UART communication on STM32 Microcontrollers using HAL

You can use the STM32CubeMX tool to create the necessary config. files to enable the UART-Drivers. The HAL library provides the necessary functions to communicate to with the UART protocol. This feature comes in handy for debugging (printing out variables).

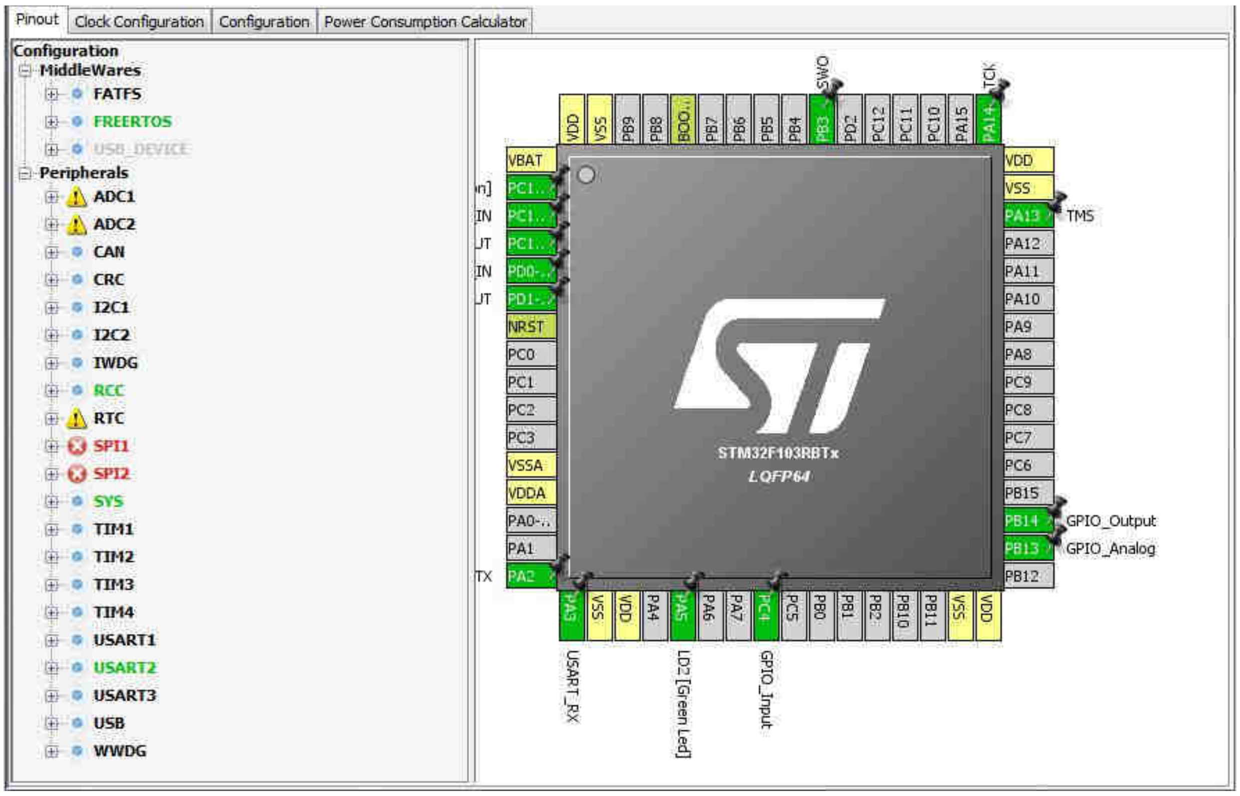
This tutorial uses the following equipment:

- NUCLEO-F072RB Board
- Keil uVision 5 with the necessary packages for Nucleo boards installed
- STLink USB Driver
- STM32CubeMX
- PuTTy, Termite or a similar Serial / COM – console

STM32CubeMX

Generating the config. files from STM32CubeMX.

1. Open STM32CubeMX and open a new project.
2. Select the Nucleo-F072RB from the Borards tab
3. Enable FreeRTOS
4. Set the RCC (HSE & LSE) to Crystal/Ceramic Resonator
5. Enable the USART2 port in Asynchronous mode
6. Go to Project > Generate code
7. Enter a project name and select MDK-ARM V5
8. Generate the code and open the project in Keil uVision



Now let's see what the code generator did

```c
void MX_USART2_UART_Init(void)
{

  huart2.Instance = USART2;
  huart2.Init.BaudRate = 9600;
  huart2.Init.WordLength = UART_WORDLENGTH_7B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  huart2.Init.OneBitSampling = UART_ONEBIT_SAMPLING_DISABLED ;
  huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  HAL_UART_Init(&huart2);

}

```

Important, to make it work you have to change the WordLength parameter to `UART_WORDLENGTH_8B,` since we are sending 8-Bit ascii chars.

You can also adjust the baud rate with the BaudRate parameter.

How to send information (strings) to the console (PC)

To send text data over the debug adapter to the USB-COM port of the computer.

Don't forget to include the string library

```c
#include "string.h"
```

Function to write directly to UART

```c
/* USER CODE BEGIN 4 */
void debugPrint(UART_HandleTypeDef *huart, char _out[]){
       HAL_UART_Transmit(huart, (uint8_t *) _out, strlen(_out), 10);
}

```

It takes the following parameters:

- A pointer to the UART instance to write the data
- The Output string (char)

Function to write to UART and new line termination

```c
void debugPrintln(UART_HandleTypeDef *huart, char _out[]){
       HAL_UART_Transmit(huart, (uint8_t *) _out, strlen(_out), 10);
       char newline[2] = "\r\n";
       HAL_UART_Transmit(huart, (uint8_t *) newline, 2, 10);
}
/* USER CODE END 4 */

```

In addition to the previous function, this one terminates the string with a newline (0x0A) and a carriage return (0x0D) command.

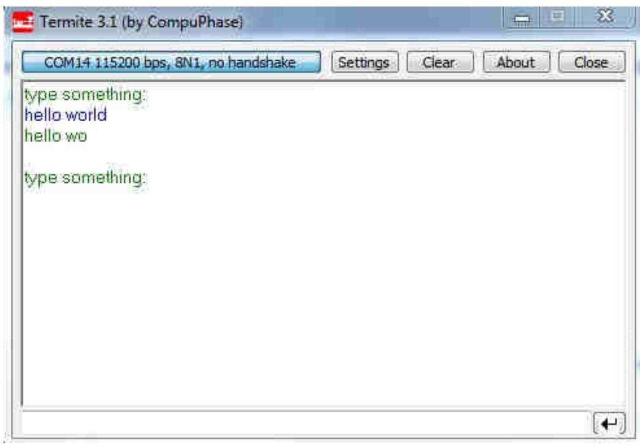
Use the functions as follows.

```c
void StartDefaultTask(void const * argument)
{

  /* USER CODE BEGIN 5 */
  for(;;)
  {
    debugPrint(&huart2,  "oi, mate!");        // print
    debugPrint(&huart2, "\r\n");             // manual new line
    debugPrintln(&huart2, "how are you?");  // print full line
    osDelay(1000);
  }
  /* USER CODE END 5 */
}

```

How to read information (strings) from the console (PC)

```c
void StartDefaultTask(void const * argument)
{

  /* USER CODE BEGIN 5 */
  for(;;)
  {
    uartPrintln(&huart2, "type something:");
    char in[8];
    HAL_UART_Receive(&huart2, (uint8_t *)in, 8, 1000);
    uartPrint(&huart2, "\n");
    HAL_UART_Transmit(&huart2, (uint8_t *)in, 8, 1);
    uartPrint(&huart2, "\n\n");
   /* USER CODE END 5 */
  }
}

```

Create a char array, where you want to save the string to.
`HAL_UART_Receive()` takes this array as the second parameter.
The third parameter specifies the length of the input string, so you have to type word of exactly this size later in the terminal.


Termite 3.1 (by CompuPhase)
COM14 115200 bps, 8N1, no handshake
Settings
Clear
About
Close

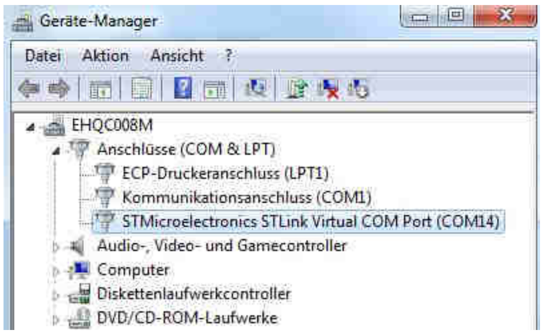
```
type something:
hello world
hello wo

type something:
```

Using a COM-port emulator to receive the output data

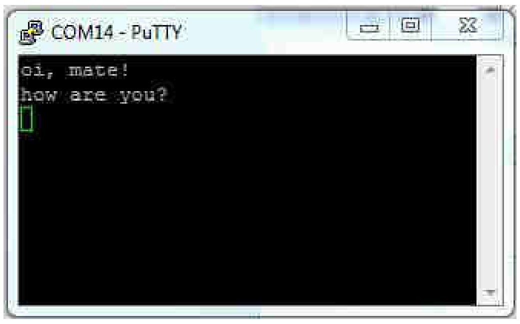
I used PuTTY to do this, but there are a ton of other tools out there, which should work just fine.

First thing, you should check to which port your STLink debugger is connected to.
Look this one up in the Device Manager.

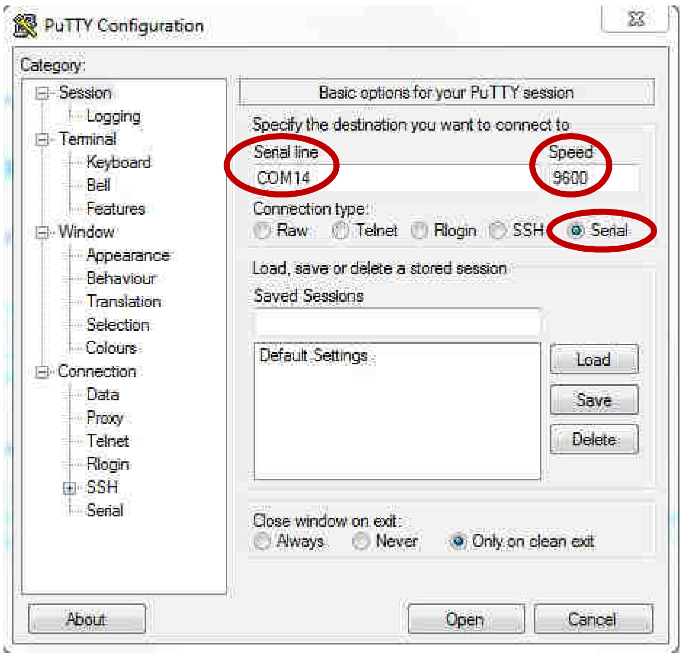
In PuTTY, switch the mode to Serial, enter your COM-Port address, and make sure, you have selected the same baud rate as configured on your STM32 chip.
Then open the connection and you should see the Output in the console.



- EHQC008M
  - Anschlüsse (COM & LPT)
    - ECP-Druckeranschluss (LPT1)
    - Kommunikationsanschluss (COM1)
    - STMicroelectronics STLink Virtual COM Port (COM14)
  - Audio-, Video- und Gamecontroller
  - Computer
  - Diskettenlaufwerkcontroller
  - DVD/CD-ROM-Laufwerke



```
oi, mate!
how are you?
```



Specify the destination you want to connect to
Serial line: COM14
Speed: 9600
Connection type: Raw Telnet Rlogin SSH [✓ Serial]
Default Settings
Close window on exit: Always Never [✓ Only on clean exit]

Document Created by Simon Burkhardt

Simon Burkhardt
electronical engineering
simonmartin.ch

History:

V1.0	tested the code, created this document
V1.1	added information on receiving strings
V1.1.1	added contact info and cc0 notice