Программисту USB-устройств Часть 1. Знакомство с USB

Дмитрий Чекунов (г. Ижевск)

Интерфейс USB, несмотря на свою «молодость», уже прочно обосновался в компьютерной технике и стремительно заменяет все «классические» интерфейсы. Если несколько лет назад разрабатывать устройство с поддержкой USB считалось модным, то теперь это становится необходимостью. Только шина USB может обеспечить высокую скорость передачи данных, «горячее» подключение устройств и бесконфликтное распределение ресурсов системы. Цикл статей предназначен для программистов, приступающих к разработке «низкоуровневого» ПО USB-устройств. Предлагаемый материал даёт возможность самостоятельно спроектировать функциональную модель устройства, составить его описание и разработать ПО, обеспечивающее корректное подключение к шине. Материал будет также полезен разработчикам драйверов USB-устройств. В этой части статьи рассмотрены организация шины и устройств USB, логика работы хоста при обнаружении на шине нового устройства и возможные типы передачи данных.

Архитектура шины USB

Топология шины USB очень похожа на дерево (рис. 1). В его корне находится ведущее устройство – хост, который осуществляет управление шиной. В задачи хоста входит: обнаружение подключения/отключения устройства, управление потоками данных, контроль статуса устройства, учёт статистики, распределение внутреннего питания между подключёнными устройствами. На шине всегда существует только один хост, поэтому направление передачи данных принято определять с его позиции. Если передача осуществляется от хоста к устройству, то поток имеет направление OUT и называется downstream. При передаче от устройства к хосту поток имеет направление IN и называется upstream.

Устройства, подключаемые к шине, являются подчинёнными и делятся на два вида: хаб и устройство, выполняющее некоторую функцию. Хаб служит разветвителем шины и предоставляет свои порты для подключения других устройств, и хабов в том числе. Хост периодически запрашивает статус хаба и по его изменению определяет подключение нового устройства или отключение работавшего. Хаб, совмещённый с хостом, называется корневым. Функции, выполняемые корневым хабом и обычным, одинаковы.

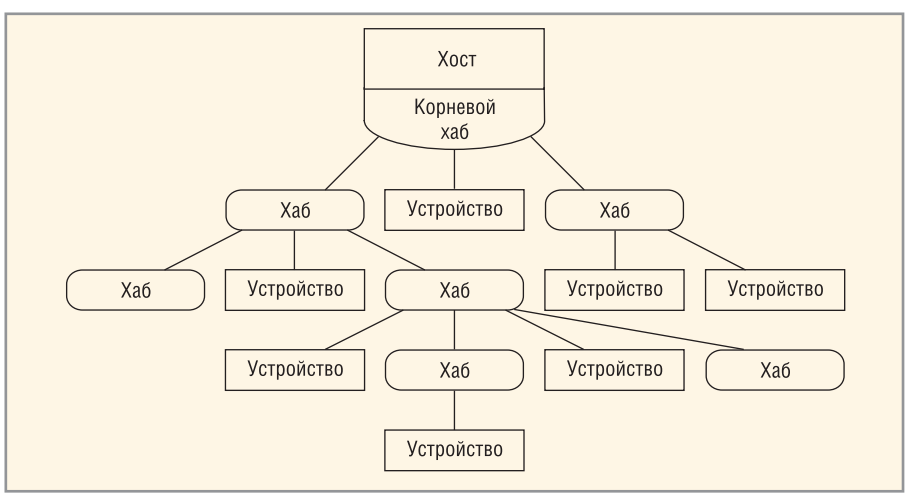

Рис. 1. Архитектура шины USB

Ветви, формируемые хабом, заканчиваются подключённым устройством, выполняющим определённую функцию. Часть портов хаба остаётся свободной, они запрещены до подключения нового устройства и на работу шины не влияют.

Подчинённые устройства не могут самостоятельно, спонтанно послать данные по шине, все операции выполняются ими только по требованию хоста. Однако, если хост перевёл устройство в режим пониженного потребления питания suspend, то при пробуждении от внешнего воздействия устройство сигнализирует об изменении своего статуса, не ожидая приглашения хоста.

Устройства могут использовать собственный источник питания или внутренний источник шины USB. Суммарный ток, потребляемый устройствами от источника шины USB, не должен превышать 1 А. Допускается подключение к шине до 127 подчинённых устройств.

На шине USB доступны три скоростных режима работы устройств: низкоскоростной, полноскоростной и высокоскоростной. В первую очередь скорость работы шины определяет хост, а уже при подключении устройство настраивается на доступную максимальную скорость. Наивысшая скорость передачи достигается при работе устройства на высокоскоростной шине.

Организация USB-устройства

Логическую организацию USB-устройства также удобно представить в виде дерева (рис. 2). Узлы ветвей (cfg, if, alt) обозначают режим работы. Окончание ветви (ep) – определённую функцию.

Выбор режима осуществляется указанием узлов нужной ветви. Установку и смену режима работы устройства выполняет хост. Режим остаётся активным до следующей команды смены режима работы. Функции, доступные в активном режиме, имеют

уникальные адреса и называются точками. Выполнение соответствующей функции происходит при обращении хоста к точке устройства. Режим, в котором отсутствуют какие-либо функции, обычно используется для перевода устройства в нерабочее состояние, например idle.

Проектирование структуры USB-устройства, режимов работы и распределение функций выполняются программистом по собственному усмотрению. Далее, при подробном рассмотрении элементов логической структуры, распределим часть ресурсов устройства, представленного на рис. 2, для некоторого абстрактного программатора микросхем. На рисунке для конфигурации, интерфейса, альтернативной установки и точки сохранены английские обозначения, соответствующие сокращениям имён переменных, используемых при программировании.

Конфигурация (cfg) является первым элементом ветви и определяет глобальные настройки. Устройство может иметь от 1 до 255 конфигураций. Нумерация начинается с 1. Если номер активной конфигурации равен 0, то считается, что устройство не сконфигурировано и не способно выполнять никаких функций.

Например, конфигурация cfg1 (рис. 2) используется для перевода устройства в «спящий» режим, cfg2 – в режим программирования микросхем с последовательным интерфейсом, cfg3 – с параллельным.

Следующий элемент ветви – интерфейс (if). В продолжение конфигурации интерфейс позволяет разделить режимы по некоторым характерным особенностям в работе устройства. Нумерация интерфейсов начинается с 0, количество в одной конфигурации – от 1 до 256.

Конфигурация cfg2 (рис. 2), используемая для программирования последовательных микросхем, содержит N интерфейсов. Выделим интерфейс if0 для работы с микросхемами I2C, интерфейс if1 – для микросхем SPI, интерфейс if2 – для микросхем 1-Wire и т.д.

Альтернативная установка (alt) является последним элементом на ветви режима работы устройства. Он определяет количество доступных точек и специфические особенности при работе с ними. Нумерация альтернативных установок начинается с 0, допустимое количество в одном интерфейсе от 1 до 256.

В структуре устройства, представленного на рис. 2, ветвь cfg2–if0 выделена для работы с микросхемами I²C. Альтернативными установками alt0, alt1, alt2, ..., altN создадим подрежимы для работы с микросхемами соответствующего объема – 128 байт, 256 байт, 512 байт, ... 64 Кбайт.

Функции, выполняемые устройством в текущем режиме, определяются доступными точками. Точка (ep) – это буфер данных, при обслуживании которого устройство выполняет определённую функцию. Альтернативная установка может содержать до 15 точек направления OUT и до 15 точек направления IN. Каждая точка имеет свой адрес, который однозначно характеризует её направление передачи данных. Адреса точек OUT находятся в диапазоне 1...0Fh, это точки, в которые хост передаёт данные. Диапазон 81h...8Fh используется для адресов точек IN, из которых хост принимает данные. Нумерация точек одного направления может быть произвольной и не обязательно последовательной, т.е. следующий набор точек является правильным: 1, 3, 4, 8 (OUT) и 81h, 82h, 86h (IN).

При выборе ветви cfg2–if0–alt0 устройство переходит в режим работы с микросхемой I²C объёмом 128 байт. Для того чтобы запрограммировать микросхему, хост должен передать данные в точку ep1. При появлении данных в буфере точки устройство выполнит программу физической записи в микросхему I²C. Для чтения данных хосту достаточно обратиться к точке ep81, после чего устройство выполнит физическое чтение из микросхемы и разместит данные в буфер точки IN.

В структуре USB-устройства особняком стоит точка ep0 – это контрольная точка. Она является двунаправленной и доступна всегда, независимо от текущего режима работы устройства. Через контрольную точку осуществляется идентификация и конфигурирование устройства.

Подключение и работа устройства

Как уже было сказано ранее, подключение устройства к шине USB происходит через порт хаба. Хост, периодически запрашивая статус хаба,

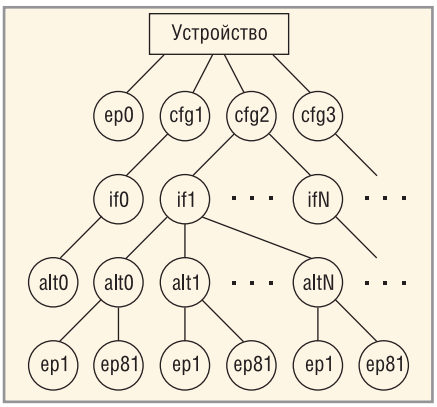

Рис. 2. Структура USB-устройства

узнаёт о подключении нового устройства и разрешает соответствующий порт. В этот момент устройство не имеет адреса и не сконфигурировано, поэтому хост обращается к нему по адресу 0 через контрольную точку (ep0). Первой командой (SET_ADDRESS) хост присваивает устройству уникальный адрес, с которым оно работает до момента отключения. Далее, используя команду GET_DESCRIPTOR, хост считывает описание устройства и описания всех конфигураций. Устройство всегда имеет как минимум одну конфигурацию. Хост командой SET_CONFIGURATION устанавливает первую доступную конфигурацию, не анализируя её назначения. После этого устройство считается сконфигурированным и готовым к работе. Полученная информация позволяет операционной системе идентифицировать устройство и загрузить соответствующий драйвер. Дальнейшее управление устройством возлагается на драйвер.

Вернемся к рис. 2. После подключения устройство находится в конфигурации cfg1, и его полный рабочий режим лежит на ветви cfg1–if0–alt0. Для того чтобы пользователь смог запрограммировать микросхему с интерфейсом SPI объёмом 512 байт, необходимо выбрать новый режим, описываемый ветвью cfg2–if1–alt2. Драйвер командой SET_CONFIGURATION с параметром 2 устанавливает конфигурацию для работы с последовательными микросхемами – cfg2, при этом активизируется продолжение ветви, определённое программистом по умолчанию, обычно if0–alt0. Следующая команда SET_INTERFACE с параметрами 1 и 2 устанавливает необходимый режим работы – if1–alt2. В новом состоянии устрой-

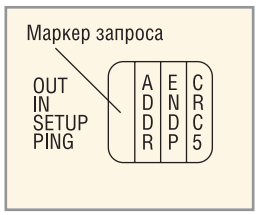

Рис. 3. Пакет запроса

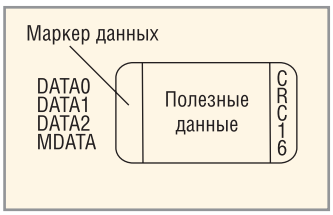

Рис. 4. Пакет данных

Рис. 5. Маркер подтверждения

ство имеет две (не считая контрольной) активные точки – ep1, ep81. Программирование микросхемы осуществляется передачей данных в точку ep1, чтение из микросхемы – запросом к точке ep81. Сменить обслуживаемый тип микросхем можно в любой момент. В случае если новая микросхема относится к семейству последовательных, драйверу достаточно выполнить команду SET_INTERFACE с соответствующими параметрами.

Для возврата устройства в режим idle необходимо выполнить команду SET_CONFIGURATION с параметром 1.

Структура информационного потока

На шине USB используется пакетная передача информации. Для обмена одним пакетом данных хост и точка устройства выполняют цикл, представляющий последовательность «запрос–данные–подтверждение». Служебная информация, сопровождающая полезные данные, однозначно определяет адресата, целостность данных и готовность точки к следующему циклу.

Всю информацию, передаваемую по шине USB, можно разделить на следующие типы:

- пакеты запроса;
- пакеты данных;
- маркеры подтверждения;
- прочие пакеты.

Пакет запроса – это пакет служебной информации. Хост посылает запрос перед обменом данными или для проверки готовности точки. Пакет запроса (рис. 3) состоит из идентификатора пакета (маркер запроса), адреса устройства ADDR, адреса точки ENDP и контрольной суммы CRC5. Запросы, доступные хосту, имеют следующее назначение:

- OUT – хост начинает передачу данных точке ENDP устройства ADDR;
- IN – хост ожидает данные из точки ENDP устройства ADDR;
- SETUP – хост начинает контрольную передачу для точки ENDP устройства ADDR;
- PING – хост проверяет готовность точки ENDP устройства ADDR. Данный запрос доступен на высокоскоростной шине для передачи BULK и обращён к точкам с направлением OUT.

Пакет данных всегда передаётся вслед за запросом. В состав пакета (рис. 4) входят идентификатор (маркер данных), полезные данные и контрольная сумма CRC16. На размер пакета данных накладывают ограничения тип передачи данных и режим работы шины USB. Существуют следующие маркеры данных:

- DATA0 – чётный пакет данных;
- DATA1 – нечётный пакет данных;
- DATA2, MDATA – дополнительные маркеры, используемые при изохронном обмене на высокоскоростной шине.

Маркеры данных позволяют не только идентифицировать пакет, но ещё и контролировать целостность потока за счёт их определённой последовательности.

Маркеры подтверждения (рис. 5) предназначены для сообщения о результатах получения данных и состоянии точки устройства. Маркеры несут следующую информацию:

- ACK – данные получены без ошибок и будут обработаны;
- NAK – для точки OUT – данные получены без ошибок, но нет возможности их обработать, и поэтому требуется повторная передача данных. Для точки IN – данные не готовы, хост может повторить запрос позднее;
- STALL – точка находится в состоянии HALT и не может выполнять свои функции без вмешательства хоста. Хост не должен повторять запрос;
- NYET – данные получены без ошибок и будут обработаны. Следующий пакет точка принять не готова. Данный маркер имеет место на высокоскоростной шине для передачи BULK и используется точками OUT.

К прочим пакетам относятся SOF, PRE, ERR, SPLIT. Для программиста представляет интерес пакет SOF. Данный пакет используется для синхронизации и передается хостом с определённым интервалом времени.

Типы передачи данных

На шине USB существует четыре типа передачи данных. Они отличаются передаваемым объёмом данных, приоритетом доставки и системой контроля и устранения ошибок.

Наиболее часто используемый тип – BULK. Для подобной передачи гарантирована доставка данных без ошибок, при этом время доставки не гарантировано и зависит от загруженности шины. Контроль данных осуществляется на уровне пакета – суммой CRC16 – и на уровне потока, где чётный и нечётный пакет имеют соответствующие маркеры – DATA0, DATA1. В случае обнаружения ошибки приёмная сторона не возвращает маркер подтверждения, тогда на передающей стороне запускается механизм автоматического повтора передачи.

Размер пакета данных может быть произвольным, в том числе и нулевым, но не должен превышать максимальное допустимое значение. Для высокоскоростной шины USB максимальное значение составляет 512 байт, для полноскоростной – 8, 16, 32 или 64 байта.

Типичная передача для полноскоростной шины показана на рис. 6. Хост в течение двух циклов передаёт данные устройству и получает подтверждение. В третьем цикле, после получения данных, устройство сообщает о невозможности обработать данные. В этом примере показано слабое место полноскоростной шины, где третий пакет данных потерян и требуется его

Рис. 6. BULK передача данных

повторная передача, что увеличивает загрузку шины. На высокоскоростной шине данный недостаток устранён с помощью запроса PING и подтверждения NYET. Для высокоскоростной шины после приёма второго пакета точка ответит подтверждением NYET. Это позволяет хосту приостановить передачу третьего пакета данных и контролировать её готовность коротким запросом PING. Продолжить передачу можно будет после получения подтверждения ACK.

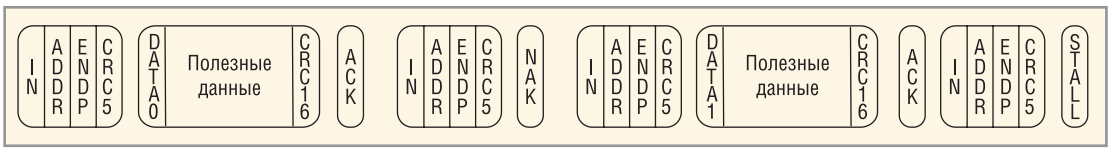

Рис. 7. BULK-приём данных

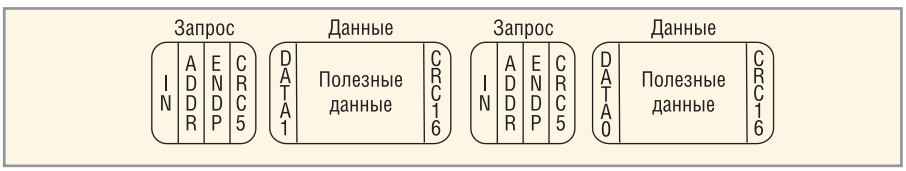

Рис. 8. Изохронный приём данных

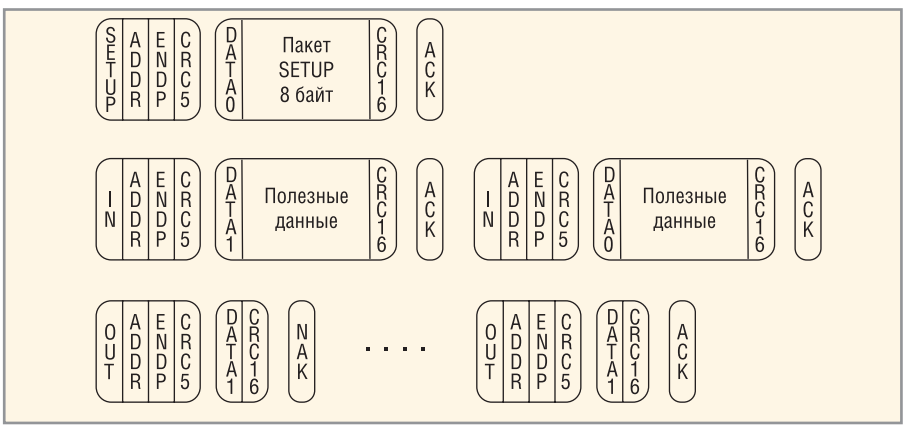

Рис. 9. Формат контрольной транзакции

BULK-приём представлен на рис. 7. Хост посылает запрос IN и подтверждает получение данных маркером ACK. Во втором цикле точка не готова передать данные, поэтому вместо пакета данных она посылает маркер NAK. После получения отказа хост повторяет запрос спустя некоторое время (цикл 3). Последний цикл заканчивается маркером STALL. Данный маркер посылает точка, не способная выполнять по каким-либо причинам свои функции; такое состояние точки называется HALT. В подобной ситуации для возобновления работы точки требуется вмешательство хоста. Точка, находящаяся в состоянии HALT, никак не влияет на работу других точек.

Второй тип передачи – INTERRUPT. Такой тип используется при необходимости обмена данными через заданный временной интервал. Хост гарантирует обмен с заданным интервалом и учитывает это при распределении загрузки шины.

Размер пакета данных для высокоскоростной шины имеет значение от 1 до 1024 байт, а для полноскоростной – от 1 до 64 байт. Интервал опроса точки также зависит от режима работы шины и находится в диапазоне 0,125...4 мс для высокоскоростного режима и 1...255 мс для полноскоростного.

Циклы обмена похожи на BULK-транзакции, представленные на рис. 6 и 7. Отсутствие данных для передачи из точки IN является штатной ситуацией, хост пошлёт следующий запрос по истечении заданного времени.

Следующий тип передачи – ISOCHRONOUS. Передачи такого типа предназначены для быстрой доставки пакетов данных, при этом контроль целостности данных сводится к минимуму и ограничен наличием контрольной суммы CRC16. Повреждённый пакет уничтожается приёмной стороной, передающая сторона о возникновении ошибки не оповещается. Данный тип передачи используется для потоков видео- и аудиоданных.

Размер пакета данных на высокоскоростной шине достигает 1024 байт, на полноскоростной – 1023.

Типичный приём информации изохронного типа показан на рис. 8. Как можно увидеть из рисунка, в циклах обмена отсутствуют маркеры подтверждения.

Последний тип передачи данных – CONTROL. Данный тип передачи используется только при обращении к контрольной точке устройства. Полная транзакция контрольной передачи (рис. 9) состоит из трёх фаз. Первая фаза, показанная в верхней части рисунка, называется SETUP, во время этой фазы хост передаёт пакет данных размером 8 байт. Данный пакет содержит требование, которое должно выполнить устройство. Вторая фаза, изображённая в средней части рисунка, – фаза данных – является необязательной. Она присутствует в случае, когда для выполнения требования необходимы дополнительные данные. Структура потока в фазе данных полностью идентична BULK-транзакции. Последняя фаза называется фазой статуса. Хост, ожидая подтверждения о выполнении требования устройством, посылает запросы. Направление запросов противоположно тем, которые использовались в фазе данных. При передаче запроса OUT хост посылает пакет данных нулевой длины. Пока устройство занято выполнением требования, оно отвечает маркером NAK, после успешного завершения – маркером ACK. Если устройство не способно выполнить требования или не поддерживает его, то в фазе данных или статуса необходимо вернуть маркер STALL.

Размер пакета данных на высокоскоростной шине составляет 64 байта, на полноскоростной – 64, 32, 16 или 8 байт.

Стандартные требования, используемые для управления устройством, передаются с помощью данного типа передачи. Программист при разработке USB-устройства должен обеспечить выполнение всех стандартных требований, а при необходимости может добавить собственные требования.

Литература

1. Universal Serial Bus Specification Revision 2.0. www.usb.org.
2. EZ-USB FX2 Technical Reference Manual. www.cypress.com.