МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

“Проектирование вычислительных систем”

Вариант №5

Студент:

Чернова Анна Ивановна

Миху Вадим Дмитриевич

Группа P34301

Преподаватель:

Пинкевич Василий Юрьевич

г. Санкт-Петербург

2024

Цель работы

1. Изучить протокол передачи данных по интерфейсу UART

2. Получить базовые знания об организации системы прерываний в микроконтроллерах на примере микроконтроллера STM32

3. Изучить устройство работы и принципы контроллера интерфейса UART, получить навыки организации обмена данными по UART в режимах опроса и прерываний

Задание

Разработать и реализовать два варианта драйверов UART для стенда SDK-1.1M: с использованием и без использования прерываний. Драйверы, использующие прерывания, должны обеспечивать работу в «неблокирующем» режиме (возврат из функции происходит сразу же, без ожидания окончания приема/отправки), а также буферизацию данных для исключения случайной потери данных. В драйвере, не использующем прерывания, функция приема данных также должна быть «неблокирующей», то есть она не должна зависать до приема данных (которые могут никогда не поступить). При использовании режима «без прерываний» прерывания от соответствующего блока UART должны быть запрещены. Написать с использованием разработанных драйверов программу, которая выполняет определенную вариантом задачу. Для всех вариантов должно быть реализовано два режима работы программы: с использованием и без использования прерываний. Каждый принимаемый стендом символ должен отсылаться обратно, чтобы он был выведен в консоли (так называемое «эхо»). Каждое новое сообщение от стенда должно выводиться с новой строки. Если вариант предусматривает работу с командами, то на каждую команду должен выводиться ответ, определенный в задании или «ОК», если ответ не требуется. Если введена команда, которая не поддерживается, должно быть выведено сообщение об этом. Скорость работы интерфейса UART должна соответствовать указанной в варианте задания.

Вариант 5

Доработать программу кодового замка. Теперь ввод кода должен происходить не с помощью кнопки стенда, а по UART. После ввода единственно верной последовательности из восьми латинских букв без учёта регистра и цифр должен загореться зелёный светодиод, обозначающий «открытие» замка. Светодиод горит некоторое время, потом гаснет, и система вновь переходит в «режим ввода». Каждый неправильно введённый элемент последовательности должен сопровождаться миганием красного светодиода и сбросом в «начало», каждый правильный – миганием жёлтого. После трёх неправильных вводов начинает мигать красный светодиод, и через некоторое время система вновь возвращается в «режим ввода». Если код не введен до конца за некоторое ограниченное время, происходит сброс в «начало». Должно быть предусмотрено изменение отпирающей последовательности, что производится следующей последовательностью действий:

– ввод символа «+»;

– ввод новой последовательности, который завершается либо по нажатию enter, либо по достижении восьми значений;

– стенд отправляет сообщение произвольного содержания, спрашивая, сделать ли последовательность активной, и запрашивает подтверждение, которое должно быть сделано вводом символа y;

– после ввода y введённая последовательность устанавливается как активная.

Включение/отключение прерываний должно осуществляться нажатием кнопки на стенде и сопровождаться отправкой в последовательный порт сообщения произвольного содержания, сообщающего, какой режим включен (с прерываниями или без прерываний).

Скорость обмена данными по UART – 9600 бит/с.

Выполнение

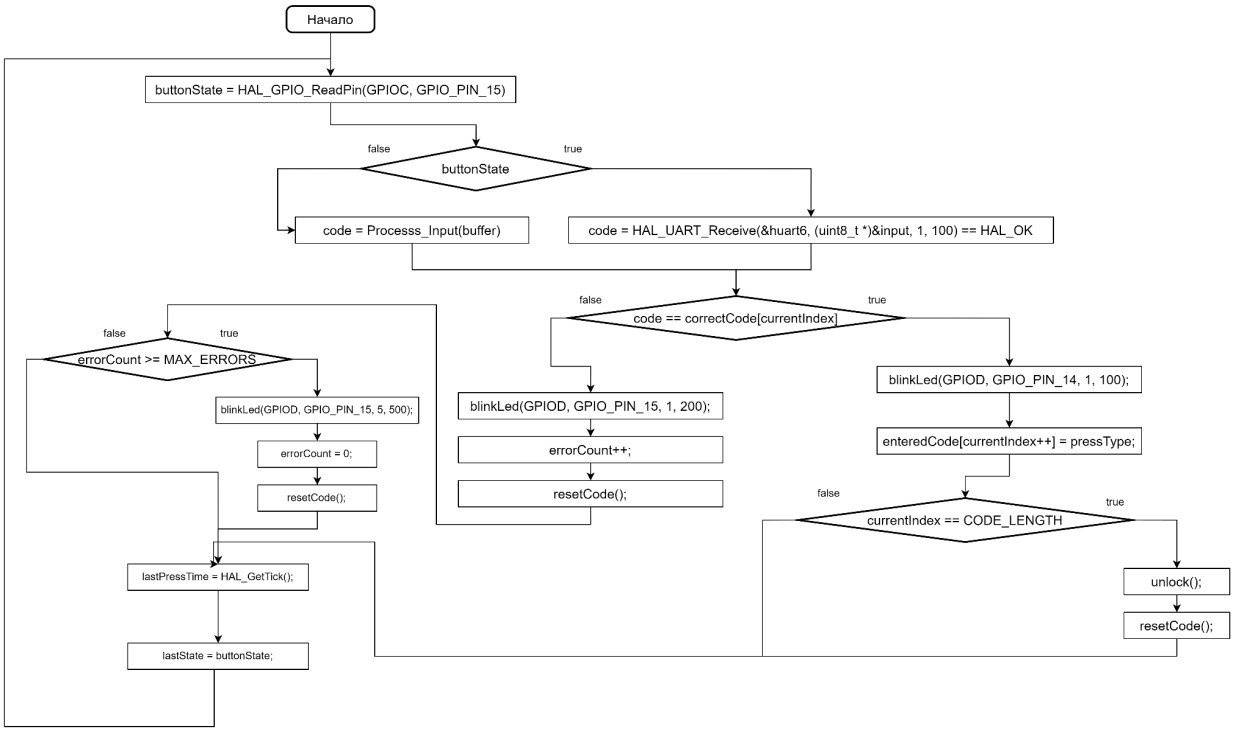
Листинг разработанной программы с комментариями

```
UART_HandleTypeDef huart6;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART6_UART_Init(void);
static void MX_NVIC_Init(void);

#define CODE_LENGTH 8
#define TIMEOUT 10000
#define MAX_ATTEMPTS 3
#define GREEN_LED_Pin GPIO_PIN_13
#define YELLOW_LED_Pin GPIO_PIN_14
#define RED_LED_Pin GPIO_PIN_15
#define BUTTON_Pin GPIO_PIN_15
#define UART_BUFFER_SIZE 64

char LOCK_CODE[CODE_LENGTH] = "stmstmst";
char input_buffer[CODE_LENGTH];
volatile uint8_t current_pos = 0;
volatile uint8_t attempt_count = 0;
volatile uint8_t mode = 0; // 0 = Unlock Mode, 1 = Change Code Mode, 2 =
Await Confirmation
volatile uint8_t uart_mode = 0; // 0 = Polling, 1 = Interrupt
uint8_t recieved_data;
char buffer = '@';
char new_code[CODE_LENGTH];
volatile uint8_t new_code_pos = 0;

char uart_buffer[UART_BUFFER_SIZE];
volatile uint8_t write_index = 0;
volatile uint8_t read_index = 0;

void UART_Init(void);
void LED_Init(void);
void System_Reset(void);
void UART_Send_Message(const char *msg);
void UART_Send_Char(char ch);
uint8_t UART_Receive_Buffer(char *data);
void UART_Buffer_Write(uint8_t data);
void Process_Input(char input);
void Change_Code_Logic(char input);
void Confirm_New_Code(char input);
void Toggle_UART_Mode(void);
```

```
void blinkLed(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, uint8_t times,
uint16_t delay);
void unlock();

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART6_UART_Init();
  MX_NVIC_Init();

  UART_Send_Message("System Ready\r\n");


  if (HAL_GPIO_ReadPin(GPIOC, BUTTON_Pin) == GPIO_PIN_RESET) {
        HAL_Delay(50);
        if (HAL_GPIO_ReadPin(GPIOC, BUTTON_Pin) == GPIO_PIN_RESET) {
              Toggle_UART_Mode();
              while (HAL_GPIO_ReadPin(GPIOC, BUTTON_Pin) ==
GPIO_PIN_RESET);
        }
  } else {
        UART_Send_Message("Polling mode enabled\r\n");
  }

  if (uart_mode == 1) {
        HAL_UART_Receive_IT(&huart6, &recieved_data, 1);
  }
  while (1) {
        char input;
        if (uart_mode == 0) {
              if (HAL_UART_Receive(&huart6, (uint8_t *)&input, 1, 100) ==
HAL_OK) {
                    Process_Input(input);
              }
        } else {
              if (buffer != '@') {
                  Process_Input(buffer);
                  buffer = '@';
              }
        }
  }
}

void UART_Send_Message(const char *msg) {
    HAL_UART_Transmit(&huart6, (uint8_t *)msg, strlen(msg),
HAL_MAX_DELAY);
}

void UART_Send_Char(char ch) {
      if (ch == '\r') {UART_Send_Message("\r\n");}
    HAL_UART_Transmit(&huart6, (uint8_t *)&ch, 1, HAL_MAX_DELAY);
}

void Process_Input(char input) {
      UART_Send_Char(input);
      if (input == '_') {
            mode = 1;
      }
      if (mode == 0) {
              if (input == LOCK_CODE[current_pos]) {
                  blinkLed(GPIOD, GPIO_PIN_14, 1, 100);
                   current_pos++;
                   if (current_pos == CODE_LENGTH) {
                        UART_Send_Message("\r\nUnlocked!\r\n");
                        unlock();
                       System_Reset();
                   }
              } else {
                  UART_Send_Message("\r\n");
                  blinkLed(GPIOD, GPIO_PIN_15, 1, 200);
                   attempt_count++;
                   if (attempt_count >= MAX_ATTEMPTS) {
                        blinkLed(GPIOD, GPIO_PIN_15, 5, 500);
                       System_Reset();
```

```
                }
            }
        } else if (mode == 1) {
            Change_Code_Logic(input);
        } else if (mode == 2) {
            Confirm_New_Code(input);
        }
}

void Change_Code_Logic(char input) {
    blinkLed(GPIOD, GPIO_PIN_14, 1, 100);
    if (input != ' ') {
        new_code[new_code_pos] = input;
        new_code_pos++;
    }
    if (new_code_pos >= CODE_LENGTH) {
        UART_Send_Message("\r\nSet as new code? (y/n)\r\n");
        mode = 2;
    }
}

void Confirm_New_Code(char input) {
    if (input == 'y' || input == 'Y') {
        strncpy(LOCK_CODE, new_code, CODE_LENGTH);
        UART_Send_Message("\r\nNew code set successfully!\r\n");
        System_Reset();
    } else if (input == 'n' || input == 'N') {
        UART_Send_Message("New code discarded.\r\n");
        System_Reset();
    }
}

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
    buffer = recieved_data;
    HAL_UART_Receive_IT(&huart6, &recieved_data, 1);
}

void unlock() {
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
  HAL_Delay(5000);
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
}

void blinkLed(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, uint8_t times,
uint16_t delay) {
  for (uint8_t i = 0; i < times; i++) {
    HAL_GPIO_WritePin(GPIOx, GPIO_Pin, GPIO_PIN_SET);
    HAL_Delay(delay);
    HAL_GPIO_WritePin(GPIOx, GPIO_Pin, GPIO_PIN_RESET);
    HAL_Delay(delay);
  }
}

void Toggle_UART_Mode(void) {
    uart_mode = !uart_mode;
    if (uart_mode) {
        UART_Send_Message("Interrupt mode enabled\r\n");
    } else {
        UART_Send_Message("Polling mode enabled\r\n");
    }
}

void System_Reset(void) {
    current_pos = 0;
    new_code_pos = 0;
    attempt_count = 0;
    mode = 0;
    memset(input_buffer, 0, CODE_LENGTH);
    blinkLed(GPIOD, GPIO_PIN_14 | GPIO_PIN_13, 3, 300);
}
```

Описание работы программы



Вывод

В ходе работы мы изучили и использовали интерфейс GPIO для обмена данными с платой и реализации кодового замка на основе светодиодов, реализовали обработку нажатия кнопки и защиты от дребезга.