МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 3
З дисципліни: *"Програмування мікроконтролерів"*
На тему: *" Робота з інтерфейсом CAN на базі мікроконтролера STM32F407"*

Лектор:
доц. каф. ПЗ
Марусенкова Т.А.

Виконав:
ст. гр. ПЗ-42
Бурець В.В.

Прийняв:
доц. каф. ПЗ
Марусенкова Т.А.

« ____ » _____ 2022 р.

∑= ____ _____.

Львів-2022

Тема роботи: Робота з інтерфейсом CAN на базі мікроконтролера STM32F407.

Мета роботи: Навчитись створювати зв'язок між кількома мікроконтролерами STM32 через інтерфейс CAN.

ЗАВДАННЯ

1. Використовуючи надані приклади кодів, створити проект з використанням заданої бібліотеки, в якому в мережу CAN надсилається повідомлення, що містить Ваше прізвище та ім'я при настанні умови, визначеної індивідуальним завданням.
2. Оформити звіт. В розділі «Теоретичні відомості» дати відповідь на контрольне запитання, визначене порядковим номером у підгрупі. В звіті навести ту частину коду проекту, що відповідає безпосередньо за виконання завдання (бібліотечні коди можна не наводити).

Індивідуальне завдання: 1. натиснення кнопки, SPL

ХІД РОБОТИ

Функція main

```
#include "stm32f4xx.h"
#include "main.h"

#include <stm32f4xx_rcc.h>
#include <stm32f4xx_gpio.h>
#include <misc.h>
#include <stm32f4xx_syscfg.h>

static __IO uint32_t isSendDataToCan = 0;
static __IO uint32_t isGetDataFromCan = 0;

static __IO uint32_t _tempTimingDelay;
void Delay(__IO uint32_t nTime)
{
  _tempTimingDelay = nTime;

  while(_tempTimingDelay != 0);
}

void TimingDelay_Decrement(void){
  if (_tempTimingDelay != 0x00){
    _tempTimingDelay--;
  }
}
void SysTick_Handler(void)
{
  TimingDelay_Decrement();
}

void EXTI0_IRQHandler(void){
        Delay(20);
        if(GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_0)){
                isSendDataToCan = 1;
        }
        EXTI_ClearITPendingBit(EXTI_Line0);
```

```
}

void CAN1_RX0_IRQHandler(void){
	isGetDataFromCan = 1;
}

int main(){
	SystemInit();
	SysTick_Config(SystemCoreClock/1000);
	CAN1_Config();

	ButtonInit();
  it_init();

	const int size= 17;
	uint32_t id = 0x10;
	uint8_t data[] = {0x56, 0x61, 0x6c, 0x65, 0x6e, 0x74, 0x79, 0x6e, 0x20, 0x42, 0x75, 0x72, 0x65, 0x74,
0x73};

	while(1)
	{
		if(isSendDataToCan == 1){
			int status = sendDataToCan(id, data, size);

			if(status !=0){
				return -1;
			}
			else{
				isSendDataToCan = 0;
			}
		}
		if(isGetDataFromCan == 1){
			int status = receiveDataFromCan(id, data, size);

			if(status !=0){
				return -1;
			}
			else{
				isGetDataFromCan = 0;
			}
		}
	};

	return 0;
}
```

Ініціалізація Can

```
#include "can.h"

int sendDataToCan(uint32_t id, uint8_t* data, int sizeInBytes){
	CanTxMsg message;
	message.DLC = 8;
	message.ExtId = id;
	message.IDE = CAN_Id_Extended;
	message.RTR = CAN_RTR_DATA;
	message.StdId = 0x321;

	if(sizeInBytes > 8 || sizeInBytes < 0)
		return -1;
	for(int i=0; i< sizeInBytes; ++i){
		message.Data[i] = *data;
	}

	int status = CAN_Transmit(CAN1, &message);
	if(status == CAN_TxStatus_Failed)
		return -1;
```

```
        return 0;
}
int receiveDataFromCan(uint32_t id, uint8_t* data, int sizeInBytes){
        int status = 0;
        CanRxMsg message;
        message.IDE = CAN_Id_Extended;
        message.DLC = sizeInBytes;
        message.ExtId = id;
        message.RTR= CAN_RTR_DATA;
        message.FMI = 0;
        CAN_Receive(CAN1,1, &message); //find fifo number
        data = message.Data;
        return status;
}
```

Ініціалізація кнопки

```
#include "button_init.h"

RCC_ClocksTypeDef RCC_Clocks;

void ButtonInit() {

        GPIO_InitTypeDef GPIO_InitStructure;
        GPIO_StructInit(&GPIO_InitStructure);

        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

        RCC_GetClocksFreq(&RCC_Clocks);
  SysTick_Config(RCC_Clocks.HCLK_Frequency / 1000);

}

void it_init() {
                //set interrupt port    port   / out port
    SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOA,EXTI_PinSource0); //associate PA0 with EXTI0


        //external interrupt
                EXTI_InitTypeDef exti;   //create struct for working with EXTI_IMR & EXTI_RTSR registers
    exti.EXTI_Line = EXTI_Line0;
    exti.EXTI_Mode = EXTI_Mode_Interrupt;    //EXTI_IMR ? EXTI_EMR means event or interrapts
    exti.EXTI_Trigger =   EXTI_Trigger_Rising_Falling;//EXTI_Trigger_Rising(Falling/Rizing_Falling)
    exti.EXTI_LineCmd = ENABLE;    //set state
    EXTI_Init(&exti);


        //nested vector interrupt controller
    NVIC_InitTypeDef nvic; // set params for contoller vector interrupts
    nvic.NVIC_IRQChannel = EXTI0_IRQn;   // set Chanal IRG for activation/disactivation
    nvic.NVIC_IRQChannelPreemptionPriority = 13;
    nvic.NVIC_IRQChannelSubPriority = 13;
    nvic.NVIC_IRQChannelCmd = ENABLE;  //set activity
    NVIC_Init(&nvic);
}
```

Ініціалізація Can

```
#include "can_init.h"

CAN_InitTypeDef CAN_InitStructure;
CAN_FilterInitTypeDef CAN_FilterInitStructure;
void CAN1_Config(void)
```

```
{
GPIO_InitTypeDef GPIO_InitStructure;
/* CAN GPIOs configuration */
RCC_AHB1PeriphClockCmd(CAN1_GPIO_CLK, ENABLE);
GPIO_PinAFConfig(CAN1_GPIO_PORT, CAN1_RX_SOURCE, GPIO_AF_CAN1);
GPIO_PinAFConfig(CAN1_GPIO_PORT, CAN1_TX_SOURCE, GPIO_AF_CAN1);
/* Налаштування RX і TX */
GPIO_InitStructure.GPIO_Pin = CAN1_RX_PIN | CAN1_TX_PIN;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
GPIO_Init(CAN1_GPIO_PORT, &GPIO_InitStructure);
/* Налаштування CAN */
/* Тактування */
RCC_APB1PeriphClockCmd(CAN1_GPIO_CLK, ENABLE);
CAN_DeInit(CAN1);
/* ініціалізація структури CAN */
CAN_InitStructure.CAN_TTCM = DISABLE;
CAN_InitStructure.CAN_ABOM = DISABLE;
CAN_InitStructure.CAN_AWUM = DISABLE;
CAN_InitStructure.CAN_NART = ENABLE;
CAN_InitStructure.CAN_RFLM = DISABLE;
CAN_InitStructure.CAN_TXFP = DISABLE;
CAN_InitStructure.CAN_Mode = CAN_Mode_Normal;
CAN_InitStructure.CAN_SJW = CAN_SJW_1tq;
CAN_InitStructure.CAN_BS1 = CAN_BS1_3tq;
CAN_InitStructure.CAN_BS2 = CAN_BS2_3tq;
CAN_InitStructure.CAN_Prescaler = 12;
CAN_Init(CAN1, &CAN_InitStructure);

CAN_FilterInitStructure.CAN_FilterNumber = 0;
CAN_FilterInitStructure.CAN_FilterMode = CAN_FilterMode_IdMask;
CAN_FilterInitStructure.CAN_FilterScale = CAN_FilterScale_32bit;
CAN_FilterInitStructure.CAN_FilterIdHigh = 0x0000;
CAN_FilterInitStructure.CAN_FilterIdLow = 0x0000;
CAN_FilterInitStructure.CAN_FilterMaskIdHigh = 0x0000;
CAN_FilterInitStructure.CAN_FilterMaskIdLow = 0x0000;
CAN_FilterInitStructure.CAN_FilterFIFOAssignment = 0;
CAN_FilterInitStructure.CAN_FilterActivation = ENABLE;
CAN_FilterInit(&CAN_FilterInitStructure);

/* Вмикаємо переривання */
CAN_ITConfig(CAN1, CAN_IT_FMP0, ENABLE);
CAN_ITConfig(CAN1, CAN_IT_FMP1, ENABLE);
CAN_ITConfig(CAN1, CAN_IT_TME, ENABLE);
}
```

ВИСНОВКИ

На даній лабораторній я організував взаємодію кількох мікроконтролерів STM32F407VG за допомогою інтерфейсу CAN. Інтерфейси були реалізовані за допомогою бібліотеки SPL.