Содержание:

1. Описание станков в целом, описание преимуществ станков на производстве, можно еще про автоматизацию труда написать, этим текстом подводим к мысли о том, что есть необходимость в таких станках( пример аргументации – работа 24/7, точность исполнения, выгода с экономической точки зрения)
2. Цель ВКР – реализация ПО и сборка оборудования под имеющиеся комплектующие, а также их «апгрейд». Данное ПО должно выполнять следующие функции:
   1. Функция #1(пример функции – вывод данных на экран или движение ножа под определенными градусами; какие технологии используются для этого ,возможно стоит добавить фотки)
   2. Функция #2
   3. Функция #3
   4. Функция #4
   5. Функция #5
3. ВОТ ТУТ ВОЗМОЖНО СТОИТ ДОБАВИТЬ ОПИСАНИЕ РАЗЛИЧНЫХ СРЕД ПРОГРАММИРОВАНИЯ(Keil, CubeMX, Atollic TrueStudio)
4. Среда программирования MbedOS
   1. Описание самой системы Mbed(мол с такого-то года разрабатывается такой-то студией, её преимущества и т.д.) стоит поискать на их официальном сайте mbed.org
   2. Встроенные функции библиотек(здесь стоит описать только те функции, которые я использовал из mbed, например pwmout и т.п.)
   3. Описание библиотеки SDFileSystem
   4. Описание библиотеки FATFileSystem.h
5. Отличие системы типа «конструктора» и готовой системы «Типа дофига мощный кодер»(описание разницы систем, которые предоставляют способ настроить свою оригинальную систему и системы предоставляющую более узкий, но зато более специализированный функционал), как пример можно привести из других отраслей, например просто начертить чертеж и создать систему, которая по заданным параметрам будет чертить чертеж(очень абстрактно, но как пример сойдет)
6. Микроконтроллер STM32 Nucleo(F103RB8T), возможно стоит добавить для разницы F103C8T6 и сказать, вот мол сравните цены и функционал, поэтому я выбрал nucleo и бла-бла
   1. Базовые характеристики(нужно взять полное описание характиристик с официального сайта, такие как: частота процессора, количество тактов в секунду, встроенный объем памяти и т.д.)
   2. Встроенные интерфейсы коммуникации(здесь описываются встроенные кнопки, интерфейсы подключения, выходные пины, настроенные сразу на нужные ножки процеессора(SCL,SDA,MISO,MOSI,CK и т.д.)
   3. Цифро-аналоговый преобразователь ??? не знаю, стоит ли это описывать, пока под вопросом
   4. Описание ЦАП ???
   5. Использование DMA ???
   6. Описание интерфейса I2C (взять с интернета описание самой технологии, картиночки найти и т.д.)
      1. Подключение I2C(сделать ссылку на предыдущий пункт, про встроенные пины SDA,SCL, мол вот так всё удобно и т.д.)
      2. Алгоритмы преобразования сигналов I2C(возможно в прошлых пунктах покроется и этот, если нет, то необходимо найти подробный алгоритм реализации, мол вот такие сигналы отсылаются, вот так принимаются, вот так обрабатываются)
   7. Описание интерфейса MISO\MOSI (повторение с предыдущим пунктом)
      1. Подключение
      2. Алгоритмы преобразования
   8. Описание способа хранения данных (продолжение предыдущего пункта, в котором описываем удобство реализованного способа, естественно до этого описываем способ хранения на внутренней памяти, а потом мол вот, флешка это супер-пупер и т.д.)
7. Описание работы датчиков движения(концевой датчик, индуктивный)
   1. Алгоритм работы
   2. Сравнение с обычными датчиками движения(инфрокрасный, ультразвуковый, фоторезисторы и что-нибудь еще, если мало текста)
   3. Назначение в имеющейся системе
8. Обоснование выбора двигателя с обратной связью
   1. Описание работы драйвера двигателя
9. Листинг программы (просмотри код программы, там нужно сделать рефактор пары функций, возможно что-то нужно усложнить для красочности, добавить пару хитро-выдуманных и ненужных алгоритмов, оформи всё красивенько, можно сделать вывод в консоль красивый и продемонстрировать его)
10. Список использованной литературы(скачать из инета типовой список, мол такие классные книжки вот читал, засунуть парочку действительно толковых ссыслок, такие как: репозитории на библиотеки, официальную документацию mbed,keil,cubeMX,stm32 и т.д.)
11. **Станок** — машина (агрегатный механизм), используемая для обработки различных материалов, либо приспособление для выполнения чего-либо.

* [металлорежущие станки](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%BB%D0%BB%D0%BE%D1%80%D0%B5%D0%B6%D1%83%D1%89%D0%B8%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA), которые можно разделить на следующие 9 групп:
  + [токарные станки](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BA%D0%B0%D1%80%D0%BD%D1%8B%D0%B5_%D1%81%D1%82%D0%B0%D0%BD%D0%BA%D0%B8) (токарно-винторезный; токарно-карусельный и пр.)
  + [сверлильные и расточные станки](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%B5%D1%80%D0%BB%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE-%D1%80%D0%B0%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D1%83%D0%BF%D0%BF%D0%B0_%D1%81%D1%82%D0%B0%D0%BD%D0%BA%D0%BE%D0%B2)
  + [фрезерные станки](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B7%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D1%81%D1%82%D0%B0%D0%BD%D0%BA%D0%B8)
  + [шлифовальный станок](https://ru.wikipedia.org/wiki/%D0%A8%D0%BB%D0%B8%D1%84%D0%BE%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA)
  + [зубообрабатывающие](https://ru.wikipedia.org/wiki/%D0%97%D1%83%D0%B1%D1%87%D0%B0%D1%82%D0%BE%D0%B5_%D0%BA%D0%BE%D0%BB%D0%B5%D1%81%D0%BE) и [резьбообрабатывающие](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B7%D1%8C%D0%B1%D0%B0" \o "Резьба) станки
  + разрезные станки (к данному типу можно отнести [гильотинные](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BB%D1%8C%D0%BE%D1%82%D0%B8%D0%BD%D0%B0) ножницы; ленточнопильные станки, предназначенные для отрезки заготовок от целого куска металла)
  + комбинированные станки
  + гидравлические, механические прессы (данный тип станка применяется для уплотнения обрабатываемого материала при помощи направленного усилия)
  + и другие станки; к ним можно отнести мини-станки, [строгальные](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%BE%D0%B3%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA), протяжные станки, заточные станки.
* [деревообрабатывающие станки](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%B0%D1%82%D1%8B%D0%B2%D0%B0%D1%8E%D1%89%D0%B8%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA);
* [камнерезные станки](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BC%D0%BD%D0%B5%D1%80%D0%B5%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA)

Также:

* [балансировочный станок](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA);
* [электроэрозионный станок](https://ru.wikipedia.org/w/index.php?title=%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D1%8D%D1%80%D0%BE%D0%B7%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA&action=edit&redlink=1) (станки для [электроэрозионной обработки](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D1%8D%D1%80%D0%BE%D0%B7%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0): копировально-прошивочные станки, координатно-прошивочные станки, проволочно-вырезные станки);
* [буровой станок](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA);
* [станок-качалка](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%BE%D0%BA-%D0%BA%D0%B0%D1%87%D0%B0%D0%BB%D0%BA%D0%B0);
* [бобинорезка](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%B1%D0%B8%D0%BD%D0%BE%D1%80%D0%B5%D0%B7%D0%BA%D0%B0);
* [печатный станок](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%87%D0%B0%D1%82%D0%BD%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA);
* [ткацкий станок](https://ru.wikipedia.org/wiki/%D0%A2%D0%BA%D0%B0%D1%86%D0%BA%D0%B8%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA)

и прочие.

В данной выпускной квалификационной работе будут рассматриваться раскроечные станки для швейного производства.

Многие современные станки снабжены системой [числового программного управления](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5).

У станка имеется [привод](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%B2%D0%BE%D0%B4) (механический, гидравлический, пневматический), с помощью которого обеспечивается передача движения рабочим органам. Комплекс этих движений называется формообразующими движениями.

Обычно содержит:

* фундаментная плита;
* [станина](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B8%D0%BD%D0%B0);
* привод: [электродвигатель](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%B4%D0%B2%D0%B8%D0%B3%D0%B0%D1%82%D0%B5%D0%BB%D1%8C);
* [коробка передач](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D0%BE%D0%B1%D0%BA%D0%B0_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87);
* рабочий орган: [шпиндель](https://ru.wikipedia.org/wiki/%D0%A8%D0%BF%D0%B8%D0%BD%D0%B4%D0%B5%D0%BB%D1%8C), [суппорт](https://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%BF%D0%BF%D0%BE%D1%80%D1%82);
* коробка подач и механизм подачи;
* [координатный стол](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B0%D1%82%D0%BD%D1%8B%D0%B9_%D1%81%D1%82%D0%BE%D0%BB&action=edit&redlink=1).

Можно выделить следующие преимущества станков на производстве по сравнению с ручным трудом:

1. Скорость работы. По сравнению с ручным трудом, скорость изготовления увеличивается в несколько раз.
2. Качество работы. Исключается человеческий фактор, дрожание рук, изношенные шаблоны.
3. Сложность работы. В отличие от ручного труда, на станке ограничением для формы детали является только инструмент, который установлен.
4. Стоимость работы.

Автоматиза́ция произво́дства — это процесс в развитии машинного производства, при котором функции управления и контроля, ранее выполнявшиеся человеком, передаются приборам и автоматическим устройствам. Введение автоматизации на производстве позволяет значительно повысить производительность труда, обеспечить стабильное качество выпускаемой продукции, сократить долю рабочих, занятых в различных сферах производства.

До внедрения средств автоматизации замещение физического труда происходило посредством механизации основных и вспомогательных операций производственного процесса. Интеллектуальный труд долгое время оставался не механизированным (ручным). В настоящее время операции физического и интеллектуального труда, поддающиеся формализации, становятся объектом механизации и автоматизации. В качестве характеристики измерения может выступать понятие уровня (степени) автоматизации.

В основе организации производственного процесса на каждом предприятии и в любом его цехе лежит рациональное сочетание в пространстве и во времени всех основных, вспомогательных и обслуживающих процессов. Особенности и методы этих сочетаний различны в разных производственных условиях, однако есть и общие принципы:

* специализации
* пропорциональности
* параллельности
* прямоточности
* минимума перерывов
* ритмичности
* замена в первую очередь неквалифицированного монотонного труда
* упрощение сложных производственных процессов заменой на множество простейших

Целью автоматизации производства является повышение производительности труда, улучшение качества продукции, устранение человека от непосредственного участия в производственном процессе и в первую очередь от тяжелых работ и работ в условиях, опасных для здоровья.

В автоматизации производства можно выделить два направления — автоматизацию производственных процессов и инженерного труда.

Автоматизация производства развивалась постепенно. На первой ступени были автоматизированы станки. В станке-полуавтомате автоматизировано управление только рабочим процессом, а загрузка и разгрузка осуществляются человеком. В станке-автомате или автоматической сборочной машине уже автоматизированы дополнительно операции загрузки и разгрузки.

Следующая ступень автоматизации — устранение человека от участия в настройке технологической системы на изготовление первого изделия и ее подналадка во времени.

Объединение станков-автоматов в линию позволяет получить более высокий уровень автоматизации. Применение автоматических линий в крупносерийном и массовом производстве дает существенный экономический эффект.

Необходимым аспектом автоматизации – замена ручного труда на станки, что объяснено вышеизложенными преимуществами.

1. Цель ВКР – реализация ПО и сборка оборудования под имеющиеся комплектующие, а также их «апгрейд». Данное ПО должно выполнять следующие функции:
   1. Управление двигателем \*\*УТОЧНЕНИЕ МОДЕЛИ ДВИГАТЕЛЯ\*\*, через драйвер \*\*уточнение модели драйвера\*\*
   2. Считывание данных с датчиков \*\*уточнение ТИПА датчика(скорее всего концевик)\*\*
   3. Хранение данных о шаблонах на sd-карте
   4. Реализация меню, для переключения между различными режимами работы станка
   5. Динамический вывод информации на LCD монитор, подключенный по шине IIC
   6. Проверка наличия заправленной ленты в станке и обработка события в случае, если уже запущен один из режимов работы, а лента закончилась
   7. \*\*незаконченный фрагмент
2. ВОТ ТУТ ВОЗМОЖНО СТОИТ ДОБАВИТЬ ОПИСАНИЕ РАЗЛИЧНЫХ СРЕД ПРОГРАММИРОВАНИЯ(Keil, CubeMX, Atollic TrueStudio)

Для реализации функций, описанных в ТЗ, необходимо выбрать набор инструментов с которым написание кода было бы легким и удобным, интеграция с новыми модулями не занимала бы много времени, а также прошивка контроллера изначально поддерживалась средой программирования. Поэтому было проведено исследование среди доступных IDE:

Keil- Полная версия:

Среда разработки, представляющая собой набор утилит для выполнения полного комплекса мероприятий по написанию программного обеспечения для микроконтроллеров.

Рабочее окно программы Keil

Keil uVision позволяет работать с проектами любой степени сложности, начиная с введения и правки исходных текстов и заканчивая внутрисхемной отладкой кода и программированием ПЗУ микроконтроллера. От разработчика скрыта большая часть второстепенных функций, что сильно разгружает интерфейс и делает управление интуитивно понятным. Однако при возрастании сложности реализуемых задач, всегда можно задействовать весь потенциал модулей, функционирующих под управлением единой оболочки. Среди основных программных средств Keil uVision можно отметить.

1. Базу данных микроконтроллеров, содержащую подробную информацию обо всех поддерживаемых устройствах. Здесь хранятся их конфигурационные данные и ссылки на источники информации с дополнительными техническими описаниями. При добавлении нового устройства в проект все его уникальные опции устанавливаются автоматически.

2. Менеджер проектов, служащий для объединения отдельных текстов программных модулей и файлов в группы, обрабатываемые по единым правилам. Подобная группировка позволяет намного лучше ориентироваться среди множества файлов.

3. Встроенный редактор, облегчающий работу с исходным текстом за счет использования многооконного интерфейса, выделения синтаксических элементов шрифтом и цветом. Существует опция настройки в соответствии со вкусами разработчика. Редактирование остается доступным и во время отладки программы, что позволяет сразу исправлять ошибки или отмечать проблемные участки кода.

4. Средства автоматической компиляции, ассемблирования и компоновки проекта, которые предназначены для создания исполняемого (загрузочного) модуля программы. При этом между файлами автоматически генерируются новые ассемблерные и компиляторные связи, которые в дальнейшем позволяют обрабатывать только те файлы, в которых произошли изменения или файлы, находящиеся в зависимости от изменённых. Функция глобальной оптимизации проекта позволяет достичь наилучшего использования регистров микроконтроллера путем неоднократной компиляции исходного кода. Компиляторы uVision работают с текстами, написанными на Си или ассемблере для контроллеров семейств ARM, MSC51, C166 и многих других. Кроме того возможно использование компиляторов других производителей.

5. Отладчик-симулятор, отлаживающий работу скомпилированной программы на виртуальной модели микропроцессора. Довольно достоверно моделируется работа ядра контроллера и его периферийного оборудования: портов ввода-вывода, таймеров, контроллеров прерываний. Для облегчения комплексной отладки разрабатываемого программного обеспечения возможно подключение программных моделей нестандартного оборудования.

6. Дополнительные утилиты, облегчающие выполнение наиболее распространенных задач. Число и набор меняется от версии к версии. Выделяют следующие из них:

• Source Browser, содержащую базу данных программных символов для быстрого поиска;

• Find in Files, предназначенную для поиска заданного кода во всех файлах указанной папки или проекта;

• Tools Menu, позволяющую использовать утилиты сторонних производителей;

• PC-Lint, анализирующую исходный текст программы с выделением потенциально опасных мест;

• Flash tool, программирующую FLASH-память микроконтроллеров.

Работа отладчика-симулятора Keil

Среда программирования разработана компанией Keil, которая была основана в Мюнхене в 1982 году братьями Гюнтером и Рейнхардом. В октябре 2005 года Keil вошла в состав американской корпорации ARM. На сегодняшний день она представляет широкий спектр различных средств для разработки программ, включающих Cи-компиляторы, макроассемблеры, отладчики, симуляторы, линкеры, IDE-приложения и оценочные платы для различных семейств микроконтроллеров.

Программа Keil uVision является платной и стоит очень больших денег. По ссылке ниже, после заполнения анкеты можно скачать демонстрационную версию. Разумеется, в ней присутствует ряд ограничений и основное из них – 32 КБ на размер программы.

Среда разработки Keil uVision представлена на английском языке.

Программа работает на персональных компьютерах под управлением только операционной системы Windows версий 2000, XP, Vista и 7.

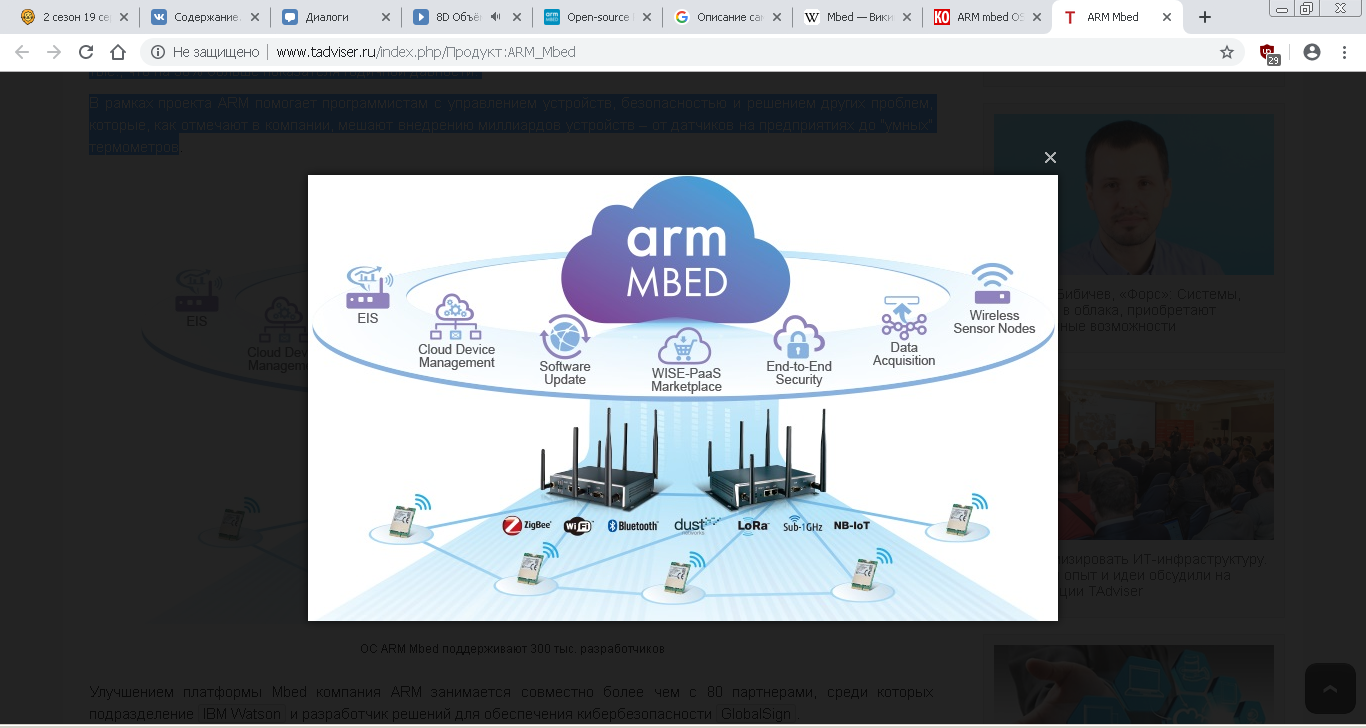
Распространение программы: Shareware (платная). Есть демоверсия с рядом ограничений, в т.ч. на размер кода - не более 32 КБ.

Keil – краткая версия(для таблицы):

**4.А**

ARM Mbed – представленная в 2009 году операционная система для устройств Интернета вещей на базе 32-разрядных микроконтроллеров семейства ARM Cortex-M. Проект был запущен компанией ARM совместно с другими производителями чипов. Платформа работает онлайн и содержит интегрированную среду разработки (IDE) включая текстовый редактор, компилятор, набор библиотек и примеры программного кода. ARM Mbed состоит из двух частей: ядра, над которым работает команда профессиональных программистов, и компонентов, в создании которых может принять участие любой желающий.

В рамках проекта ARM помогает программистам с управлением устройств, безопасностью и решением других проблем, которые, как отмечают в компании, мешают внедрению миллиардов устройств – от датчиков на предприятиях до "умных" термометров.



Важной составляющей проекта Mbed является система автоматического конфигурирования и сборки, как в online режиме так и в offline, т.е. локально на компьютере пользователя. Конфигурировать предлагается путем редактирования файлов в формате .json со специальными именами. Затем находящиеся в проекте Python скрипты преобразуют эти файлы в заголовочные файлы, файлы рабочих пространств выбранной пользователем IDE, командные файлы линкеров и прочие вспомогательные файлы.

Но проблема описанного метода в непрозрачности с точки зрения исходных текстов, так как нам очень трудно отследить где и что в исходниках меняет система конфигурирования. С другой стороны в нашем случае нет никаких мотивов поддерживать способность проекта автоматически переноситься на разные IDE.

Поэтому от такого подхода решено было отказаться. Как писалось в предыдущей статье, был просто сформирован в online проект для IDE IAR, получена неструктурированная куча файлов в рабочем пространстве IDE, затем выполнена их систематизация и отброшено ненужное. В результате больше не нужно делать конфигурирование через .json файлы и осталось только три конкретных места где находятся параметры влияющие на конфигурацию Mbed:

* Опции компилятора в среде IDE
* Командный файл MKE18F512xxx16\_flash.icf линкера
* Заголовочный файл mbed\_config.h

В файле mbed\_config.h можно насчитать около 130 дефайнов, что поначалу очень напрягает. Но к счастью большинство из них относится к стекам беспроводных протоколов, которые к данному моменту в проекте не используются. Для удобства записи были отсортированы чтобы актуальные разместились вверху.

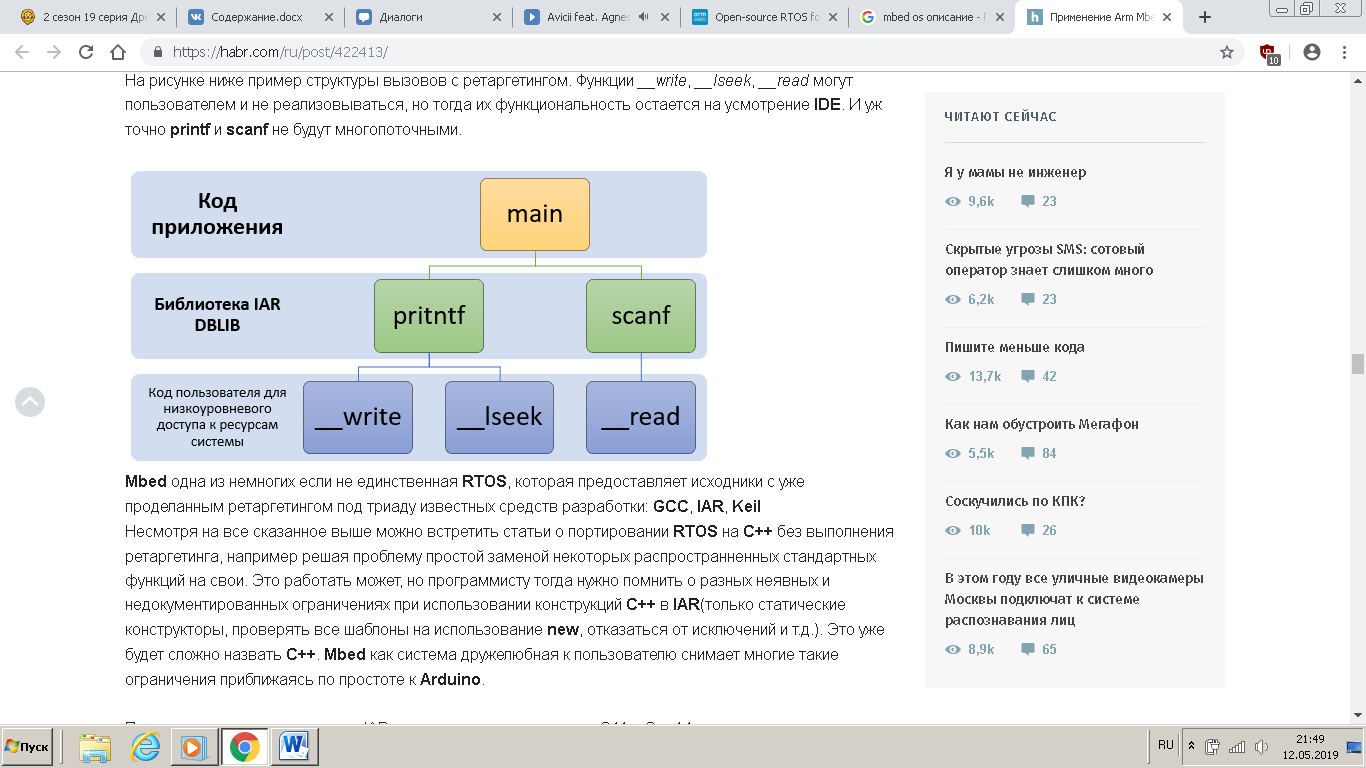
API верхнего уровня в Mbed написано на C++, поэтому этот язык приходится использовать и в прикладном коде. Но тут есть нюансы о которых необходимо знать.

Использование C++ для RTOS в малых встраиваемых системах еще сравнительная редкость. Проблема здесь в том, что успешные проекты RTOS стремятся быть мультиплатформенными, а С++ предъявляет повышенные требования к менеджменту ресурсов платформы по сравнению с С. Причина в стремлении скрыть от пользователя детали низкоуровневого управления ресурсами. Речь прежде всего о ресурсах памяти. Конструкторы, деструкторы, потоки, исключения c автоматической деструкцией, шаблоны объектов структур данных и др. используют неявные операции с динамической памятью. Но ресурс оперативной памяти RAM в малых системах очень ограничен. Оперативная память — самый дефицитный ресурс в таких системах и особенно в RTOS. В RTOS каждой задаче выделяется стек, его точный размер разработчик заранее спрогнозировать не может и поэтому выбирает с запасом. Таким образом наличие RTOS с десятком задач сразу вызывает необходимость в RAM размером от 10 до 30 кБ. Еще много памяти нужно для различных парсеров и протоколов (HTTP, HTML...) и файловых систем. Если применяется дисплей, то еще более увеличиваются требования к свободной RAM.

Библиотеки сред разработки типа IAR оснащаются неплохими менеджерами динамической памяти, но они рассчитаны на однопоточную среду исполнения. Чтобы они начали работать в RTOS нужно написать дополнительный код. Этот процесс называется retargeting.

В RTOS написанных на C ретаргетинг как правило не производится. Поскольку там нет неявных операций с динамической памятью на уровне языка, то все операции выполняются явно вызовом собственных потокобезопасных вариантов функций malloc и free. Программист имеет полный контроль над операциями с динамической памятью и легко может применять все возможные меры по ее экономии.

В случае C++ если мы хотим использовать все возможности этого языка нам придется делать retargeting. Но retargeting в каждой среде разработки процесс сугубо индивидуальный. Это и усложняет жизнь разработчикам RTOS.



На рисунке ниже пример структуры вызовов с ретаргетингом. Функции \_\_write, \_\_lseek, \_\_read могут пользователем и не реализовываться, но тогда их функциональность остается на усмотрение IDE. И уж точно printf и scanf не будут многопоточными.

Mbed одна из немногих если не единственная RTOS, которая предоставляет исходники с уже проделанным ретаргетингом под триаду известных средств разработки: GCC, IAR, Keil

Несмотря на все сказанное выше можно встретить статьи о портировании RTOS на C++ без выполнения ретаргетинга, например решая проблему простой заменой некоторых распространненных стандартных функций на свои. Это работать может, но программисту тогда нужно помнить о разных неявных и недокументированных ограничениях при использовании конструкций C++ в IAR(только статические конструкторы, проверять все шаблоны на использование new, отказаться от исключений и т.д.). Это уже будет сложно назвать C++. Mbed как система дружелюбная к пользователю снимает многие такие ограничения приближаясь по простоте к Arduino.

**6.а**

Сравним характеристики двух микроконтроллеров:

* 1. STM32 Nucleo F401RE
* Ядро: Cortex-M4
* Рабочая частота: 84 МГц
* Разрядность: 32 бита
* Набор машинных инструкций: ARMv7E-M
* Производительность: 105 DMIPS
* Объём SRAM: 96 кБ
* Объём Flash: 512 кБ
* Количество цифровых пинов: до 81
* Количество аналоговых входов: 10
* Разрядность АЦП: 12 бит
* Количество SPI: 4
* Количество I²C: 3
* Количество UART: 3
* Количество таймеров: 10
* Напряжение питания: 5 В, 7–12 В
* Цена: 1990 рублей
  1. STM32F103C8T6
* Ядро: arm cortex-m3
* Рабочая частота: 72 МГц
* Разрядность: 32 бита
* Объём SRAM: 20 кБ
* Объём Flash: 64 кБ
* Разрядность АЦП: 12 бит
* Количество SPI: 2
* Количество I²C: 2
* Количество UART: 3
* Количество таймеров: 4
* Напряжение питания: 2 – 3.6 В
* Цена: 220 рублей

Анализ и сравнение характеристик показывает, что микроконтроллер STM32 Nucleo F401RE выигрывает по большему количеству параметров, а именно по следующим: рабочая частота, объем SRAM, объем FLASH, количество SPI, количество I²C, количеству таймеров и напряжения питания.

**6.б**

1. STM32 Nucleo F401RE

Интерфейсы: SPI, I^2S, UART

1. STM32F103C8T6

Встроенные интерфейсы: can, i2c, irda, lin, spi, uart, usb

Встроенная периферия dma, pwm, pdr, por, pvd, pwm, tempsensor, wdt

**6.ф**

Для увеличения эффективности, упрощения схемотехнических решений, Philips разработала простую двунаправленную двухпроводную шину для так называемого "межмикросхемного" (inter-IC) управления.

Шина получила название - InterIC, или IIC (I2C) шина.

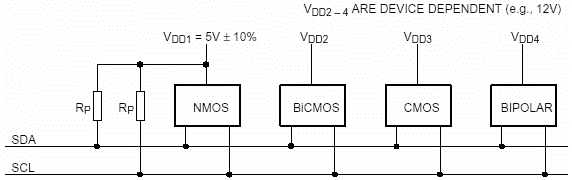
В настоящее время только Philips производит более 150 наименований I2C-совместимых устройств, функционально предназначенных работы в электронном оборудовании различного назначения. В их числе ИС памяти, видеопроцессоров и модулей обработки аудио- и видео-сигналов, АЦП и ЦАП, драйверы ЖК-индикаторов, процессоры со встоенным аппаратным контроллером I2C шины и многое другое.

I2C шина является одной из модификаций последовательных протоколов обмена данных. В стандартном режиме обеспечивается передача последовательных 8-битных данных со скоростью до 100 кбит/с, и до 400 кбит/с в "быстром" режиме. Для осуществления процесса обмена информацией по I2C шине, используется всего два сигнала линия данных SDA линия синхронизации SCL Для обеспечения реализации двунаправленности шины без применения сложных арбитров шины выходные каскады устройств, подключенных к шине, имеют открытый сток или открытый коллектор для обеспечения функции монтажного "И". ( подключения)

Простая двухпроводная последовательная шина I2C минимизирует количество соединения между ИС, ИС имеют меньше контактов и требуется меньше дорожек. Как результат - печатные платы становятся более простыми и технологичными при изготовлении. Интегрированный I2C-протокол устраняет необходимость в дешифраторах адреса и другой внешней логике согласования.

Максимальное допустимое количество микросхем, подсоединённых к одной шине, ограничивается максимальной емкостью шины 400 пФ.

Встроенный в микросхемы аппаратный алгоритм помехоподавления обеспечивает целостность данных в условиях помех значительной величины. Все I2C-совместимые устройства имеют интерфейс, который позволяет им связываться друг с другом по шине даже в том случае, если их напряжение питания существенно отличается. На следующем рисунке представлен принцип подключения нескольких ИМС с различными напряжениями питания к одной шине обмена.



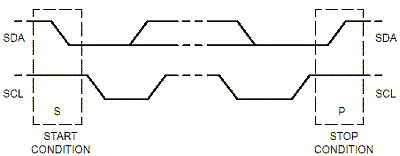
Каждое устройство распознается по уникальному адресу и может работать как передатчик или приёмник, в зависимости от назначения устройства.

Кроме того, устройства могут быть классифицированы как ведущие и ведомые при передаче данных. Ведущий - это устройство, которое инициирует передачу данных и вырабатывает сигналы синхронизации. При этом любое адресуемое устройство считается ведомым по отношению к ведущему.

Исходя из спецификации работы шины, в каждый отдельный момент в шине может быть только один ведущий, а именно то устройство, которое обеспечивает формирование сигнала SCL шины. Ведущий может выступать как в роли ведущего-передатчика, так и ведущего-приемника. Тем не менее - шина позволяет иметь несколько ведущих, накладывая определенные особенности их поведения в формировании сигналов управления и контроля состояния шины. Возможность подключения более одного ведущего к шине означает, что более чем один ведущий может попытаться начать пересылку в один и тот же момент времени. Для устранения "столкновений", который может возникнуть в данном случае, разработана процедура арбитража - поведения ведущего при обнаружении "захвата" шины другим ведущим.

Процедура синхронизации двух устройств. Эта процедура основана на том, что все I2C-устройства подключаются к шине по правилу монтажного И. В исходном состоянии оба сигнала SDA и SCL находятся в высоком состоянии.

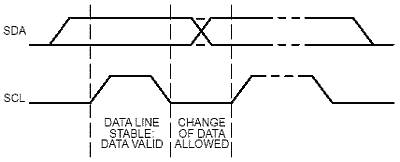
Процедура обмена начинается с того, что ведущий формирует состояние СТАРТ - ведущий генерирует переход сигнала линии SDA из ВЫСОКОГО состояния в НИЗКОЕ при ВЫСОКОМ уровне на линии SCL. Этот переход воспринимается всеми устройствами, подключенными к шине как признак начала процедуры обмена.



Генерация синхросигнала - это всегда обязанность ведущего; каждый ведущий генерирует свой собственный сигнал синхронизации при пересылке данных по шине.

Процедура обмена завершается тем, что ведущий формирует состояние СТОП - переход состояния линии SDA из низкого состояния в ВЫСОКОЕ при ВЫСОКОМ состоянии линии SCL.

Состояния СТАРТ и СТОП всегда вырабатываются ведущим. Считается, что шина занята после фиксации состояния СТАРТ. Шина считается освободившейся через некоторое время после фиксации состояния СТОП.



При передаче посылок по шине I2C каждый ведущий генерирует свой синхросигнал на линии SCL.

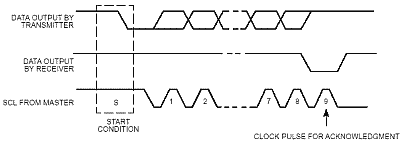
После формирования состояния СТАРТ, ведущий опускает состояние линии SCL в НИЗКОЕ состояние и выставляет на линию SDA старший бит первого байта сообщения. Количество байт в сообщении не ограничено.

Спецификация шины I2C разрешает изменения на линии SDA только при НИЗКОМ уровне сигнала на линии SCL.

Данные действительны и должны оставаться стабильными только во время ВЫСОКОГО состояния синхроимпульса.

Для подтверждения приема байта от ведущего - передатчика ведомым - приемником в спецификации протокола обмена по шине I2C вводится специальный бит подтверждения, выставляемый на шину SDA после приема 8 бита данных.

Таким образом передача 8 бит данных от передатчика к приемнику завершаются дополнительным циклом (формированием 9-го тактового импульса линии SCL), при котором приемник выставляет низкий уровень сигнала на линии SDA, как признак успешного приема байта.



Подтверждение при передаче данных обязательно. Соответствующий импульс синхронизации генерируется ведущим. Передатчик отпускает (ВЫСОКОЕ) линию SDA на время синхроимпульса подтверждения. Приёмник должен удерживать линию SDA в течение ВЫСОКОГО состояния синхроимпульса подтверждения в стабильном НИЗКОМ состоянии.

В том случае, когда ведомый-приёмник не может подтвердить свой адрес (например, когда он выполняет в данный момент какие-либо функции реального времени), линия данных должна быть оставлена в ВЫСОКОМ состоянии. После этого ведущий может выдать сигнал СТОП для прерывания пересылки данных.

Если в пересылке участвует ведущий-приёмник, то он должен сообщить об окончании передачи ведомому-передатчику путем не подтверждения последнего байта. Ведомый-передатчик должен освободить линию данных для того, чтобы позволить ведущему выдать сигнал СТОП или повторить сигнал СТАРТ.

Синхронизация выполняется с использованием подключения к линии SCL по правилу монтажного И.

Это означает, что ведущий не имеет монопольного права на управление переходом линии SCL из НИЗКОГО состояния ВЫСОКОГО. В том случае, когда ведомому необходимо дополнительное время на обработку принятого бита, он имеет возможность удерживать линию SCL в низком состоянии до момента готовности к приему следующиго бита. Таким образом, линия SCL будет находиться в НИЗКОМ состоянии на протяжении самого длинного НИЗКОГО периода синхросигналов.

Устройства с более коротким НИЗКИМ периодом будут входить в состояние ожидания на время, пока не кончится длинный период. Когда у всех задействованных устройств кончится НИЗКИЙ период синхросигнала, линия SCL перейдет в ВЫСОКОЕ состояние. Все устройства начнут проходить ВЫСОКИЙ период своих синхросигналов. Первое устройство, у которого кончится этот период, снова установит линию SCL в НИЗКОЕ состояние. Таким образом, НИЗКИЙ период синхролинии SCL определяется наидлиннейшим периодом синхронизации из всех задействованных устройств, а ВЫСОКИЙ период определяется самым коротким периодом синхронизации устройств.

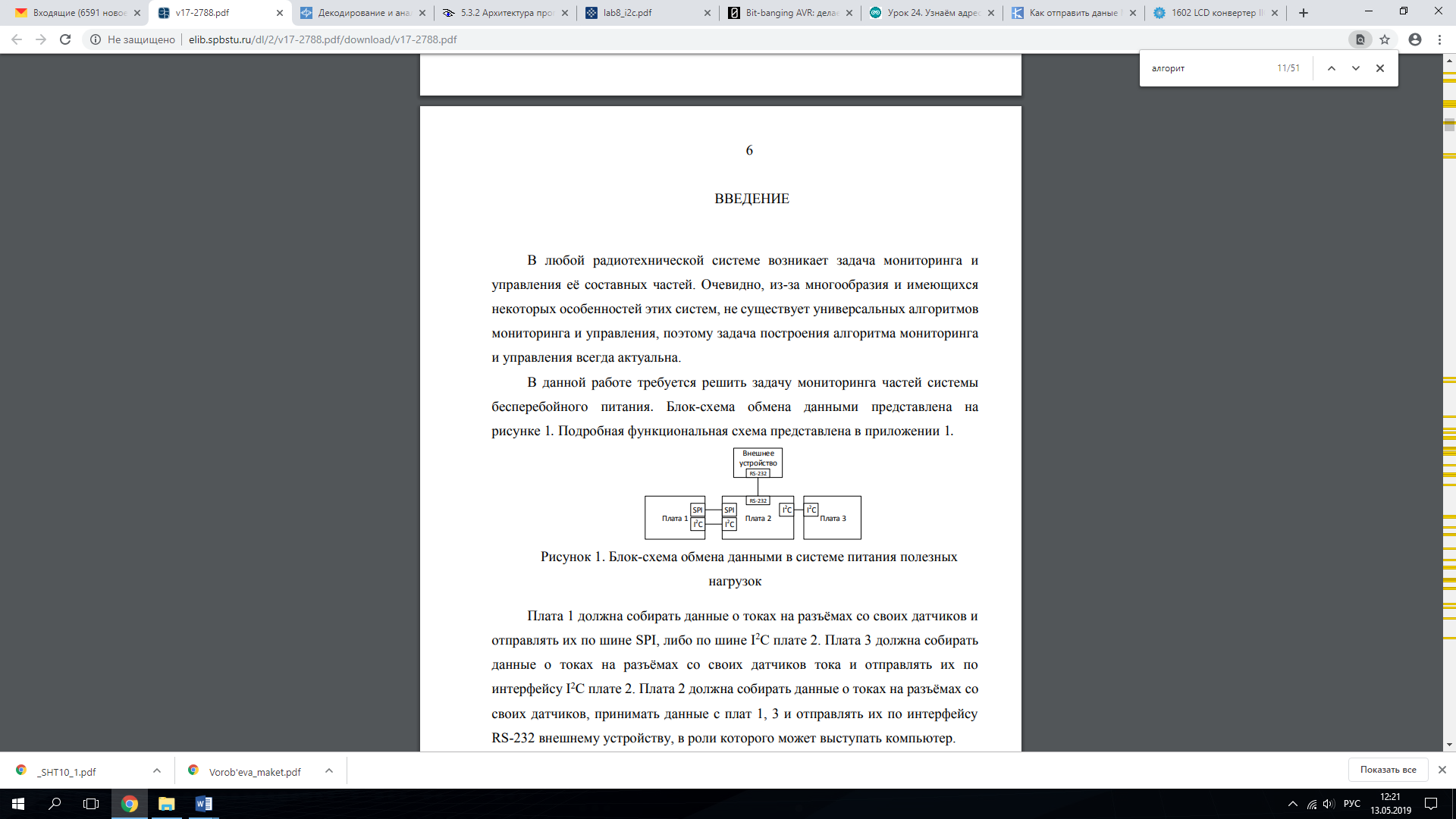
Механизм синхронизации может быть использован приемниками как средство управления пересылкой данных на байтовом и битовом уровнях.

На уровне байта, если устройство может принимать байты данных с большой скоростью, но требует определенное время для сохранения принятого байта или подготовки к приему следующего, то оно может удерживать линию SCL в НИЗКОМ состоянии после приема и подтверждения байта, переводя таким образом передатчик в состояние ожидания.

На уровне битов, устройство такое как микроконтроллер без встроенных аппаратных цепей I2C или с ограниченными цепями может замедлить частоту синхроимпульсов путем продления их НИЗКОГО периода. Таким образом скорость передачи любого ведущего адаптируется к скорости медленного устройства.

II.

Блок-схема обмена данными представлена на рисунке.



Плата 1 должна собирать данные о токах на разъёмах со своих датчиков и отправлять их по шине SPI, либо по шине I2C плате 2. Плата 3 должна собирать данные о токах на разъёмах со своих датчиков тока и отправлять их по интерфейсу I2C плате 2. Плата 2 должна собирать данные о токах на разъёмах со своих датчиков, принимать данные с плат 1, 3 и отправлять их по интерфейсу RS-232 внешнему устройству, в роли которого может выступать компьютер.

Плата 1 и плата 2 могут быть связаны с помощью I 2C или SPI. Для определения наиболее предпочтительного способа следует рассмотреть эти интерфейсы.

Необходимость же разработки нового блока питания явилось следствием предъявляемых к нему требований:

Иметь возможность работать от

* электрической сети 220В переменного тока
* аккумуляторной батареи 12В или иного источника питания с выходным
* напряжением 12В
* возможность питания от 13-24В
* Устройство должно иметь
* один выход 24 В ± 1% с максимальной нагрузкой 3А
* два выхода по 48 В ± 1% и макс нагрузкой 0,5А на каждом
* два выхода по 12 В ± 1% и макс нагрузкой 1А на каждом
* иметь внутренний источник питания на случай пропажи внешнего питания

Температурный режим: -40°С - +50°С

Имеющиеся в продаже источники питания не подходят по температурному режиму, либо нет требуемых выдаваемых напряжений, либо они не удовлетворяют требованиям по входному напряжению. Кроме того, они представляют собой громоздкие устройства. В связи с этим возникла необходимость в разработке источника бесперебойного питания с характеристиками, указанными выше.

Уже разработаны методики мониторинга радиотехнических систем. Но в нашей задаче явно получается многопроцессорная система, в которой потребуется исследование влияния работы микроконтроллеров друг на друга при их совместной работе в смысле качества выполнения требуемых функций.

**6.г**

SPI (Serial Peripheral Interface, SPI bus — последовательный периферийный интерфейс, шина SPI) — 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 (англ. four-wire) интерфейсом.

В отличие от стандартного последовательного порта (англ. standard serial port), SPI является синхронным интерфейсом, в котором любая передача синхронизирована с общим тактовым сигналом, генерируемым ведущим устройством (процессором). Принимающая (ведомая) периферия синхронизирует получение битовой последовательности с тактовым сигналом. К одному последовательному периферийному интерфейсу ведущего устройства-микросхемы может присоединяться несколько микросхем. Ведущее устройство выбирает ведомое для передачи, активируя сигнал «выбор кристалла» (англ. chip select) на ведомой микросхеме. Периферия, не выбранная процессором, не принимает участия в передаче по SPI.

В SPI используются четыре цифровых сигнала:

* MOSI — выход ведущего, вход ведомого (*Master Out Slave In*). Служит для передачи данных от ведущего устройства ведомому.
* MISO — вход ведущего, выход ведомого (*Master In Slave Out*). Служит для передачи данных от ведомого устройства ведущему.
* SCLK или SCK — последовательный тактовый сигнал (*Serial Clock*). Служит для передачи тактового сигнала для ведомых устройств.
* CS или SS — выбор микросхемы, выбор ведомого (*Chip Select, Slave Select*).

Конкретные имена портов интерфейса SPI могут различаться в зависимости от производителя аппаратных средств, при этом возможны следующие варианты:

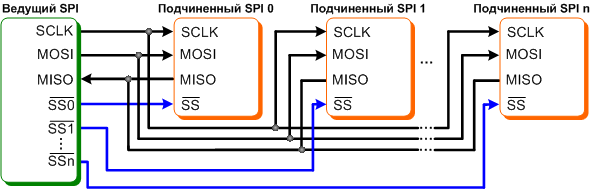
Передача осуществляется пакетами. Длина пакета, как правило, составляет 1 байт (8 бит), при этом известны реализации SPI с иной длиной пакета, например, 4 бита. Ведущее устройство инициирует цикл связи установкой низкого уровня на выводе выбора подчиненного устройства (SS) того устройства, с которым необходимо установить соединение. При низком уровне сигнала SS:

* схемотехника ведомого устройства находится в активном состоянии;
* вывод MISO переводится в режим «выход»;
* тактовый сигнал SCLK от ведущего устройства воспринимается ведомым и вызывает считывание на входе MOSI значений передаваемых от ведущего битов и сдвиг регистра ведомого устройства.

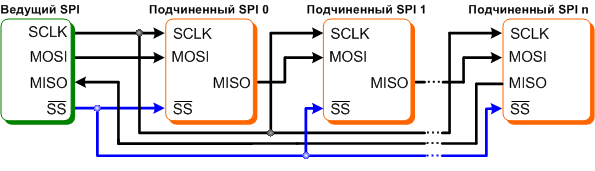
Подлежащие передаче данные ведущее и ведомое устройства помещают в сдвиговые регистры. После этого ведущее устройство начинает генерировать импульсы синхронизации на линии SCLK, что приводит к взаимному обмену данными. Передача данных осуществляется бит за битом от ведущего по линии MOSI и от ведомого по линии MISO. Передача осуществляется, как правило, начиная со старших битов, но некоторые производители допускают изменение порядка передачи битов программными методами. После передачи каждого пакета данных ведущее устройство, в целях синхронизации ведомого устройства, может перевести линию SS в высокое состояние.

* 1. Подключение

Существует три типа подключения к шине SPI, в каждом из которых участвуют четыре сигнала. Самое простое подключение, в котором участвуют только две микросхемы. Здесь, ведущий шины передает данные по линии MOSI синхронно со сгенерированным им же сигналом SCLK, а подчиненный захватывает переданные биты данных по определенным фронтам принятого сигнала синхронизации. Одновременно с этим подчиненный отправляет свою посылку данных. Представленную схему можно упростить исключением линии MISO, если используемая подчиненная ИС не предусматривает ответную передачу данных или в ней нет потребности. Одностороннюю передачу данных можно встретить у таких микросхем как ЦАП, цифровые потенциометры, программируемые усилители и драйверы. Таким образом, рассматриваемый вариант подключения подчиненной ИС требует 3 или 4 линии связи. Чтобы подчиненная ИС принимала и передавала данные, помимо наличия сигнала синхронизации, необходимо также, чтобы линия SS была переведена в низкое состояние. В противном случае, подчиненная ИС будет неактивна. Когда используется только одна внешняя ИС, может возникнуть соблазн исключения и линии SS за счет жесткой установки низкого уровня на входе выбора подчиненной микросхемы. Такое решение крайне нежелательно и может привести к сбоям или вообще невозможности передачи данных, т.к. вход выбора микросхемы служит для перевода ИС в её исходное состояние и иногда инициирует вывод первого бита данных.



Независимое подключение к шине SPI



Каскадное подключение к шине SPI

* 1. Алгоритм преобразования

Если не использовать библиотечные функции по работе с SPI и работать с регистрами напрямую, то алгоритм работы примерно такой:

1. Cначала иницилизируют интерфейс SPI через регистр управления (для ATmega он SPCR)

* отключаем SPI от выходов микроконтроллера (обнуляют бит) или выключаем модуль SPI
* обнуляем регистр управления SPI
* конфигурируем интерфейс SPI, т.е:
  + если позволяет микроконтроллер то подключаем вывод синхронизации и вывод данных (в ATmega такой возможности нет, вывод синхронизации и данных используется всегда)
  + если позволяет микроконтроллер, то выбираем какой регистр сдвига мы будем использовать, 8-битный, 16-и битный. т.е. какими данными мы будем обмениваться по SPI: 8, 16 и т.п. битными.  
    (В ATmega такой возможности нет, по умолчанию используются 8 бит).
  + разрешаем или запрещаем прерывания по spi
  + определяем режим работы SPI
  + определяем как передаются биты, сначала старший разряд или младший
  + определяем полярность сигнала синхронизации.
  + определяем по какому фронту определяется сигнал синхронизации (по спадающему или нарастающему).
  + определяем скорость передачи данных (определить делитель)
  + устанавливаем на вывод SS единицу и иницилизируем как выход (единица означает «ведомому сигнал не принимать», ноль «ведомому сигнал принимать»)
  + иницилизируем на выход выводы MOSI и SCK, а выход MISO на вход (если это не делает автоматически микроконтроллер, для ATmega328p это приходиться делать «вручную»
  + включаем режим ведущего
  + подключит SPI к выходам микроконтроллера или включить модуль SPI

2. Работа интерфейса

* устанавливаем на SS логической ноль (зависит от микроконтроллера, в одних — это надо делать самим программно (ATmega), т.е. мы устанавливаем 0-ль перед передачей и 1-цу после, а в других, выход SS устанавливается в 0-ль автоматически после записи данных в регистр передатчика и 1-цу по окончанию передачи)
* записываем данные в регистр данных передатчика, тем самым запускаем их передачу по MOSI
* ожидаем окончание передачи через цикл while пока в регистре состояния SPI не будет выставлен бит прерывания (для ATmege это бит «SPIF» регистра SPCR)
* устанавливаем на SS логическую единицу

3. Отключить SPI от выхода микроконтроллера или отключить модуль SPI