Национальный исследовательский университет ИТМО
Факультет Программной Инженерии и Компьютерной Техники

Вариант №3
Лабораторная работа №4
«Интерфейс I2C и матричная клавиатура»
По дисциплине:
«Проектирование вычислительных систем»

Работу выполнили:
Никонова Наталья Игоревна
Сенина Мария Михайловна
Группа: P34102
Преподаватель:
Пинкевич Василий Юрьевич

Санкт-Петербург
2023

Цель работы

1. Получить базовые знания об интерфейсе I2C и особенностях передачи данных по данному интерфейсу.
2. Получить базовые знания об устройстве и принципах работы контроллера интерфейса I2C в микроконтроллерах и получить навыки его программирования.

Задание

Разработать программу, которая использует интерфейс I2C для считывания нажатий кнопок клавиатуры стенда SDK-1.1.

Подсистема опроса клавиатуры должна удовлетворять следующим требованиям:

- реализуется защита от дребезга;
- нажатие кнопки фиксируется сразу после того, как было обнаружено, что кнопка нажата (с учетом защиты от дребезга), а не в момент отпускания кнопки; если необходимо, долгое нажатие может фиксироваться отдельно;
- кнопка, которая удерживается дольше, чем один цикл опроса, не считается повторно нажатой до тех пор, пока не будет отпущена (нет переповторов);
- распознается и корректно обрабатывается множественное нажатие (при нажатии более чем одной кнопки считается, что ни одна кнопка не нажата, если это не противоречит требованиям к программе);
- всем кнопкам назначаются коды от 1 до 12 (порядок на усмотрение исполнителей).

Программа должна иметь два режима работы, переключение между которыми производится по нажатию кнопки на боковой панели стенда:

- режим тестирования клавиатуры;
- прикладной режим.

Уведомление о смене режима выводится в UART.

В режиме тестирования клавиатуры программа выводит в UART коды нажатых кнопок.

В прикладном режиме программа обрабатывает нажатия кнопок и выполняет действия в соответствии с вариантом задания.

Задания аналогичны вариантам лабораторной работы №3, за исключением того, что ввод символов должен выполняться не с клавиатуры через UART, а с помощью клавиатуры стенда. Выбор кнопок клавиатуры стенда, играющих роль кнопок клавиатуры компьютера должен выполняться по усмотрению исполнителей. В отчете необходимо привести описание функций кнопок в реализованной программе.

Вариант лабораторной работы №3

Вариант 3

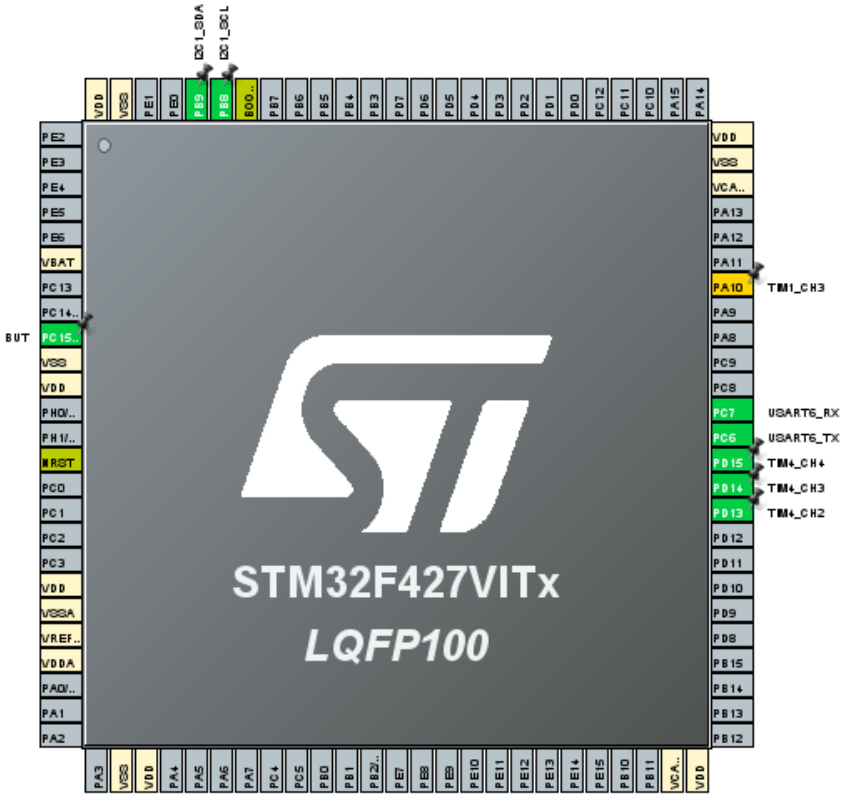
Реализовать имитатор гирлянды с плавными включениями/выключениями светодиодов и переходами между цветами. Должно быть предусмотрено четыре разных предустановленных режима анимации и один пользовательский режим, который можно настраивать. В каждом режиме должно быть задействовано минимум два светодиода, при этом не обязательно, чтобы они могли гореть одновременно.

Действия стенда при получении символов от компьютера:

Символ	Действие
«1»	Переход на следующий режим.
«2»	Возврат к предыдущему режиму.
«3»	Ускорение воспроизведения анимации на 10 % от текущей скорости.
«4»	Замедление воспроизведения анимации на 10 % от текущей скорости.
«5»	Вывод в UART параметров текущего сохраненного пользовательского режима.
«Enter»	Вход в меню настройки.
На усмотрение исполнителей	Ввод параметров пользовательского режима: цвет светодиода, яркость, необходимость плавного перехода, длительность, конец последовательности и т.п.

После ввода каждого символа в UART должно выводиться сообщение о том, какой режим активирован, или текущие настройки, вводимые в меню.

Используемые контакты

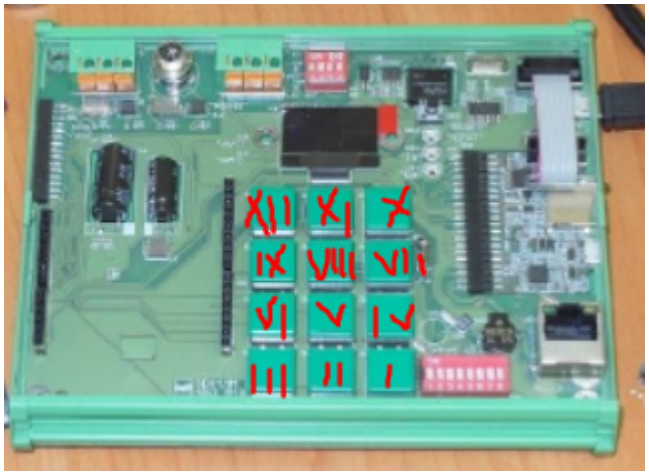


- **PC15** – перехватывает нажатия кнопки (обозначен как BUT, настроен на GPIO_INPUT)
- **PD13 (TIM4_CH2)** – на 2ом канале стоит PWM 4го таймера. Он управляет зеленым светодиодом.
- **PD14 (TIM4_CH3)** – на 3ем канале стоит PWM 4го таймера. Управляет желтым светодиодом.

- **PD15 (TIM4_CH4)** – на 4ом канале стоит PWM 4го таймера. Управляет красным светодиодом.
- **PA10 (TIM1_CH3)** – получает прерывания от таймера
- **PC7 (USART_RX)** – получает прерывание RX о том, что uart завершил прием данных
- **PC6 (USART_TX)** – получает прерывание TX о том, что uart завершил отправку данных
- **PB3, PB4, PA13, PA14, PA15** – J-Tag для отладки

Описание алгоритма

Нумерация и функции кнопок



Кнопки нумеруются справа-налево и снизу-вверх. На рисунке обозначена нумерация в римских цифрах. Функциональность представлена в таблице.

№ кнопки	Функция в режиме проигрывания	Функция в режиме настройки
1	Переход в следующий режим воспроизведения	При вводе цвета - зеленый, при вводе яркости и длительности - соот. значения
2	Переход в предыдущий режим воспроизведения	При вводе цвета - желтый, при вводе яркости и длительности - соот. значения
3	Ускорение воспроизведения на 10%	При вводе цвета - красный, при вводе яркости и длительности - соот. значения
4	Замедление воспроизведения на 10%	При вводе яркости и длительности - соот. значения
5	Вывод конфигурации пользовательского режима	При вводе яркости и длительности - соот. значения
6	-	При вводе яркости и длительности - соот. значения

7	-	При вводе яркости и длительности - соот. значения
8	-	При вводе яркости и длительности - соот. значения
9	-	При вводе яркости и длительности - соот. значения
10	-	При вводе плавного перехода - есть плавный переход
11	-	При вводе плавного перехода - нет плавного перехода
12	Переход в режим настройки	Сохранение введенной конфигурации и переход в режим воспроизведения

Модуль расширителя портов I/O

Содержит в себе удобные функции-оболочки:

- для чтения входного порта PCA9538_Read_Register;
- для чтения/записи любого из регистров PCA9538_Write_Register/PCA9538_Read_Inputs.

Они нужны чтобы не передавать одинаковые параметры, плюс туда подключены на всякий случай маски для адреса устройства на бит, показывающий. происходит ли чтение или запись.

Они работают по опросу с таймаутом в 100мс - мы честно не успели переделать на прерывания.

```
HAL_StatusTypeDef PCA9538_Read_Register(uint16_t addr, pca9538_regs_t reg, uint8_t*
buf, I2C_HandleTypeDef hi2c1) {

	return HAL_I2C_Mem_Read(&hi2c1, addr | 1, reg, 1, buf, 1, 100);

}


HAL_StatusTypeDef PCA9538_Write_Register(uint16_t addr, pca9538_regs_t reg,
uint8_t* buf, I2C_HandleTypeDef hi2c1) {

	return HAL_I2C_Mem_Write(&hi2c1, addr & 0xFFFE, reg, 1, buf, 1, 100);

}


HAL_StatusTypeDef PCA9538_Read_Inputs(uint16_t addr, uint8_t* buf,
I2C_HandleTypeDef hi2c1) {

	return PCA9538_Read_Register(addr, INPUT_PORT, buf, hi2c1);

}
```

Модуль клавиатуры

Его основная функция - CheckedRow, которая по “условному” номеру строки возвращает “условный” номер нажатой кнопки.

“Условный” номер строки - число, в котором установлен в ноль бит с тем же номером, с каким нам нужна строка.

“Условный” номер нажатой кнопки - число, в котором установлен в единицу бит с тем же номером, с каким есть нажатая кнопка.

Определение какая кнопка нажата реализовано через логическое И с инверсным вариантом регистра (т.е. 1 в бите соответствующей нажатой кнопке, а у нас на вход с клавиатуры получается наоборот, 0 где нажата кнопка). Если получилось в результате ноль - значит только эта кнопка была нажата, и мы возвращаем соответствующее значение. Если было нажато более одной кнопки, то считаем то ничего не нажато и возвращаем ноль.

Пример:

Допустим, при опросе какой-либо строки у нас была нажата самая левая кнопка. Тогда нам вернется X110 XXXX (X обозначены биты, не относящиеся к столбцам)

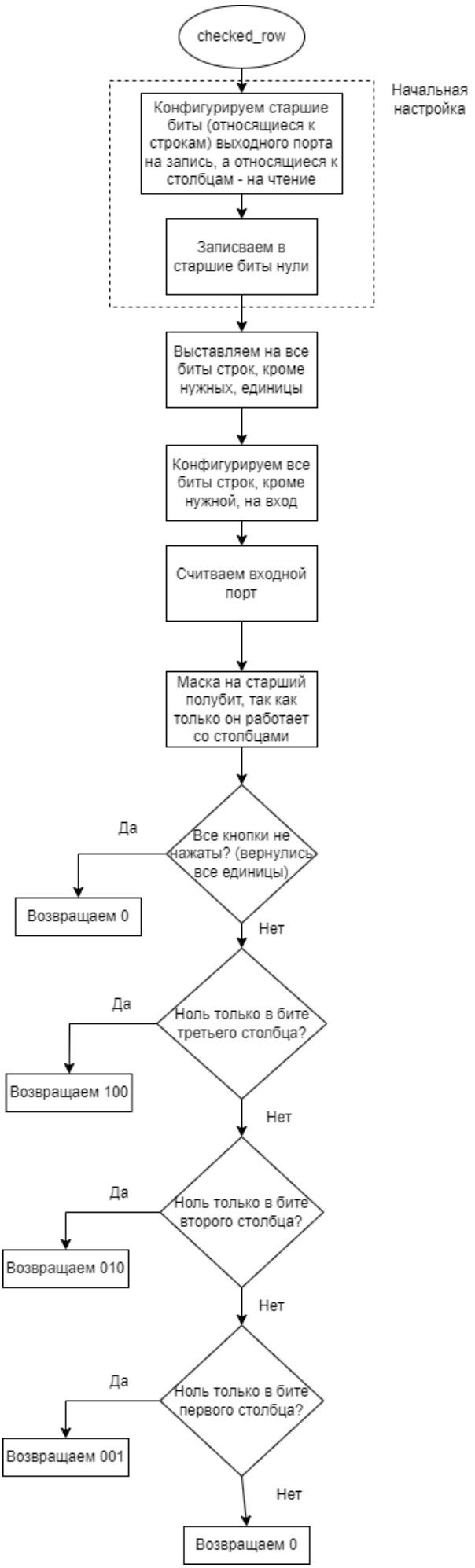
После маски получили 0110 0000

Инверсный вариант для проверки нажатия именно этой кнопки - 0001 0000.

Через операцию И получаем ноль, а так как в bat у нас отрицание, то мы туда попадаем. Если бы был ещё один ноль в каком-либо другом бите столбцов, то ноль в результате И мы бы не получили, а значит вышли бы на последнюю ветку else и вернули, что никакая кнопка не нажата.

```
#define KBRD_RD_ADDR 0xE3

#define KBRD_WR_ADDR 0xE2

#define ROW1 0xFE //1111 1110

#define ROW2 0xFD //1111 1101

#define ROW3 0xFB //1111 1011

#define ROW4 0xF7 //1111 0111


HAL_StatusTypeDef Set_Keyboard(I2C_HandleTypeDef hi2c1) {

	HAL_StatusTypeDef ret = HAL_OK;

	uint8_t buf;


	buf=0x70;

	ret = PCA9538_Write_Register(KBRD_WR_ADDR, CONFIG, &buf, hi2c1);

	if( ret != HAL_OK ) {

		return ret;

	}


	buf = 0;
```

```
	ret = PCA9538_Write_Register(KBRD_WR_ADDR, OUTPUT_PORT, &buf, hi2c1);

	return ret;
}

uint8_t Check_Row( uint8_t  Nrow, I2C_HandleTypeDef hi2c1 ) {
	uint8_t Nkey = 0x00;
	HAL_StatusTypeDef ret = HAL_OK;
	uint8_t buf = 0;
	uint8_t kbd_in;

	ret = Set_Keyboard(hi2c1);

	buf = Nrow;
	ret = PCA9538_Write_Register(KBRD_WR_ADDR, OUTPUT_PORT, &buf, hi2c1);
	ret = PCA9538_Write_Register(KBRD_WR_ADDR, CONFIG, &buf, hi2c1);

	buf = 0;
	ret = PCA9538_Read_Inputs(KBRD_RD_ADDR, &buf, hi2c1);

	kbd_in = buf & 0x70;
	Nkey = kbd_in;
	if( kbd_in != 0x70) {
		if( !(kbd_in & 0x10) ) {
			Nkey = 0x04;
		} else if( !(kbd_in & 0x20) ) {
			Nkey = 0x02;
		} else if( !(kbd_in & 0x40) ) {
			Nkey = 0x01;
		} else {
			Nkey = 0x00;
		}
	}
	else Nkey = 0x00;

	return Nkey;
}
```


checked_row
Начальная настройка
Конфигурируем старшие биты (относящиеся к строкам) выходного порта на запись, а относящиеся к столбцам - на чтение
Записываем в старшие биты нули
Выставляем на все биты строк, кроме нужных, единицы
Конфигурируем все биты строк, кроме нужной, на вход
Считываем входной порт
Маска на старший полубит, так как только он работает со столбцами
Все кнопки не нажаты? (вернулись все единицы)
Да
Возвращаем 0
Нет
Ноль только в бите третьего столбца?
Да
Возвращаем 100
Нет
Ноль только в бите второго столбца?
Да
Возвращаем 010
Нет
Ноль только в бите первого столбца?
Да
Возвращаем 001
Нет
Возвращаем 0

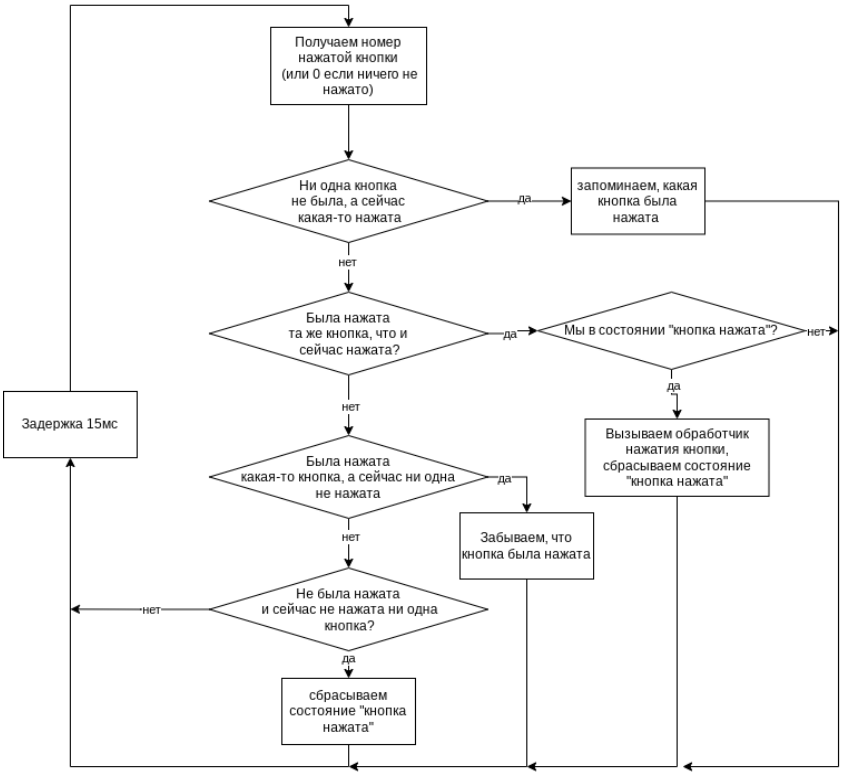
Основной модуль

Основная логика реализуется в колбеке таймера 1 и главном цикле while. Там мы ждём, нажатий кнопок (боковой и и клавиатуры), проверяем их на дребезг и в случае дребезга мы запускаем логику обработки значения этой кнопки. Дребезг везде обрабатывается похоже.

Логика обработки боковой кнопки


```
if (tick % 5 == 0) { // логика работы с дребезгом и кнопками
 int but = HAL_GPIO_ReadPin(BUT_GPIO_Port, BUT_Pin);
 but = !but;


  if (but == 1 && noisy_but == 0) {
     noisy_but = 1;
  } else if (but == 1 && noisy_but == 1) {
              if (is_pressed_but == 0) {
                is_pressed_but = 1;
                kb_testing = !kb_testing;
              }
       } else if (noisy_but == 1 && but == 0) {
              noisy_but = 0;
       } else if (but == 0 && noisy_but == 0) {
              is_pressed_but = 0;
            }
}
```

Логика обработки клавиатуры




```
while (1) {
  uint8_t p_key = get_pressed_key();
  if (p_key != 0 && noisy_key == 0) {
    noisy_key = p_key;
  } else if (p_key != 0 && noisy_key == p_key) {
    if (is_pressed_key == 0) {
      is_pressed_key = 1;
      handler_pressed_key(p_key);
    }
  } else if (noisy_key != 0 && p_key == 0) {
    noisy_key = 0;
  } else if (noisy_key == 0 && p_key == 0) {
    is_pressed_key = 0;
  }
  HAL_Delay(15);
}
```

Заключение

В этой лабораторной работе мы познакомились с работой с протоколом I2C и клавиатурой подключенной через расширитель портов PCA9538. Разобрались, как избежать дребезга на клавиатуре и кнопке.