютер, можно использовать программу PuTTY.

Для этого настраиваем программу следующим образом:

Во вкладке Session выбираем Serial.  
В строке Serial line пишем номер порта, к которому вы подключили плату.  
В строке Speed пишем это скорость работы. (нужно, чтобы совпадало с .USART\_BaudRate при инициализации)  
Переходим во вкладку Serial

Data bits ставим 8.  
Stop bits – 1.  
Parity и Flow control надо выставить None.  
Нажимаем Open и получаем консоль.

Теперь поговорим о приёме данных с некоторого устройства на микроконтроллер.

Можно настроить микроконтроллер так, чтобы при приёме данных происходило прерывание.

Для этого нужно объявить переменную, в которую будут заноситься новые данные:

uint8\_t receivedData[16];

*// Счетчик принятых байт*

uint8\_t receivedDataCounter = 0;

Инициализация для приёма данных может выглядеть следующим образом:

*// Тактирование, куда ж без него-то*

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1, ENABLE);

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE);

*// Настраиваем ножки контроллера, тут все понятно*

GPIO\_InitTypeDef gpio;

GPIO\_StructInit(&gpio);

gpio.GPIO\_Mode = GPIO\_Mode\_AF;

gpio.GPIO\_Pin = GPIO\_Pin\_9;

gpio.GPIO\_Speed = GPIO\_Speed\_50MHz;

gpio.GPIO\_OType = GPIO\_OType\_PP;

gpio.GPIO\_PuPd = GPIO\_PuPd\_UP;

GPIO\_Init(GPIOA, & gpio);

gpio.GPIO\_Mode = GPIO\_Mode\_AF;

gpio.GPIO\_Pin = GPIO\_Pin\_10;

gpio.GPIO\_Speed = GPIO\_Speed\_50MHz;

gpio.GPIO\_OType = GPIO\_OType\_PP;

gpio.GPIO\_PuPd = GPIO\_PuPd\_UP;

GPIO\_Init(GPIOA, & gpio);

*// Обязательно вызываем эту функцию*

GPIO\_PinAFConfig(GPIOA, GPIO\_PinSource9, GPIO\_AF\_USART1);

GPIO\_PinAFConfig(GPIOA, GPIO\_PinSource10, GPIO\_AF\_USART1);

*// Настраиваем модуль USART*

USART\_InitTypeDef usart;

USART\_StructInit(&usart);

usart.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;

usart.USART\_BaudRate = 9600;

USART\_Init(USART1, &usart);

*// Включаем прерывания и запускаем USART*

NVIC\_EnableIRQ(USART1\_IRQn);

USART\_Cmd(USART1, ENABLE);

*// включаем прерывание по приему данных*

USART\_ITConfig(USART1, USART\_IT\_RXNE, ENABLE);

А вот сам обработчик прерываний:

void USART1\_IRQHandler()

{

*// Убеждаемся, что прерывание вызвано новыми данными в регистре данных*

if (USART\_GetITStatus(USART1, USART\_IT\_RXNE) != RESET)

{

*// То ради чего все затеяли – принимаем данные )*

receivedData[receivedDataCounter] = USART\_ReceiveData(USART1);

*// Приняли? Увеличиваем значение счетчика!*

receivedDataCounter ++;

*// Приняли 16 байт – выключаем все нафиг, данные у нас ;)*

if (receivedDataCounter == 16)

{

USART\_ITConfig(USART1, USART\_IT\_RXNE, DISABLE);

}

*// Чистим флаг прерывания*

USART\_ClearITPendingBit(USART1, USART\_IT\_RXNE);

}

}

Если