

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ
И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ _____

РУКОВОДИТЕЛЬ:

(должность, учёная степень, звание) / _____ / _____ / _____
(подпись) (дата защиты) (инициалы, фамилия)
доцент , к.т.н А. А. Попов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ (ПРОЕКТУ)

Транслятор цифр в код Морзе.

ПО КУРСУ: «ПРОГРАММИРОВАНИЕ ВСТРОЕННЫХ ПРИЛОЖЕНИЙ»

РАБОТУ ВЫПОЛНИЛ (А) СТУДЕНТ (КА) ГРУППЫ: _____ / _____
(инициалы, фамилия)
Z6431 В. В. Чупринский

/ _____ / 13.05.2020
(подпись студента) (дата отчета)

Санкт-Петербург 2020

Содержание

Содержание	2
1. Задание на курсовое проектирование.....	3
2. Техническое задание на прибор	4
2.1. Основные требования.....	6
3. Схемы и алгоритмы работы.....	8
4. Тестирование.....	17
5. Заключение.....	20
6. Список использованной литературы:	21
Приложение 1. Код программы.....	22

Санкт-Петербургский
государственный
университет
аэрокосмического
приборостроения

1. Задание на курсовое проектирование.

Транслятор цифр в код Морзе.

Используя первые 10 линий PORTA реализовать ввод, через кнопки, созданные скриптом отладчика, цифр от 0 до 9 и вывод их в окно отладчика (Debug (printf) Viewer). В момент ввода цифры начинать её передачу в коде Морзе по линии PC13, ввод следующей цифры возможен только по окончанию передачи. Скорость передачи 60 знаков в минуту. Какие кнопки создать и какими функциями наделить решить самостоятельно.



Санкт-Петербургский
государственный
университет
аэрокосмического
приборостроения

2. Техническое задание на прибор

Поскольку курсовое проектирование выполняется удалённо, без отладочной платы, то необходимо выполнить проект на симуляторе в среде Keil uVision5. Таким образом, для имитации нажатия кнопок необходимо использовать скриптовый отладчик Keil.

Будем считать, что у нас подключено следующее периферийное оборудование: виртуальная клавиатура с цифрами 0-9, меняющие уровень сигнала во входах ПВВ PA0-PA9 (панель скриптового отладчика Keil), светодиод, зажигающийся в зависимости от состояния линии PC13 (будем наблюдать в Logic Analyzer) и экран с автоматической прокруткой для отображения текста (Debug (printf) Viewer).

Согласно заданию, скорость передачи должна соответствовать 60 знакам в минуту, однако особенностью Кода Морзе является то, что символы кодируются последовательностью сигналов переменной длины. Принцип кодирования азбуки Морзе исходит из того, что буквы, которые чаще употребляются в английском языке, кодируются более простыми сочетаниями точек и тире.

Так как нам необходимо передавать только цифры, рассмотрим коды Морзе для цифр от 0 до 9:

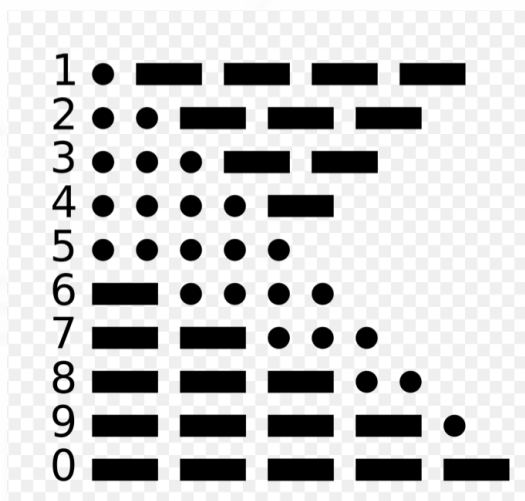


Рисунок 1. – Коды Морзе для цифр 0-9.

Согласно рекомендации МСЭ-R М.1677-1 действуют следующие правила по интервалу между сигналами и их продолжительности:

- 1) Тире равно трем точкам.
- 2) Интервал между сигналами, образующими одну букву, равен одной точке.

- 3) Интервал между двумя буквами равен трем точкам.
- 4) Интервал между двумя словами равен семи точкам.

Таким образом очевидно, что самый длительный символ (для цифр) для передачи кодом Морзе будет символ "0". Для передачи такого символа необходимо 22 минимальных интервалов равных по длительности передачи «точки».

Тогда для соблюдения скорости передачи 60 символов в минуту длительность сигнала «точка» должна составлять $1 \text{ сек} / 20 \approx 45 \text{ мс}$. Тогда длительность сигнала тире составит 135 мс.

Таким образом, скорость передачи будет не ниже 60 знаков в минуту.

Кодирование сигналов кода Морзе на линии PC13 необходимо осуществлять изменением состояния линии с 0 на 1 и обратно. 0 обозначает отсутствие передаваемого сигнала, 1 – наличие. Длительность нахождения линии в состоянии 1 должна обуславливать передаваемый сигнал (см. рис. 2)

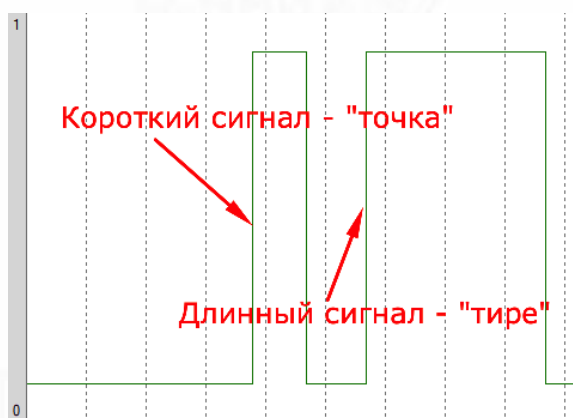


Рисунок 2. – Кодирование сигналов "точка", "тире".

Согласно заданию, ввод следующей цифры возможен только после окончания передачи, таким образом, все нажатия виртуальной клавиатуры с цифрами во время выполнения передачи должны отбрасываться и не обрабатываться.

Также, необходимо дублировать вывод введенной цифры в окно отладчика (Debug (printf) Viewer).

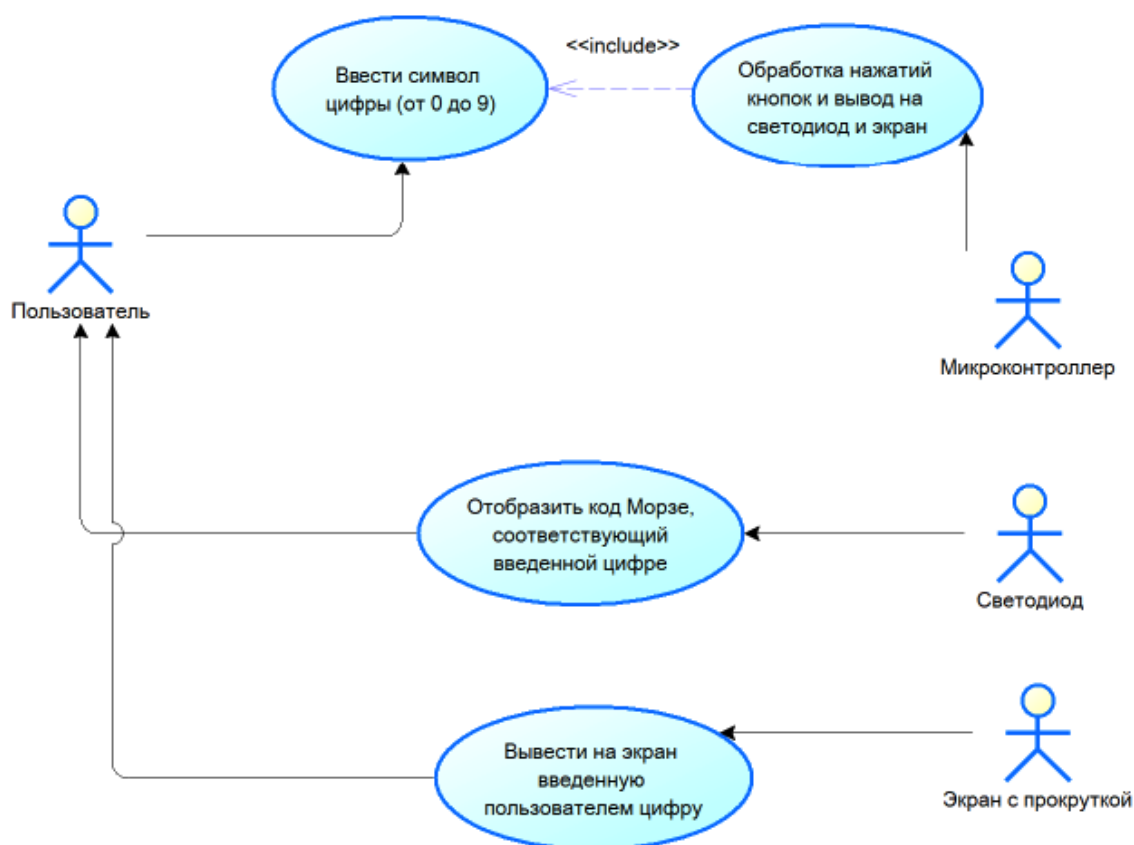


Рисунок 3. – Диаграмма использования.

Таблица 1 Назначение кнопок

Название кнопки	0	1	2	3	4	5	6	7	8	9
Цифра	0	1	2	3	4	5	6	7	8	9

2.1. Основные требования

Входы	Десять кнопок, соответствующих десяти цифрам от 0 до 9
Выходы	Индикация в виде светодиода, отображающего коды Морзе: короткая вспышка – «точка», длинная вспышка – «тире» Экран с автоматической прокруткой для отображения текста (введенной цифры)
Функции	При отсутствии ввода (изначальное состояние) светодиод должен быть выключен. При нажатии кнопки от 0 до 9 отображать светодиодом соответствующий код Морзе, после чего светодиод должен

	<p>переходит в начальное состояние (выключен).</p> <p>Коды Морзе:</p> <table border="1"> <tr><td>0</td><td>- - - - -</td></tr> <tr><td>1</td><td>. - - - -</td></tr> <tr><td>2</td><td>. . - - -</td></tr> <tr><td>3</td><td>. . . - -</td></tr> <tr><td>4</td><td>. . . . -</td></tr> <tr><td>5</td><td>.</td></tr> <tr><td>6</td><td>-</td></tr> <tr><td>7</td><td>- - . . .</td></tr> <tr><td>8</td><td>- - - . .</td></tr> <tr><td>9</td><td>- - - - .</td></tr> </table> <p>Нажатые кнопки должны выводиться на экран с прокруткой. После вывода цифры должна быть выведена пауза (отсутствие сигнала) длительностью 3 «точки» в качестве разделителя между знаками в слове. Пауза между элементами одного знака — одна точка.</p> <p>При нажатии кнопки во время отображения кода Морзе, соответствующего ранее введенной цифре, прибор должен игнорировать нажатия и продолжать обработку ввода только после завершения отображения кода Морзе.</p>	0	- - - - -	1	. - - - -	2	. . - - -	3	. . . - -	4 -	5	6	-	7	- - . . .	8	- - - . .	9	- - - - .
0	- - - - -																				
1	. - - - -																				
2	. . - - -																				
3	. . . - -																				
4 -																				
5																				
6	-																				
7	- - . . .																				
8	- - - . .																				
9	- - - - .																				
Особенности	Отсутствуют																				
Питание	Питание от сети переменного тока через стандартный блок питания (USB адаптер).																				
Размеры и вес	Достаточно маленькие, чтобы использовать на рабочем столе																				
Стоимость производства	Стоимость отладочной платы STM32F103C8 составляет 130 рублей. Стоимость 10-кнопочной клавиатуры – 120 рублей. Стоимость экрана – 200 рублей. Светодиод, провода и корпус – 100 рублей. Сборка и тестирование -50 рублей. Таким образом стоимость производства изделия должна составить не более 600 рублей.																				

3. Схемы и алгоритмы работы

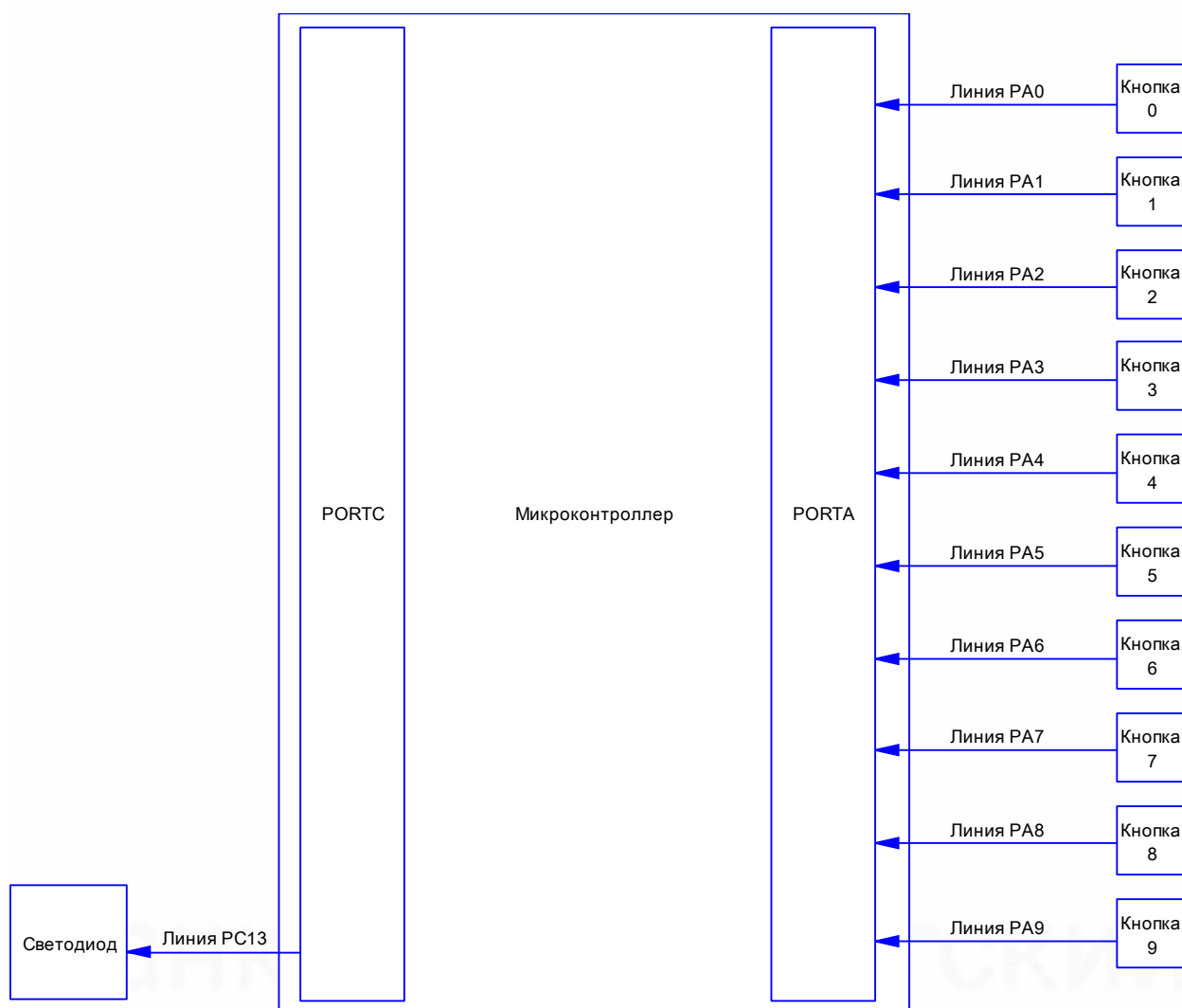


Рисунок 4. – Структурная схема подключение периферийных устройств к МК.

Для настройки работы портов PORTA, PORTC используется регистр RCC_APB2ENR (APB2 peripheral clock enable register).

Для настройки входов с линий PA0-PA7 используется регистр GPIOA_CRL (Port configuration register low).

Для настройки входов с линий PA8-PA9 используется регистр GPIOA_CRH (Port configuration register high).

Для настройки внешних прерываний с линий PA0-PA9 используются регистры AFIO_EXTICR1 (External interrupt configuration register 1), AFIO_EXTICR2 (External interrupt configuration register 2), AFIO_EXTICR3 (External interrupt configuration register 3).

Для установки начальных значений линий PA0-PA9 используется регистр GPIOA_ODR (Port output data register).

Для настройки выхода с линии PC13 используется регистр GPIOC_CRH (Port configuration register high).

Для настройки прерываний используются регистры EXTI_FTSR (Falling trigger selection register) и EXTI_IMR (Interrupt mask register).

Для установки значения линий PC13 используется регистр GPIOC_ODR (Port output data register).

Для работы системы будем использовать модель прерываний, поскольку модель опроса состояний усложнит программу.

Изначально, после сброса, система находится в неинициализированном состоянии. Далее происходит автоматическая инициализация, и система переходит в состоянии ожидания поступления внешнего сигнала (см. рис. 5).

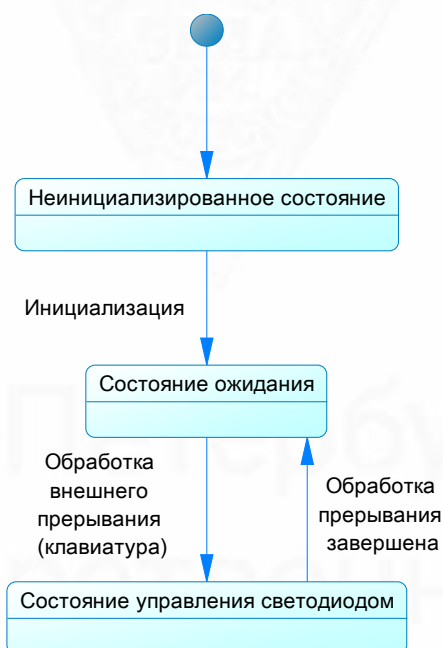
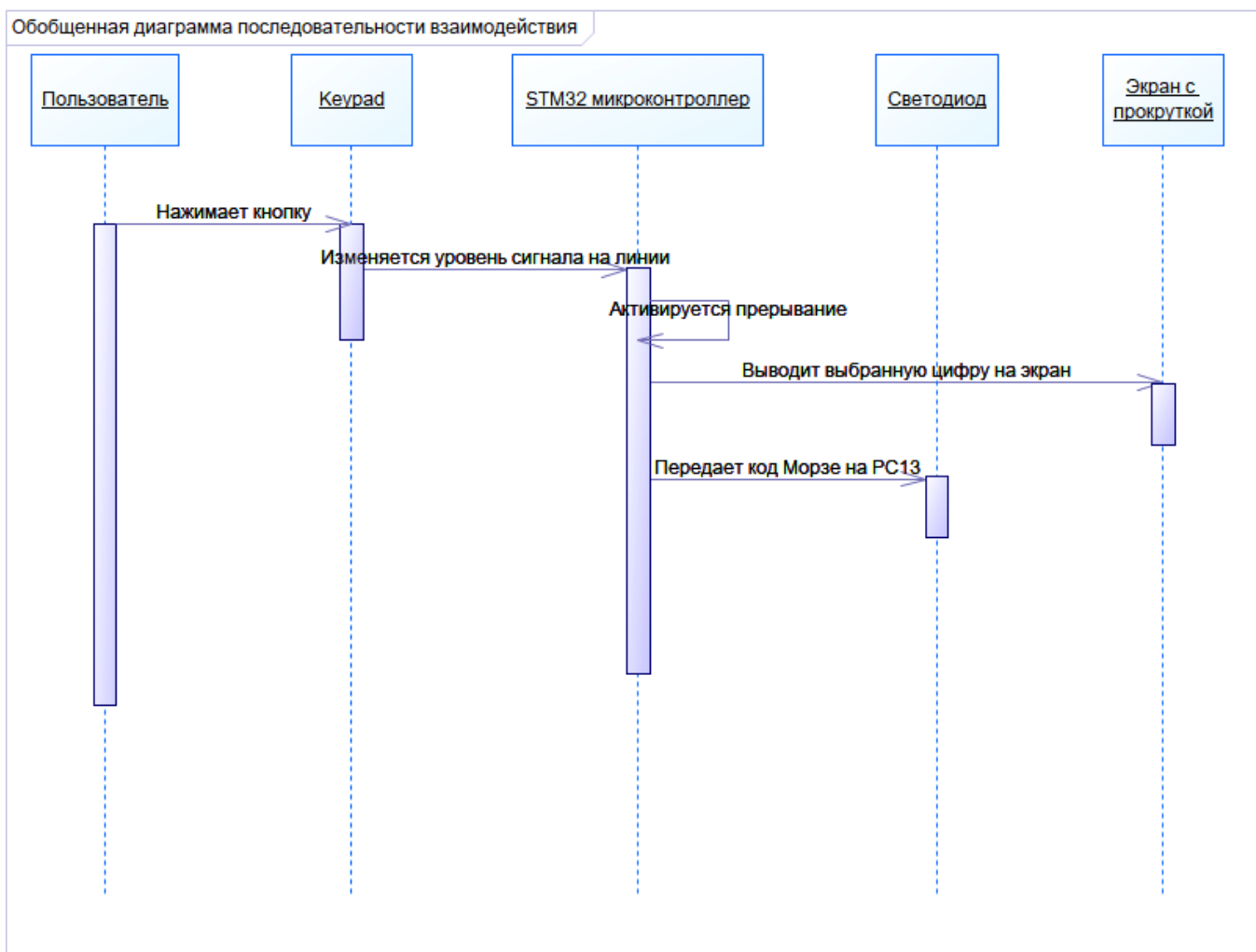


Рисунок 5. – Диаграмма состояний системы.

При поступлении внешнего сигнала (ввод с клавиатуры) система переходит в состояние управления светодиодом, в котором происходит вывод сигнала азбуки Морзе на внешний светодиод. После завершения вывода сигнала система автоматически переходит в состояние ожидания.

Рассмотрим работу системы на обобщенной диаграмме последовательности взаимодействия:



Выводы по диаграмме:

1) Будем использовать прерывания для инициации обработки нажатия кнопок. Каждая кнопка (0-9) будет присоединена к соответствующей линии (PA0 – PA9). Система прерываний микроконтроллера STM32F103C8 позволяет адресовать до 20 прерываний от входных линий, что позволит создать прерывание для каждой кнопки.

2) Необходим таймер для отсчета времени с целью генерации сигналов (точка/тире/пауза) заданной длительности. Для этого будем использовать системный таймер, счетчик задержки и обработчик прерываний системного таймера.

3) Для работы таймера необходима настройка прерываний системного таймера SysTick_IRQn и прерываний обработки ввода, при этом для поддержания необходимой длительности генерируемых сигналов приоритет системного таймера должна быть выше приоритета обработчиков ввода.

Детализируем состояния системы на диаграмме активности функции main:

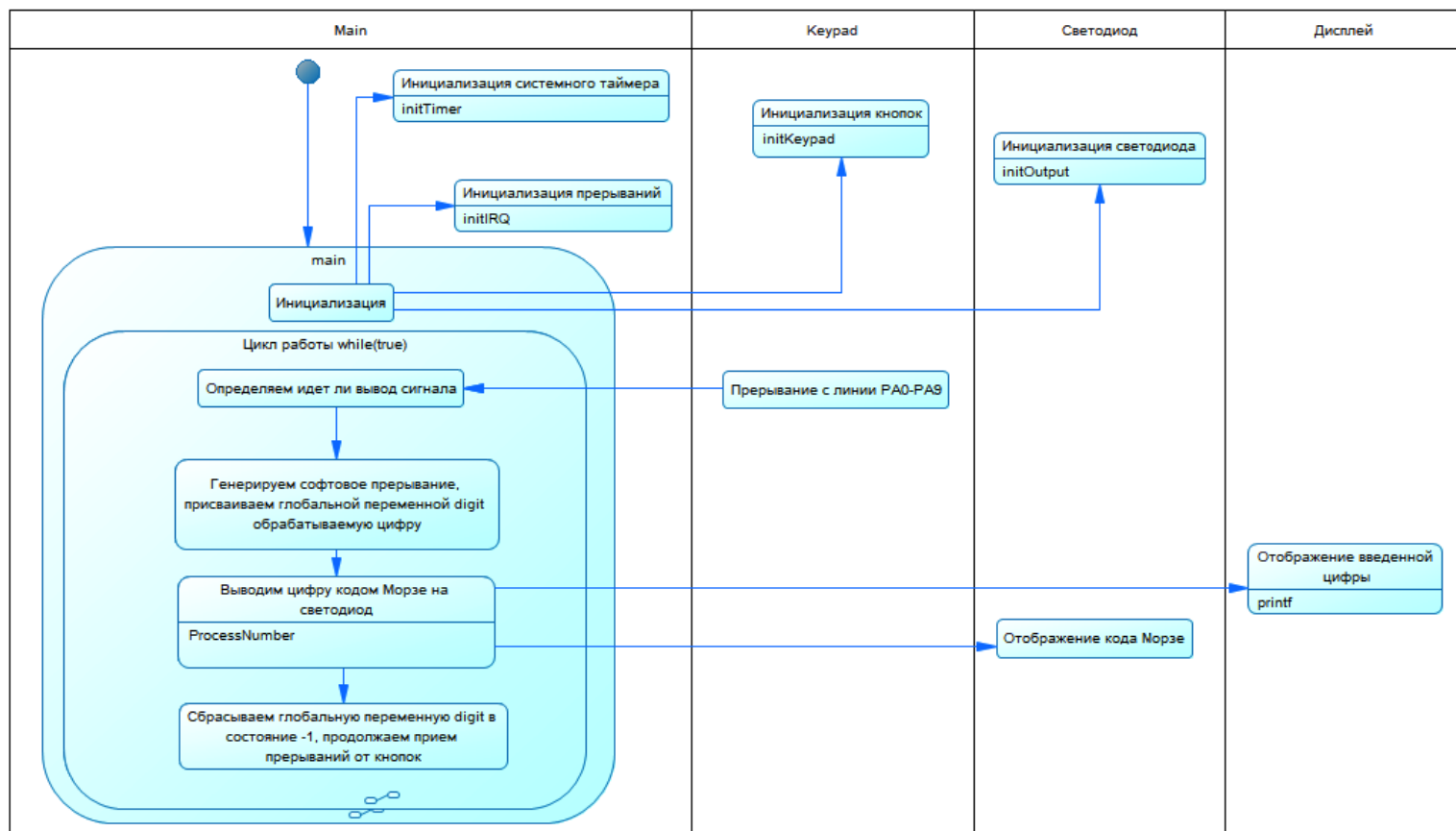


Рисунок 6. – Диаграмма активности функции main.

Отсюда следует выделить следующие основные управляющие структуры:

- 1) Функция main() с бесконечным циклом.
- 2) Функция initTimer(), реализующая инициализацию системного таймера.
- 3) Функция initKeypad(), реализующая инициализацию кнопок (разрешение работы порта GPIO A, настройка линий PA0-PA9, настройка EXTI).
- 4) Функция initOutput(), реализующая инициализацию светодиода (линия PC13).
- 5) Функция initIRQ (), реализующая настройку прерываний.

Пример функции main:

```
int main() {
    initTimer();
    initKeypad();
    initOutput();
    initIRQ();
    while(1){}
}
```

Инициализация кнопок:

```
void initKeypad() {
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN; // разрешаем работу GPIO A
    // обнуляем значения регистров
    GPIOA->CRL = 0;
    GPIOA->CRH = 0;
    //PA0-PA9 Input, Pull up
    SET_BIT(GPIOA->CRL, GPIO_CRL_CNF0_1 | GPIO_CRL_CNF1_1 | GPIO_CRL_CNF2_1
            | GPIO_CRL_CNF3_1 | GPIO_CRL_CNF4_1 | GPIO_CRL_CNF5_1 | GPIO_CRL_CNF6_1 |
            GPIO_CRL_CNF7_1);
}
```

```

SET_BIT(GPIOA->CRH, GPIO_CRH_CNF8_1 | GPIO_CRH_CNF9_1);
// pull up
SET_BIT(GPIOA->ODR, GPIO_ODR_ODR0 | GPIO_ODR_ODR1 | GPIO_ODR_ODR2 |
        GPIO_ODR_ODR3 | GPIO_ODR_ODR4 | GPIO_ODR_ODR5 | GPIO_ODR_ODR6 | GPIO_ODR_ODR7
        | GPIO_ODR_ODR8 | GPIO_ODR_ODR9);
// выбираем в качестве внешних входов EXTI линии:
// EXTIn = PAn
AFIO->EXTICR[0] = AFIO_EXTICR1_EXTI0_PA | AFIO_EXTICR1_EXTI1_PA |
AFIO_EXTICR1_EXTI2_PA | AFIO_EXTICR1_EXTI3_PA;
AFIO->EXTICR[1] = AFIO_EXTICR2_EXTI4_PA | AFIO_EXTICR2_EXTI5_PA |
AFIO_EXTICR2_EXTI6_PA | AFIO_EXTICR2_EXTI7_PA;
AFIO->EXTICR[3] = AFIO_EXTICR3_EXTI8_PA | AFIO_EXTICR3_EXTI9_PA;
}

```

Инициализация светодиода:

```

void initOutput() {
    RCC->APB2ENR |= RCC_APB2ENR_IOPCEN; // разрешаем работу GPIO C
    // PC13, Output mode, max speed 50 MHz, General purpose output push-pull
    GPIOC->CRH &= ~(GPIO_CRH_MODE13 | GPIO_CRH_CNF13);
    SET_BIT(GPIOC->CRH, GPIO_CRH_MODE13);
}

```

Инициализация прерываний:

```

void initIRQ() {
    // прерывание на спад сигнала
    SET_BIT(EXTI->FTSR, EXTI_FTSR_TR0 | EXTI_FTSR_TR1 | EXTI_FTSR_TR2 |
            EXTI_FTSR_TR3 | EXTI_FTSR_TR4 | EXTI_FTSR_TR5 | EXTI_FTSR_TR6 | EXTI_FTSR_TR7
            | EXTI_FTSR_TR8 | EXTI_FTSR_TR9);
    // разрешаем прерывания внешних линий 0-9, 10 - софтовое прерывание
    SET_BIT(EXTI->IMR, EXTI_IMR_MR0 | EXTI_IMR_MR1 | EXTI_IMR_MR2 |
            EXTI_IMR_MR3 | EXTI_IMR_MR4 | EXTI_IMR_MR5 | EXTI_IMR_MR6 | EXTI_IMR_MR7 |
            EXTI_IMR_MR8 | EXTI_IMR_MR9 | EXTI_IMR_MR10);
    NVIC_EnableIRQ(EXTI0_IRQn);
    NVIC_EnableIRQ(EXTI1_IRQn);
    NVIC_EnableIRQ(EXTI2_IRQn);
    NVIC_EnableIRQ(EXTI3_IRQn);
    NVIC_EnableIRQ(EXTI4_IRQn);
    NVIC_EnableIRQ(EXTI9_5_IRQn);
    NVIC_EnableIRQ(EXTI15_10_IRQn);
    NVIC_SetPriority(EXTI0_IRQn, 7);
    NVIC_SetPriority(EXTI1_IRQn, 7);
    NVIC_SetPriority(EXTI2_IRQn, 7);
    NVIC_SetPriority(EXTI3_IRQn, 7);
    NVIC_SetPriority(EXTI4_IRQn, 7);
    NVIC_SetPriority(EXTI9_5_IRQn, 7);
    NVIC_SetPriority(EXTI15_10_IRQn, 15);
}

```

Рассмотрим алгоритм обработки прерывания с линий PA0 – PA9 на рисунке 7.

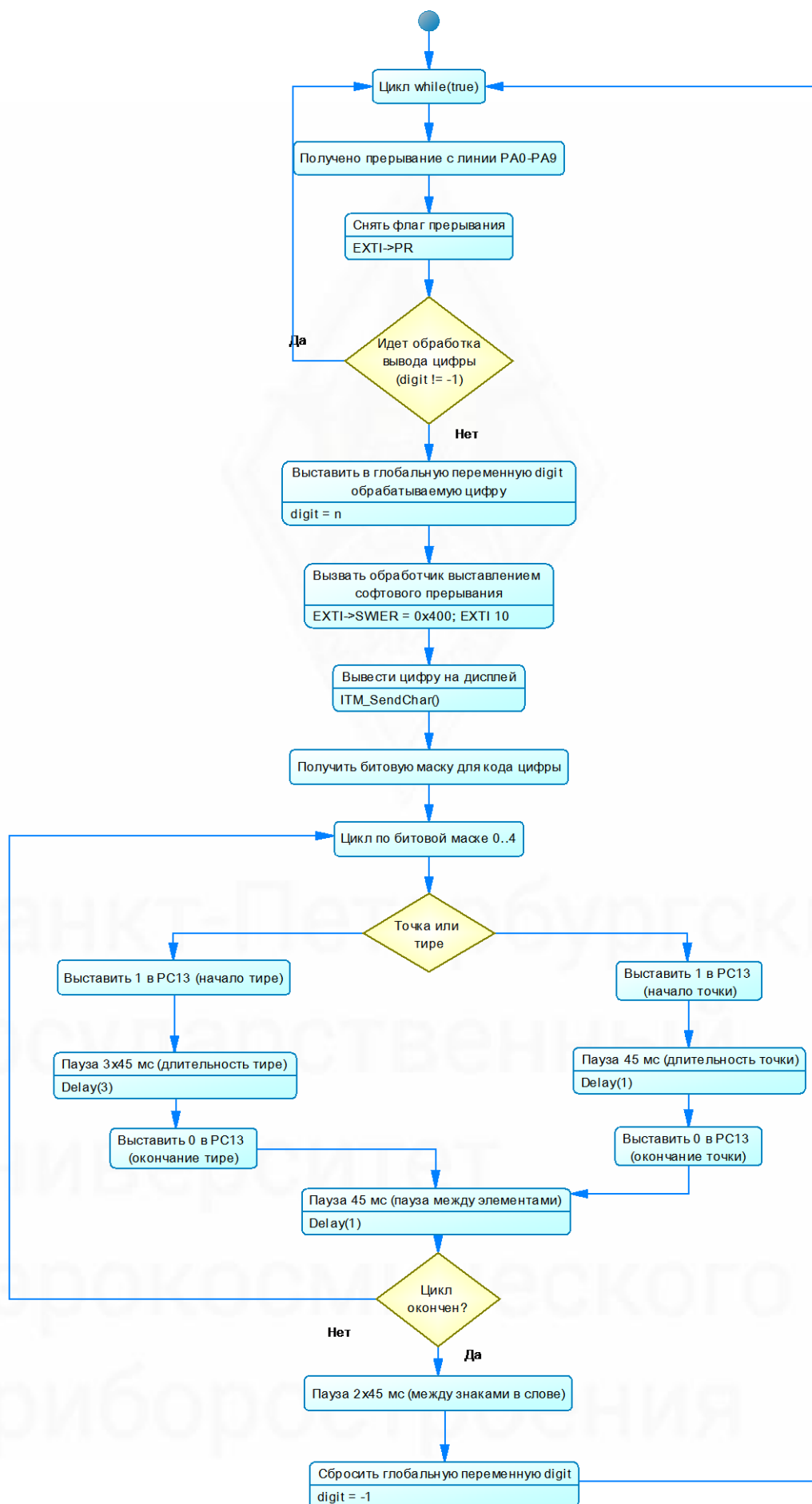


Рисунок 7. – Алгоритм работы функций обработки ввода.

Пример обработчика прерываний с клавиатуры:

```
void EXTI0_IRQHandler(void)
{
    EXTI->PR = EXTI_PR_PR0;
    if (digit == -1) {
        digit = 0;
        // вызываем обработчик софтовым прерыванием
        EXTI->SWIER = 0x400;
    }
}
```

Для того, чтобы при поступлении ввода с кнопок во время вывода цифры в коде Морзе не происходил запуск нового цикла обработки, выставим приоритет обработчика софтового прерывания EXTI10 ниже, чем приоритет обработчика прерываний с кнопок EXTI0-9. Таким образом, если идет цикл вывода кода Морзе, обработчик прерывания кнопок определит, что идет обработка с помощью глобальной переменной digit и обработка ввода будет прервана, что требуется по условиям задания.

Пример обработчика софтового прерывания:

```
// софтовое прерывание вызывает генератор Морзе
void EXTI15_10_IRQHandler(void)
{
    uint32_t pending = EXTI->PR;
    if (pending & (1 << 10)) {
        EXTI->PR = 1 << 10;
        ProcessNumber(digit);
        digit = -1;
    }
}
```

Также рассмотрим алгоритм работы функции задержки Delay.

Системный таймер изначально находится в неинициализированном состоянии. После инициализации он переходит в рабочее состояние и начинает генерировать системные прерывания, обрабатываемые функцией SysTick_Handler.

Инициализация таймера:

```
void initTimer() {
    SystemCoreClockUpdate();
    // прерывание каждые 1мс
    if (SysTick_Config(SystemCoreClock / 1000))
    {
        /* Capture error */
        while (1);
    }
    NVIC_SetPriority(SysTick_IRQn, 4);
}
```

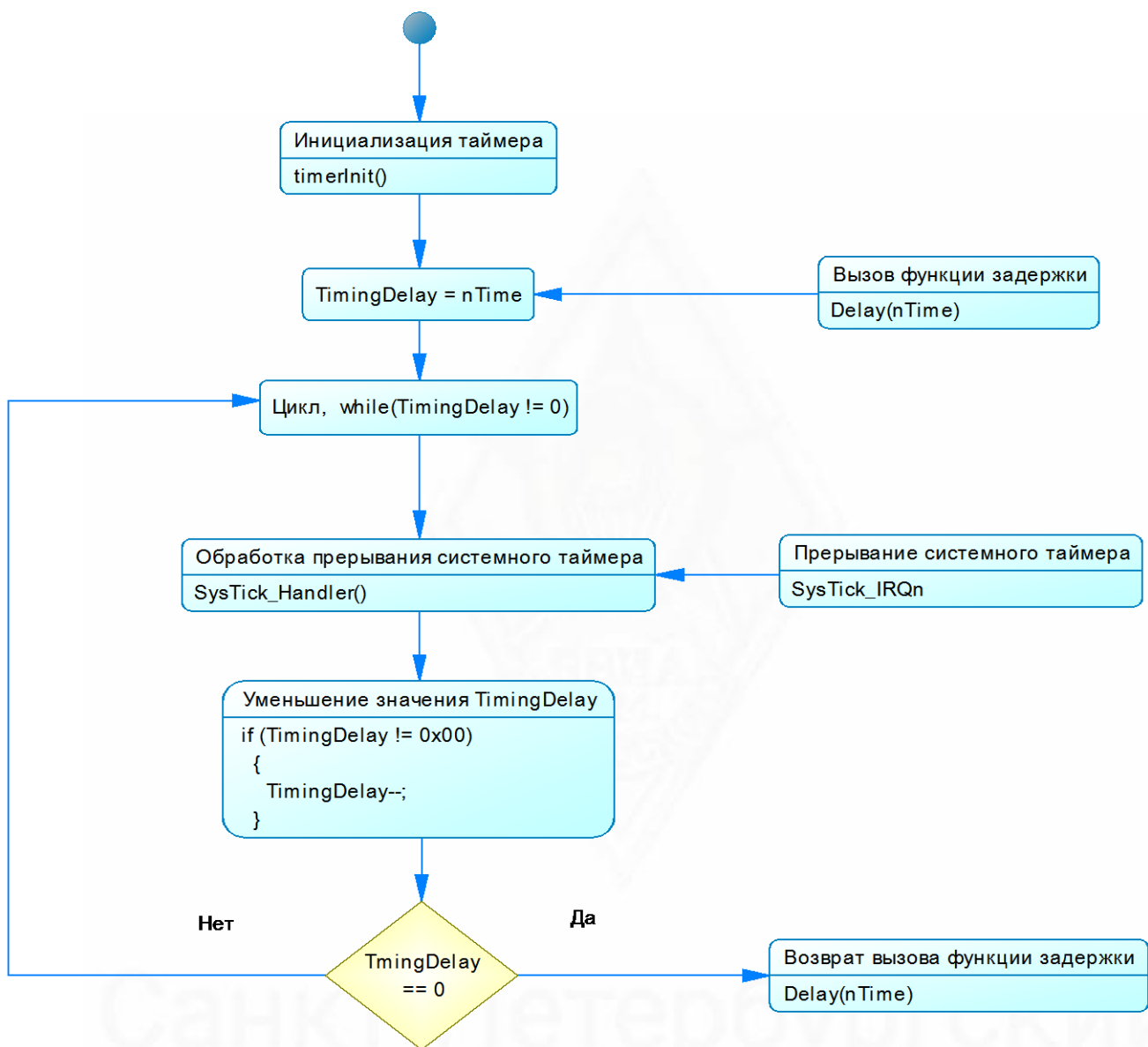


Рисунок 8. – Алгоритм работы функции задержки.

После инициализации значение переменной `TimingDelay == 0`. При вызове функции `Delay` значение переменной устанавливается в необходимое, после этого функция ожидает обнуления этой переменной. Прерывание системного таймера, происходящее каждую миллисекунду вызывает обработчик `SysTick_Handler`, который при каждом вызове уменьшает значение `TimingDelay` на единицу. После того, как значение `TimingDelay` становится равным 0, происходит возврат из функции `Delay`.

Пример кода обработчика:

```

// выставаем задержку nTime мс
void Delay(volatile uint32_t nTime)
{
    TimingDelay = nTime;
    while(TimingDelay != 0);
}
  
```

```

// уменьшаем счетчик задержки
void TimingDelay_Decrement(void)
{
    if (TimingDelay != 0x00)
    {
        TimingDelay--;
    }
}

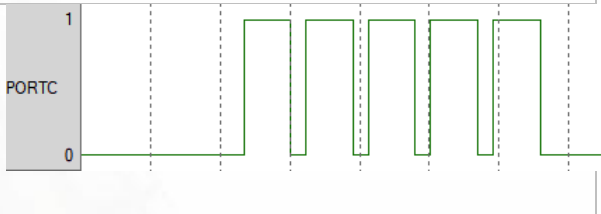
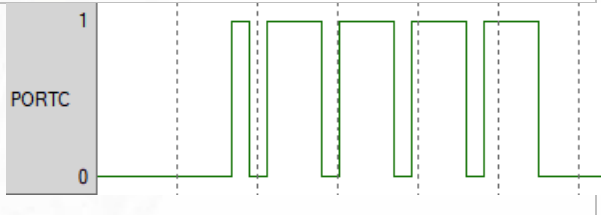
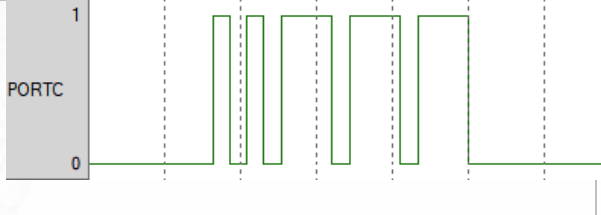
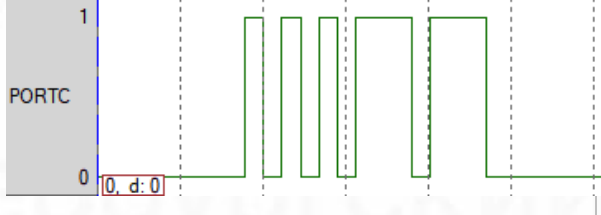
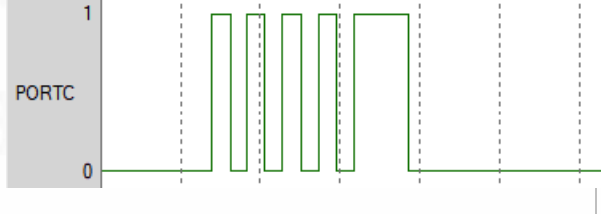
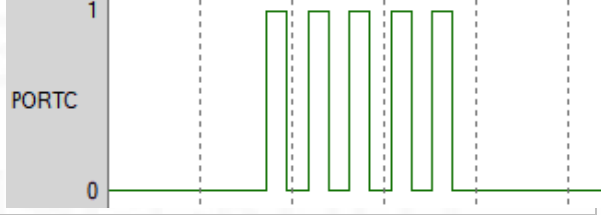
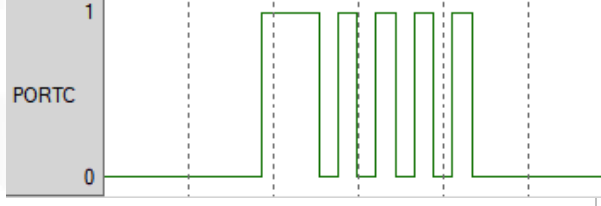
// обработчик прерывание системного таймера
void SysTick_Handler(void)
{
    TimingDelay_Decrement();
}
...
// прерывание каждые 1мс
if (SysTick_Config(SystemCoreClock / 1000))
{
    /* Capture error */
    while (1);
}
...

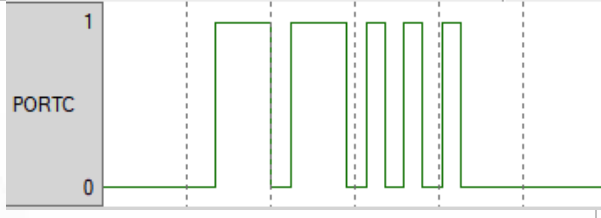
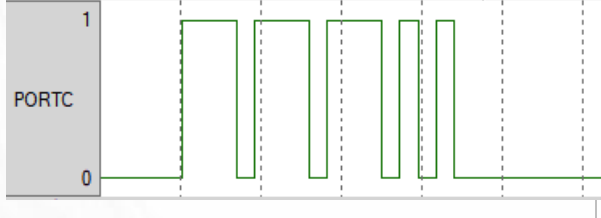
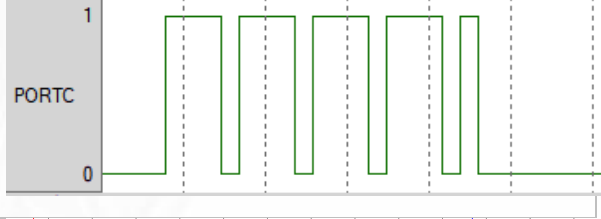
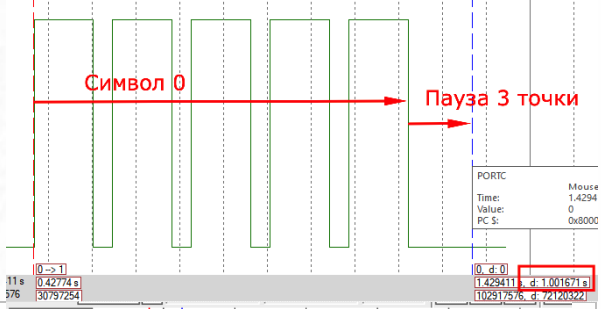
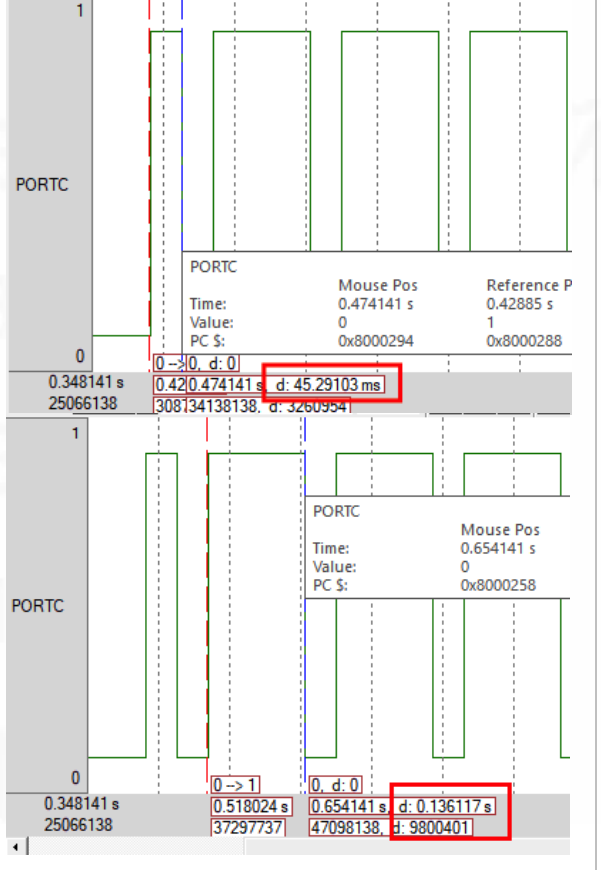
```

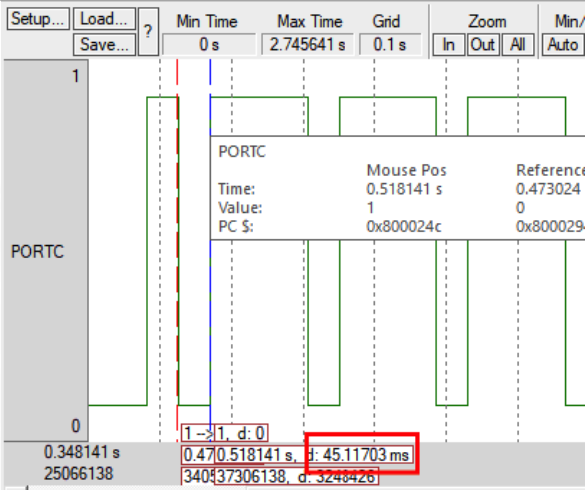
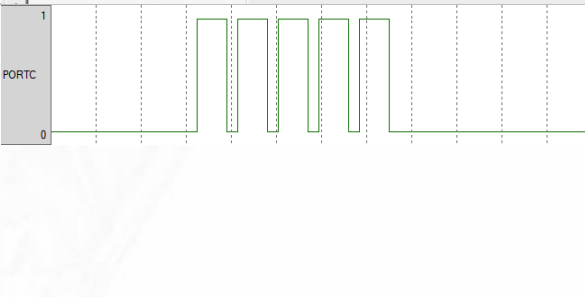
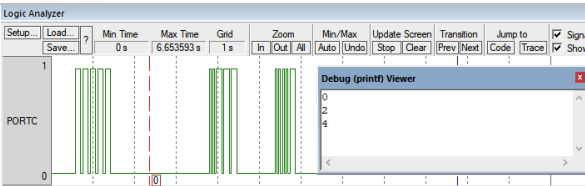
Санкт-Петербургский
государственный
университет
аэрокосмического
приборостроения

4. Тестирование

Тесты для контроля соответствия прибора техническому заданию

№	Описание	Ожидаемый результат	Фактический результат
	При нажатии на кнопку 0 должен выводиться соответствующий код Морзе на светодиод	Выводится код Морзе — — — — —	
	При нажатии на кнопку 1 должен выводиться соответствующий код Морзе на светодиод	Выводится код Морзе · — — — —	
	При нажатии на кнопку 2 должен выводиться соответствующий код Морзе · · — — —	Выводится код Морзе · · — — —	
	При нажатии на кнопку 3 должен выводиться соответствующий код Морзе · · · — —	Выводится код Морзе · · · — —	
	При нажатии на кнопку 4 должен выводиться соответствующий код Морзе · · · · —	Выводится код Морзе · · · · —	
	При нажатии на кнопку 5 должен выводиться соответствующий код Морзе · · · · ·	Выводится код Морзе · · · · ·	
	При нажатии на кнопку 6 должен выводиться соответствующий код Морзе — · · · ·	Выводится код Морзе — · · · ·	

При нажатии на кнопку 7 должен выводиться соответствующий код Морзе на светодиод	Выводится код Морзе —	
При нажатии на кнопку 8 должен выводиться соответствующий код Морзе на светодиод	Выводится код Морзе —	
При нажатии на кнопку 9 должен выводиться соответствующий код Морзе на светодиод	Выводится код Морзе —	
Скорость передачи для цифры 0 должна составлять 60 символов в минуту	Длительность передачи цифры 0 – 1000 мс	
Длительность тире должна составлять 3 длительности точки	Длительность точки = 45 мс, длительность тире = 135 мс	

	Пауза между элементами одного знака – одна точка	Длительность паузы = 45 мс	
	Ввод следующей цифры возможен только по окончанию передачи. Будем нажимать кнопку во время передачи символа	Выводится код только одного символа	
	Нажатые кнопки должны выводиться на экран с прокруткой.	При нажатии кнопки в окно выводятся соответствующая цифра	

Санкт-Петербургский
государственный
университет
аэрокосмического
приборостроения

5. Заключение

В результате выполнения данного проекта было осуществлено проектирование устройства на базе микроконтроллера. Были получены навыки в области архитектуры и программного обеспечения встроенных систем, а также знания о структуре, функциях и основах программирования микроконтроллеров, позволяющих решать вопросы анализа функционирования программного обеспечения встраиваемых систем.



Санкт-Петербургский
государственный
университет
аэрокосмического
приборостроения

6. Список использованной литературы:

1. Микроконтроллеры. Разработка встраиваемых приложений: учебное пособие / А.Е. Васильев; С.-Петербургский государственный политехнический ун-т. - СПб. : Изд-во СПбГПУ, 2003. - 211 с.
2. The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors. Third Edition. Joseph Yiu. ARM Ltd., Cambridge, UK. [электронный ресурс] // URL: <https://www.pdfdrive.com/the-definitive-guide-to-arm-cortex-m3-and-cortex-m4-processors-e187111520.html> (дата обращения 12.05.2020).
3. Джозеф Ю. Ядро Cortex-M3 компании ARM. Полное руководство. 2012. ISBN: 978- 5-94120-243-0. [электронный ресурс] // URL: <https://b-ok.xyz/book/2373589/b5c3ad> (дата обращения 12.05.2020).
4. RM0008. Reference manual STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs [электронный ресурс] // URL: https://www.st.com/resource/en/reference_manual/cd00171190-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf (дата обращения 12.05.2020).
5. Datasheet STM32F103x8 STM32F103xB Medium-density performance line ARM®-based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. Interfaces. [электронный ресурс] // URL: <https://www.st.com/resource/en/datasheet/stm32f103c8.pdf> (дата обращения 12.05.2020).
6. Мартин М. Инсайдерское руководство по STM32 [электронный ресурс] // URL: <https://istarik.ru/file/STM32.pdf> (дата обращения 12.05.2020).
7. Рекомендация МСЭ-R М.1677-1 Международный код Морзе. Международный союз электросвязи 2009 [электронный ресурс] //URL: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.1677-1-200910-I!!PDF-R.pdf (дата обращения 12.05.2020)

Приложение 1. Код программы

```
#include "RTE_Components.h"
#include CMSIS_device_header
#include <stdio.h>
// задержка таймера
static volatile uint32_t TimingDelay;
// цифра для вывода
static volatile int8_t digit = -1;
// длительность точки - 45 мс
const uint32_t dotDurationMs = 45;
// коды Морзе для цифр от 0 до 9
// 0 - точка, 1 - тире, RTL, 5 значимых позиций
const int8_t morseCodes[] = {
    0x1F, // 00011111 - 0
    0x1E, // 00011110 - 1
    0x1C, // 00011100 - 2
    0x18, // 00011000 - 3
    0x10, // 00010000 - 4
    0x00, // 00000000 - 5
    0x01, // 00000001 - 6
    0x03, // 00000011 - 7
    0x07, // 00000111 - 8
    0x0F }; // 00001111 - 9

// выставяем задержку nTime мс
void Delay(volatile uint32_t nTime)
{
    TimingDelay = nTime;
    while(TimingDelay != 0);
}

// уменьшаем счетчик задержки
void TimingDelay_Decrement(void)
{
    if (TimingDelay != 0x00)
    {
        TimingDelay--;
    }
}

// обработчик прерывание системного таймера
void SysTick_Handler(void)
{
    TimingDelay_Decrement();
}

void initKeypad() {
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN; // разрешаем работу GPIO A
    // обнуляем значения регистров
    GPIOA->CRL = 0;
    GPIOA->CRH = 0;
    //PA0-PA9 Input, Pull up
    SET_BIT(GPIOA->CRL, GPIO_CRL_CNF0_1 | GPIO_CRL_CNF1_1 | GPIO_CRL_CNF2_1
        | GPIO_CRL_CNF3_1 | GPIO_CRL_CNF4_1 | GPIO_CRL_CNF5_1 | GPIO_CRL_CNF6_1 |
        GPIO_CRL_CNF7_1);
    SET_BIT(GPIOA->CRH, GPIO_CRH_CNF8_1 | GPIO_CRH_CNF9_1);
    // pull up
    SET_BIT(GPIOA->ODR, GPIO_ODR_ODR0 | GPIO_ODR_ODR1 | GPIO_ODR_ODR2 |
        GPIO_ODR_ODR3 | GPIO_ODR_ODR4 | GPIO_ODR_ODR5 | GPIO_ODR_ODR6 | GPIO_ODR_ODR7
        | GPIO_ODR_ODR8 | GPIO_ODR_ODR9);
    // выбираем в качестве внешних входов EXTI линии:
    // EXTIin = PAn
    AFIO->EXTICR[0] = AFIO_EXTICR1_EXTI0_PA | AFIO_EXTICR1_EXTI1_PA |
        AFIO_EXTICR1_EXTI2_PA | AFIO_EXTICR1_EXTI3_PA;
    AFIO->EXTICR[1] = AFIO_EXTICR2_EXTI4_PA | AFIO_EXTICR2_EXTI5_PA |
        AFIO_EXTICR2_EXTI6_PA | AFIO_EXTICR2_EXTI7_PA;
    AFIO->EXTICR[3] = AFIO_EXTICR3_EXTI8_PA | AFIO_EXTICR3_EXTI9_PA;
}

void initOutput() {
    RCC->APB2ENR |= RCC_APB2ENR_IOPCEN; // разрешаем работу GPIO C
    // PC13, Output mode, max speed 50 MHz, General purpose output push-pull
    GPIOC->CRH &= ~(GPIO_CRH_MODE13 | GPIO_CRH_CNF13);
    SET_BIT(GPIOC->CRH, GPIO_CRH_MODE13);
}
```

```

}

void initIRQ() {
    // прерывание на спад сигнала
    SET_BIT(EXTI->FTSR, EXTI_FTSR_TR0 | EXTI_FTSR_TR1 | EXTI_FTSR_TR2 |
            EXTI_FTSR_TR3 | EXTI_FTSR_TR4 | EXTI_FTSR_TR5 | EXTI_FTSR_TR6 | EXTI_FTSR_TR7
            | EXTI_FTSR_TR8 | EXTI_FTSR_TR9);
    // разрешаем прерывания внешних линий 0-9, 10 - софтовое прерывание
    SET_BIT(EXTI->IMR, EXTI_IMR_MR0 | EXTI_IMR_MR1 | EXTI_IMR_MR2 |
            EXTI_IMR_MR3 | EXTI_IMR_MR4 | EXTI_IMR_MR5 | EXTI_IMR_MR6 | EXTI_IMR_MR7 |
            EXTI_IMR_MR8 | EXTI_IMR_MR9 | EXTI_IMR_MR10);
    NVIC_EnableIRQ(EXTI0_IRQn);
    NVIC_EnableIRQ(EXTI1_IRQn);
    NVIC_EnableIRQ(EXTI2_IRQn);
    NVIC_EnableIRQ(EXTI3_IRQn);
    NVIC_EnableIRQ(EXTI4_IRQn);
    NVIC_EnableIRQ(EXTI9_5_IRQn);
    NVIC_EnableIRQ(EXTI15_10_IRQn);
    NVIC_SetPriority(EXTI0_IRQn, 7);
    NVIC_SetPriority(EXTI1_IRQn, 7);
    NVIC_SetPriority(EXTI2_IRQn, 7);
    NVIC_SetPriority(EXTI3_IRQn, 7);
    NVIC_SetPriority(EXTI4_IRQn, 7);
    NVIC_SetPriority(EXTI9_5_IRQn, 7);
    NVIC_SetPriority(EXTI15_10_IRQn, 15);
}

void initTimer() {
    SystemCoreClockUpdate();
    // прерывание каждые 1мс
    if (SysTick_Config(SystemCoreClock / 1000))
    {
        /* Capture error */
        while (1);
    }
    NVIC_SetPriority(SysTick_IRQn, 4);
}

int main() {
    initTimer();
    initKeypad();
    initOutput();
    initIRQ();
    while(1){}
}

void Wait(uint32_t nCycles) {
    Delay(nCycles * dotDurationMs);
}

void Dot() {
    GPIOC->ODR = 1 << 13;
    Wait(1); // точка
    GPIOC->ODR = 0 << 13;
    Wait(1); // пауза между элементами знака
}

void Dash() {
    GPIOC->ODR = 1 << 13;
    Wait(3); // тире, длительность - 3 точки
    GPIOC->ODR = 0 << 13;
    Wait(1); // пауза между элементами знака
}

// вывод цифры азбукой морзе и на экран
void ProcessNumber(int number)
{
    ITM_SendChar(0x30 + number);
    ITM_SendChar('\n');
    uint8_t i, code, bit;
    code = morseCodes[number];
    for (i = 0; i < 5; i++) {
        bit = (code & (1 << i)) >> i;
        if (bit == 0)
        {
            Dot();

```

```

        }
        else
        {
            Dash();
        }
    }
    Wait(2); //пауза между знаками в слове-3 точки (одна после элемента+2)
}

void EXTI0_IRQHandler(void)
{
    EXTI->PR = EXTI_PR_PR0;
    if (digit == -1) {
        digit = 0;
        // вызываем обработчик софтовым прерыванием
        EXTI->SWIER = 0x400;
    }
}

void EXTI1_IRQHandler(void)
{
    EXTI->PR = EXTI_PR_PR1;
    if (digit == -1) {
        digit = 1;
        // вызываем обработчик софтовым прерыванием
        EXTI->SWIER = 0x400;
    }
}

void EXTI2_IRQHandler(void)
{
    EXTI->PR = EXTI_PR_PR2;
    if (digit == -1) {
        digit = 2;
        // вызываем обработчик софтовым прерыванием
        EXTI->SWIER = 0x400;
    }
}

void EXTI3_IRQHandler(void)
{
    EXTI->PR = EXTI_PR_PR3;
    if (digit == -1) {
        digit = 3;
        // вызываем обработчик софтовым прерыванием
        EXTI->SWIER = 0x400;
    }
}

void EXTI4_IRQHandler(void)
{
    EXTI->PR = EXTI_PR_PR4;
    if (digit == -1) {
        digit = 4;
        // вызываем обработчик софтовым прерыванием
        EXTI->SWIER = 0x400;
    }
}

void EXTI9_5_IRQHandler(void)
{
    // определим нажатую кнопку
    uint32_t pending = EXTI->PR;
    uint8_t i;
    for (i = 5; i < 10; i++)
    {
        if (pending & (1 << i)) {
            EXTI->PR = 1 << i;
            if (digit == -1) {
                digit = i;
                // вызываем обработчик софтовым прерыванием
                EXTI->SWIER = 0x400;
            }
            return;
        }
    }
}
}

```



```
// софтовое прерывание вызывает генератор Морзе
void EXTI15_10_IRQHandler(void)
{
    uint32_t pending = EXTI->PR;
    if (pending & (1 << 10)) {
        EXTI->PR = 1 << 10;
        ProcessNumber(digit);
        digit = -1;
    }
}
```

Код отладчика Keil для имитации кнопок (скрипт ini файла):

```
PORTA &= ~0x3FF;
SIGNAL void toggle_A0_pin() {
    PORTA |= 0x01;
    swatch (0.01);
    PORTA ^= 0x01;}
SIGNAL void toggle_A1_pin() {
    PORTA |= 0x02;
    swatch (0.01);
    PORTA ^= 0x02;}
SIGNAL void toggle_A2_pin() {
    PORTA |= 0x04;
    swatch (0.01);
    PORTA ^= 0x04;}
SIGNAL void toggle_A3_pin() {
    PORTA |= 0x08;
    swatch (0.01);
    PORTA ^= 0x08;}
SIGNAL void toggle_A4_pin() {
    PORTA |= 0x10;
    swatch (0.01);
    PORTA ^= 0x10;}
SIGNAL void toggle_A5_pin() {
    PORTA |= 0x20;
    swatch (0.01);
    PORTA ^= 0x20;}
SIGNAL void toggle_A6_pin() {
    PORTA |= 0x40;
    swatch (0.01);
    PORTA ^= 0x40;}
SIGNAL void toggle_A7_pin() {
    PORTA |= 0x80;
    swatch (0.01);
    PORTA ^= 0x80;}
SIGNAL void toggle_A8_pin() {
    PORTA |= 0x100;
    swatch (0.01);
    PORTA ^= 0x100;}
SIGNAL void toggle_A9_pin() {
    PORTA |= 0x200;
    swatch (0.01);
    PORTA ^= 0x200;}
DEFINE BUTTON "Key0 PA0", "toggle_A0_pin()"
DEFINE BUTTON "Key1 PA1", "toggle_A1_pin()"
DEFINE BUTTON "Key2 PA2", "toggle_A2_pin()"
DEFINE BUTTON "Key3 PA3", "toggle_A3_pin()"
DEFINE BUTTON "Key4 PA4", "toggle_A4_pin()"
DEFINE BUTTON "Key5 PA5", "toggle_A5_pin()"
DEFINE BUTTON "Key6 PA6", "toggle_A6_pin()"
DEFINE BUTTON "Key7 PA7", "toggle_A7_pin()"
DEFINE BUTTON "Key8 PA8", "toggle_A8_pin()"
DEFINE BUTTON "Key9 PA9", "toggle_A9_pin()"
```