Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Дисциплина «Мобильные вычислительные системы»

«К защите допустить»

Руководители курсового проекта

Доцент кафедры ПИКС *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*В.С. Колбун

\_\_.\_\_.2022

Доцент кафедры ПИКС \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.Ч. Ролич

\_\_.\_\_.2022

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему

**«МОБИЛЬНЫЙ СКАНЕР NFC-МЕТОК НА БАЗЕ МИКРОКОНТРОЛЛЕРНОГО ЯДРА TENSILICA XTENSA»**

БГУИР КП 1–39 03 02 076 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  (подпись студента) | А.Д. Каменкова |
|  |  | Курсовой проект представлен на проверку *\_\_*.*\_\_*.2022  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  (подпись студента) |

Минск 2022

**РЕФЕРАТ**

**Каменкова, А.Д.** Мобильный сканер NFC-меток на базе микроконтроллерного ядра *Tensilica Xtensa*: пояснительная записка к курсовому проекту / А.Д. Каменкова – Минск: БГУИР, 2022. –Х с.

БГУИР КП 1-39 03 02 № 076 ПЗ

Пояснительная записка Х страниц, Х рисунка, Х источника, 6 графических материалов.

АНАЛИЗ МЕТОДОВ ПРОЕКТИРОВАНИЯ МОБИЛЬНОГО СКАНЕРА NFC-МЕТОК НА БАЗЕ МИКРОКОНТРОЛЛЕРНОГО ЯДРА *TENSILICA XTENSA,* С УЧЕТОМ ОСОБЕННОСТЕЙ АРХИТЕКТУР, СТРУКТУРОЙ САМОГО МИКРОКОНТРОЛЛЕРА И ЕГО ПРИНЦИПАХ ФУНКЦИОНИРОВАНИЯ, КОТОРЫЙ МОЖЕТ БЫТЬ ИСПОЛЬЗОВАН В ПРОМЫШЛЕННЫХ СЕТЯХ И ИНТЕГРИРОВАН НЕПОСРЕДСТВЕННО В СИСТЕМУ ДИАГНОСТИКИ АВТОМОБИЛЬНОЙ ЭЛЕКТРОНИКИ

*Цель* *проектирования*: проектирование мобильного сканера *NFC-*меток на базе микроконтроллерного ядра *Tensilica Xtensa*.

*Методология проведения работы*: В процессе решения поставленных задач использованы методы проектирования и программирования микроконтроллеров, использующихся во встраиваемых системах.

*Результаты работы*: выполнен анализ предметной области и потребностей пользователей, спроектировано мобильное автономное устройство, предназначенное для работы в пропускных системах для контроля посещения бытовых и/или служебных помещений работниками или проживающими, разработано демонстративное программное обеспечение для работы устройства.

*Область применения результатов*: для обеспечения работы пропускной системы на основе чипа, имеющего *NFC*-метку, для контроля входа/выхода рабочего персонала частных организаций и домоуправленческой деятельности.

**ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ И ТЕРМИНОВ**

СОДЕРЖАНИЕ

Введение1

1 Общетехническое обоснование разработки прибора1

1.1 Анализ исходных данных2

1.2 Теоретические сведения и принципы функционирования отдельных

узлов прибора5

1.2.1 Стандарт *NFC*-кодирования3

1.2.2 Типы *NFC*-меток 3

1.2.3 Обзор принципов действия, структурных решений и

микропроцессорной базы современных приборов сканирования

*NFC*-меток3

1.2.4 Архитектура ядра *Tensilica Xtensa*3

1.2.5 Структура микроконтроллера *ESP32*3

1.2.6 Структура модуля *ESP32-WROOM-32*3

1.2.7 Регистровая модель интерфейса *GPIO* микроконтроллера с ядром

*Tensilica Xtensa*3

1.2.8 Физический и канальный уровни интерфейсов *I2C*, *SPI* и *UART*3

1.2.9 Регистровая модель *I2C*, *SPI* и *UART* микроконтроллерного ядра

*Tensilica Xtensa*3

1.2.10 Принципы функционирования контроллера вложенных векторов

прерываний микроконтроллера с ядром *Tensilica Xtensa*3

1.2.11 Методика организации обработки прерываний интерфейса

*UART*3

1.2.12 Архитектура микроконтроллера *PN532*3

1.2.13 Структура, логика функционирования и интерфейсы связи

адаптера *PN532\_NFC\_RFID\_Module* от *Adafruit*3

1.2.14 Работа адаптера *PN532\_NFC\_RFID\_Module* в режимах

считывания и записи *NFC*-меток3

1.2.15 Стандарты *ASCII* и *UNICODE* хранения символов3

1.2.16 Форматы представления строк в памяти микроконтроллера3

1.2.17 Обобщённая методика декодирования сообщений3

1.2.18 Алгоритм формирования очереди сообщений, принимаемых от

адаптера *NFC*-меток3

1.2.19 Принципы функционирования блока *DMA* прямого доступа к

памяти3

1.2.20 Применение *DMA* в формировании очереди входных сообщений3

1.2.21 Анализ функций стандартной библиотеки «*string.h*»;3

1.2.22 Алгоритм декодирования сообщений адаптера *NFC*-меток3

1.2.23 Структура, логика функционирования и организация сдвиговой

линейки на основе сдвигового регистра *HC595*3

1.2.24 Структура и логика функционирования дисплейного модуля

*HY32D* на базе видеопроцессора *ILI9341*3

1.2.25 Структура и логика функционирования контроллеров *ADS7846* и

*STMPE811* сенсорной панели3

1.2.26 Методика организации пользовательского интерфейса3

1.2.27 Принципиальные основы и схемы зарядки литий-ионных

аккумуляторных батарей;3

2 Разработка структурной электрической схемы мобильного сканера

*NFC-*меток4

2.1 Обоснование базовых блоков структурной схемы сканера *NFC-*меток

2.2 Обоснование связей структурной схемы сканера *NFC-*меток.

3 Разработка принципиальной электрической схемы сканера *NFC-*меток4

3.1 Обоснование выбора САПР для разработки принципиальной

электрической схемы

3.2 Описание используемых библиотечных элементов и процесса их

создания.

3.3 Обоснование выбора базовых компонентов принципиальной схемы

сканера *NFC-*меток

3.4 Обоснование связей принципиальной электрической схемы сканера

*NFC-*меток.

3.5 Анализ и обоснование принципиальной электрической схемы зарядки

аккумуляторной батареи.

4 Разработка модели и алгоритма функционирования сканера *NFC-*меток4

4.1 Моделирование в среде высокоуровневого программирования

алгоритмов декодирования сообщений *NFC*-адаптера 6

4.1.1 Обоснование модели сканера *NFC-*меток.6

4.1.2 Моделирование сообщений *NFC*-адаптера.6

4.1.3 Разработка диаграммы состояний сканера *NFC-*меток.6

4.1.4 Разработка схемы алгоритма функционирования сканера

*NFC-*меток.6

4.1.5 Моделирование пользовательского интерфейса сканера

*NFC-*меток.6

5 Разработка конструкции проектируемого прибора5

5.1 Выбор и обоснование элементной базы.6

5.2 Выбор и обоснование конструктивных элементов и установочных

изделий6

6 Расчёт конструктивно-технологических параметров проектируемого

прибора 5

6.1 Проектирование печатного модуля6

6.1.1 Выбор типа конструкции печатной платы, класса точности и шага

координатной сетки3

6.1.2 Выбор и обоснование метода изготовления электронного

модуля3

6.1.3 Расчёт конструктивно-технологических параметров электронного

модуля. Определение габаритных размеров и толщины печатной

платы3

6.1.4 Определение элементов проводящего рисунка 3

6.2 Выбор и обоснование материалов конструкции и защитных покрытий,

маркировки деталей и сборочных единиц 6

7 Применение средств автоматизированного проектирования при разработке

прибора5

Заключение74

Список использованных источников75

Приложение А (обязательное) Отчет о проверке на заимствования

в системе «Антиплагиат»78

Приложение Б (обязательное) Перечень элементов79

Приложение В (обязательное) Спецификация80

Приложение Г (обязательное) Визуализированная трёхмерная модель81

Приложение Д (обязательное) Листинг программного кода82

Приложение Е (обязательное) Ведомость документов курсового проекта83

**ВВЕДЕНИЕ**

Новейшие информационные системы дают возможность оптимизации управления банковскими безналичными расчетами, а также делают перемещение в черте цивилизованного мира комфортным имея лишь пластиковую карту с *NFC*-меткой, по размерам меньше ладони, либо не имея и её, ведь данная технология встраивается в мобильный телефон. В конечном итоге вложения в инновации оборачиваются для владельцев предприятий полным контролем над процессами перемещения сотрудников и повышением финансовой грамотности с четкой электронной отчетностью о действиях.

*Актуальность темы* в том, что применение данной технологии позволяет снизить нагрузку на специалистов банковской сферы и сферы безопасности, а также на окружающую среду избежав затрат на бумагу для печати талонов или бумажных банкнот, а также металлов на производство ключей.

*Объектом исследования* является технология производства и программирования малогабаритных сканеров, с использованием *NFC*-меток.

*Предметом исследования* являются инструментальные средства разработки программируемого вычислительного средства, выполняющего функцию сканера *NFC*-меток и передачи принятой информации на управляющее устройство.

*Целью курсового проектирования* является разработка мобильного сканера *NFC-*меток на базе микроконтроллерного ядра *Tensilica Xtensa* c функцией считывания *NFC*-меток и отображение их значений на дисплее.

*Задачами курсового проектирования* является разработка структурной электрической схемы устройства, принципиальной электрической, модели и алгоритма функционирования, проектирование и разработка конструкции изделия с расчетом конструктивно-технологических параметров.

Курсовая работа проверена в системе «Антиплагиат». Процент оригинальности составляет 1000000%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников». Скриншот приведён в приложении А.

**1 ОБЩЕТЕХНИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ**

**ПРИБОРА**

**1.1 Анализ исходных данных**

С каждым днем большую популярность набирают компактные программируемые приборы, с возможностью их перенастройки и обновления с минимальными затратами на данный процесс. Микроконтроллер – это специальная микросхема, предназначенная для управления различными электронными устройствами.

Разработка электронного устройства, предназначенного для использования в качестве системы контроля посещения (входа-выхода) с использованием *NFC*-меток должна удовлетворять следующим условиям:

1 Размеры устройства: не более 80 × 30 × 10 мм.

2 Масса – не более 0,1 кг.

3 Потребляемый ток: средний – 10 мкА, пиковый – не более 20 мА.

4 Основное питание от аккумуляторной батареи типа 2032 напряжением 3В (емкость 220 мАч).

**1.2 Теоретические сведения и принципы функционирования**

**отдельных узлов прибора**

**1.2.1 Стандарт *NFC*-кодирования**

*Near Field Communication (NFC)* – «коммуникация ближнего поля» технология беспроводной передачи данных малого радиуса действия, которая дает возможность обмена данными между устройствами, находящимися на расстоянии около 10 сантиметров. При этом гаджеты могут не соприкасаться. Скорость передачи информации – 424 килобит в секунду.

В основе технологии лежит индукция магнитного поля. Индукционная катушка создает в одном устройстве электромагнитное поле. Катушка во втором, попадая в действие созданного поля создает электрический ток, который преобразуется в сигнал. Устройства по очереди создают поля и обмениваются данными. В пассивном режиме поле создает только одно из устройств. Например, таким образом работают карточки [1].

По сути NFC является частным случаем *RFID (Radio Frequency Identification)* – механизма радиочастотного обмена данными, хранящимися в транспондерах или метках, который основан на стандартах *ISO/IEC 18092 NFC IP-1, JIS X 6319-4* и [*ISO/IEC 14443*](https://ru.wikipedia.org/wiki/ISO/IEC_14443)для [бесконтактных смарт-карт](https://securityrussia.com/blog/smart-karty.html#4).

***NFC*** устройство работает на частоте 13,56 МГц и состоит из считывателя (ридера) и антенны, или из метки и антенны. Ридер генерирует радиочастотное поле, которое может взаимодействовать с меткой или с другим ридером. Ридер – это устройство *NFC*, работающее в режиме активной коммуникации. Метка – это устройство *NFC*, которое работает в режиме пассивной коммуникации.

До стандартов *NFC* существовали другие стандарты, которые позже были взяты в основу стандарта *NFC*, например, [*ISO 14443*](https://ru.wikipedia.org/wiki/ISO/IEC_14443). Он описывает частотный диапазон, метод модуляции и протокол обмена бесконтактных пассивных карт ([*RFID*](https://ru.wikipedia.org/wiki/RFID)) ближнего радиуса действия (до 10 см) на магнитосвязанных индуктивностях.

*NFC* — технология с открытой платформой, стандартизированная в [*ECMA-340*](https://www.ecma-international.org/publications/standards/Ecma-340.htm) и [*ISO/IEC 18092*](https://www.iso.org/standard/56692.html). Эти стандарты определяют схемы модуляции, кодирование, скорости передачи и радиочастотную структуру интерфейса устройств NFC, а также схемы инициализации и условия, требуемые для контроля над конфликтными ситуациями во время инициализации — и для пассивных, и для активных режимов *NFC*. Кроме того, они также определяют протокол передачи, включая протокол активации и способ обмена данными.

Радиоинтерфейс для NFC стандартизирован в:

*–* [*ISO/IEC 18092*](https://www.iso.org/standard/56692.html)*/*[*ECMA-340*](https://www.ecma-international.org/publications/standards/Ecma-340.htm): *Near Field Communication Interface and Protocol-1 (NFCIP-1);*

– [ГОСТ Р ИСО/МЭК 18092-2015](http://docs.cntd.ru/document/1200123920) Информационные технологии.

– *NFCIP-1* [*ISO / IEC 21481*](https://www.iso.org/standard/56855.html)*/*[*ECMA-352*](https://www.ecma-international.org/publications/standards/Ecma-352.htm)*: Near Field Communication Interface and Protocol-2 (NFCIP-2)* [2].

В дополнение к уже существующим стандартам *NFC Forum* собрали лучшее из этих стандартов в  документы, описывающие работу устройств, которые используют технологию *NFC* и назвали их спецификациями. Например, в спецификации *NFC Analog Technical Specification* рассматриваются аналоговые радиочастотные характеристики устройства с поддержкой *NFC*. Эта спецификация включает в себя требования к мощности антенны, требования к передаче, требования к приемнику и формы сигналов (время /частота /характеристики модуляции).

Спецификация *NFC Analog 2.0* ввела активный режим связи для обмена данными *P2P* и технологию *NFC-V* в режиме опроса. Версия 2.0 обеспечивает полную совместимость с устройствами, соответствующими ISO/IEC 14443 или ISO/IEC 18092. По этим спецификациям существует следующие способы связи для устройств *NFC: NFC-A, NFC-B, NFC-F,* и пять типов *NFC*-меток. Устройства NFC могут быть активной или пассивной коммуникации и поддерживать один (или несколько) из 3 режимов работы.

*–* Тип связи *NFC-A* основан на стандарте *ISO/IEC 14443A* для бесконтактных карт. Типы связи отличаются используемыми режимами кодирования сигнала и модуляции. *NFC-A* использует код Миллера и амплитудную модуляцию. Двоичные данные передаются со скоростью около 106 Кбит/с, сигнал должен изменяться от 0% до 100%, чтобы различать двоичную 1 и двоичный 0.

– Тип связи *NFC-B* основан на стандарте *ISO/IEC 14443B* для бесконтактных карт. *NFC-B* использует метод манчестерского кодирования. Двоичные данные также передаются со скоростью около 106 Кбит/с. Здесь вместо 100% используется 10% изменение амплитуды для двоичного 0 (то есть низкого уровня) и 100% для двоичной 1 (то есть высокого). В манчестерском кодировании переход с низкого на высокий уровень представляет двоичный 0, а переход с высокого на низкий уровень представляет двоичную 1.

– Тип связи *NFC-F* основан на стандарте *FeliCA JIS X6319-4*, также известный как просто [*FeliCa*](https://en.wikipedia.org/wiki/FeliCa)*.* Стандарт регулируется японской [*JICSAP*](http://www.jicsap.com/). Там эта технология, и наиболее популярна. Скорость передачи данных 212 / 424 Кбит/с, используется манчестерское кодирование и амплитудная модуляция.

Сфера применения технологии не ограничивается только оплатой покупок в магазине. Технология позволяет:

– Подтверждение личности. Телефон может показывать паспортные данные, водительское удостоверение, пропуск на работу. Это удобно, потому что смартфон всегда под рукой;

– Цифровой ключ. Открывайте замки подтверждая личной владельца. Так как передача данных происходит на расстоянии не более десяти сантиметров злоумышленникам будет трудно перехватить сигнал;

– Метки (*NFC* теги).

Теги *NFC* — это маломощные соединения для передачи данных, которые могут быть достигнуты через физический контакт на совместимых устройствах. Это позволяет совершать платежи и по-разному взаимодействовать с предметами. Можно оплачивать товары, открывать двери, активировать замки и взаимодействовать практически со всем, что совместимо с *NFC*.

**1.2.2 Типы *NFC*-меток**

Это круглая пассивная антенна (микрочип), диаметром два сантиметра. Используется для бесконтактного обмена информацией. Работает без подзарядки, гибкая, на метку записывается информация. Их выпускают в виде наклеек, карточек и брелоков.

[](https://public-pc.com/wp-content/uploads/2021/03/15.jpg)

Рисунок 1.1 – *NFC*-меток в виде наклейки

Внутри каждого *NFC* устройства или наклейки находится чип или же интегрированная схема (ИС). Эти микро устройства хранят вашу информацию, а также контролируют типы доступа к ней.

Разные *NFC* чипы имеют разный обьем памяти и разные настройки. Это влияет на обьем информации, который вы можете хранить на определенном чипе, а также на тип его блокировки и дргуие важные факторы. *NFC* устройства содержат в себе чипы, которые производит *NXP* – лидер по производству чипов.

Чипы NFC устройств разделяют на следующие типы *Ultralight, Ultralight C, Standard (Classic) 1K*а также новые *NTAG203.* Таблица 1.1 приводит пример параметров чипов для различных нужд.

Таблица 1.1 – Разновидности *NFC* чипов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Ultralight* | *Ultralight C* | *Standard 1K* | *Desfire 4k* | *NTAG203* |
| Объём памяти | 64 байтов | 192 байтов | 1024 байтов | 168 байтов |
| Память пользователя | 46 байтов | 137 байтов | 716 байтов | 137 байтов |
| Длинна *URL* | 41 символов | 132 символов | 256 символов | 132 символов |

Продолжение таблицы 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Длинна текста | Да | Да | Нет | Нет |
| Совместимость с мобильным телефоном | Экономичность, короткие ссылки, смарт-постер и общее *NFC* примениние | Для приложений с обязательным шифрованием | Рекомендованы для больинства общих применений NFC и при высоких нагрузках | Последний чип, лучшее исполнение. Лучший для всего начиная от маркетинга до приложений для запуска задач |
| Форум NFC типу 2 | Да | Да | Нет | Да |
| Серийный номер | Да | Да | Да | Да |
| Криптография | Нет | *3DES* | *Crypto-1* | Нет |
| Диапазон сканирования | 7 | 4 | 6 | 9 |

Рассмотрим подробнее характеристики, относительно которых рассматривается функционал того или иного *NFC* устройства:

– Объём памяти: это общий объём памяти чипа. Некоторые из них программируются только один раз (*OTP*), некоторые из них используются для блокировки и так далее. Большинство используется для того чтобы пользователь мог считывать\записывать информацию.

– Память пользователя: это общий объём бит, который нужен для вас. Это доступный объём памяти для хранения информации.

– Длинна *URL*: как правило, это максимальная длинна  *URL*, которую может хранить ваш чип, исключая  *http://* или *http://www* часть.

– Длинна текста: как правило, это максимальная длинна нешифрованного текста, который может храниться на чипе.

– Совместимость с мобильным телефоном: если вы используете приложение для *Android* для самостоятельного кодирования, то некоторые функции блокирования не будут действовать для 1к.

– Форум *NFC* типу 2: спецификации совместимости с *NFC* форумом второго типа.

– Серийный номер: чип  имеет уникальный серийный номер с целью его идентификации.

– Криптография: функция безопасности чипа, которая может помочь предотвратить клонирование. Это очень продвинутая функция, которая требует специальных знаний и очень редко нужна для обычного использования.

– Диапазон сканирования: Средний диапазон сканирования *NFC* определенного устройства, которое продается на *NFC*. Это является показателем относительного расстояния диапазона сканирования чипаде используются.

Метка программируется через программу, установленную на смартфоне. Далее поднесите смартфон к метке. Произойдет синхронизация. Выполнится действие, которое было запрограммированно на микрочипе.

**1.2.3 Обзор принципов действия, структурных решений и**

**микропроцессорной базы современных приборов**

**сканирования *NFC*-меток**

Данные технологии создаются для автоматизации, ускорения и упрощения работы, создания комфортного использования устройств. Рассмотрим подробнее самые распространенные структурные решения и их использование:

1 Использование смарт-карт и карт лояльности. Эмуляция банковских карт не самое простой процесс, потому рассмотрим аналоговую технологию – карты лояльности. Их разновидностей существует достаточное количество: и с магнитной лентой, с *QR*-кодом, и со штрих-кодом. Добавив карту лояльности, уменьшается количество переносимого пластика. Но дополнительной функцией является реализация метода для передачи данных о карте лояльности в систему торговой точки (*POS*) продавца, то есть возможные скидки по карте лояльности будут учитываться прямо во время оплаты заказа. Карта лояльности должна содержать в себе *NFC*. Карты со штрихкодом или *QR*-кодом необходимо предоставить до оплаты для сканирования их с экрана телефона.

Карты лояльности сохраняются в платежном приложении вместе с привязанной банковской картой и используются в режиме эмуляции карты *NFC*. Эти данные хранятся в защищенном элементе ([*Security Element*](https://www.kaspersky.ru/blog/secure-element/20556/)или *SE*) в телефоне. *SE* — это защищенная от взлома платформа, как правило, это чип с памятью, которая позволяет безопасно хранить конфиденциальные данные, защищенные криптографическими ключами. *SE* может быть интегрирован в *SIM*-карту, выпущенную мобильной сетью пользователя, или чипом, встроенным в телефон производителем устройства. Эта реализация хранит учетные карт лояльности в том же домене безопасности, что и данные платежной карты, привязанные к платежному приложению. И хотя данные карт лояльности, не то чтобы жизненно необходим безопасно хранить, но при хранении в *SE* они защищены от стирания или изменения другими приложениями.

В работе бесконтактной карты добавляется *NFC* модуль, который обеспечивает бесконтактное соединение со считывателем банковских карт. Что же происходит в случае эмулирования карты мобильным телефоном. Чтобы не записывать на чип *SE* в мобильном устройстве платежные приложения всех банковских карт, которыми пользуется владелец устройства, которые к тому же надо персонализировать, т.е. передать данные о выпущенных картах и хранить их в защищенном виде, была сформулирована роль *TSM (Trusted Service Manager*), который объединяет с одной стороны поставщиков услуг (*Service Provider TSM)*, а с другой стороны чипы *Secure Element (Secure Element Issuer TSM).TSM* — *Trusted Service Manager* — уникальный посредник, который владеет ключами. Это аппаратно-программный комплекс, предоставляющий технологические отношения между операторами связи и поставщиками услуг.

Ключевые услуги доверенной третьей стороны включают защищенную загрузку и менеджмент контента элемента безопасности, выполняемый при взаимодействии с провайдерами мобильных сервисов. Это могут быть банки, транспортные компании, поставщики и агрегаторы услуг.

Однако эта технология платежей все равно требовала присутствия физического защищенного элемента на мобильном устройстве. Что давало определенные ограничения, например, если производитель мобильного устройства не включил *SE* в свою платформу, в этом случае, требовалось менять *SIM*-карту на карту с поддержкой *SE* у мобильного оператора. Эмуляция карты неотделима от понятия «токенизация», потому что это следующая ступень защиты платежных данных в виртуальном мире после *TSM*, который выдавал ключи.

[Токен](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BA%D0%B5%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D1%8C)) — это ссылка (то есть идентификатор), которая сопоставляется с конфиденциальными данными через систему токенизации. Сопоставление исходных данных с токеном использует методы, которые делают невозможным обратное преобразование токенов в исходные данные вне системы токенизации, например, с использованием токенов, созданных при помощи случайных чисел. Т.е. вместо номера вашей карты API хранит токен, полученный от банка-эмитента, который бесполезен в том виде, в котором он хранится. Даже если его узнают третьи лица, воспользоваться им будет невозможно.

2 Считыватели дверей и бесконтактных карт. Самым распространённым примером является дверь с *NFC* у входа в дом. Что же касается крупных компаний, производств и заводов, то для контроля доступа уже не первый год используют *RFID*-пропуски, которые активно заменяют на *NFC*. Переход от *RFID* с оборудования низкочастотного диапазона на более функциональные высочастотные устройства, со [*ISO / IEC 14443*](https://ru.wikipedia.org/wiki/ISO/IEC_14443)позволяет выполнять дополнительные функции, кроме обычного контроля доступа, такие как оплата проживания, создание пропуска, проверка личности и предоставление других разрешений. Это упрощает предоставление услуг: гостиничный бизнес, авиаперевозки (посадочные талоны в телефоне вместо сканирования QR-кодов на билетах), пропускной контроль, ограничение физического доступа и т.п.

3  Консорциум *Car Connectivity Consortium (CCC*), в которую входят практически все основные технологические компании и автомобильные производители, утвердил [первую версию электронного ключа](https://carconnectivity.org/wp-content/uploads/CCC-Digital-Key-Brochure.pdf). С его помощью владельцы автомобилей смогут закрывать и открывать двери, а также заводить свою машину через приложение. Соответственно, чтобы открыть машину, необходимо поднести телефон к ручке двери, а чтобы завести авто — расположить телефон в специальном местена приборной панели внутри, где будет располагаться беспроводная зарядки, а после включается зажигание.

Что касается безопасности такого подхода: технология цифрового ключа или *SmartKey* (или *Digital Key*) разработана таким образом, что ключ не хранится и не передается в открытом виде. Цифровой ключ, это какой-то оригинальный набор данных, которые производитель автомобиля зашивает в прошивку автомобиля вместе с набором функций, которые доступны по этому ключу. Он же (производитель автомобиля) является *TSM (Trusted Service Manager*) для пользователей ключа, т.е. пользователь не получает ключ от автомобиля, он получает набор зашифрованных данных, которые являются ключом к расшифровке оригинального ключа, и хранятся они в[*SE*](https://www.kaspersky.ru/blog/secure-element/20556/) мобильного устройства, соответственно.

*NFC* используется только для передачи этих зашифрованных данных автомобилю. Учитывая, что *NFC* работает на расстоянии около 10 см., практически невозможно просканировать и узнать эти данные. Еще важной частью архитектуры безопасности является *TEE*, это так называемая *Trusted Execution Environment* или безопасная среда исполнения, является безопасной площадью основного процессора и гарантирует защиту кода и данных, загруженных внутри, в отношении конфиденциальности и целостности.

4 *NFC*-метки. Метки *NFC* представляют собой миниатюрные наклейки – пассивные устройства, работающие сами по себе. Технические характеристики меток следующие:

– Срок хранения наклеек – до 10 лет.

– *NFC*-чипы способны не выйти из строя даже после попадания в воду (при надлежащей конструкции).

– Не потерять ценные качества при температурах от -25 до +85 градусов.

– Рабочая частота – 13,56 МГц.

– Объем памяти – зависит от конкретного производителя, но обычно приближается к 888 байтам.

– Скорость чтения – 106 Кбит/c.

Многие до сих пор не знают о существовании таких прекрасных вещей, как *NFC*-метки, с помощью которых можно запрограммировать необходимые сценарии действий.

**1.2.4 Архитектура ядра *Tensilica Xtensa***

Технология фирмы *Tensilica* позволяет разработчикам «систем на кристалле» адаптировать структуру процессора под особенности решаемой задачи, достигая при этом существенного повышения производительности и энергоэффективности. Современные цифровые системы управления все чаще делаются однокристальными, и термин «система на кристалле» (*System-on-Chip SoC, СнК*) уже стал достаточно привычным. Успехи полупроводниковых технологий позволяют при очень низкой конечной цене размещать на кристалле многие миллионы транзисторов и реализовывать в системе высокопроизводительный процессор, емкую подсистему памяти, обширный набор периферийных устройств. Это позволяет использовать *СнК* практически во всех областях человеческой деятельности.

В то же время присутствует стремление к повышению эффективности цифровых систем управления – естественно желание получить требуемый набор функций за минимальную цену. Разработчик не хочет иметь в системе ничего лишнего, чем, в общем-то, и объясняется огромная номенклатура готовых к употреблению *СнК*, различающихся тактовыми частотами и разрядностями процессора, объемами памяти, количеством и возможностями периферийных подсистем.

Любая конкретная прикладная программа почти всегда не в полной мере использует возможности процессора, и в то же время отдельные фрагменты алгоритма не удается достаточно эффективно реализовать фиксированной системой его команд. Кроме того, каждое из семейств с его конкретным процессором требует использования своего набора средств поддержки разработки – трансляторов, программаторов, отладчиков, симуляторов.

Цифровой процессор в составе «системы на кристалле» используется для универсальных компьютеров. Фирмы-производители стремились получить в одном чипе максимальную производительность путем повышения тактовой частоты и уровня внутреннего параллелизма. Тактовая частота тесно связана с внутренней структурой процессорного конвейера: один из путей ее повышения – увеличение количества ступеней в конвейере с одновременным уменьшением логической глубины каждой ступени. Но при этом существенно возрастает объем оборудования, из-за чего растет и мощность, потребляемая процессором, причем гораздо быстрее, нежели производительность. Второй недостаток «длинного» конвейера – большие дополнительные потери времени в случае нарушения идеальной работы конвейера. Аналогичный эффект наблюдался и при использовании суперскалярных архитектур: они также требовали существенного увеличения объема оборудования, в результате чего потребление энергии опять росло быстрее, нежели производительность.

В то же время в управляющих (встраиваемых) системах требования к производительности далеко не всегда были главной целью, важнее комплексный критерий, отражающий соотношение характеристик производительность – «потребляемая мощность – площадь на кристалле – цена». Для оценки производительности и энергетической эффективности в последнее время вместо указания тактовой частоты или производительности все более широко используются удельные характеристики MIPS/МГц, *MIPS*/мВт и т.п. Разработчики микроконтроллеров уделяют большое внимание энергоэффективности систем, для чего при проектировании целенаправленно стремятся не только достичь высокой производительности, но и понизить при этом удельные затраты энергии.

Уже с самого возникновения однокристальных микроконтроллеров их разработчики стали проектировать вычислительные ядра, специально предназначенные для встраиваемых применений. Их особенностями были:

– ограниченный набор команд, поскольку в управляющих программах относительно редко требуются сложные операции, даже такие как целочисленное деление и умножение;

– развитая подсистема ввода/вывода дискретных сигналов (параллельные порты);

– поддержка побитовых операций с регистрами периферийных устройств.

Наиболее популярные высокопроизводительные 32-разрядные процессорные *RISC*-архитектуры, используемые в СнК, – это *ARM* и *MIPS*, разработанные одноименными фирмами. Для интеграции процессорного ядра в СБИС системный разработчик должен приобрести лицензию, после чего он получает описание процессора на языке описания аппаратуры (*HDL*), пригодное для использования в системах автоматизации проектирования СБИС.

*TIE (Tensilica Instructions Extension language)* – это язык описания аппаратуры (*HDL*), похожий на широко известный *Verilog*. Язык *TIE* используется не только для представления результата работы транслятора *XPRES*, но и позволяет раз-работнику вручную описывать добавляемые команды, дополнительные регистры процессора, исполнительные узлы и т. п.

*Xtensa Processor Generator* – набор программных средств для автоматического внесения изменений, выбранных разработчиком, во все компоненты описания проекта, которые содержатся в базе данных о конфигурируемом процессоре.

Во-первых, это файлы описания аппаратуры, используемые технологами на последующих этапах проектирования СБИС (синтез на уровне вентилей, расположение на кристалле, трассировка межсоединений, анализ потребляемой мощности, анализ временных характеристик и т. п.).

Во-вторых, это набор программных средств трансляции и отладки с учетом новых возможностей процессора: компиляторы, ассемблер, линкер, отладчик, средства профилирования.

В-третьих, это программные средства моделирования модифицированного процессора: несколько симуляторов, выполняющих моделирование на разных уровнях детализации, от уровня процессорных команд (самый быстрый) до уровня работы отдельных вентилей чипа (самый медленный, но самый детальный), анализаторов производительности (трассировщиков) и т. п.

Разработчик системы, используя модифицированные средства трансляции и отладки, имеет возможность немедленно проверить эффективность новой конфигурации процессора при выполнении реальных управляющих программ. Процесс проектирования выполняется итеративно. После внесения изменений можно повторять цикл «трансляция — выбор реконфигурации – анализ производительности и затрат» до получения результата, устраивающего разработчика.

На рисунке 1.2 (заимствован из [19], разд. 1) представлены результаты ускорения для различных алгоритмов при максимальном использовании всех возможностей технологий *Xtensa*.

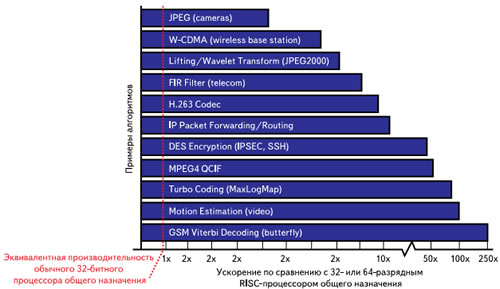


Рисунок 1.2 – Достижимые степени ускорения, предоставляемые технологией *Xtensa* для разных алгоритмов

Интегрированная среда *Xtensa Xplorer IDE* используется как «контейнер» для всех остальных программных инструментов и позволяет выполнять все действия как по оптимизации архитектуры, так и по разработке и отладке программ для процессора. На рис. 6 приведен результат работы компилятора *XPRES*, показывающий соотношение между ускорением и затратами оборудования.

Как видно из таблицы 1.2 (заимствована из [18]), для четырех разных алгоритмов выигрыш в потреблении энергии при сохранении производительности может быть от 2 раз до почти двух порядков. Плата за это – увеличение площади на кристалле.

Таблица 1.2 – Эффективность оптимизации процессоров *Xtensa*

по энергопотреблению

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм | Показатель | Процессор до оптимизации | Оптимизированный процессор | Уменьшение потребления энергии |
| Скалярное произведение векторов | Площадь чипа | 0,9 | 1,3 | 2х |
| Тысяч тактов | 12 | 5,9 |
| Мощность (мВт/МГц) | 0,3 | 0,3 |
| Энергия (мкДж) | 3,3 | 1,6 |
| Алгоритм блочного шифрования AES | Площадь чипа (мм2) | 0,4 | 0,8 | 82х |
| Тысяч тактов | 283 | 2,8 |
| Мощность (мВт/МГц) | 0,2 | 0,3 |
| Энергия (мкДж) | 61,1 | 0,7 |
| Алгоритм декодирования Витерби | Площадь чипа (мм2) | 0,5 | 0,6 | 33х |
| Тысяч тактов | 280 | 7,6 |
| Мощность (мВт/МГц) | 0,2 | 0,3 |
| Энергия (мкДж) | 65,7 | 2 |
| Быстрое преобразование Фурье | Площадь чипа (мм2) | 0,4 | 0,6 | 22х |
| Тысяч тактов | 326 | 13,8 |  |
| Мощность (мВт/МГц) | 0,2 | 0,2 |  |
| Энергия (мкДж) | 56,6 | 2,5 |  |

Процессоры *Xtensa* уже используются в принтерах и сканерах фирм *Epson* и *Hewlett-Packard*, в компонентах сетевой инфраструктуры *Cisco*, в видеокартах от *ATI* и *NVIDIA*, в медиаплеерах, *DVD*-проигрывателях, *HDTV*-видеокамерах.

Фирма *Tеnsilica* предлагает комплексную и законченную технологию и инструментальные средства, позволяющие индивидуализировать процессорное ядро под класс решаемых задач. Одновременно эта технология в значительной степени способна решить и проблему пиратства в области микроконтроллерных программ, поскольку такого же, как у вас, варианта процессора (и соответствующих средств поддержки разработки) не будет ни у кого. Одна из наиболее ресурсоемких операций в компрессии видео, используемая для детекции движения в кадре – это *SAD (Sum of Absolute Differences*, сумма модулей разностей). Команда использует возможности организации 128-битовой разрядности в конвейере нового исполнительного блока, предназначенного для выполнения этой команды, и в интерфейсе памяти в архитектуре *Xtensa*, что позволяет обрабатывать одновременно 16 пикселей. Конвейер исполнительного блока выполняет сначала вычисление разностей, затем операцию взятия модуля и, наконец, суммирование с накоплением. Использование этой команды уменьшило требуемую производительность процессора с 640 до 14 млн кмд/с (в 46 раз).

Основной путь для достижения энергосбережения в СнК состоит в отключении тактирования узлов, которые в некоторые периоды времени не используются. В процессорах Xtensa вдобавок имеется возможность деактивировать блоки памяти на периоды, когда эти блоки не используются.

Кроме того, при введении изменений в конфигурацию процессора может быть достигнуто уменьшение количества процессорных тактов и, соответственно, энергии, необходимой для выполнения прикладного алгоритма, в результате чего процессор при обеспечении той же средней производительности большую часть времени может находиться в энергосберегающих режимах либо работать на пониженной тактовой частоте: и то и другое приведет к снижению потребляемой мощности.

**1.2.5 Структура микроконтроллера *ESP32***

Микроконтроллер *ESP32* – это одна из самых доступных и мощных платформ для создания умных проектов с поддержкой *Wi-Fi*. Придя на смену *ESP8266*, этот чип дал новые возможности для разработчиков, хотя по-прежнему остались старые проблемы с поддержкой и документацией. Устройство представляет собой систему на кристалле, построенную по технологии *TSMC* 40 нм, с *Wi-Fi* и *Bluetooth* контроллерами. Оно оснащено двухъядерным 32-битным процессором, который работает на частотах 80, 160 или 240 МГц. Также в систему интегрированы антенные коммутаторы, радиочастотные компоненты, фильтры, усилители, модули управления питанием. Подключается *ESP32* к компьютеру через обычный *USB*-провод.

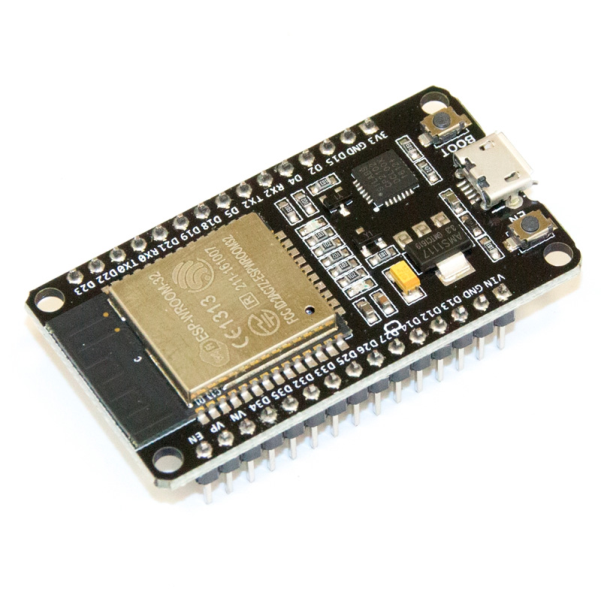


Рисунок 1.3 – Микроконтроллер *ESP32*

Технические характеристики *ESP32*:

– двух- или одноядерный 32-битный процессор *Tensilica Xtensa* *LX6;*

– тактовая частота – 160 или 240 МГц;

– 520 Кб *SRAM;*

– максимальный ток потребления 260 мА, в спящем режиме – 10 мА;

– стандарты беспроводной связи – *Wi-Fi*: 802.11 b / g / N, *Bluetooth*: v4.2 *BR/EDR and BLE*;

– наличие датчиков температуры, Холла, *Touch*-сенсоров;

– инфракрасное дистанционное управление;

– можно подключать двигатели и светодиоды через *PWM*- разъем;

– стандарт *IEEE 802.11* с поддержкой *WFA, WPA/WPA2 и WAPI;*

– возможность безопасной загрузки;

– шифрование флэш диска.

Также в модуле присутствует встроенное управление энергопитанием. Для этого используются линейный регулятор, индивидуальное питание для *RTC* (ядро низкого энергопотребления), пробуждение по таймеру или сенсорному датчику.

Большинство проектов реализуется на *Arduino IDE и Espruino*.

**1.2.6 Структура модуля *ESP32-WROOM-32***

Расположение выводов зависит от производителя. Например, есть плата *ESP32 DEVKIT V1 DOIT*, у которой 36 контактов. Распиновка представлена на рисунке 1.4.

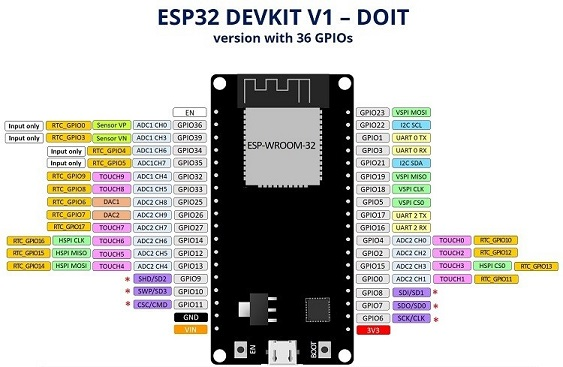


Рисунок 1.4 – Распиновка *ESP32 DEVKIT V1 DOIT*

К портам *GRIO* 0, 4, 2, 15, 13, 12, 14, 27, 33 и 32 подключены сенсорные выводы. Они могут использоваться для вывода ESP32 из глубокого сна. Они фиксируют любое изменение электрического заряда. Порты с 34 по 39 используются только для ввода информации. На них отсутствуют подтягивающие резисторы, поэтому их нельзя использовать как выходы. На некоторых платах отображаются контакты 6-11. Они подсоединены к к интегрированной *SPI flash*. Их не используют в проектах.

Есть различные версии платы *ESP32* размерами 5х5 мм или 6х6 мм. Также на основе *ESP32* существуют модули *SMT* для интегрирования в другие платы [20].

В качестве подсказки того, за какие функции отвечают разные контакты [*ESP32*](https://wikihandbk.com/ruwiki/index.php?title=ESP32&action=edit&redlink=1), можно использовать рисунок 1.5.

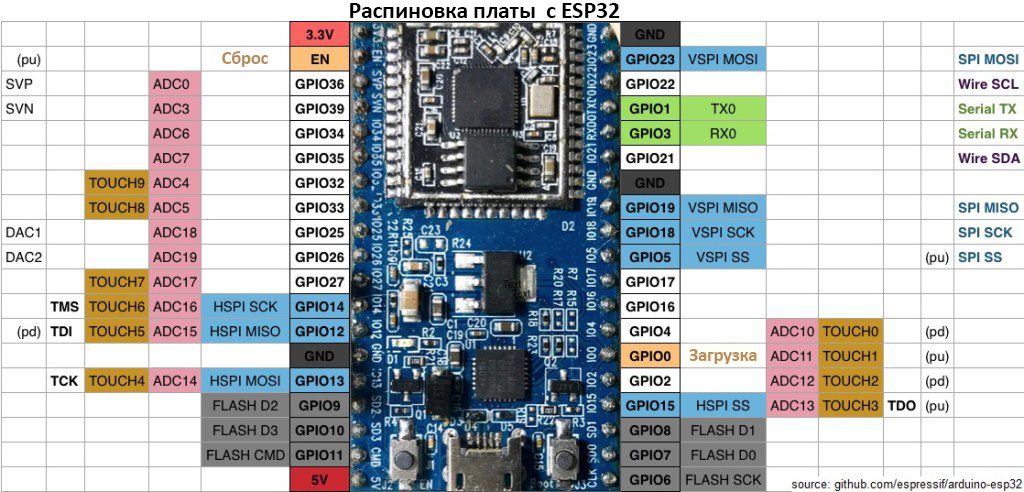
[](https://wikihandbk.com/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:ESP32_dev_board_pinout.jpg)

Рисунок 1.5 – Распиновка *ESP32*

[*ESP32*](https://wikihandbk.com/ruwiki/index.php?title=ESP32&action=edit&redlink=1) оснащен новыми функциями относительно более старой версии того же модуля [*ESP8266*](https://wikihandbk.com/wiki/ESP8266), но самые главные из них – это два ядра и функционал для беспроводной коммуникации по [*Wi-F*i](https://wikihandbk.com/ruwiki/index.php?title=WiFi&action=edit&redlink=1) и [*Bluetooth*](https://wikihandbk.com/ruwiki/index.php?title=Bluetooth&action=edit&redlink=1).

Функции и характеристики у данной платы следующие:

1 Беспроводная связь:

[*Wi-Fi*](https://wikihandbk.com/ruwiki/index.php?title=WiFi&action=edit&redlink=1) – скорость 150 Мбит/сек в режиме HT40;

[*Bluetooth*](https://wikihandbk.com/ruwiki/index.php?title=Bluetooth&action=edit&redlink=1)*–*[*BLE*](https://wikihandbk.com/ruwiki/index.php?title=BLE&action=edit&redlink=1)*(Bluetooth Low Energy)* и стандартный [*Bluetooth*](https://wikihandbk.com/ruwiki/index.php?title=Bluetooth&action=edit&redlink=1).

2 Процессор: 32-битный 2-ядерный микропроцессор [*Tensilica Xtensa LX6*](https://wikihandbk.com/ruwiki/index.php?title=Tensilica_Xtensa_LX6&action=edit&redlink=1), умеющий работать на 160 и 240 МГц.

*ROM*: 448 Кб.

*SRAM*: 520 Кб.

3 Низкое энергопотребление: например чтоб сделать аналогово-цифровые преобразования, к примеру, в режиме глубокого сна.

4 Функционал для ввода/вывода данных на периферийные устройства: 5 Передача данных с *DMA*; поддерживаются емкостные датчики, [АЦП](https://wikihandbk.com/ruwiki/index.php?title=%D0%90%D0%A6%D0%9F&action=edit&redlink=1), [ЦАП](https://wikihandbk.com/ruwiki/index.php?title=%D0%A6%D0%90%D0%9F&action=edit&redlink=1), [*I2C*](https://wikihandbk.com/ruwiki/index.php?title=I2C&action=edit&redlink=1)*,*[*UART*](https://wikihandbk.com/ruwiki/index.php?title=UART&action=edit&redlink=1)*,*[*SPI*](https://wikihandbk.com/ruwiki/index.php?title=SPI&action=edit&redlink=1)*,*[*I2S*](https://wikihandbk.com/ruwiki/index.php?title=I2S&action=edit&redlink=1)*,*[*RMII*](https://wikihandbk.com/ruwiki/index.php?title=RMII&action=edit&redlink=1)и [ШИМ](https://wikihandbk.com/ruwiki/index.php?title=%D0%A8%D0%98%D0%9C&action=edit&redlink=1).

6 Безопасность: Аппаратные ускорители для *AES* и *SSL/TLS.*

7 Подготовка модуля *ESP32* для *IDE Arduino*

8 Чип [*ESP32*](https://wikihandbk.com/ruwiki/index.php?title=ESP32&action=edit&redlink=1) встраивается в *IDE Arduino* аналогично [*ESP8266*](https://wikihandbk.com/wiki/ESP8266), но на данный момент пока не все работает как нужно. Есть аддон, благодаря которому[*ESP32*](https://wikihandbk.com/ruwiki/index.php?title=ESP32&action=edit&redlink=1)можно программировать при помощи *IDE Arduino* и ее языка программирования.

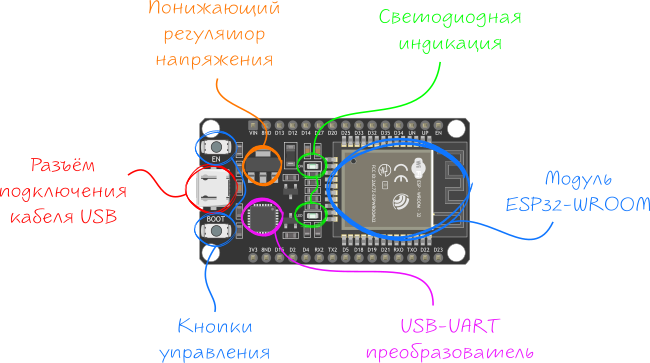


Рисунок 1.6 – Функциональная составляющая платы [*ESP32*](https://wikihandbk.com/ruwiki/index.php?title=ESP32&action=edit&redlink=1)

При подключении  [*ESP32*](https://wikihandbk.com/ruwiki/index.php?title=ESP32&action=edit&redlink=1) к компьютеру нужно сделать следующее: откройте меню «инструменты» > «плата (Tools > Board)» и выберать свою плату. В нашем случае это «*ESP32 Dev Module*».Выберите нужный [*COM*](https://wikihandbk.com/ruwiki/index.php?title=COM&action=edit&redlink=1)-порт. Для переключения  [*ESP32*](https://wikihandbk.com/ruwiki/index.php?title=ESP32&action=edit&redlink=1) перешел в режим загрузки кода, зажмите кнопку *BOOT* и нажмите на кнопку EN. Это перезагрузит [*ESP32*](https://wikihandbk.com/ruwiki/index.php?title=ESP32&action=edit&redlink=1).

**1.2.7 Регистровая модель интерфейса *GPIO* микроконтроллера с**

**ядром *Tensilica Xtensa***

В *ESP* используются *GPIO* от 0 до 16, но не все их можно использовать так CPIO от 6 до 11 используются памятью, из остальных *GPIO* часть так же имеет специфическое использование и об этом ниже.

*GPIO* 0 – для старта модуля должен быть высокий уровень, High, единица, т. е. не должен быть никуда подключен или, что лучше, подтянут к питанию 3,3В через резистор (если будет подтянут к земле включится режим перепрошивки).

*GPIO* 1 – используется в *Serial* порту на отправку (TX) можно использовать если не нужен *Serial* порт. И помним - при старте модуля любая прошивка будет дергать TX т. е. *GPIO1*, отправляя отладочную информацию в *UART0* на скорости 74480, а значит, будет, например, перещелкивается реле и загораться светодиод, но можно подключать датчики или кнопку.

*GPIO* 2 – во время включения на нем должен быть высокий уровень, High, единица т. е. не должен быть никуда подключен или, что лучше, должен быть подтянут к питанию 3,3V через резистора потом его можете использовать (На практике как правило никуда не подтянут)

*GPIO* 3 – используется в *Serial* порту на получение (RX) можно использовать если не нужен *Serial* порт.

*GPIO 6-GPIO* 11 – используются для работы с флэш памятью

Лучше не использовать ни один из контактов GPIO с 6 по 11[22].

*GPIO 6 – SCLK* (тактовый сигнал);

*GPIO 7 MISO* (от ведомого к ведущему канал 1);

*GPIO 8 MOSI (*от ведущего к ведомому канал 1);

*GPIO 9 MISO* (от ведущего к ведомому канал 2);

*GPIO 10 MOSI* (от ведущего к ведомому канал 2) Возможно использование при выборе прошивки модуля типа «DOUT» или «DIO»;

*GPIO 11 CS0* (выбор микросхемы, выбор ведомого);

*GPIO*15 – во время включения на нем должен быть низкий уровень, *LOW*, ноль, должен быть подтянут к земле через резистор, потом его можете использовать;

GPIO16 используется для выхода модуля из сна (для выхода из режима глубокого сна модуль перезагружает сам себя, подавая низкий уровень на GPIO16).

При практическом использовании GPIO16:

– появляется высокий уровень при загрузке микроконтроллера и довольно длительно.

– при прошивке модуля появляется высокий уровень

– при использовании *millis*() вместо *delay*() циклически перезагружает процессор.

Использование портов в строгом порядке продлевает срок службы микроконтроллера и гарантирует бесперебойную работу.

**1.2.8 Физический и канальный уровни интерфейсов *I2C*, *SPI* и**

***UART***

Интерфейс *I²C* (IIC) — это достаточно широко распространённый сетевой последовательный интерфейс, придуманный фирмой *Philips* и завоевавший популярность относительно высокой скоростью передачи данных (в современных микросхемах до 400 кбит/с), дешевизной и простотой реализации.

Физически сеть представляет собой двухпроводную шину, линии которой называются *DATA* и *CLOCK* (необходим ещё и третий провод – земля, но интерфейс принято называть двухпроводным по количеству сигнальных проводов) Образец таковой физической сети представлен на рисунке 1.7

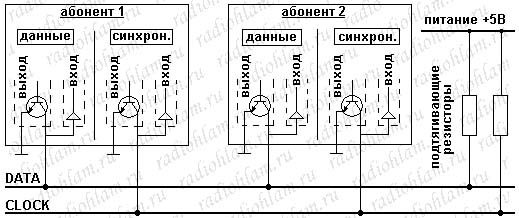


Рисунок 1.7 – Шина на интерфейсе *I²C*

Соответственно, по линии *DATA* передаются данные, линия *CLOCK* служит для тактирования. К шине может быть подключено до 128 абонентов, каждый со своим уникальным номером. В каждый момент времени информация передаётся только одним абонентом и только в одну сторону.

Устройства *I²C* имеют выход с «открытым коллектором». Когда выходной транзистор закрыт – на соответствующей линии через внешний подтягивающий резистор устанавливается высокий уровень, когда выходной транзистор открыт – он притягивает соответствующую линию к земле и на ней устанавливается низкий уровень (смотрите рисунок). На рисунке треугольниками на входе показано, что входы высокоомные и, соответственно, влияния на уровни сигналов на линиях они не оказывают, а только «считывают» эти уровни. Обычно используются уровни 5В или 3,3В.

С точки зрения логики любое устройство на шине *I²C* может быть одного из двух типов: *Master* (ведущий) или *Slave* (ведомый). Обмен данными происходит сеансами. *Master*-устройство полностью управляет сеансом: инициирует сеанс обмена данными, управляет передачей, подавая тактовые импульсы на линию *Clock*, и завершает сеанс.

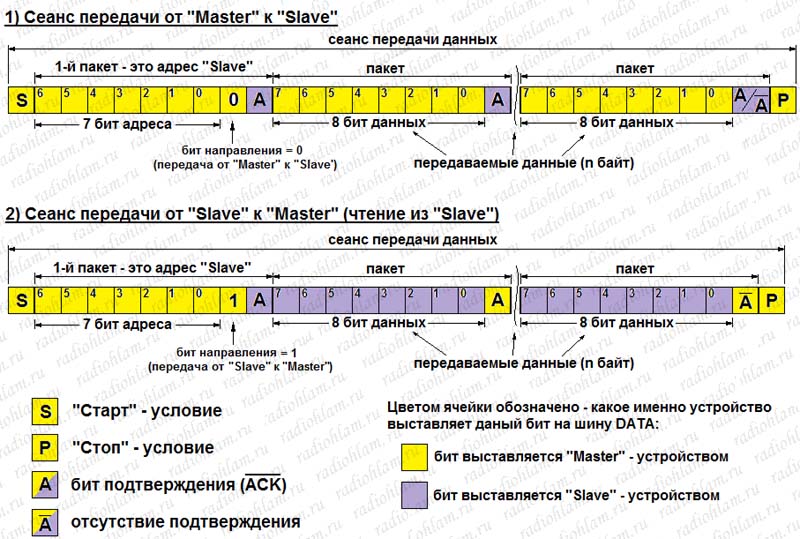


Рисунок 1.8 – Сеанс передачи *Master*-устройство управляет сеансом

Кроме этого, в зависимости от направления передачи данных и *Master* и *Slave* -устройства могут быть «приёмниками» или «передатчиками». Когда *Master* принимает данные от *Slave* – он является «приёмником», а *Slave* – «передатчиком». Когда же *Slave* принимает данные от *Master*, то он уже является «приёмником», а *Master* в этом случае является «передатчиком».

Нельзя путать тип устройства *Master* со статусом «передатчика». Несмотря на то, что при чтении *Master* информации из *Slave*, последний выставляет данные на шину *Data*, делает он это только тогда, когда *Master* ему это разрешит, установкой соответствующего уровня на линии *Clock*. Так что, хотя *Slave* в этом случае и управляет шиной *Data*, — самим обменом всё равно управляет *Master*.

В режиме ожидания обе сигнальные линии (*Data* и *Clock*) находятся в состоянии высокого уровня (притянуты к питанию).

Каждый сеанс обмена начинается с подачи *Master* так называемого Start-условия. «Старт-условие» – это изменение уровня на линии *Data* с высокого на низкий при наличии высокого уровня на линии *Clock*.

После подачи «старт-условия» первым делом *Master* должен сказать с кем он хочет пообщаться и указать, что именно он хочет – передавать данные в устройство или читать их из него. Для этого он выдаёт на шину 7-ми битный адрес *Slave*-устройства, с которым хочет общаться, и один бит, указывающий направление передачи данных .Первый байт после подачи «старт»-условия всегда всеми *Slave* воспринимается как адресация.

Поскольку направление передачи данных указывается при открытии сеанса вместе с адресацией устройства, то для того, чтобы изменить это направление, необходимо открывать ещё один сеанс (снова подавать «старт»-условие, адресовать это же устройство и указывать новое направление передачи).

После того, как *Master* скажет, к кому именно он обращается и укажет направление передачи данных, — начинается собственно передача: *Master* выдаёт на шину данные для *Slave* или получает их от него. Эта часть обмена (какие именно данные и в каком порядке *Master* должен выдавать на шину, чтобы устройство его поняло и сделало то, что ему нужно) уже определяется каждым конкретным устройством.

Заканчивается каждый сеанс обмена подачей *Master* так называемого «стоп»-условия, которое заключается в изменении уровня на линии *Data* с низкого на высокий, опять же при наличии высокого уровня на линии *Clock*. Если на шине сформировано «стоп»-условие, то закрываются все открытые сеансы обмена.

Во время сеанса любые изменения на линии *Data* при наличии высокого уровня на линии *Clock* запрещены, поскольку в это время происходит считывание данных «приёмником». Если такие изменения произойдут, то они в любом случае будут восприняты либо как «старт»-условие (что вызовет прекращение обмена данными), либо как «стоп»-условие (что будет означать окончание текущего сеанса обмена). Соответственно, во время сеанса обмена установка данных «передатчиком» (выставление нужного уровня на линии *Data*) может происходить только при низком уровне на линии *Clock*. Передача состоит из пакетов по девять бит, передаваемых в обычной положительной логике (то есть высокий уровень –это 1, а низкий уровень – это 0). Из них 8 бит передаёт «передатчик» «приёмнику», а последний девятый бит передаёт «приёмник»» «передатчику». Биты в пакете передаются старшим битом вперёд. Последний, девятый бит называется битом подтверждения *ACK* (от английского слова *acknowledge* – подтверждение). Он передаётся в инвертированном виде, то есть 0 на линии соответствует наличию бита подтверждения, а 1 — его отсутствию. Бит подтверждения может сигнализировать как об отсутствии или занятости устройства (если он не установился при адресации), так и о том, что «приёмник» хочет закончить передачу или о том, что команда, посланная *Master*, не выполнена.

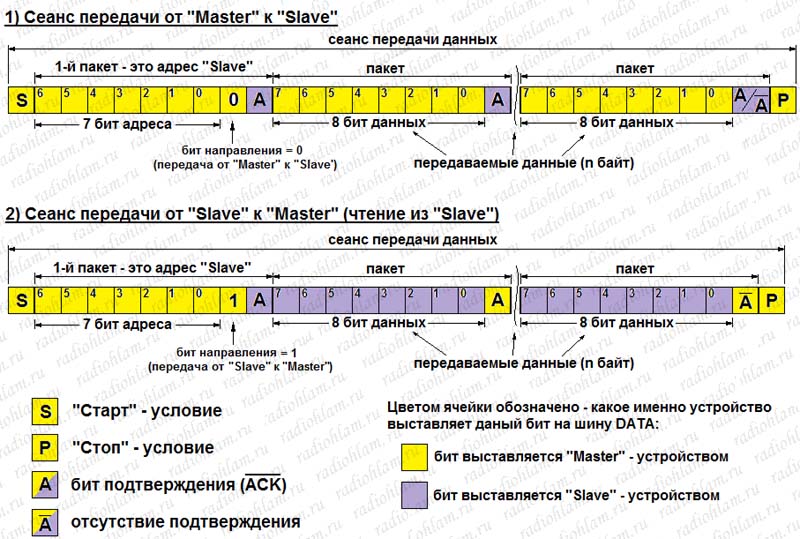


Рисунок 1.9 – Сеанс передачи от *Slave* к *Master*

Каждый бит передаётся за один такт. Та половина такта, во время которой на линии *Clock* установлен низкий уровень, используется для установки бита данных на шину передающим абонентом (если предыдущий бит передавал другой абонент, то он в это время должен отпустить шину данных). Та половина такта, во время которой на линии *Clock* установлен высокий уровень, используется принимающим абонентом для считывания установленного значения бита с шины данных.

На практике, не в производственной среде данный интерфейс практически не используется.[24].

Как уже упоминалось ранее в главах 1.2.2, протокол *CAN* разработан для автомобильной промышленности и впоследствии стал стандартом в области создания бортовых сетей автомобилей, железнодорожного транспорта и т.д. он позволяет создавать сети с развитыми средствами контроля ошибок, скоростью передачи до 1Мбит/с и пакетами содержащими не более восьми байтов данных. В данном протоколе нет строгого определения физического уровня, поэтому для передачи сообщений может использоваться, например, витая пара или оптоволокно. На деле *CAN* реализует канальный уровень, т.е. осуществляет формирование пакетов сообщений, ограничение распространения ошибок, подтверждение приема и арбитража [25].

*SPI* (*Serial Peripheral Interface, SPI bus*) – последовательный синхронный стандарт передачи данных в режиме полного дуплекса, предназначенный для обеспечения простого и недорогого высокоскоростного сопряжения микроконтроллеров и периферии. *SPI* также иногда называют четырёхпроводным (*our-wire*) интерфейсом.

Принцип обмена данными по *SPI* показан на рисунках 1.10-11.

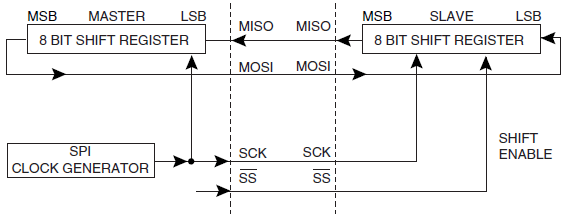


Рисунок 1.10 – Принцип обмена данными по *SPI*

Сигналы, используемые данным интерфейсом, имеют следующее назначение (см. рисунок 1.11):

*MOSI – Master Output / Slave Input*. Выход ведущего / вход ведомого;

*MISO – Master Input / Slave Output*. Вход ведущего / выход ведомого;

*SLK – Serial Clock*. Сигнал синхронизации;

*SS – Slave Select*. Выбор ведомого. [26].

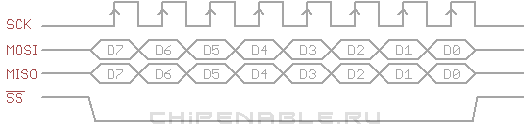


Рисунок 1.11 – Принцип командного обмена данными по *SPI*

Физически *SPI* реализуется на основе сдвигового регистра, который выполняет и функцию передатчика, и функцию приемника.

*RS-232* ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Recommended Standard 232*, другое название *EIA232*) – стандарт [физического уровня](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) для [асинхронного интерфейса (*UART*)](https://ru.wikipedia.org/wiki/UART). Устройство, поддерживающее этот стандарт, широко известно как [последовательный порт](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D1%80%D1%82) [персональных компьютеров](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80). Исторически стандарт имел широкое распространение в [телекоммуникационном](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8) оборудовании. В настоящее время используется для подключения к компьютерам широкого спектра оборудования, нетребовательного к скорости обмена, особенно при значительном удалении его от компьютера и отклонении условий применения от стандартных. В компьютерах, занятых офисными и развлекательными приложениями, практически вытеснен интерфейсом [USB](https://ru.wikipedia.org/wiki/USB).

*RS-232* обеспечивает передачу данных и некоторых специальных сигналов между терминалом ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Data Terminal Equipment, DTE*) и коммуникационным устройством ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Data Communications Equipment, DCE*) на расстояние до 15 метров на максимальной скорости (115200 [бод](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%B4)). Так как этот интерфейс известен не только простотой программирования, но и неприхотливостью, в реальных условиях это расстояние увеличивается во много раз с примерно пропорциональным снижением скорости.

Протокол интерфейса предполагает два режима передачи данных – [синхронный](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и [асинхронный](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), а также два метода управления обменом данных: аппаратный и программный. Каждый режим может работать с любым методом управления. В протоколе также предполагается вариант управления передачей данных по специальным сигналам, устанавливаемым хостом (*DSR* — сигнал состояния готовности, *DTR* — сигнал готовности передачи данных).

Универсальный асинхронный   приёмопередатчик (УАПП, [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Universal Asynchronous Receiver-Transmitter, UART*) — узел вычислительных устройств, предназначенный для организации связи с другими цифровыми устройствами. Преобразует передаваемые данные в последовательный вид так, чтобы было возможно передать их по одной физической цифровой линии другому аналогичному устройству. Метод преобразования хорошо стандартизован и широко применяется в компьютерной технике (особенно во [встраиваемых устройствах](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D1%82%D1%80%D0%B0%D0%B8%D0%B2%D0%B0%D0%B5%D0%BC%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и [системах на кристалле (*SoC*)](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BD%D0%B0_%D0%BA%D1%80%D0%B8%D1%81%D1%82%D0%B0%D0%BB%D0%BB%D0%B5)).

Представляет собой логическую схему, с одной стороны подключённую к шине вычислительного устройства, а с другой имеющую два или более выводов для внешнего соединения.

*UART* может представлять собой отдельную микросхему (например, *Intel* I8251, I8250) или являться частью [большой интегральной схемы](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0) (например, [микроконтроллера](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80)). Используется для передачи данных через [последовательный порт](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D1%80%D1%82) [компьютера](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80).

Передача данных в *UART* осуществляется по одному [биту](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82) в равные промежутки времени. Этот временной промежуток определяется заданной скоростью *UART* и для конкретного соединения указывается в [бодах](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%B4) (что в данном случае соответствует битам в секунду). Существует общепринятый ряд стандартных скоростей: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600 бод. Скорость в бодах иногда называют [сленговым](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%B5%D0%BD%D0%B3) словом бодрейт или битрейт.

Помимо информационных битов, *UART* автоматически вставляет в поток синхронизирующие метки, так называемые стартовый и стоповый биты. При приёме эти лишние биты удаляются из потока. Обычно стартовый и стоповый биты обрамляют один байт информации (8 бит), при этом младший информационный бит передаётся первым, сразу после стартового.

Встречаются реализации *UART*, передающие по 5, 6, 7 или 9 информационных бит. Обрамлённые стартом и стопом биты являются минимальной посылкой. Некоторые реализации *UART* используют два стоповых бита при передаче для уменьшения вероятности рассинхронизации приёмника и передатчика при плотном трафике. Приёмник игнорирует второй стоповый бит, воспринимая его как короткую паузу на линии.

Принято соглашение, что пассивным (в отсутствие потока данных) состоянием входа и выхода *UART* является логическая 1. Стартовый бит всегда является логическим 0, поэтому приёмник *UART* ждёт перепада из 1 в 0 и отсчитывает от него временной промежуток в половину длительности бита (середина передачи стартового бита). Если в этот момент на входе всё ещё 0, то запускается процесс приёма минимальной посылки. Для этого приёмник отсчитывает 9 битовых длительностей подряд (для 8-битных данных), и в каждый момент фиксирует состояние входа. Первые 8 значений являются принятыми данными, последнее значение проверочное (стоп-бит). Значение стоп-бита всегда равно 1. Если реально принятое значение иное, UART фиксирует ошибку.

Для формирования временных интервалов передающий и приёмный *UART* имеют источник точного времени (тактирования). Точность этого источника должна быть такой, чтобы сумма погрешностей (приёмника и передатчика) установки временного интервала от начала стартового импульса до середины стопового импульса не превышала половины (а лучше четверти) битового интервала. Для 8-битной посылки это значение 0,5/9,5 ≈ 5 %. На практике, с учётом возможных искажений сигнала в линии, общая ошибка тактирования должна быть не более 3 %. Поскольку в худшем случае ошибки тактов приёмника и передатчика могут суммироваться, то рекомендуемый допуск на точность тактирования *UART* — не более 1,5 %.

Поскольку синхронизирующие биты занимают часть битового потока, то результирующая пропускная способность *UART* меньше скорости соединения. Например, для 8-битных посылок формата [8-N-1](https://ru.wikipedia.org/wiki/%D0%A3%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B0%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%B8%D1%91%D0%BC%D0%BE%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%82%D1%87%D0%B8%D0%BA#%D0%9A%D0%BE%D1%80%D0%BE%D1%82%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C_%D0%BF%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%BE%D0%B2) синхронизирующие биты занимают 20 % потока, что при физической скорости линии 115 200 [бод](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%B4) означает полезную скорость передачи данных 92 160 [бит/с](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82_%D0%B2_%D1%81%D0%B5%D0%BA%D1%83%D0%BD%D0%B4%D1%83) или 11 520 байт/с.

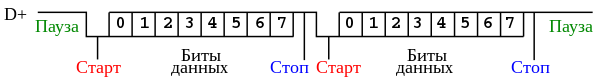
[](https://commons.wikimedia.org/wiki/File:Puerto_serie_Rs232-ru.svg?uselang=ru)

Рисунок 1.12 – Принцип командного обмена данными по *UART*

Логическая схема *UART* имеет входы-выходы с логическими уровнями, соответствующими полупроводниковой технологии схемы: [КМОП](https://ru.wikipedia.org/wiki/%D0%9A%D0%9C%D0%9E%D0%9F), [ТТЛ](https://ru.wikipedia.org/wiki/%D0%A2%D0%A2%D0%9B) и т. д. Такой физический уровень может быть использован в пределах одного устройства, однако непригоден для коммутируемых длинных соединений по причине низкой защищённости от электрического разрушения и помех. Для таких случаев были разработаны специальные [физические уровни](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C), такие как [токовая петля](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%B5%D1%82%D0%BB%D1%8F), [RS-232](https://ru.wikipedia.org/wiki/RS-232), [RS-485](https://ru.wikipedia.org/wiki/RS-485), [LIN](https://ru.wikipedia.org/wiki/LIN) и тому подобные.

Специфической разновидностью физического уровня асинхронного интерфейса является физический уровень [*IrDA*](https://ru.wikipedia.org/wiki/IrDA).

Существуют физические уровни *UART* для сложных сред. В некотором смысле стандартный компьютерный телефонный [модем](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BC) также можно назвать специфическим физическим уровнем асинхронного интерфейса. Существуют специальные микросхемы проводных модемов, сделанных специально как физический уровень асинхронного интерфейса (то есть протокольно прозрачные). Выпускается также радиоканальный физический уровень в виде модулей радиоприёмников и радиопередатчиков.

Для преобразования логических входов-выходов *UART* в сигналы соответствующего физического уровня применяют специальные электронные схемы, именуемые драйверами. Для всех популярных физических уровней существуют интегральные драйверы в виде микросхем.

Логика *UART* обычно позволяет производить одновременную передачу и прием. Эта способность часто обозначается сленговым словом дуплекс. Однако не все физические уровни позволяют одновременно передавать данные в обе стороны. В таких случаях принято говорить о полудуплексной связи. Существуют также решения, в которых передача данных физически возможна только в одну сторону, тогда говорят о симплексной связи.

Изначально *UART* предназначался для связи двух устройств по принципу «точка-точка». Впоследствии были созданы физические уровни, которые позволяют связывать более двух *UART* по принципу «один говорит – несколько слушают». Такие физические уровни называют сетевыми. Существуют реализации типа общая шина (когда все приемопередатчики подключены к одному проводу) и кольцо (когда приемники и передатчики соединяют попарно в замкнутое кольцо). Первый вариант проще и встречается гораздо чаще. Второй вариант сложнее, но надежнее и быстрее: гарантируется работоспособность всех узлов (передающий узел услышит эхо своего сообщения, только если оно успешно ретранслировано всеми узлами); любой узел может начинать передачу в любой момент, не заботясь о риске коллизии. Наиболее известные сетевые физические уровни — [*RS-485*](https://ru.wikipedia.org/wiki/RS-485) и [*LIN*](https://ru.wikipedia.org/wiki/LIN).

**1.2.9 Регистровая модель *I2C*, *SPI* и *UART* микроконтроллерного**

**ядра *Tensilica Xtensa***

При составлении программ весьма важно знать, какие из регистров микропроцессора являются программно-доступными регистрами, в которых можно хранить подлежащие обработке данные (операнды) и управляющие сигналы (команды).

В регистровой модели можно выделить две группы регистров:

– регистры общего назначения (РОН), предназначенные для хранения операндов (в том числе адресных кодов). Эта группа регистров образует внутреннюю память микропроцессора;

– служебные регистры, предназначенные для управления исполняемой программой, обеспечения требуемого режима работы процессора, организации обращения к памяти и выполнения других функций. Состав и количество служебных регистров определяются архитектурой микропроцессора.

Как уже упоминалось ранее в 1.2.2-1.2.4 данный микроконтроллер имеет большое количество периферийных интерфейсов, подробнее остановимся на некоторых из них:

*I²C* – [последовательная](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) асимметричная [шина](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D0%BD%D0%B0_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)) для связи между [интегральными схемами](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0) внутри [электронных приборов](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE). Использует две двунаправленные линии связи (*SDA* и *SCL*), применяется для соединения низкоскоростных [периферийных компонентов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B8%D1%84%D0%B5%D1%80%D0%B8%D0%B9%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE) с [процессорами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) и [микроконтроллерами](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80) (например, на [материнских платах](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D0%BD%D1%81%D0%BA%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D0%B0), во [встраиваемых системах](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D1%82%D1%80%D0%B0%D0%B8%D0%B2%D0%B0%D0%B5%D0%BC%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), в [мобильных телефонах](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD)).

Чип также имеет сопроцессор с низким энергопотреблением, который можно использовать вместо ЦП для экономии энергии при выполнении задач, не требующих больших вычислительных мощностей, таких как мониторинг состояния пинов. *ESP32* объединяет богатый набор периферийных устройств, начиная от емкостных сенсорных датчиков, датчиков Холла, интерфейса *SD-*карты, *Ethernet*, высокоскоростного *SPI, UART, I²S и I²C* [29].

Как таковой аппаратной поддержки передачи по *I²C***-**шине у контроллера нет, нет такого модуля, нет даже специальных регистров для этого. Но в то же время посредством функционала библиотеки *SDK* всё же такая поддержка имеется, хоть и программная. Видимо, это обусловлено тем, что шина *I²C* используется везде для передачи данных и без неё никак нельзя просто обойтись.

*SPI* – последовательный синхронный стандарт передачи данных в режиме полного [дуплекса](https://ru.wikipedia.org/wiki/%D0%94%D1%83%D0%BF%D0%BB%D0%B5%D0%BA%D1%81_(%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8)), предназначенный для обеспечения простого и недорогого высокоскоростного сопряжения микроконтроллеров и периферии. *SPI* иногда называют четырёхпроводным интерфейсом [31].

По стандарту каждое передаваемое слово данных (асинхронный фрейм) требуют наличия стартового бита и как минимум одного стопового бита. Также опционально может присутствовать бит четности, который создается от диапазона бит от 7 до 12. Формат принимаемого и передаваемого фрейма управляется регистром *Line control register (UARTx\_LCR*). Данные всегда передаются так, что самый младший бит слова *(least significant bit, LSB*) передается первым, и за ним идут биты по нарастанию старшинства, самый старший бит слова (*most significant bit, MSB*) передается последним. На рисунке 1.13 оказана физическая диаграмма типичного потока бит, который можно увидеть на выводе TX.



Рисунок 1.13 – Как выглядит поток бит на выводе передачи *TX*

Процессор *BF538* предоставляет набор регистров управления и состояния для каждого *UART*, которые выполнены в соответствии с общепринятыми индустриальными стандартами. Это регистры, отображенные на адресам памяти, так называемые *memory-mapped registers* (*MMR*), которые имеют ширину в 1 байт, и могут быть прочитаны/записаны в общем адресном пространстве памяти как полу-слова (2 байта), где старший байт заполнен нулем.

С целью соответствия со стандартными индустриальными интерфейсами, несколько регистров привязаны к одному адресу. Регистры защелки делителя (*divisor latch registers, UARTx\_DLH* и *UARTx\_DLL*) используют общий адрес с регистром буфера передачи (*transmit holding register, UARTx\_THR*), регистром буфера приема (*receive buffer register, UARTx\_RBR*) и регистром разрешения прерываний (interrupt enable register, *UARTx\_IER*). Бит защелки делителя (*divisor latch access bit, DLAB)* в регистре управления линией *(line control register, UARTx\_LCR*) управляет набором регистров, который доступен в настоящий момент. Для доступа к этим регистрам программное обеспечение может использовать 16-разрядные инструкции загрузки/сохранения (*16-bit word load/store instructions).*

Раздельную буферизацию имеют и канал передачи, и канал приема. Регистр *UARTx\_THR* буферизирует регистр сдвига передачи *(transmit shift register, TSR)* и регистр *UARTx\_RBR* буферизирует регистр сдвига приема (*receive shift register, LSR*). Регистры сдвига недоступны для прямого доступа со стороны программного обеспечения.

Все вышеприведённые регистровые модели довольно популярны и их можно программировать в соответствии с назначением прибора.

**1.2.10 Принципы функционирования контроллера вложенных**

**векторов прерываний микроконтроллера с ядром *Tensilica***

***Xtensa***

Технологии конфигурируемых и расширяемых процессоров Xtensa компании *Tensilica* были обоснованы следующим: фирма начала с разработки нового 32-разрядного процессорного *RISC*-ядра *Xtensa* [18], а затем на его основе создала набор программных средств для автоматизированного расширения базовой архитектуры с целью ее оптимизации с учетом особенностей алгоритма. Одновременно с генерацией описания модифицированной процессорной архитектуры автоматически вносятся изменения и дополнения в набор средств трансляции и отладки.

Коды команд имеют длину 24 бита. Наиболее употребительные команды имеют альтернативную кодировку длиной 16 бит. Это свойство обеспечивает исключительно высокую плотность кода. 24-битовые и 16-битовые команды могут произвольно смешиваться, средняя длина команды в типичной программе составляет около 20 бит (два с половиной байта, против 3,5 для архитектуры x86).

32-битовая адресация обеспечивает доступ к 4-Гбайт физическому адресному пространству, общему для программ и данных. В ходе оптимизации архитектуры системы команд проектировщик системы может использовать следующие возможности:

– выбор расширений процессорной архитектуры из набора опций, предопределенных разработчиками Tensilica. Технология *Fusion* — формирование новых более сложных команд, каждая из которых есть «сплав» нескольких прежних простых. Одна новая сложная команда при этом выполняется быстрее, чем последовательность простых;

– выявление в алгоритме участков с параллелизмом данных (возможностей векторизации при обработке) и добавление в систему команд инструкций и регистров *SIMD (Single Instruction Multiple Data*);

Технология *FLIX* (*Flexible Length Instruction Xtensions*) — выявление в алгоритме участков с возможностью явного параллелизма команд и добавление в набор команд инструкций «со сверхдлинным командным словом» (*Very Large Instruction Word, VLIW)*.

В таблице 1.2 перечислены предопределенные опции, которые можно выбираться при расширении архитектуры (перечень заимствован из [19], разд. 4). Расширения можно выбирать независимо одно от другого, за немногими исключениями, когда одно из расширений обязательно требует наличия другого. В некоторых случаях пары расширений являются взаимно исключающими.

Таблица 1.3 – Перечень предопределенных расширений базовой архитектуры

|  |  |
| --- | --- |
| *The Code Density Option* | Альтернативная (укороченная до 16 бит) кодировка для наиболее часто используемых команд (аналогично архитектурам *ARM* и *MIPS*). |
| *The Loop Option* | Аппаратная поддержка циклических конструкций; счетчик количества повторений цикла и адрес последней команды тела цикла хранятся в специальных регистрах. |
| *The Extended L32R Option* | Расширенная версия команды загрузки регистра из памяти. При адресации используется дополнительный регистр, участвующий в формировании адреса операнда-источника. |
| *The 16-bit Integer Multiply Option* | Команды умножения 16×16 бит без знака и со знаком, с формированием 32-битового результата. |
| *The 32-bit Integer Multiply Option* | Команды умножения 32×32 бит без знака и со знаком, с формированием 64-битового результата. |
| *The 32-bit Integer Divide Option* | Команды знакового и беззнакового деления 32/32 с формированием частного либо остатка. |
| *The MAC16 Option* | Набор команд и дополнительных регистров для выполнения операций «умножение с накоплением». |
| *The Miscellaneous Operations Option* | Набор команд для выполнения действий «с насыщением», для определения максимальных и минимальных значений, для преобразования типов. |
| *The Code Density Option* | Альтернативная (укороченная до 16 бит) кодировка для наиболее часто используемых команд (аналогично архитектурам *ARM* и *MIPS*). |
| *The Coprocessor Option* | Группа дополнительных регистров, которые могут использоваться при добавлении дополнительных конвейеров обработки (сопроцессоров) к исполнительному блоку. |
| *The Timer Interrupt Option* | Добавление аппаратного таймера и до четырех каналов сравнения с уставками, а также запросов прерываний по переполнению таймера и по срабатыванию компараторов. |

Исчерпывающее описание расширений можно найти в объёмном документе [18]. На рис. 1.14 (заимствован из [19], разд. 1) показаны возможности модификации архитектуры процессора *Xtensa LX2*. На рисунке различные части отмечены разными цветами:

– голубой – блоки базовой архитектуры.

– бирюзовый – конфигурируемые блоки.

–светло-зеленый – опциональные блоки, разработанные и оттестированные фирмой *Tensilica*.

–розовый – опциональные блоки *Tensilica* с возможностью реконфигурации.

– зеленый – блоки, полностью определяемые разработчиком СБИС.

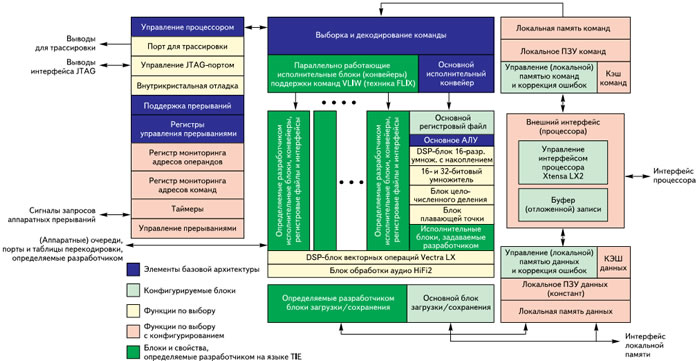


Рисунок 1.14 – Блок-схема архитектуры процессоров *Xtensa LX6*

Средства поддержки проектирования. Для автоматизации процесса реконфигурации фирма *Tensilica* разработала набор программных средств, который обеспечивает не только автоматическое выполнение всех этапов оптимизации, но и широкие возможности по автоматической верификации произведенных изменений и дополнений архитектуры.

Технология конфигурирования процессорной архитектуры иллюстрируется диаграммой, приведенной на рис. 1.15 (заимствована из [19], разд. 1), также отмечены программные компоненты поддержки оптимизации и автоматической генерации изменений. Главные из них коротко будут описаны далее.

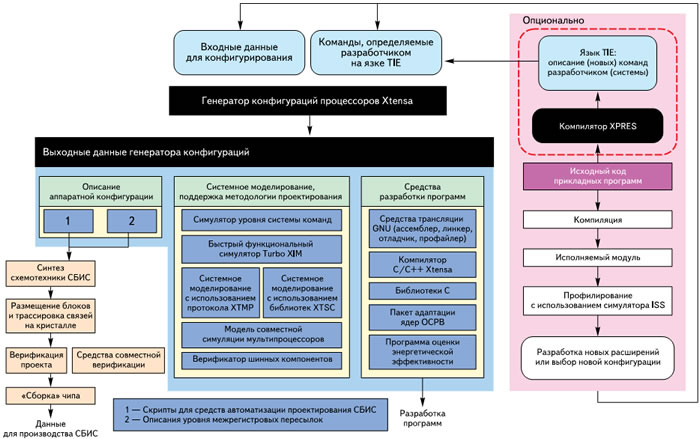


Рисунок 1.15 –Конфигурирование процессора *Xtensa*

Основное программное средство, используемое для оптимизации процессора, – компилятор *XPRES (Xtensa PRocessor Extension Synthesis*), предназначением которого является автоматизированная генерация расширений системы команд для архитектуры *Xtensa*. Этот процесс осуществляется за четыре этапа, каждый из которых может быть выполнен полностью автоматически, но допускает и предполагает вмешательство разработчика:

1 Компиляция исходного кода с языка С/С++ в базовую систему команд процессора *Xtensa*:

– профилирование (анализ времени выполнения) с выделением циклических участков;

– анализ возможностей векторизации данных для отдельных фрагментов программы;

– формирование рекомендаций по улучшению исходного кода для программиста.

2 Формирование набора вариантов расширений архитектуры процессора и базовой системы команд:

– возможности использования команд со структурой SIMD;

– возможности формирования команд со сверхдлинным командным словом (VLIW);

– рекомендации по свойствам слотов в таких командах;

– оценка соотношения между ростом производительности и дополнительными аппаратными затратами для каждого варианта расширений архитектуры;

– выбор наилучшего набора расширений.

3 Необязательный этап дополнительной оптимизации, в ходе которой оператор может проверить эффективность дополнительных вариантов модификации процессора (выполняется в диалоговом режиме с использованием возможностей программных средств оценки эффективности).

4 Автоматическая модификация программных средств трансляции, симуляторов, технологической документации для систем поддержки проектирования и производства СБИС с учетом внесенных в процессор изменений.

В ходе анализа *XPRES* способен за короткое время просмотреть большое количество вариантов модификации свойств процессора, оценить как достижимое ускорение, так и затраты: дополнительную площадь на кристалле, дополнительное количество логических вентилей, увеличение потребляемой микросхемой мощности, изменение размера кода при использовании расширенной системы команд и другие характеристики. Разработчик имеет возможность просмотреть все варианты модификации и выбрать тот, который кажется ему наилучшим [23].

**1.2.11 Методика организации обработки прерываний интерфейса**

***UART***

В самом начале процедуры передатчик отправляет линию в низкий уровень — это старт бит. Зафиксировав что линия просела, приемник выжидает интервал Т1 и считывает первый бит, потом через интервалы Т2 считываются остальные биты. Последний бит это стоп бит, говорящий о том, что передача этого байта завершена. В конце байта, перед стоп битом, может быть и бит четности. Который получается если поксорить между собой все биты, для контроля качества передачи. Также может быть два стопа, опять же для надежности. Битов может быть не 8, а 9. О всех этих параметрах договариваются на берегу, до начала передачи. Самым же популярным является 8 бит, один старт один стоп, без четности.

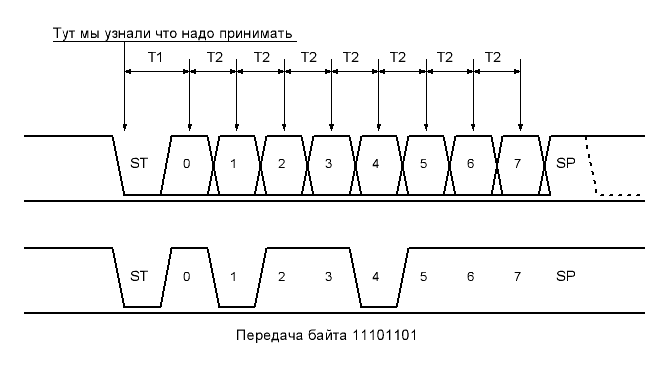


Рисунок 1.16 – Прием-передача битов при прерываниях

Причем все реализовано аппаратно. При необходимости завести второй *UART*, тогда придется делать его программно.

По такому же протоколу работает *COM* порт компьютера, разница лишь в разнице напряжений, поэтому именно этот протокол я буду использовать для связи микроконтроллера с компом. Для преобразования напряжений [можно использовать *RS232-TTL* конвертер или применить](http://easyelectronics.ru/svyaz-mikrokontrollera-s-kompyuterom-cherez-rs232.html) встроенный в [*Pinboard*](http://shop.easyelectronics.ru/index.php?productID=147)мост *USB-UART*, который образовывает в системе виртуальный *COM PORT*.

Аппаратная часть.Ну тут все просто, соединяем крест-накрест приемник с передатчиком и готово.

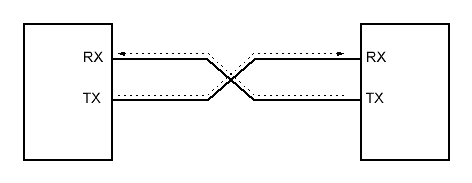


Рисунок 1.17 – Протокол работы *COM-*порта

Внутри находится накопительный сдвиговый регистр, в котором происходит сборка байта из битов и регистры данных *UDR*, куда этот бит передается. Такая структура исключает возможность считать не до конца полученный байт. Буфер приема состоит из двух байт, что позволяет ему держать два в уме и один еще засасывать в сдвиговый регистр.

У *AVR* есть регистр *UDR* это *UART Data Register* (в некоторых контроллерах он может зваться *UDR0* или аналогично). На деле это два разных регистра, но имеют один адрес. Просто на запись попадает в один (регистр передатчика), а на чтение берет из другого (регистр приемника). Таким образом достаточно помещать данные в этот регистр и они уйдут приемнику, и, наоборот, считывать их оттуда по приходу.

О том, что байт пришел в регистр *UDR* сообщает прерывание по завершению приема, которое вызывается сразу же, как приемник обнаружит все биты. Поскольку передача идет довольно медленно, то большие данные помещать в регистр *UDR* нельзя — нужно дождаться окончания передачи предыдущего байта. О том, что *UDR* пуст и готов к приему нового байта сигнализирует бит *UDRE*, он же вызовет аппаратное прерывание по опустошению буфера.

Настройка *UART*. Все настройки приемопередатчика хранятся в регистра конфигурации. Это *UCSRA, UCSRB* и *UCSRС*. А скорость задается в паре *UBBRH:UBBRL*.

Регистр UCSRA. Здесь для работы нужны только биты *RXC* и *TXC –* это флаги завершения приема и передачи, соответственно. *RXC* встанет когда непрочитанный байт вылезет в регистр *UDR*, а *TXC* встает когда последний стоп-бит прошел, а новое значение в *UDR* не поступило. Т.е. после прохода всех байтов.

Также одновременно с этими флагами вызывается прерывание (если оно было разрешено). Сбрасываются они аппаратно – принимающий после чтения из регистра *UDR*, передающий при переходе на прерывание, либо программно (чтобы сбросить флаг программно в него надо записать 1).

Биты *UDRE* сигнализирует о том, что регистр *UDR* приемника пуст и в него можно поместить новый байт. Сбрасывается он аппаратно после засылки в *UDR* какого либо байта. Также генерируется прерывание «Регистр пуст»

Бит *U2X* — это бит удвоения скорости при работе в асинхронном режиме. Его надо учитывать при расчете значения в *UBBRH:UBBRL*

Регистр *UCSRB*. Тут в первую очередь это биты *RXEN* и *TXEN* – разрешение приема и передачи. Стоит их сбросить как выводы *UART* тут же становятся обычными ножками I/O.

Биты *RXCIE, TXCIE, UDRIE* разрешают прерывания по завершению приема, передачи и опустошении буфера передачи UDR.

Регистр *UCSRC* Бит селектора *URSEL* дело в том, что по неизвестной причине создатели решили сэкономить байт адреса и разместили регистры *UCSRC* и *UBRRH* в одной адресном пространстве. Для определения места записи необходимо следить за старшим битом. Данный момент необходимо смотреть в даташите этот момент — есть там бит *URSEL* или нет.

Остальные биты задают число стопов, наличие и тип контроля четности. Если оставить все по дефолту то будет стандартный режим. Надо только выставить формат посылки. Делается это битами *UCSZ0, UCSZ1* и UCSZ2 (этот бит тусуется в регистре *UCSRB*). Для стандартной 8ми битной посылки туда надо записать две единички.

Скорость обмена. Тут все зависит от пары *UBBRx.* Вычисляется требуемое значение по формуле 1.1 для *U2X*=0 и для *U2X*=1 :

UBBR=XTAL/(16 · baudrate) – 1 (1.1)

UBBR=XTAL/(8 · baudrate) – 1 (1.2)

где: *XTAL* – рабочая тактовая частота контроллера.

*baudrate* –требуемая скорость, с тем условием что чем медленней тем надежней. 9600 в большинстве случаев хватает).

Также возможны ошибки при приеме. За них отвечают флаги в регистре *UCSRA.*

*FE* — ошибка кадрирования. Т.е. мы ждали стоп бит, а пришел 0.

*OR* — переполнение буфера. То есть данные поступают, а из UDR их считать не предоставляется возможным .

*PE* — не совпал контроль четности.

Работа на прерываниях Решение тут одно – использование прерываний хотя бы на прием и на передачу. Самое часто использованное прерывание TX\_OK нужно для оповещения какого-либо события повесить. Оно сработает когда произойдет два события: *UDR* кончится и в сдвиговом регистре *USART* все биты уйдут в провод. В принципе, многие юзают его вместо *UDRE*, но тогда между двумя уходящими байтами будет промежуток периодом в 1 байт, что является не особо грамотным использованием памяти.

**1.2.12 Архитектура микроконтроллера *PN532***

**1.2.13 Структура, логика функционирования и интерфейсы связи**

**адаптера *PN532\_NFC\_RFID\_Module* от *Adafruit***

**1.2.14 Работа адаптера *PN532\_NFC\_RFID\_Module* в режимах**

**считывания и записи *NFC*-меток**

**1.2.15 Стандарты *ASCII* и *UNICODE* хранения символов**

**1.2.16 Форматы представления строк в памяти микроконтроллера**

**1.2.17 Обобщённая методика декодирования сообщений**

**1.2.18 Алгоритм формирования очереди сообщений, принимаемых**

**от адаптера *NFC*-меток**

**1.2.19 Принципы функционирования блока *DMA* прямого доступа**

**к памяти**

**1.2.20 Применение *DMA* в формировании очереди входных**

**сообщений**

**1.2.21 Анализ функций стандартной библиотеки «*string.h*»;**

**1.2.22 Алгоритм декодирования сообщений адаптера *NFC*-меток**

**1.2.23 Структура, логика функционирования и организация**

**сдвиговой линейки на основе сдвигового регистра *HC595***

Рассмотрим ситуацию, когда вам нужно больше выходов, чем может предложить микроконтроллер. В этом случае самый простой выход – использовать сдвиговый регистр.

HC595 – восьмиразрядный сдвиговый регистр с последовательным вводом, последовательным или параллельным выводом информации, с триггером-защелкой и тремя состояниями на выходе. Другими словами этот регистр позволяет контролировать 8 выходов, используя всего несколько выходов на самом контроллере. При этом несколько таких регистров можно объединять последовательно для каскадирования.

В данной схеме используется принцип синхронизированной последовательной передачи сигнаналов. Необходимые значения сигнала передаются в регистр один за другим, при этом регистр получает синхронизирующий сигнал, который заставляет его считать сигнал с входа.

*HC595* может отдавать сигналы не только параллельно, но и последовательно. Это необходимо при объединении нескольких регистров, для получения 16 и более выходов.

Три возможных состояния на выходе, упомянутые выше, означают, что выход регистра может иметь не только логический ноль или единицу ([*HIGH* или *LOW*](http://arduino.ru/Reference/Constants)), но и быть в высокоомном состоянии – когда выход отключен от схемы. В высокоомное состояние не может быть переведен отдельный выход, а только все выходы регистра разом.

После подачи питания на регистр и активации входов, на схему на выходах появятся случайные значения.

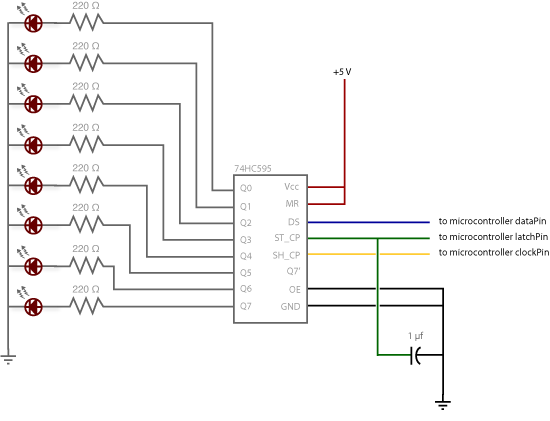


Рисунок 1.32 – Схема подключения

Можно контролировать пин *MR* и *OE* непосредственно с микроконтроллером, чтобы обнулить входы и/или подключить выходы в нужный момент. Для упрощения схемы и минимизации количества задействованных выходов стоит использовать более простую схему, так как значения регистров и выводов будут перезаписаны, как только программы начнет работать [39].

**1.2.24 Структура и логика функционирования дисплейного модуля**

***HY32D* на базе видеопроцессора *ILI9341***

**1.2.25 Структура и логика функционирования контроллеров**

***ADS7846* и *STMPE811* сенсорной панели**

**1.2.26 Методика организации пользовательского интерфейса**

**1.2.27 Принципиальные основы и схемы зарядки литий-ионных**

**аккумуляторных батарей**

[Литий](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%82%D0%B8%D0%B9)[-ионный](https://ru.wikipedia.org/wiki/%D0%98%D0%BE%D0%BD) аккумулятор (*Li-ion*) – тип [электрического аккумулятора](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BA%D0%BA%D1%83%D0%BC%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80), который широко распространён в современной [бытовой электронной технике](https://ru.wikipedia.org/wiki/%D0%91%D1%8B%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B0) и находит своё применение в качестве источника энергии в [электромобилях](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C) и накопителях энергии в энергетических системах. Это самый популярный тип аккумуляторов в таких устройствах как [сотовые телефоны](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD), [ноутбуки](https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%83%D1%82%D0%B1%D1%83%D0%BA), [цифровые фотоаппараты](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D1%84%D0%BE%D1%82%D0%BE%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82), [видеокамеры](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE%D0%BA%D0%B0%D0%BC%D0%B5%D1%80%D0%B0) и электромобили[46].Внутренние электрохимические процессы протекают одинаково и не зависят от форм-фактора и исполнения полости.

Наиболее правильным способом заряда литиевых аккумуляторов является заряд в два этапа. Несмотря на более сложный контроллер заряда, это обеспечивает более полный заряд *li-ion* аккумуляторов, не снижая срока их службы. Речь идет о двухэтапном профиле заряда литиевых аккумуляторов, сокращенно именуемым *CC/CV* (*constant current, constant voltage*). Итак, рассмотрим оба этапа заряда подробнее:

1. На первом этапе должен обеспечиваться постоянный ток заряда. Величина тока составляет 0.2-0.5С. Для ускоренного заряда допускается увеличение тока до 0.5-1.0С (где С – это емкость аккумулятора).

Для обеспечения постоянного зарядного тока заданной величины, схема зарядного устройства (ЗУ) должна уметь поднимать напряжение на клеммах аккумулятора. По сути, на первом этапе ЗУ работает как классический стабилизатор тока.

В момент, когда напряжение на аккумуляторе поднимется до значения 4.2 вольта, аккумулятор наберет приблизительно 70-80% своей емкости (конкретное значение емкости будет зависит от тока заряда: при ускоренном заряде будет чуть меньше, при номинальном – чуть больше). Этот момент является окончанием первого этапа заряда и служит сигналом для перехода ко второму (и последнему) этапу.

2. Второй этап заряда – это заряд аккумулятора постоянным напряжением, но постепенно снижающимся (падающим) током.

На этом этапе ЗУ поддерживает на аккумуляторе напряжение 4.15-4.25 вольта и контролирует значение тока.

По мере набора емкости, зарядный ток будет снижаться. Как только его значение уменьшится до 0.05-0.01С, процесс заряда считается оконченным.

Важным нюансом работы правильного зарядного устройства является его полное отключение от аккумулятора после окончания зарядки. Это связано с тем, что для литиевых аккумуляторов является крайне нежелательным их длительное нахождение под повышенным напряжением, которое обычно обеспечивает ЗУ.

Предварительный этап заряда – этот этап используется только для глубоко разряженных аккумуляторов (ниже 2.5 В) для вывода их на нормальный эксплуатационный режим. На этом этапе заряд обеспечивается постоянным током пониженной величины до тех пор, пока напряжение на аккумуляторе не достигнет значения 2.8 В.

Все схемы, приведенные ниже, подходят для зарядки любого литиевого аккумулятора, остается только определиться с зарядным током и элементной базой. Схема простого зарядного устройства на основе микросхемы *LM317* с индикатором заряда (см.рис 1.40):

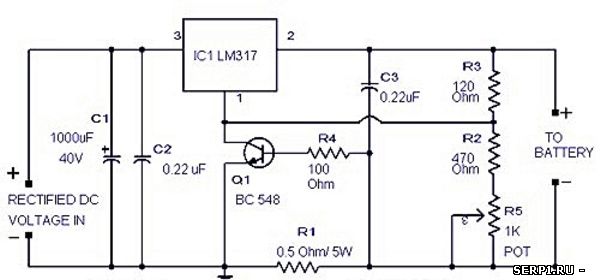


Рисунок 1.40 – зарядное устройство на основе микросхемы *LM317*

Вся настройка сводится к установке выходного напряжения 4.2 вольта с помощью резистора *R8* (без подключенного аккумулятора) и установке тока заряда путем подбора резисторов *R4*, *R6*. Мощность резистора *R1* – не менее 1 Ватт. Микросхема *lm317* широко применяется в различных стабилизаторах напряжения и тока (в зависимости от схемы включения).

*MAX1551/MAX155*5 – специализированные зарядные устройства для Li+ аккумуляторов, способные работать от *USB* или от отдельного адаптера. Единственное отличие этих микросхем – *МАХ1555* выдает сигнал для индикатора процесса заряда, а *МАХ1551* – сигнал того, что питание включено.

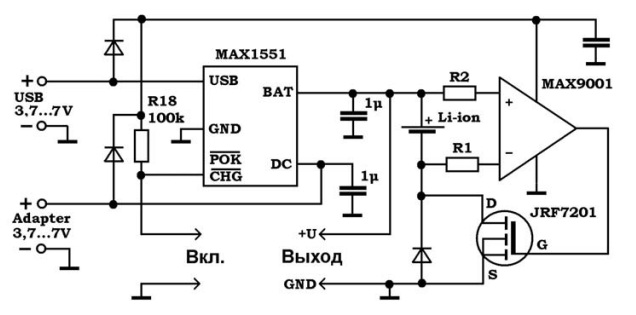


Рисунок 1.41 – Зарядное устройство на основе микросхемы *MAX1551*

Максимальное входное напряжение от *DC*-адаптера – 7 В, при питании от *USB* – 6 В. При снижении напряжения питания до 3.52 В, микросхема отключается и заряд прекращается. Микросхема сама детектирует на каком входе присутствует напряжение питания и подключается к нему.

Есть еще одна незаслуженно обделенная вниманием микросхема от компании *Microchip* – *MCP73812*. На ее базе получается очень бюджетный вариант зарядки [48].

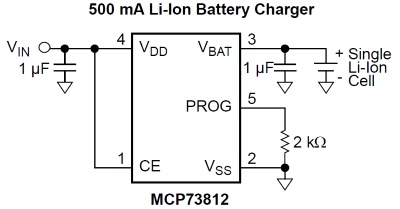


Рисунок 1.42 – Зарядное устройство на основе микросхемы *MCP73812*

Аналогами микросхем, приведенных выше являются: *GL317, SG31, SG317, UC317T, ECG1900, LM31MDT, SP900, КР142ЕН12, КР1157ЕН1*.

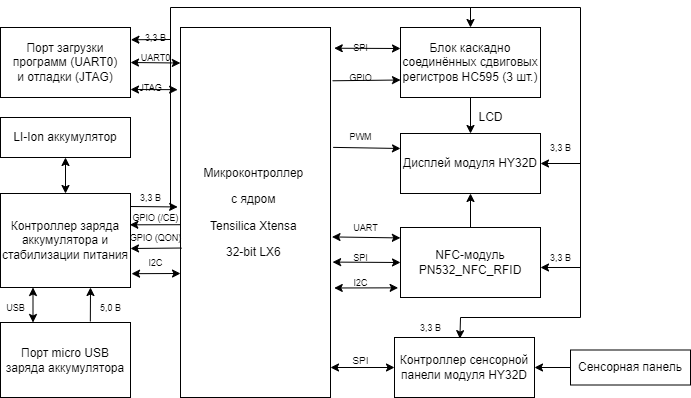
**2 РАЗРАБОТКА СТРУКТУРНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ**

**МОБИЛЬНОГО СКАНЕРА *NFC-*МЕТОК**

**2.1 Обоснование базовых блоков структурной схемы сканера**

***NFC-*меток**

В главе 1.1 были выбраны компоненты для проектирования схем прибора (см.рис.1.1). Структурная схема представляет собой набор блоков, представляющих собой основные функциональные части изделия, а также набор связей, отражающих взаимодействие между ними. Основной задачей структурной схемы является отображение принципа работы изделия в самом общем виде.Схема данного прибора из множества логически обособленных блоков, выполняющих определённые функции:



1 Порт *USB Bq24295* внутрисхемной отладки и зарядки аккумулятора представляет собой блок, объединяющий порт *USB* для подключения источника питания к стенду для зарядки аккумуляторной батареи или для подключения стенда к компьютеру для загрузки / отладки программы.

3. *Li-Ion* аккумулятор на 18650 Ватт, тип [электрического аккумулятора](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BA%D0%BA%D1%83%D0%BC%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80), который широко распространён в современной [бытовой электронной технике](https://ru.wikipedia.org/wiki/%D0%91%D1%8B%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B0) и находит своё применение в качестве источника энергии в [электромобилях](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C) и накопителях энергии в энергетических системах;

4. *MicroSD* является блоком, состоящим из разъёма для *SD-*карты, а также нескольких пассивных элементов для обеспечения её стабильной работы. Используя данный блок, возможно ознакомиться с интерфейсом *SDIO*, основополагающими принципами архитектур построения устройств, предназначенных для хранения данных, с особенностями реализации файловых систем, а также с методами сжатия и создания файлов для хранения информации

5. Сдвиговый регистр *HC595*, их использование приводит к увеличению количества выходов микроконтроллера при помощи сдвигового регистра, причём что это не требует больших затрат.

6. Дисплей модуля *HY32D*. Жидкокристаллический дисплей, чаще всего представлен в размере 3,2-дюйма, с сенсорным экраном, представляющий собой матрицу пикселей размером 800x480, оснащённую контроллером без видеопамяти. Данный блок предоставляет возможность для вывода графической информации на дисплей. Предназначен для взаимодействия с прибором;

7. MPC2551 CAN-приемопередатчик, шина *CAN* характеризуется высокими скоростью передачи данных и помехоустойчивостью, а также способностью обнаруживать любые возникающие ошибки.

8. [Cистема на кристалле](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BD%D0%B0_%D0%BA%D1%80%D0%B8%D1%81%D1%82%D0%B0%D0%BB%D0%BB%D0%B5) *ESP32-D2WD* с интегрированным [*Wi-Fi*](https://ru.wikipedia.org/wiki/Wi-Fi) и [*Bluetooth*](https://ru.wikipedia.org/wiki/Bluetooth) контроллерами и антеннами. В серии *ESP32* используется микроконтроллерное ядро [*Tensilica*](https://ru.wikipedia.org/w/index.php?title=Tensilica&action=edit&redlink=1)[*Xtensa*](https://ru.wikipedia.org/w/index.php?title=Xtensa&action=edit&redlink=1) *LX6* в вариантах с двумя и одним ядром. В систему интегрирован радиочастотный тракт: [симметрирующий трансформатор](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%80%D1%83%D1%8E%D1%89%D0%B8%D0%B9_%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%BE%D1%80&action=edit&redlink=1), встроенные антенные коммутаторы, радиочастотные компоненты, [малошумящий усилитель](https://en.wikipedia.org/wiki/Low-noise_amplifier), усилитель мощности, фильтры и модули управления питанием.

9. Керамический антенный модуль *2450AT18B100E* является базовой единицей построения сложных антенных решеток, позволяющий улучшить технологичность сборки, повысить точность и плотность размещения гидроакустических датчиков, либо просто принимать информацию с радиоволн

10. Кварцевый резонатор *Cristal 2520* это электронный прибор, в котором [пьезоэлектрический эффект](https://ru.wikipedia.org/wiki/%D0%9F%D1%8C%D0%B5%D0%B7%D0%BE%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82) и явление [механического резонанса](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%80%D0%B5%D0%B7%D0%BE%D0%BD%D0%B0%D0%BD%D1%81) используются для построения [высокодобротного](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%B1%D1%80%D0%BE%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C) резонансного элемента электронной схемы

11. Контроллер заряда *BQ24296RGER* и набор стабилизаторов питания на 3,3 В, 2,8 В, 2,5 В и 1,2 В. Задачами для данного блока являются зарядка аккумулятора, а также предоставление стабильного питания необходимой величины для других блоков схемы.

Структурная электрическая схема представляет собой документ, содержащий все основные функциональные части устройства (элементы, устройства и функциональные группы) и основные взаимосвязи между ними. Графическое построение схемы должно обеспечивать наилучшее представление о последовательности взаимодействия функциональных частей в устройстве. На линиях взаимосвязей рекомендуется стрелками обозначать направление хода процессов, происходящих в устройстве.  
Основными составляющими структурной электрической схемы являются следующие:  
1 Микроконтроллер. Он является центральной частью устройства, представляющую собой микросхему, которая отвечает за логику работы в целом и определяет предельно допустимые параметры.  
2 Порт MicroUSB. С его помощью будет происходить зарядка аккумуляторной батарее через блок зарядки аккумуляторной батареи BQ24295.  
3 Контроллер заряда и защиты аккумулятора. Основной его функцией является поддержка постоянного питания в схеме, когда батарея разряжается или полностью. Кроме этого, блок зарядки содержит светодиод, показывающий состояние заряда батареи.  
4 Порт загрузки и отладки программ. Он предназначен для загрузки программы в устройство.  
5 Стабилизатор питания. Данный компонент предназначен для контроля входного напряжения для других компонентов ( от 3 до 5В).  
6 Преобразователи напряжения. Данный компонент состоит повышающего преобразователя DC-DC для преобразования напряжения на дисплей, сканер BAR-кодов и NFC-меток. Также в этот компонент входят инвертирующий преобразователь напряжения DC-DC и делитель напряжения для управления контрастностью дисплея.  
7 Дисплей. При помощи него выводится информация после анализа поступивших данных.  
8 Адаптер считывания BAR-кодов. Он предназначен для считывания информации с BAR-кодов и её передачи в микроконтроллер.  
9 Адаптер считывания NFC-меток. При помощи него считывается информация с NFC-метки и передается в микроконтроллер.

**2.2 Обоснование связей структурной схемы сканера *NFC-*меток**

Микроконтроллер STM32F103C8T6 является входной точкой к практически всем элементам структуной схемы, поскольку именно он отвечает за локигку функционирования саставных частей устройства, обрабатывая, анализируя и определяя их разрешения на то или иное событие.  
Контроллер заряда аккумуляторной батареи связан с микроконтроллеро по порту miniUSB.  
Протокол последовательной отладки проводов (SWD), который представляет собой двухконтактный последовательный интерфейс служит для загрузки и отладки программы.  
Дисплей подключен к микроконтроллеру при помощи шины I2C с соответствующим адаптером для дисплея. Данная шина является ассиметричной и предназначается для связи между интегральными схемами внутри электронных приборов. Она использует две двунаправленные линии связи(SDA и SCL), которые применяются для соединения низкоскоростных периферийных компонентов с процессорами и микроконтроллерами.  
Для связи с адаптером считывания BAR-кодов используется шина UART, которая состоит из двух выводов RX и TX. При помощи вывода RX происходит принятие данных, а при помощи TX передача. Таким образом для связи двух устройств необходимо чтобы вывод RX одного устройства соединить с выводом TX другого устройства.  
Также одним из важных взаимодействий является соединение адаптера считывания NFC-меток с микроконтроллером при помощи шины SPI. Данная шина является четырехпроводной, с последовательным синхронным интерфейсом передачи данных. Он использует следующие сигналы:  
– MISO (master in slave out) – вход ведущего, выход ведомого;  
– MOSI (master out slave in) – выход ведущего, вход ведомого;  
– SCK (serial clock) – сигнал тактирования;  
– SS (slave select) – сигнал выбор ведомого.

**3 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ СКАНЕРА *NFC-*МЕТОК**

**3.1 Обоснование выбора САПР для разработки принципиальной**

**электрической схемы**

В качестве САПР для разработки проекта отладочного стенда была выбрана программа *Altium Designer*.

*Altium Designer* – это система автоматизированного проектирования, предоставляющая широчайшие возможности по созданию электронных устройств [50].

Состав программного пакета *Altium Designer* включает весь необходимый набор инструментов для создания, редактирования и правки работ на основе электрических и программируемых интегральных схем.

Редактор схем позволяет работать с проектами любого размера и сложности, преобразовывая их в простейшие подблоки. Цифро-аналоговое моделирование учитывает почти все реальные параметры и предоставляет в распоряжение конструктора огромное количество различных анализов, включая анализы переходных процессов, частотный, шумов, передаточных функций, Фурье, методом *Monte-Carlo*, с изменением значений температуры.

На схемотехническом уровне проверяются и устраняются различные импедансы и перекрестные отражения. Редактор печатных плат программы содержит уникальные средства для автоматического и интерактивного размещения компонентов. Топологический трассировщик *Situs* использует полностью настраиваемый алгоритм для решения задач разводки печатных плат с большой плотностью установки элементов. Он может работать по неортогональным направлениям и с самостоятельным выбором слоев. Постоянно обновляемые библиотеки программы хранят более 90 тысяч компонентов. Многие из них имеют модели посадочных мест, *IBIS* и *SPICE*-модели, а также *3D*-модели. Каждую из них можно создать в программе самостоятельно с минимальными затратами времени путем последовательного ввода сведений о компоненте.

**3.2 Описание используемых библиотечных элементов и процесса**

**их создания.**

Для большинства стандартных компонентов, таких как резисторы, конденсаторы, диоды, катушки индуктивности, корпуса и посадочные места являются стандартными. Поэтому нет необходимости в ручной отрисовке посадочных мест и условных графических обозначений, а возможно использовать готовую открытую библиотеку компонентов, разработанную в соответствии со стандартами ЕСКД [7].

Для остальных компонентов, таких как микросхемы, держатель батареи, антенна, библиотечные элементы создаются вручную, либо с использованием инструмента *IPC* *Compliant* *Footprint* *Wizard*, который позволяет автоматически создавать посадочные места стандартных типов корпусов на основе данных от производителя компонента.

Процесс создания библиотечных элементов проходит по общему алгоритму для каждого компонента, но различаться лишь размерами корпуса, существующих и представленных в официальном документе условных графических обозначений, а также количеством выводов:

1 Создание УГО. Сначала создается схематическое отображение элемента в SchLib т.е. его отображение, которое появляется при добавлении элемента на SchDoc. Основной задачей является правильно расставить выводы элемента, так чтобы на схеме было удобно подключать его к другим компонентам.

2 Создание посадочного места с прикреплением трехмерной модели. Alitum Designer предоставляет возможность создавать посадочные места в PcbLib вручную, либо же загружать их со встроенных, либо сторонних библиотек.

Все элементы, использованные во время проекта, создавались по следующему алгоритму:

– создание обозначения элемента на принципиальной схеме;

– создание посадочного места компонента;

– добавление *3D-*модели.

В качестве примера для описания процесса создания обозначения элемента на принципиальной схеме был выбран процессор *ESP32-D2WD* (см. рисунок 3.1).

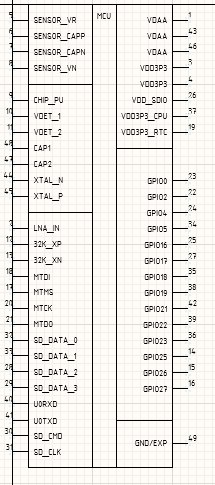


Рисунок 3.1 – Обозначение процессора *ESP32-D2WD* на схеме

Для начала необходимо нарисовать контур элемента, используя инструмент прямоугольник. Далее необходимо добавить выводы компонента и для удобства сгруппировать их по выполняемым ими функциям. Завершающим этапом является запись наименования компонента в поле *Comment*.

Далее необходимо создать посадочное место для компонента и привязать к нему *3D-*модель. На этом этапе все компоненты можно условно разделить на две группы: компоненты, для которых возможно сгенерировать посадочное место и *3D-*модель, используя инструменты *Altium Designer*, например, *Footprint Wizard*, и компоненты, для которых необходимо вручную начертить посадочное место и отдельно добавить к нему *3D-*модель.

Примером компонента из первой группы является корпус микроконтроллера. Процесс его создания заключается в запуске *Footprint Wizard* (см. рисунок 3.2), выборе типа корпуса и вводе размеров из документации на микроконтроллер.

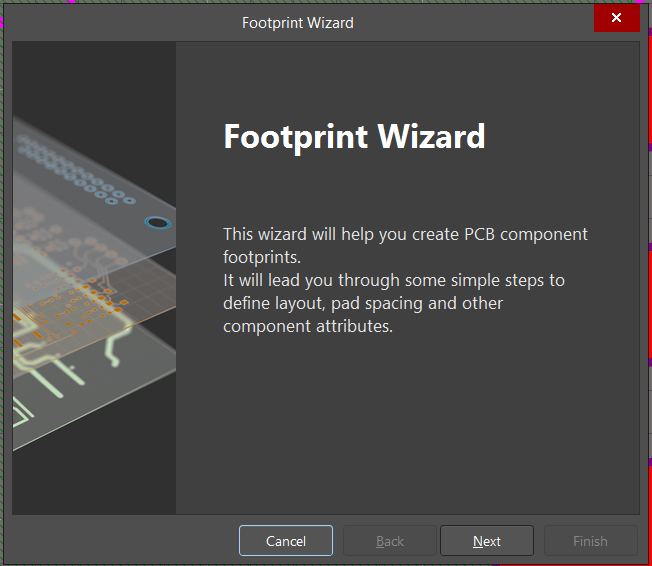


Рисунок 3.2 – *Footprint Wizard*

В результате будет получено посадочное место, изображённое на рисунке 3.3.

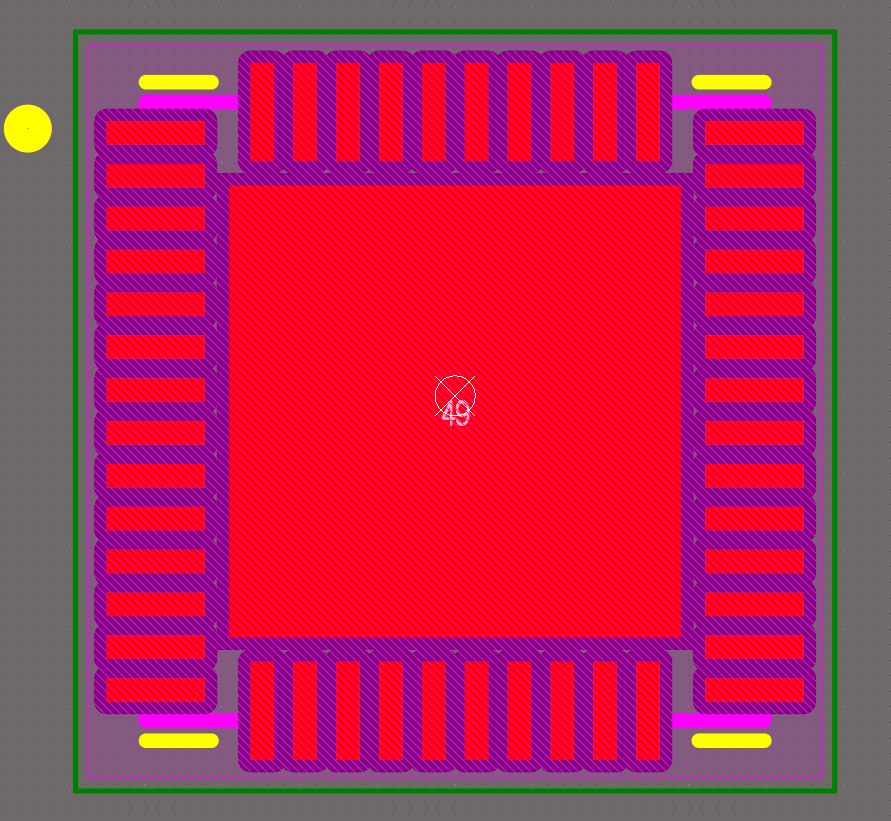


Рисунок 3.3 – Посадочное место для корпуса

А также автоматически будет сгенерирована *3D-*модель, показанная на рисунке 3.4.

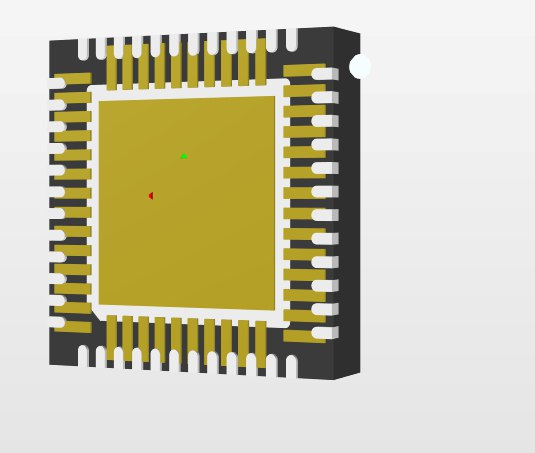


Рисунок 3.4 – *3D-*модель корпуса

Примером компонента из второй группы служит непосредственно смоделированная плата. Процесс её создания начался с расстановки отверстий для выводов согласно документации. В результате была получена схема печатной платы, с указанием всех элементов(см. рис. 3.5).

**3.3 Обоснование выбора базовых компонентов принципиальной**

**схемы сканера *NFC-*меток**

**3.4 Обоснование связей принципиальной электрической схемы**

**сканера *NFC-*меток.**

Так как данный стенд подразумевается как портативное устройство, то ходе проектирования было решено, что стенд будет состоять из двух плат, скрепляемых разъёмами. На верхней плате будут расположены камера, дисплей, акселерометр, микрофон и светодиоды. Остальные же компоненты будут расположены на нижней плате. Таким образом были разработаны две принципиальные схемы: для верхней и для нижней плат соответственно. Обе схемы являются вертикально иерархическими. Это значит, что принципиальные схемы разных блоков стенда расположены на отдельных листах, соединение которых происходит на главном листе с помощью портов. Такой подход был выбран в связи с большим количеством деталей, чтобы не перегружать принципиальную схему и соответственно повысить её читаемость.

**3.5 Анализ и обоснование принципиальной электрической схемы**

**зарядки аккумуляторной батареи.**

При проектировании схемы зарядки аккумуляторной батареи была использована рекомендуемая производителем принципиальная схема из документации на контроллер питания [5]. Также, приняв во внимание ёмкость батареи, которая равна 1200 мА\*ч, было решено ограничить максимальный ток, потребляемый контроллером до 600 мА. В результате, исходя из данных, указанных в документации, величина резистора, подключенного между общей шиной и выводом *ILIM* контроллер зарядки равна 620 Ом.