Санкт-Петербургский государственный университет

Чубуков Филипп Александрович

Выпускная квалификационная работа

Поддержка драйвера USB host для STM32 в ОС Embox

Уровень образования: бакалавриат

Направление *09.03.04 «Программная инженерия»*

Основная образовательная программа *СВ.5080.2017 «Программная инженерия»*

Научный руководитель:
к.ф.-м.н., доц. Д.В. Луцив

Консультант:
Разработчик ООО «Embox» А.В. Бондарев

Рецензент:
Разработчик ООО «Embox» А. И. Калмук

Санкт-Петербург
2021

Saint Petersburg State University

Philipp Chubukov

Bachelor's Thesis

Support for the USB host driver for STM32 in Embox OS

Education level: bachelor

Speciality *09.03.04 «Software Engineering»*

Programme *CB.5080.2017 «Software Engineering»*

Scientific supervisor:
C.Sc., docent D.V. Luciv

Consultant:
software engineer at "Embox" Co. Ltd., A.V. Bondarev

Reviewer:
software engineer at "Embox" Co. Ltd., A.I. Kalmuk

Saint Petersburg
2021

Оглавление

Введение

В настоящее время широкое распространение получили микроконтроллеры. Более того, без них невозможна автоматизация заводов и фабрик, а также реализация многих современных технологий, например, умного дома [4].

USB (Universal Serial Bus) — последовательный интерфейс для подключения периферийных устройств к вычислительной технике. В настоящее время он получил широкое распространение и стал стандартом де факто [11]. Многие микроконтроллеры поддерживают работу с USB, в том числе, в роли USB-хоста, когда имеется возможность подключать USB-устройство к плате, которая может работать с устройством, получая таким образом дополнительную функциональность.

STM32 является семейством 32-битных микроконтроллеров на основе ARM-процессоров серии Cortex M [8]. Они специально разработаны для небольших встраиваемых систем, имея сбалансированные рабочие характеристики, стоимость и энергопотребление, доминируют на рынке 32-битных микроконтроллеров. Представители этого семейства могут обладать весьма большой, по сравнению с другими микроконтроллерами, производительностью и различной периферией. В том числе, многие STM32 могут работать в режиме USB-хоста [3].

Микроконтроллер является набором физических устройств, объединенных на небольшой плате. Для его работы требуется специальный программный код, который будет им управлять, а также достаточно эффективно распределять и использовать весьма скромные, по сравнению с персональными компьютерами, ресурсы для решения различных задач. Для работы с периферийными устройствами используются драйвера, которые выполняют роль связующего звена между физическим устройством и операционной системой. В том числе, для возможности работы микроконтроллера в режиме USB-хоста необходим специальный драйвер, который будет управлять хост-контроллером и следить за выполнением USB-передач.

Embox является кроссплатформенной операционной системой ре-

ального времени для встраиваемых систем [1]. Она лишь частично поддерживает семейство микроконтроллеров STM32, в частности, не поддерживает работу устройства, на котором она размещена, в режиме USB-хоста. В виду актуальности этой задачи и популярности микроконтроллеров STM32 появилась задача реализации драйвера USB хост-контроллера для семейства плат STM32 в операционной системе Embox.

1. Постановка задачи

Целью работы является создание драйвера для поддержки USB хост-контроллера семейства микроконтроллеров STM32 в операционной системе Embox. Для достижения этой цели были поставлены следующие задачи:

- Сделать обзор предметной области;
- Спроектировать и реализовать драйвер хост-контроллера STM32 в операционной системе Embox;
- Протестировать полученную реализацию.

2. Обзор предметной области

2.1. USB-интерфейс

USB (Universal Serial Bus) — универсальная последовательная шина, которая предназначена для подключения периферийных устройств к вычислительной технике [12].

2.1.1. Устройства на USB-шине

На шине представлено несколько типов устройств: Хост, Хаб и Конечное устройство, иерархия которых представлена на рисунке 1 (Конечные устройства располагаются в листьях хабов).

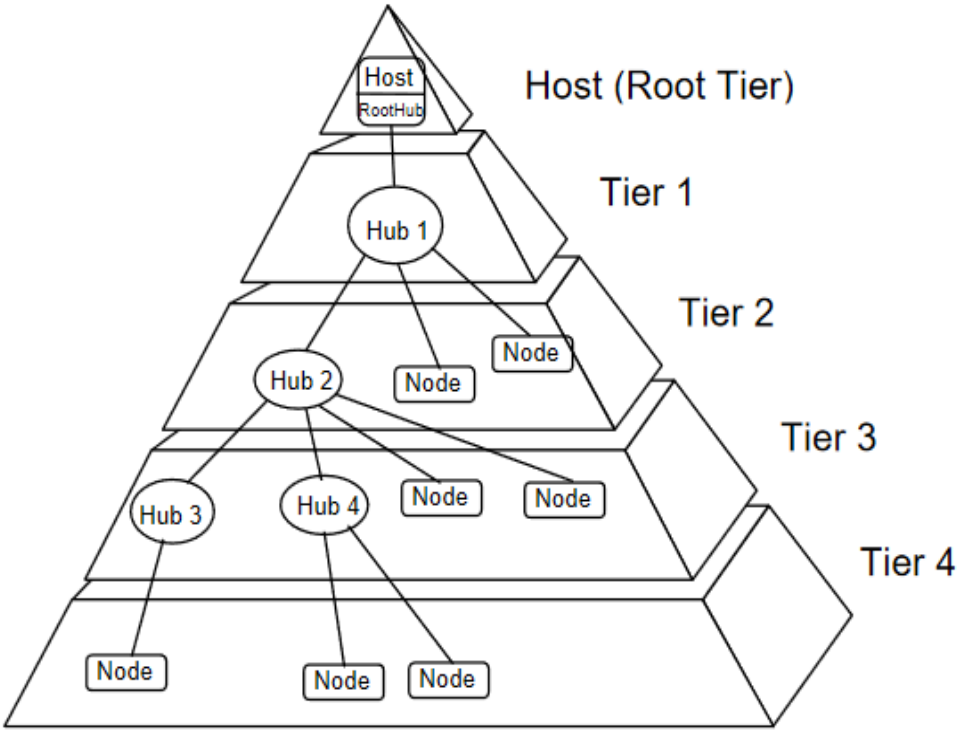

Рис. 1: Иерархия устройств на шине USB [12].

Хост-контроллер — главное управляющее устройство, которое обеспечивает связь и управление всеми конечными устройствами. Взаимодействие с ОС осуществляется с помощью драйвера, который управляет хост-контроллером и выстраивает логику выполнения USB-передач.

Хаб — физическое устройство для подключения конечных устройств,

предоставляет порты и транслирует передачи от хост-контроллера и обратно. Хост-контроллер имеет свой корневой хаб.

Конечные устройства — USB-устройства, которые мы подключаем к USB шине [12].

2.1.2. Типы USB-передач

Как хаб, так и конечное устройство представляют из себя набор конечных точек, к которым открываются каналы, по которым организовываются передачи данных. Существует несколько типов USB-передач для различных задач.

- Управляющие передачи (Control transfers) — предназначены для управления USB-устройствами и получения от них контрольной информации.
- Передачи массивов данных (bulk transfers) — предназначены для передачи больших объемов данных, довольно медленные, но будут гарантированно доставлены. Например, при работе с флэшкой.
- Прерывания (Interrupts) — спонтанные небольшие передачи, которые требуют быстрой обработки, примером могут служить компьютерные мыши и клавиатуры.
- Изохронные передачи (Isochronous transfers) — предназначены для передачи потоковых данных в реальном времени, имеют высокий приоритет и занимают много трафика на шине, но не гарантируют доставку, подходят для передачи видео или аудиопотока [12].

2.1.3. Драйвер Хост-контроллера

На рисунке 2 изображена модель USB-стека. Сверху находятся клиентские приложения, которые работают с USB-устройствами. Ниже находится уровень библиотеки USB, на нем происходит инициализация хост-контроллера и USB-устройств в системе и формирование USB-передач. Затем идет драйвер хост-контроллера, который уникален для

каждого различного хоста, на его плечи ложится физическое управление хост-контроллером, отслеживание его статуса, обработка прерываний, выстраивание логики энумерации USB-устройств и выполнения USB-транзакций. И последние два уровня это физические устройства - хост-контроллер и конечные USB-устройства, подключенные к шине.

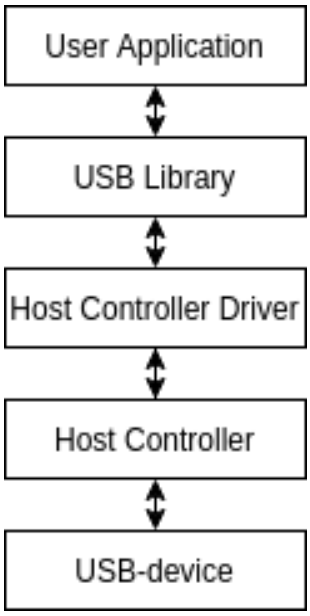

Рис. 2: Модель USB-стека.

2.2. OCPB Embox

Embox (Essential toolbox for embedded development) — это свободная кроссплатформенная операционная система реального времени, разрабатываемая на языке Си для встраиваемых систем. Весь архитектурно-зависимый код представлен в виде отдельных модулей, которые организовываются и добавляются с помощью системы Mybuild. Благодаря этому мы можем добавлять в сборку только необходимую для нашей задачи функциональность, не тратя ресурсы на лишние модули, что будет очень полезно в случае микроконтроллеров, когда производительность сильно ограничена.

Embox портирован на различные архитектуры, такие как x86, ARM, SPARC, MIPS и другие, а также на различные устройства, в том числе

на семейство микроконтроллеров STM32 [1].

В Embox реализован уровень библиотеки USB, который общается с клиентским ПО, реализует логику работы с USB-устройствами, в том числе их энумерацию, инициализацию в системе Embox и генерацию USB-передач.

2.3. STM32

STM32 — Семейство микроконтроллеров на основе 32-битных ARM процессоров Cortex-M7F, Cortex-M4F, Cortex-M3, Cortex-M0, которые производятся компанией STMicroelectronics. Помимо ядра каждый микроконтроллер состоит из статической RAM, flash памяти,отладочного интерфейса и различной периферии.

Семейство ARM Cortex с профилем M, которое используется в микроконтроллерах STM32 это процессоры с сокращенным набором инструкций (RISC), специально разработанные для небольших систем, которым необходимы ресурсы с учетом как можно меньшего электропотребления. Также это семейство ядер выполнено в гарвардской архитектуре, что позволяет выполнять операции параллельно [3].

Каждый процессор из этой линейки соответствует своему подсемейству микроконтроллеров STM32, в каждом из которых представлено много различных плат с различной периферией, обычно для каждой задачи подбирается определенный микроконтроллер.

2.3.1. Хост-контроллер STM32

Подавляющее количество представителей семейства микроконтроллеров STM32 могут работать в режиме USB-устройства, но многие представители подсемейств STM32F2, STM32F4, STM32F7 поддерживают двойную роль, как устройства, так и USB хоста.

USB OTG FS — хост-контроллер семейства микроконтроллеров STM32, выполняет двойную роль в архитектуре USB, то есть поддерживает работу и в режиме USB-устройства, и в режиме USB-хоста. Полностью соответствует стандартам On-The-Go Supplement to the USB 2.0

и USB 2.0. В режиме хоста поддерживает full speed (12 мегабит/сек) и low speed (1.5 мегабит/сек) [9]

2.3.2. Управление хост-контроллером STM32

Управление хост-контроллером выполняется путем чтения и записи CRS (Control and Status) регистров, которые располагаются по определенным адресам памяти микроконтроллера. Читать и записывать в эти регистры, в зависимости от их назначения, имеет возможность как сам хост-контроллер, так и драйвер хост-контроллера [9].

Большинство основных событий на шине USB приводят к генерации ядром STM32 прерываний, которые необходимо обрабатывать на уровне драйвера хост-контроллера. Например, такие как:

- Подключение устройства — ядро OTG FS создает прерывание порта хоста, вызванное тем, что хост-контроллер записывает бит подключения устройства в регистр состояния порта. Затем необходимо начать процесс энумерации подключенного устройства.
- Энумерация — после того, как хостом детектируется подключение устройства, запускается процесс энумерации, начинающийся отправкой команды USB-reset, после выполнения которой устройство готово к энумерации с помощью дальнейших конфигурационных команд.
- Отключение устройства — это событие приводит к вызывания. прерывания отключения, вызванному записью хост-контроллером бита отключения устройства в регистр состояния порта [9].

2.3.3. Выполнение USB-транзакций хост-контроллером

Ядро OTG FS располагает восемью каналами хоста, каждый из которых может быть сконфигурирован для передачи определенного типа передач в определенную сторону с определенной скоростью, максимальным размером пакета и конечным устройством. В задачу драйвера входит выполнить эту конфигурацию и открыть необходимые каналы.

Затем необходимо подавать USB-транзакции из очереди в эти каналы, в ядро хост-контроллера встроен планировщик, который может самостоятельно изменять порядок выполнения запросов USB-транзакций, которые ему выдает драйвер. В соответствии со спецификацией USB в начале кадра выполняются периодические транзакции (isochronous и interrupt) и затем не периодические (control и bulk) [12].

Отслеживание статуса USB-транзакций выполняется путем обработки прерываний, которые генерируются при обработке передач хост-контроллером [9].

3. Особенности реализации

3.1. Развертывание окружения разработки

Для разработки, внедрения и тестирования была выбрана плата F429ZI - Nucleo, в виду ее актуальной поддержки в ОС Embox, доступности и возможности работы в режиме USB-хоста [6].

1. OpenOCD (Open On-Chip Debugger) — предоставляет инструментарий отладки, внутрисхемного программирования, внутрисхемного тестирования для встраиваемых систем. Предоставляет доступ к адаптеру для отладки, который встроен в наш микроконтроллер. С его помощью мы отслеживаем состояние подключенного к компьютеру микроконтроллера в реальном времени [7].

2. С помощью отладчика gdb-multiarch в связке с openocd мы подключаемся к микроконтроллеру, можем управлять им, загружать ОС Embox и отлаживать код [2].

3. Minicom — это консольная программа операционной системы linux для установки сеанса связи с устройствами через последовательный com-порт. С ее помощью мы подключаемся к плате, на которой запущен ОС Embox и можем управлять ей со стороны компьютера [5].

Таким образом мы получаем систему разработки, в которой мы можем легко загружать и отлаживать наш код.

3.2. HAL-библиотека

HAL (Hardware Abstraction Layer) — библиотека из серии stm32cube, предоставляемых производителем STMelectronics, основной задачей которой является сокращение времени разработки и упрощение портирования кода на различные семейства STM32. Представляет собой аппаратную абстракцию, обертку над низкоуровневыми операциями над

регистрами, позволяя нам абстрагироваться от этого этапа и заниматься построением логики нашего драйвера [10].

На рисунке 3 представлена работа USB-стека в связке с библиотекой HAL, теперь драйвер отдает команды и отслеживает состояние хост-контроллера через ее интерфейс.

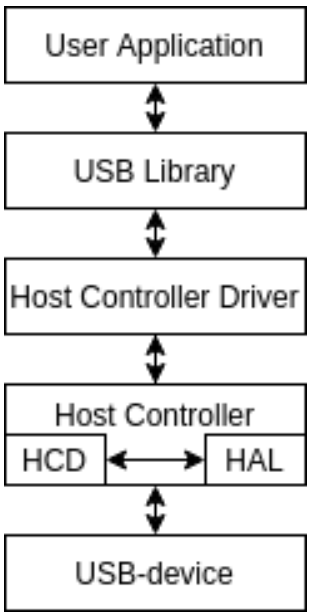

Рис. 3: Модель USB-стека с HAL-библиотекой.

3.3. Интерфейс драйвера хост-контроллера

Для реализации драйвера хост-контроллера нам потребуется инициализировать и реализовать несколько функций, которые будут вызываться на уровне библиотеки USB или операционной системы:

- stm32hc_init — вызывается при старте ОС Embox и начинает процесс инициализации хост-контроллера в системе;
- stm32hc_start — вызывается на уровне библиотеки USB, инициализирует необходимую периферию, включает ядро USB в режиме хоста, начинает процесс инициализации корневого хаба в системе Embox;
- stm32_roothub_control — выполняет запросы к корневому хабу

хост-контроллера, возвращает его описание и управляет его портами;

- stm32_request — получает USB-транзакции к конечным устройствам от уровня библиотеки USB и организовывает их выполнение;
- stm32_irq_handler — работает с прерываниями, которые генерирует ядро USB.

3.4. Реализация драйвера хост-контроллера

Драйвер представляет из себя отдельный модуль системы, реализованный на языке программирования СИ с соответствующим Mybuild файлом, который содержит описание модуля драйвера и зависимостей, необходимых для его корректной сборки и работы.

3.4.1. Инициализация Хост-контроллера

Во время запуска ОС Embox подключает все модули, включенные в сборку и вызывает функции инициализации, если это необходимо. В нашем случае вызывается функция stm32_hc_init, она создает новый хост-контроллер драйвер в системе, подключает к прерываниям функцию stm32_irq_handler и запускает процесс инициализации драйвера в системе Embox, который выполняется на уровне библиотеки USB.

Во время этого процесса в первую очередь вызывается функция stm32_hc_start, которая отвечает за включение USB-ядра в режиме хоста. В первую очередь нам необходимо инициализировать структуру управления хост-контроллером HCD_HandleTypeDef, предоставляемую библиотекой HAL, и вызвать функции этой библиотеки, которые произведут действия с регистрами для запуска USB-ядра в режиме хоста. Но также, помимо действий с регистрами, одинаковых для различных микроконтроллеров, необходимо правильно сконфигурировать периферию, то есть настроить контакты ввода и вывода нужным образом, а это уже задача, специфичная для разных плат. Для этого бы-

ла реализована функция HAL_HCD_MspInit которая вызывается со стороны библиотеки HAL и инициализирует контроллер необходимым способом.

После успешного запуска USB-ядра в режиме хоста вызывается функция usb_new_device, которая регистрирует корневой хаб хост-контроллера в системе Embox. Регистрация осуществляется путем отправки USB-запросов корневому хабу для получения информации о нем. Реализация этой логики представлена в функции stm32_root_hub_control, также эта функция предназначена для работы с портами корневого хаба, например получения информации о их состоянии или их сброса и очистки.

При успешном прохождении запуска хост-контроллера и регистрации его корневого хаба в системе Embox, STM32 полностью готов для начала работы с USB-устройствами.

В случае необходимости отключения хост-контроллера (например для его перезагрузки) реализована функция stm32_hc_stop, которая выполняет необходимые действия по отключению.

3.5. Работа с USB-устройствами

Основные события, связанные с USB-устройствами приводят к генерации ядром USB различных прерываний.

При подключении устройства к USB-порту STM32 генерируется прерывание состояние порта, выставляя бит PCDET регистра HPRT, сигнализирующий о детектировании устройства. Наш обработчик прерываний должен обработать эту ситуацию, сначала необходимо обработать регистры, это ложится на плечи библиотеки HAL, а именно функции HAL_HCD_IRQHandler, которая выставляет необходимые регистры и вызывает пользовательскую функцию Connect_Callback, которая сообщает системе, что было подключено устройство.

Первое, что необходимо сделать после подключения устройства, согласно спецификации, это сбросить порт, что должно привести к генерации прерывания включения порта, что позволит нашей системе узнать, что порт успешно сброшен и готов к следующим действиям.

Затем начинается энумерация подключенного USB-устройства, во время которой со стороны библиотеки USB отправляются управляющие передачи, с помощью которых мы получаем информацию о подключенном устройстве.

В случае отключения USB-устройства от порта генерируется прерывание состояния порта, выставляя бит DISCINT в регистре GINTSTS, сигнализирующий об отключении устройства. Драйвер закрывает все открытые каналы и очищает порт.

3.5.1. Выполнение USB-транзакций

Уже на этапе энумерации USB-устройства с уровня библиотеки USB начинают приходить USB-транзакции, которые необходимо обработать и подать на очередь выполнения хост-контроллеру. Функция stm32_request принимает необходимую для выполнения транзакцию и подает ее дальше, в зависимости от ее типа.

В первую очередь необходимо открыть канал передачи в нужную сторону, в определенную конечную точку устройства. Это реализуется с помощью функции библиотеки HAL — HAL_HCD_HC_Init, которая вызывается перед отправкой USB-транзакции. Мы открываем один восходящий и один нисходящий для каждого типа передач.

Далее, в зависимости от направления, мы кладем передачу в очередь выполнения для соответствующего канала с помощью функции HAL_HCD_HC_SubmitRequest. В случае управляющей передачи, она состоит из SETUP части и DATA части, которые по отдельности кладутся в очередь.

При изменении состояния выполнения передачи ядро USB генерирует прерывание, которое вызывает пользовательскую функцию NotifyURBChange_Callback, внутри которой мы проверяем, что передача была выполнена успешно и можем сообщать об этом уровню библиотеки USB.

4. Апробация

4.1. Ручное тестирование

На этапе разработки, тестирование успешной инициализации хоста, подключения и энумерации выполнялись в ручную.

Первый этап это успешная инициализация корневого хаба, который видно в списке USB-устройств, на рисунке 4 представлена успешная инициализация модуля stm32_hc, в котором находится реализация драйвера хост-контроллера и вывод команды lsusb, который выводит наличие USB-устройства в системе, это наш корневой хаб.

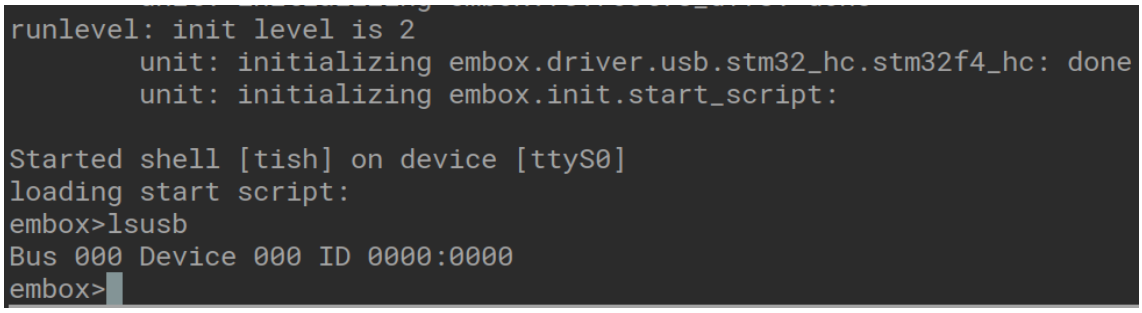

Рис. 4: Успешная инициализация модуля и хост-контроллера.

Второй этап это успешное подключение USB-устройства, его энумерация и регистрация в системе, на рисунке 5 представлена успешная инициализация модуля stm32_hc, в котором находится реализация драйвера хост-контроллера, вывод команды lsusb, который показывает наличие двух USB-устройств в системе, это корневой хаб и USB-флешка, подключенная к микроконтроллеру и вывод содержимого папки /dev, в котором мы видим /dev/sda, это наша USB-флешка.

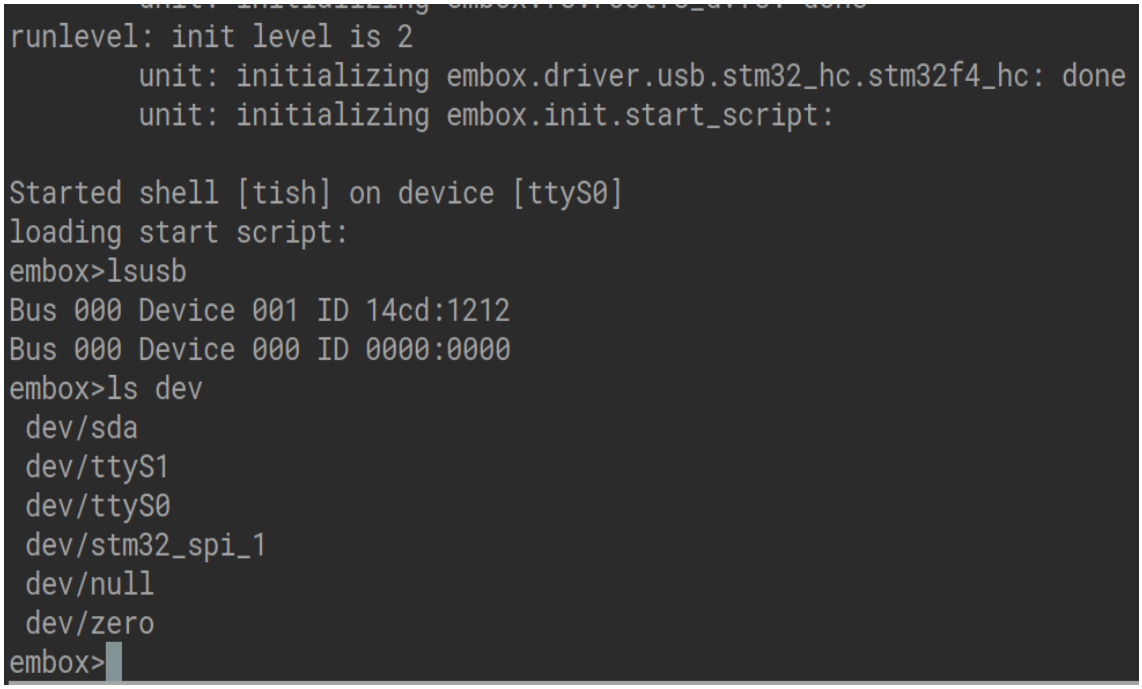

Рис. 5: Успешная инициализация USB-устройства в системе.

Третий этап это работа с USB-устройством, на рисунке 6 изображено успешное монтирование устройства /dev/sda, чтение файла read_test и запись файла write_test на USB-устройство.

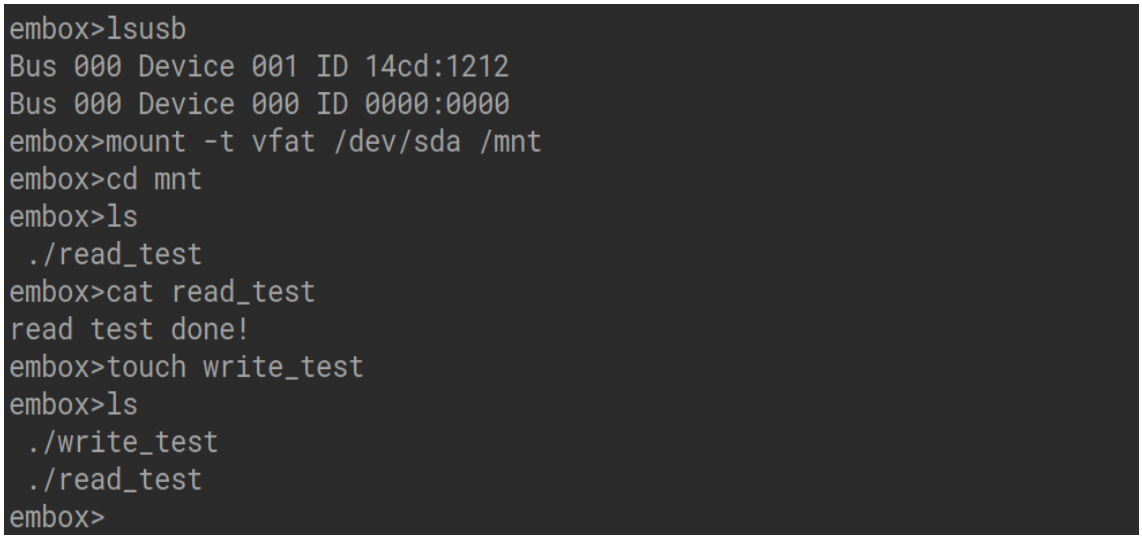

Рис. 6: Успешная работа с USB-устройством.

И на рисунке 7 изображено это же USB-устройство, подключенное к системе Linux после работы с ним в Embox, содержимое файла

read_test соответсвует выводу в Embox, а write_test корректно отображается в системе.

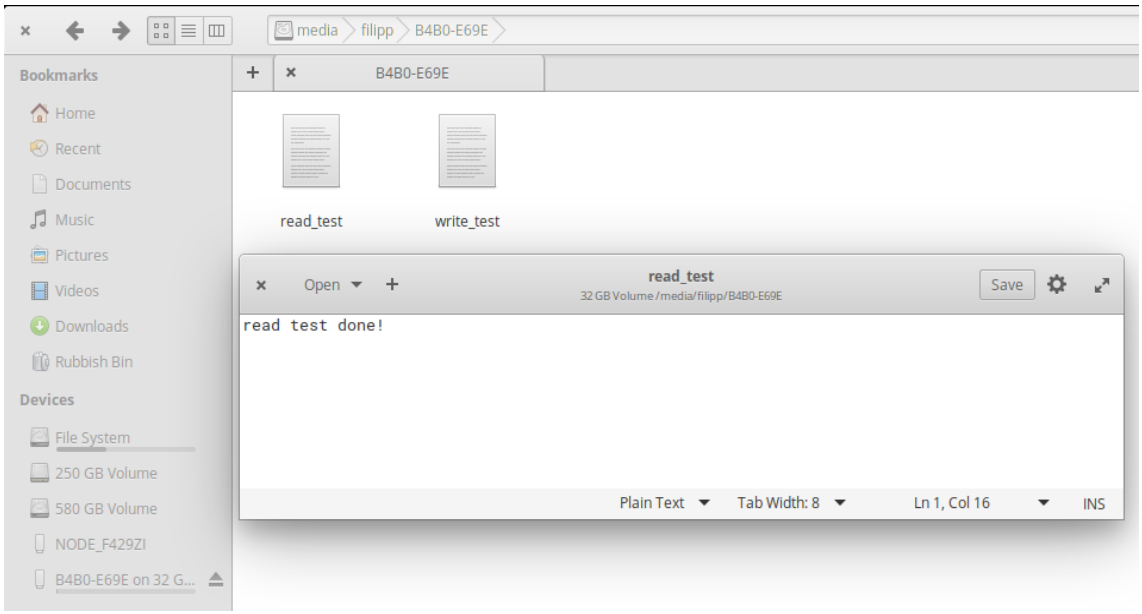

Рис. 7: Содержимое USB-устройства.

4.2. Автоматическое тестирование

Для автоматического тестирования в ОС Embox имеется команда block_dev_test, которая проверяет успешность записи и чтения на блочное устройство. На рисунке 8 представлен вывод проверки нескольких блоков на чтение и запись.

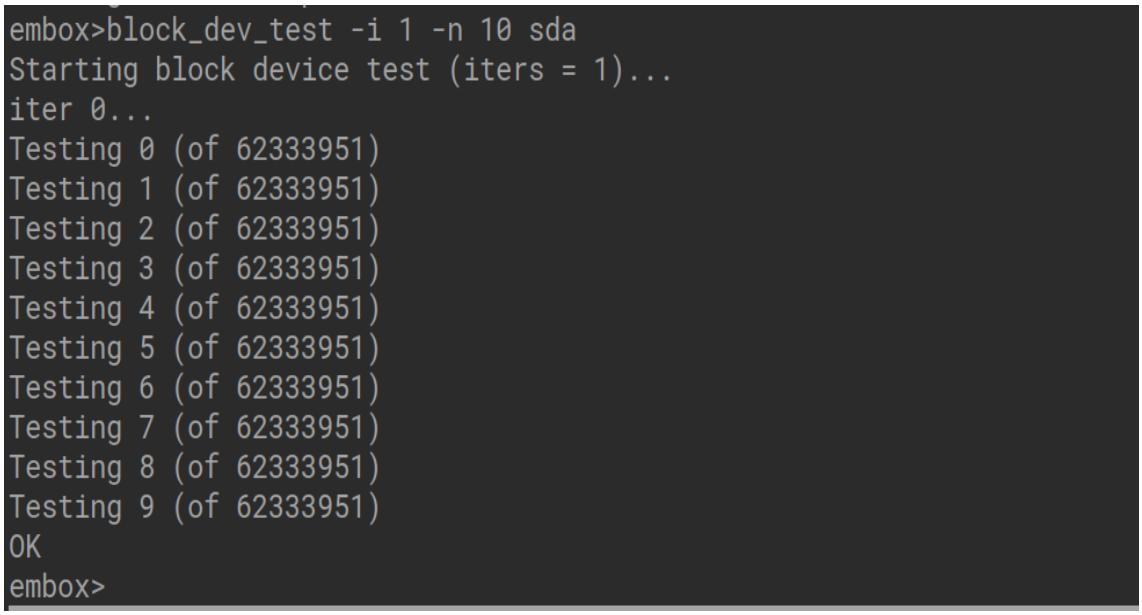

Рис. 8: Успешная проверка на чтение и запись с помощью команды block_dev_test.

Заключение

В ходе данной работы были получены следующие результаты.

- Проведен обзор предметной области, в котором был разобран USB-интерфейс, работа USB в ОС Embox и в семействе микроконтроллеров STM32.
- Спроектированы и реализованы функции интерфейса драйвера, с помощью которых выполняется управление хост-контроллером STM32 со стороны операционной системы Embox. В данный момент реализация находится в ветке репозитория Embox по адресу https://github.com/embox/embox/tree/stm32_hcd .
- Проведено тестирование с подключенным к плате USB-устройством, которое показало работоспособность полученной реализации.

Список литературы

[1] Embox. — Access mode: http://www.embox.rocks/ (online; accessed: 2021-05-20).

[2] GDB: The GNU Project Debugger. — Access mode: https://www.gnu.org/software/gdb/ (online; accessed: 2021-05-20).

[3] Martin Trevor. The insider's guide to the STM32. — Access mode: https://www.google.ru/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjDgbaMu9XtAhWMlYsKHadiD7cQFjAAegQIARAC&url=http://www.emcu.it/InsideCORTEX-1221142709.pdf (online; accessed: 2021-05-20).

[4] Microcontroller Wiki. — Access mode: https://en.wikipedia.org/wiki/Microcontroller (online; accessed: 2021-05-20).

[5] Minicom. — Access mode: https://salsa.debian.org/minicom-team/minicom (online; accessed: 2021-05-20).

[6] Nucleo-f429zi information. — Access mode: https://www.st.com/resource/en/data_brief/nucleo-f429zi.pdf (online; accessed: 2021-05-20).

[7] Openocd. — Access mode: http://openocd.org/ (online; accessed: 2021-05-20).

[8] STM32. — Access mode: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html (online; accessed: 2021-05-20).

[9] STM32 F4 Documentation. — Access mode: https://www.manualslib.com/manual/1249201/Stmicroelectronics-Stm32f405.html (online; accessed: 2021-05-20).

[10] STM32 HAL. — Access mode: https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroele pdf (online; accessed: 2021-05-20).

[11] USB Wiki. — Access mode: https://en.wikipedia.org/wiki/USB (online; accessed: 2021-05-20).

[12] Universal Serial Bus rev 2.0 Specification. — Access mode: https://www.usb.org/document-library/usb-20-specification (online; accessed: 2021-05-20).