

# REPORT

---

Title: RC Car Control with Bluetooth



Submission Date	2021.12.3	Major	Mechanical and control engineering
Subject	Embedded Controller	Professor	Young-Keun Kim
Name	Yeong-Won Song	Student Number	21700375
Partner Name	Jin-Su Yu	Partner Number	21700469

---

# Contents

<b>I. Introduction .....</b>	<b>3</b>
1.1 Purpose .....	3
1.2 Parts List.....	3
<b>II. Procedure .....</b>	<b>4</b>
2.1 Problem .....	4
2.2 Configuration .....	5
<b>III. Resources.....</b>	<b>9</b>
3.1 Bluetooth Module.....	9
3.2 RC Servo motor: RC servo motor (SG90) .....	9
3.3 DC Motor driver.....	10
<b>IV. Conclusion.....</b>	<b>12</b>
<b>Appendix</b>	

# I. Introduction

## 1.1 Purpose

Design a simple program to control an RC car steering and speed by sending the command message from PC via Bluetooth.

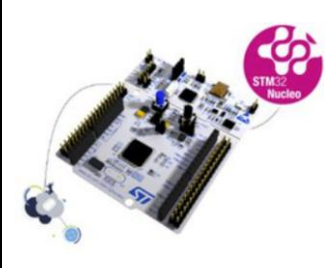



Motor 1: RC servo – for Steering (LEFT or RIGHT)

Motor 2: DC motor– for Speed and Heading Direction (FWD or BWD)

## 1.2 Parts List

Parts	Specification	Quantity
Servo motor	SG90	1
DC motor	L9110s	1
Bluetooth	HC-06	1
breadboard	-	1
NUCLEO-F411RE	Arm®(a) Cortex®-M4 with FPU	1

**Table 1. Part List**

			
NUCLEO-F411RE	RC Servo-Motor	Breadboard	DC-Motor Driver

**Table 2. experiment equipment**

## II. Procedure

### 2.1 Problem

Create a program to control two DC motors by giving a command from PC using Bluetooth module. The program should perform the following tasks by the user keyboard input. You must use appropriate interrupts

Print the status every 1 sec such as “(“RC car: DIR: 00[deg] VEL: 00[%] FWD”)

- Steering: RC Servo (Motor 1)
  - Divide 180° into 4 intervals.
  - Initially start the angle of RC servo motor at 90°.
  - Steering control with keyboard key. Increase or decrease the angle of RC servo motor by 45° each time you push the arrow key “RIGHT” or “LEFT”, respectively.
- Speed: DC motor (Motor 2)
  - Initially configure the duty ratio of DC motor at “0%”
  - Increase or decrease the speed by 25% duty each time you push the arrow key “UP” or “DOWN”, respectively.
  - The RC car driving direction should be forward or backward by pressing the key “F” or “B”, respectively.
  - The RC car must stop running when the key “S” is pressed.
  - Apply PWM to (A-IA) of motor driver
  - Apply Direction (H or L) to (A-IB) of motor driver

Key	Task	Comment
UP	Motor Speed Increases	Increase PWM duty ratio by 25% for each press. Initial value: duty 0% The maxim duty value should be 100%.
DOWN	Motor Speed decreases	Decrease PWM duty ratio by 25% for each press The minimum duty value should be 0%.
LEFT	Left turn	Turn Left by 45degree
RIGHT	Right turn	Turn Right by 45degree
F	Forward direction	Go forward. Default setting DIR=1
B	Backward direction	Reverse driving. DIR=0
S	Stop	Make duty=0% for both motors

## 2.2 Configuration

You are free to select appropriate configurations for the design problem. Fill in the table.

Functions	Register	PORT_PIN	Configuration
System Clock	RCC		PLL 84MHz
delay_ms	SysTick		1KHz
Motor DIR	Digital Out	PA5	Output Medium speed Push-Pull No Pull-up Pull down
TIMER	TIMER2		TIM2: Counter Period 50Hz
	TIMER3		TIM3: Counter Period 500Hz
RC servo angle	PWM1	PA1_Ch2(TIM2)	PWM period: 20ms PWM duty ratio: 0.5ms to 2.5ms
DC Motor Speed	PWM2	PA6_Ch1(TIM3)	PWM period: 2ms PWM duty ratio: 0ms to 2ms
RS-232 USB cable (ST-LINK)	USART2	TXD: PA2 RXD: PA3	No Parity, 8-bit Data, 1-bit Stop bit 38400 baud-rate
Bluetooth	USART1	TXD: PA9 RXD: PA10	No Parity, 8-bit Data, 1-bit Stop bit 9600 baud-rate

- Flow Chart

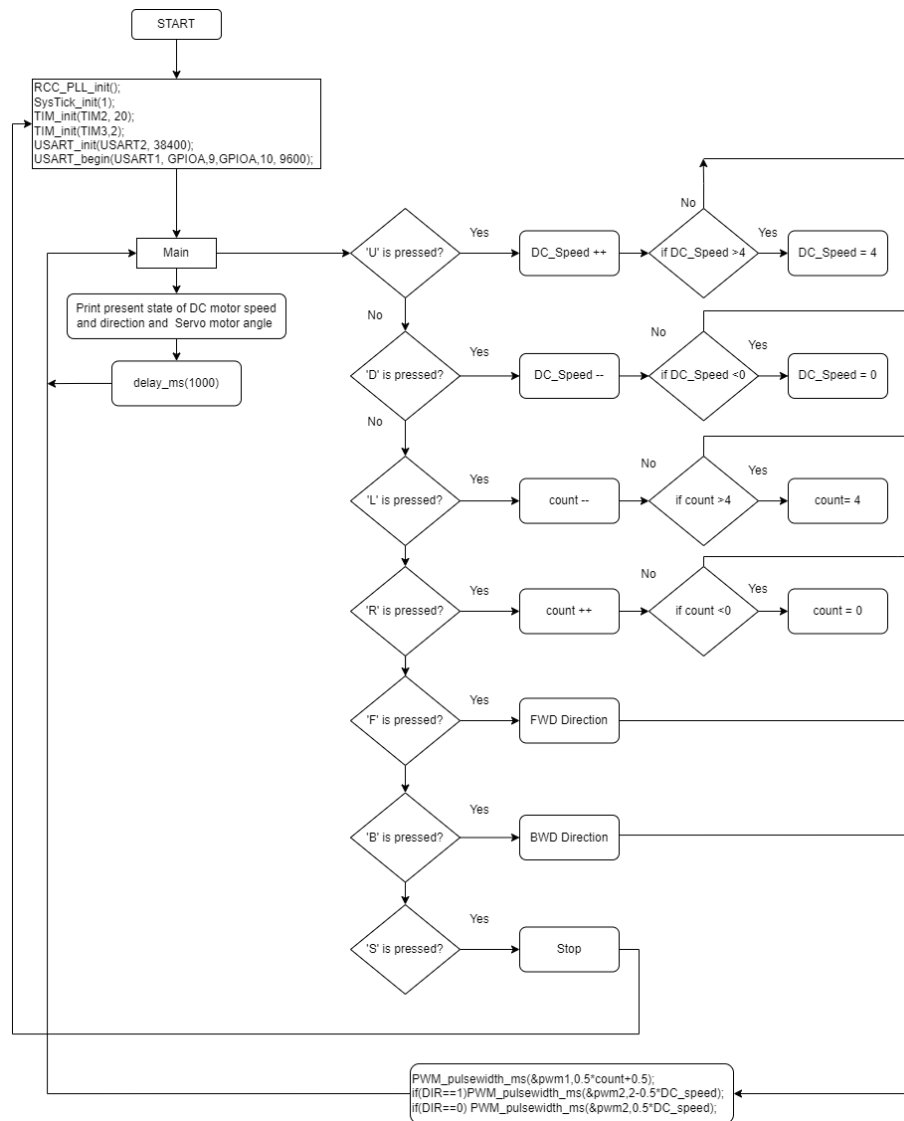


Figure 1. Flow chart

- External Circuit

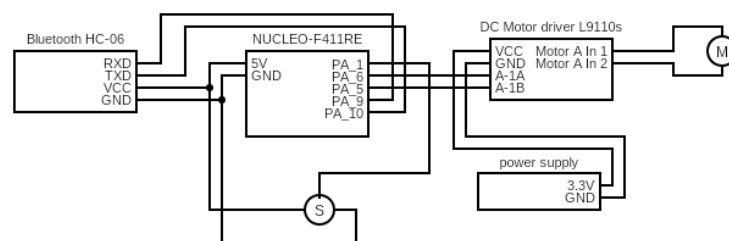


Figure 2. External circuit

## - Main source code

```

/**
*****
* @author  SONG YEONG WON
* @Mod     2021-11-24 YWSONG
* @brief   Embedded Controller:  USART communication
*
*****
*/

#include "ecHAL.h"

// USART2 : MCU to PC via usb
// USART1 : MCU to MCU2

uint8_t mcu2Data = 0;
uint8_t pcData = 0;
int idx = 0;
int maxBuf=10;
uint8_t buffer[100]={0,};
int endChar = 13;

PWM_t pwm1;
PWM_t pwm2;
//Servo angle
static int count= 2;

//DC speed
static int DC_speed= 0;
static int DIR = 0;

static int Angle[5] = {0,45,90,135,180}; //Angle state
static int VEL[5] = {0,25,50,75,100}; //Dc speed state
static char *Direction[2] = {"FWD","BWD"}; //Direction state Default FWD

static uint8_t endBuf[2] = {'\r\n', '\r\n'};

void setup(void);
void USART1_IRQHandler(void);
void USART2_IRQHandler(void);
void State_input(void);

int main(void) {
// Initialization -----
setup();
printf("Hello RC car\r\n");
// Infinite Loop -----
while(1){
printf("RC car -> DIR : %d[deg] VEL : %d[per] %s \r\n", Angle[count],VEL[DC_speed],Direction[DIR]);
delay_ms(1000);
}

// Initialization
void setup(void){
RCC_PLL_init();
SysTick_init(1);
PWM_pulsewidth_ms(&pwm1,1.5); //Initiall Servo motor angle 90
//Motor direction Pin
GPIO_init(GPIOA,5,OUTPUT);
GPIO_ospeed(GPIOA, 5,MEDIUM_SPEED);
GPIO_otype(GPIOA, 5, OUTPUT_PP);
GPIO_pupdr(GPIOA, 5, NOPUPD);

//TIM 2 -> RC Servo angle
TIM_init(TIM2, 20);
PWM_init(&pwm1,GPIOA,1);
PWM_period_ms(&pwm1,20);

//TIM 3 -> DC Motor Speed
TIM_init(TIM3,2);
PWM_init(&pwm2,GPIOA,6);
PWM_period_ms(&pwm2,2);

// USART configuration
USART_init(USART2, 38400);
USART_begin(USART1, GPIOA,9,GPIOA,10, 9600); // PA9 - RXD , PA10 - TXD
}

```

```

void USART1_IRQHandler(){ //USART1 INT
    if(is_USART_RXNE(USART1)){
        mcu2Data = USART_getc(USART1);
        USART_write(USART1,&mcu2Data,1);
        if(mcu2Data==endChar){
            //bReceive=1;
            indx = 0;
            USART_write(USART1, &endBuf, 1);
            State_input();
        }
        else{
            if(indx>maxBuf){
                indx =0;
                memset(buffer, 0, sizeof(char) * maxBuf);
                printf("ERROR : Too long string\r\n");
            }
            buffer[indx] = mcu2Data;
            indx++;
        }
    }
}

void USART2_IRQHandler(){ //USART2 INT
    if(is_USART_RXNE(USART2)){
        pcData = USART_getc(USART2);
        USART_write(USART1,&pcData,1); // transmit char to USART1
        printf("%c",pcData);          // echo to sender(pc)

        if(pcData==endChar){
            printf("\r\n");           // to change line on PC display
        }
    }
}

void State_input(void){
    //Forward, Backward, Stop
    printf("buffer : %s\r\n",buffer);
    switch(buffer[0]){
        case 'U' : DC_speed++; if(DC_speed>4)DC_speed=4; break; //UP
        case 'D' : DC_speed--; if(DC_speed<0)DC_speed=0; break; //DOWN
        case 'L' : count--; if(count<0)count=0; break; //LEFT
        case 'R' : count++; if(count>4)count=4; break; //RIGHT
        case 'F' : GPIO_write(GPIOA,5,LOW); DIR=0; break; //FORWARD
        case 'B' : GPIO_write(GPIOA,5,HIGH); DIR=1; break; //BACKWARD
        case 'S' : count=2; DC_speed=0; break; //STOP
        default: printf("ERROR : TYPE U,D,L,R,F,B,S ONLY!!\r\n"); break;
    }
    buffer[0]=NULL;

    PWM_pulsewidth_ms(&pwm1,0.5*count+0.5); //Servo Motor angle (LEFT/RIGHT)
    if(DIR==1){                               //DC Motor control
        PWM_pulsewidth_ms(&pwm2,2+0.5*DC_speed);
    }else PWM_pulsewidth_ms(&pwm2,0.5*DC_speed);
}

```

Figure 3. Main code



## II. Resources

### 3.1 Bluetooth Module

Search for the Bluetooth module specification sheet (HC-06) and study the pin configurations. The default PIN number is 1234.

Example of connecting to USART1

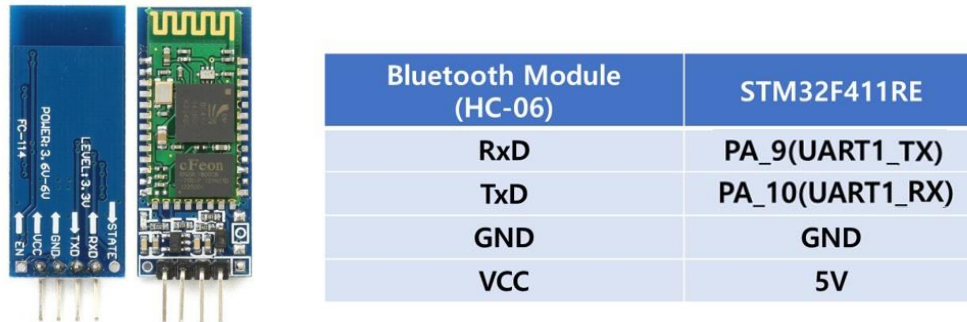


Figure 4. Connection of a bluetooth module

### 3.2 RC Servo Motor: RC Servo Motor (SG90)

An RC servo motor is a tiny and light weight motor with high output power. It is used to control rotation angles, approximately 180 degrees (90 degrees in each direction) and commonly applied in RC car, and Small-scaled robots.

The angle of the motor can be controlled by the pulse width (duty ratio) of PWM signal. The PWM period should be set at **20ms or 50Hz**. Refer to the data sheet of the RC servo motor for detailed specifications.

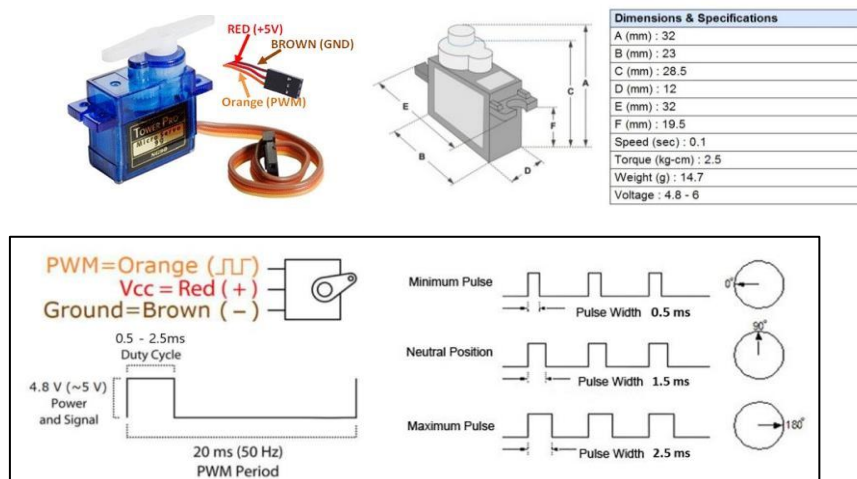


Figure 5. Operation of Servo Motor

### 3.3 DC Motor driver

Connect DC motor drive(L9110s) module pins to MCU as shown below.

DO NOT use MCU's VCC to motor driver. You should use external voltage source.

- IA: PWM pin
- IB: Direction Pin (Digital Out H or L)

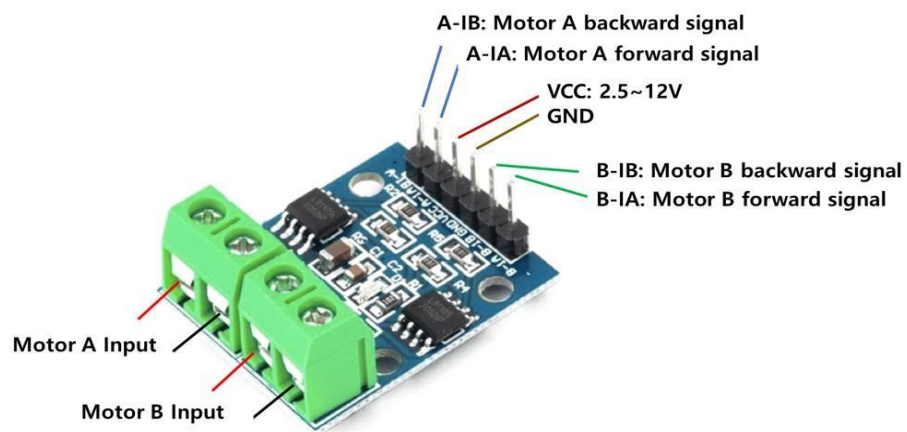


Figure 6. DC Motor driver

## - Experimental Result

```

RC car -> DIR : 90[deg] VEL : 0[per] FWD
RC car -> DIR : 90[deg] VEL : 0[per] FWD
RC car -> DIR : 90[deg] VEL : 0[per] FWD
RC car -> DIR : 90[deg] VEL : 0[per] FWD
buffer : U
RC car -> DIR : 90[deg] VEL : 25[per] FWD
RC car -> DIR : 90[deg] VEL : 25[per] FWD
buffer : U
RC car -> DIR : 90[deg] VEL : 50[per] FWD
RC car -> DIR : 90[deg] VEL : 50[per] FWD
buffer : U
RC car -> DIR : 90[deg] VEL : 75[per] FWD
buffer : U
RC car -> DIR : 90[deg] VEL : 100[per] FWD
RC car -> DIR : 90[deg] VEL : 100[per] FWD
buffer : D
RC car -> DIR : 90[deg] VEL : 75[per] FWD
buffer : D
RC car -> DIR : 90[deg] VEL : 50[per] FWD
RC car -> DIR : 90[deg] VEL : 50[per] FWD
buffer : B
RC car -> DIR : 90[deg] VEL : 50[per] BWD
RC car -> DIR : 90[deg] VEL : 50[per] BWD
RC car -> DIR : 90[deg] VEL : 50[per] BWD
buffer : L
RC car -> DIR : 45[deg] VEL : 50[per] BWD
buffer : L
RC car -> DIR : 0[deg] VEL : 50[per] BWD
buffer : R
RC car -> DIR : 45[deg] VEL : 50[per] BWD
buffer : R
RC car -> DIR : 90[deg] VEL : 50[per] BWD
buffer : R
RC car -> DIR : 135[deg] VEL : 50[per] BWD
buffer : R
RC car -> DIR : 180[deg] VEL : 50[per] BWD
buffer : L
RC car -> DIR : 135[deg] VEL : 50[per] BWD
RC car -> DIR : 135[deg] VEL : 50[per] BWD
buffer : S
RC car -> DIR : 90[deg] VEL : 0[per] BWD
RC car -> DIR : 90[deg] VEL : 0[per] BWD
RC car -> DIR : 90[deg] VEL : 0[per] BWD
buffer : F
RC car -> DIR : 90[deg] VEL : 0[per] FWD
RC car -> DIR : 90[deg] VEL : 0[per] FWD
buffer : U
RC car -> DIR : 90[deg] VEL : 25[per] FWD

```

**Figure 7. experimental result in tera-term.**

The speed of DC-Motor increases by 25%. When the 'U' button is pressed, DC-Motor velocity increased, the 'D' button is pressed, it operated slowly. When the 'L' button is pressed, the Servo-motor rotates 45 degrees to the left, and when the 'R' button is pressed, it rotates 45 degrees to the right. The 'F' button returns to the forward direction of DC-Motor and the 'B' button returns to the backward direction of DC-Motor. The 'S' button allows both Servo-motor and DC-motor to return to their initial state, and the direction remains the same before initialization. In figure 7, it can be seen that the DC-motor and Servo-motor operate independently according to the user button input. In addition, when the 'S' button was pressed, it became an initial state, and it was confirmed that it operated normally according to the user button input again.

### III. Conclusion

Through this experiment, I tried to control RC Car through Bluetooth. Various registers such as USART, timer, interrupt, and SysTick were used to control according to the purpose. Various functions were programmed so that they could be executed independently and well, and easily controlled according to the user's convenience. Furthermore, I think it will be easier and simpler to use if all of this is made into one API. In addition, for efficient program execution, the code was simplified, and each responded quickly according to the user's input.

#### - Troubleshooting

**Q.** How did the code make it as efficient as possible?

**A.** At first, it was very inefficient to define the operating state for each state using the switch statement. So, without using the switch statement, the formula was generalized and changed to one line. The states of DC-motor and Servo-motor were changed through the input variables.

# Appendix

- **Video demo Link**

Click [here](#) watch video.

- **Documentation**

---

## USART

### Header File

```
#include "ecUSART.h"

#ifndef __EC_USART_H
#define __EC_USART_H

#include "stm32f411xe.h"
#include <stdio.h>
#include "ecGPIO.h"
#include "ecRCC.h"

#define POL 0
#define INT 1

// You can modify this
#define BAUD_9600 0
#define BAUD_19200 1
#define BAUD_57600 2
#define BAUD_115200 3
#define BAUD_921600 4

// ***** USART 2 (USB) *****
// PA_2 = USART2_TX
// PA_3 = USART2_RX
// Alternate function(AF7), High Speed, Push pull, Pull up
// APB1
// *****

// ***** USART 1 *****
// PB_6 = USART1_TX (default) // PA_9 (option)
// PB_3 = USART1_RX (default) // PA_10 (option)
// APB2
// *****

// ***** USART 6 *****
// PA_11 = USART6_TX (default) // PC_6 (option)
// PA_12 = USART6_RX (default) // PC_7 (option)
// APB2
// *****

/* Given to Students */
void USART_write(USART_TypeDef* USARTx, uint8_t* buffer, uint32_t nBytes);
void USART_delay(uint32_t us);

/* Exercise*/
void USART_begin(USART_TypeDef* USARTx, GPIO_TypeDef* GPIO_TX, int pinTX, GPIO_TypeDef* GPIO_RX, int pinRX,
int baud);
void USART_init(USART_TypeDef* USARTx, int baud);
// default pins.
uint8_t USART_getc(USART_TypeDef * USARTx);
uint32_t is_USART_RXNE(USART_TypeDef * USARTx);

#endif
```

---

## USART\_write()

Transmit the data. User can send the input data between different usart communications.

```
void USART_write(USART_TypeDef* USARTx, uint8_t* buffer, uint32_t nBytes);
```

### Parameters

- **USARTx:** select USART nubmer
- **buffer:** user input buffer
- **nBytes:** TXE (TX empty) bit

### Example code

```
void USART2_IRQHandler(){ //USART2 INT
    if(is_USART_RXNE(USART2)){
        pcData = USART_getc(USART2);
        USART_write(USART1,&pcData,1); // transmit char to USART1
        printf("%c",pcData);           // echo to sender(pc)
        if(pcData==endChar){
            printf("\r\n");             // to change line on PC display
        }
    }
}
```

## USART\_begin()

Select the USART number and decide the corresponding GPIO port and pinnumber of RX and TX Pin. The baud-rate of usart communication is determined.

```
void USART_begin(USART_TypeDef* USARTx, GPIO_TypeDef* GPIO_TX, int pinTX, GPIO_TypeDef* GPIO_RX, int pinRX,
int baud);
```

### Parameters

- **USARTx:** select USART nubmer
- **GPIO\_TX:** Select a designated TX port according to USART.
- **pinTX:** Select a designated TX pin number according to USART and GPIO\_TX port.
- **GPIO\_RX:** Select a designated RX port according to USART.
- **pinRX:** Select a designated RX pin number according to USART and GPIO\_RX port.
- **baud:** Select USART baud-rate

### Example code

```
USART_begin(USART1, GPIOA,9,GPIOA,10, 9600); // PA9 - RXD , PA10 - TXD
```

## USART\_init()

Initializes GPIO port and pin number with default setting in USART communication.

```
void USART_init(USART_TypeDef* USARTx, int baud);
```

### Parameters

- **USARTx:** select USART nubmer
- **baud:** Select USART baud-rate

### Example code

```
USART_init(USART2, 38400);
```

---

### USART\_getc()

Read USART communication data.

```
uint8_t USART_getc(USART_TypeDef * USARTX);
```

#### Parameters

- **USARTx**: Select the USART number to read the data.

#### Example code

```
mcu2Data = USART_getc(USART1);  
pcData = USART_getc(USART2);
```

### is\_USART\_RXNE()

Check the status of USART. Received data is ready to be read or Data is not received.

```
uint32_t is_USART_RXNE(USART_TypeDef * USARTX);
```

#### Parameters

- **USARTx**: Select the USART number

#### Example code

```
void USART1_IRQHandler(){           //USART1 INT  
    if(is_USART_RXNE(USART1)){  
        mcu2Data = USART_getc(USART1);  
        USART_write(USART1,&mcu2Data,1);  
        if(mcu2Data==endChar){  
            //bReceive=1;  
            indx = 0;  
            USART_write(USART1, &endBuf, 1);  
            State_input();  
        }  
        else{  
            if(indx>maxBuf){  
                indx =0;  
                memset(buffer, 0, sizeof(char) * maxBuf);  
                printf("ERROR : Too long string\r\n");  
            }  
            buffer[indx] = mcu2Data;  
            indx++;  
        }  
    }  
}
```

---

**Figure 8. Documentation of USART**