ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)**

Кафедра вычислительных машин, комплексов, систем и сетей
Ю.С. Гладышев

ИНТЕРФЕЙСЫ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

ПОСОБИЕ

по проведению практических занятий
по теме «ИНТЕРФЕЙС USB 2.0»

*для студентов II курса
специальности 230101 (направления 09.03.01)
и IV курса специальности 090302 (10.05.02)
очной формы обучения*

Москва-2015

ББК 6Ф7.3
Г52

Рецензент канд. техн. наук, доц. Л.А. Вайнейкис

Гладышев Ю.С.

Г52 Интерфейсы вычислительных систем: пособие по проведению практических занятий по теме «Интерфейс USB 2.0». - М.: МГТУ ГА, 2015. - 32 с.

Данное пособие издается в соответствии с рабочей программой учебной дисциплины «Интерфейсы вычислительных систем» по учебному плану для студентов II курса специальности 230101 (направления 09.03.01) и IV курса специальности 090302 (10.05.02) очной формы обучения.

Рассмотрено и одобрено на заседаниях кафедры 25.03.14 г. и методического совета 25.03.14 г.

Подписано в печать 10.10.2014 г.

Печать офсетная	Формат 60x84/16	1,64 уч.-изд. л.
1,86 усл. печ. л.	Заказ № 1883/	Тираж 100 экз.

Московский государственный технический университет ГА
125993 Москва, Кронштадтский бульвар, д.20
Редакционно-издательский отдел
125493 Москва, ул. Пулковская, д.6а

1. Ведение

Процессор работает под управлением программы, которая находится в ОЗУ, при этом он обрабатывает данные, которые также находятся в ОЗУ. Результаты обработки информации ПРЦ1 должен выводить из своего ОЗУ1 в ПУ, а новую порцию данных вводить в своё ОЗУ1 из ПУ.

ПУ, в свою очередь, является специальной ЭВМ, которая содержит свой процессор (ПРЦ2), предназначенный для выполнения операций ввода/вывода.

Например, процессор клавиатуры:

1) вычисляет скэн-код нажатой клавиши и помещает его в БР передатчика.
2) Затем скэн-код автоматически (без участия процессора) передаётся в БР приёмника, находящийся в контроллере клавиатуры.
3) На следующем этапе скэн-код, пройдя известные преобразования, в виде ASCII-кода попадает в ОЗУ1 под управлением ПРЦ1.

Процесс вывода информации на печать также можно представить состоящим из трёх этапов:

1) На первом этапе ПРЦ1 помещает байты выводимых данных из ОЗУ1 в БР передатчика.
2) На втором этапе байты данных автоматически передаются из БР передатчика в БР приёмника.
3) На третьем этапе байты данных под управлением ПРЦ2 (принтера) пересылаются из БР приёмника в ОЗУ1 принтера, а затем распечатываются на бумаге.

Передача данных из БР передатчика в БР приёмника реализуется посредством интерфейсов. Простейшим интерфейсом ввода является ряд тристабильных вентилей. Простейшим интерфейсом выводя является ряд D-триггеров. Для организации автоматической передачи данных между БР передатчика и БР приёмника простейшие интерфейсы дополняются логическими схемами, повышающими интеллект интерфейса (он становится контроллером), вплоть до применения специализированных процессоров.

Можно сказать, что обмен данными между ОЗУ1-ЭВМ и ОЗУ2-ПУ осуществляется путём взаимодействия между двумя ПРЦ, примеры которого для клавиатуры и принтера были описаны выше.

Для повышения скорости обмена информацией между ЭВМ и ПУ большую роль играют линии связи. Современные технологии позволили создать новые линии связи, которые способны существенно повысить скорость передачи данных. Так, например, интерфейс USB 2.0 может обеспечить скорость передачи данных 480 Мбит/с. Если эту величину сравнить со скоростью передачи 19200бит/с (0.0192≈0.02Мбит/с) интерфейса RS-232C, то скорость увеличится примерно в 24000 раз. Такое увеличение скорости позволяет посредством такого интерфейса подключить к ЭВМ много устройств, т.е. , используя одни и те же линии передачи данных (шину USB), организовать много каналов связи ЭВМ и ПУ.

Отметим ряд основных особенностей интерфейса USB 2.0 по сравнению с предыдущим интерфейсом RS-232C.

1) В традиционном интерфейсе любое ПУ представлено на системной магистрали в виде набора регистров своего контроллера. БР данных и РКС (регистра команд и состояний), которые определяют физический объект устройства PDO (Physical Device Object). В USB 2.0 контроллер «прописывает» на системной магистрали шину к которой можно подключить до 127 ПУ, почему он называется хост-контроллером (HC).

2) Обмен между буфером HC и ОЗУ ЭВМ осуществляется по прерыванию. Для его обеспечения HC выделяется отдельная линия запросов прерывания. Для обмена между буфером HC и буфером ПУ HC использует «поллинг опроса», что позволяет экономить линии запроса прерывания.

3) Передача данных между каждым ПУ на шине и соответствующими буферами хоста (HC) разбивается на транзакции, которые передаются вперемежку по расписанию по шине USB, при этом не нарушая процесса обмена между ПУ и ЭВМ в общем.

4) Поскольку к шине USB 2.0 можно подключить много устройств, то на шине USB 2.0 одновременно может быть организовано много каналов связи.

На множестве каналов связи реализуется 4 типа передач, которые удовлетворяют всему многообразию ПУ подключаемых к шине USB 2.0, причём из них два типа передач привязаны ко времени – это:

1. изохронные передачи (isochronous transfers),
2. передачи по прерыванию (interrupt transfers),

и два типа передач не связанных со временем – это:

3. управляющие передачи (control transfers),
4. передачи массивов (bulk transfers)

Каждый из этих типов предъявляет свои требования к полосе пропускания шины. Исходя из них определяется составляется расписание движения транзакций в составе кадров по шине USB. Для примера на рис.1 отражено распределение полосы пропускания в кадре для универсального хост-контроллера (UHC).

SOF	Изохронные транзакции	Транзакции прерываний	Транзакции управления	Транзакции передач массивов	Свободное время	***EOF***

Рис.1. Распределение полосы пропускания для UHC

2. Взаимодействие ПУ и ПК посредством шины USB.

2.1 Компоненты взаимодействия шины USB.

Исходные данные для планирования расписания движения транзакций осуществляются на основе запросов пользователя к драйверу устройства USBDD, который формирует пакеты запросов ввода/вывода (***IRPS*** – Input/Output Request Packet).

Компонентами, взаимодействующими на шине USB являются: *USBDD,*

*USBD, HCD, HC, SIE, логическое устройство USB, функциональное устройство USB. С*хема взаимодействия перечисленных компонентов USB представлена на рис.2.

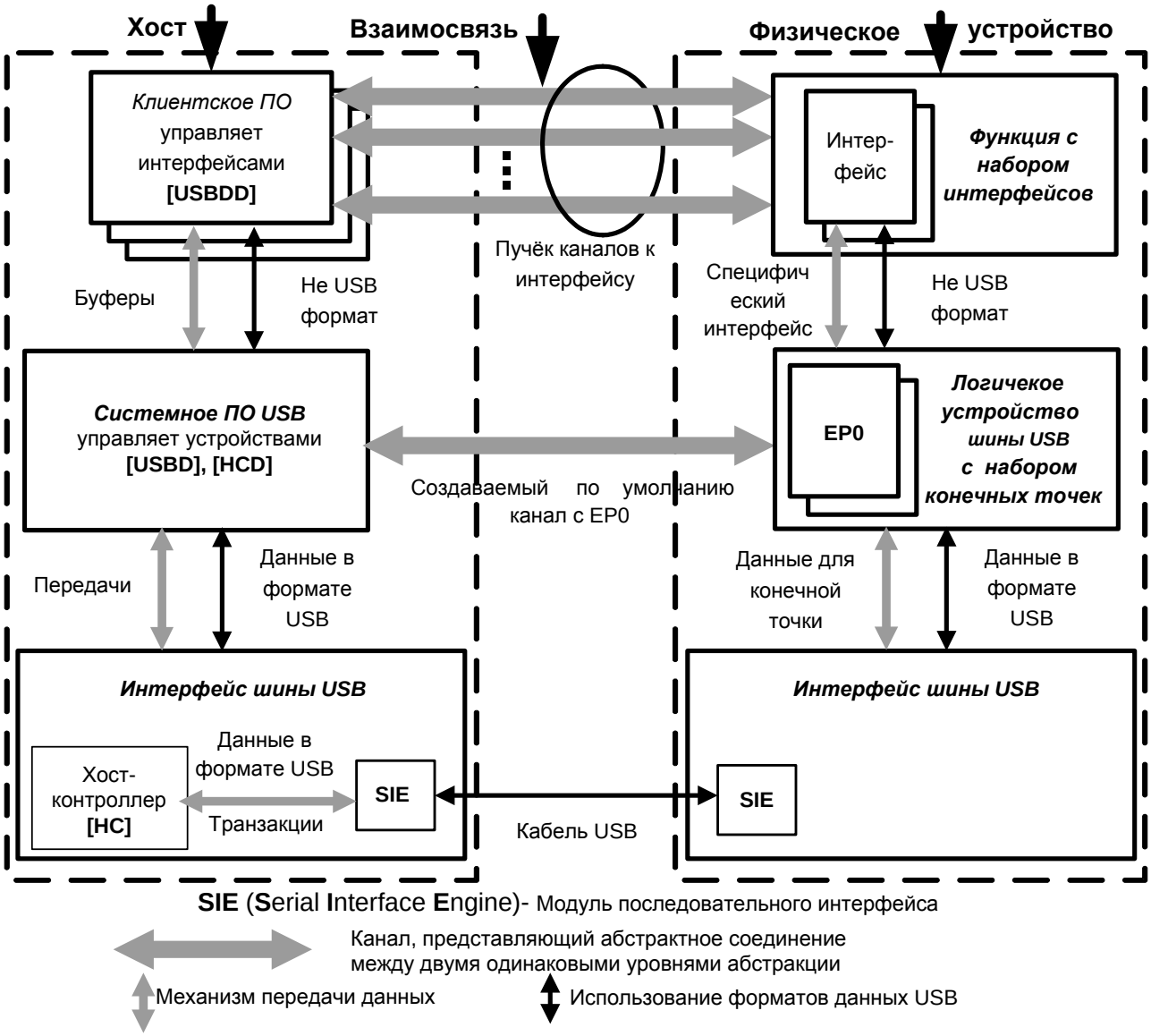


Рис.2. Взаимодействие компонентов USB.

Рассмотрим каждый из компонентов:

1) ***Клиентское программное обеспечение (CSw-Client Software)*** - драйверы устройств USB (***USBDD****-USB Device Driver*), обеспечивающие доступ к устройствам со стороны прикладного ПО. Эти драйверы взаимодействуют с устройствами только через программный интерфейс с общим драйвером шины USB (USBD). Непосредственного обращения к каким-либо регистрам аппаратных средств драйверы устройств USB не выполняют. ***CSw*** отвечает за формирование пакета запроса на ввод/вывод (***IRP, Input/output Request Packet***), который содержит сведения об адресе и длине буфера в оперативной памяти, куда или откуда эти данные будут помещаться/извлекаться.

2) ***Компонент системное обеспечение USB*** состоит из драйвера шины USB (***USBD- Universal Serial Bus Driver***) и драйвера хост-контроллера (***HCD- Host Controller Driver***). ***USBD*** «заведует» всеми USB-устройствами системы, их нумерацией, конфигурированием, предоставлением служб, распределением пропускной способности шины, мощности питания и т. д.;

Когда клиентский компонент передаёт IRP запрос компоненте системного обеспечения, то USBD преобразует его в одно или несколько транзакций шины и затем передаёт получившийся перечень транзакций HCD (рис.3).

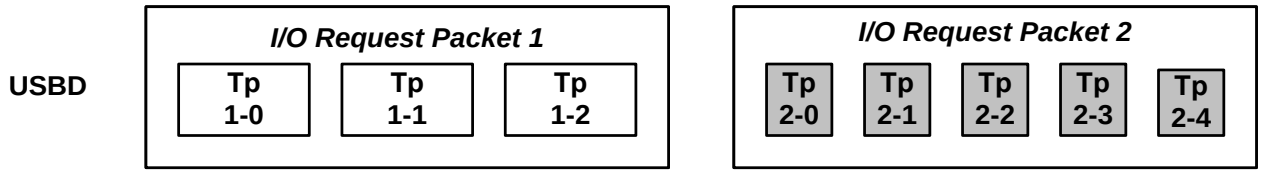


Рис.3. Разбиение запросов на перечень транзакции драйвером USBD

HCD выполняет с этим перечнем транзакций следующее:

1. планирует исполнение полученных транзакций, добавляя их к списку транзакций (рис.4);
2. извлекает из списка очередную транзакцию и передаёт её компоненте HC интерфейса шины USB;
3. отслеживает состояние каждой транзакции вплоть до её завершения.

Системный драйвер (***USBD+HCD***) отвечает за выполнение следующих действий:

4. распределение полосы пропускания шины USB,
5. назначение логических адресов каждому физическому USB-устройству.

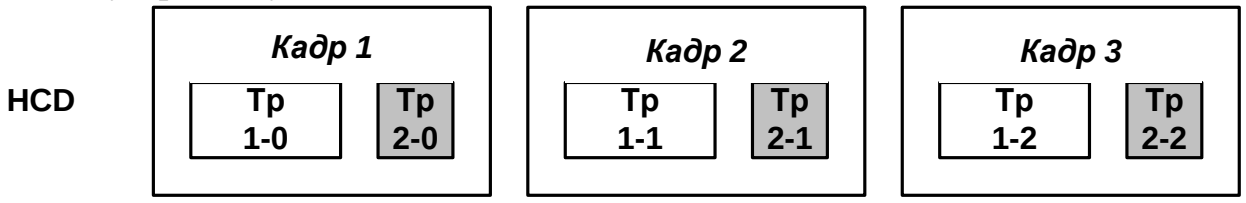


Рис. 4. Планирование транзакций драйвером HCD

3) ***Компонент хост-контроллер интерфейса шины USB (HC)*** полученные отдельные транзакции преобразует в соответствующую последовательность операций шины. В результате формирует пакеты (рис.5). Пакеты передаются последовательной передачей бит по методу **NRZI.**

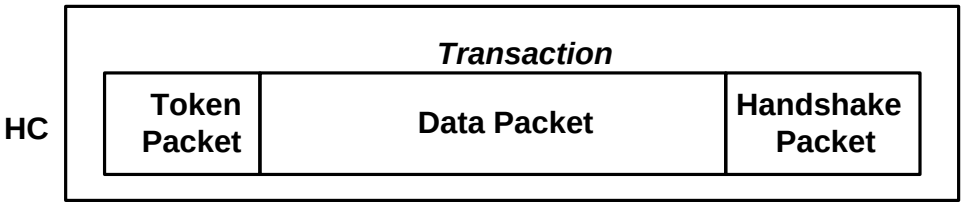


Рис.5. Формирование транзакции из пакетов хост-контроллером HC

4) ***Компонент интерфейса шины периферийного устройства –SIE-*** взаимодействует с интерфейсным компонентом шины USB на стороне хоста и передаёт кадры данных от хоста периферийному устройству в формате, определяемом спецификацией USB. Затем он передаёт эти кадры вверх - компоненте логического USB- устройства.

5) ***Компонент логическое USB- устройство*** представляет собой набор независимых конечных точек (***Endpoint, EP***), с которыми хост-контроллер (и клиентское ПО) обменивается информацией.
Каждому логическому устройству USB (как функции, так и хабу) конфигурационная часть ПО хоста назначает свой адрес (1-127), уникальный на данной шине USB.

Каждая конечная точка логического устройства идентифицируется своим номером (0-15) и направлением передачи (***IN*** - передача к хосту, ***OUT*** - от хоста). Точки ***IN4*** и ***OUT4,*** к примеру, представляют собой разные конечные точки, с которыми могут общаться даже модули клиентского ПО. Набор конечных точек зависит от устройства, но всякое устройство USB обязательно имеет двунаправленную конечную точку 0 (***EP0***), через которую осуществляется его общее управление. Для прикладных целей используются конечные точки с номерами 1-15 (1-2 для низкоскоростных устройств).

Адрес устройства, номер и направление конечной точки однозначно идентифицируют приемник или источник информации при обмене НС с устройствами USB. Каждая конечная точка имеет набор характеристик, описывающих поддерживаемый тип передачи данных (изохронные данные, массивы, прерывания, управляющие передачи), размер пакета, требования к частоте обслуживания.

6) ***Компонент функциональное устройство*** может выполнять несколько функциональных задач: например, привод CD-ROM может обеспечивать проигрывание аудиодисков и работать как устройство хранения данных.

Для решения каждой задачи в устройстве определяется ***интерфейс - набор конечных точек, предназначенных для выполнения данной задачи, и правила их использования***. Таким образом, каждое устройство должно обеспечивать один или несколько интерфейсов.

Наличие нескольких интерфейсов позволяет нескольким драйверам, каждый из которых обращается только к своему интерфейсу (представляющему часть устройства USB), работать с одним и тем же устройством USB. Каждый интерфейс может иметь один или несколько альтернативных вариантов (альтернативных установок - alternate settings), из которых в данный момент активным может быть только один. Варианты различаются наборами (возможно, и характеристиками) используемых конечных точек.

Набор одновременно поддерживаемых интерфейсов составляет конфигурацию устройства. Устройство может иметь одну или несколько

возможных конфигураций, из которых на этапе конфигурирования хост выбирает одну, делая ее активной. От выбранной конфигурации зависит доступная функциональность, и зачастую - потребляемая мощность. Пока устройству не назначен номер выбранной конфигурации, оно не может функционировать в прикладном смысле и ток потребления от шины не должен превышать 100 мА. Хост выбирает конфигурацию исходя из доступности всех ресурсов, затребованных данной конфигурацией, включая и ток потребления от шины.

2.2 Схема передачи данных между компонентами

На рис.6 представлена схема взаимодействия компонентов при передаче данных.

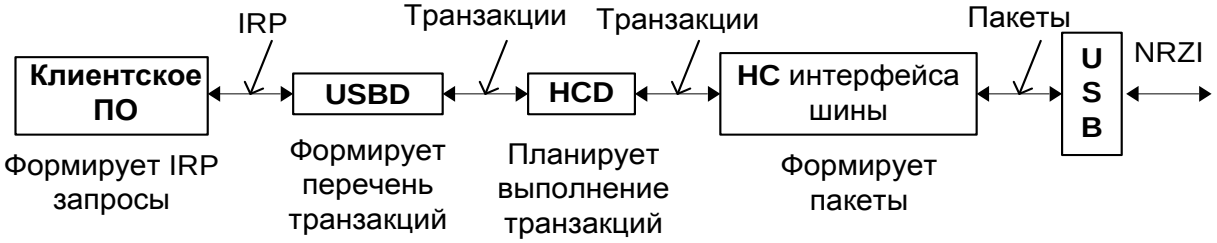


Рис.6. Схема передачи данных между компонентами.

Для передачи или приема данных клиентское ПО (***USBDD***) посылает по каналу *пакет запроса ввода/вывода* - ***IRP*** *(Input/Output Request Packet)* и ждет уведомления о завершении его отработки. Формат IRP определяется реализацией драйвера ***USBD*** в конкретной ОС. В IRP имеются только сведения о запросе (*местоположение буфера передаваемых данных в оперативной памяти и длина передачи*); от свойств конкретного текущего подключения (скорость, допустимый размер пакета) драйвер устройства (клиентское ПО) абстрагируется.

Отработкой запроса в виде транзакций на шине USB занимается драйвер ***USBD***; при необходимости он разбивает на части длинные запросы (пакеты), пригодные для передачи за одну транзакцию. Действие остальных компонент описано выше, а также ясно из рис.6.

3. Модель передачи данных.

Каждое устройство USB представляет собой набор независимых конечных точек (Endpoint-EP), с которыми хост-контроллер обменивается информацией. Конечные точки описываются следующими параметрами:

1. требуемой частотой доступа к шине и допустимыми задержками обслуживания;
2. требуемой полосой пропускания канала;
3. номером точки;
4. требованиями к обработке ошибок;
5. максимальными размерами передаваемых и принимаемых пакетов;
6. типом обмена;
7. направлением обмена (для сплошного и изохронного обменов).

Каждое устройство обязательно имеет конечную точку с номером 0 (***EP0***), используемую для инициализации, общего управления и опроса его состояния. Эта точка всегда сконфигурирована при включении питания и подключении устройства к шине. Она поддерживает передачи типа управление (Control).

Кроме нулевой точки, устройства-функции могут иметь дополнительные точки, реализующие полезный обмен данными. Низкоскоростные (LS) устройства могут иметь до двух дополнительных точек, полноскоростные (FS)- до 16 точек ввода и 16 точек вывода (протокольное ограничение). Точки не могут быть использованы до их конфигурирования (установления согласованного с ними канала).

Каналом (Pipe) в USB называется ***модель передачи данных между*** хост-контроллером и конечной точкой (Endpoint) устройства. Имеются два типа каналов: ***потоки*** (Stream) и ***сообщения*** (Message).

- ***Поток*** доставляет данные от одного конца канала к другому, он всегда однонаправленный. Один и тот же номер конечной точки может использоваться для двух поточных каналов ввода и вывода. Поток может реализовывать следующие типы обмена: ***сплошной, изохронный*** и ***прерывания***. Доставка всегда идет в порядке "первым вошел - первым вышел" (FIFO); с точки зрения USB, данные потока неструктурированны.
- ***Сообщения*** - формат, определенный спецификацией USB. Хост посылает *запрос* к конечной точке, после которого передается (принимается) пакет *сообщения* за которым следует пакет с информацией *состояния* конечной точки. Последующее сообщение нормально не может быть послано до обработки предыдущего, но при отработке ошибок возможен сброс не обслуженных сообщений. Для доставки сообщений используется только обмен типа - ***управление.***

С каналами связаны характеристики, соответствующие конечной точке (полоса пропускания, тип сервиса, размер буфера и т. п.). Каналы организуются при конфигурировании устройств USB. Для каждого включенного устройства существует канал сообщений (***Control Pipe 0***), по которому передается информация конфигурирования, управления и состояния.

4. Типы передач данных

USB поддерживает как однонаправленные, так и двунаправленные режимы связи. Передача данных производится между ПО хоста и конечной точкой устройства. Устройство может иметь несколько конечных точек, связь с каждой из них (канал) устанавливается независимо.

Архитектура USB допускает четыре базовых типа передачи данных:

- ***Управляющие посылки*** (***Control Transfers)***, используемые для конфигурирования во время подключения и в процессе работы для управления устройствами. Протокол обеспечивает гарантированную доставку данных.

Длина поля данных управляющей посылки не превышает 64 байт HS/FS и 8 байт LS.

- ***Сплошные передачи (Bulk Data Transfers)*** сравнительно больших пакетов без жестких требований ко времени доставки. Передачи занимают всю свободную полосу пропускания шины. Пакеты имеют поле данных размером 8, 16, 32, 64 байта FS и до 512 байт для HS. Приоритет этих передач самый низкий, они могут приостанавливаться при большой загрузке шины. Допускаются только на полной скорости передачи.
- ***Прерывания (Interrupt)***- короткие (1-1024) байта HS, (1-64) байт FS, (1-8) байт LS. Прерывания имеют спонтанный характер и должны обслуживаться не медленнее, чем того требует устройство. Предел времени обслуживания устанавливается в диапазоне 1-255 мс для полной скорости и 10-255 мс - для низкой.
- ***Изохронные передачи (Isochronous Transfers)*** - непрерывные передачи в реальном времени от (1-1024) байт HS/FS, занимающие предварительно согласованную часть пропускной способности шины и имеющие заданную задержку доставки. В случае обнаружения ошибки изохронные данные передаются без повтора - недействительные пакеты игнорируются. Пример - цифровая передача голоса. Пропускная способность определяется требованиями к качеству передачи, а задержка доставки может быть критичной, например, при реализации телеконференций.

Полоса пропускания шины делится между всеми установленными каналами. Выделенная полоса закрепляется за каналом, и если установление нового канала требует такой полосы, которая не вписывается в уже существующее распределение, запрос на выделение канала отвергается.

Архитектура USB предусматривает внутреннюю буферизацию всех устройств, причем, чем большей полосы пропускания требует устройство, тем больше должен быть его буфер. USB должна обеспечивать обмен с такой скоростью, чтобы задержка данных в устройстве, вызванная буферизацией, не превышала нескольких миллисекунд.

Изохронные передачи классифицируются по способу синхронизации конечных точек - источников или получателей данных - с системой: различают асинхронный, синхронный и адаптивный классы устройств, каждому из которых соответствует свой тип канала USB.

5. Протокол

Все обмены (***транзакции***) по USB обычно состоят из трёх пакетов. Каждая транзакция планируется и начинается по инициативе контроллера, который посылает *пакет-маркер* (*Token Packet*). Он описывает тип и направление передачи, адрес устройства USB и номер конечной точки. В каждой транзакции возможен обмен только между адресуемым устройством (его конечной точкой) и хостом. Адресуемое маркером устройство распознает свой адрес и готовится к обмену. Источник данных,

определенный маркером, передает пакет данных (или уведомление об отсутствии данных, предназначенных для передачи). После успешного приема пакета приемник данных посылает *пакет подтверждения* (*Handshake Packet*).

Планирование транзакций обеспечивает управление поточными каналами. На аппаратном уровне использование отказа от транзакции (Nack) при недопустимой интенсивности передачи предохраняет буферы от переполнения сверху и снизу. Маркеры отвергнутых транзакций повторно передаются в свободное для шины время. Управление потоками позволяет гибко планировать обслуживание одновременных разнородных потоков данных.

Устойчивость к ошибкам обеспечивают следующие свойства USB:

1) Высокое качество сигналов, достигаемое благодаря дифференциальным приемникам/передатчикам и экранированным кабелям.

Защита полей управления и данных CRC-кодами.

2) Обнаружение подключения и отключения устройств и конфигурирование ресурсов на системном уровне.

3) Самовосстановление протокола с тайм-аутом при потере пакетов.

4) Управление потоком для обеспечения изохронности и управления аппаратными буферами.

5) Независимость функций от неудачных обменов с другими функциями.

Для обнаружения ошибок передачи каждый пакет имеет контрольные поля CRC-кодов, позволяющие обнаруживать все одиночные и двойные битовые ошибки. Аппаратные средства обнаруживают ошибки передачи, а контроллер автоматически производит трехкратную попытку передачи. Если повторы безуспешны, сообщение об ошибке передается клиентскому ПО.

6. Форматы пакетов

Байты передаются по шине последовательно, начиная с младшего бита. Все посылки организованы в пакеты. Каждый пакет начинается с поля синхронизации ***Sync***, которое представляется последовательностью состояний **KJKJKJKK** (кодированную по NRZI), следующую после состояния ***Idle***. Последние два бита (**КК**) являются маркером начала пакета **SOP**, используемым для идентификации первого бита идентификатора пакета PID.

Идентификатор пакета является 4-битным полем PID[3:0], идентифицирующим тип пакета (таблица 1), за которым в качестве контрольных следуют те же 4 бита, но инвертированные.

Таблица 1.

Тип PID	***Имя PID***	***PID[3:0]***	***Содержимое и назначение***
Token	OUT	0001	Адрес функции и номер конечной точки - маркер транзакции функции
Token	IN	1001	Адрес функции и номер конечной точки - маркер транзакции хоста
Token	SOF	1001	Маркер начала кадра

Таблица 1(продолжение)

Тип PID	*Имя PID*	*PID[3:0]*	*Содержимое и назначение*
Token	SETUP	1101	Адрес функции и номер конечной точки - маркер транзакции с управляющей точкой
Data	Data0 Datal	0011 1011	Пакеты данных с четным и нечетным PID чередуются для точной идентификации подтверждений
Handshake	Аск	0010	Подтверждение безошибочного приема пакета
Handshake	NAK	1010	Приемник не сумел принять или передатчик не сумел передать данные. Может использоваться для управления потоком данных (неготовность). В транзакциях прерываний является признаком отсутствия не обслуженных прерываний
Handshake	STALL	1110	Конечная точка требует вмешательства хоста
Special	PRE	1100	Преамбула передачи на низкой скорости
	SS		Расщепленный запуск
	CS		Расщепленное завершение

6.1 Формат пакетов-маркеров **IN, OUT, SETUP**.

SYNC[8]	PID[4]	Check[4]	**Func[7]**	**EndPoint[4]**	CRC[5]	EOP[2]

- PID может принимать значения: 1001, 0001, 1101.
- **Func [7]**- содержит адрес функции (номер физического устройства , который может быть в диапазоне от 0-127)
- **Endpoint[4]**- адрес конечной точки (в диапазоне от 0-15)
- CRC[5] – циклический контрольный код.

6.2 Формат пакета-маркера SOF.

SYNC[8]	PID[4]	Check[4]	**Frame[11]**	CRC[5]	EOP[2]

Пакет ***SOF*** используется для отметки начала кадра. На шинах LS/FS период формирования кадров равен 1 мс. На шине HS формируются микро-кадры с периодом – 125 мкс. Соотношение кадров и микро-кадров отражено на рис. 7.

И хотя хост контроллер оперирует 32-битным счетчиком кадров, в маркере SOF передается только младшие 11 бит, остальная часть номера кадра содержится в регистре ввода-вывода хост-контроллера (***FRNUM***).

- Frame[11] - содержит номер кадра, который последовательно (циклически) увеличивается для очередного кадра.
- CRC[5] – циклический контрольный код, вычисленный по полю Frame.

Каждый кадр заканчивается временным **n**-битовым интервалом ***EOF*** (End Of Frame), что показано на рис. 8.

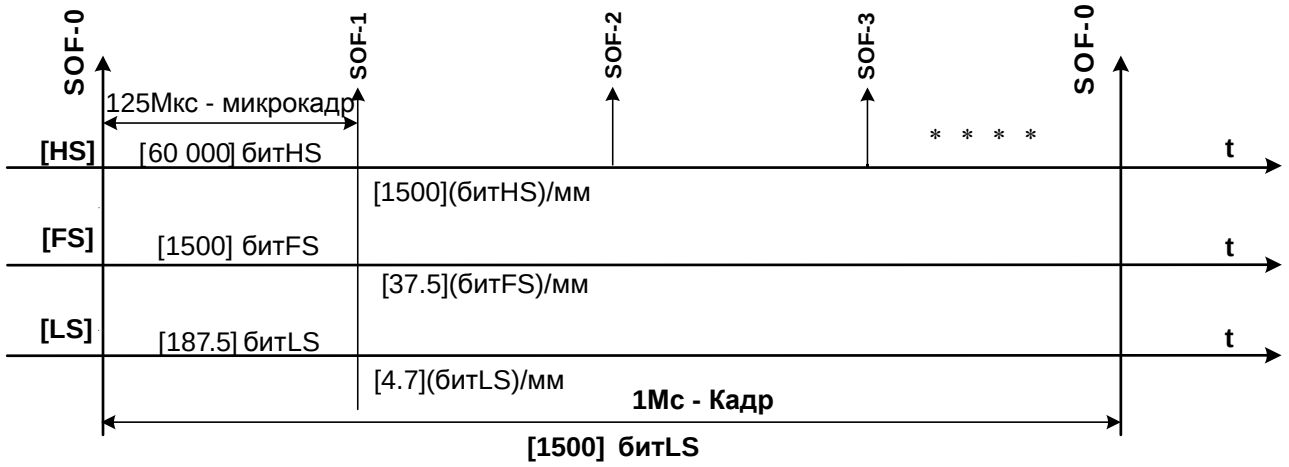


Рис. 7. Шкалы времени для шин LS/FS/HS

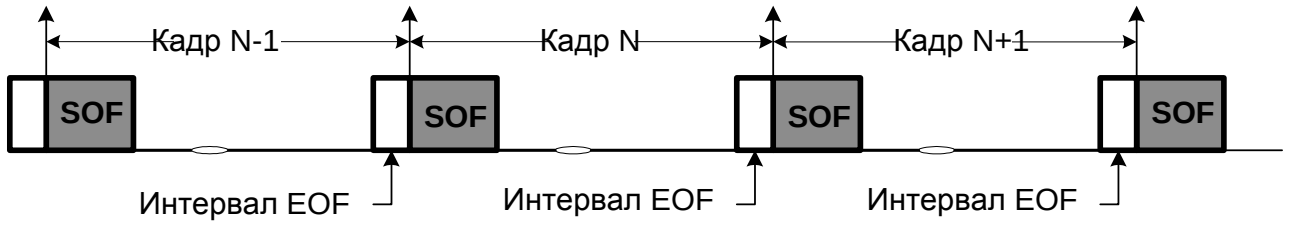


Рис. 8. - Поток кадров USB

6.3 Формат пакета данных Data0, Data1, Data2, MData .

SYNC[8]	PID[4]	Check[4]	**Data[0..8192]**	CRC[16]	EOP[2]

- Data [0..8192]- содержит данные размером от 0 до 1023 байт.
- CRC[16] - циклический контрольный код, вычисленный по полю Data.
- PID[4] – кодирует 4 вида пакетов данных, которые используются для подстройки фазы (***Data0***, ***Data1***); для указания неразрывности при передаче частями (транзакциями) большого буфера (***MData***); для подтверждения завершения некоторых транзакций (***Data2***).

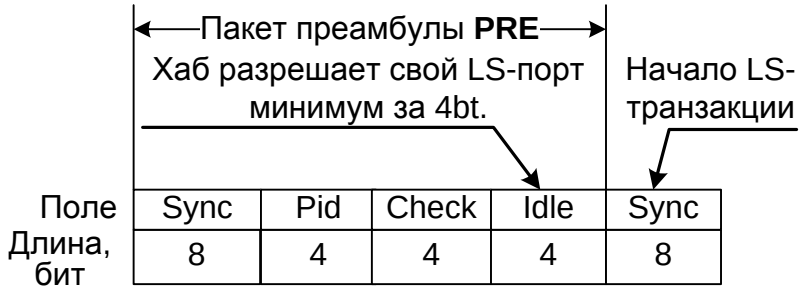
6.4 Формат пакетов подтверждения **Ack, Nak, Stall, Nyet**

SYNC[8]	PID[4]	Check[4]	EOP[2]

Поле данных пакета подтверждения пустое.

6.5 Формат пакета-преамбулы PRE.

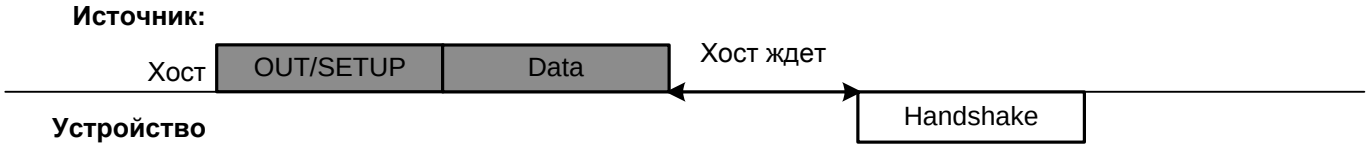
Для направления трафика с шины HS на шину LS (а не FS) используется специальный пакет-преамбула ***PRE***, формат которого имеет следующий вид.



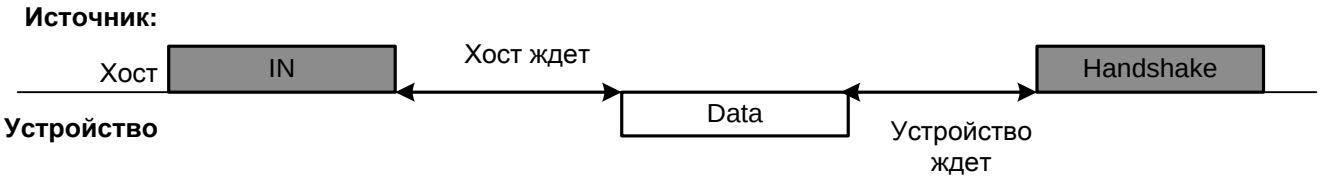
7. Протоколы транзакций для различных типов передач.

Все обмены (транзакции) по USB состоят из трех пакетов (иногда из двух). Хост-контроллер, инициирующий обмен, посылает маркер-пакет то есть пакет типа ***token***. Он описывает тип и направление передачи, адрес устройства USB и номер конечной точки. В каждой транзакции возможен обмен только между устройством и хостом. Адресуемое маркером устройство распознает свой адрес и готовится к обмену. Источник данных, определенный маркером, передает пакет данных или уведомление об отсутствии данных для передачи. После успешного приема пакета приемник данных посылает пакет подтверждения (Handshake).

Типовые последовательности пакетов в транзакциях приведены на рис. 9.



а)



б)

Рис.9. Последовательность пакетов в транзакциях на шине USB:
а - вывод; б - ввод данных

В спецификации USB определены следующие типы транзакций:

- ***изохронная передача данных*** (рис.10)**:**

1) хост посылает маркер ***Out***, содержащий номер функции и номер конечной точки, для которой предназначены данные;

2) хост посылает выбранной конечной точке пакет данных со сброшенным битом синхронизации (то есть пакет типа ***Data0***).

Изохронная передача данных заключается в выполнении транзакции

передачи данных без подтверждения от конечной точки к хосту.

Рис.10. Изохронная передача данных

- ***изохронный прием данных*** (рис.11)**:**

1) хост посылает маркер ***In***, содержащий номер функции и номер конечной точки, от которой запрашиваются данные;

2) выбранная конечная точка передает хосту пакет данных со сброшенным битом синхронизации (то есть пакет типа ***Data0***).

Изохронный прием данных заключается в выполнении транзакции приема пакета данных без подтверждения от хоста к конечной точке.

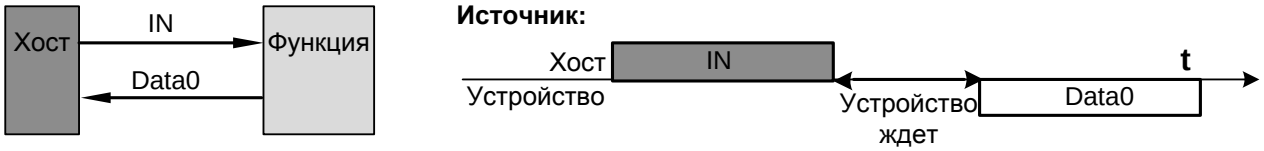


Рис.11. Изохронный приём данных

Для изохронной передачи важна синхронизация устройств и контроллера. Есть три варианта:

- синхронизация внутреннего генератора устройства с маркерами SOF;
- подстройка частоты кадров под частоту устройства;
- согласование скорости передачи (приема) устройства с частотой кадров.

Подстройка частоты кадров контроллера возможна, естественно, под частоту внутренней синхронизации только одного устройства. Подстройка осуществляется через механизм обратной связи, который позволяет изменять период кадра в пределах ±1 битового интервала.

Состояние триггера данных при изохронной передаче игнорируется, но рекомендуется сбросить его в ноль перед началом передачи. Изохронную передачу могут выполнять только полно-скоростные устройства. Максимальный размер пакет данных при изохронной передаче -1023 байта.

- ***передача команды:*** (рис.9-а)**:**

1) хост посылает маркер ***Setup***, содержащий номер функции и номер конечной точки, для которой предназначена команда;

2) хост посылает выбранной конечной точке пакет данных со сброшенным битом синхронизации (то есть пакет типа ***Data0***), содержащий 8-байтовый код команды;

3) функция посылает хосту пакет подтверждения.

- ***передача массивов данных*** (рис. 12)**:**

1) хост посылает маркер ***Out***, содержащий номер функции и номер конечной точки, для которой предназначены данные;

2) хост посылает выбранной конечной точке пакет данных ***Data0*** (***Data1***);

3) функция посылает хосту пакет подтверждения.

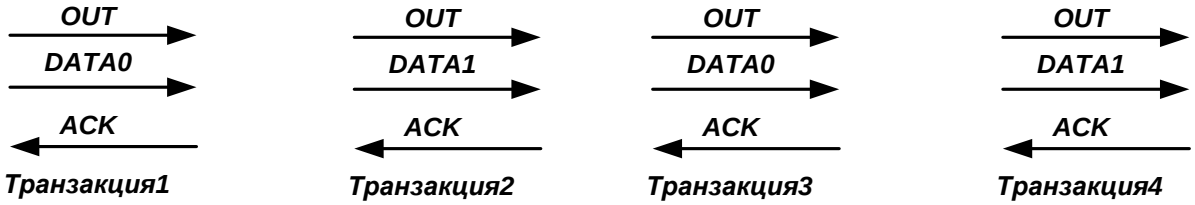


Рис 12. Передача массива данных от хоста

- ***прием массивов данных*** (рис. 13)**:**

1) хост посылает маркер ***In***, содержащий номер функции и номер конечной точки, от которой запрашиваются данные;

2) выбранная конечная точка передает хосту пакет данных ***Data0*** или пакет подтверждений ***Nak***-данные не готовы;

3) если хост получил пакет данных, он посылает пакет подтверждения ***Ack***.

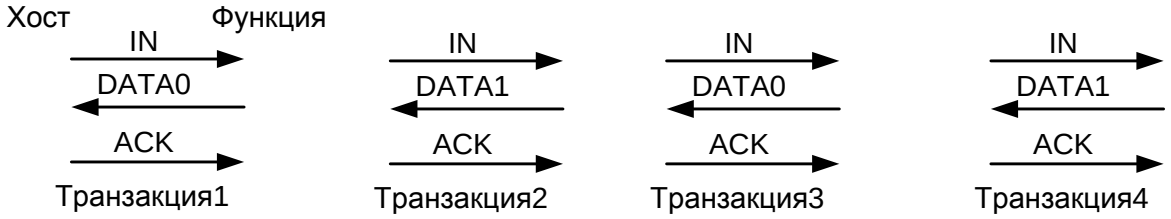


Рис 13. Прием массива данных хостом.

При *приеме* или *передаче* каждого блока данных происходит переключение триггера данных. Первый передаваемый (или принимаемый) блок имеет тип ***Data0***, следующий ***Data1*** и так далее. Максимальный размер пакета при передаче по прерыванию для низкоскоростных устройств не может быть более 8 байт, а для высокоскоростных более 64 байт.

- ***Передача по прерыванию*****:** (рис. 14)

Передача по прерыванию заключается в выполнении транзакции передачи пакета данных с подтверждением от конечной точки к хосту.

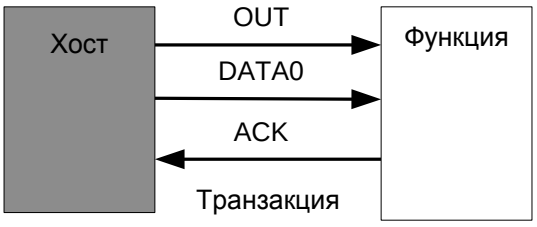


Рис 14. Передача по прерыванию.

- ***Прием данных по прерыванию*:** (рис.15)

Прием данных по прерыванию заключается в выполнении транзакции приема данных с подтверждением.

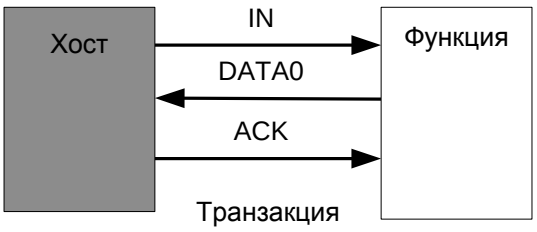


Рис.15. Прием данных по прерыванию.

При приеме или передаче каждого блока данных происходит переключение триггера данных. Первый передаваемый (или принимаемый) блок имеет тип ***Data0***, следующий ***Data1*** и так далее. Максимальный размер пакета при передачи по прерыванию для низкоскоростных устройств не может быть более 8 байт, а для высокоскоростных более 64 байт.

8. Совместная работа устройств с разными скоростями

Спецификация USB 1.x описывает две скорости работы устройств (LS - 1.5 Мбит/с и FS -12 Мбит/с), спецификация USB 2.0 предусматривает повышение пропускной способности до 480 Мбит/с, а USB 3.0 до 5 Гбит/с.

Спецификация USB гарантирует работу USB-устройств с разной скоростью на одной шине. Если бы низкоскоростные устройства работали «напрямую», то хост (или хаб) тратил бы значительную часть времени на обмен с этими устройствами, точнее на ожидание маркера SOF от таких устройств.

Для решения проблем совместимости введены два ключевых решения:

- в состав хаба (USB 2.0) добавлен *транслятор транзакций (**TT**),* который буферизирует поступающий с медленного порта кадр, а затем на максимальной скорости передаёт его хосту, или же буферизирует получаемый на максимальной скорости кадр от хоста, передавая его затем устройству на меньшей, приемлемой для него скорости;
- хаб USB не транслирует трафик на свои нисходящие порты, к которым подключены низкоскоростные устройства, до тех пор, пока хост-контроллер не передаст специального маркера-преамбулы низкоскоростного обмена (***PRE***).

Очевидно, что низкоскоростные транзакции расходуют время кадра весьма неэффективно, но в USB 1x с этим мирятся ради возможности подключения дешевых устройств и упрощения хабов, которые являются просто повторителями сигналов. Заметим, что маркеры ***SOF*** не транслируются на низкоскоростные порты, так что изохронный обмен, для которого они необходимы, для LS-устройств невозможен и не поддерживается.

Хост-контроллер USB 2.0 фактически содержит в себе два контроллера: EHC для работы с HS-устройствами и контроллер UHC (или OHC) для работы с

LS-устройствами и FS-устройствами. Корневой хаб имеет равноправные порты, но в процессе конфигурирования в зависимости от свойств подключённого к нему устройства (или хаба) каждый порт соединяется с соответствующим контроллером.

Хост-контроллер USB 3.0 включает в себя аддитивно контроллер EHC 2.0 (LS/FS/HS) и контроллер USB 3.0 (SS-Super Speed), причём USB 2.0-устройства можно подключать к хабу USB 3.0 и, наоборот, USB 3.0-устройства можно подключать к хабу USB 2.0.

8.1 Структура хаба USB 2.0 [Hub-ступица колеса, втулка]

Хаб является ключевым элементом системы PnP в архитектуре USB. Хаб является кабельным концентратором, точки подключения называются портами хаба. Каждый хаб преобразует одну точку подключения в их множество. Структура хаба USB 2.0 приведена на рис.16.

Хаб состоит из набора портов, контроллера хаба (устройство-функция USB, подключенная к внутреннему порту), повторителя, транслятора транзакций, маршрутизирующей логики портов и цепей управления подачей питания. Хаб USB 1.x проще: в нем отсутствует транслятор транзакций и логика маршрутизации нисходящих портов - они все подключаются к повторителю.

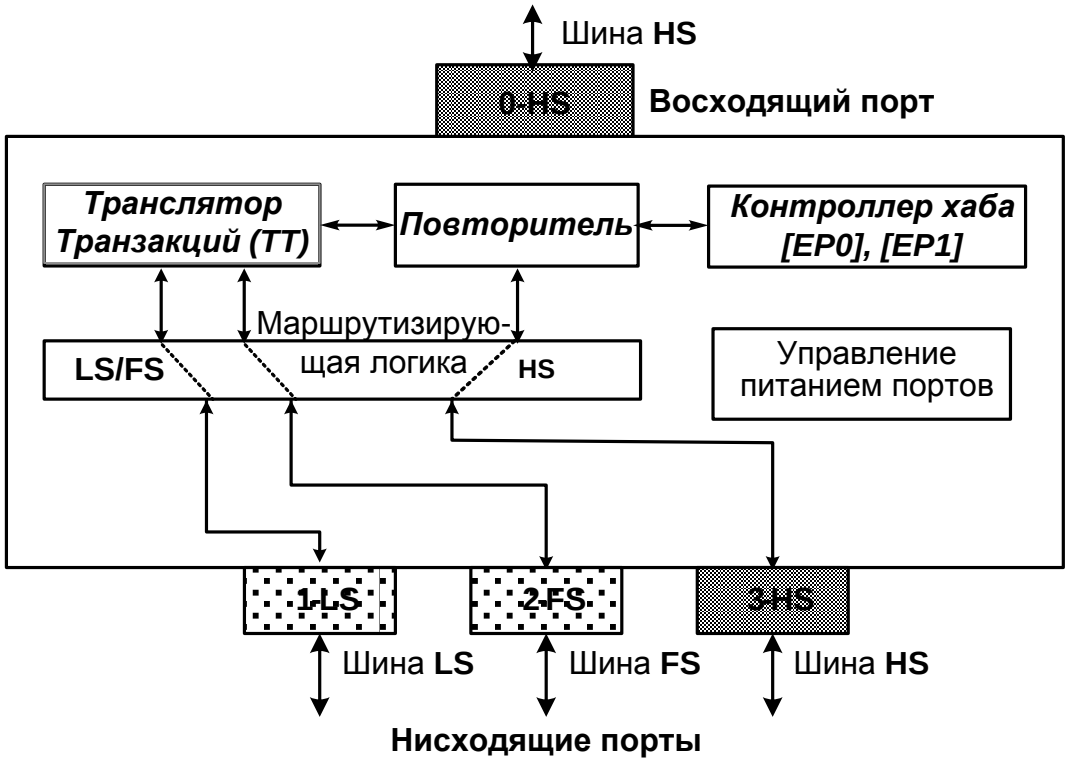


Рис. 16. Структура хаба USB 2.0

8.2 Порты хаба

- Со стороны *восходящего порта хаб* выглядит как любое устройство USB. Этот порт *всегда включен и разрешен.* Для хабов USB 1.x он всегда полно-скоростной, для хабов USB 2.0 восходящий порт всегда высокоскоростной, хотя может работать и в полно-скоростном режиме.
- *Нисходящие порты* хаба имеют комплект приемопередатчиков.

Хост управляет нисходящими портами и определяет их состояние, посылая запросы к контроллеру хаба. Каждый из этих портов может определить, подключено ли к нему устройство и на какой скорости оно работает.

8.3 Транслятор транзакций TT

Спецификация USB гарантирует работу USB-устройства с разной скоростью на одной шине. Для решения проблем совместимости введены два ключевых решения:

- (USB 2.0) в состав кадра добавлен *транслятор транзакций TT,* который буферизует поступающий с медленного порта кадр, а затем на максимальной скорости передаёт его хосту, или же буферизует получаемый на максимальной скорости кадр от хоста, передавая его затем устройству на меньшей, приемлемой для него скорости;
- хаб USB не транслирует трафик на свои нисходящие порты, к которым подключены низкоскоростные устройства, до тех пор, пока HC не передаст специального маркера-преамбулы низкоскоростного обмена (PRE).

Маркер ***PRE*** игнорируется всеми устройствами, кроме хабов. Пакетом преамбулой HC гарантирует, что следующий пакет будет им передан на низкой скорости. Этим пакетом будет маркер, определяющий тип транзакции с LS-устройством, а в транзакции вывода - и пакет данных.

Хаб разрешает транслировать на свой нисходящий порт с LS-устройством только один пакет, следующий за преамбулой; по концу пакета (увидев EOP на низкой скорости) он снова запрещает трансляцию. Чтобы хаб успел переключить режим своего приёмопередатчика, между преамбулой и последующим пакетом вводится зазор (4 битовых интервала FS). Для ответа LS-устройства никаких преамбул не нужно - хабы способны прозрачно передавать восходящий трафик на обеих скоростях (LS и FS). HC, естественно, должен принимать пакеты и на LS и на FS. *Очевидно, что LS - транзакции расходуют время кадра весьма неэффективно.*

Транслятор выполняет расщепленные транзакции ввода/вывода и транслирует маркеры микро-кадров в маркеры кадров, передаваемые в порты FS.

Расщепление транзакций организуется хостом, который знает текущую топологию шины (к портам каких хабов USB 2.0 подключены устройства или хабы 1.x). Расщепленные транзакции выполняются в два-три этапа, в зависимости от типа и направления передачи:

1) между хостом и транслятором выполняется специальная транзакция ***Start Split*** (***SS***-расщепленный запуск). Она несет всю информацию, необходимую для запуска транзакции с целевым устройством. На этом этапе транслятор играет роль специфически адресуемого устройства USB;

2) между транслятором и целевым устройством (хабом) USB 1.0 выполняется обычная транзакция, в которой транслятор играет роль хост-контроллера;

3) между хостом и транслятором выполняется специальная транзакция ***Complete Split*** (***CS***-расщепленное завершение), которая доносит хосту результаты выполнения транзакции хаба с целевым устройством. Здесь транслятор также играет роль специфически адресуемого устройства USB. Для изохронного вывода этот этап отсутствует, поскольку никакого ответа от целевого устройства не предусмотрено.

Во всех этих транзакциях используется обычная «молчаливая» реакция на прием поврежденных пакетов и механизм тайм-аутов.

Каждый нулевой маркер микро-кадра хаб транслирует в виде полно-скоростного маркера кадра SOF. Расщепленные транзакции на стороне FS/LS выполняются внутри этих кадров. Старт расщепленных транзакций хост планирует на нулевой микро-кадр, чтобы к концу последнего (седьмого) микро-кадра расщепленная транзакция могла быть завершена и все данные оказались бы переданы (чтобы не накапливать переходящих остатков). Временные соотношения между транзакциями на обеих сторонах транслятора иллюстрирует рис.17, где в качестве примера рассмотрено расщепление транзакции ввода.

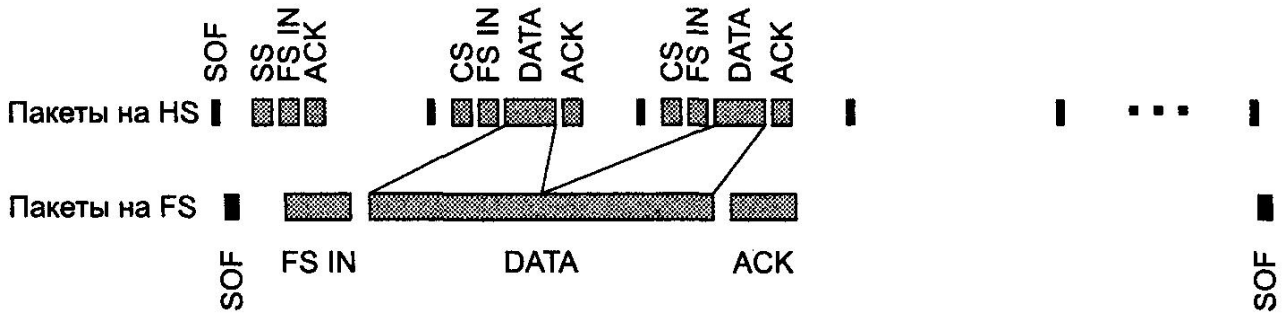


Рис. 17. Трансляция транзакций

Общая структура транслятора транзакций приведена на рис.18. HS-обработчик (HS Handler) помещает информацию расщепленных запусков в свои буферы, а по запросам завершений выбирает из буферов результаты и передает их в виде пакетов хосту. Этот обработчик отрабатывает все обычные функции протокола USB, включая генерацию и проверку CRC, посылку хосту подтверждений и т. п.

F/LS-обработчик (F/LS-Handler) по выбранным из буферов запросам запусков формирует обычные транзакции USB, начинающиеся с маркеров IN, OUT, SETUP (а для LS-портов и с преамбулы PRE). Результаты этих

транзакций (данные и подтверждения) он помещает в буферы.

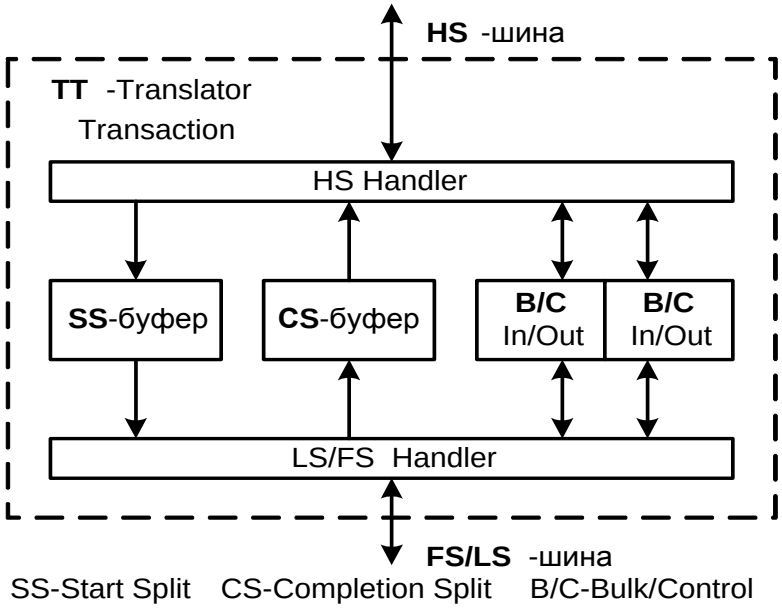


Рис.18. Структура транслятора транзакций

Транслятор транзакций обладает буферами, в которых размещаются все необходимые данные о текущих выполняемых расщепленных транзакциях. По завершении транзакции (на обеих сторонах) буфер освобождается для обслуживания следующих расщепленных транзакций.

Периодические транзакции (изохронные и прерывания), критичные к времени выполнения, транслятором обрабатываются по конвейерной схеме. Данные запусков периодических транзакций помещаются в ***SS-буфер***, из которого они выбираются F/LS-обработчиком (буфер реализует дисциплину FIFO) и запускаются в виде транзакций на вторичную шину. Результаты этих транзакций помещаются в ***CS-буфер*** (тоже FIFO), откуда их извлекает хост. За порядок наполнения и опустошения этих буферов отвечает хост - он планирует транзакции *SS* и *CS.*

Транзакции запусков проходят без подтверждений со стороны транслятора; хост узнает о судьбе транзакции только в фазе завершения.

Периодические транзакции (изохронные и прерывания) критичны к времени выполнения, что накладывает отпечаток на их исполнение в расщепленном виде.

Непериодические транзакции (управление и передача массивов) обрабатываются иначе: у транслятора имеется *два или более буфера* ***B/C-In/Out,*** каждый из которых обслуживает по одной транзакции, от начала и до конца. Здесь транзакции запусков подтверждаются транслятором: ответ *NAK* означает невозможность приема транзакции к исполнению (нет свободных буферов), то есть хосту следует повторить попытку запуска. Ответ *ACK* означает, что транзакция принята к исполнению и через некоторое время следует забрать

результат, выполнив транзакцию завершения.

Спецификация предусматривает два варианта реализации транслятора; какой именно применяется, можно узнать из кода протокола интерфейса хаба:

- по одному транслятору на каждый нисходящий порт (код протокола - 2); это самый производительный вариант для умножения пропускной способности шины для устройств FS/LS;
- один транслятор на все порты (код протокола - 1); здесь возможности умножения пропускной способности зависят от объема буферов периодических транзакций и количества буферов для непериодических.

В расщепленных транзакциях фаза маркера, определяющая продолжение транзакции, состоит из двух пакетов-маркеров: специального маркера *SPLIT,* формат которого приведен на рис. 19, за которым следует маркер обычной транзакции *(IN, OUT, SETUP). Пакет-преамбула PRE в расщепленных транзакциях не используется* - скорость целевого устройства задается в маркере *SPLIT*, что позволяет хабу провести транзакцию на нужной скорости. Транслятор сам генерирует преамбулу, если вторичный порт, через который обращаются к LS-устройству, опознал FS-подключение (это означает, что между хабом, транслирующим транзакцию, и целевым устройством есть хаб USB 1.x).

Маркер *SPLIT* адресуется к порту хаба (номера в полях HubAddr и PortAddr), а следующий за ним обычный маркер адресует конечную точку целевого устройства. Маркеры *SPLIT,* используемые для запуска *(SS)* и завершения *(CS)* расщепленных транзакций, различаются полем **SC**: 0 - для *SS,* 1 - для *CS.*

Поле ET описывает тип целевой конечной точки, с которой будет производиться транзакция (00 - *Control,* 01 - *Isochronous,* 10 - *Bulk,* 11 - *Interrupt).*

В зависимости от типа транзакции поля **S** и **E** трактуются по-разному:

- Для *управляющих* транзакций и *прерываний* поле S определяет скорость (0 - FS, 1 - LS), E = 0.
- Для остальных транзакций, кроме запуска изохронного вывода, поля S и E содержат нули.
- В транзакциях запуска *изохронного вывода* поля S и E трактуются как:

10 -стартовый пакет, 01 - последний пакет, 00 - промежуточный, 11 - в пакете все данные транзакции.

Поле	Sync	Pid	Check	**Hub Addr**	SC	**Port Addr**	S	E	ET	CRC	EOP
Длина, бит	32	4	4	7	1	7	1	1	2	5	8

Рис.19. Формат маркеров SPLIT (SS при SC=0, CS при SC=1)

Непериодические транзакции (***управления и передачи массивов***) не критичны к времени выполнения; малый размер блока данных (до 64 байт)

позволяет их расщеплять, используя по одной транзакции запуска и завершения.

9. Трассировка и распределение полосы частот шины

USBD должен определять, может ли шина поддерживать требуемый канал, перед тем как устанавливать связь с конечной точкой устройства (EP). С этой целью он обращается к дескриптору EP, в одном из полей которого содержится максимальное значение буфера данных для этой точки. Этот буфер содержит только полезные данные (payload). Для передачи этих данных по шине требуются накладные расходы (overhead), которые займут часть полосы пропускания шины, соответственно USBD должен учесть и это для определения полного времени шины, требуемого для поддержания этой EP. Как только полное время передачи будет вычислено, USBD проверяет расписание движения кадров, для того, чтобы убедиться, что новый канал передачи может быть поддержан.

Назовём основные параметры, которые должны быть известны, для того, чтобы вычислить требования к полосе шины для данного канала.

1) Количество байт данных (number of data bytes).

Эта информация считывается из поля ***MaxPacketSize*** в дескрипторе конечной точки EP.

2) Тип передачи (transfer type)

Isochronous и ***interrupt*** передачи требуют гарантированной полосы шины, в то время как ***control*** передачи должны иметь обязательные 10% полосы. Передачи массивов (***bulk***) не гарантированы во время любого кадра. Они осуществляются в свободные промежутки времени, т.е. имеют самый низкий приоритет.

Тип передачи (соответственно и тип транзакции) определяет количество накладных расходов (***overhead***). Например, изохронные передачи не сопровождаются пакетами квитирования.

В общем накладные расходы включают:

- *token packet* - sync bits (8) + PID (8) + Address (11) + CRC (5) +EOP (2) = 34 bits
- *data packet overhead* - sync bits (8) + PID (8) + CRC (16) + EOP (2) = 34 bits
- *handshake* (если есть) - sync bits (8) + PID (8) + EOP (2) = 18 bits

Тип передачи закодирован в поле атрибутов в дескрипторе EP.

3) Время восстановления хоста (host recovery time).

Это время, необходимое ***HC*** для завершения последней передачи и подготовки для следующей.

4) Установка хаба в режим LS (hub low-speed setup).

Когда целью транзакции является низкоскоростное устройство, то для направления трафика с шины HS на шину LS (а не на FS) требуется пакет преамбулы PRE, который содержит дополнительную задержку.

5) Время на вставку служебных бит **(**bit stuffing time**).**

Поскольку вставка бит является функцией потока данных, который неизвестен ПО хоста заранее, то используют худший случай теоретического максимума необходимого количества дополнительных битовых интервалов, которые должны быть учтены в ***overhead.***

bst=1.1667*8*number of bytes

6) Глубина топологии (depth in topology)

Время, требуемое для посылки пакетов к устройству и время, требуемое для получения ответного пакета зависит от количества кабельных сегментов, которые посылка должна пройти.

Для расчёта полосы пропускания следует пользоваться схемами расчёта задержек, приведёнными в приложении 2.

10. Расчёт полосы пропускания для клавиатуры USB

10.1 Пример схемы подключения клавиатуры-USB

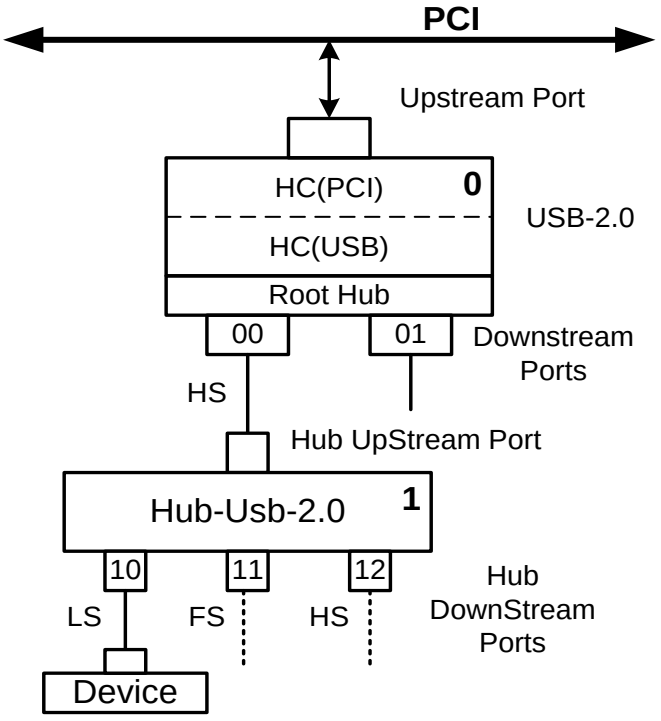


Рис.20. Схема подключения клавиатуры.

Общий вид схемы подключения клавиатуры-USB приведён на рис.20.

Структура хаба USB 2.0 (рис. 16) в свой состав включает транслятор транзакций TT (рис. 18), содержащий буферы, посредством которых осуществляется согласование скоростей шин HS и LS/FS. HS-обработчик (**HS Handler**) помещает информацию расщепленных запусков в свои буферы, а по запросам завершений выбирает из буферов результаты и передает их в виде пакетов хосту. Этот обработчик отрабатывает все обычные функции протокола USB, включая генерацию и проверку CRC, посылку хосту подтверждений и т. п.

F/LS-обработчик (**F/LS-Handler**) по выбранным из буферов запросам запусков формирует обычные транзакции USB, начинающиеся с маркеров IN, OUT, SETUP (а для LS-портов и с преамбулы **PRE**). Результаты этих транзакций (данные и подтверждения) он помещает в буферы.

Расщепленные транзакции выполняются в два-три этапа, в зависимости от типа и направления передачи:

1. между **хостом и транслятором** выполняется специальная транзакция *Start Split (SS* - расщепленный запуск). Она несет всю информацию, необходимую для запуска транзакции с целевым устройством. На этом этапе транслятор играет роль специфически адресуемого устройства USB;

2. между **транслятором и целевым устройством (или хабом) USB 1.0** выполняется обычная транзакция, в которой транслятор играет роль хост-контроллера;

3. между **хостом и транслятором** выполняется специальная транзакция *Complete Split* (*CS* - расщепленное завершение), которая доносит хосту результаты выполнения транзакции хаба с целевым устройством. Здесь транслятор также играет роль специфически адресуемого устройства USB. Для изохронного вывода этот этап отсутствует, поскольку никакого ответа от целевого устройства не предусмотрено

Транслятор выполняет расщепленные транзакции ввода/вывода и транслирует маркеры микро-кадров в маркеры кадров, передаваемые в порты FS.

Расщепление транзакций организуется хостом, который знает текущую топологию шины (к портам каких хабов USB 2.0 подключены устройства или хабы 1.x).

Для адресации буферов транслятора транзакций используется маркер-пакет **Split**, который определяет адрес буфера трансляции через адреса хаба и порта, скорость обмена с буфером со стороны шины LS/FS, тип конечной точки и ряд другой информации.

Формат маркер-пакета **Split** показан на рис. 19.

Для направления трафика на шину LS (а не FS) используется специальный пакет-преамбула ***PRE*** (6.5).

Транзакция расщеплённого запуска ***ввода по прерыванию*** *(Interrupt-IN)* состоит из последовательности маркеров ***SS, PRE*** и ***IN***. Данные от устройства хост получит в пакетах 1-2 транзакций завершения. Максимальный пакет данных по прерыванию на шине LS равен 8 байт.

Транзакция завершения состоит из последовательности маркеров ***CS*** и ***IN***, на которую транслятор отвечает пакетом данных или подтверждения *(NAK, NYET, STALL* или *ERR* с вышеописанными значениями). Если транслятор принял еще не все данные от конечной точки целевого устройства, они придут в пакете ***MDATA*** (если принято менее трех байт, транслятор пошлет *NYET),* на что хост должен повторить транзакцию завершения в следующем микро-кадре. Последняя порция данных придет в пакете ***DATA0*** или ***DATA1***.

10.2 Расчёт транзакций для клавиатуры

Исходные данные:

1) Схема подключения клавиатуры имеет вид (рис.20)
2) Передача данных от клавиатуры характеризуется следующими параметрами:

- Тип передачи - ***Interrupt***,
- Направление передачи - ***In***,
- Шина подключения - ***LS***,
- HID-устройство. ***Payload=8byte***

3) Время обслуживания, $t_{обсл}$***=10мс*** из диапазона (10-255)мс для HID-устройств на шине LS.

4) Среднее время задержки, $t_{зад}$***=32bt*** (bt=бит-тайм)

Решение:

1 Определим размер одной транзакции на шине LS для клавиатуры (1Tp/LS)

Общий вид временной диаграммы транзакции прерываний для клавиатуры на шине LS определяется в соответствии с приложением 1 (рис.32-а, б). В соответствии с этой схемой транзакция прерываний состоит из двух пакетов, причём устройство в ответ на пакет ***In*** может отвечать пакетами: ***Data0, Data1, MData, Nak, Nyet, Stall, Error***. В этой связи временная диаграмма транзакции прерываний на шине LS выглядит как на рис.21.

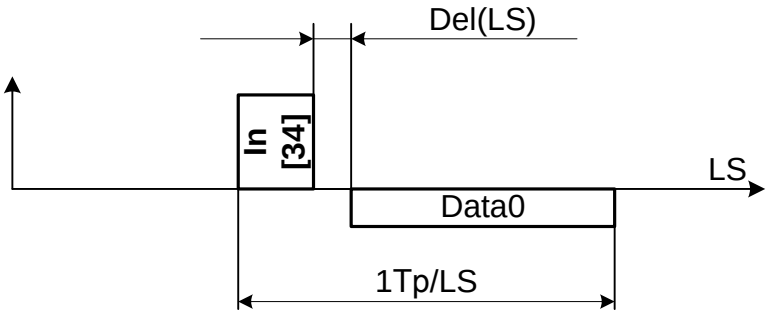
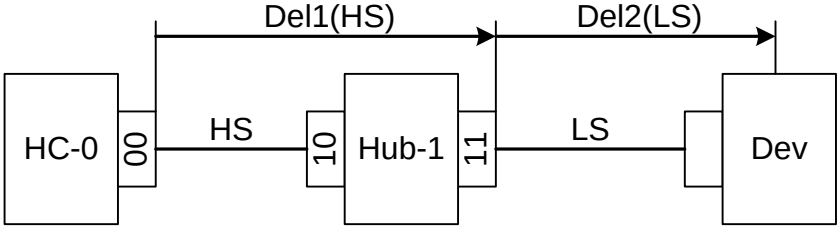


Рис. 21. Транзакция прерываний на шине LS

1.1 Размер пакета ***In=34bt***

1.2 Размер пакета *Data0* с учётом механизма вставки бит (bitstaffing)=1.1667*[64]Payload+[34]OverheadData=[74.7]+[34]=108.7bt. ***Data0=108.7bt***

1.3 Для расчёта задержек на шине LS построим схему распространения трафика (рис.22) в соответствии со схемой подключения клавиатуры (рис. 20).



Del1(HS) - задержка на 1 кабельном сегменте [HS] вместе с одним хабом [HS].
Del2(LS) - задержка на кабельном сегменте [LS] и устройстве в одном направлении. **bt** - (бит-тайм)

Рис.22. Схема для расчёта задержек

Значения задержки Del(LS)=Del1(HS)+Del2(LS)=***16bt/LS*** приняты по рекомендации приложения 2, причём Del1(HS)<<Del2(LS).

На основании вышеизложенного размер одной транзакции на шине LS составит:

$$1Tp(LS) = [34]+[108.7]+[16] = 158.7bt$$

2 *Построение временной диаграммы расщеплённой транзакции для клавиатуры.*

Удобно планировать транзакции запуска (SS) и транзакции завершения (CS), привязываясь к отметкам времени микрокадров.

За время микрокадра (125мкс) по шине LS можно передать 187.5bt(LS). Это означает, что 1Tp(LS) может быть транслирована на шину HS за время одного микрокадра т.к. 187.5bt>158.7bt. (Одной транзакции SS будет соответствовать одна транзакция CS)

Временная диаграмма работы канала клавиатуры теперь может быть построена. Она имеет вид как на рис. 23.

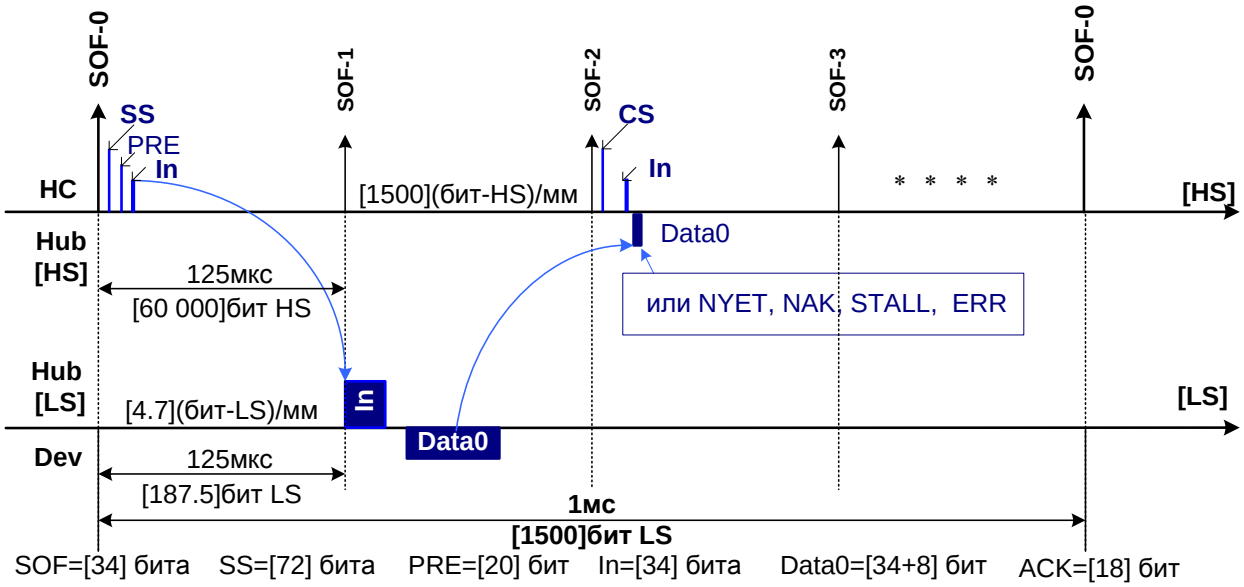


Рис.23. Ввод по прерыванию от клавиатуры

3 *Определение частоты запуска транзакций клавиатуры.*

Поскольку время обслуживания, $t_{обсл}$***=10мс***, то запуск SS-транзакций и соответствующих им CS-транзакций следует производить через каждые 10 кадров. В этом случае будет обеспечена скорость передачи клавиатуры до ***V***=8байт/10мс=***800байт/сек.***

Упражнения:

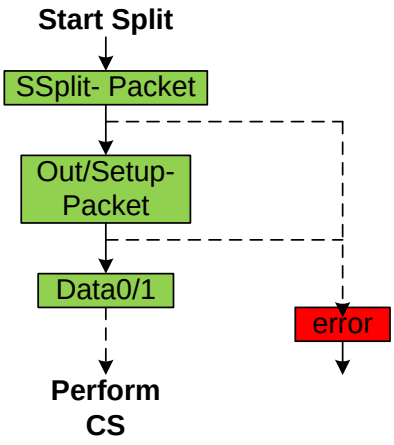
1. Определить протокол Split-транзакции изохронного вывода для устройства, подключённого к полноскоростному порту корневого хаба.
2. Определить протокол Split-транзакции изохронного ввода для устройства, подключённого к полноскоростному порту корневого хаба.
3. Определить протокол Split-транзакции вывода по прерыванию для устройства, подключённого к полноскоростному порту корневого хаба.
4. Определить протокол Split-транзакции ввода по прерыванию для устройства, подключённого к полноскоростному порту корневого хаба.
5. Построить временную диаграмму транзакции ввода данных с клавиатуры, если устройство подключено к низкоскоростному порту (LS) хаба USB 2.0. Оценить (подсчитать) загруженность шины HS при организации данного канала.
6. Решить пример расчёта транзакций для принтера, если
   1) Принтер подключён к полноскоростному порту корневого хаба.
   2) Передача данных от принтера характеризуется следующими параметрами:
      - Тип передачи - ***Bulk***,
      - Направление передачи - ***Out***,
      - Шина подключения - ***FS***,
      - HID-устройство. ***Payload=64byte***
   3) Время обслуживания: свободное
   4) Среднее время задержки, $t_{зад}$=15bt (bt=бит-тайм)
7. Построить временную диаграмму транзакций ввода/вывода данных в НЖД, если устройство подключено к высокоскоростному (HS) порту хаба USB 2.0. Исходные данные определены следующими параметрами:

1) Скорость передачи данных устройств массовой памяти НЖД, CD, DVD и т.д. принять равной (Vdev = **25-500**)Мб/с
2) «Накладные расходы», (Overhead) с учётом задержек равны **55** байтам для обмена массивами данных на шине **HS**.
3) Максимальный размер пакета Payload при передаче массивов для HS-устройств не более **512** байт (1-512).

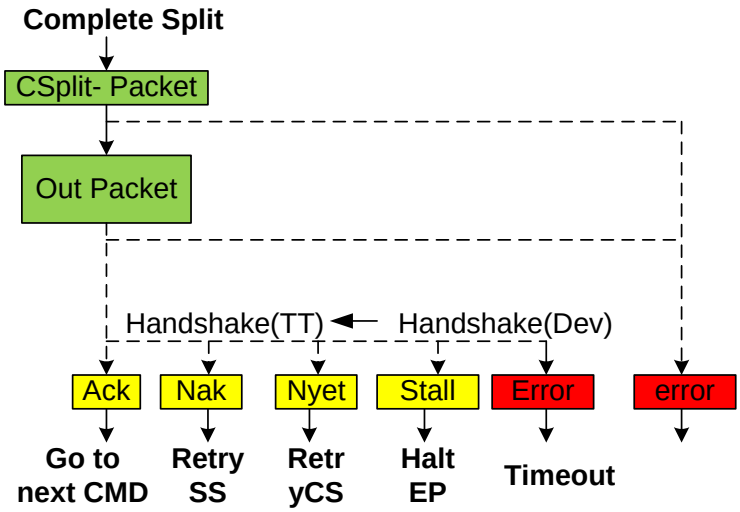
Приложения

1. **Split-последовательности для различных типов передач.**

1.1 Interrupt-OUT- (SS-CS) Sequence

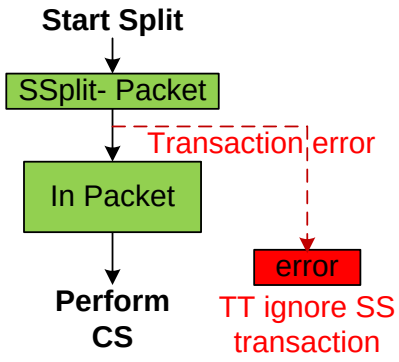


Последовательность запуска транзакции вывода по прерыванию.

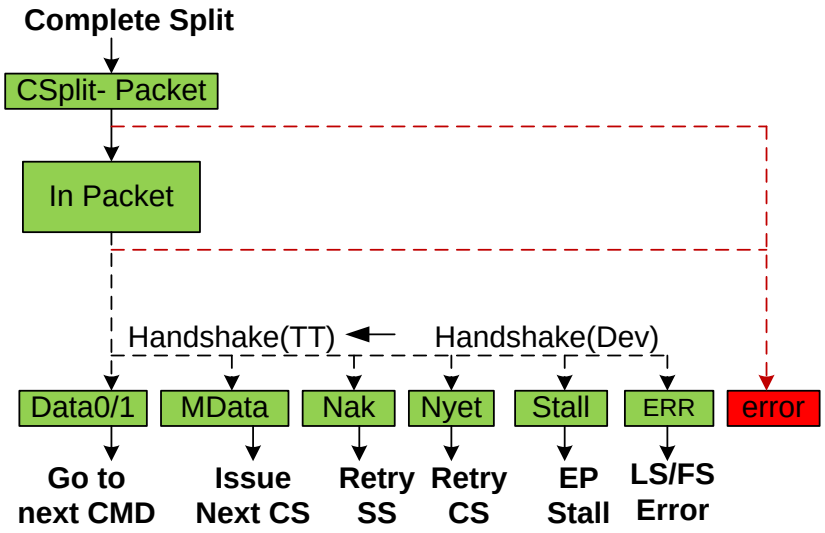


Последовательность завершения транзакции вывода по прерыванию.

1.2 Interrupt-IN- (SS-CS) Sequence



Последовательность запуска транзакции ввода.



Последовательность завершения транзакции ввода.

1.3 Bulk-Control OUT (SS-CS) Sequence

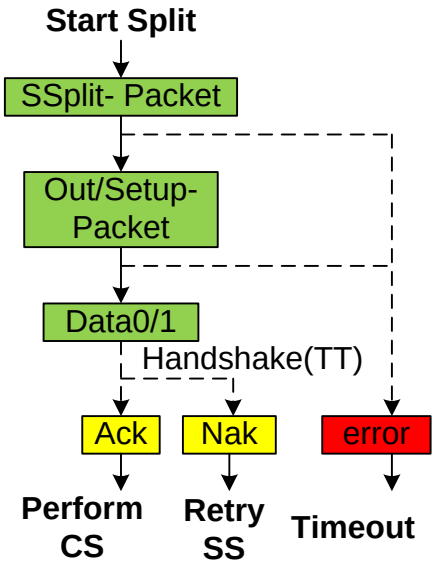


Рис. XX. Последовательность запуска транзакции вывода.

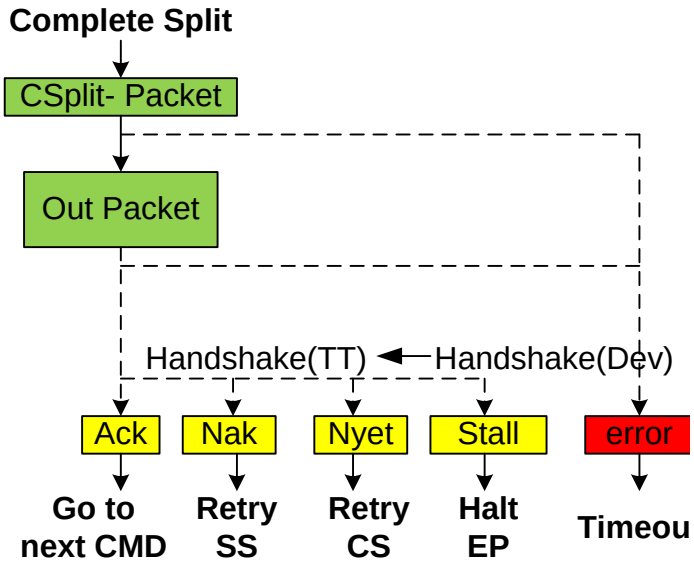
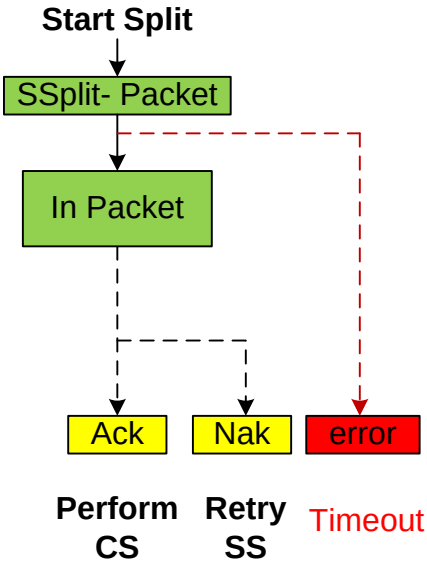
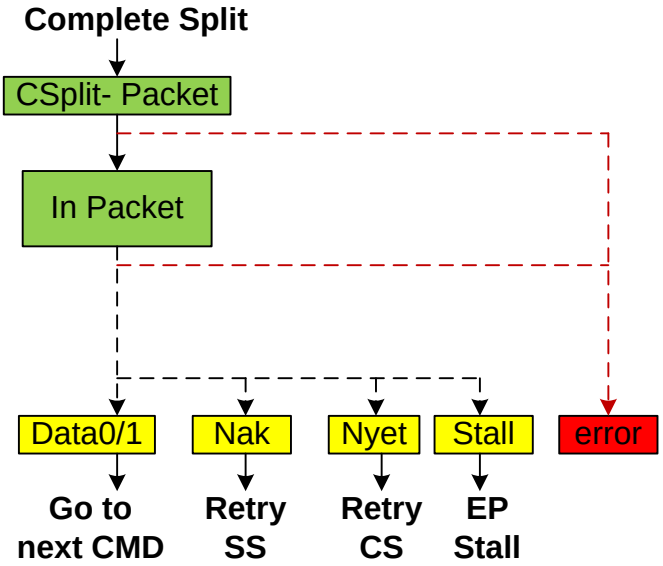


Рис. XX. Последовательность завершения транзакции вывода.

1.4 Bulk-Control IN (SS-CS) Sequence

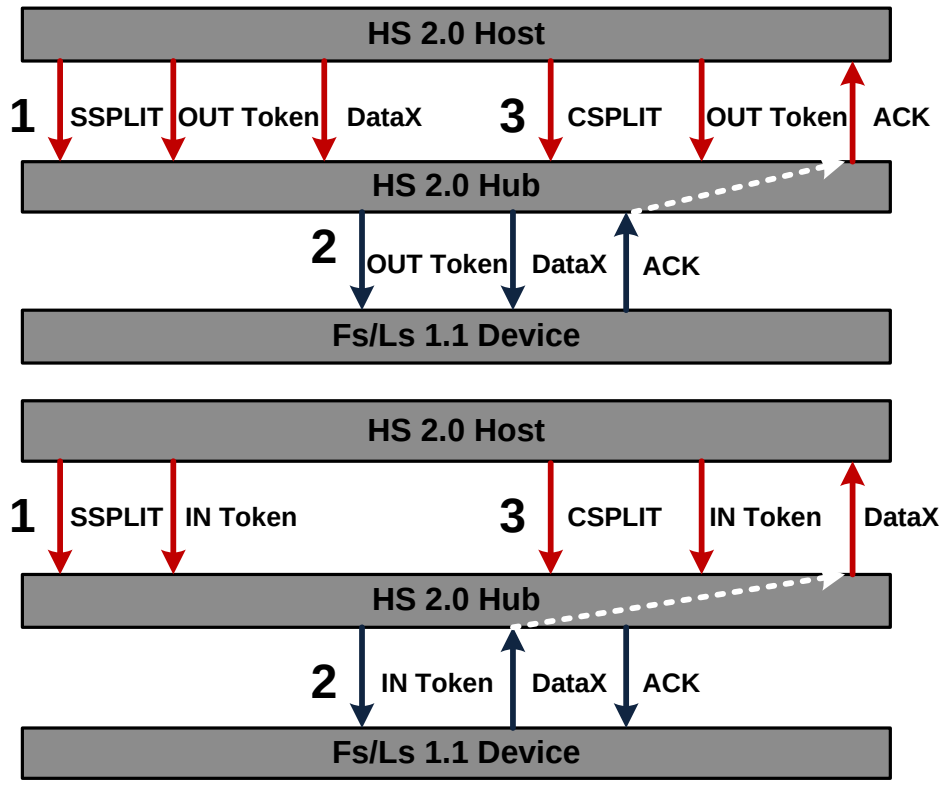


Последовательность запуска транзакции ввода.



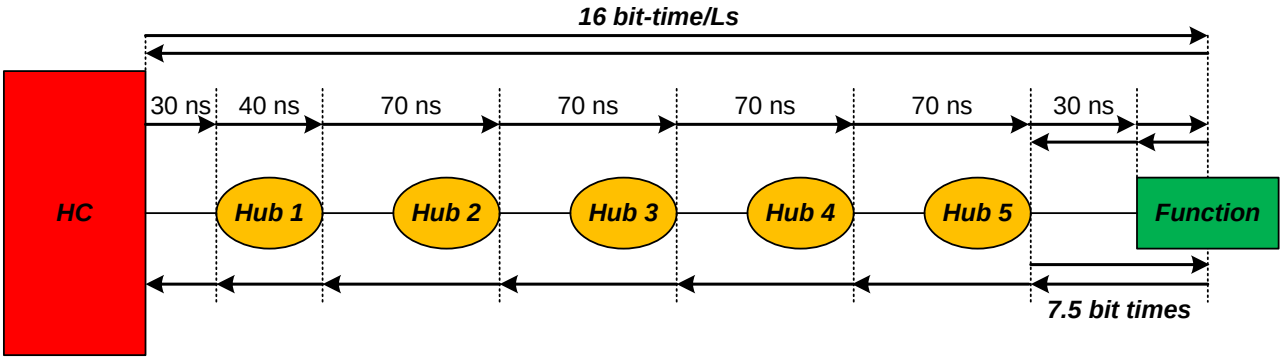
Последовательность завершения транзакции ввода.

1.5 OUT/IN - SS/CS sequences с переменным блоком передаваемых данных.

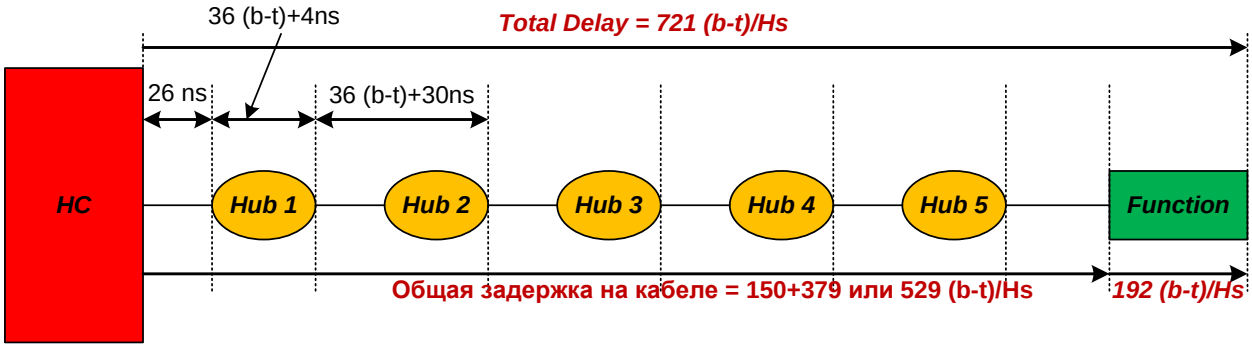


2. Схемы для учёта задержек при подключении устройств к шине USB 2.0.

2.1 Наихудший случай временных задержек между ***HC*** и ***Function*** в одном направлении для ***FS/LS***



2.2 Наихудший случай временных задержек между ***HC*** и ***Function*** в одном направлении для ***HS***



3. **Скорость передачи данных некоторых устройств**

1.	**LS:** Interactive Devices **(10-100)Кб/с**	Клавиатура, мышь, перо и т. д.
2.	**FS:** Звуковые и аудио устройства **(500-10000)Кб/с**	Принтер, сканер, цифровые аудио и т. д.
3.	**HS:**Video, Disk, Lan **(25-500)Мб/с**.	Массовая память: НЖД, CD,DVD и т.д.

Список литературы

1. Don Anderson USB System Architecture (USB 2.0), тех.док, 1999, электр
2. Агуров П.В. Интерфейс USB. Практика использования и программирования.- СПб.: БХВ-Петербург, 2004г

Содержание

Содержание