

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

**Разработка микропрограммы устройства USB HID Joystick**

**ПО ДИСЦИПЛИНЕ «ИНТЕРФЕЙС “ЧЕЛОВЕК – ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ”»**

Факультет: АВТФ Преподаватель: Токарев В.Г.

Группа: АВТ-918

Студенты: Ванин К.Е.

Слепцов С.Р.

Саркисян С.Р.

# Цель работы

Закрепление основных понятий о взаимодействии USB HID устройства с операционной системой, принципов формирования дескрипторов различных по классам USB HID устройств, знаний об архитектуре микроконтроллеров с ядром Cortex.

# Задание к лабораторной работе

* сгенерировать Report Descriptor для устройства USB HID Joystik;
* изменить PID параметр Device Discriptor;
* разработать процедуры чтения значений датчиков положения и обзора, чтения значений нажатых кнопок, подключенных к разъемам CN1 и CN2 в соответствии со схемой Рис. 1;
* разработать процедуру, формирующую репорт на основе полученных значений.
* проверить работу готового устройства, произвести калибровку (как показано на Рис. 2), зайдя в меню "Устройства и принтеры","Параметры игровых устройств управления"

# Ход работы

Согласно методическим рекомендациям к выполнению лабораторной работы, было имплементировано новое устройство.

Был сгенерирован Report Descriptor, изменены параметры Device Descriptor.

В соответствие со схемой, изображенной на рис. 1, были разработаны процедуры чтения датчиков положения и обзора, значений нажатых кнопок.

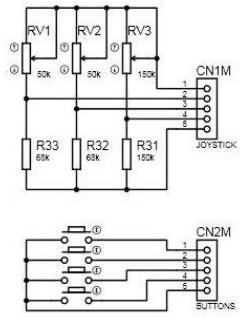


Рис. 1 – Схема элементов управления.

Была разработана процедура, формирующая репорт на основе полученных значений.

После всех выполненных действий была произведена калибровка устройства, затем – проверка его работы. Результат проверки представлен на рис. 2 и рис. 3.

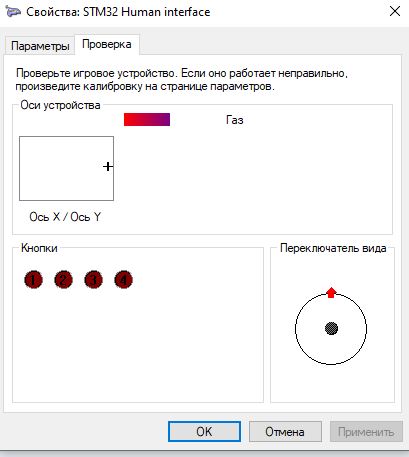


Рис. 2 – Первоначальное положение на осях X / Y.

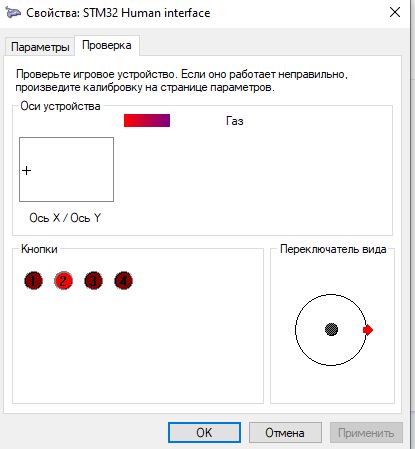


Рис. 3 – Положение на осях X / Y после перемещения элемента управления.

# Выводы

В ходе выполнения лабораторной работы были закреплены основные понятия о взаимодействии USB HID устройств с операционной системой, принципы формирования дескрипторов различных по классам USB HID устройств, знания об архитектуре микроконтроллеров с ядром Cortex.

В предыдущей лабораторной работе с помощью шаблона проекта, подготовленного с помощью генератора исходных кодов STM32CubeMX®, мы создали устройство USB HID Mouse. Теперь программный код был доработан так, чтобы у нас получился USB HID Joystick.

# Листинг

**Файл main.c**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* File Name : main.c

\* Date : 24/010/2022 14:08:29

\* Description : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* COPYRIGHT(c) 2016 STMicroelectronics

\*

\* Redistribution and use in source and binary forms, with or without modification,

\* are permitted provided that the following conditions are met:

\* 1. Redistributions of source code must retain the above copyright notice,

\* this list of conditions and the following disclaimer.

\* 2. Redistributions in binary form must reproduce the above copyright notice,

\* this list of conditions and the following disclaimer in the documentation

\* and/or other materials provided with the distribution.

\* 3. Neither the name of STMicroelectronics nor the names of its contributors

\* may be used to endorse or promote products derived from this software

\* without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

\* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

\* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

\* DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE

\* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL

\* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR

\* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

\* CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,

\* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE

\* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* Includes ------------------------------------------------------------------\*/

#include "stm32f1xx\_hal.h"

#include "usb\_device.h"

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private variables ---------------------------------------------------------\*/

ADC\_HandleTypeDef hadc1;

I2C\_HandleTypeDef hi2c2;

SPI\_HandleTypeDef hspi2;

TIM\_HandleTypeDef htim4;

UART\_HandleTypeDef huart1;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_ADC1\_Init(void);

static void MX\_I2C2\_Init(void);

static void MX\_SPI2\_Init(void);

static void MX\_TIM4\_Init(void);

static void MX\_USART1\_UART\_Init(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

int main(void)

{

\_\_IO uint16\_t aADCxConvertedValues[4];

uint8\_t HID\_Buffer[4];

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration----------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* Configure the system clock \*/

SystemClock\_Config();

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_ADC1\_Init();

MX\_I2C2\_Init();

MX\_SPI2\_Init();

MX\_TIM4\_Init();

MX\_USART1\_UART\_Init();

MX\_USB\_DEVICE\_Init();

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

HAL\_ADC\_Start(&hadc1); // программный запуск ADC

/\* Ждем, пока преобразование не завершится \*/

HAL\_ADC\_PollForConversion(&hadc1, 10); //хэндл, таймаут

aADCxConvertedValues[0] = HAL\_ADC\_GetValue(&hadc1); //забираем результат

HAL\_ADC\_Start(&hadc1); // программный запуск ADC

HAL\_ADC\_PollForConversion(&hadc1, 10); //хэндл, таймаут

aADCxConvertedValues[1] = HAL\_ADC\_GetValue(&hadc1); //забираем результат

HAL\_ADC\_Start(&hadc1); // программный запуск ADC

HAL\_ADC\_PollForConversion(&hadc1, 10); //хэндл, таймаут

aADCxConvertedValues[2] = HAL\_ADC\_GetValue(&hadc1); //забираем результат

HAL\_ADC\_Start(&hadc1); // программный запуск ADC

HAL\_ADC\_PollForConversion(&hadc1, 10); //хэндл, таймаут

aADCxConvertedValues[3] = HAL\_ADC\_GetValue(&hadc1); //забираем результат

/\* проверяем, все ли преобразования завершены \*/

if(HAL\_ADC\_GetState(&hadc1) == HAL\_ADC\_STATE\_EOC\_REG)

{

/\* процедура после завешения всех преобразований \*/

// ………………………

}

HID\_Buffer[2] = 0;

HID\_Buffer[1] = 0;

if((aADCxConvertedValues[0] - 2000) > 50 || (aADCxConvertedValues[0] - 2000) < -50){

HID\_Buffer[1] = (aADCxConvertedValues[0] - 2000)>>4;

}

if((aADCxConvertedValues[1] - 2000) > 50 || (aADCxConvertedValues[1] - 2000) < -50){

HID\_Buffer[2] = (aADCxConvertedValues[1] - 2000)>>4;

}

HID\_Buffer[0] = 0;

HID\_Buffer[3] = 0;

USBD\_HID\_SendReport(&hUsbDeviceFS, HID\_Buffer, 4);

HAL\_Delay(20);

}

/\* USER CODE END 3 \*/

}

/\*\* System Clock Configuration

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct;

RCC\_ClkInitTypeDef RCC\_ClkInitStruct;

RCC\_PeriphCLKInitTypeDef PeriphClkInit;

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.HSEPredivValue = RCC\_HSE\_PREDIV\_DIV1;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL9;

HAL\_RCC\_OscConfig(&RCC\_OscInitStruct);

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_SYSCLK|RCC\_CLOCKTYPE\_PCLK1;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2);

PeriphClkInit.PeriphClockSelection = RCC\_PERIPHCLK\_ADC|RCC\_PERIPHCLK\_USB;

PeriphClkInit.AdcClockSelection = RCC\_ADCPCLK2\_DIV6;

PeriphClkInit.UsbClockSelection = RCC\_USBPLLCLK\_DIV1\_5;

HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInit);

\_\_HAL\_RCC\_AFIO\_CLK\_ENABLE();

}

/\* ADC1 init function \*/

void MX\_ADC1\_Init(void)

{

ADC\_ChannelConfTypeDef sConfig;

/\*\*Common config

\*/

hadc1.Instance = ADC1;

hadc1.Init.ScanConvMode = ADC\_SCAN\_ENABLE; //Сканирование каналов включено

hadc1.Init.ContinuousConvMode = DISABLE; //Режим непрерывной конверсии отключен

hadc1.Init.DiscontinuousConvMode = ENABLE; //Последовательность с остановками

hadc1.Init.NbrOfDiscConversion = 1; //Стоп после каждого преобразования

hadc1.Init.ExternalTrigConv = ADC\_SOFTWARE\_START; //Программный старт

hadc1.Init.DataAlign = ADC\_DATAALIGN\_RIGHT; //Данные сдвинуты вправо

hadc1.Init.NbrOfConversion = 4; //Общее число преобразований

HAL\_ADC\_Init(&hadc1); //инициализируем заданные параметры

/\*\*Configure Regular Channel

\*/

sConfig.Channel = ADC\_CHANNEL\_1; //первый канал PA1

sConfig.Rank = 1; //первый номер в очереди

sConfig.SamplingTime = ADC\_SAMPLETIME\_1CYCLE\_5; //Сэмпл тайм 1.5 такта

HAL\_ADC\_ConfigChannel(&hadc1, &sConfig); //инициализируем параметры

sConfig.Channel = ADC\_CHANNEL\_2; //второй канал PA2

sConfig.Rank = 2; //второй номер в очереди

sConfig.SamplingTime = ADC\_SAMPLETIME\_1CYCLE\_5; //Сэмпл тайм 1.5 такта

HAL\_ADC\_ConfigChannel(&hadc1, &sConfig); //инициализируем параметры

sConfig.Channel = ADC\_CHANNEL\_3; //третий канал PA3

sConfig.Rank = 3; //третий номер в очереди

sConfig.SamplingTime = ADC\_SAMPLETIME\_1CYCLE\_5; //Сэмпл тайм 1.5 такта

HAL\_ADC\_ConfigChannel(&hadc1, &sConfig); //инициализируем параметры

sConfig.Channel = ADC\_CHANNEL\_16; //16 канал датчик температуры

sConfig.Rank = 4; //четвертый номер в очереди

sConfig.SamplingTime = ADC\_SAMPLETIME\_1CYCLE\_5; //Сэмпл тайм 1.5 такта

HAL\_ADC\_ConfigChannel(&hadc1, &sConfig); //инициализируем параметры

}

/\* I2C2 init function \*/

void MX\_I2C2\_Init(void)

{

hi2c2.Instance = I2C2;

hi2c2.Init.ClockSpeed = 100000;

hi2c2.Init.DutyCycle = I2C\_DUTYCYCLE\_2;

hi2c2.Init.OwnAddress1 = 0;

hi2c2.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;

hi2c2.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLED;

hi2c2.Init.OwnAddress2 = 0;

hi2c2.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLED;

hi2c2.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLED;

HAL\_I2C\_Init(&hi2c2);

}

/\* SPI2 init function \*/

void MX\_SPI2\_Init(void)

{

hspi2.Instance = SPI2;

hspi2.Init.Mode = SPI\_MODE\_MASTER;

hspi2.Init.Direction = SPI\_DIRECTION\_2LINES;

hspi2.Init.DataSize = SPI\_DATASIZE\_8BIT;

hspi2.Init.CLKPolarity = SPI\_POLARITY\_LOW;

hspi2.Init.CLKPhase = SPI\_PHASE\_1EDGE;

hspi2.Init.NSS = SPI\_NSS\_SOFT;

hspi2.Init.BaudRatePrescaler = SPI\_BAUDRATEPRESCALER\_2;

hspi2.Init.FirstBit = SPI\_FIRSTBIT\_MSB;

hspi2.Init.TIMode = SPI\_TIMODE\_DISABLED;

hspi2.Init.CRCCalculation = SPI\_CRCCALCULATION\_DISABLED;

HAL\_SPI\_Init(&hspi2);

}

/\* TIM4 init function \*/

void MX\_TIM4\_Init(void)

{

TIM\_ClockConfigTypeDef sClockSourceConfig;

TIM\_MasterConfigTypeDef sMasterConfig;

TIM\_OC\_InitTypeDef sConfigOC;

htim4.Instance = TIM4;

htim4.Init.Prescaler = 0;

htim4.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim4.Init.Period = 0;

htim4.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

HAL\_TIM\_Base\_Init(&htim4);

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

HAL\_TIM\_ConfigClockSource(&htim4, &sClockSourceConfig);

HAL\_TIM\_PWM\_Init(&htim4);

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

HAL\_TIMEx\_MasterConfigSynchronization(&htim4, &sMasterConfig);

sConfigOC.OCMode = TIM\_OCMODE\_PWM1;

sConfigOC.Pulse = 0;

sConfigOC.OCPolarity = TIM\_OCPOLARITY\_HIGH;

sConfigOC.OCFastMode = TIM\_OCFAST\_DISABLE;

HAL\_TIM\_PWM\_ConfigChannel(&htim4, &sConfigOC, TIM\_CHANNEL\_1);

HAL\_TIM\_PWM\_ConfigChannel(&htim4, &sConfigOC, TIM\_CHANNEL\_2);

HAL\_TIM\_PWM\_ConfigChannel(&htim4, &sConfigOC, TIM\_CHANNEL\_3);

HAL\_TIM\_PWM\_ConfigChannel(&htim4, &sConfigOC, TIM\_CHANNEL\_4);

}

/\* USART1 init function \*/

void MX\_USART1\_UART\_Init(void)

{

huart1.Instance = USART1;

huart1.Init.BaudRate = 115200;

huart1.Init.WordLength = UART\_WORDLENGTH\_8B;

huart1.Init.StopBits = UART\_STOPBITS\_1;

huart1.Init.Parity = UART\_PARITY\_NONE;

huart1.Init.Mode = UART\_MODE\_TX\_RX;

huart1.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart1.Init.OverSampling = UART\_OVERSAMPLING\_16;

HAL\_UART\_Init(&huart1);

}

/\*\* Configure pins as

\* Analog

\* Input

\* Output

\* EVENT\_OUT

\* EXTI

\*/

void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct;

/\* GPIO Ports Clock Enable \*/

\_\_GPIOC\_CLK\_ENABLE();

\_\_GPIOD\_CLK\_ENABLE();

\_\_GPIOA\_CLK\_ENABLE();

\_\_GPIOB\_CLK\_ENABLE();

/\*Configure GPIO pin : PC13 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_13;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_LOW;

HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct);

/\*Configure GPIO pins : PA0 PA15 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_0|GPIO\_PIN\_15;

GPIO\_InitStruct.Mode = GPIO\_MODE\_EVT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pins : PA4 PA5 PA6 PA7 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6|GPIO\_PIN\_7;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pin : PB12 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_12;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_LOW;

HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);

/\*Configure GPIO pin : PA8 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_8;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_LOW;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*\*ADC1 GPIO Configuration

PA1 ------> ADC1\_IN1

PA2 ------> ADC1\_IN2

PA3 ------> ADC1\_IN3

\*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3;

GPIO\_InitStruct.Mode = GPIO\_MODE\_ANALOG;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t\* file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif

/\*\*

\* @}

\*/

/\*\*

\* @}

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMic