VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
Faculty of Computer Science and Engineering

MICROCONTROLLER - MICROPROCESSOR

LAB 5

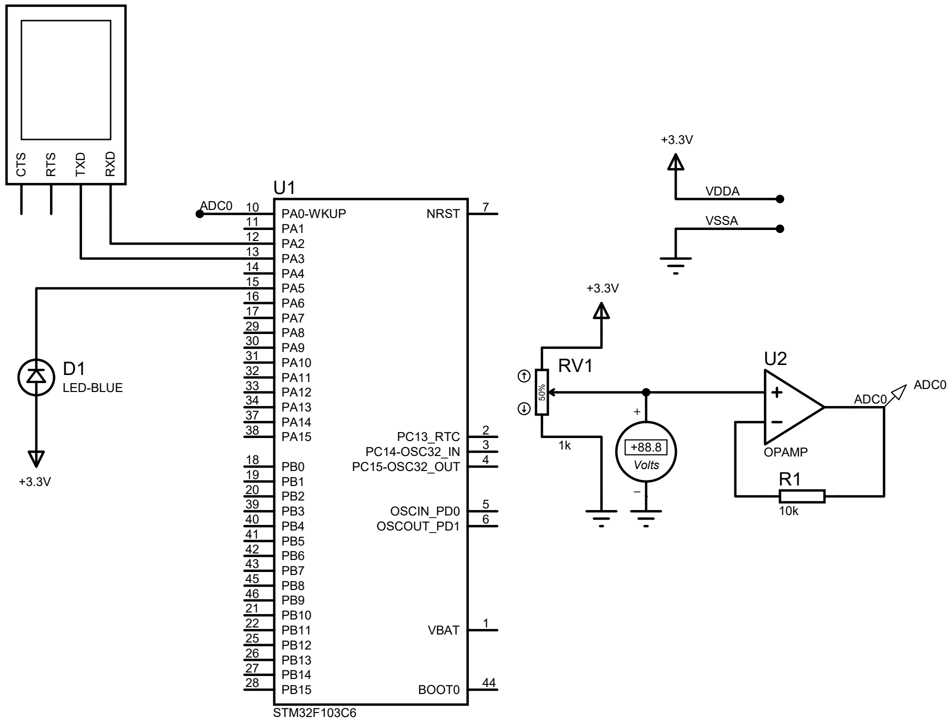
UART

INSTRUCTOR:	Huỳnh Phúc Nghị
STUDENT:	Tạ Gia Bảo - 2110795
GitHub submission:	`github.com/zabao-qt/GIT_LAB5`

Contents

1 Schematic

1.1 Proteus UART



2 Source code

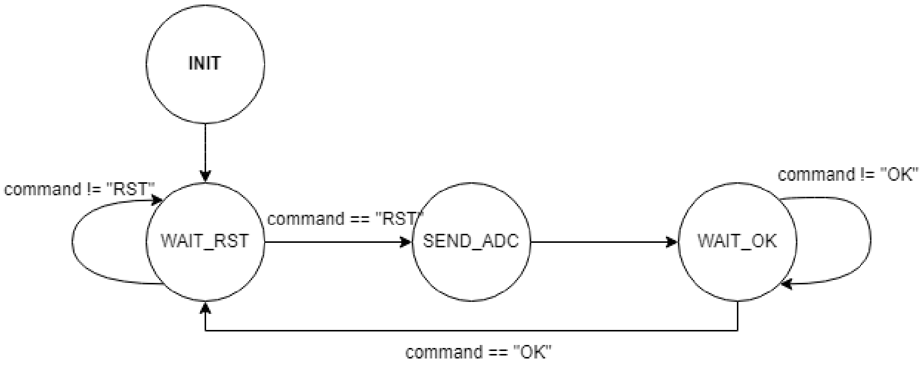
2.1 global.h

Listing all the global variables.

```
#ifndef INC_GLOBAL_H_
#define INC_GLOBAL_H_

#include "main.h"

// Status command parser
#define INIT_STR     0
#define WAIT_END     1

// Status UART communication
#define WAIT_RST     10
#define SEND_ADC     11
#define WAIT_OK      12
#define MAX_BUFFER_SIZE 30

extern int status_parser;
extern int status_uart;

// Variables to read data
extern uint8_t temp;
extern uint8_t buffer[MAX_BUFFER_SIZE];
extern uint8_t index_buffer;
extern uint8_t buffer_flag;

// Variables to read command
extern uint8_t command_flag;
extern uint8_t command[MAX_BUFFER_SIZE];
extern uint8_t command_index;

// String to display console
extern char str[50];
// ADC Value
extern uint32_t ADC_value;

#endif /* INC_GLOBAL_H_ */
```

2.2 FSM implementation

2.2.1 uart_communiation_fsm() design



2.2.2 uart_communiation_fsm()

This FSM has 3 states: WAIT_RST, SEND_ADC, and WAIT_OK. We begin at the initial state WAIT_RST, checks command "RST". If true, update the ADC value, send to UART, set timer for 3s and move to SEND_ADC state.

```
case WAIT_RST:
    // If command has completed and command = "RST" -> status = SEND_ADC, update
    ADC_Value, flag = 0 and setTimer
    if (command_flag == 1) {
        command_flag = 0;
        if (command[0] == 'R' && command[1] == 'S' && command[2] == 'T') {
            // Get ADC value
            HAL_ADC_Start(&hadc1);
            ADC_value = HAL_ADC_GetValue(&hadc1);
            HAL_ADC_Stop(&hadc1);
            HAL_UART_Transmit(&huart2, (void*)str, sprintf(str, "\r\n"), 1000);
            status_uart = SEND_ADC;
            setTimer(1, 3000);
        }
    }
    break;
```

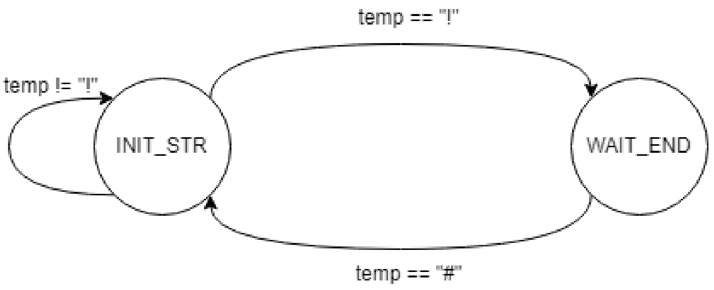
Then, the state transmit ADC value to UART.

```
case SEND_ADC:
    // Display ADC Value console, status = WAIT_OK
    HAL_UART_Transmit(&huart2, (void*)str, sprintf(str, "!ADC=%lu#\r\n", ADC_value
    ), 1000);
    status_uart = WAIT_OK;
    break;
```

Next, it checks if a command has been completed and if the command is "OK". If true, it sends a newline character over UART, transitions to the WAIT_RST state, and clears the timer. If false, it checks if a timer flag is set. If true, it transitions to the SEND_ADC state and sets the timer for another 3s.

```
case WAIT_OK:
    // If command has completed and command = "OK" -> status = WAIT_RST and
    clearTimer
    if (command_flag == 1) {
        command_flag = 0;
        if (command[0] == 'O' && command[1] == 'K') {
            HAL_UART_Transmit(&huart2, (void*)str, sprintf(str, "\r\n"), 1000);
            status_uart = WAIT_RST;
            clearTimer(1);
        }
    }
    // Else, if each after 3s the system doesn't receive string "OK" -> status =
    SEND_ADC
    if(timer_flag[1] == 1) {
        status_uart = SEND_ADC;
        setTimer(1, 3000);
    }
    break;
```

2.2.3 command_parser_fsm() design



2.2.4 command_parser_fsm()

This FSM has 2 states: INIT_STR and WAIT_END. It processes characters received over UART and assembles them into a command until it detects the end marker "#". It begins with INIT_STR as the initial state. Then checks if temp is "#" to start new command. If true, switch to WAIT_END state and reset.

```
case INIT_STR:
    if(temp == '!') {
        status_parser = WAIT_END;
        command_index = 0;
    }
    break;
```

WAIT_END state processes characters until the end marker "#" is received. If "#" is detected, switch back to initial state, else check if there is "!" to start new command.

```
case WAIT_END:
    if(temp == '#') {
        status_parser = INIT_STR;
        command[command_index] = '\0';
        command_flag = 1;
    }
    else {
        if (temp == '!')
            command_index = 0;
        else {
            command[command_index++] = temp;
            if (command_index == MAX_BUFFER_SIZE) command_index = 0;
        }
    }
    break;
```

2.3 main(void)

In the main infinite loop, we toggle LED Blinky and call two FSMs.

```
int main(void)
{
    // Other function calls
    while (1)
    {
        if (timer_flag[0] == 1) {
              HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
              setTimer(0, 1000);
        }
        // If received byte, call command_parser_fsm function
        if (buffer_flag == 1) {
          command_parser_fsm();
          buffer_flag = 0;
        }
        // uart_communiation_fsm function
        uart_communiation_fsm(hadc1, huart2);
    }
}
```