Санкт-Петербургский политехнический университет Петра Великого
Институт электроники и телекоммуникаций
Высшая школа прикладной физики и космических технологий

Курсовая работа

Русская клавиатура для телефона

по дисциплине «Цифровые устройства и микропроцессоры»

направление 11.03.02 - «Инфокоммуникационные технологии и системы связи»

Выполнил
студент гр. 4931102/20101 Суханов С.С.

Преподаватель Тетерин П.С.

«___» _____ 2025 г.

Санкт-Петербург
2025

Оглавление

Задача

Реализовать функциональность русской клавиатуры с кнопочных телефонов.

Схемы подключения

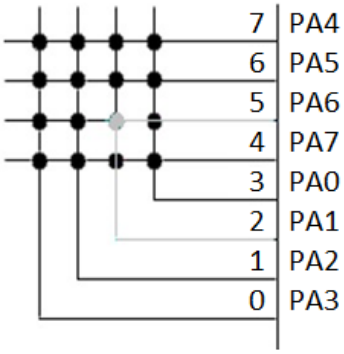

Рис. 2 Схема подключения клавиатуры к микроконтроллеру.

На рисунке 2 клавиатура подключена к 8 пинам микроконтроллера от PA0 до PA7, где PA0-PA3 являются портами, которые принимают напряжения с клавиатуры, а PA4-PA7 – посылают напряжения на клавиатуру.

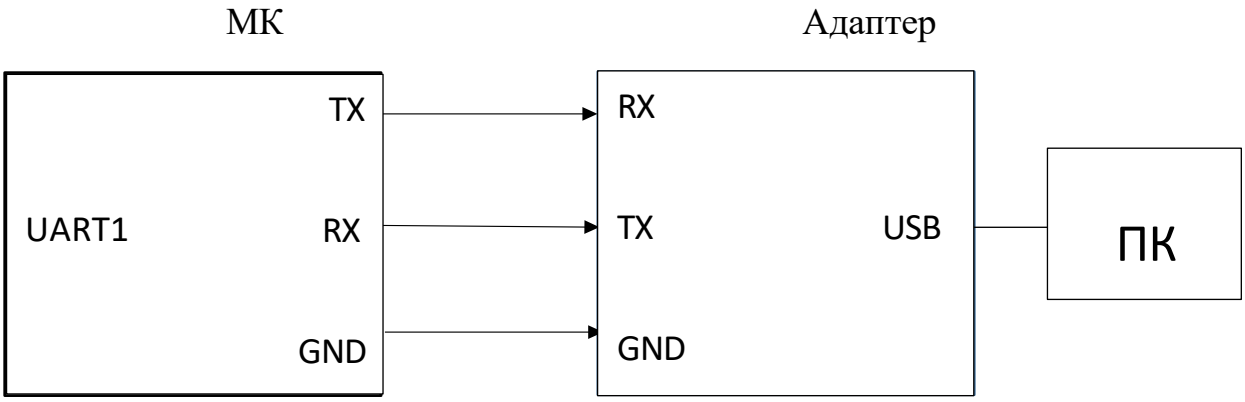

Рис.2 Схема подключения ПК, адаптер и МК.

На схеме показано, как микроконтроллер и USB-адаптер объединяются в единую цепь для организации связи с ПК: вывод TX порта UART1 микроконтроллера соединяется с входом RX адаптера, а вывод RX микроконтроллера соответственно подключается к выходу TX адаптера, при этом общий провод GND микроконтроллера и адаптера обязательно связывается, чтобы обеспечить общий нулевой потенциал. USB-разъем адаптера служит интерфейсом к компьютеру, преобразуя уровни UART в USB-сигналы для передачи данных на ПК и приёма команд с него.

Блок схемы алгоритмов программ и их описание

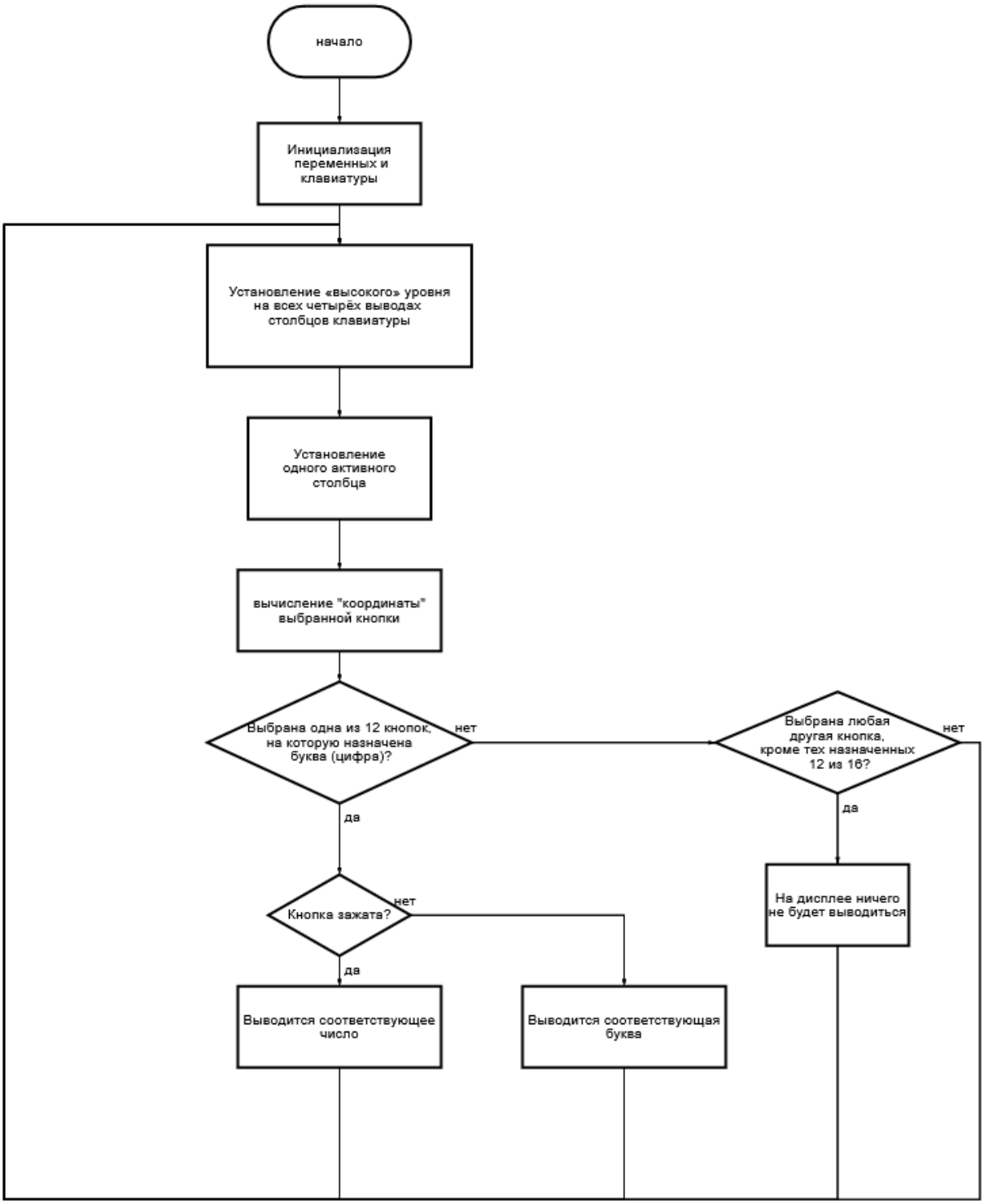

Таблица 1. Команды для клавиатуры.

Номер кнопки	**Команда (буквы/символы/цифры, которые будут выводиться на дисплей)**	
1	**Буквы/символы и спец. команды**	**Цифры и Спец. команды**
	.,?!	1
2	ABC,	2
3	DEF,	3
4	GHI,	4
5	JKL,	5
6	MNO,	6
7	PQRS	7
8	TUV	8
9	WXYZ	9
10	+*#	Перенос строки
11	space	0
12	delete	
13-16	Ничего не выводится на дисплей	

Схема начинается с инициализации всех необходимых переменных и клавиатуры, после чего на четыре выхода, отвечающие за столбцы, устанавливается «высокий» уровень. Далее в бесконечном цикле поочерёдно активируется каждый столбец клавиатуры и определяется, какая именно кнопка нажата — сначала вычисляются координаты выбранного столбца и строки, затем проверяется её номер. Если это одна из первых 12 кнопок, то на дисплей выводится соответствующая буква таблице 1 (если кнопка зажата, то выводится соответствующее число), а если нажата кнопка 13-16, то ничего не произойдет. После обработки нажатия/зажатия схема возвращается к активации следующего столбца и повторяет весь алгоритм заново.

Вывод: в ходе работы была реализована следующая задача: вывод на дисплей соответствующей буквы/цифры в зависимости от нажатой кнопки, что позволило закрепить опыт подключения клавиатуры к МК и дисплею.

Приложение

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "main.h"


#define HOLD_THRESHOLD          700   // ms
#define MULTITAP_TIMEOUT        1000   // ms



//const char *letters_map[12] = {
//        "", ".,?!", "АБВГ",  "ДЕЖЗ",  "ИЙКЛ",
//            "МНОП", "РСТУ",  "ФХЦЧ",  "ШЩЪЫ",
//            "ЬЭЮЯ",  "+*#",     " "
//};

const char *letters_map[12] = {
          "", ".,?!", "ABC",   "DEF",  "GHI",
              "JKL", "MNO",  "PQRS",  "TUV",
             "WXYZ", "+*#",     " "
```

```
};

const char *digits_map[12] = {
    "", "1", "2", "3", "4", "5", "6", "7", "8", "9", "\r\n", "0"
};

uint8_t i = 0;
uint16_t row_pins[4] = { GPIO_PINS_4, GPIO_PINS_5, GPIO_PINS_6, GPIO_PINS_7 };



uint8_t button_state = 0;
uint16_t button_count = 0;
uint8_t last_state = 0;

uint8_t key;

char str[20];

uint8_t last_key;
uint16_t timeout;



int main(void)
{
        system_clock_config();
        SysTick_Config(system_core_clock / 1000);

        keyboard_init();
```

```
	set_all_cols_high();
	uart_init();

	while(1) { }
}




void SysTick_Handler(void)
{
	key = keypad_scan();
	state_change_button();
}


void state_change_button(void)
{
	if (key)
	{
		if (key == last_state)
		{
			if (button_count < 5) // 5 ms
			{
				button_count++;
			}
			else if (button_state == 0)
			{
				timeout = 0;
```

```
                multi_tap(key);
                button_state = 1;
            }
            else if (button_count < HOLD_THRESHOLD)
            {
                button_count++;
            }
            else if (button_state == 1)
            {
                if (key == 10)
                {
                    uart_send_string("\b \b");
                    sprintf(str, "%s", digits_map[key]);
                }
                else
                    sprintf(str, "\b%s", digits_map[key]);
                uart_send_string(str);
                button_count = 1;
            }
        }
        last_state = key;
    }
    else
    {
        if (button_count > 0)
        {
            button_count--;
        }
        else if (button_state == 1)
```

```
            {
                button_state = 0;
            }
            else if (++timeout == MULTITAP_TIMEOUT)
            {
                last_key = 0;
            }
        }
}


void multi_tap (uint8_t k)
{
        if (k == 11)
        {
            sprintf(str, "%s", letters_map[k]);
            uart_send_string(str);
            last_key = 0;
        }
        else if (k == 12)
        {
            uart_send_string("\b \b"); //delete
            last_key = 0;
        }
        else if (last_key == k)
        {
            sprintf(str, "\b%c", letters_map[k][++i % 3]);
            uart_send_string(str);
        }
```

```
        else
        {
                uint8_t i = 0;
                sprintf(str, "%c", letters_map[k][i]);
                uart_send_string(str);
                last_key = k;
        }
}


void keyboard_init(void)
{
        // clock PORT A
        CRM->apb2en_bit.gpioaen = 1;

        // C3
        GPIOA->cfglr_bit.iomc1 = 0x01;
        GPIOA->cfglr_bit.iofc1 = 0x00;
        // C2
        GPIOA->cfglr_bit.iomc2 = 0x01;
        GPIOA->cfglr_bit.iofc2 = 0x00;
        // C1
        GPIOA->cfglr_bit.iomc3 = 0x01;
        GPIOA->cfglr_bit.iofc3 = 0x00;

        // R1
        GPIOA->cfglr_bit.iomc4 = 0x00;
        GPIOA->cfglr_bit.iofc4 = 0x02;
        GPIOA->odt_bit.odt4 = 1;
```

```
	// R2
	GPIOA->cfglr_bit.iomc5 = 0x00;
	GPIOA->cfglr_bit.iofc5 = 0x02;
	GPIOA->odt_bit.odt5 = 1;

	// R3
	GPIOA->cfglr_bit.iomc6 = 0x00;
	GPIOA->cfglr_bit.iofc6 = 0x02;
	GPIOA->odt_bit.odt6 = 1;

	// R4
	GPIOA->cfglr_bit.iomc7 = 0x00;
	GPIOA->cfglr_bit.iofc7 = 0x02;
	GPIOA->odt_bit.odt7 = 1;
}


void set_all_cols_high(void)
{
	GPIOA->odt |=	GPIO_PINS_1;
	GPIOA->odt |=	GPIO_PINS_2;
	GPIOA->odt |=	GPIO_PINS_3;
}


void set_col_low(uint8_t col)
{
	set_all_cols_high();
```

```
	if (col == 1)
		GPIOA->odt &=   ~GPIO_PINS_3;
	else if (col == 2)
		GPIOA->odt &=   ~GPIO_PINS_2;
	else if (col == 3)
		GPIOA->odt &=   ~GPIO_PINS_1;

	uint16_t t = 10000;
	while (t > 0) { t--; };
}


uint8_t keypad_scan(void)
{
	for (int col = 1; col < 4; col++) {
		set_col_low(col);

		for (int row = 0; row < 4; row++) {
			if (!gpio_input_data_bit_read(GPIOA, row_pins[row]))
				return row * 3 + col;
		}
	}
	return 0;
}


void uart_init (void)
{
	// Set USART1
```

```
	CRM->apb2en_bit.gpioaen = 1;

	// TX
	GPIOA->cfghr_bit.iomc9 = 0x01;
	GPIOA->cfghr_bit.iofc9 = 0x02;

	// RX
	GPIOA->cfghr_bit.iomc10 = 0x00;
	GPIOA->cfghr_bit.iofc10 = 0x02;
	GPIOA->odt_bit.odt10 = 1;

	CRM->apb2en_bit.usart1en = 1;
	USART1->ctrl1_bit.ten = 1;
	USART1->baudr_bit.div = 12500; // for 9600 baud
	USART1->ctrl1_bit.uen = 1;
}


void uart_send_string(const char *s)
{
	while (*s) {
		while (!(USART1->sts_bit.tdbe == 0))
		USART1->dt = *s++;
	}
}

#pragma once

#include "at32f403a_407.h"
```

```
#include "at32f403a_407_clock.h"

void state_change_button(void);


void keyboard_init(void);
void set_all_cols_high(void);
void set_col_low(uint8_t col);
uint8_t keypad_scan(void);



void uart_init (void);
void uart_send_string(const char *s) ;

void multi_tap (uint8_t k);
void hold_tap (uint8_t k);
```