

Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ «

Кафедра «Информационная безопасность систем и технологий»

ОТЧЕТ

о выполнении лабораторной работы №3

«Работа с последовательным интерфейсом передачи данных UART
микроконтроллера семейства STM32»

Дисциплина: ПМК

Группа 21ПТ2

Выполнил студент: Юдин Н.М.

Отметка о сдаче (количество баллов):

Дата сдачи:

Принял: Хворостухин С.П.,

2023

1 Цель работы: Ознакомиться с программными средствами работы по последовательному интерфейсу UART микроконтроллера STM32.

2 Задание:

2.1 Создать проект в среде Keil uVision5.

2.2 Выполнить настройку режима отладки для проекта.

2.3 Разработать программу для микроконтроллера STM32F103RB, реализующую режим командной строки по каналу UART для управления таймером в режиме широтно-импульсной модуляции. Программа должна обеспечивать:

– получение байт данных из терминала по каналу UART и выдачу их обратно в терминал

- режим «эхо»; – формирование строк из получаемых данных (в качестве признака конца строки используется байт 0x0D), сравнение полученных строк с набором команд:

– start - запуск формирования сигнала широтно-импульсной модуляции;

– stop - остановка формирования сигнала широтно-импульсной модуляции;

– period - передача параметра периода сигнала широтно-импульсной модуляции, значение передается в строке с командной, через пробел, в цифровом формате;

– pulse - передача параметра продолжительности сигнала широтно-импульсной модуляции, значение передается в строке с командной, через пробел, в цифровом формате;

– show - запрос вывода в терминал параметров сигнала широтно-импульсной модуляции: период, продолжительность, количество сформированных импульсов.

Для формирования сигнала широтно-импульсной модуляции использовать первый канал таймера TIM3, один отсчет таймера - 1мкс

2.4 Выполнить симуляцию разработанной программы.

3 Результаты работы:

3.1 Был создан проект в среде Keil uVision5.

3.2 Была выполнена настройка режима отладки для проекта.

3.3 Была разработана программа согласно варианту задания. Вариант задания приведен ниже на рисунке 1. Реализация программы приведена в приложении А.

Номер варианта	Контроллер	Скорость передачи, бод
3	USART1	9600

Рисунок 1 - Вариант задания

Для инициализации портов ввода-вывода была реализована функция `void initPort(void)`. В данной функции происходит инициализация входного порта PA9 и инициализация порта выхода PA10. Для инициализации USART1 была реализована функция `void initUSART(void)`. Бaudрейт был выбран в соответствии с вариантом и равен 9600. Для инициализации NVIC (контроллера прерываний) была реализована функция `void initNVIC(void)`.

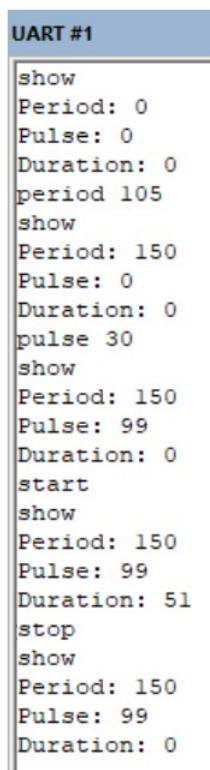
Для инициализации ШИМ была реализована функция `void interruptPWM(void)`. Для вывода в окно USART данных целыми строками была реализована функция `void sendToUSART(char * string)`. Для возможности вывода данных строками был реализован цикл ожидания обработки одного символа.

Для вычисления длительности сигнала была реализована функция `uint16_t diffTime(uint16_t a, uint16_t b)`. Данная функция вычисляет разницу во времени между значениями a и b.

Для обработки работы с командной строкой с помощью USART1 была реализована функция `void USART1_IRQHandler(void)`. В данной функции реализован режим эхо.

В теле программы происходит вызов функции `RCC_APB2PeriphClockCmd()` с параметрами `RCC_APB2Periph_USART1 | RCC_APB2Periph_GPIOA`, `ENABLE`, что включает тактирование для USART1 и GPIOA. Затем идет вызов реализованных функций. Сама программа реализована в виде бесконечного цикла, в котором находится блок `if`, срабатывающий при передаче команды `start` в USART1, в нем происходит вычисление продолжительности сигнала.

Была выполнена симуляция разработанной программы. Результат приведен ниже на рисунке 2.



```
UART #1
show
Period: 0
Pulse: 0
Duration: 0
period 105
show
Period: 150
Pulse: 0
Duration: 0
pulse 30
show
Period: 150
Pulse: 99
Duration: 0
start
show
Period: 150
Pulse: 99
Duration: 51
stop
show
Period: 150
Pulse: 99
Duration: 0
```

Рисунок 2 - Результат симуляции программы

4 Вывод: Было произведено ознакомление с программными средствами работы по последовательному интерфейсу UART микроконтроллера STM32.

Приложение А

(обязательное)

Реализация программы

```
#include "stm32f10x.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_tim.h"
#include "stm32f10x_usart.h"

temp_arr[10]; uint8_t
temp; uint8_t temp_char;

NVIC_InitTypeDef nvic;
TIM_TimeBaseInitTypeDef
timerPWM; TIM_OCInitTypeDef
oc; int
need_formation_duration = 0;
uint16_t value1 = 0; uint16_t
value2 = 0; int duration = 0;
int arr_size = 0;

void sendToUSART(char * string)
{ uint8_t i = 0;
  while(string[i]) {
    USART_SendData(USART1, string[i]); while(!
    USART_GetFlagStatus(USART1, USART_FLAG_TXE)) {
      }
    i+
    +;
  }
}

void initGPIO() {
  GPIO_InitTypeDef port;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA |
  RCC_APB2Periph_USART1,
  ENABLE);

  port.GPIO_Pin = GPIO_Pin_9;
  port.GPIO_Speed = GPIO_Speed_50MHz;
```

```

    port.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &port);

    port.GPIO_Pin = GPIO_Pin_10;
    port.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &port);
}

void initUSART() {
    USART_InitTypeDef usart;
    usart.USART_BaudRate = 9600;
    usart.USART_WordLength =
    USART_WordLength_8b;
    usart.USART_StopBits = USART_StopBits_1;
    usart.USART_Parity = USART_Parity_No;
    usart.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    usart.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
    USART_Init(USART1, &usart);

    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
    USART_Cmd(USART1, ENABLE);
}

void initNVIC() { nvic.NVIC_IRQChannel =
    USART1_IRQn;
    nvic.NVIC_IRQChannelPreemptionPrior
    ity = 0; nvic.NVIC_IRQChannelCmd =
    ENABLE;
    nvic.NVIC_IRQChannelSubPriority = 0;
    NVIC_Init(&nvic);
}

void interruptPWM() {
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

    TIM_TimeBaseStructInit(&timerPWM);
    timerPWM.TIM_Prescaler = 108 - 1;
    timerPWM.TIM_Period = 0;
    TIM_TimeBaseInit(TIM3, &timerPWM);

    TIM_OCStructInit(&oc);
    oc.TIM_OCMode = TIM_OCMode_PWM1;
    oc.TIM_OutputState= TIM_OutputState_Enable;
    oc.TIM_Pulse = 0;
    TIM_OC1Init(TIM3, &oc);

    TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);
    TIM_Cmd(TIM3, ENABLE);
    NVIC_EnableIRQ(TIM3_IRQn);
}

```

```

void USART1_IRQHandler() { if(USART_GetFlagStatus(USART1,
    USART_FLAG_RXNE)) { USART_ClearITPendingBit(USART1,
    USART_FLAG_RXNE);
    temp_arr[temp] = USART_ReceiveData(USART1);
    temp_char = temp_arr[temp];
    temp++;
    if( temp_char != 0x0D ) {
        USART_SendData(USART1, temp_char);
    }
    if( temp_char == 0x0D )
        { arr_size = temp
          - 1; temp = 0;

          if(arr_size == 5 && temp_arr[0] == 's' && temp_arr[1] == 't' &&
temp_arr[2]
== 'a' && temp_arr[3] == 'r' && temp_arr[4] == 't')
            { need_formation_duration = 1;
              sendToUSART("\n");
            }

            if(arr_size == 4 && temp_arr[0] == 's' && temp_arr[1] == 't' &&
temp_arr[2]
== 'o' && temp_arr[3] == 'p') {
                duration = 0;
                sendToUSART("\n");
            }

            if(temp_arr[0] == 'p' && temp_arr[1] == 'e' &&
temp_arr[2] == 'r' &&
temp_arr[3] == 'i' && temp_arr[4] == 'o' && temp_arr[5] == 'd') {
                int ov = 0;
                if(arr_size >
2) { int p =
2;

                    while(temp_arr[p] !=
0x0D) { ov +=
temp_arr[p];
p++;

                    }
                } timerPWM.TIM_Period
= ov; sendToUSART("\n");
            }

            if(temp_arr[0] == 'p' && temp_arr[1] == 'u' &&
temp_arr[2] == 'l' &&
temp_arr[3] == 's' && temp_arr[4] == 'e') {
                int ov = 0;
                if(arr_size >
5) { int p =
5;

                    while(temp_arr[p] !=
0x0D) { ov +=
temp_arr[p];

```

```

        p++;
    }
    } oc.TIM_Pulse
    = ov;
    sendToUSART("\
n");
}

if(temp_arr[0] == 's' && temp_arr[1] == 'h' &&
temp_arr[2] == 'o' &&
temp_arr[3] == 'w') { sendToUSART("\
n");
    sendToUSART("Period:
"); char ttt[20];
    int i = 0;
    sprintf(ttt, "%d", timerPWM.TIM_Period);
    sendToUSART(ttt);

    sendToUSART("\n");
    sendToUSART("Pulse: ");
    char ggg[20];
    sprintf(ggg, "%d", oc.TIM_Pulse);
    sendToUSART(ggg);

    sendToUSART("\n");
    sendToUSART("Duration:
"); char mmm[20];
    sprintf(mmm, "%d",
duration);
    sendToUSART(mmm);
    sendToUSART("\n");
}
}
}

uint16_t diffTime(uint16_t a, uint16_t b)
{ return ( a >> b ) ? ( a - b ) :
(UINT16_MAX - b + a);
}

int main() {
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1 | RCC_APB2Periph_GPIOA
, ENABLE);
    temp = 0;
    initNVIC();
    initGPIO();
    initUSART();
    while(1) {
        if(need_formation_duration == 1) {
            duration = timerPWM.TIM_Period -
oc.TIM_Pulse; need_formation_duration
= 0;

```



```
        }  
    }  
    return 0;  
}
```