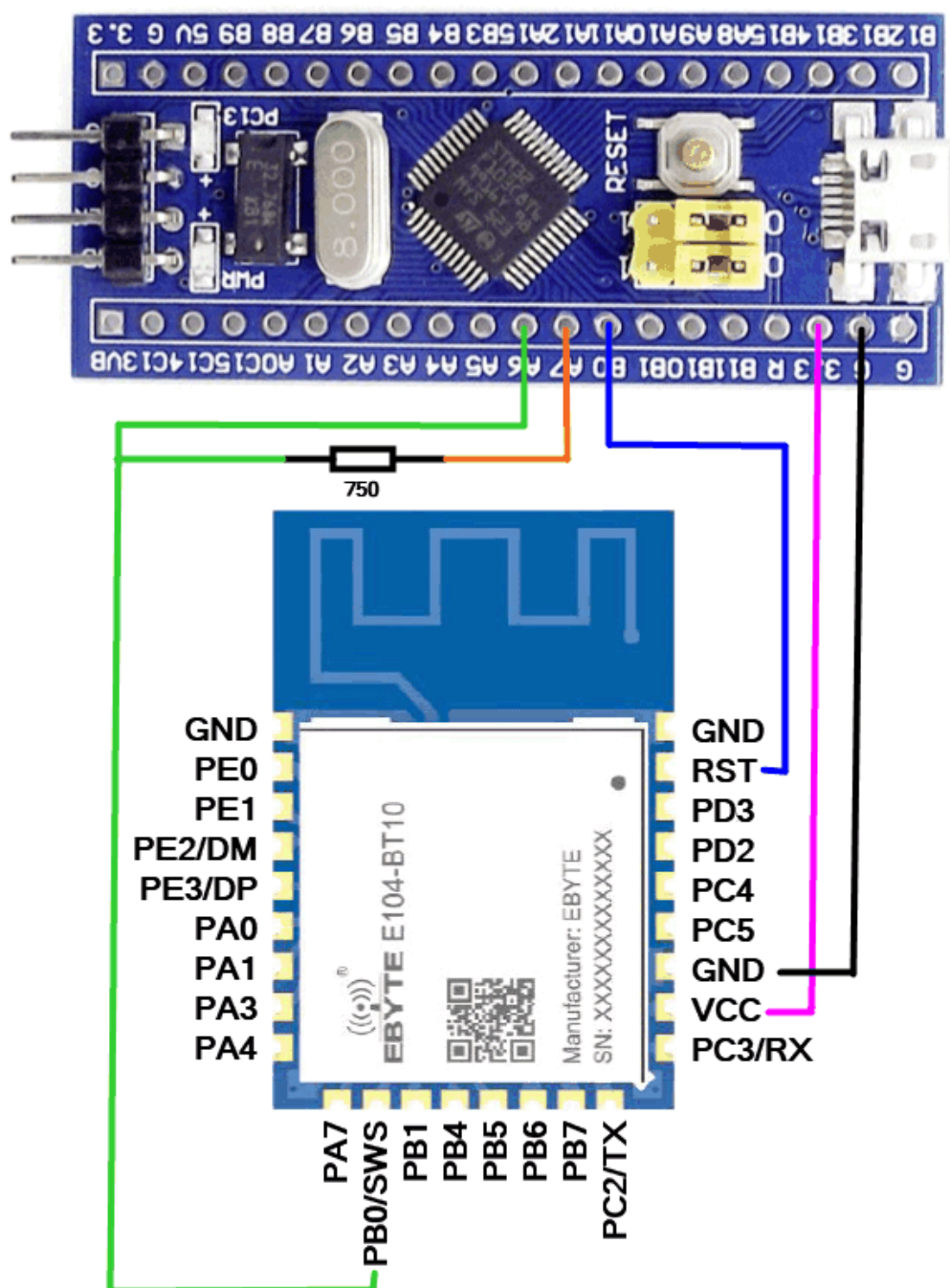


## TLSR-Tool (beta) - програматор на STM32F103C6T8.



### Описания по модулям E104-BT10-N/G :

<http://www.ebyte.com/en/product-view-news.aspx?id=616>

<http://ebyte.com/data-download.aspx?id=472&pid=35#load>

Основное использование:

После подключения модуля и запуска TlsrTool:

- 1) выбираем com порт
- 2) нажимаем “Activate”
- 3) жмем “Flash Erase”\*
- 4) по кнопке “FWrite” выбираем файл для прошивки и ждем завершения.

Далее можем сбросить модуль кнопкой “Reset”, тем самым запустив прошитую программу.

\*В текущей версии TlsrTool.exe перед записью flash необходимо её очистить – “Erase Flash”. Стирание при записи пока не вставлено, т.к. не было нужды и не было известно какой размер сектора flash, а так-же для ускорения программа пропускает запись байтов 0xff, плюс для возможности патча поверх предыдущей записи. Вставить стирание секторов - это дело не более 10 минут... Но это безобразие надо переписывать, т.к. оно лепилось ‘находу’, как пишут скрипт для bash при тестах нового - было неизвестно что ещё потребуется и какой нужен алгоритм

По поводу кода от Telink для TLSR8269 - у них ещё не выложен хидер описания регистров для него. Есть только для 8267.

Он совместим в одну сторону и не всё - проект на 8267 будет работать на 8269, но не наоборот. И это не обязательно, примерно (по моим пробам) на 80%. Чего-то всё-таки разное. Похоже с настройками clk, т.к. модули Telink с 12 MHz...

Ну и половина кода си у них не дописана - вставки от другой версии и требуют полного переписывания. Видимо торопились выложить хоть что-то работающее на 8269 без возможности изменения опций проектов...

Для сборки проектов на TLSR8269 для модуля E104\_BT10 требуется исправить:

1. В исходниках от Telink - частоту кварца на 16 MHz. Обычно это в файле типа:  
ble\_it\_mesh\vendor\8267\_master\_kma\_dongle\app\_config.h

Исправить:

```
#define CRYSTAL_TYPE XTAL_12M // extern 12M crystal
```

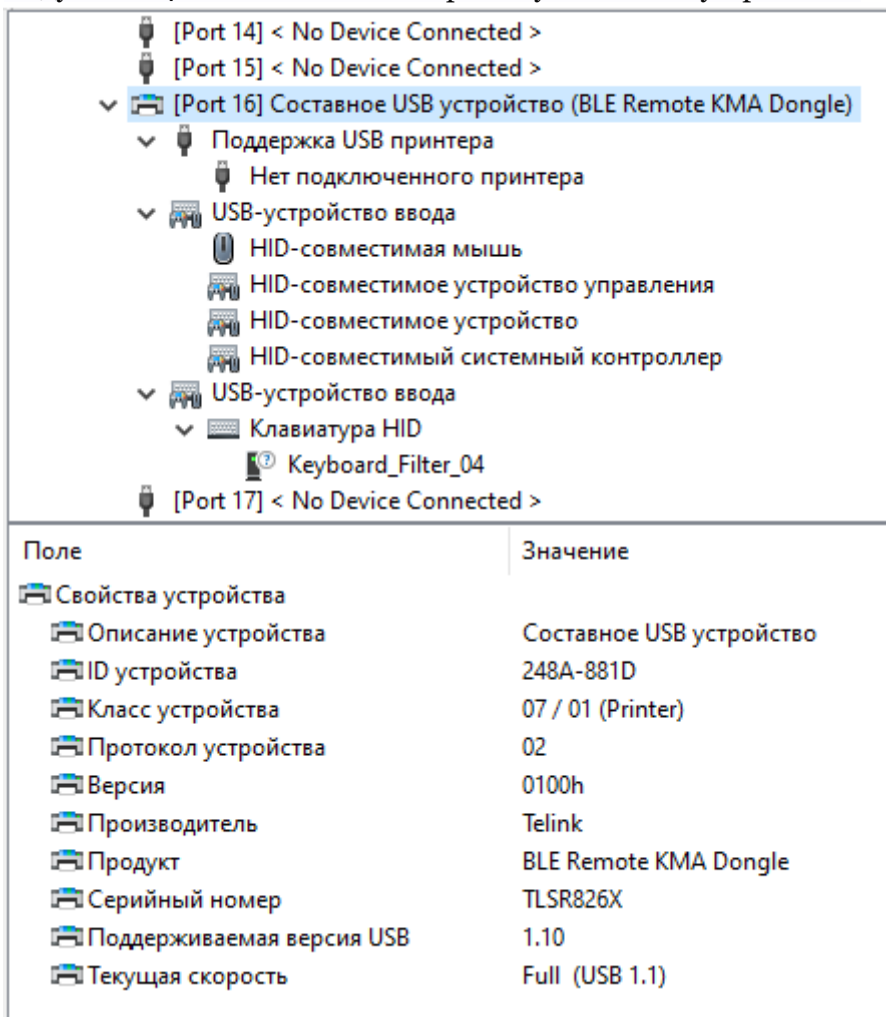
на

```
#define CRYSTAL_TYPE XTAL_16M // extern 16M crystal
```

Посмотреть и исправить другие хидеры проекта на совпадение с CRYSTAL\_TYPE

2. При подключении проводов USB к модулю (DP DM) требуется сигнал активации. Использовать патч “Patch USB DP pull up” (см. в конце документа).

После сборки проекта ble\_lt\_mesh\8269\_mesh\_master\_dongle его можно прошить в модуль E104-BT10 и в компьютере получите такое устройство:



The screenshot shows the Windows Device Manager. Under 'Ports (COM & LPT)', there are three entries: [Port 14] < No Device Connected >, [Port 15] < No Device Connected >, and [Port 16] Составное USB устройство (BLE Remote KMA Dongle). The last entry is expanded, showing a tree of USB devices: Поддержка USB принтера (Нет подключенного принтера), USB-устройство ввода (HID-совместимая мышь, HID-совместимое устройство управления, HID-совместимое устройство, HID-совместимый системный контроллер), and another USB-устройство ввода (Клавиатура HID, Keyboard\_Filter\_04). [Port 17] < No Device Connected > is also shown.

Поле	Значение
Свойства устройства	
Описание устройства	Составное USB устройство
ID устройства	248A-881D
Класс устройства	07 / 01 (Printer)
Протокол устройства	02
Версия	0100h
Производитель	Telink
Продукт	BLE Remote KMA Dongle
Серийный номер	TLSR826X
Поддерживаемая версия USB	1.10
Текущая скорость	Full (USB 1.1)

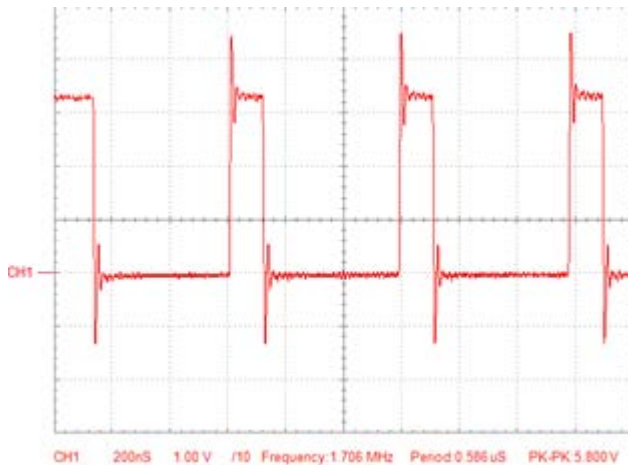
Это устройство работает с Telink утилитами.

## Дополнения.



Swire реализовал как USB-COM на STM32F103C8T6 (или CKS32F103C8T6).

При кварце 16 MHz в модулях E104-BT10, по умолчанию имеем такой расклад CLK swire (так отвечает модуль):



### Формат Telink Swire:

Swire Bit (один бит шины):

12345 (5 CLK swire)

0 \_ \_ \_ \_

1 \_ \_ \_ \_

Передача байта:

Стартовый бит (cmd), 8 бит команды/данных (старший бит первым), bit end

Итого 10 бит.

Порядок передачи:

1-ый байт: Команда START = cmd бит "1", 8 бит байта 0x5A, end бит "0"

2-ый байт: Адрес addrH = cmd бит "0", 8 старших бит адреса, end бит "0"

3-ый байт: Адрес addrL = cmd бит "0", 8 младших бит адреса, end бит "0"

4-ый байт: WR\_ID = cmd бит "0", 1 бит чтение/записи ("1" - чтение, "0" - запись), 7 бит ID устройства, end бит "0"

5-ый байт: Данные:

1. При чтении мастер запускает cmd бит "0", далее устройство отвечает 8-ю битами данных и end битом "0"

2. При записи мастер передает cmd бит "0", 8 бит данных, end бит "0"

Адрес автоматически увеличивается на единицу.

...

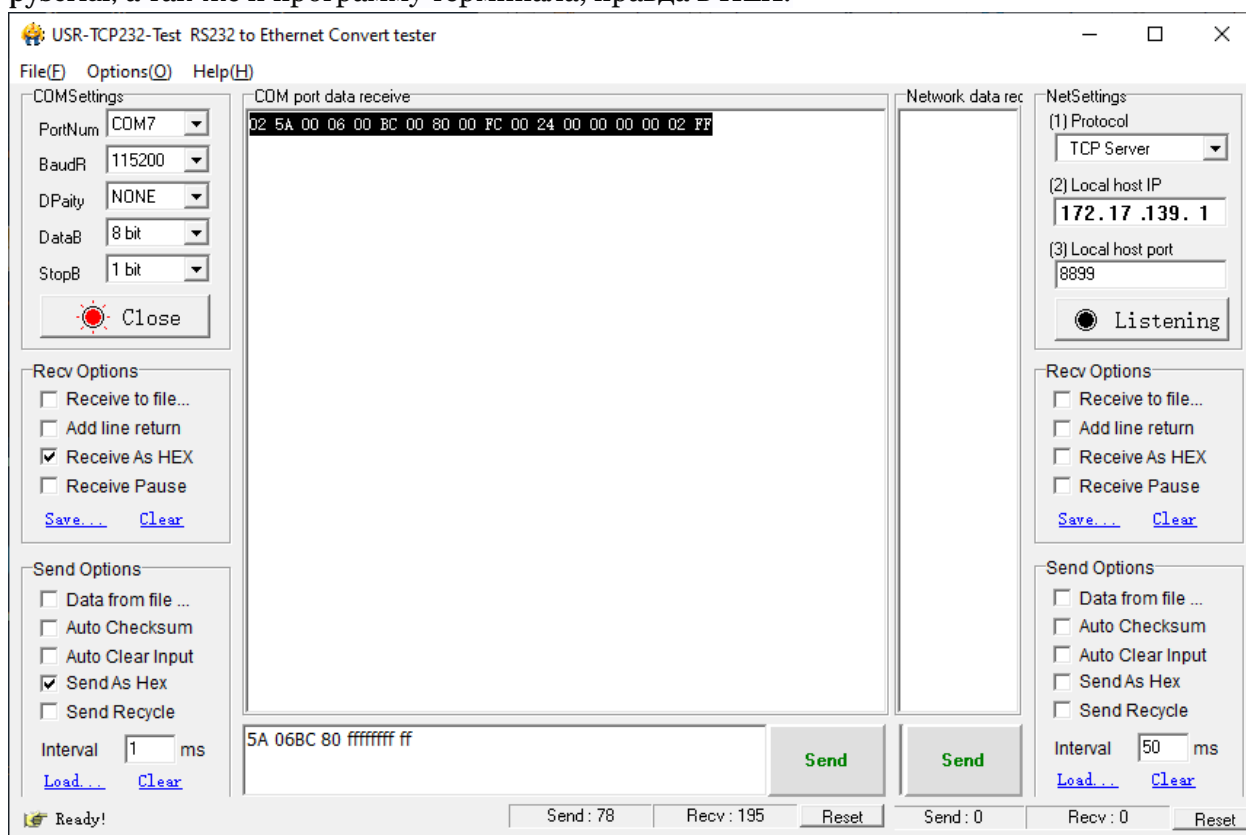
N-ый байт: Команда END = cmd бит "1", байт 0xFF, end бит "0"

Скорость шины запроса swire (к модулю) может варьироваться в достаточном пределе, а ответ чип дает на своей установке делителя в reg\_swire\_clk\_div. По умолчанию там число 5, что при кварце в 16 MHz дает период передачи бита равный 0.58675 us.

Если скорость не соответствует, сначала передаем последовательность установки скорости (5a, 00, b2, 00, делитель, ff), а затем уже читаем. На всякий случай в программе TlsrTool сделана авто-подборка частоты swire – кнопка “ASpeed”.

Telink в своих утилитах и документации предлагает ручную установку синхронизации скоростей swire и в 2 раза ниже, чем по умолчанию использует чип. Автоматом сделать не смогли (?). В моем адаптере для swire на STM32F103C8T6 имеется ограничение для CLK SPI1 в 32 MHz – это  $32/5 = 6.4$  mbps для swire master. Но, для устойчивости приема на стробе в 32 MHz имеем ограничение на приемную CLK swire в 1.5 раза, т.е. прием swire до  $32/5/1.5 = 4.266667$  mbps. Т.к. swire модуля работает на более медленной частоте, то STM32 использует такт SPI в 9 МГц и 7 тактов на бит (1.285714 mbps или один бит в 0.77777 мкс). Такая дискретность подходит для работы с модулями TLSR8269 с кварцами 12 и 16 МГц.

USB у STM32 настроен как USB-COM порт. Это дает возможность использование Python с pyserial, а так же и программу терминала, правда в HEX:



Тогда в COM-порт вводятся байты, как они будут переданы в swire.

Старты и стопы на передачу STM32 разгребает сам, как и декодирование приема.

В примере передается такая последовательность байт: 5A 06 BC 80 ff ff ff ff ff:

Это значит, что будет передано на swire '5A 06 BC 80' (START, addr, RW\_ID), принято 4 байта, и передан END.

На приеме с wire вместо одного переданного байта выдается два принятых байта (swire использует 10 бит на байт) и переданная последовательность удваивается.

Байт1:

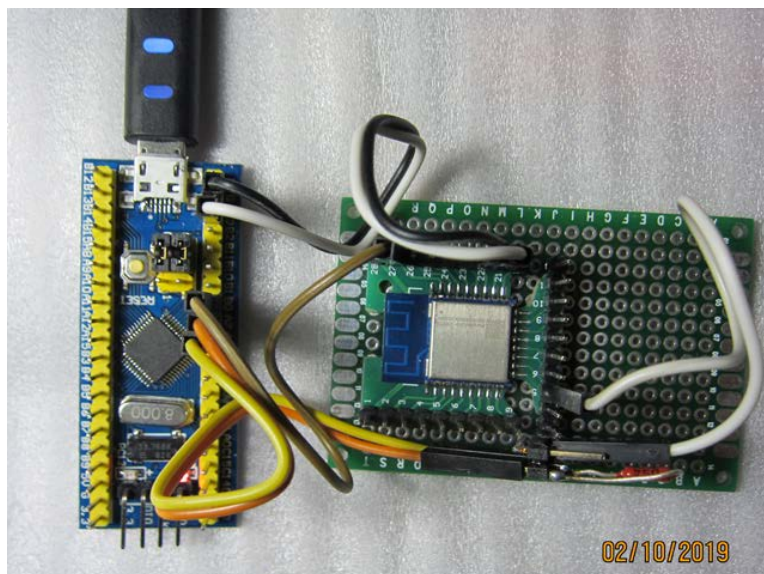
Если принят (декодирован) успешно, то два бита: Бит1 = cmd бит, Бито = end бит, остальные = 0.

Если не успешно (не было приема/строба и т.д.) то данный байт = 0xFF.

Байт2:

Сами 8 бит данных.

Пока такой протокол, но кому надо тот сможет поменять в исходнике программы STM32F103C8T6...



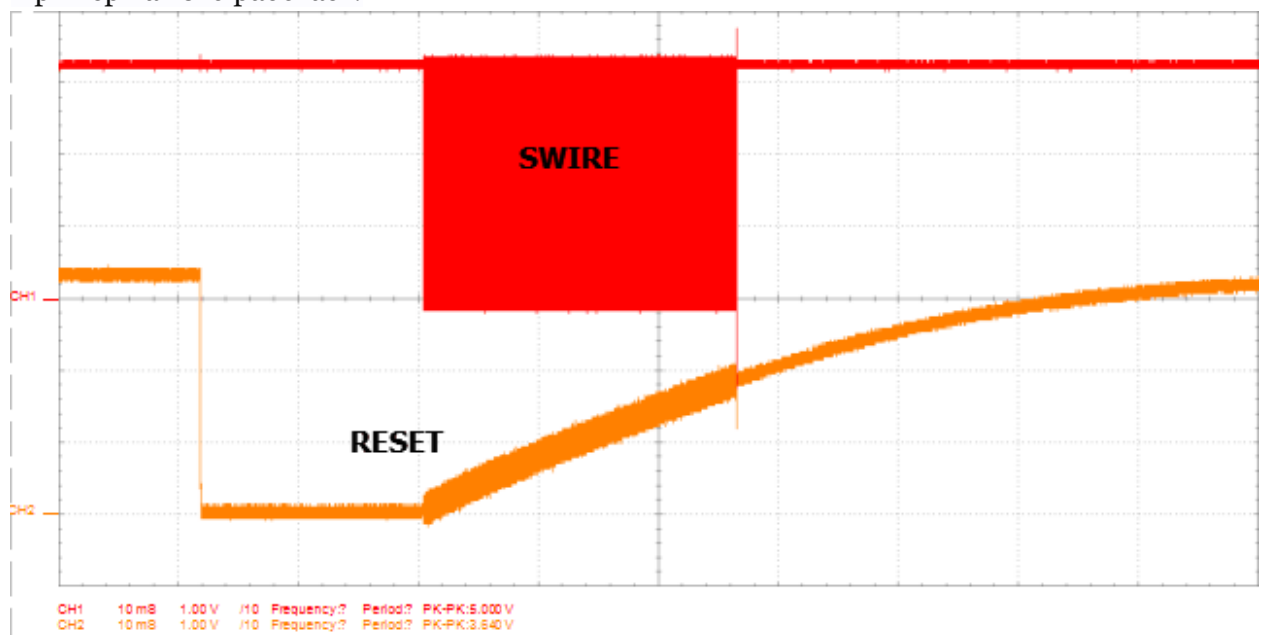
От платы с STM32F103C8 используются 3 вывода:

1. SWO Выход swire (GPIO\_PA7 STM32F103): на резистор в 750 Ом и на SWI.
2. SWI Вход swire (GPIO\_PA6 STM32F103): на SWS модуля TLSR8269
3. RST выход с ОК (GPIO\_PB0 STM32F103): на Reset модуля TLSR8269.

Светодиод (GPIO\_PC13 STM32F103) мигает при транзакции.

Сигнал RST желателен для активации убитого или ещё как замученного модуля.

Пример как это работает:



Данную процедуру, назовем "Activate", выполняет сам STM32 по спец команде.

Доп. команды:

[0x55,0x00] - притянуть RST к GND (в "Activate" не встроена, должна исполняться до него)

[0x55,0x01] - отпустить RST (в "Activate" уже встроена)

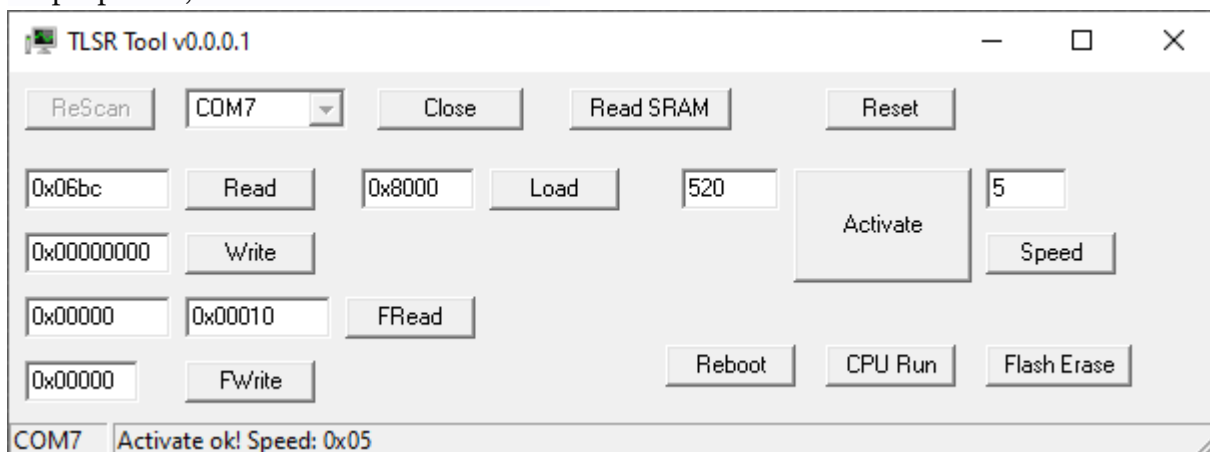
[0x55,0x02,nnnn] - "Activate", где nnn - кол-во циклов (65535 циклов = 3.3 секунды)

Для текущих модулей достаточно 520.

За это время конденсатор на сбросе успевает зарядиться (после отпущения RST).

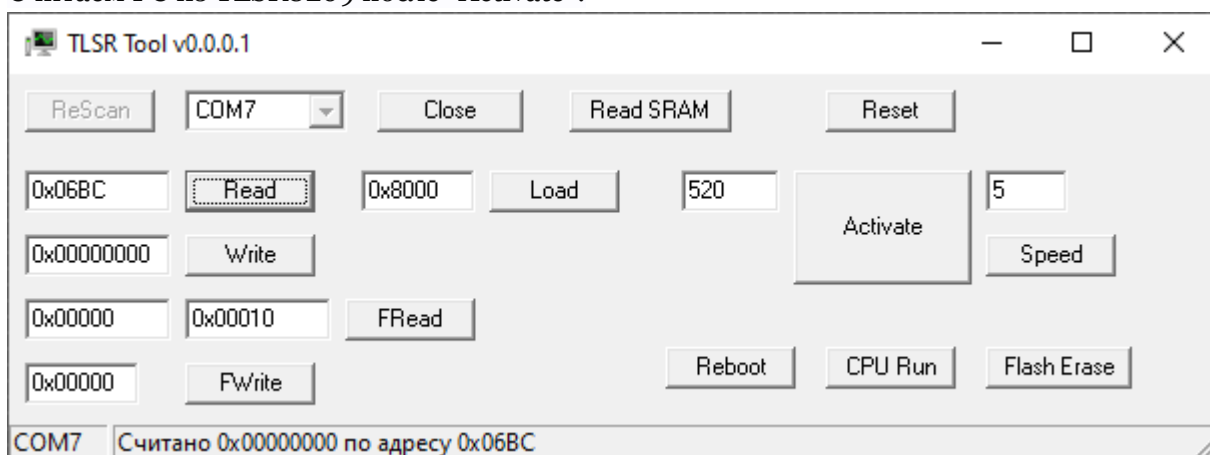


В программе, после нажатия "Activate":



Останавливает CPU модуля до исполнения любой команды.

Считаем PC из TLSR8269 после "Activate":



Адрес PC = 0x00000000.

Скорость swire из модуля указана в строке состояния и выставлена в меню программы после "Activate".

Далее можно прочитать Flash, стереть Flash, записать рабочую прошивку и т.д.

Help-а в данной программе нет и не будет никогда: При подносе курсора - на все пипки вылезают всплывающие подсказки.

Я не собираюсь раскрашивать ничего - пусть берется и занимается кто-то другой для выкладывания "в народ". Для этого отдаю все исходники...

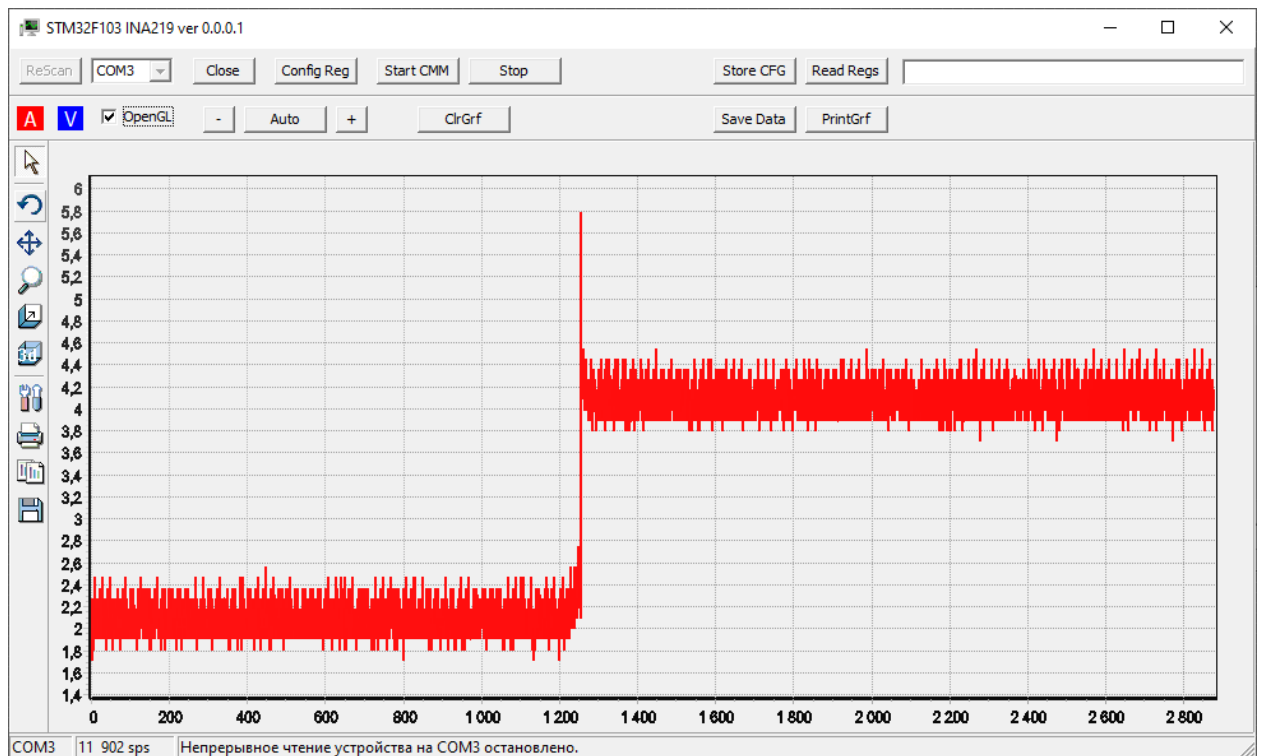
Дополнение по "Активации".

Узнать, что "Активация" выполнена успешно можно и по току потребления.

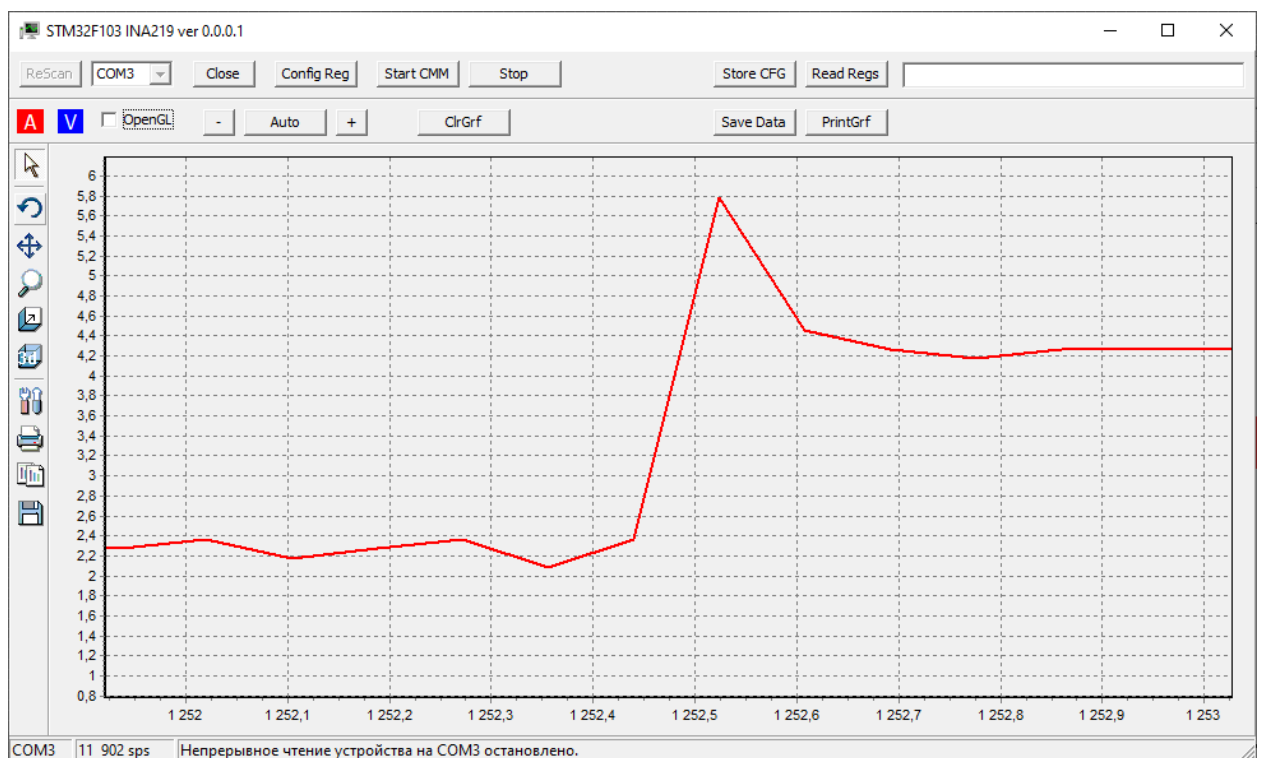
При активном "reset" модуль E104-BT10 с TLSR8269 потребляет ~2.3 мА.

После "Активации", когда CPU остановлен, потребление ~4.2 мА.

Пример замера на INA219 в её предельной скорости замеров:



И уточнение, сколько был активен TLSR8269:



INA219 не справляется - менее её минимального замера (~84 мкс)

Без автоматического RST процедура "Активации" усложняется - вам придется вручную попасть "сбросом" модуля на EVK Telink с точностью в мкс. На их форуме жалуются, что надо тыркать reset и снимать емкость со сброса...

С TLSR Tool можно и вручную дергать "reset", выставив предельное кол-во циклов в 65535, что дает вам время в 3.3 секунды, когда надо успеть нажать "reset" на модуле... или включить ему питание.



## Patch USB DP pull up.

Для правильного выставления активации USB в файле проекта 'rwm\_8267.h' требуется заменить процедуру `usb_dp_pullup_en()` на эту:

```
static inline void usb_dp_pullup_en (int en)
{
    #if (MCU_CORE_TYPE == MCU_CORE_8269) // add pvvx for 8269
    /*
     * The DP pin also supports 1.5KΩ pull-up resistor for USB use. The 1.5KΩ pull up
     * resistor is disabled by default and can be enabled via clearing analog register
     * afe3V_reg00<4>. For the DP pin, user can only enable either 1.5KΩ pull up or
     * 1MΩ/10KΩ pull-up/ 100KΩ pull-down resistor at the same time.
     */
        unsigned char dat = ReadAnalogReg(0x00);
        if (en) {
            dat &= ~(BIT(4)); // enable usb dp 1.5KOhm pull up resistor
        }
        else {
            dat |= BIT(4); // disable usb dp 1.5KOhm pull up resistor
        }

        WriteAnalogReg (0x00, dat);
    #else
    /*
     Wake up mux ANA_E<3> pull up/down controls
     afe3V_reg08<7:6>:
     00 -- No pull up/down resistor
     01 -- 1MOhm pull-up resistor
     10 - 10kOhm pull-up resistor
     11 - 100kOhm pull-down resistor
     */
        unsigned char dat = ReadAnalogReg(0x08);
        if (en) {
            dat = (dat & 0x3f) | BIT(7);
        }
        else {
            dat = (dat & 0x3f) | BIT(6);
        }

        WriteAnalogReg (0x08, dat);
    #endif
}
```