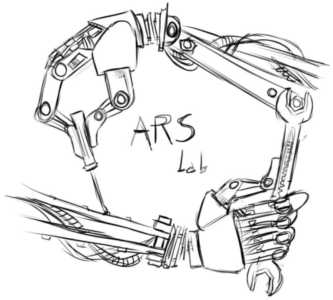
Using the UART with STM32 Microcontrollers

Corrado Santoro

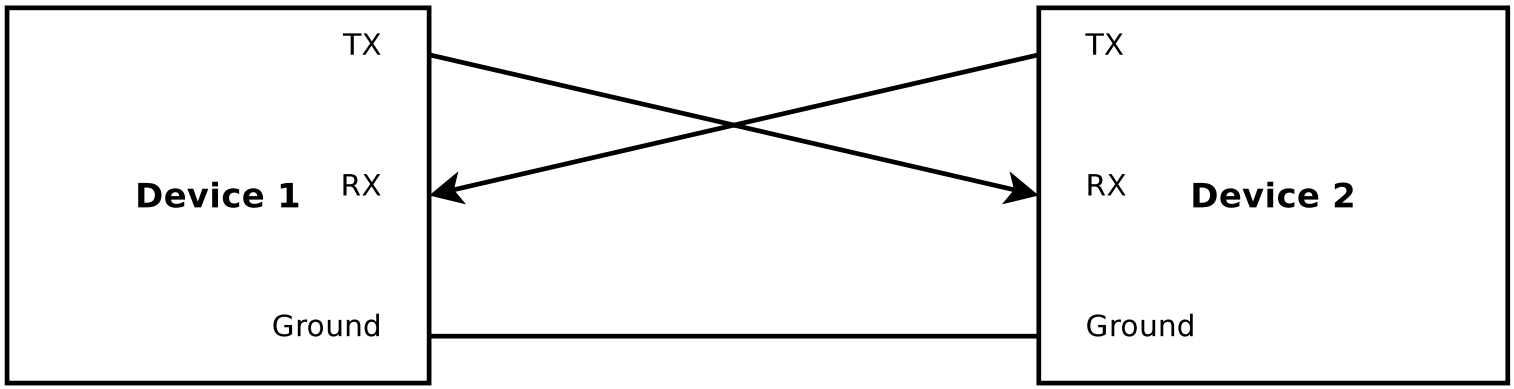
ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



L.S.M. Course

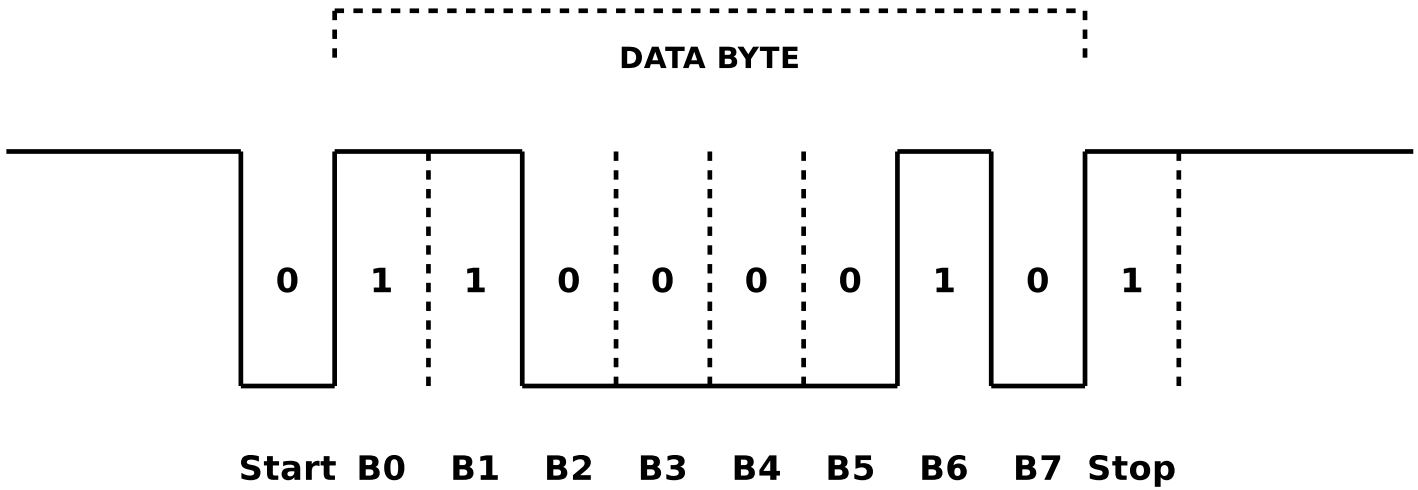
UART in MCUs

- UART = **Universal Asynchronous Receiver/Trasmitter**
- It is commonly referred as **serial port** (or **COM port**)
- It is a peripheral for point-to-point communication between two devices
- Communication occurs **in serial**, i.e. one bit at time
- Two communication PINs: **RX** and **TX**



UART trasmission basics

- When no transmission, the line is set to **Logical "1"**
- Then the software triggers the trasmission of a byte (e.g. "C", hexcode **43**, binary **0100 0011**
- First a **Logical "0"** is transmitted, called **start bit**
- Then the byte is transmitted **LSB first**
- An additional **parity bit** may follow (not in the example); it used for error checking
- One or two **stop bits** (**Logical "1"**) ends the transmission



UART partity

In serial communication, the **parity bit** may be set as:

- **NONE**, the parity bit is not transmitted
- **MARK**, the parity bit is transmitted as logical **"1"**
- **SPACE**, the parity bit is transmitted as logical **"0"**
- **ODD**, the number of "1" in the byte + the parity must be **odd**
- **EVEN**, the number of "1" in the byte + the parity must be **even**

Examples:

- **Data byte = 0x43, Parity Even**, bit stream transmitted (b0 to b7 + parity): `1100 0010 1`
- **Data byte = 0x43, Parity Odd**, bit stream transmitted (b0 to b7 + parity): `1100 0010 0`

UART trasmission parameters

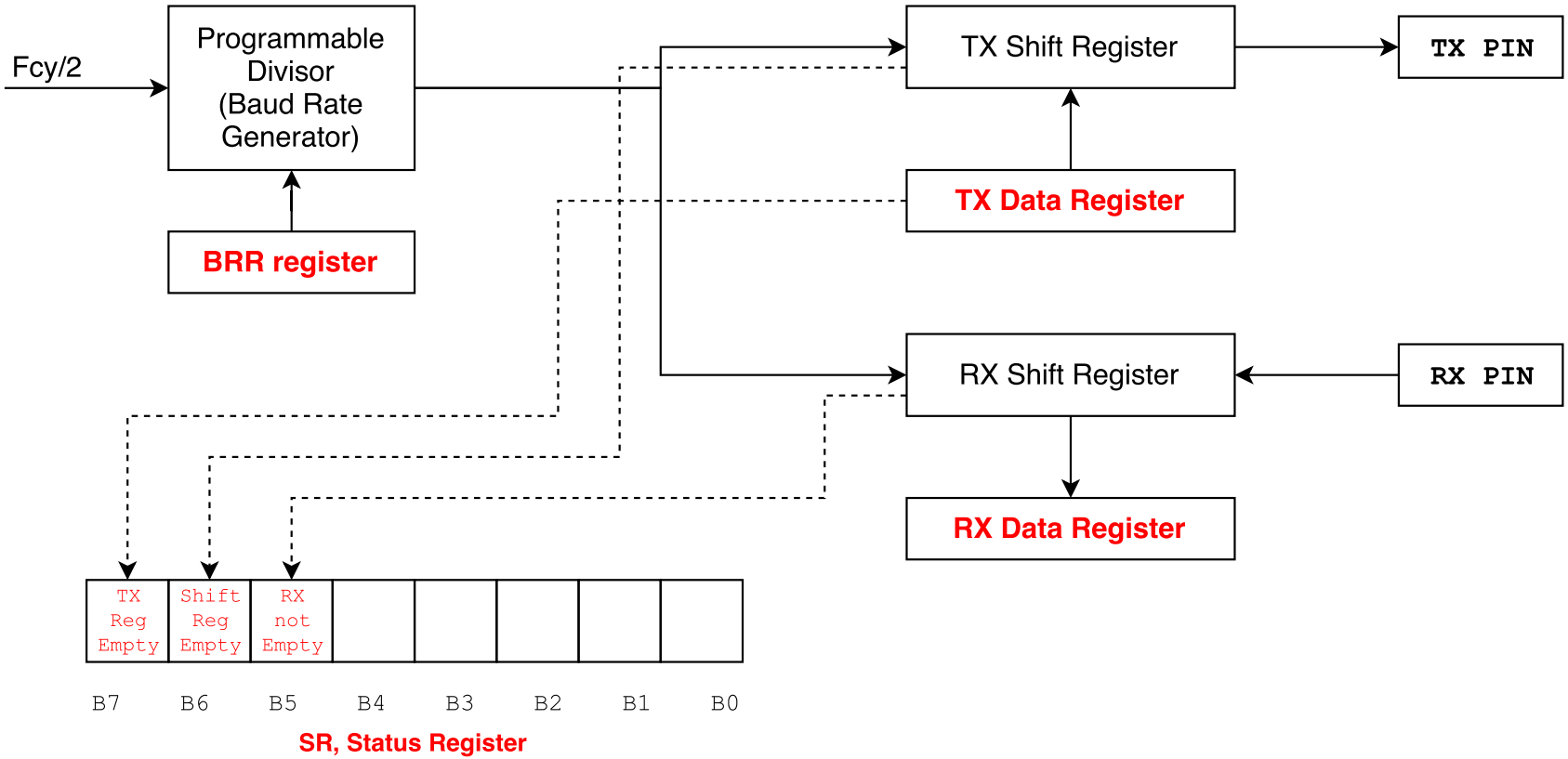
The following parameters must be set in the UART hardware:

- **transmission speed**, in **bps = Bit Per Second** or **baud**
- **number of bits per character**, usually **8**
- **presence/absence of partity bit**, usually **absent**
- **number of stop bits**, usually **1**

A setting **19200,8,N,1** means:

- speed = 19200 bit-per-second;
- bits = 8;
- parity = None;
- stop bits = 1.

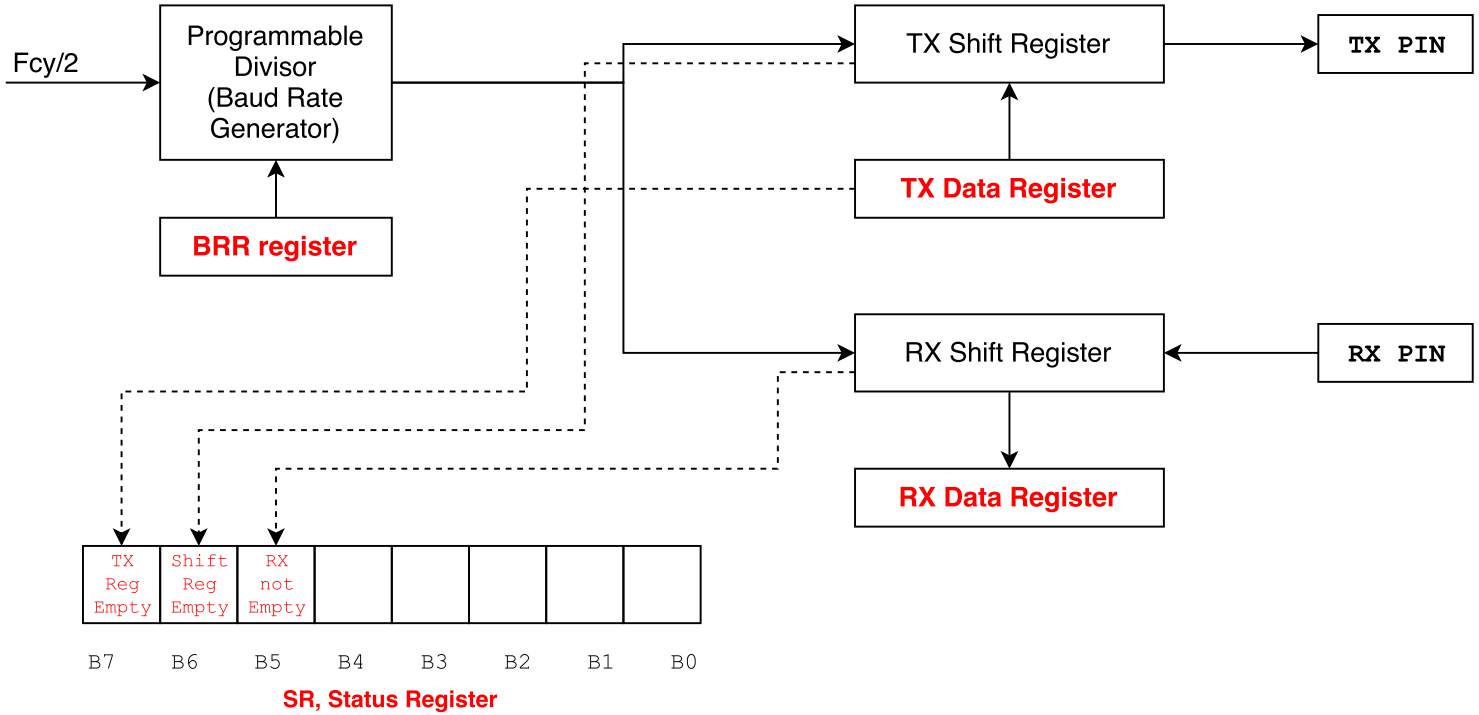
Simplified Schematics of UART



Three main blocks:

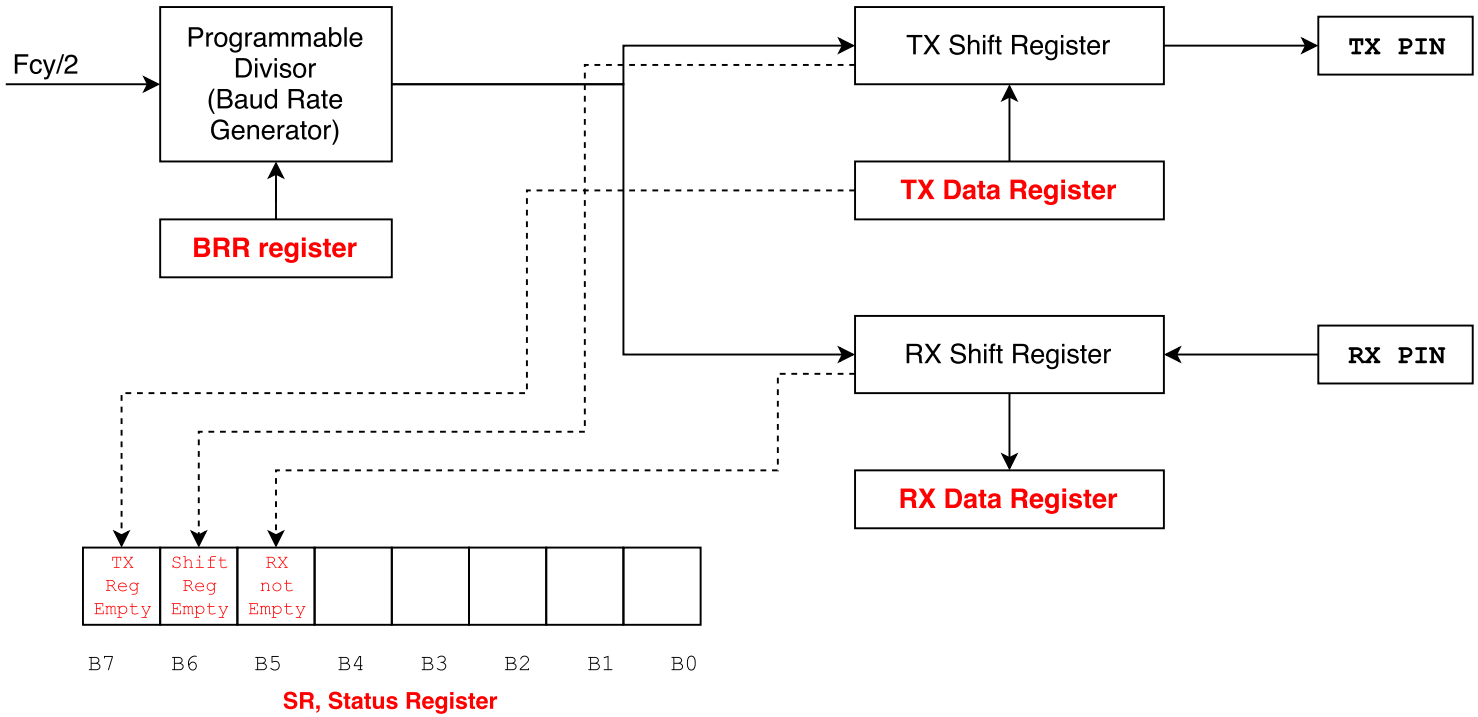
- **Baud Rate Generator**
- **Transmitter Circuit**
- **Receiver Circuit**

UART—Baud Rate Generation



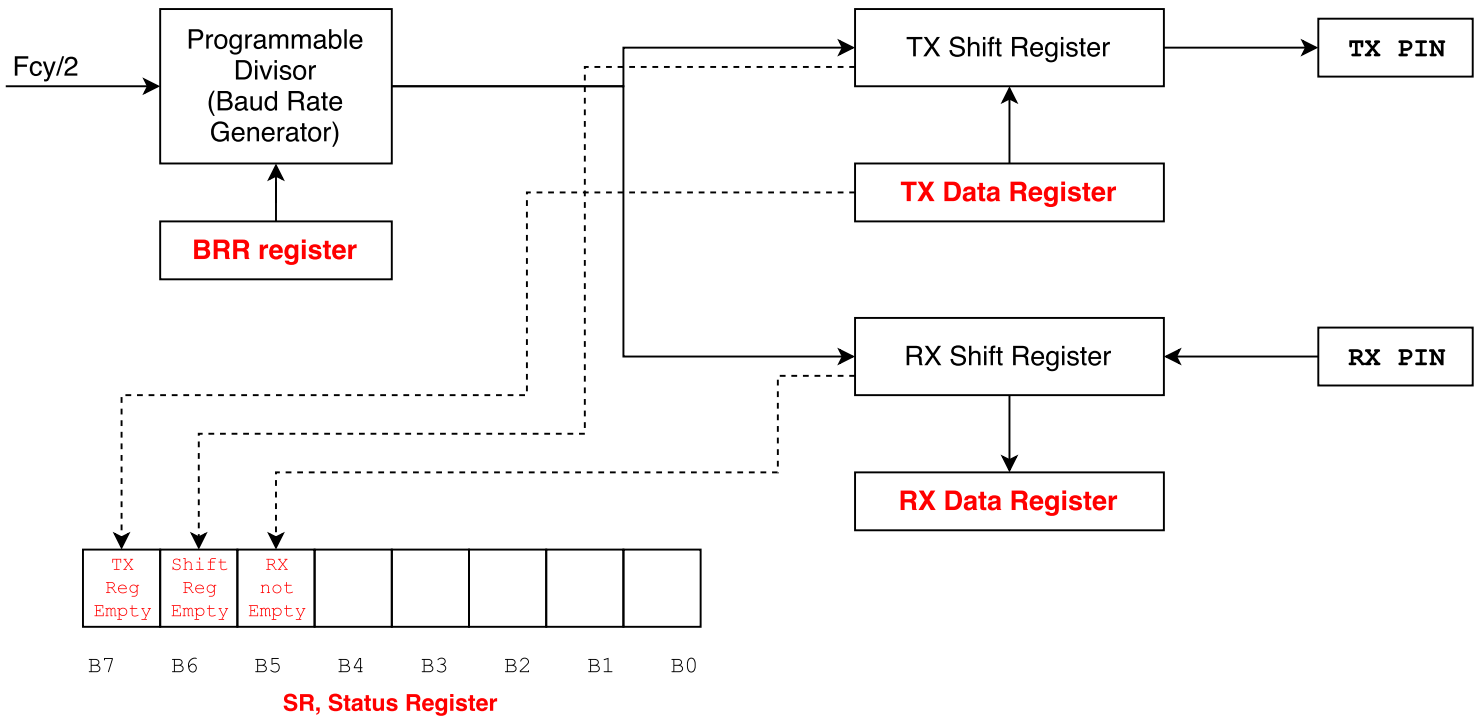
- The **Baud Rate Generator** is responsible of generating the clock for data transmission
- It is a **programmable divisor** that starts from half of the CPU clock frequency ($\frac{84MHz}{2} = 42MHz$ in our boards)

UART—Data Reception



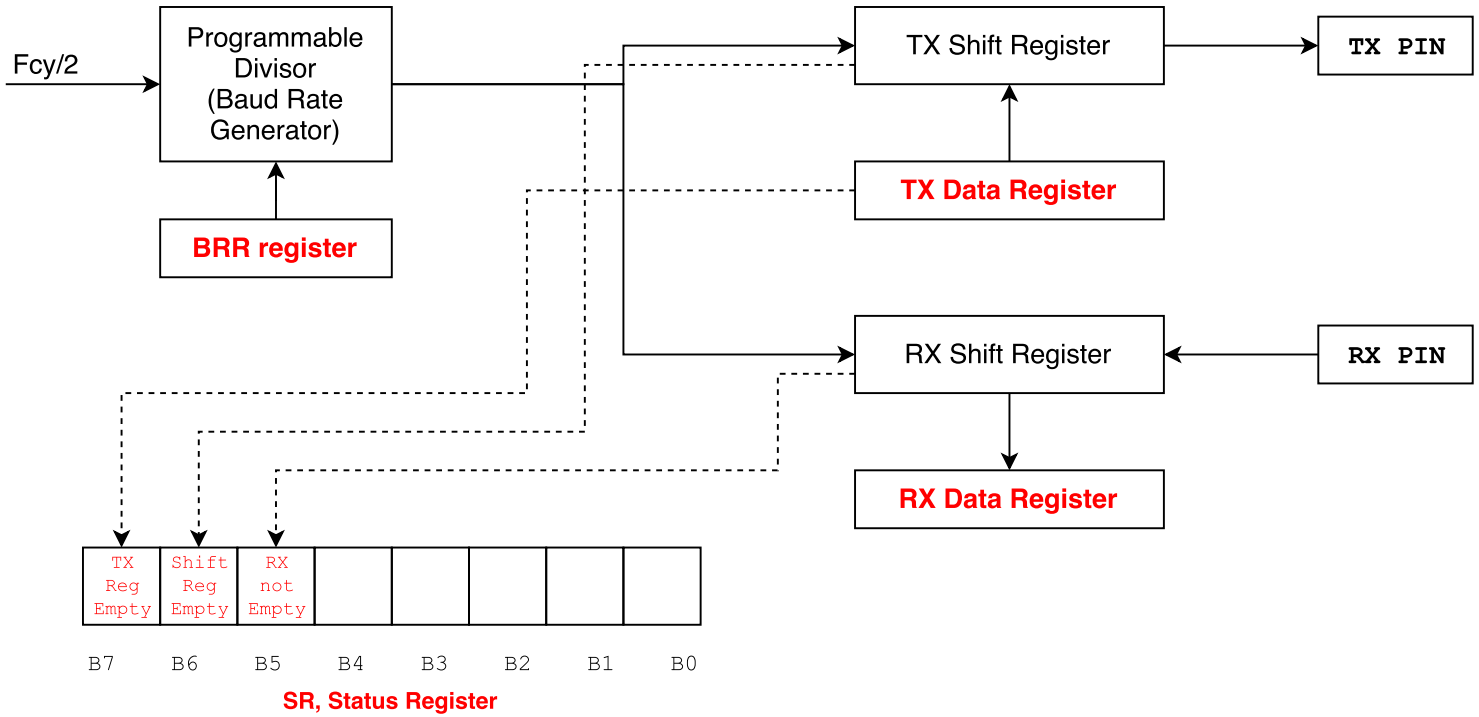
- The **Receiver Circuit** is responsible of receiving data bits, checking correctness and deliver the complete data byte to the software
- It has a **shift register** that gathers one bit at time
- When a byte is completed, the content of the shift register is copied into the **RX Data Register** and the bit **``RX Register Not Empty''** is set in the **Status Register**

UART—Data Transmission (I)



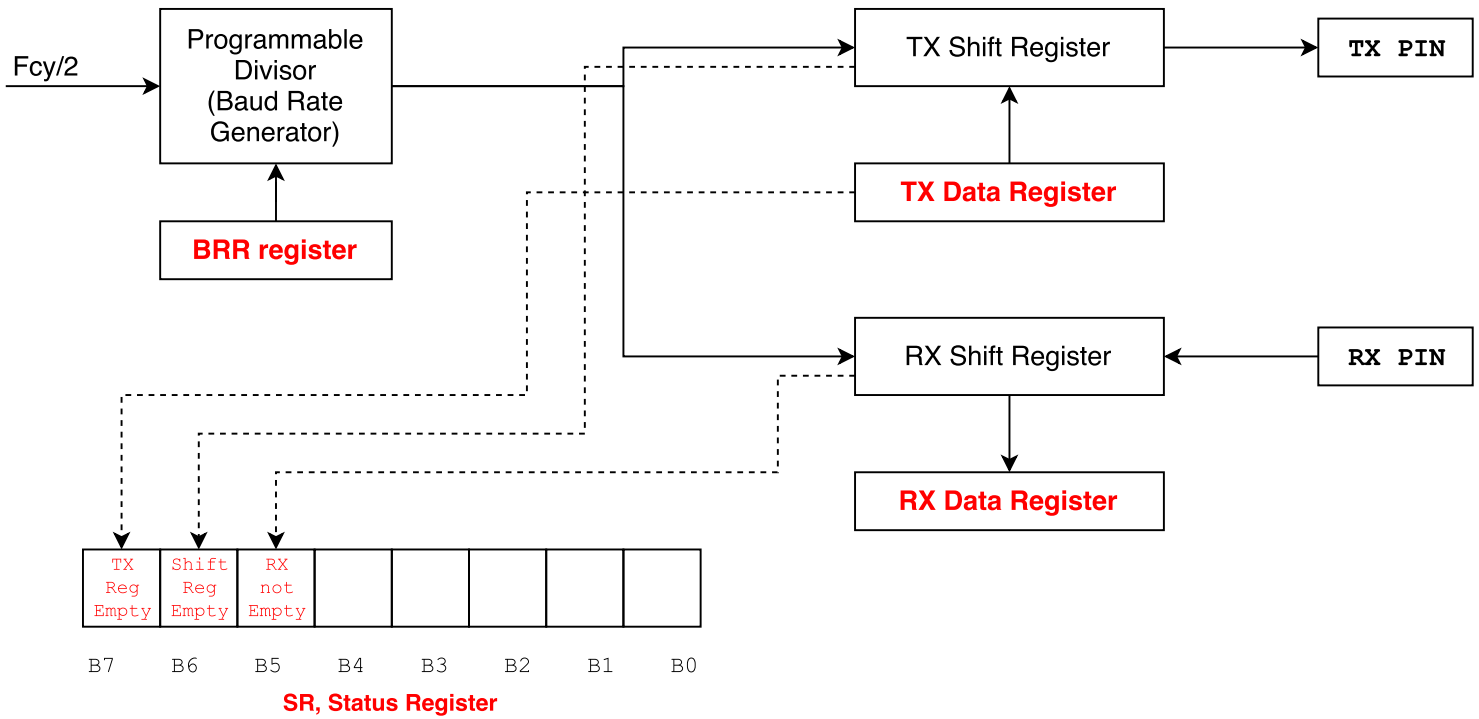
- The **Transmitter Circuit** is responsible of trasmitting data bits
- Data to be trasmitted is loaded (by the software) into the **TX Data Register**
- The value of **TX Data Register** is then copied o the **TX shift register** and the bit **``TX Register Empty''** is set in the **Status Register**

UART—Data Transmission (II)



- Data in the TX shift register is sent one bit at time
- As soon as all the 8 bits are transmitted, the bit **``TX Shift Register Empty''** is set in the **Status Register**

UART—Status Register



- Events occurring in the UART (data tramission, data reception, errors) are signaled by setting proper bits in the **Status Register**
- Each event can be also configured to generate a **IRQ**
- In this way, transmission and reception can be performed with interrupt-driven routine instead of using the classical polling

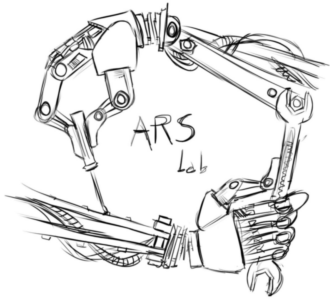
stm32_unict_lib Functions for UART

- Configure the USART2 and set baud rate to 115200 bps:
  ```
  void CONSOLE_init(void);
  ```
- Output messages to USART2:
  ```
  int printf(...);
  ```
- Check if a data byte has been received:
  ```
  int kbhit(void);
  ```
- Read/Wait for a new data byte:
  ```
  char readchar(void);
  ```

Using the UART with STM32 Microcontrollers

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



L.S.M. Course