TeraSense USB API 2.0

Версия документа 0.2

Введение

Настоящий документ посвящен описанию архитектуры интерфейса доступа к камерам TeraSense по шине USB версии 2.0. Рассмотрены основные отличия от архитектуры протокола версии 1.0 и преимущества, которые дает новый протокол разработчикам и пользователям.

Основные отличия от протокола версии 1.0

Протокол версии 2.0 является органичным развитием протокола версии 1.0, основанного на использовании FPGA модулей OpalKelly и предоставляемых производителем библиотек. Новый протокол разработан с целью устранить основные недостатки протокола версии 1.0 перечисленные ниже:

1. Зависимость от производителя FPGA модулей и предоставляемых им закрытых библиотек. Новый протокол основан на использовании FPGA модулей ZTEX с собственной прошивкой. Чужой закрытый код не использоваться.
2. Необходимость установки специфических USB драйверов. Новый протокол использует режим эмуляции последовательного порта, поэтому для него не требуется установка драйверов (по крайней мере на windows 10).
3. Сложный низкоуровневый протокол версии 1.0 требует использования громоздких библиотек. В новом протоколе есть возможность использования как базовых низкоуровневых команд, так и макрокоманд (например, старт/стоп), реализующих необходимую последовательность низкоуровневых команд непосредственно в устройстве. В результате для использования протокола 2.0 необходимо минимальное количество кода, который пользователь может написать сам без использования специфических библиотек.
4. Необходимость использования настроек (калибровок), специфических для конкретного устройства. В протоколе версии 2.0 добавлена возможность хранения настроек непосредственно в устройстве, их автоматической загрузки в FPGA и последующего использования.
5. В протоколе версии 1.0 отсутствует возможность потоковой передачи данных (стриминга) по инициативе устройства. Поскольку только хост может выступать инициатором передачи, длительные периоды неактивности клиентского приложения (например, вследствие загруженности хоста) приводят к переполнению буфера кадров устройства и потере данных. Новый протокол использует буферизацию данных последовательного порта, реализованную операционной системой, а также возможность передачи данных по инициативе устройства, что уменьшает вероятность потери данных вследствие неактивности клиентского приложения.

6. Низкая помехозащищенность, свойственная USB устройствам, проявляется в том, что передача данных по протоколу 1.0 может прерваться из-за помех, так что для восстановления работоспособности устройства потребуется его физическое переподключение. Новый протокол поддерживает автоматическое переподключение, после которого устройство продолжит работу без вмешательства пользователя.
7. Протокол версии 1.0 невозможно прозрачно преобразовать в сетевое соединение (например, чтобы иметь возможность управлять устройством на большом удалении от него). Поскольку протокол версии 2.0 использует эмуляцию последовательного порта, преобразование его в сетевое соединение не требует никакой дополнительной логики.

В следующих разделах мы рассмотрим подробнее реализацию описанных выше функциональных особенностей протокола 2.0.

Аппаратная платформа

Модули FPGA модули ZTEX содержат богатый набор аппаратных средств, показанный на следующем рисунке.

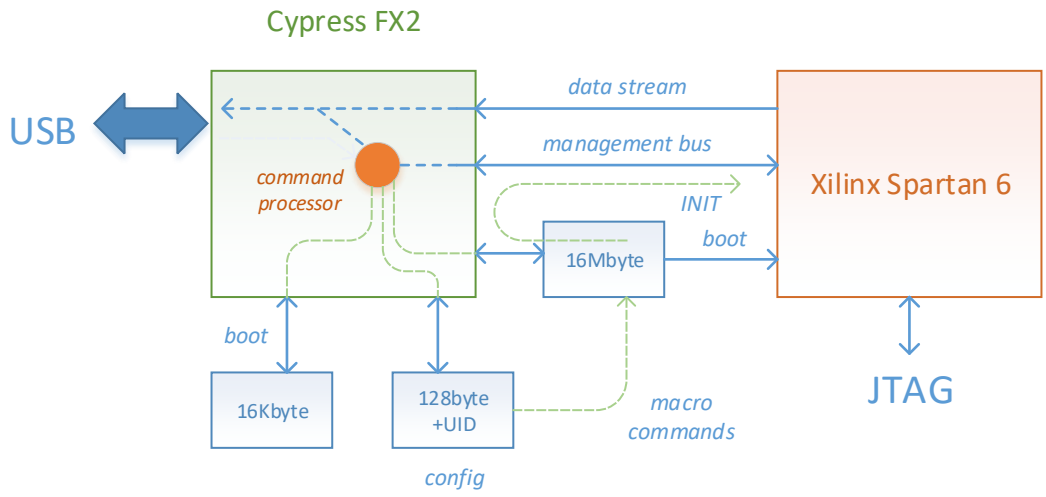

Рисунок 1. Аппаратная платформа ZTEX 2.01.

Центральная роль в реализации протокола отведена USB процессору Cypress FX2LP, который позволяет передавать данные по шине USB на скорости до 480Mbit/сек. К нему подключены 3 микросхемы энергонезависимой памяти. Память емкостью 16Kbyte используется для загрузки программы в FX2 при подключении питания. Флэш-память емкостью 16Мбайт хранит конфигурационный битстрим для FPGA, который автоматически загружается при подключении питания, а также макрокоманды и набор первоначальных настроек для FPGA. Последний также хранится в виде макрокоманды INIT, исполняющейся автоматически при подключении питания. Микросхема памяти емкостью 128 байт хранит идентификатор модели, уникальный идентификатор устройства, а также набор описателей макрокоманд. Сами макрокоманды хранятся в старших адресах 16 мегабайтной флэш-памяти в виде последовательности низкоуровневых команд.

Микросхема FPGA связана с процессором FX2 двумя шинами. Медленная шина управления предназначена для инициализации и проверки статуса, этой шиной управляет FX2. Вторая шина предназначена для быстрой передачи потока данных, ей управляет FPGA. При этом данные сразу передаются в шину USB без всякой обработки со стороны FX2, что позволяет передавать их на максимальной скорости.

Поступающие по USB шине запросы всегда обрабатываются процессором команд. Мы будем называть их управляющими пакетами. Они выполняют функции управления процессором, чтения / записи памяти, а также управления FPGA и чтения его статуса. Управляющие запросы могут активизировать режим потоковой передачи данных от устройства. После этого процессор команд может по-прежнему получать команды, но не может отвечать на них. Выход из режима потоковой передачи происходит при получении команды сброса, которая выключает потоковую передачу и переводит FPGA в состояние, идентичное состоянию после включения питания.

Проблема помехоустойчивости USB

Данные передаются по шине USB в виде пакетов, ограниченных по длине и снабженных контрольной суммой, что позволяет обнаруживать повреждение данных в процессе передачи. В этом эта шина похожа на Ethernet. Теоретически протокол позволяет обрабатывать ошибки передачи данных за счет повторной передачи пакетов, но на практике обнаружение ошибок обычно приводит к разрыву соединения и прекращению передачи данных. Это отчасти связано со сложностью протокола, в котором участвуют и служебные пакеты, потеря которых приводит к непредвиденным последствиям. Разорванное соединение можно переоткрыть заново, но в отличие от Ethernet в USB деление данных на пакеты непрозрачно для приложений, поэтому приложение в общем случае не в состоянии определить, какая часть отправленных данных была получена адресатом, а какая была потеряна. Точно также оно не в состоянии определить, какая часть принятых данных была потеряна. Однако в одном частном случае детектирование потерянных данных становится возможным. Для этого приложение должно само поделить поток данных на пакеты так, чтобы в каждом USB пакете содержалось целое число пакетов уровня приложения. Для этого достаточно выбрать размер пакета так, чтобы он был делителем максимального размера USB пакета[1]. В протоколе версии 2.0 все данные пересылаются пакетами по 64 байта. Каждый пакет снабжен порядковым номером, который позволяет определить, какие пакеты были уже обработаны, а какие – потеряны безвозвратно.

Структура пакетов протокола

В протоколе используются 2 вида пакетов. Управляющие пакеты обрабатываются процессором команд, потоковые пакеты используются для потоковой передачи данных от FPGA к клиенту без всякой обработки со стороны процессора FX2. Клиент различает эти 2 вида пакетов в зависимости от значения старшего бита первого байта – он равен 1 для управляющих пакетов и 0 для потоковых.

---

[1] 64 байта для полной скорости, 512 байт для высокой скорости.

Следующий рисунок иллюстрирует структуру управляющего пакета.

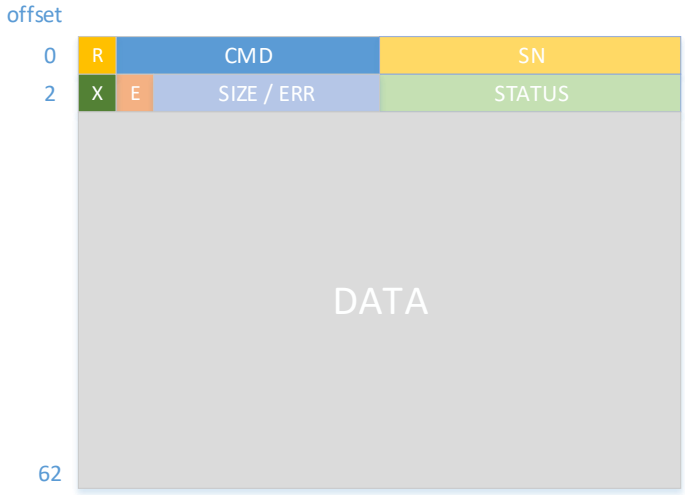

Рисунок 2. Структура управляющего пакета.

Управляющий пакет начинается с 4 байтного заголовка, за которым следует область данных размером 60 байт. Назначение полей заголовка приведено в следующей таблице:

CMD	Код команды.
SN	Порядковый номер. Используется для сопоставления ответов с запросами, а также для детектирования потерянных пакетов.
R	В принятом пакете означает запрос подтверждения. В ответе устройства выставляется всегда.
X	Флаг, включающий транзакционный режим. Команда с таким флагом выполняется строго один раз. Для этого контроллер запоминает порядковый номер SN последней выполненной команды и сравнивает его[2] с номером полученной команды. Если номера при сравнении отличаются более чем на единицу, контроллер переходит в состояние ошибки и перестает выполнять транзакционные команды. Сброс ошибки возможен с помощью отдельной команды.
SIZE / ERR	Размер пересылаемых данных (0..60). Может означать размер запрашиваемых данных (в запросе чтения).
E	Флаг ошибки выполнения команды. Если установлен в ответе, поле SIZE содержит код ошибки.
STATUS	Статус контроллера. Содержит код последней ошибки и флаг успешной инициализации FPGA.

[2] С учетом возможного переполнения.

Пакеты потоковых данных также начинаются с 4 байтного заголовка, в котором вместо единичного бита R всегда передается нулевой бит. Следующий рисунок иллюстрирует структуру потокового пакета.

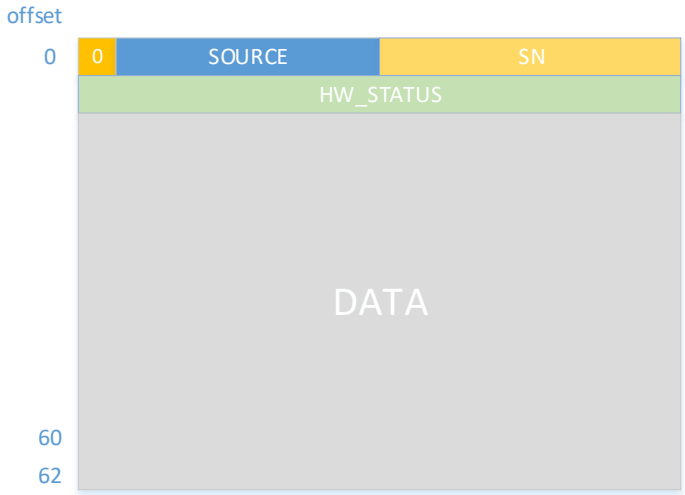

Рисунок 3. Структура потокового пакета.

Порядковый номер SN используется для обнаружения потерянных пакетов. Клиент, обнаружив потерю пакета, посылает контроллеру команду сброса, после чего заново активизирует режим передачи данных. Поле SOURCE может использоваться для маркировки источника данных в случае, если их несколько. В текущей версии это поле содержит нули. Поле HW_STATUS содержит аппаратный статус, специфичный для конкретной реализации, который формируется FPGA.

Размер данных в заголовке отсутствует, так как непрерывный поток данных всегда разбивается на пакеты до полного заполнения очередного пакета.

Структура конфигурационных данных

Конфигурационная память содержит 128 байт доступных для записи и 6 байт с уникальным идентификатором устройства, запрограммированным на заводе изготовителе. Первые 128 байт поделены на 2 равные части. В первой части содержится информация о модели устройства – идентификатор модели, версия и размеры сенсора в пикселах. Далее следует область дополнительных конфигурационных данных XDATA, специфичных для конкретной модели сенсора. В двумерных камерах здесь хранится информация о паттерне пикселей в выходном потоке данных, поскольку в общем случае пиксели читаются не построчно, а поблочно. Для линейного сенсора эта область не используется.

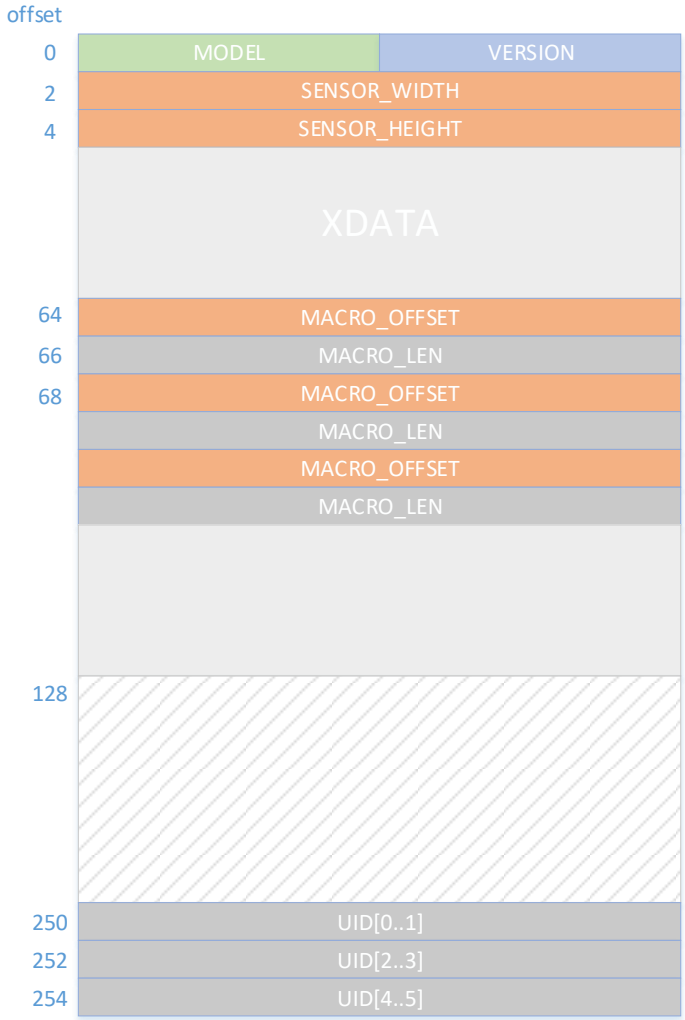

Рисунок 4. Структура данных в конфигурационной памяти.

Во второй половине конфигурационной памяти содержится до 16 дескрипторов макрокоманд. Каждый из них состоит из поля смещения в адресном пространстве flash-памяти, измеренном в килобайтах, и поля длины последовательности управляющих команд, которая хранится по этому смещению. Первый элемент таблицы макрокоманд соответствует команде INIT, которая выполняется автоматически при включении питания для инициализации FPGA. Остальные макрокоманды исполняются при получении соответствующих управляющих пакетов. Неиспользованные элементы таблицы макрокоманд заполняются нулями[3].

[3] Нулевое смещение не может использоваться для хранения макрокоманд, поскольку по нулевому смещению хранится конфигурационный битстрим FPGA.