МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Институт информатики и кибернетики

Кафедра информационных систем и технологий

КУРСОВОЙ ПРОЕКТ ПО ДИСЦИПЛИНЕ

«Микропроцессорные средства и системы»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

«Проектирование микропроцессорных устройств

на базе однокристальных микроконтроллеров»

|  |  |
| --- | --- |
| Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | И.И. Алеев |
|  |  |
| Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Д.А. Ворох |
|  |  |

САМАРА 2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования «Самарский национальный исследовательский   
университет имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра информационных систем и технологий

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

Обучающемуся Алееву Ибрагиму Ильясовичу группы 6404-090301D

Тема проекта: ***«***Проектирование микропроцессорных устройств

на базе однокристальных микроконтроллеров» в соответствии заданием:

Разработать контроллер, выполняющий следующие функции:

* Многоканальное измерение амплитуды в диапазоне (0-5)В, количество каналов -8, погрешность измерения -0,05%;
* Источники измерительного сигнала запускаются одновременно по команде ОМК;
* По каждому каналу определить средне-арифметическое из 16 измерений;
* Порядок опроса каналов и номер выводимого на индикацию канала задается с клавиатуры и контролируется 7-сегментным индикатором;
* В момент окончания работы включить красный светодиод и данные передать по интерфейсу RS-232 на скорости 1200б/с, если пришел внешний запрос прерываний.

|  |  |  |
| --- | --- | --- |
| Планируемые результаты освоения образовательной программы (компетенции) | Планируемые результаты курсового проектирования | Содержание задания |
| ПК-1 Способен обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности  ПК-1.4. Проектирует информационные системы, разрабатывает программу работ по экспериментальной проверке ее эффективности | Знать: технические характеристики микроконтроллеров и инструментальных средств.  Уметь: проектировать устройства на микроконтроллерах.  Владеть: навыками  решения задач интегрирования в систему микропроцессорных устройств и их комплексной отладки с целью повышения эффективности | 1.Анализ полученного задания  2.Разработка структурной организации  3.Анализ алгоритмов обработки и управления  4.Разработка принципиальной электрической схемы  5.Выбор инструментальных средств и языка программирования  6. Разработка программы  7.Тестирование разработанного контроллера  8. Разработка инструкции по эксплуатации  9. Заключение |

Дата выдачи задания 20 сентября 2023 г.

Срок представления на кафедру пояснительной записки 24 декабря 2023 г.

Руководитель курсового проекта

доцент каф. ИСТ к.т.н. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д.А. Ворох

(подпись)

Задание принял к исполнению

обучающийся группы № 6404-090301D \_\_\_\_\_\_\_\_\_\_\_\_\_ И.И. Алеев

(подпись)

РЕФЕРАТ

**Пояснительная записка к курсовому проекту:** 27 стр., 20 рисунков, 1 таблица, 4 источника, 2 приложения.

МИКРОКОНТРОЛЛЕР, АМПЛИТУДА, АЦПБ СРЕДНЕ-АРИФМЕТИЧЕСКОЕБ 7-СЕГМЕНТНЫЙ ИНДИКАТОРБ КЛАВИАТУРАБ RS-232.

В соответствии с техническим заданием спроектирована система, основанная на однокристальном микропроцессоре семейства atmega – ATMEGA8535. Реализован алгоритм многоканального измерения амплитуды, обработки и передачи данных. Программное обеспечение было реализовано в среде разработки Microchip studio отладка произведена в среде Proteus 8.

Выполнена отладка и проверка схемотехники и программного обеспечения на работоспособность.

# ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧНИЙ, ЕДИНИЦ И ТЕРМИНОВ

МПС – микропроцессорная система;

МПУ – микропроцессорное устройство;

ОМК – однокристальный микроконтроллер;

ПБ – процессорный блок;

ПЗУ (ROM) – постоянное запоминающее устройство;

ОЗУ (RAM) – оперативно запоминающее устройство;

ГСА – граф-схема алгоритма;

Рг – регистр;

ШД – шина данных;

СТ – счетчик таймер.

СОДЕРЖАНИЕ

[ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧНИЙ, ЕДИНИЦ И ТЕРМИНОВ 5](#_Toc158849209)

[ВВЕДЕНИЕ 7](#_Toc158849210)

[СТРУКТУРНАЯ ОРГАНИЗАЦИЯ 8](#_Toc158849211)

[АЛГОРИТМЫ ОБРАБОТКИ И УПРАВЛЕНИЯ 9](#_Toc158849212)

[ПРИНЦИПИАЛЬНАЯ ЭЛЕКТРИЧЕСКАЯ СХЕМА 11](#_Toc158849213)

[1. Блок микропроцессора: 11](#_Toc158849214)

[2. Распределение выводов микропроцессора 12](#_Toc158849215)

[ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ 13](#_Toc158849216)

[1. Обоснование выбора языка программирования 13](#_Toc158849217)

[2. Обоснование выбора среды проектирования и набора инструментов 13](#_Toc158849218)

[3. Обработка внешних прерываний 13](#_Toc158849219)

[4. Настройка таймера 14](#_Toc158849220)

[АЛГОРИТМ ТЕСТИРОВАНИЯ 17](#_Toc158849221)

[ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ КОНТРОЛЛЕРА 23](#_Toc158849222)

[ЗАКЛЮЧЕНИЕ 24](#_Toc158849223)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 25](#_Toc158849224)

[ПРИЛОЖЕНИЕ А. ПРИНЦИПИАЛЬНАЯ ЭЛЕКТРИЧЕСКАЯ СХЕМА 26](#_Toc158849225)

[ПРИЛОЖЕНИЕ Б. СПЕЦИФИКАЦИЯ 27](#_Toc158849226)

# ВВЕДЕНИЕ

В рамках курсового проекта необходимо реализовать систему, предназначенную для измерения длительности импульса, обработки и передачи данных.

В процессе работы над проектом необходимо оформить структурную, принципиальную электрическую, алгоритмическую схему работы устройства, написать управляющее ПО и осуществить отладку устройства в симуляторе схем Proteus, произвести тестирование программы и создать пакет необходимой документации.

# СТРУКТУРНАЯ ОРГАНИЗАЦИЯ

Для реализации поставленной задачи необходимо иметь следующие блоки:

Микроконтроллер

Разъём подключения

Система электропитания

GPIO

PORT

RS-232

UART

LED

АЦП

Вход импульсов

ПБ

ОЗУ

ПЗУ

ОМК

Рисунок 1 – Структурная схема контролера

Для программной реализации на основе микроконтроллера он должен обладать следующим набором периферийных блоков:

* АЦП с разрядностью не менее 10 бит. Количество каналов – не менее 8 шт;
* Интерфейс UART, симплекс в режиме передачи - 1 шт
* Контакты портов ввода/вывода GPIO - 23 шт
* Интерфейс ISP для программирования и отладки
* Сигнал сброса

# АЛГОРИТМЫ ОБРАБОТКИ И УПРАВЛЕНИЯ

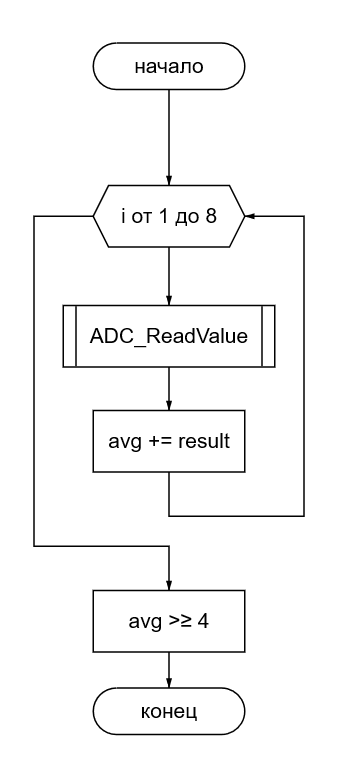


Рисунок 2 – Алгоритм обработки значений АЦП.

Код данного алгоритма приведён ниже:

*uint16\_t* ADC\_avg16(*uint8\_t* ch)

{

*uint8\_t* sampli = 0;

avg = 0;

for(sampli = 0; sampli < 16; sampli ++)

avg += ADC\_ReadChannel(ch);

avg >>= 4;

return avg;

}

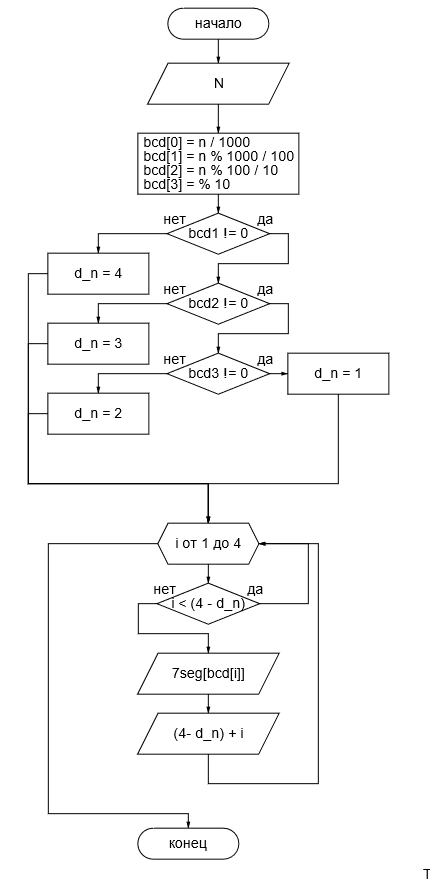


Рисунок 3 – Алгоритм вывода чисел

Данный алгоритм использует приведение чисел к 10 системе счисления и таблицу для вывода на семисегментные индикаторы. Код данного алгоритма приведён ниже:

void bcd\_print(*uint16\_t* n)

{

*uint8\_t* \_7seg[] = {

0b0111111, // 0

0b0000110, // 1

0b1011011, // 2

0b1001111, // 3

0b1100110, // 4

0b1101101, // 5

0b1111101, // 6

0b0000111, // 7

0b1111111, // 8

0b1101111 // 9

};

*uint8\_t* i = 0;

*uint8\_t* bcd[] = {

n / 1000,

(n % 1000) / 100,

(n % 100) / 10,

n % 10,

};

*uint8\_t* dn\_count = bcd[0] ? 4 : bcd[1] ? 3 : bcd[2] ? 2 : 1;

for (i = 0; i < 4 ; i ++)

{

if (i < (4 - dn\_count))

{

D\_PORT = 0;

continue;

}

D\_PORT = \_7seg[bcd[i]];

DC\_PORT &= ~(DC\_1\_MASK | DC\_2\_MASK);

*uint8\_t* dc\_i = (4 - dn\_count) + i;

DC\_PORT |= ( dc\_i & 1 ? DC\_1\_MASK : 0) | ( dc\_i & 2 ? DC\_2\_MASK : 0);

*\_delay\_ms*(70);

}

}

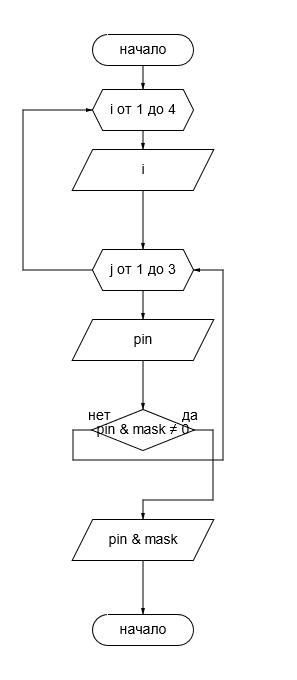


Рисунок 4 – Алгоритмы чтения с клавиатуры

С помощью данного алгоритма происходит проверка написанного.

key\_t getkey()

{

*uint8\_t* maskc[] = {

KP1\_MASK,б+6+++6

KP2\_MASK,

KP3\_MASK

};

*uint8\_t* maskr[] = {

KPA\_MASK,

KPB\_MASK,

KPC\_MASK,

KPD\_MASK

};

*uint8\_t* i, j;

for (i = 0; i < 4; i++)

{

KP\_PORT &= ~(

KPA\_MASK |

KPB\_MASK |

KPC\_MASK |

KPD\_MASK

);

KP\_PORT |= maskr[i];

*\_delay\_ms*(17);

for(j = 0; j < 3; j ++)

{

if (KP\_PIN & maskc[j])

return (key\_t)(i \* 3 + j);

}

}

return k\_none;

}

# ПРИНЦИПИАЛЬНАЯ ЭЛЕКТРИЧЕСКАЯ СХЕМА

Рисунок 5. Принципиальная электрическая схема

### Блок микропроцессора:

В ходе выполнения курсового проекта необходимо выбрать ОМК в соответствии с требованиями, которые налагаются содержанием технического задания, обладать функциональной полнотой, чтобы минимизировать число внешних компонентов, необходимых для реализации требуемого МПУ, иметь минимальное количество неиспользуемых линий ввода-вывода и минимальные энергопотребление.

Основой системы стал микропроцессор STM32F103C6 - микроконтроллер семейства STM32F1 от компании STMicroelectronics.

Основные характеристики STM32F103C6:

- ядро ARM Cortex-M3 с частотой до 72 МГц

- 32 Кбайт флэш-памяти

- 10 Кбайт ОЗУ.

- поддерживает интерфейсы: USB, CAN, SPI, I2C, UART и т.д.

- имеет встроенный DMA-контроллер

С помощью DMA-контроллера возможно быстро передавать данные между памятью и периферийными устройствами без участия процессора. STM32F103C6 - это доступный и эффективный микроконтроллер, который подходит для широкого спектра приложений, включая промышленность, автомобильную электронику, медицинскую технику и т.д.

### Распределение выводов микропроцессора

Начальное распределение выводов микропроцессора проводилось в среде разработки CubeIDE с использованием библиотеки HAL (Hardware Abstraction Layer).

Центральным блоком на схеме является ОМК STM32F103C6.

К контакту PA2 подключён вход внешнего прерывания.

К контактам PA4-PA7 подключена линия адреса для записи результата во внешнее ОЗУ.

К контакту PA8 подключён вход импульса;

К контакту PA10 подключён вход UART в режиме симплекса на приём;

К контактам PB1, PB2 подключены зелёный и красный индикаторы соответственно;

К контактам PB – PB5 подключены контрольные сигналы ОЗУ: CS, WR, RD соответственно;

К контактам PB6, PB7 подключены сигналы линии I2C: sda и scl соответственно;

К контактам P8 – PB15 подключены сигналы данных ОЗУ.

# ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

### Обоснование выбора языка программирования

Для разработки программы выбран язык Си, он является языком общего назначения. Обладает высокой переносимостью и широко используется в среде программирования. В отличие от низкоуровневого языка ассемблера, Си обеспечивает удобочитаемость, более высокую скорость разработки и упрощает её процесс.

### Обоснование выбора среды проектирования и набора инструментов

В качестве среды разработки выбран STM32CubeIDE - платформа разработки на C/C++ с функциями отладки для микроконтроллеров и микропроцессоров STM32. Он достаточно прост в освоении, имеет широкий функционал, предоставляет доступ к широкому набору библиотек периферийных устройств, это значительно ускоряет разработку и упрощает процесс реализации. это продвинутая

### Обработка внешних прерываний

Для включения режима внешних прерываний необходимо установить вывод микроконтроллера в соответствующий режим – генерирование прерываний при восходящем импульсе.

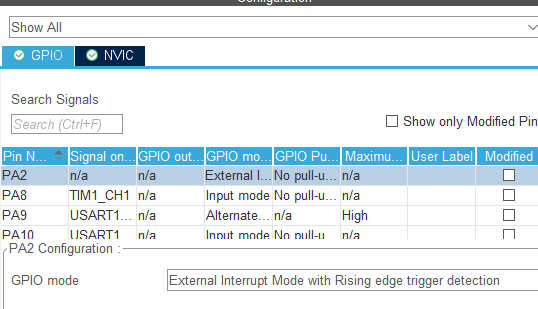


Рисунок 6 Конфигурация вывода в режиме прерываний по восходящему фронту

Далее необходимо разрешить выполнение прерываний

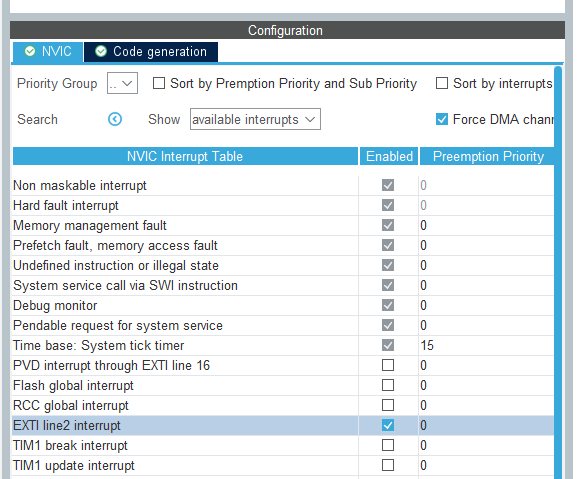


Рисунок 7 Разрешение генерации прерываний

После этого необходимо реализовать обработчик.

**void** **HAL\_GPIO\_EXTI\_Callback**(uint16\_t GPIO\_Pin)

{

}

### Настройка таймера

Для непосредственно считывания импульса необходимо использовать таймер с частотой более 10 МГц и кратной 10. Частота работы таймера настраивается в окне «Clock Configuration» путём изменения множителя «PLL», изменения частоты внешнего генератора, изменения коэффициента деления в различных узлах.

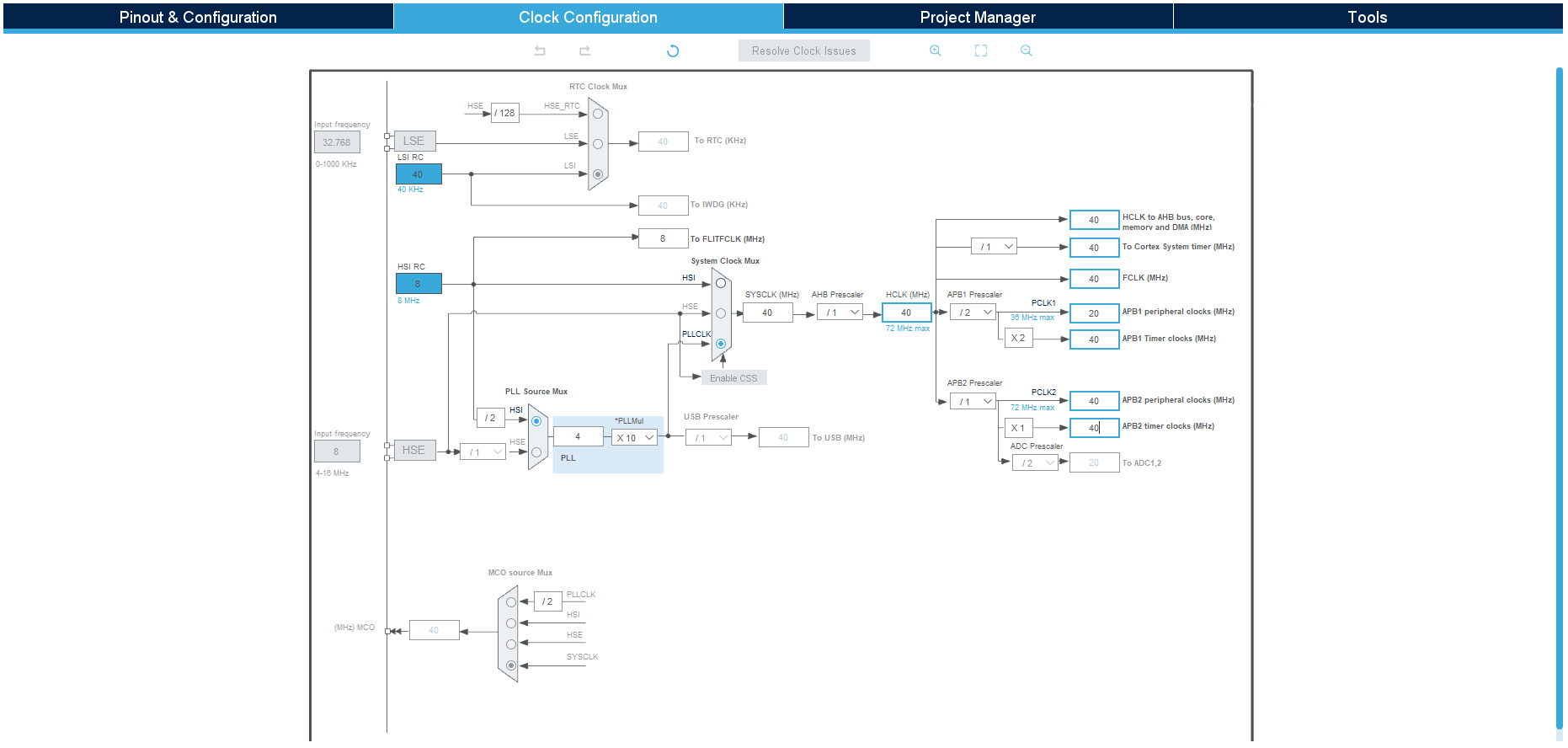


Рисунок 8 – Настройка тактирования подсистем ОМК.

Так же кроме тактирования необходимо произвести и другие настройки.

В первую очередь количество каналов и функции каналов, а так же применить тактирование, которое было настроено до этого момента.

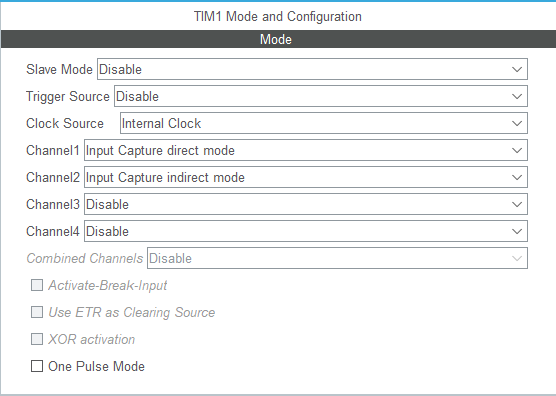


Рисунок 9 – Настройка режима работы таймера

Затем необходимо настроить параметры работы таймера в каждом режиме. Происходит это в окне ниже:

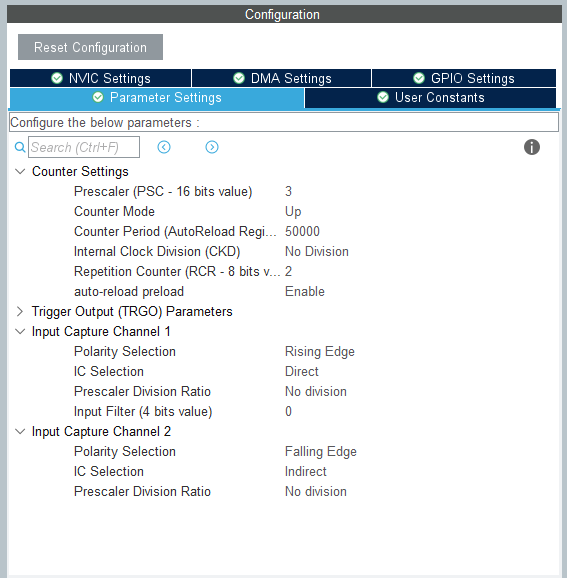


Рисунок 10 – Окно параметров работы таймера.

Некоторые важные пункты в этом окне:

* Prescaler – определяет на сколько делится частота, от которой работает таймер.
* Counter Period – определяет период, максимальное рабочее значение, в котором таймер работает.
* Repetition Counter – определяет количество переполнений после которого будет вызвано прерывание.
* Auto-reload preload – определяет, когда будет применено новое значение счётчика таймера.

АЛГОРИТМ ТЕСТИРОВАНИЯ

Краткий алгоритм действия для работы с программой:

1. Запустить среду Proteus, открыть файл проекта;
2. Запустить среду STM32CubeIDE, открыть файлы проекта, выполнить компиляцию;
3. В свойствах микроконтроллера указать бинарный файл программы в папке Debug;
4. Настроить нуль-терминал;
5. Выбрать в устройстве «COMPIM» один из созданных виртуальных COM-портов;
6. Подключить терминал к другому виртуальному com-порту, относящийся к той же паре, что и в предыдущем пункте;
7. Запустить симуляцию;
8. Отправить через терминал бинарные данные в формате «$XX$XX$XX$XX$YY$YY$YY$YY$ZZ$ZZ$ZZ$ZZ$AA$BB$CC$GG», где XXXXXXXX – 32битное 16ричное представление 1 диаметра; YYYYYYYY – 32битное 16ричное представление 2 диаметра; ZZZZZZZZ – 32битное 16ричное представление 3 диаметра; AA – 8битное 16ричное представление количества изделий 1 диаметра; BB – 8битное 16ричное представление количества изделий 1 диаметра; CC – 8битное 16ричное представление количества изделий 1 диаметра; GG – 8битное 16ричное представление количества изделий 1 диаметра;
9. По окончанию количества изделий нажать кнопку для получения данных по I2C;
10. При необходимости, в окне I2C Debugger посмотреть данные;
11. Остановить симуляцию.

Подробный алгоритм действия с программой:

1. Запустить среду Proteus, открыть файл проекта:

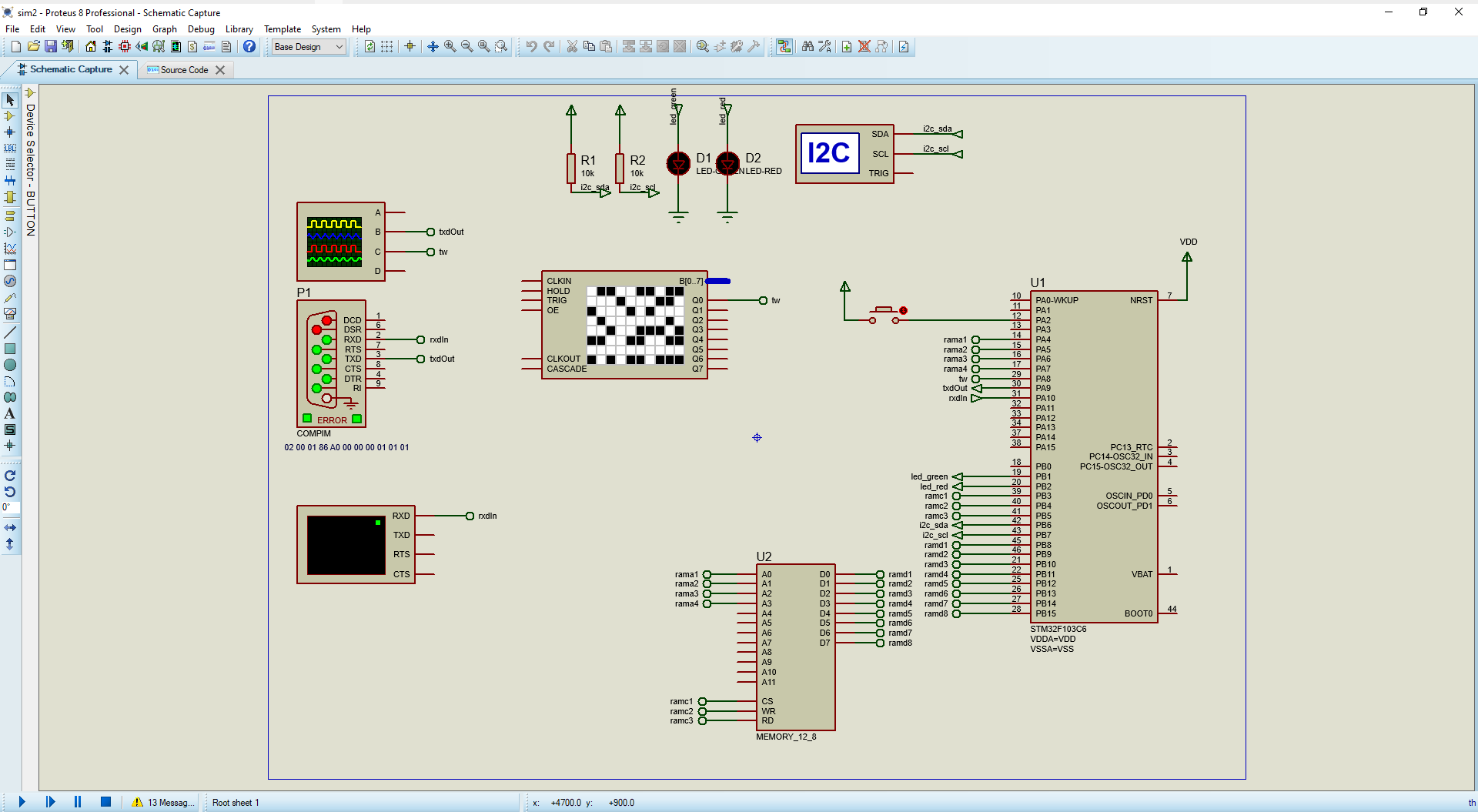


Рисунок 11 – Окно среды Proteus

1. Запустить среду STM32CubeIDE, открыть файлы проекта, выполнить компиляцию

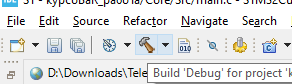


Рисунок 12 – Кнопка сборки проекта среде CubeIDE.

1. В свойствах микроконтроллера указать бинарный файл программы в папке Debug

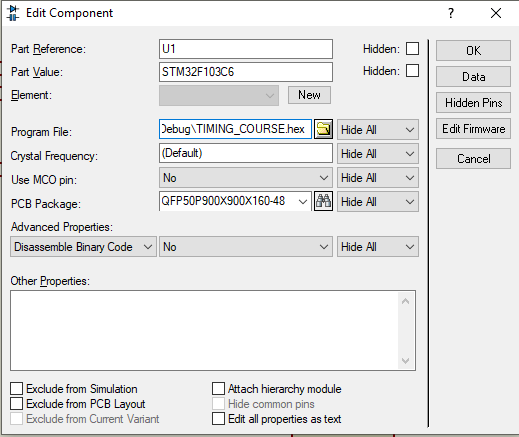


Рисунок 13 – Окно настроек устройства.

1. Настроить нуль-терминал

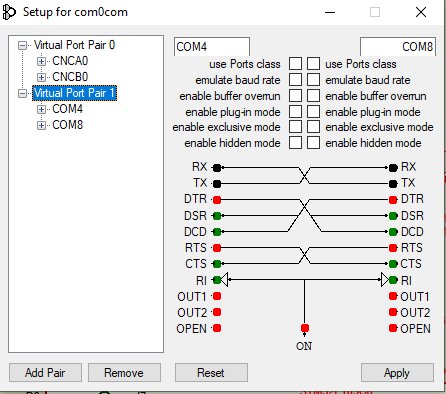


Рисунок 14 – Окно настройки нуль-терминала «com0com»

1. Выбрать в устройстве «COMPIM» один из созданных виртуальных COM-портов

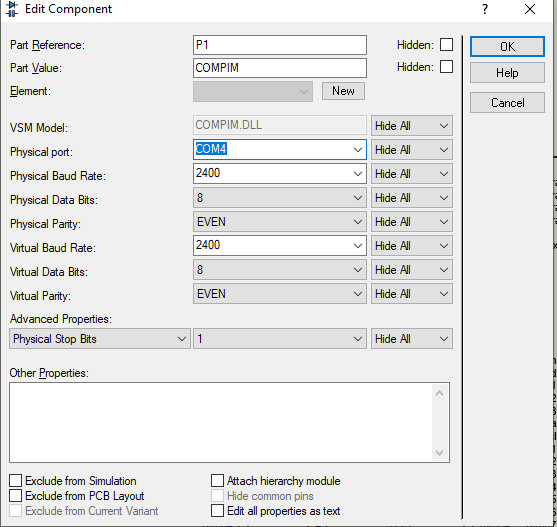


Рисунок 15 – Окно настройки устройства «COMPIM»

1. Подключить терминал к другому виртуальному com-порту, относящийся к той же паре, что и в предыдущем пункте

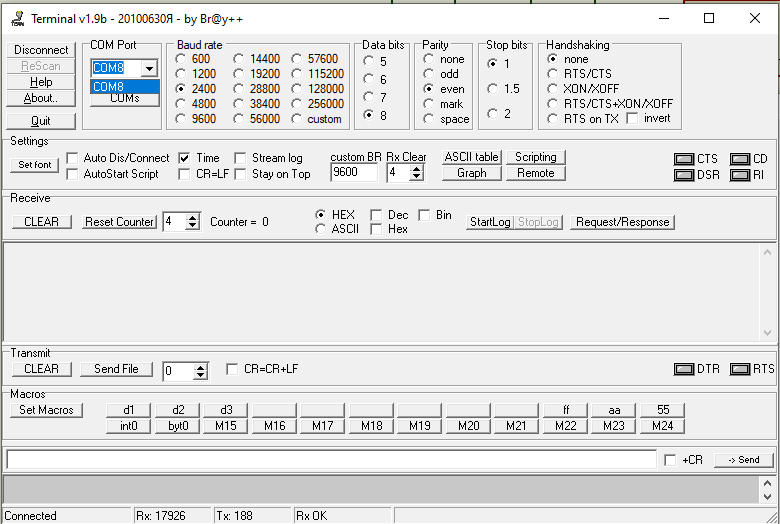


Рисунок 16 – Окно терминала «Terminal 1.9b» с выбранным портом

1. Запустить симуляцию

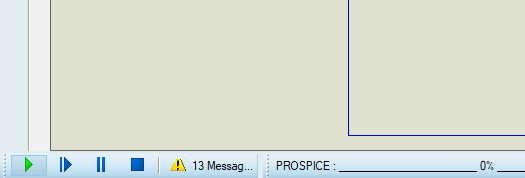


Рисунок 17 – Окно среды «Proteus» с выделенной кнопкой запуска симуляции.

1. Отправить через терминал бинарные данные в формате «$XX$XX$XX$XX$YY$YY$YY$YY$ZZ$ZZ$ZZ$ZZ$AA$BB$CC$GG», где XXXXXXXX – 32битное 16ричное представление 1 диаметра; YYYYYYYY – 32битное 16ричное представление 2 диаметра; ZZZZZZZZ – 32битное 16ричное представление 3 диаметра; AA – 8битное 16ричное представление количества изделий 1 диаметра; BB – 8битное 16ричное представление количества изделий 1 диаметра; CC – 8битное 16ричное представление количества изделий 1 диаметра; GG – 8битное 16ричное представление количества изделий 1 диаметра;

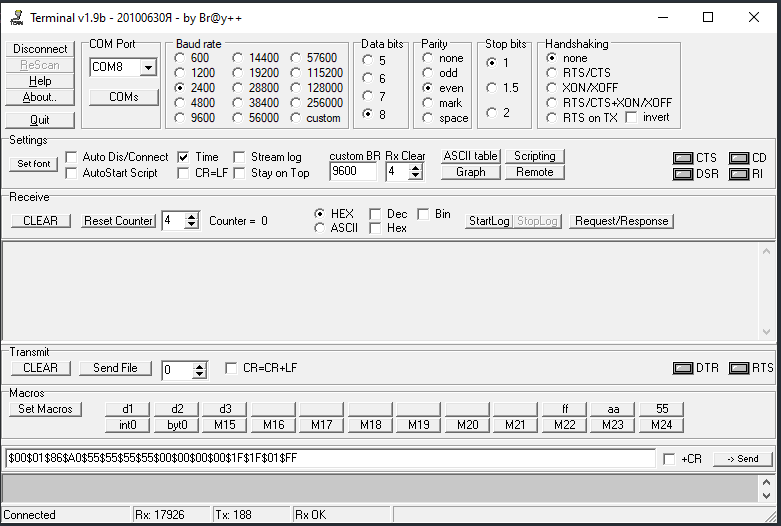


Рисунок 18 – Окно подключенного терминала с введённой последовательностью.

1. По окончанию количества изделий нажать кнопку для получения данных по I2C;

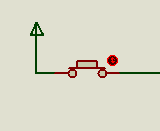


Рисунок 19 – Кнопка на схеме в среде «Proteus»

1. При необходимости, в окне I2C Debugger посмотреть данные;

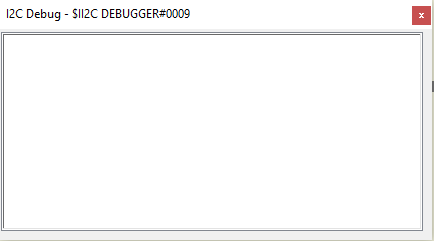


Рисунок 20 – Пустое окно устройства «I2C Debugger»

1. Остановить симуляцию.

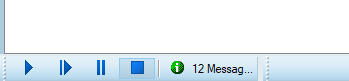


Рисунок 21 - Окно среды «Proteus» с выделенной кнопкой остановки симуляции.

Краткое описание кода:

uint32\_t diams [3] - Массив принятых диаметров

uint8\_t cnts [3] - Массив количества изделий для каждого диаметра

uint8\_t gen\_cnt - Общее количество изделий

uint32\_t falling - Размер текущего шара

# ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ КОНТРОЛЛЕРА

1. Соединить передающее устройство UART с разъёмом устройства;
2. Соединить ведомое устройство I2C с разъёмом устройства;
3. Передать данные по UART интерфейсу;
4. Начать сканирование изделий;
5. При сканировании изделия с диаметром соответствующим настройкам будут перемещается в нужный бункер. Чтобы это показать зажигается зелёный светодиод;
6. Иные изделия будут перемещается в бункер брака. Данное действия показывается красным светодиодом;
7. По окончанию изделий нажать на кнопку для передачи данных по I2C;

# ЗАКЛЮЧЕНИЕ

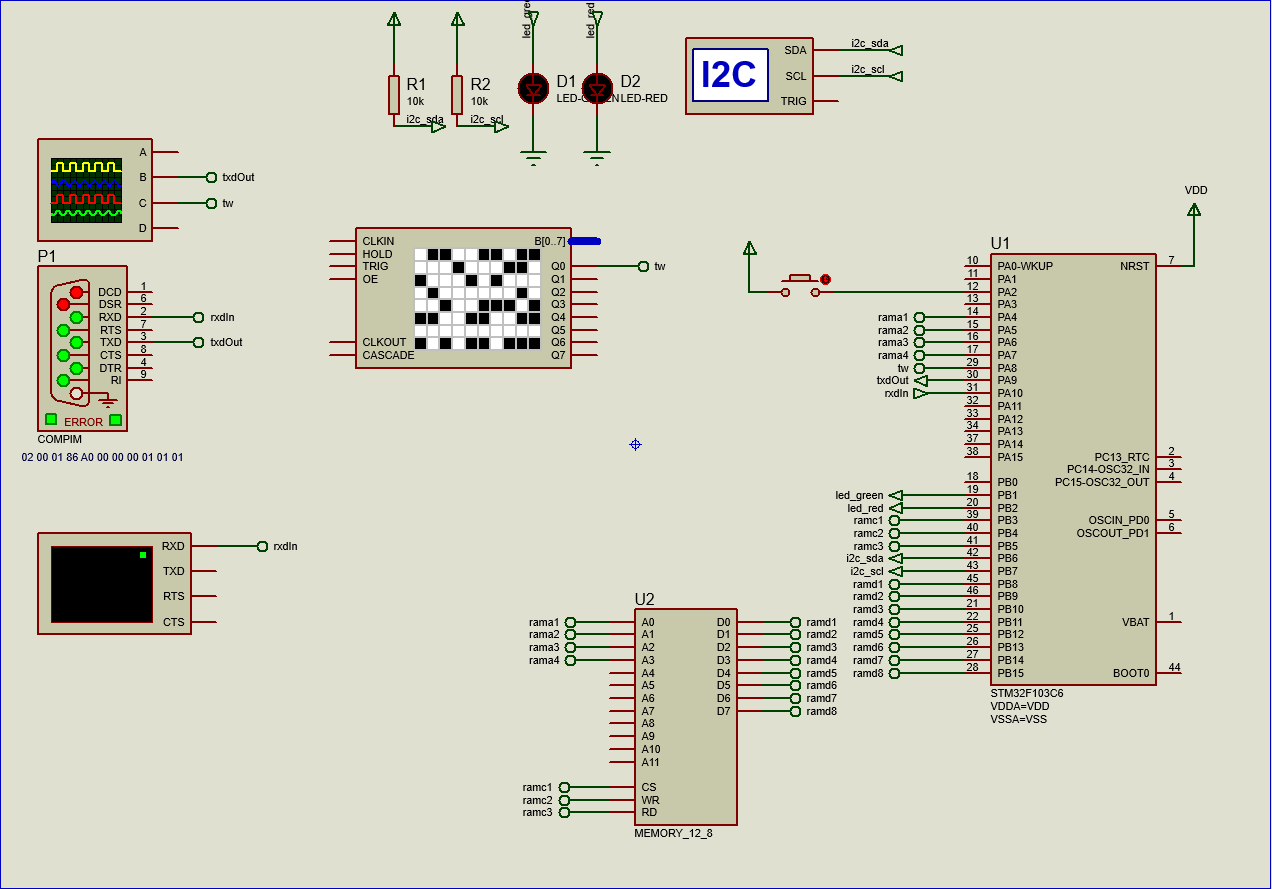
В рамках курсового проекта была реализована система, предназначенная для измерения длительности импульса, обработки и передачи данных.

В процессе работы над проектом была оформлена структурная, принципиальная электрическая, алгоритмическая схема работы устройства, так же написано управляющее ПО и осуществлена отладка устройства в симуляторе схем Proteus, произведено тестирование программы и создан пакет необходимой документации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Лабораторный практикум по изучению микроконтроллеров STM32 на базе отладочного модуля STM32F3 Discovery и STM32F4 Discovery / Бугаев В.И., Мусиенко М.П., Крайнык Я.М. – Москва-Николаев: МФТИ-ЧГУ, 2014. – 26 с.*
2. *Программирование STM32. Часть 15: Внешние прерывания EXTI . Россия, 2018.[Электронный ресурс]. URL: https://dimoon.ru/obuchalka/stm32f1/programmirovanie-stm32-chast-15-vneshnie-preryivaniya-exti.html (дата обращения: 27.10.2023)*
3. *Mainstream Performance line, Arm Cortex-M3 MCU with 32 Kbytes of Flash memory, 72 MHz CPU, motor control, USB and CAN* // ST URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f103c6.html#documentation (дата обращения: 10.12.2023).
4. *Таймеры stm32 HAL* // stD URL: https://istarik.ru/blog/stm32/121.html (дата обращения: 13.12.2023).

# ПРИЛОЖЕНИЕ А. ПРИНЦИПИАЛЬНАЯ ЭЛЕКТРИЧЕСКАЯ СХЕМА



# ПРИЛОЖЕНИЕ Б. СПЕЦИФИКАЦИЯ

Таблица Б.1 – Спецификация элементов

|  |  |  |  |
| --- | --- | --- | --- |
| Обозначение | Наименование | Количество | Описание |
| U1 | STM32F103C6 | 1 | Однокристальный микроконтроллер |
| U2 | MEMORY\_12\_8 | 1 | Внешняя ОЗУ |
| R1, R2 |  | 2 | Резисторы подтяжки |
| D1, D2 |  | 2 | Светодиоды индикации |
| P1 | COMPIM | 1 | Вход UART |