Санкт-Петербургский государственный университет

Кафедра системного программирования

Чубуков Филипп Александрович

Улучшение USB стека Embox

Курсовая работа

Научный руководитель:
ассистент А. П. Козлов

Санкт-Петербург
2020

Оглавление

Введение

С момента своего появления USB интерфейс получил сильное распространение и практически стал основным интерфейсом подключения периферии к цифровой технике [2]. Сегодня почти ни один компьютер и подобное ему устройство не обходится без этого интерфейса, а его актуальность сохранится еще очень долго. Поэтому любая современная ОС должна поддерживать USB в максимальном объеме, ведь без этого теряется львиная доля функциональности устройства.

В Embox представлен свой USB интерфейс, но в данный момент он не обладает необходимой функциональностью и не поддерживает некоторые устройства, а именно: поддерживаются не все хост-контроллеры, отсутствует поддержка устройств с поточным типом передачи данных [1].

Встала задача выбора спецификации хост-контроллера, реализация которой будет выполнена в в рамках этой работы и выбор пал на UHCI, в виду следующих рассуждений

- Отсутствие поддержки этого хост-контроллера в проекте
- Возможность внедрения в уже реализованный USB стек Embox
- Его более простая, по сравнению с другими хост-контроллерами, реализация при полном соблюдении спецификаций USB на приоритеты обработки передач, что делает его важным для модульной системы Embox, позволяя создавать более простые сборки

1. Цель работы.

Цель данной работы заключается в улучшении имеющегося в Embox USB-интерфейса и добавления поддержки нового хост-контроллера. Для её достижения были сформулированы следующие задачи:

1. Сделать обзор устройства USB интерфейса.
2. Сделать обзор USB стека в Embox.
3. Добавить поддержку нового хост-контроллера.
4. Провести тестирование реализации.

2. Обзор проведенных исследований

2.1 Устройство USB интерфейса

USB (universal serial bus) — универсальная последовательная шина для подключения периферийных устройств к технике. На шине USB есть три типа устройств: хост-контроллер, хаб и конечное устройство. Их расположение представлено на рисунке 1.

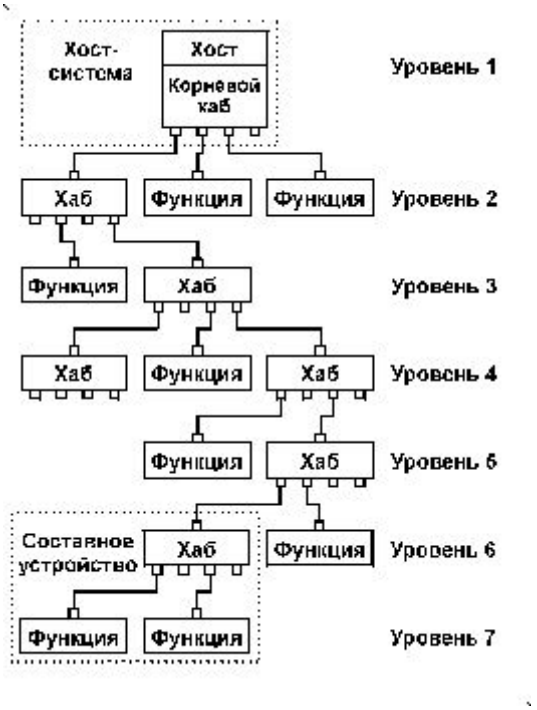

Рис. 1 Дерево устройств на шине USB [4]

Хост-контроллер — главное устройство, которое обеспечивает связь операционной системы со всеми устройствами, которые подключены к шине USB. Взаимодействие с ОС осуществляется с помощью драйвера, который индивидуален для каждой спецификации хост-контроллера.

Хаб — необходим для физического подключения конечного устройства к шине USB. Для этого предоставляет порты и транслирует трафик от хост-контроллера к конечным устройствам (нисходящий порты) и обратно (восходящие порты).

Конечные устройства (функции) — устройства, которые подключаются к шине USB [3].

Любое логическое устройство, как хаб так и функция, это набор конечных точек (endpoints), к которым открываются каналы и организовываются передачи данных. Для выполнения различных задач существуют следующие типы передач:

1) Изохронные передачи (isochronous transfers) — предназначены для передачи потоковых данных в реальном времени. Гарантируют время доставки, но не гарантируют, что все данные будут доставлены. Имеют наивысший приоритет и могут занимать до 90% пропускной способности канала, но если происходит ошибка или заканчивается время то данные не будут доставлены. Передачи этого типа используются, например, для камер или аудиоколонок.
2) Прерывания (interrupts) — предназначены для спонтанных небольших сообщений, но с гарантированным временем обслуживания и гарантированной доставкой. Примером может служить USB клавиатура или компьютерная мышь.
3) Передача массивов данных (bulk data transfers) — предназначены для передачи данных когда требуется гарантия доставки в целостности, а не скорость. Такие передачи имеют самый низкий приоритет, занимая оставшуюся после других пропускную способность шины. Флешки и принтеры используют этот тип передачи.
4) Управляющие передачи (control transfers) — передачи типа запрос-ответ. С помощью них передаются команды управления устройствами[3].

2.2 Спецификации хост-контроллера

Хост-контроллер берет на себя задачу общения с конечными устройствами, но для связи с ним операционной системе необходим драйвер для данного контроллера [3].

Для разных версия USB представлены различные спецификации хост-контроллеров:

1) USB 1.1 — в рамках этой спецификации представлены два интерфейса хост-контроллера: UHCI (Universal Host Controller Interface) и OHCI (Open Host Controller Interface). Они различаются методом доступа к регистрам.
2) USB 2.0 — соответствует спецификации EHCI (Enhanced Host Controller Interface).
3) USB 3.0 — соответствует спецификации XHCI (eXtensible Host Controller Interface) [5].

Драйвер хост-контроллера выполняет задачу управления контроллером, составления расписания его передач и обработки ошибок. Общение с клиентскими программами и операционной системой происходит на более высоком уровне архитектуры — драйвере шины USB, который отправляет запросы драйверу хост-контроллера [3]. Абстрактная модель взаимодействия представлена на рисунке 2.

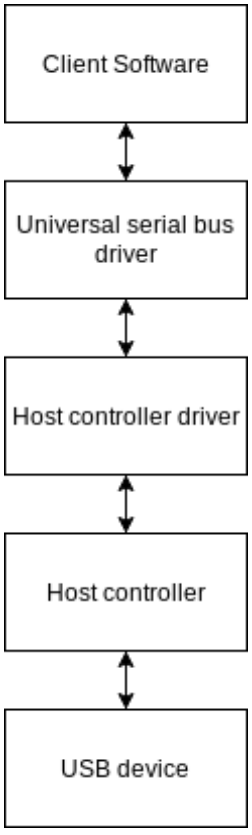

Рис. 2 Модель взаимодействия устройств

2.3 USB интерфейс в Embox

Embox — это конфигурируемая операционная система реального времени, разрабатываемая для встраиваемых систем [1].

Одной из основных особенностей Embox является максимальная структурированность, позволяющая представить ядро как набор взаимосвязанных модулей, каждому из которых можно задать требуемые параметры. Это дает возможность тонко настроить систему и создать образ, нацеленный на конкретную задачу, будь то средство для тестирования аппаратного обеспечения, либо полноценная операционная система с виртуальной памятью, включающая сетевую и файловую подсистемы. К тому же такой принцип построения упрощает отладку отдельных модулей системы, а также портирование на новые платформы.

В Embox уже реализован уровень драйвера шины USB, что упрощает нашу задачу внедрения нового драйвера хост-контроллера и позволяет сосредоточиться именно на нем.

3. Реализация

3.1 Обзор Universal Host Controller Interface

В процессе работы была создана реализация драйвера хост-контроллера для устройства, которое имеет следующую структуру.

3.1.1 Генерация кадров

Хост-контроллер поддерживает доставку данных в реальном времени, генерируя Start Of Frame (SOF) пакет каждые 1 мс. В каждый SOF пакет добавляется номер кадра, в котором он выполняется. По истечении 1 мс наступает состояние End Of Frame (EOF) и начинается новый кадр, и хост контроллер генерирует новый SOF пакет с новым соответствующим номером.

Внутри кадра данные передаются в виде пакетов информации. За время кадра сначала обрабатывается очередь изохронных передач, те, которые не успевают за выделенное время теряются. Затем обрабатываются остальные типы из общей очереди [6]. Распределение продемонстрировано на рисунке 3.

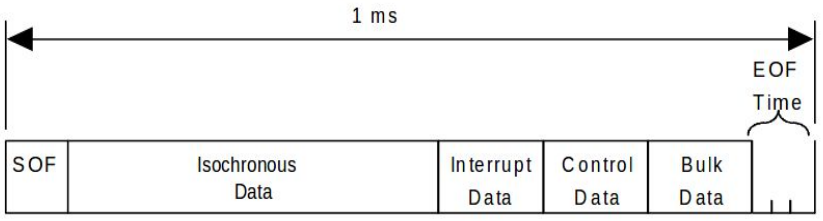

Рис. 3 Распределение передач на кадр [6]

3.1.2 Базовые регистры UHCI

Для прямой связи с хост-контроллером используются базовые регистры, которые лежат в пространстве PCI-устройства.

Основные блоки регистров [6]:

- USBCMD — командные регистры, запись в регистр приводит к выполнению команды.
- USBSTS — регистры статуса, дают информацию о состоянии хост-контроллера.
- USBINTR — регистры работы с прерываниями.
- FRNUM — регистр, соответствующий номеру текущего кадра.
- FLBASEADD — регистр, соответствующий адресу списка указателей на кадр.
- PORTSC — регистр для получения информации о состоянии портов.

3.1.3 Структура данных Frame List

Представляет собой массив из 1024 записей, каждая из которых соответствует отдельному кадру. Запись является ссылкой на транзакции которые должен выполнять хост-контроллер.

Заполняется этот массив драйвером хост-контроллера согласно запросам от драйвера шины USB. Контроллеру остается выбрать нужную запись в массиве с помощью регистра FLBASEADD и счетчика кадров и выполнять транзакции по ссылке [6].

3.1.4 Структура данных Transfer Descriptor

Дескрипторы передач содержат указатель на буфер данных, которые мы хотим передать и управляющие поля.

Для различных типов передач существуют разные дескрипторы передач. Для изохронных передач драйвер хост-контроллера соединяет их в очереди, а ссылку на первый дескриптор в очереди, кладет в соответствующую запись в списке кадров [6].

Остальные типы передач объединяются в очереди с помощью следующей структуры - Queue Head, пример расписания UHCI предложен на рисунке 4.

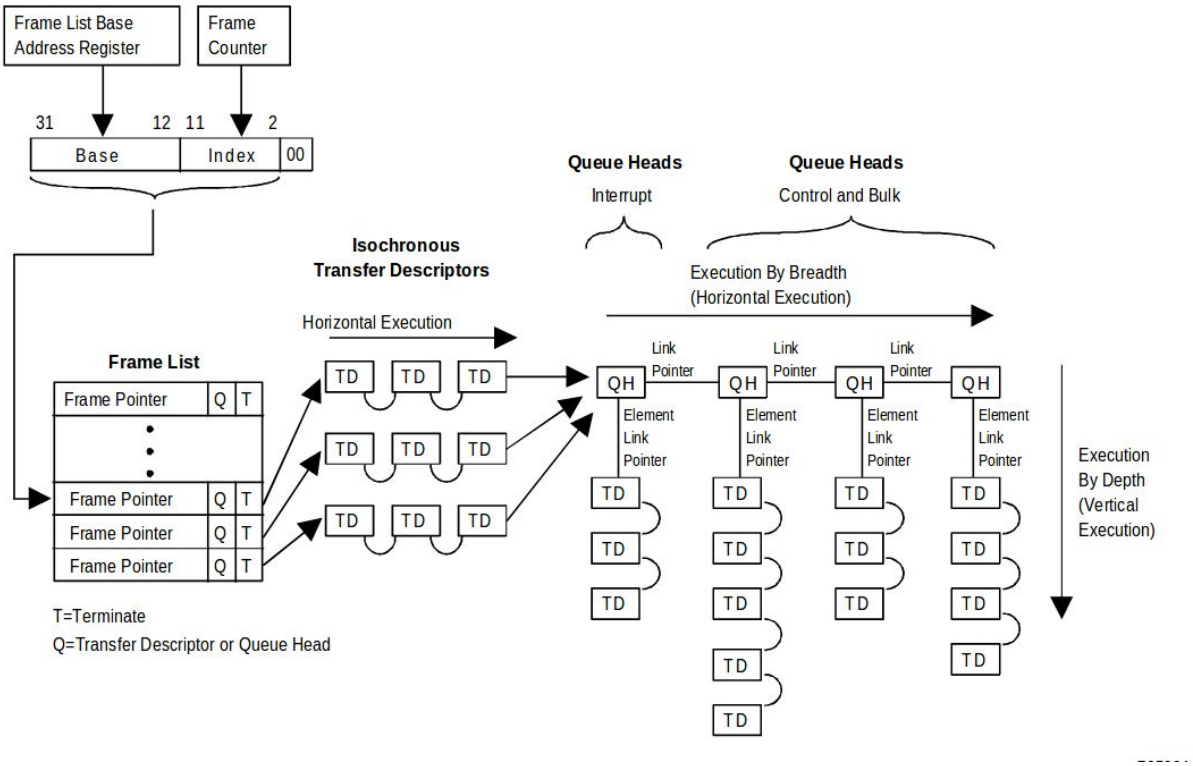

рис. 4 Пример расписания UHCI [6]

3.1.5 Структура данных Queue Head

Главы очереди предназначены для организации не изохронных дескрипторов передач в очереди. Последний дескриптор изохронной очереди ссылается на QH, который объединяет список передач одного типа. Сперва обрабатываются прерывания, затем управляющие запросы, и в конце массивы передачи данных [6].

3.2 Выбор архитектуры

Архитектура x86 является основной платформой для разработки в Embox, поэтому было принято решение сосредоточиться на реализации драйвера и его тестировании именно на этой архитектуре.

3.3 Эмуляция архитектуры и устройства

Средствами QEMU выполняется эмуляция необходимой архитектуры и хост-контроллера определенной спецификации.

Для выполнения этой задачи был изменен скрипт запуска Embox — auto_qemu, в него добавлено подключение необходимого хост-контроллера в виде эмуляции pci-устройства.

Реализация представлена в scripts/qemu/

- auto_qemu — скрипт запуска QEMU

3.4 Реализация драйвера

Реализация представлена в src/drivers/usb/hc

- uhci_pci.c, uhci.h — драйвер хост-контроллера.
- Mybuild — описание модуля драйвера и зависимостей, необходимых для его корректной сборки и работы.

Первое, что было необходимо сделать, это определение хост-контроллера и его регистрация в системе. Embox умеет работать с pci-устройствами и при запуске системы сопоставляет устройство и драйвер, если таковой имеется. Далее необходимо принимать запросы с уровней выше, организовывать их в расписание для хост-контроллера и, при необходимости, отвечать на них.

В реализации представлены необходимые функции для корректной работы с драйвером шины USB:

- uhci_start — инициализация структур хост-контроллера, запуск и регистрация в системе
- uhci_stop — остановка работы хост-контроллера
- uhci_root_hub_control — работа с корневым хабом хост-контроллера
- uhci_request — получение, составление расписания и отправка хост-контроллеру запросов от драйвера шины USB

3.5 Внедрение реализации в Embox

Для внедрения была создана новая конфигурация для ОС Embox, за основу взята templates/x86/qemu, в которую был добавлен реализованный модуль uhci.

Реализация представлена в templates/x86/test/uhci:

- mods.config — содержит описание подключаемых модулей

3.6 Процесс сборки и запуска Embox с реализацией

1) make confload-x86/test/uhci — загрузка необходимой конфигурации в директорию conf
2) make — запускает процесс компиляции в соответствии с файлом conf/mods.config, компилирую необходимые модули в нужном порядке.
3) ./scripts/qemu/auto_qemu — запуск виртуальной машины вместе со скомпилированным Embox.

4. Тестирование.

4.1 Проверка инициализации хост-контроллера

Было необходимо проверить корректность инициализации pci-устройства в системе.

Рисунок 5 демонстрирует информацию запуска модулей Embox, где происходит инициализация устройства, которое наш драйвер посчитал хост-контроллером необходимой спецификации.

```
runlevel: init level is 1
        unit: initializing embox.net.neighbour: done
        unit: initializing embox.mem.slab: done
        unit: initializing embox.driver.pci: done
        unit: initializing embox.driver.usb.hub: done
        unit: initializing embox.net.net_entry: done
        unit: initializing embox.net.tcp: done
        unit: initializing embox.driver.tty.serial: done
        pci: INTEL UHCI usb host driver inserted
        pci: INTEL UHCI usb host handles 8086:7020 bus 0 slot 1 func 2
```

рис. 5 Инициализация устройства

Рисунок 6 демонстрирует вывод команды lspci в Embox, на котором мы видим, что это pci-устройство является хост-контроллером спецификации UHCI.

```
root@embox:/#lspci
00: 0.0 (PCI dev 8086:1237) [6 0]
        Host bridge: Intel Corporation Intel 82441 CPU to PCI bridge (rev 02)
00: 1.0 (PCI dev 8086:7000) [6 1]
        ISA bridge: Intel Corporation Intel 82371SB (PIIX3) ISA (rev 00)
00: 1.1 (PCI dev 8086:7010) [1 1]
        IDE controller: Intel Corporation Intel 82371SB (PIIX3) IDE (rev 00)
00: 1.2 (PCI dev 8086:7020) [12 3]
        USB device: Intel Corporation PIIX3 UHCI USB host (rev 01)
```

Рис. 6 Вывод lspci

Из этого примера видно, что устройство правильно идентифицировано и проинициализировано драйвером.

4.2 Работа с USB-устройством

В рамках второго теста было проведено тестирование работы usb-устройства на примере флеш накопителя.

Для этого в Embox уже есть готовый скрипт, который создает образ диска, а затем добавляет его к скрипту запуска QEMU в виде usb флеш-накопителя.

На рисунке 7 продемонстрировано монтирование образа флеш-накопителя, чтение созданного ранее файла и создание нового файла средствами Embox.

```
root@embox:/#mount -t vfat /dev/sda mnt
root@embox:/#cat mnt/read_test
READ_TEST SUCCESS
root@embox:/#touch mnt/write_test
root@embox:/#ls /mnt
 /mnt/read_test
 /mnt/write_test
root@embox:/#umount mnt
root@embox:/#ls /mnt
root@embox:/#
```

Рис. 7 Работа с флеш-накопителем в ОС Embox

Чтение прошло успешно, теперь проверим запись, проверив наличие созданного файла в образе диска, примонтиров его средствами linux, на рисунке 8 продемонстрировано монтирование и вывод файлов в образе.

```
13:12:31 ~/Documents/kursovaya/embox [0]$ sudo mount -t vfat usbdisk.img ./tmp
13:12:33 ~/Documents/kursovaya/embox [0]$ ls ./tmp
read_test  write_test
13:12:35 ~/Documents/kursovaya/embox [0]$ sudo umount ./tmp
13:12:38 ~/Documents/kursovaya/embox [0]$ ls ./tmp
13:12:39 ~/Documents/kursovaya/embox [0]$
```

Рис. 8 Проверка действий, совершенных в ОС Embox

Тестирование показало, что реализация драйвера хост-контроллера, внедренная в Embox, функционирует.

Заключение.

В ходе работы получены следующие результаты:

1. Сделан обзор USB интерфейса.
2. Сделан обзор USB стека проекта Embox.
3. Реализована поддержка спецификации UHCI хост-контроллера.
4. Проведено тестирование работоспособности драйвера хост-контроллера

Список литературы

[1] Embox wiki. — URL: https://github.com/embox/embox/wiki (дата обращения: 10.05.2020)

[2] USB — URL: https://en.wikipedia.org/wiki/USB (дата обращения: 10.05.2020)

[3] Universal Serial Bus rev 1.1 Specification — URL: http://esd.cs.ucr.edu/webres/usb11.pdf (дата обращения: 10.05.2020)

[4] USB-devices tree — URL: http://kazus.ru/nuke/spaw/images/1/top_usb.gif

[5] Host controller interface — URL: https://en.wikipedia.org/wiki/Host_controller_interface_(USB,_Firewire) (дата обращения 10.05.2020)

[6] UHCI design guide — URL: ftp://ftp.netbsd.org/pub/NetBSD/misc/blymn/uhci11d.pdf (дата обращения 10.05.2020)