Branca Documentation

Release 1.0

Silica

Mar 16, 2017

Contents

Version 2.00

Date 13 november 2013

We made this manual very carefully, but in any case:

CHAPTER 1

Introduction

The SW is based on **STM32F0-Discovery** and use the **STM Library ver.1.0.0**

I developed the SW using **KEIL C Compiler free version** (ver.4.60.0.0) but I kept the structure of the STM library so is theoretically possible rebuild my project using:

Atollic, IAR and Tasking.

This SW is certainly not exhaustive but is intended to give you the basic functions to manage the STM WiFi module.

As improvements are recommended to implement error handling more in-depth than is currently implemented.

Before you continue please see this **tutorial and read the manual:** STM_WiFi_info

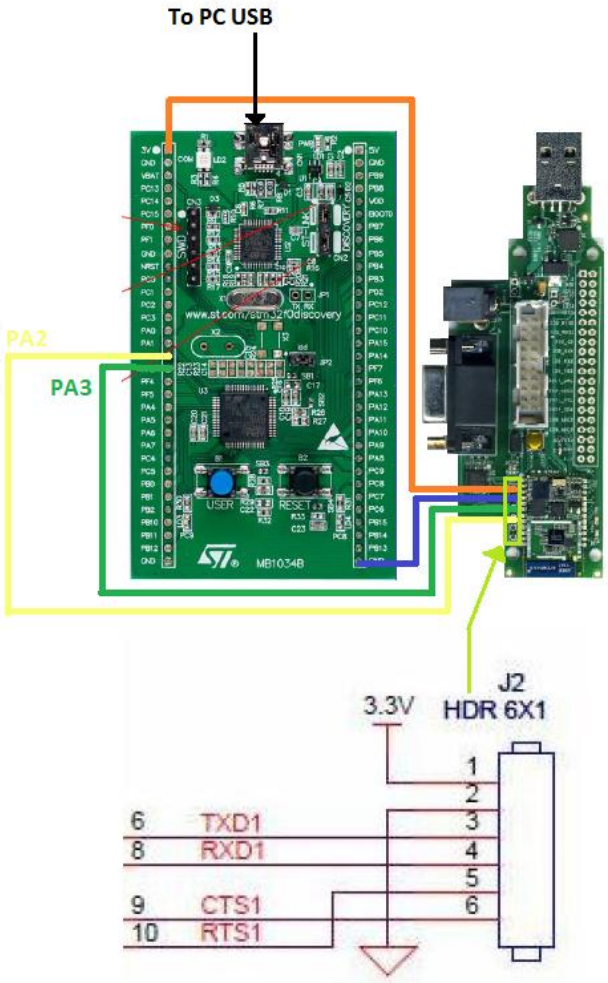
See also here:

http://www.emcu.it/WiFi/WiFi.html

How to connect STM WiFi module to STM32F0-Discovery

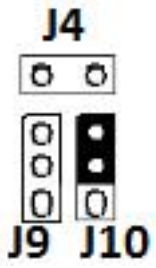
Connections for: SAGRAD EvaBoard

TX	UART Transmit line	PA2	USART2_TX
RX	UART Receive line	PA3	USART2_RX



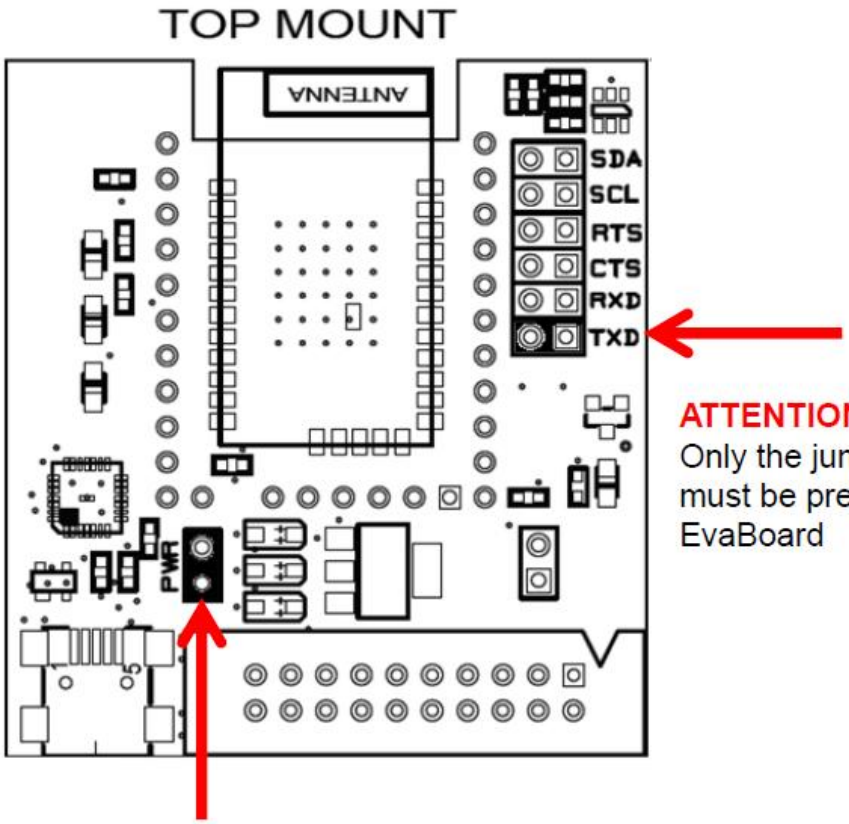
If you need to monitor the strings that are sent from STM32F0-Discovery and STM WiFi module do this:

Remove the jumper **J9** and **J4** , next connect the STM WiFi module to the **Tera Term** .



The connections are shown below for **SILICA STM WiFi EvaBoard** .

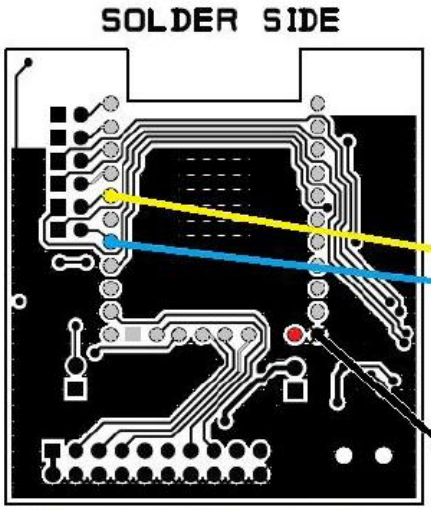
Connections for: SILICA STM WiFi EvaBoard



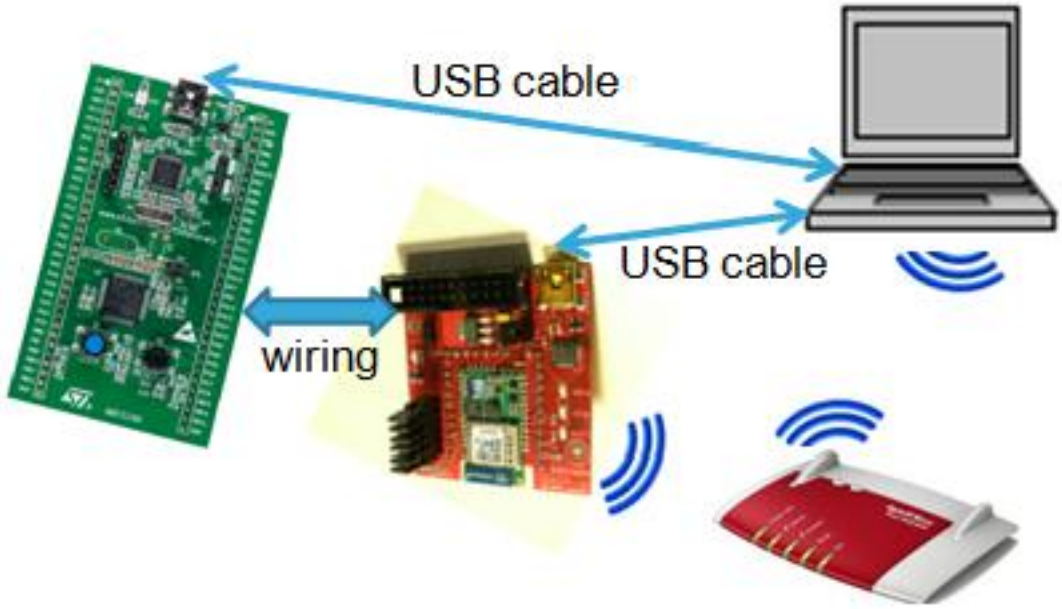
ATTENTION:
Only the jumpers: **PWR** and **TXD** must be present on SILICA STM WiFi EvaBoard

TX	UART Transmit line	PA2	USART2_TX
RX	UART Receive line	PA3	USART2_RX

5V tollerant

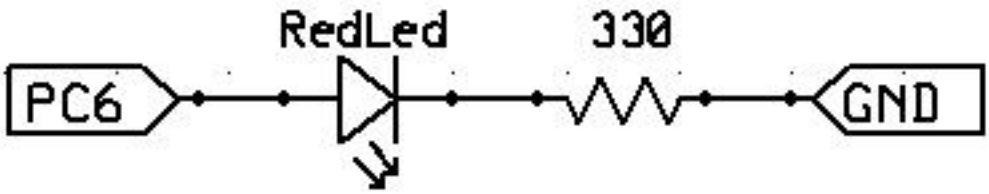


RXD1	8	UART1 Receive data input	5V tolerant
TXD1	6	UART1 Transmit data output	5V Tolerant



Optionally: RED led

if you connect a led (see schematic below)



from **PC6** and **GND** , you have the ability to monitor the waiting for the response from the STM WiFi module.

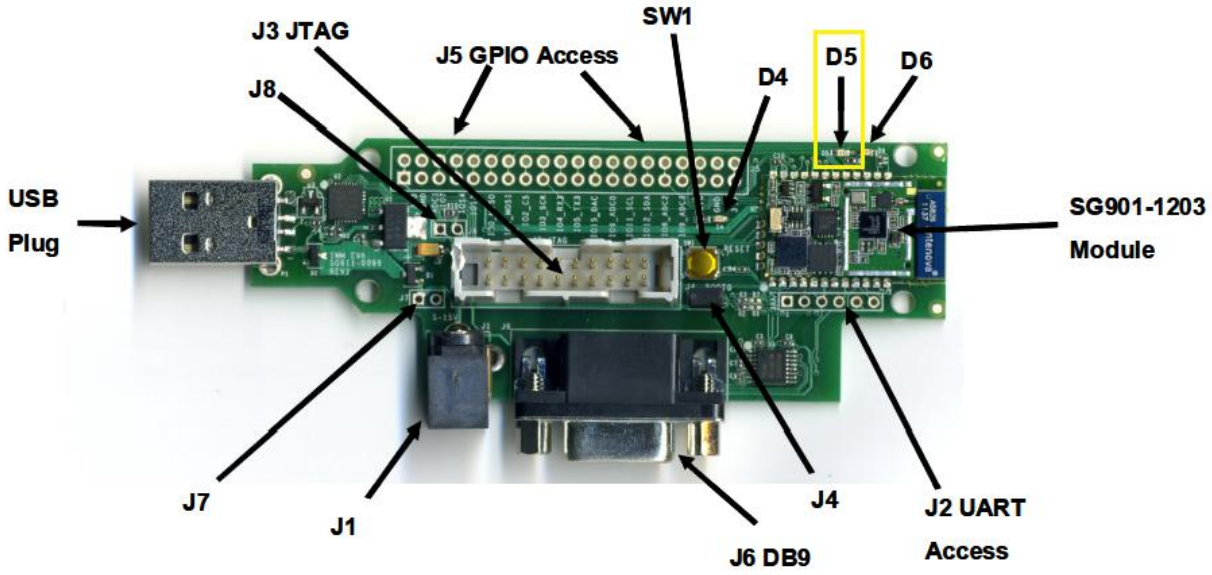
How to use the software

For do the connection (after loaded the SW on STM32F0-Discovery) **just press and release the blue button** on the STM32F0-Discovery. At this point you see the **Blue led** that flashing and the **Red led** that changes from OFF to ON.

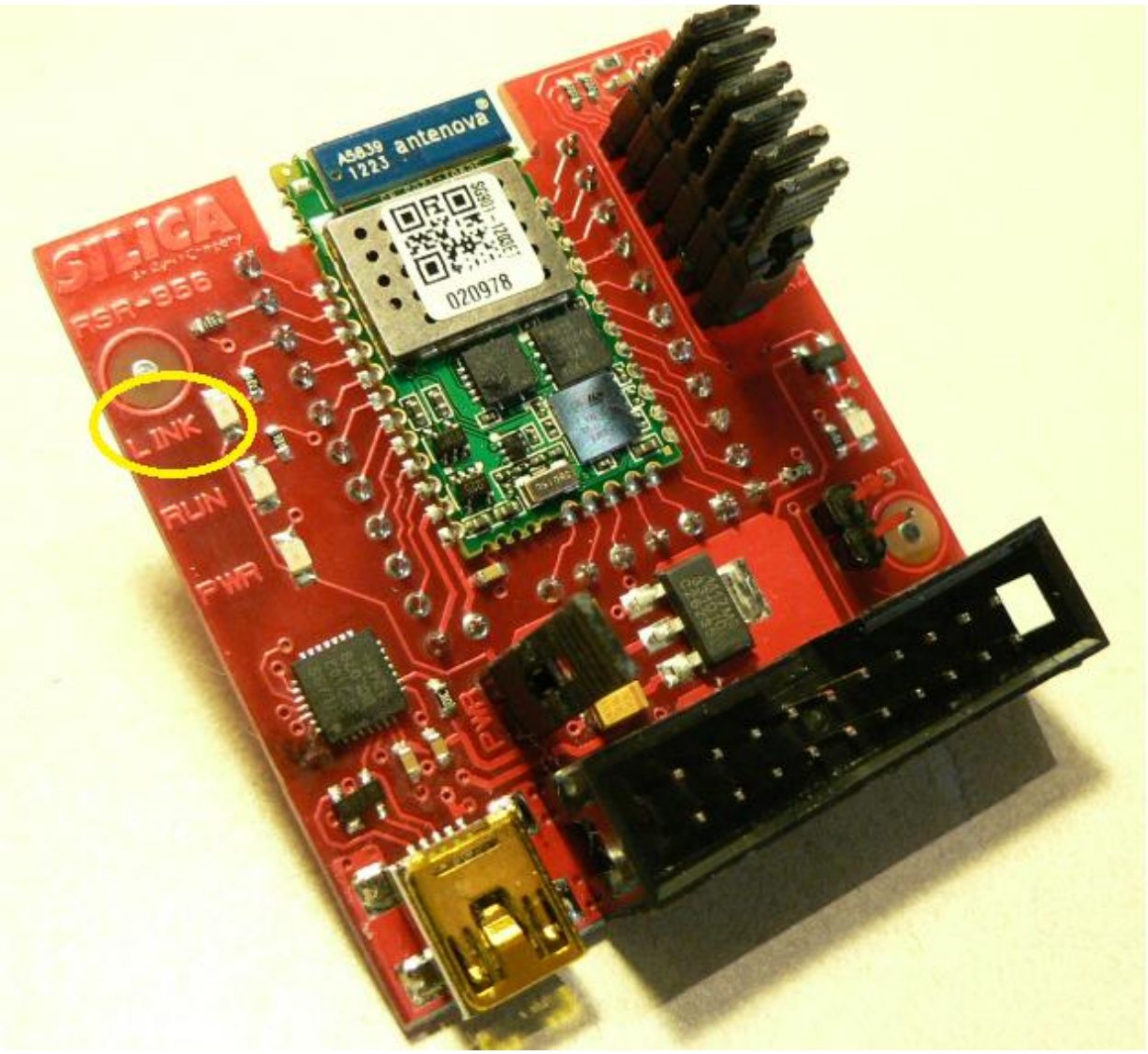
After some seconds, **Blue** and **Green** LED flash, and this means that the module STM WiFi is trying to connect to your WiFi router.

After 2 0/60 sec , **Blue** and **Green** LEDs turn OFF and this means that the connection is made.

If you use the **SAGRAD WiFi module** , the **led D5 must be ON** , this means that the WiFi connection is active.



If you use the **SILICA STM WiFi EvaBoard** , the led **LED2** (LINK) must be ON, this means that the WiFi connection is active.



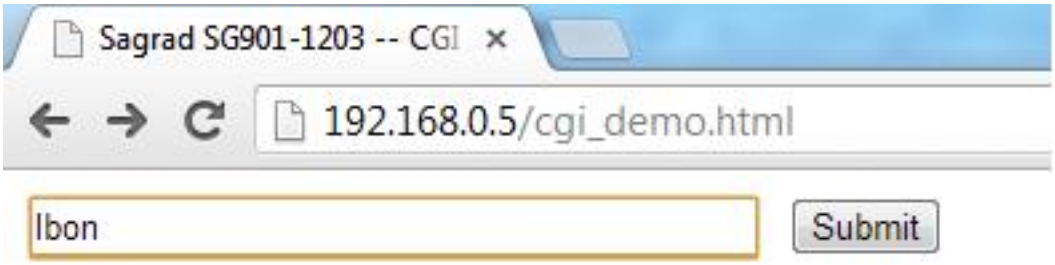
At this point, it is also loaded on the STM WiFi module, the html page named: **led.hmtl**

This page show the status of the LEDs mounted on the STM32F0-Discovery .

Now open the html page: **cgi_demo.html** this page is used to send commands to STM WiFi Module.

- To do this you need to know the IP address of the STM WiFi module.
- To find it I recommend you to use the **Angry IP Scanner** .
- **Suppose the the STM WiFI IP is:** 168.169.0.5
- Open your browser and type:

  192.168.0.5/cgi_demo.html



The custom commands (implemented on STM32F0-Discovery) for control the STM WiFi module are:

- **lgon** – TurnON the green LED
- **lgoff** – TurnOFF the green LED
- **lbon** – TurnON the blue LED
- **lboff** – TurnOFF the blue LED
- **X** – Clear RxBuffer
- **reset** – reset the STM WiFi module, it reload the WiFi configuration

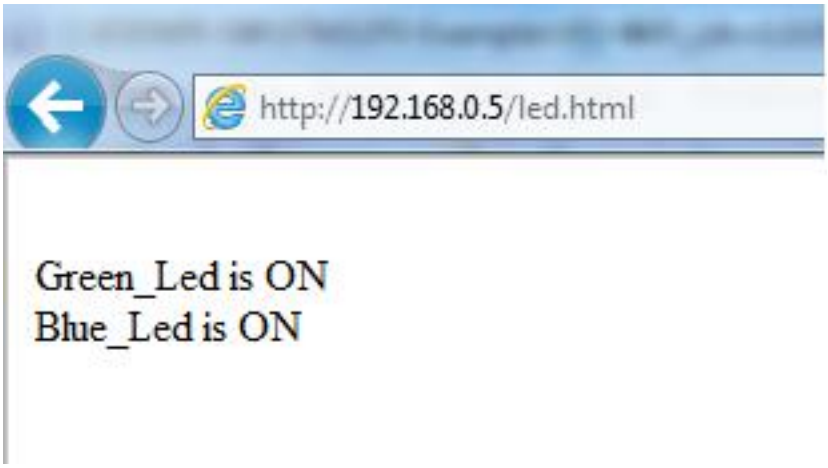
  received from STM32F0-Discovery.

  During the reset the Blue and Green Leds are flashing.

You have the possibility to see the LEDs status in the page:

192.168.0.5/led.html

Remember to reload the **led.html ** page after each command sent using the **cgi demo.html** page.



The Web pages

The web pages that I used were put in some strings, see below.

```
uint8_t TxBuffer_led_pageLVoffLBoff[] = "<html><head><meta content='\22'text/html;␣
↪charset=ISO-8859-1\0x22 http-equiv=\0x22content-type\0x22><title>Leds.html</title></
↪head><body> <br>Green_Led is OFF <br>Blue_Led is OFF<br></body></html>\r\n";
```

```
uint8_t TxBuffer_led_pageLVonLBoff[] = "<html><head><meta content='\22'text/html;␣
↪charset=ISO-8859-1\0x22 http-equiv=\0x22content-type\0x22><title>Leds.html</title></
↪head><body> <br>Green_Led is ON <br>Blue_Led is OFF<br></body></html>\r\n";

uint8_t TxBuffer_led_pageLVonLBon[] = "<html><head><meta content='\22'text/html;␣
↪charset=ISO-8859-1\0x22 http-equiv=\0x22content-type\0x22><title>Leds.html</title></
↪head><body> <br>Green_Led is ON <br>Blue_Led is ON <br></body></html>\r\n";

uint8_t TxBuffer_led_pageLVoffLBon[] = "<html><head><meta content='\22'text/html;␣
↪charset=ISO-8859-1\0x22 http-equiv=\0x22content-type\0x22><title>Leds.html</title></
↪head><body> <br>Green_Led is OFF <br>Blue_Led is ON <br></body></html>\r\n";
```

To fit into a character string the character: **“**
you must use the following syntax: **‘22’**

The variables

The Leds status

```
uint8_t LedG=0; // Led Greem 0==OFF 1==ON
uint8_t LedB=0; // Led Blue 0==OFF 1==ON
```

The Leds flashing

```
uint8_t LBflash=0; // Led Blue 0==FlashOFF 1==FlashON
uint8_t LGflash=0; // Led Green 0==FlashOFF 1==FlashON
```

The underlying variables are used to configure the STM WiFi module.
Remember to configure the parameters in according to your WiFi network.
In particular, be sure to specify:

- **RouterName** (see the orange line below)
- **RouterPW** (see the yellow line below – password)

```
uint8_t TxBuffer_RouterName[] = "at+s.ssidtxt=U739\n\r";
uint8_t TxBuffer_RouterPW[] = "at+s.scfg=wifi_wpa_psk_text,Ria1\n\r";
uint8_t TxBuffer_RouterPotectionMode[] = "at+s.scfg=wifi_priv_mode,2\n\r";
uint8_t TxBuffer_RouterRadioInSTAMode[] = "at+s.scfg=wifi_mode,1\n\r";
uint8_t TxBuffer_RouterDHCPclient[] = "at+s.scfg=ip_use_dhcp,1\n\r";
uint8_t TxBuffer_RouterSaveSettings[] = "at&w\n\r";
uint8_t TxBuffer_RouterSoftReset[] = "at+cfun=1\n\r";
```

See also the function: *ConfigureWiFi* The underlying variables are used to test the STM WiFi module.

```
uint8_t TxBuffer_AT[] = "at\n\r";
uint8_t TxBuffer_FAIL1[] = "+WIND:42:RX_MGMT:";
uint8_t TxBuffer_FAIL2[] = "+WIND:43:RX_DATA:";
uint8_t TxBuffer_FAIL3[] = "+WIND:44:RX_UNK:";
uint8_t TxBuffer_FAIL4[] = "+WIND:34:WiFi Unhandled Event:";
uint8_t TxBuffer_FAIL5[] = "ERROR:";
```

The RxBuffer; contain the characters received (under interrupt) from USART2 that is connect to STM WiFi module.

```
uint8_t RxBuffer[RXBUFFERSIZE];
```

The **RXBUFFERSIZE** is define in the file: **Definizioni.h**

Virtual EEPROM emulation address defined by the user: **not used in this example** .

I left because it can be useful for some future application. For more info see here: http://www.emcu.it/STM32F0xx/STM32F0-USART1-USART2-SysTick-IO/STM32F0-USART1-USART2-SysTick-IO.html

```
uint16_t VirtAddVarTab[NB_OF_VAR] = {0x5555, 0x6666, 0x7777};
uint16_t VarDataTab[NB_OF_VAR] = {0, 0, 0};
uint16_t VarValue = 0;
uint16_t NumPressBott=0x30;
```

The definitions

The commands for control the leds are shown below.

```
// Define the commands for the Red, Blue Led OFF and ON
#define BLed_OFF GPIO_ResetBits(GPIOC, GPIO_Pin_8) // Blue LED OFF
#define BLed_ON GPIO_SetBits(GPIOC, GPIO_Pin_8) // Blue LED ON
#define GLed_OFF GPIO_ResetBits(GPIOC, GPIO_Pin_9) // Green LED OFF
#define GLed_ON GPIO_SetBits(GPIOC, GPIO_Pin_9) // Green LED ON
#define RLed_OFF GPIO_ResetBits(GPIOC, GPIO_Pin_6) // Red LED OFF
#define RLed_ON GPIO_SetBits(GPIOC, GPIO_Pin_6) // Red LED ON
```

Warning: In the section of the variables there are two commands to flash the LED:

uint8_t LBflash=0; // Led Blue 0==FlashOFF 1==FlashON
uint8_t LGflash=0; // Led Green 0==FlashOFF 1==FlashON

This variables are used in the file **stm32f0xx_it.c** under systick interrupt, see below.

```
/**
* @brief This function handles SysTick Handler.
* @param None
* @retval None
*/
void SysTick_Handler(void)
{
  TimingDelay_Decrement();
  // Used by Delay(nnnn);
  // Routine for Flasing LED ************************************************
  TLampeggio++;
  if (TLampeggio >= rifTLampeggio)
  {
    if (LBflash==1) // Toggle BLUE LED
      GPIOC->ODR ^= GPIO_Pin_8;
    if (LGflash==1) // Toggle GREEN LED
      GPIOC->ODR ^= GPIO_Pin_9;
    TLampeggio=0;
```

```
  }
}
```

Also there are two variables used for remember the status of Green and Blue LEDs that are:

```
uint8_t LedG=0; // Led Greem 0==OFF 1==ON
uint8_t LedB=0; // Led Blue 0==OFF 1==ON
```

Custom commands (implemented on STM32F0-Discovery) to control STM WiFi module are:

- **lgon** – TurnON the green LED
- **lgoff** – TurnOFF the green LED
- **lbon** – TurnON the blue LED
- **lboff** – TurnOFF the blue LED
- **X** – Clear RxBuffer
- **reset** – reset the STM WiFi module, it reload the WiFi configuration

The definitions are shown below.

```
#define RxLBOFF "lboff" // Led Blue OFF
#define RxLBON "lbon" // Led Blue ON
#define RxLGOFF "lgoff" // Led Green OFF
#define RxLGON "lgon" // Led Green ON
#define RxClrBuf "X" // Clear RxBuffer
#define RxReset "reset" // Reset the STM WiFi module,it reload the WiFi config.
                        // received from STM32F0-Discovery.
                        // During the reset the Blue Led is flashing.
```

The received strings used for test the status of STM WiFi are below

```
#define WiFi_IP "WiFi Up" // This means that STM WiFi is connected to WiFi Network
#define WiFi_OK "OK" // This is the answer STM WiFi if a command is successful
```

The defines below are in the file: **Definizioni.h**

```
#define RXBUFFERSIZE 0x7FE // 2Kbyte
#define rifTLampeggio 500 // 500 == 500mS is flashing time
#define DlyAttesaRx 200 // 200 == 200mS is time that the SW waiting before to analyze␣
↪the RxBiffer
#define DlyUpLoadHTMLcode 5000 // 5000 == 5sec Delay time for waiting the upload of␣
↪html page
#define DlyBeforeClrRxBuffer 1000 // 1000 == 1sec Delay time to waiting before clear␣
↪the RxBuffer
#define FAIL 0
#define PASS 1
```

The important functions

The function below is used to configure the **PA0 in Input**. At PA0 is connected the Blue Button that is located on the STM32F0-Discovery.

```
void PA0_InFloating(void);
```

The function below is used to configure the **Output the pins: PC6, PC8 and PC9**. On this pins are connected the Red the Blue and the Green Leds.

```
void PC6PC8andPC9output(void);
```

The function below does a **delay** in mS.

```
void Delay(__IO uint32_t nTime);
Delay(500); // it does a delay of 500mS.
```

The function below is used to **find pBuffer2** (single character) **in pBuffer1**.

Remember that:

- **pBuffer2** must be a single character
- **pBuffer1** normally is the RxBuffer

```
uint8_t BuffOneCH(uint8_t* pBuffer1, uint8_t* pBuffer2, uint16_t BufferLength);
```

This function returns:

- **PASS** if pBuffer2 is present in pBuffer1
- **FAIL** if pBuffer2 is not present in pBuffer1

The function below is used for **clear ** the ** pBuffer**.

```
static void Fill_Buffer(uint8_t *pBuffer, uint16_t BufferLength);
```

The function below is used to see if: **pBuffer2 is in pBuffer1**.

It's the main search function used in this SW.

```
uint8_t Search_B2inB1(uint8_t* pBuffer1, uint8_t* pBuffer2, uint16_t Buffer1Length,␣
↪uint16_t Buffer2Length);
```

This function returns:

- **PASS** if pBuffer2 is present in pBuffer1
- **FAIL** if pBuffer2 is not present in pBuffer1

The function below does the **configuration of the USART2**.

```
void STM_EVAL_COM_2_Init(USART_InitTypeDef* USART_InitStruct);
```

Remember, before you call this function, you must:

- Declare: USART_InitTypeDef USART_InitStructure;
- Configure the parameters of the USART

See below.

```
USART_InitTypeDef USART_InitStructure;
/* USARTx configuration ------------------------------------------------------*/
/* USARTx configured as follow:
   - BaudRate = 115200 baud
   - Word Length = 8 Bits
   - One Stop Bit
   - NO parity
   - NO hardware flow control
```

```
   - Receive and transmit enabled
*/

USART_InitStructure.USART_BaudRate = 115200;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl USART_HardwareFlowControl_None;
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
STM_EVAL_COM_2_Init(&USART_InitStructure);
```

This function is not used, but I left because it can be useful for some future application.

The function below is used for find: pBuffer2 in pBuffer1. First is searched the first character of pBuffer2 in pBuffer1. Next the comparisons continue until the end of the length of pBuffer2 or the first difference of the two buffers.

```
uint8_t SearchBuffer2inBuffer1(uint8_t* pBuffer1, uint8_t* pBuffer2, uint16_t␣
↪Buffer1Length, uint16_t Buffer2Length);
```

This function return:

- **PASSED** if pBuffer2 is present in pBuffer1
- **FAILED** if pBuffer2 is not present in pBuffer1

The function below is used for **configure the STM WiFi module**.

```
uint8_t ConfigureWiFi(void);
```

It send to STM WiFi module:

- **Router Name** and test the OK answer from STM WiFi module
- **Router Password** and test the OK answer from STM WiFi module
- **Router Potection Mode** and test the OK answer from STM WiFi module
- **Router Radio in STA Mode** and test the OK answer from STM WiFi module
- **Router DHCP Client** and test the OK answer from STM WiFi module
- **Save Settings** and test the OK answer from STM WiFi module
- **Turn LEDs** (Blue and Green on STM32F0-Discovery) to flashing
- **Router Soft Reset**
- **Wait until the WiFi Connection ** **is OK** (test the answer → :WiFi Up: (WiFi_IP))
- **Prepare the HTML page** named **LED.HTML** and test the OK answer from STM WiFi module
- **Prepare to UpLoad** the **LED.HTML** page and test the OK answer from STM WiFi module
- **UpLoad ** the ** LED.HTML** page and test the OK answer from STM WiFi module
- **Turn OFF ** the ** LEDs** (Blue and Green on STM32F0-Discovery)

This function return:

- **PASS** if WiFi connection is OK
- **FAIL** if WiFi connection is FAIL

Warning: In the **main.c** there are definitions of strings used to connect the STM WiFi module to your WiFi Router. **Remember to configure the parameters in according to your WiFi network.**

In particular, be sure to specify:

- **Router Name** (see the orange line below)
- **Router Password** (see the yellow line below – password)

For example:

```
uint8_t TxBuffer_RouterName[] = "at+s.ssidtxt=NETGEAR-3G\n\r";
uint8_t TxBuffer_RouterPW[] = "at+s.scfg=wifi_wpa_psk_text, free\n\r";
```

```
/* Private variables ---------------------------------------------------------*/

uint8_t TxBuffer[] = "at\n\r";
uint8_t TxBuffer_RouterName[] = "at+s.ssidtxt=[redacted]\n\r";
uint8_t TxBuffer_RouterPW[] = "at+s.scfg=wifi_wpa_psk_text,[redacted]\n\r";
uint8_t TxBuffer_RouterPotectionMode[] = "at+s.scfg=wifi_priv_mode,2\n\r";
uint8_t TxBuffer_RouterRadioInSTAMode[] = "at+s.scfg=wifi_mode,1\n\r";
uint8_t TxBuffer_RouterDHCPclient[] = "at+s.scfg=ip_use_dhcp,1\n\r";
uint8_t TxBuffer_RouterSaveSettings[] = "at&w\n\r";
uint8_t TxBuffer_RouterSoftReset[] = "at+cfun=1\n\r";
```

The function below is used for **check the commands received from STM WiFi module and if they are correct apply them**. Also **test some STM WiFi module errors**.

```
void TestRxCommand(void);
```

Until now commands accepted are:

- **lgon** – TurnON the green LED
- **lgoff** – TurnOFF the green LED
- **lbon** – TurnON the blue LED
- **lboff** – TurnOFF the blue LED
- **X** – Clear RxBuffer
- **reset** – STM32F0-Discovery reload the WiFi configuration to the STM WiFi module.

  During the reset, the Blue and Green Leds are flashing.

The STM WiFi module errors that (up to now) are tested are:

- Test if received fromSTM WiFI: **+WIND:42:RX_MGMT:** - Unhandled Event: - From network means FAIL. Variable → TxBuffer_FAIL1
- Test if received fromSTM WiFI: **+WIND:43:RX_DATA:** - Unhandled Event: - From network means FAIL Variable → TxBuffer_FAIL2
- Test if received fromSTM WiFI: **+WIND:44:RX_UNK:** - Unhandled Event: - From network means FAIL Variable → TxBuffer_FAIL3
- Test if received fromSTM WiFI: **+WIND:34:WiFi** - Unhandled Event: - From network means FAIL Variable → TxBuffer_FAIL4
- Test if received fromSTM WiFI: **ERROR:** - From network means FAIL Variable → TxBuffer_FAIL5

in case of errors is called the function:

```
ResetSTMWiFIModule_retainsLEDs();
```

The function below is used for
upload into the STM WiFi module the appropriate led.html page.
To perform this function, **LoadAppropite_LedPage**, check the value of:
LedG and **LedB**.

```
void LoadAppropite_LedPage(void);
```

The function below **IS NOT USED**.

I left this function in software for future implementations.

The function below is used for ** reset the STM WiFi module.**

This charging's last loaded configuration to the module STM WiFI and clears the RxBuffer.

```
void ResetSTMWiFIModule(void)
```

The function below is used for ** reset the STM WiFi module but retains the status of the LEDs.**

This function call the: ConfigureWiFi();

```
void ResetSTMWiFIModule_retainsLEDs(void)
```

ATTENTION

At the end of this funtion there is the line:

```
LGflash=1; // Green LED flashing
```

In the final application, this line, should be REMOVED.

Please visit: www.emcu.it

- search