1 Работа по Ethernet

Для работы по Ethernet используются два TCP соединения с крейтом: одно для передачи команд крейту (порт 11110) и одно для передачи данных.

Формат команд описан в документе ltr_tcp_commands.txt.

Инициализация крейта по Ethernet выполняется следующим образом:

1. Передается команда INIT без данных, запрашивается 510 байт. В ответе крейт высылает в этих 510 байт информацию от FPGA, в которой каждый чётный байт равен 0xE0, а нечётный - собственно информация. Анализируется реально только первые 500 байт! В новых прошивках последние 10 байт могут использоваться для иного назначения.

2. Передается команда GET_NAME с запрашиваемым размером, равным 16

3. Чтение дескриптора с помощью команды GET_ARRAY:

   - Чтение описателя модуля типа t_RawDevDescr (t_ltr_eth_raw_descr) по адресу LTR010_MODULE_DESCRIPTOR_ADDRESS (LTR_CRATE_ADDR_MODULE_DESCR) с информацией о серийном номере + ещё какое-то имя, тип ЦПУ и частота (в новой версии использую только серийный номер и тип ЦПУ)
   - Чтение версии загрузчика по адресу LTR010_BOOT_LOADER_VERSION_ADDRESS (LTR_CRATE_ADDR_BOOTLOADER_VER). В старом сервере читалось 16 байт, а используется только 4 (4-ый байт старшая версия, 1-ый младшая). В новом читаю 4.
   - Чтение версии прошивки по адресу LTR010_FIRMWARE_VERSION_ADDRESS (LTR_CRATE_ADDR_FIRMWARE_VER). С размером то же самое что и с версией загрузчика.

4. Чтение ревизии платы через GET_ARRAY по адресу 0x90008001 (LTR_CRATE_ADDR_BOARD_REV) ревизии платы - 1 байт.

5. Открытие сокета для приёма данных

6. Установка маски скорости на медленную скорость.

2 Работа по USB

Используется 0-ая управляющая контрольная точка для передачи управляющих пользовательских запросов (для всех запросов тип VENDOR в соответствии со спецификацией USB) и две конечные точки типа BULK — одна на прием и одна на передачу. Номера конечных точке следует определить по стандартному дескриптору конфигурации USB — как первые найденные типа BULK в нужных направлений (в разных крейтах используются разные адреса конечных точек. Плюс в некоторых объявлены реально не используемые точки типа interrupt).

32-битный параметр управляющего запроса представляет передается следующем образом: старшие 16 байт в поле index, младшие — в поле value.

Инициализация крейта по USB выполняется в следующем порядке:

1. Получение имени устройства для определения типа крейта с помощью управляющего запроса LTR_USB_CMD_GET_MODULE_NAME (11) на 16 байт.

2. LTR010: попытка перевода в режим загрузчика через управляющий запрос LTR_USB_CMD_CALL_APPLICATION (15) с параметром LTR_CRATE_ADDR_BOOTLOADER_START (0x83003C00)

3. LTR010: попытка перевода в режим приложения через управляющий запрос LTR_USB_CMD_CALL_APPLICATION (15) с параметром LTR_CRATE_ADDR_FIRMWARE_START (0x83000000). Неудача выполнения запроса означает, что находимся в загрузчике. В этом случае завершаем сразу инициализацию.

4. получение описателя модуля с помощью запроса LTR_USB_CMD_GET_ARRAY (2) (аналогично Ethernet)

5. чтение версии загрузчика модуля (аналогично Ethernet)
6. чтение версии прошивки модуля (аналогично Ethernet)
7. LTR030, LTR031: чтение ревизии платы (аналогично Ethernet). При нахождении в загрузчике чтение ревизии не проходит (это единственный признак, который для LTR030/LTR031 отличает загрузчик от основной прошивки)
8. инициализация DMA через запрос LTR_USB_CMD_SET_DMA_PARAMETRS (4). index = 3, value = 0
9. сброс ПЛИС через запрос LTR_USB_CMD_RESET_FPGA (3)
10. LTR010, не в загрузчике: загрузка ПЛИС (с разбором текстового .ttf файла прошивки, где каждый байт представлен в виде трех ASCII символов и разделены запятой) по 4096 байт с помощью запроса LTR_USB_CMD_PUT_ARRAY начиная с адреса LTR_CRATE_ADDR_FPGA_FIRMWARE (0x86000000).
11. инициализация ПЛИС с помощью запроса LTR_USB_CMD_INIT_FPGA (3)
12. инициализация DMA через запрос LTR_USB_CMD_SET_DMA_PARAMETRS (4). index = 3, value = 0x0700
13. если не в загрузчике, то ждем по BULK начальный блок данных от ПЛИС (формат аналогичен ответу на INIT по Ethernet):
    - LTR010: передается 500 байт данных
    - LTR021: передается 500 байт, но в них после нужной информации идут нули, т.е. формат не совпадает с нужным. В некоторых прошивках ответ может вообще не прийти (это может также зависеть от наличия вставленного модуля), эту ситуацию приходится считать нормальной
    - LTR030/LTR031:
      - крейт настроен по USB, сбор данных был остановлен: передается 510 байт
      - крейт настроен по USB, сбор данных был запущен и не остановлен при передергивании кабеля или перезапуске сервера: данные могут прийти до нужных 510 байт! кроме того 510 байт могут и не прийти (но придет хотя бы мусор). Кроме того, исключение о изменении состава модулей может прийти до 510 байт или после, а если 510 байт нет, то может не прийти вообще.
      - крейт настроен по Ethernet: нет данных по BULK (т.к. BULK вообще не включен в этом режиме). Отчасти это может служить косвенным признаком того, что крейт настроен на Ethernet. По USB доступно только изменение настроек (интерфейс/адрес)
      - крейт находится в загрузчике: нет данных по BULK (отличие от предыдущего варианта только в отсутствии обработки команды получения ревизии...)

3 Разбор потока данных от крейта

Данные делятся на 4 типа:

- Данные от модуля - передаются клиенту без обработки
- Команды от модуля - так же передаются модулю за исключением синхрометок и статуса для LTR34
- Команды от крейта - обрабатываются сервером. Это синхрометки и исключительные ситуации. Последние передаются в виде двух слов - 0xFFFFFFFF, 0xFFFFAAXX (второе слово нарушает правило признака от кого данные и требует отдельной обработки), где XX - номер исключения:
  - 0 — системная ошибка крейта (по ней просто читаются системные регистры крейта)
  - 1 — признак, что изменился состав модулей. По этому исключению необходимо прочитать из системного регистра новый состав

  - 2 — признак переполнения внутреннего буфера для LTR021
  - 3 — признак, высылаемый на каждый пропущенный буфер при длительном переполнении
  - 4 — какой-то признак от LTR021, названный unknown sync — что значит непонятно
- Пользовательские данные - складываются в отдельный виртуальный канал (в ветке 2.0. эта возможность сейчас не поддерживается)

4 Протокол работы между клиентом и LTR-сервером

Обмен выполняется по сокетам с использованием протокола TCP (порт 11111). Соединения делятся на 2 типа:

- Управляющие (с сервером в целом или с крейтом) — по ним передаются только команды в виде запрос-ответе
- Потоковые (с модулем) — по ним передается только команда инициализации, после чего сервер просто передает все принятые от клиента данные в крейт и наоборот, вставляя только номер слота при передаче (разбор самих данных идет только для LTR34 для контроля его внутренней очереди).

4.1 Управляющие команды

Каждая команда состоит из следующих слов:

- 4 байта — признак начала команды — CONTROL_COMMAND_START (0xFFFFFFFF)
- 4 байта — код команды — CONTROL_COMMAND_BASE_ (0xABCDEF00) + N, где N — от 0 до 0xFF
- произвольное количество данных (зависит от команды)

На команду возвращается ответ:

- 4 байта — код ответа:
  - LTR_CLIENT_ACK_GOOD (0xABCDEFEE) - команда выполнена успешно
  - LTR_CLIENT_ACK_MODULE_IN_USE (0xABCDEFEE) - модуль используется (при попытке соединится с модулем, с которым уже установлено соединение)
  - LTR_CLIENT_ACK_BAD (0xABCDEFFF) - ошибка (до версии 2.0.0.0 — возвращался при любой ошибке при выполнении команды)
  - 0xABCDEF00 + код ошибки (не совпадающий ни с одним из вышеперечисленных) - команда завершена с ошибкой, код которой указан в младшем байте (введен в версии сервера 2.0.0.0)
- произвольное количество данных (зависит от команды и кода ответа)

4.2 Расширение управляющих команд

Проблема протокола в том, что если серверу приходит неизвестная команда, то он не знает сколько в ней данных и вынужден разорвать соединение с клиентом, т.к. теряется синхронизация.

Чтобы решить эту проблему, все новые команды, которые вводятся в версии сервера 2.0.0.0 и выше, представлены в следующем формате:

Команда:

- 4 байта — признак начала команды — CONTROL_COMMAND_START (0xFFFFFFFF)
- 4 байта — код команды — LTR_CLIENT_CMD_EX_HDR_SIZE (0xAC000000) + N, где N — от 0 до 0xFFFF
- 4 байта — количество данных на передачу в байтах

- 4 байта — запрашиваемый размер на прием в байтах
- указанное количество байт данных на передачу

Ответ:

- 4 байта — код ответа
- 4 байта — количество данных в ответе (не должно быть больше запрашиваемого)
- указанное количество данных

Таким образом, если код команды начинается с 0xAC00, то сервер знает что размер команды и размер ответа должны быть включены в заголовок команды и может обработать нужным образом неизвестные команды.

4.3 Инициализация соединения

Каждое соединение начинается с посылки команды CONTROL_COMMAND_INIT_CONNECTION (0). Команда передается в следующем формате:

- 4 байта — признак начала команды — CONTROL_COMMAND_START (0xFFFFFFFF)
- 4 байта — код команды — 0xABCDEF00
- 16 байт — серийный номер крейта, с которым устанавливается соединение. Если первый байт 0 => с первым найденным, при этом в ответе это поле будет заполнено серийный номером крейта, с которым реально установлено соединение.
- 2 байта — номер слота - младший байт — сам номер, старший байт - флаги: DR00 0III:
  - D - флаг отладки (в ветке 2.0.0.0 — не поддерживается)
  - R - raw data (в ветке 2.0.0.0 — не поддерживается)
  - I - Если не ноль, то явно задает интерфейс крейта, с которым нужно соединится. Может использоваться в случае если крейт подключен и по Ethernet и по USB (для настройки). Введен только с версии сервера 2.0.1.1.
- 4 байта — В запросе не имеет значения. В ответе возвращается текущее значение меток tmark для крейта, с которым установлено соединение.

В ответ приходит подтверждение. Если код подтверждения LTR_CLIENT_ACK_GOOD или LTR_CLIENT_ACK_MODULE_IN_USE, то после подтверждения приходит копия запроса, с вставленным реальным серийным номером крейта и значением tmark.