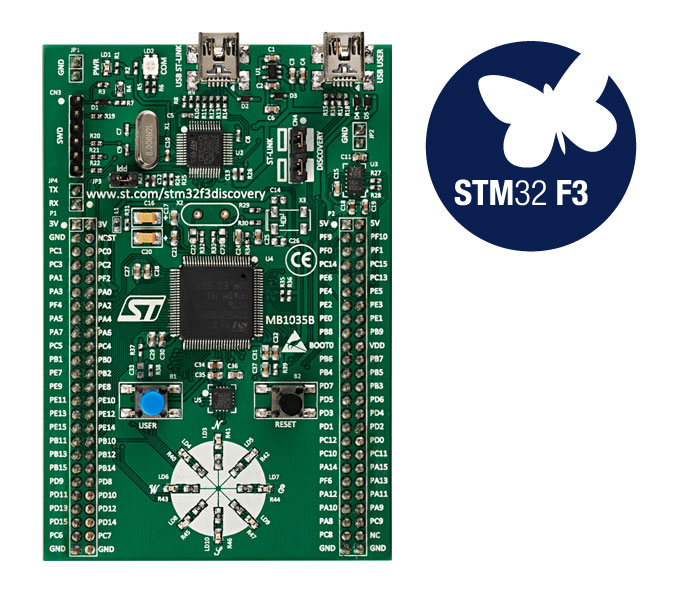
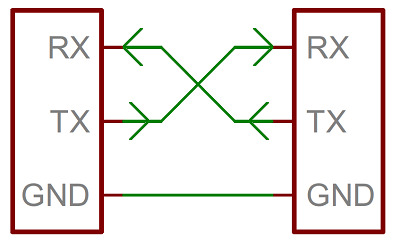
Основы работы с UART в STM32F3



# Основные сведенье об UART

UART (universal asynchronous receiver-transmitter, универсальный асинхронный приёмопередатчик) является одним из наиболее старых стандартов передачи данных. Однако за счет простоты он до сих широко распространен и используется в микроконтроллерах. Отметим его основные особенности:

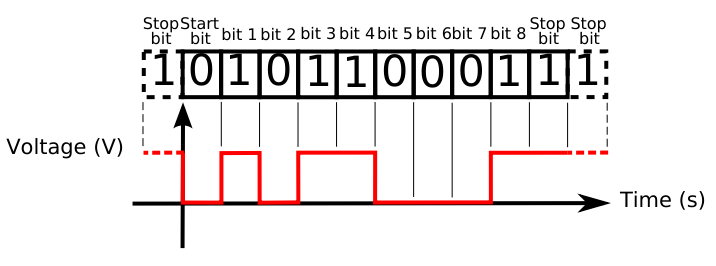
1. Протокол является асинхронным. Это значит, что приемник и передатчик имеют свои генераторы, работающие на одинаковой частоте, и отдельная линия с тактовым сигналом для синхронизации им не требуется.
2. Передача данных происходит последовательно. В каждый момент времени может передаваться только один бит информации.
3. UART поддерживает полнодуплексный режим. Т.е. он может одновременно передавать, и принимать данные.
4. В минимальной конфигурации, достаточно 3-х линий: ground, RX (receiver), TX (transmitter), которые соединяются следующим образом:



1. Данные в линиях кодируются высоким (логическая 1) и низким уровнем напряжения (логический 0). Во время простоя на линии подается высокий уровень напряжения.
2. Данные передаются пакетом, который обычно содержит 1 байт информации и имеет следующий формат:



* 1. старт-бит – нулевой бит для индикации начала передачи
  2. данные – от 5 до 9 бит данных в зависимости от настроек системы
  3. бит четности (parity bit) – используется для контроля целостности переданной информации (опционален)
  4. стоп-бит – единичный бит для индикации конца передачи



# Инициализация UART в STM32F3

Для настройки UART достаточно использовать datasheet для STM32F303xC (в нем описаны альтернативные функции пинов) и reference manual (в нем описаны биты регистров). Ввиду того что некоторые UART имеют возможность использования синхронного режима, и обозначаются в документации как USART.

Рассмотрим по пунктам общую процедуру настройки UART для приема и передачи данных. В каждом пункте будет указан регистр, который нужно изменить/проверить.

1. используя документацию (datasheet) выберете любое устройство UART за исключением UART1, и определите пины, к которым оно подключено. Примечание: обычно имеется несколько вариантов подключения устройства к пинам
2. сконфигурируйте пины для UART:
   1. включите тактирующий сигнал для пинов портов (регистр RCC -> AHBENR)
   2. установите высокую скорость соответствующего порта (регистр GPIOx -> OSPEEDR)
   3. выключите подтягивающие резисторы (регистр GPIOx->PUPDR)
   4. установить двухтактный режим (pull/push) (регистр GPIOx-> OTYPER)
   5. выбрать альтернативную функцию на работу с UART (регистр GPIOx->AFR[0])
   6. переключить пин на альтернативную функцию (регистр GPIOx->MODER)
3. включите тактирующий сигнал для UART (регистр RCC->APB1ENR или RCC->APB2ENR в зависимости от номера UART)
4. настройте UART
   1. включите трансмиттер и ресивер (регистр USARTx->CR1, биты «Transmitter enable» и «Receiver enable»)
   2. выберете источник тактирующего сигнала (регистр RCC->CFGR3, биты «<uart name> clock source selection»). В наиболее простом случае можно указать SYSCLC
   3. рассчитайте и установить значение baud rate register регистра (USARTx->BRR). Инструкция по расчету есть в reference manual
5. запустите UART (регистр USARTx->CR1, бит «USART enable»)
6. проверьте, что приемник и передатчик запущены (регистр USARTx->ISR, биты «Receive enable acknowledge flag» и «Transmit enable acknowledge flag»). Возможно необходимо будет подождать какое-то время, пока советующие биты не будут установлены в 1

На этом базовая настройка UART закончена, и его можно начать использовать. Если нужны какие-то нестандартные параметры (бит четности, длина слова не 8 бит), их нужно установить до запуска UART.

## Передаче данных

Передача байта по UART состоит из следующих шагов:

1. проверять и ждать, пока бит TXE (Transmit data register empty), не будет установлен в 1 (регистр USARTx->ISR)
2. записать байт для передачи в transmit data register (UARTx->TDR)

Для передачи нескольких байтов, данную процедуру нужно повторить несколько раз.

## Примем данных

Приме байта по UART состоит из следующих шагов:

1. поверить, имеются ли данные для приема (регистр UARTx->ISR, бит «Read data register not empty»). Если данных нет, то можно либо ждать, либо выполнять другую работу
2. проверить overrun бит (UARTx->ISR, бит «Overrun error»), и очистить его (UARTx->IСR, бит «Overrun error clear flag»). Если случилось переполнение (данные не были считаны, и пришли новые, которые некуда записывать), то пока бит не будет сброшен, новые данные не будут приниматься
3. считать байт из receive data register (регистр UARTx->RDR)

Для считывания нескольких байтов/символов, необходимо повторить данную процедуру несколько раз.

# Работа на стороне хоста

Для работы с UART на стороне хоста имеются адаптеры USB – TTL serial. При подключении такие адаптеры определяются как виртуальные COM порты. С ними можно работать с использованием любой программы и библиотеки для работы с COM портом. В папке с методическими пособиями доступна программа Termite.exe.

## Форматированный ввод/вывод

Посимвольное формирование сообщений не всегда удобно, особенно если мы передаем не сырые данные, а сообщения для пользователя или отладочную информацию. Чтобы упростить ввод/вывод, перенаправить стандартные потоки ввода-вывода на UART, что позволит на использовать **printf/scanf**.

Допустим мы уже реализовали следующие функции для отправки/считывания символов через UART:

void sendSym(char sym) {

…

}

char readSym() {

…

}

тогда для использования для использования pfrintf/scanf, необходимо реализовать функции:

#include "stdio.h"

…

/\*\*

\* printf hook

\*/

int fputc(int ch, FILE\* f)

{

sendSym(ch);

return ch;

}

/\*\*

\* scanf hook

\*/

int fgetc(FILE\* stream)

{

return readSym();

}

После того, как вы добавить код выше в вашу программу, вы сможете использовать в ней стандартные функции ввода-вывода C: printf/scanf.