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

CC02 — Lab Report

Microprocessor - Microcontroller
Lab 5

Supervisors: Nguyen Thien An
Students: Nguyen Thanh Tai 2252722

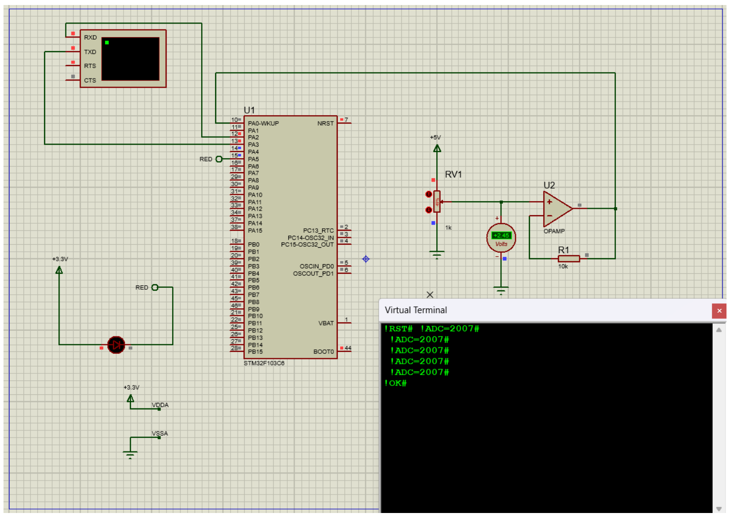
Ho Chi Minh City, November 26, 2024

Contents

1 Schematic

The GitHub link for the lab schematics is at here or in this link: https://github.com/ThanhTaiNguyen24/mcu-mpu-lab1.

The schematic for the lab work is located here:



```
!RST# !ADC=2007#
 !ADC=2007#
 !ADC=2007#
 !ADC=2007#
 !ADC=2007#
!OK#
```

Figure 1: *The schematic for Lab 5.*

2 Problem

This is the source code for the lab work problem:

commandparser.c

```c
#include "commandparser.h"

int state = 0 ;
int check = 0;
void command_parser_fsm() {
      if (strstr((const char *)buffer, "!RST#") != NULL) {
        state = INFO;
      }
      else if (strstr((const char *)buffer, "!OK#") != NULL) {
        state = INIT;
        check = 0;
      }
}

void uart_communiation_fsm (){
  switch(state){
  case INIT:
    break;
  case INFO:
    if (timer_flag0 == 1){
    if (check == 0){
      HAL_ADC_Start(&hadc1);
      HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
      HAL_ADC_Stop(&hadc1);
      check = 1;
    }
    ADC_value = HAL_ADC_GetValue(&hadc1);

    int len = sprintf(str," !ADC=%lu# \r\n",ADC_value);
    HAL_UART_Transmit (& huart2 , ( void *) str , len, 1000);

         memset(buffer, 0, sizeof(buffer));
         index_buffer = 0;
         setTimer0(1000);
    }
    break;
  default:
    break;
  }
}
```

main.c

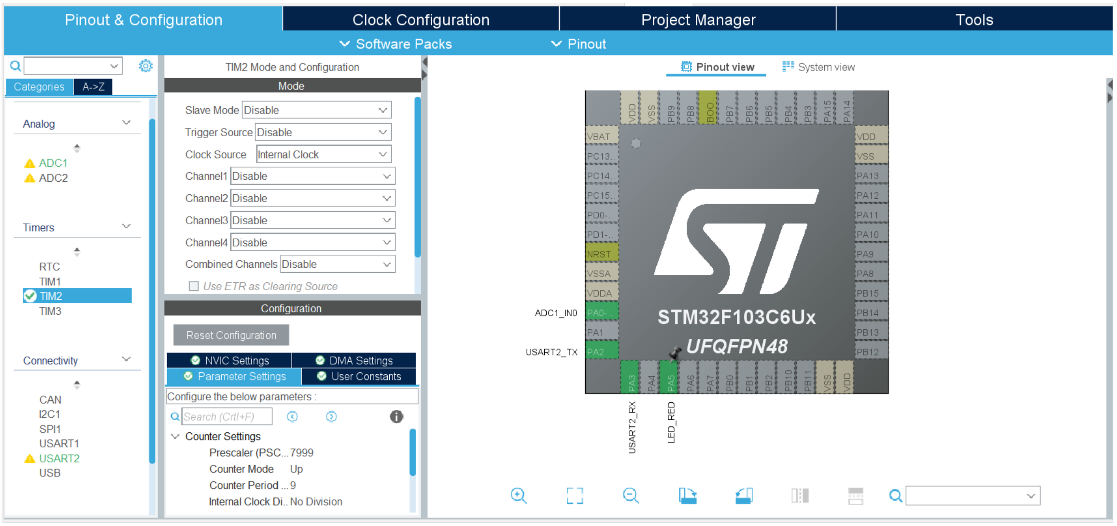
```c
uint32_t ADC_value = 0;
char str[32];
uint8_t temp = 0;
uint8_t buffer[MAX_BUFFER_SIZE];
uint8_t index_buffer = 0;
uint8_t buffer_flag = 0;
#define DELETE_CHAR 0x08

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
    if(huart->Instance == USART2){
      if (temp==DELETE_CHAR){
      if (index_buffer>0){
        index_buffer--;
      }

      HAL_UART_Transmit(&huart2, (uint8_t *)"\b \b", 3, 50); // for the
    termination of the backspace
      }
      else {
        if (temp=='!'){
          index_buffer = 0;
        }
      HAL_UART_Transmit(&huart2, &temp, 1, 50);
      buffer[index_buffer++] = temp;
      if(index_buffer == MAX_BUFFER_SIZE) index_buffer = 0;
      }
      buffer_flag = 1;
      HAL_UART_Receive_IT(huart, &temp, 1);
    }
}

setTimer1(100);
setTimer0(100);

while (1)
{
    if (buffer_flag == 1) {
    command_parser_fsm();
    buffer_flag = 0;
    }
    uart_communiation_fsm();
}
```

timer.c

```c
int timer_flag0;
int timer_flag1;

int timer_counter0;
int timer_counter1;

int TIMER_CYCLE = 10;

void setTimer0(int duration) {
  timer_counter0 = duration / TIMER_CYCLE;
  timer_flag0 = 0;
}


void setTimer1(int duration) {
  timer_counter1 = duration / TIMER_CYCLE;
  timer_flag1 = 0;
}

void timer_run() {
  if (timer_counter0 > 0) {
    timer_counter0--;
    if (timer_counter0 == 0) {
      timer_flag0 = 1;
    }
  }
  if (timer_counter1 > 0) {
    timer_counter1--;
    if (timer_counter1 == 0) {
      timer_flag1 = 1;
    }
  }
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    timer_run();
}
```

3 Explaination

A project that has pin corresponding to the Proteus schematic with 10ms timer interrupt, UART and ADC.



TIM2 Mode and Configuration

Mode

Slave Mode Disable
Trigger Source Disable
Clock Source Internal Clock
Channel1 Disable
Channel2 Disable
Channel3 Disable
Channel4 Disable
Combined Channels Disable
Use ETR as Clearing Source

Configuration

Reset Configuration
NVIC Settings, DMA Settings, Parameter Settings, User Constants

Configure the below parameters :

Counter Settings
Prescaler (PSC... 7999
Counter Mode Up
Counter Period ... 9
Internal Clock Di.. No Division

Figure 2: *Lab 5 project.*

The **HAL_UART_RxCpltCallback** function is a crucial part of the UART communication process in embedded systems. This callback function is triggered whenever a UART receive complete interrupt occurs, specifically for the USART2 instance. It handles backspace operation and for other characters, it resets the index_buffer if the character is '!', transmits the received character, stores it in the buffer, and increments the index_buffer. If the buffer reaches its maximum size (MAX_BUFFER_SIZE), it resets to avoid overflow. The function sets a buffer_flag to indicate new data has been received and re-enables the UART receive interrupt for the next character.

```c
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
    if(huart->Instance == USART2){
        if (temp==DELETE_CHAR){
    if (index_buffer>0){
      index_buffer--;
    }
    HAL_UART_Transmit(&huart2, (uint8_t *)"\b \b", 3, 50); // for the termination
    of the backspace
        }
        else {
      if (temp=='!'){
        index_buffer = 0;
      }
    HAL_UART_Transmit(&huart2, &temp, 1, 50);
    buffer[index_buffer++] = temp;
    if(index_buffer == MAX_BUFFER_SIZE)                 index_buffer = 0;
```

```
        }
  buffer_flag = 1;
  HAL_UART_Receive_IT(huart, &temp, 1);
    }
}
```

Within the infinite loop, the system checks if new data has been received. If so, it calls the finite state machine of command parser to process the received commands and then resets the buffer_flag to 0. Additionally, the finite state machine of uart communication is called continuously to manage the UART communication state machine, ensuring efficient data transmission and reception.

```
while (1)
{
    if (buffer_flag == 1) {
    command_parser_fsm();
    buffer_flag = 0;
    }
    uart_communiation_fsm();
}
```

The finite state machine of command parser parses commands received via UART, checking for specific strings like **!RST** and **!OK** to set the system state accordingly. If **!RST** is detected, the state is set to INFO, and if **!OK** is found, the state is set to INIT.

The finite state machine of uart communication handles the UART communication based on the current state. In the INFO state, it reads an ADC value (which is kept as the previous packet through check variable), formats it into a string, and transmits it via UART. It also manages a timer to periodically reset the buffer and index.

```
int state = 0 ;
int check = 0;
void command_parser_fsm() {
    if (strstr((const char *)buffer, "!RST#") != NULL) {
  state = INFO;
    }
    else if (strstr((const char *)buffer, "!OK#") != NULL) {
        state = INIT;
  check = 0;
    }
}

void uart_communiation_fsm (){
    switch(state){
    case INIT:
  break;
    case INFO:
  if (timer_flag0 == 1){
      if (check == 0){
          HAL_ADC_Start(&hadc1);
      HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
      HAL_ADC_Stop(&hadc1);
      check = 1;
    }
    ADC_value = HAL_ADC_GetValue(&hadc1);
    int len = sprintf(str," !ADC=%lu# \r\n",ADC_value);
    HAL_UART_Transmit (& huart2 , ( void *) str , len, 1000);

            memset(buffer, 0, sizeof(buffer));
            index_buffer = 0;
            setTimer0(1000);
  }
  break;
    default:
  break;
    }
}
```

References