# TeraSense USB API 2.0

Версия документа 0.1

## 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.

Поскольку поток данных со стороны последовательного порта один, FX2 работает либо в режиме потоковой передачи, либо в режиме обработки управляющих команд, что вполне соответствует типичным сценариям использования[[1]](#footnote-1). Выбор режима осуществляется с помощью флага Data Terminal Ready, который передается в устройство с помощью стандартных для последовательного порта управляющих запросов.

## 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[[2]](#footnote-2). 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

Управляющий пакет начинается с 4 байтного заголовка, где содержится поле команды CMD, поле размера пересылаемых данных SIZE и поле порядкового номера SN. Кроме того, в заголовке есть три битовых флага:

|  |  |
| --- | --- |
| R | В принятом пакете означает запрос подтверждения. В ответе устройства выставляется всегда. |
| X/E | Если установлен в запросе, то команда выполняется строго один раз (транзакционно). Для этого контроллер запоминает порядковый номер SN последней выполненной команды и сравнивает его[[3]](#footnote-3) с номером полученной команды. Бит, установленный в ответе, сигнализирует об ошибке, возникшей в процессе обработки запроса. При этом в поле SIZE передается код ошибки. |
| S | Если в команде установлен этот флаг, контроллер переключается в режим потоковой передачи данных сразу после завершения ее обработки. |

После заголовка пересылаются данные в запросах на запись и в ответах на запросы чтения.Следующий рисунок иллюстрирует структуру управляющего пакета, описанную выше.



Рисунок 2. Структура управляющего пакета.

Пакеты потоковых данных содержат завершающий заголовок (footer) в конце пакета, как показано на рисунке ниже.



Рисунок 3. Структура потокового пакета.

В заголовке содержится порядковый номер пакета SN и смещение маркера в пакете MARK\_OFFSET. Маркеры обозначают начало очередного кадра. Если в пакете несколько маркеров, поле MARK\_OFFSET содержит смещение первого из них. Если в пакете нет маркеров, то это поле содержит все единичные биты. Маркер используется для того, чтобы восстановить разбиение потока данных на кадры в случае потери части пакетов вследствие разрыва соединения. Размер данных в заголовке отсутствует, так как поток данных всегда разбивается на пакеты до полного заполнения очередного пакета, поскольку данные генерируются непрерывно.

## 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[[4]](#footnote-4).

1. При необходимости передавать поток данных и команды одновременно легко реализовать 2 последовательных порта в одном устройстве. [↑](#footnote-ref-1)
2. 64 байта для полной скорости, 512 байт для высокой скорости. [↑](#footnote-ref-2)
3. С учетом возможного переполнения 16-битного поля SN. [↑](#footnote-ref-3)
4. Нулевое смещение не может использоваться для хранения макрокоманд, поскольку по нулевому смещению хранится конфигурационный битстрим FPGA. [↑](#footnote-ref-4)