ПРОГРАММНЫЙ ПРОДУКТ

«Драйвер контроллера PCI-E для системы квантовой криптографии A&B»

Функциональное описание

Санкт-Петербург
2016 г.

Оглавление

1. Назначение системы....3
2. Краткое описание функциональности....3
3. Установка драйвера в систему....3
   3.1. Работа с исходными кодами....4
   3.2. Использование статической библиотеки....5
4. Программный интерфейс драйвера....5
   4.1. Вспомогательные типы....5
      4.1.1. Константы....5
      4.1.2. Перечисляемые типы....6
      4.1.3. Массив AnBTableMemorySize....8
      4.1.4. Объединение AnBReg....9
      4.1.5. Структура AnBRegInfo....11
   4.2. Объектная модель....11
      4.2.1. Перечислимый тип Errors и структура error....11
      4.2.2. Структура device_id....11
      4.2.3. Структура device_list....11
      4.2.4. Класс AnBDriverAPI....12
5. Описание кодов ошибок....19

1. Назначение системы

Драйвер разрабатывается для системы квантовой криптографии A&B (контроллер). Драйвер предназначен для организации взаимодействия прикладного программного обеспечения персональных компьютеров системы с контроллером A&B посредством интерфейса PCI-e.

Данный документ предназначен для пользователя, которым является программист высокоуровневого приложения, использующего драйвер.

2. Краткое описание функциональности

Драйвер представляет собой модуль ядра операционной системы, а также статическую библиотеку, реализующую программный интерфейс и часть функциональности драйвера.

Драйвер предоставляет программный интерфейс к регистрам контроллера, а также к его внутренним адресным пространствам. Драйвер позволяет получать данные от контроллера, которые передаются в режиме прямого доступа в память. Драйвер позволяет записывать в контроллер таблицы, а также считывать их в бинарном виде.

Драйвер позволяет получить список контроллеров, подключенных к системе.

Драйвер позволяет получать дамп регистров и таблиц контроллера в понятном для пользователя формате.

3. Установка драйвера в систему

Драйвер поставляется в формате RPM-пакета. Данный пакет содержит:

- src/ – исходные коды драйвера и утилиты тестирования (включая Makefile и сценарий сборки RPM-пакета);
- AnB.ko – бинарный файл модуля;
- devel/libAnB.a – бинарный файл статической библиотеки;
- devel/AnBDefs.h – заголовочный файл, содержащий описание вспомогательных типов драйвера;
- devel/driver.h – заголовочный файл, содержащий описание объектной модели, которая используется в библиотеке;
- test/AnBApp – утилита для тестирования драйвера;
- doc/ – документация, включая данный документ.

При установке пакета в файловую систему по адресу /tmp/AnB/ помещаются файлы, приведенные в списке выше. Также производится загрузка модуля ядра (AnB.ko) в операционную систему.

При удалении пакета модуль ядра выгружается из ОС, а файлы из /tmp/AnB удаляются.

Модуль ядра при необходимости можно загружать и выгружать вручную командами **insmod** и **rmmod**. При перезагрузке системы необходимо загрузить модуль вручную, поскольку пакет не настраивает систему на его автоматическую загрузку.

3.1. Работа с исходными кодами

Исходные коды располагаются в директории src/. Файлы в этой директории организованы следующим образом:

- Makefile – инструкция для утилиты **make**, содержащая правила для сборки модуля, библиотеки, тестового приложения, RPM-пакета и очистки рабочего пространства;
- PhotonAnB.spec – инструкция для утилиты **rpmbuild**, содержащая прямые правила для сборки RPM-пакета;
- module/ – директория с исходными кодами модуля ядра и с соответствующим Makefile-ом;
- lib/ – директория с исходными кодами статической библиотеки драйвера и соответствующим Makefile-ом;
- app/ – директория с исходными кодами утилиты тестирования драйвера и с соответствующим Makefile-ом.

Для сборки пакета необходимо выполнить следующую команду:

```
$ make rpm
```

После успешного выполнения собранный пакет можно будет найти в директории rpm/ (она будет создана в src/).

Для отдельной сборки модуля ядра необходимо выполнить следующую команду:

```
$ make module
```

Для отдельной сборки статической библиотеки необходимо выполнить следующую команду:

```
$ make lib
```

Для отдельной сборки утилитлы тестирования необходимо выполнить следующую команду:

```
$ make app
```

Для очистки рабочего пространства необходимо выполнить команду:

```
$ make clean
```

Для установки модуля в систему необходимо выполнить команду:

```
$ make install
```

Если необходимо, предыдущая команда также пересоберет модуль ядра.

Для удаления модуля из системы необходимо выполнить команду:

```
$ make remove
```

3.2. Использование статической библиотеки

Статическая библиотека libAnB.a предназначена для использования в приложении пользователя, которому необходимо осуществлять работу с устройствами A&B, подключенными к системе.

Помимо библиотеки существует два заголовочных файла, о которых упоминалось в пункте 3: driver.h и AnBDefs.h. Данные заголовочные файлы описывают непосредственно программный интерфейс драйвера (см. пункт 4).

Заголовочные файлы и саму библиотеку можно найти в директории devel/ (/tmp/AnB/devel) после установки пакета.

Библиотека должна быть слинкована с приложением в соответствии с правилами, определенными в описании утилит **gcc** или **g++** (или схожих используемых утилит-компиляторов). При линковке необходимо использовать ключ -pthread (помимо прочих).

4. Программный интерфейс драйвера

Драйвер представляет собой программную систему, состоящую из двух частей: модуля ядра ОС и объектной модели на C++, которая реализует программный интерфейс и вспомогательные функции драйвера и реализована в качестве статической библиотеки.

Программный интерфейс описан с помощью двух заголовочных файлов:

- driver.h – содержит описание объектной модели библиотеки;
- AnBDefs.h – содержит определения вспомогательных типов драйвера.

4.1. Вспомогательные типы

Вспомогательные типы описаны в заголовочном файле AnBDefs.h. В исходных кодах его можно найти в директории module/ (также его копия есть в директории lib/, данная копия является временной и не должна быть использованной). В установленном пакете его можно найти в директории devel/ (/tmp/AnB/devel/).

4.1.1. Константы

ANB_REG_MAX – максимально допустимый адрес регистра (значение по умолчанию – 0xFC).

ANB_REG_MIN – минимально допустимый адрес регистра (значение по умолчанию – 0x00).

AnBTableMaxSize – максимальный размер таблицы в словах (значение по умолчанию – 4096).

AnBBufferMinSize – максимальный размер фрейма буфера в байтах (значение по умолчанию – 65536).

AnBBuffers – количество фреймов в буфере (значение по умолчанию – 16).

Под буфером понимается область памяти, разбитая на равновеликое количество фреймов, каждый из которых используется для записи данных в пределах одного цикла прямого доступа к памяти.

4.1.2. Перечисляемые типы

Тип	Имя	Значение	Описание
AnBDevices			Представляет режимы работы контроллера A&B
	Alice	0	Режим работы – Алиса
	Bob	1	Режим работы – Боб
	AnBDevice_Illegal	2	Неверный режим (используется для проверки)
DumpSources			Представляет источники шестнадцатеричных дампов
	BAR1_RNG	0	Источник – таблица RNG
	BAR1_DAC	1	Источник – таблица DAC
	BAR1_CAL	2	Источник – AnBRegs таблица DAC_CAL
	BAR0	3	Источник – банк регистров
	DumpSource_Illegal	4	Неверный источник (используется для проверки)

Тип	Имя	Значение	Описание
DestTables			Представляет таблицы для записи или чтения
	TableRNG	0	Таблица RNG
	TableDAC	1	Таблица DAC
	TableCAL	2	Таблица DAC_CAL
	DestTable_Illegal	3	Неверная таблица (используется для проверки)
AnBTableModes			Представляет режимы таблиц
	Mode1	0	Первый режим
	Mode2	1	Второй режим
	Mode3	2	Третий режим
	Mode4	3	Четвертый режим
	Mode5	4	Пятый режим
	Mode6	5	Шестой режим
	AnBTableMode_Illegal	6	Неверный режим таблицы (используется для проверки)
AnBRegs			Представляет адреса регистров с известным назначением
	RegPattern	0x00	Регистр, содержащий константную строку «PCIe» в кодировке ASCII

Тип	Имя	Значение	Описание
	RegMode	0x08	Регистр, содержащий настройку режима контроллера A&B и выбор таблицы
	RegDMA	0x0C	Регистр настройки прямого доступа к памяти
	RegAddr_l	0x10	Регистр, содержащий младшие 4 байта адреса, используемого для прямого доступа к памяти
	RegAddr_h	0x14	Регистр, содержащий старшие 4 байта адреса, используемого для прямого доступа к памяти
	RegSize	0x18	Регистр, содержащий размер области в байтах, используемой для прямого доступа к памяти
	RegTable	0x24	Регистр, содержащий настройки размера таблицы и ее режима
	RegIRQ	0x38	Регистр статуса и разрешения прерываний

Таблица 1. Описание перечислимых типов.

4.1.3. Массив AnBTableMemorySize

Данный массив представляет максимальные размеры таблиц в байтах. Сам массив имеет размерность, которая соответствует значению DestTable_Illegal (см. таблицу 1). И определяет следующие значения в зависимости от индекса:

0 – 2048 (DestTables::TableRNG)

1 – 4096 * 4 (DestTables::TableDAC)

2 – 128 * 4 (DestTables::TableCAL)

4.1.4. Объединение AnBReg

Данное объединение представляет конкретный регистр с известным назначением в зависимости от контекста, который определяется пользователем. Данный тип объединяет структуры и поля, каждые из которых соответствуют определенному значению типа AnBRegs (см. таблицу 1).

Элемент объединения	Поле элемента	Разрядность	Описание
const char pattern[4]			Представляет регистр, расположенный по адресу AnBRegs::RegPattern
struct mode			Представляет регистр, расположенный по адресу AnBRegs::RegMode
	mode	1	Полю соответствуют значения типа AnBDevices
	reserved	23	Зарезервировано
	table	2	Полю соответствуют значения типа DestTables
struct dma			Представляет регистр, расположенный по адресу AnBRegs::RegDMA
	enabled	1	Данное поле содержит бит разрешения прямого доступа к памяти
DMAAddressLow			Представляет регистр, расположенный по адресу AnBRegs::RegAddr_l
DMAAddressHigh			Представляет регистр, расположенный по адресу AnBRegs::RegAddr_h
DMASize			Представляет регистр, расположенный по адресу AnBRegs::RegSize

Элемент объединения	Поле элемента	Разрядность	Описание
struct table			Представляет регистр, расположенный по адресу AnBRegs::RegTable
	size	16	Размер таблицы от 1 до AnBTableMaxSize включительно
	mode	4	Поле представляется значениями типа AnBTableModes
struct irq_status			Представляет регистр, расположенный по адресу AnBRegs::RegIRQ
	frame	1	Флаг прерывания фрейма PCI
	dma	1	Флаг прерывания цикла прямого доступа к памяти
	calibration	1	Флаг прерывания калибровки
unsigned int raw			Представляет содержимое регистра в сыром виде

Таблица 2. Описание объединения AnBRegs.

4.1.5. Структура AnBRegInfo

Структура AnBRegInfo содержит два поля:

- union AnBReg value – представляет значение регистра (см. таблицу 2);
- enum AnBRegs address – представляет адрес регистра.

Данная структура используется для исчерпывающего описания конкретного регистра, для которого известно его назначение (адрес которого представлен типом AnBRegs).

4.2. Объектная модель

Библиотека libAnB.a определяет класс AnBDriverAPI, перечислимый тип Errors, а также две структуры: error, device_id и device_list – описанные в заголовочном файле driver.h.

Данный класс и структуры являются частью пространства имен AnB.

4.2.1. Перечислимый тип Errors и структура error

Структура error представляет информацию об ошибке. Данная структура содержит два поля:

- enum Errors type – поле, содержащее код ошибки драйвера, представленное перечислимым типом Errors;
- int errn – поле, содержащее соответствующий код внутренней ошибки системы (соотвествует переменной errno).

В пункте 5 будут подробно описаны причины возникновения ошибок.

4.2.2. Структура device_id

Структура device_id содержит два поля:

- mutable char* const dev_path – строка, представляющая имя файла, сопоставленного устройству;
- mutable AnBDriverAPI* dev_ref – указатель на объект класса AnBDriverAPI.

Указатели, которые представлены полями данной структуры не должны освобождаться в пользовательской программе, если данная структура была получена в результате вызова метода GetDevicesList (см. пункт 4.2.4.1).

4.2.3. Структура device_list

Данная структура содержит два поля:

- const struct device_id* array – массив структур device_id;
- unsigned int count – количество элементов в массиве array.

Данная структура содержит деструктор, который освобождает указатель array, а также указатели, которые содержаться в каждой структуре device_id, являющейся элементом массива array.

Указатель, который представлен полем array данной структуры не должен освобождаться в пользовательской программе, если данная структура была получена в результате вызова метода GetDevicesList (см. пункт 4.2.4.1).

4.2.4. Класс AnBDriverAPI

Класс AnBDriverAPI реализует способ взаимодействия пользовательской программы с конкретным устройством. Данный класс позволяет получить список устройств, подключенных к системе, а также отслеживать ошибки, возникающие в процессе работы.

Каждый объект данного класса представляет конкретное устройство. Объект данного класса не может быть инстациирован напрямую. Для получения массива объектов необходимо воспользоваться методом GetDevicesList (см. пункт 4.2.4.1).

4.2.4.1. Метод GetDevicesList

Данный метод является статическим членом класса AnBDriverAPI.

Данный метод возвращает значение типа bool.

Данный метод принимает в качестве аргумента ссылку на структуру device_list (см. пункт 4.2.3).

В случае успешного завершения метода структура, переданная в него по ссылке, будет инициализирована, а метод вернет true. Успешным завершением метода является наличие в системе совместивых устройств и корректная инициализация соответствующих им объектов класса AnBDriverAPI. Структура device_list, переданная по ссылке, будет содержать массив структур device_id (см. пункт 4.2.2), каждая из которых соответствует объекту класса AnBDriverAPI, а также количество структур в массиве, которое также соотвествует количеству устройств в системе.

В случае неуспешного завершения метода содержимое структуры, переданной в него по ссылке, должно считаться некорректным после завершения метода. Использование структуры в дальнейшем приведет к непредсказуемому поведению системы. В случае неуспешного завершения метод возвращает значение false.

Прототип метода GetDevicesList выглядит следующим образом:

```
static bool GetDevicesList(struct device_list& devices);
```

4.2.4.2. Метод ErrorStr

Данный метод является статическим членом класса AnBDriverAPI.

Данный метод возвращает указатель на нуль-терминированную строку, содержащую описание ошибки, которая представлена аргументом.

Данный метод в качестве аргумента принимает ссылку на структуру error (см. пункт 4.2.1).

Указатель, возвращаемый данным методом, должен быть при необходимости освобожден в пользовательской программе (оператором delete[]).

Прототип данного метода выглядит следующим образом:

```
static const char* ErrorStr(const struct error& e);
```

4.2.4.3. Метод GlobalError

Данный метод является статическим членом класса AnBDriverAPI.

Данный метод в качестве аргумента принимает ссылку на структуру error (см. пункт 4.2.1). Метод инициализирует структуру, переданную по ссылке, в соответствии с последней ошибкой, возникшей при последнем вызове статического метода класса AnBDriverAPI.

Прототип данного метода выглядит следующим образом:

```
static void GlobalError(struct error& e);
```

4.2.4.4. Метод LastError

Вариации данного метода являются членами класса AnBDriverAPI.

Данный метод существует в двух вариациях.

1) Метод в качестве аргумента принимает ссылку на структуру error (см. пункт 4.2.1). Метод инициализирует структуру, переданную по ссылке, в соответствии с последней ошибкой, возникшей при последнем вызове метода класса AnBDriverAPI.

   Прототип метода выглядит следующим образом:

   ```
   void LastError(struct error& e);
   ```

2) Метод возвращает указатель на нуль-терминированную строку, содержащую описание ошибки, возникшей при последнем вызове метода класса AnBDriverAPI.

   Указатель, возвращаемый методом, должен быть при необходимости освобожден в пользовательской программе (оператором delete[]).

   Прототип метода выглядит следующим образом:

   ```
   const char* LastError();
   ```

4.2.4.5. Метод RegRawRead

Данный метод является членом класса AnBDriverAPI.

Использование данного метода не считается безопасным. Такое использование может в некоторых случаях приводить к непредсказуемой работе системы (например, в случае чтения регистра статуса прерываний).

Метод возвращает значение типа bool.

В качестве аргумента метод принимает ссылку на структуру AnBRegInfo (см. пункт 4.1.5). Аргумент представляет регистр, из которого производится чтение.

В случае успеха данный метод вернет значение true. Поле value аргумента, переданного по ссылке, будет инициализировано значением регистра, который расположен по адресу, соответствующему полю address этого аргумента.

Данный метод вернет false в случае возникновении ошибки. Значение поля value в таком случае будет не определено.

Прототип данного метода выглядит следующим образом:

```
bool RegRawRead(AnBRegInfo& reg);
```

4.2.4.6. Метод RegRawWrite

Данный метод является членом класса AnBDriverAPI.

Использование данного метода не считается безопасным. Такое использование может в некоторых случаях приводить к непредсказуемой работе системы (например, в случае записи регистра разрешения работы DMA).

Метод возвращает значение типа bool.

В качестве аргумента метод принимает ссылку на структуру AnBRegInfo (см. пункт 4.1.5). Аргумент представляет регистр, в который производится запись.

В случае успеха данный метод вернет значение true. В регистр, который расположен по адресу, соответствующему полю address, будет записано значение поля value переданного по ссылке аргумента.

Данный метод вернет false в случае возникновении ошибки.

Прототип данного метода выглядит следующим образом:

```
bool RegRawWrite(const AnBRegInfo& reg);
```

4.2.4.7. Метод GetDump

Данный метод является членом класса AnBDriverAPI.

Данный метод возвращает указатель на нуль-терминированную строку, которая содержит шестнадцатеричный дамп области памяти, представленной аргументом, в понятном для человека формате. В случае ошибки будет возвращен nullptr.

Метод в качестве аргумента принимает значение типа DumpSources (см. таблицу 1).

Ниже приведен пример получаемого дампа для регистрового банка:

```
00000000 50 43 49 65 01 00 00 00 01 00 00 00 00 00 00 00 |PCIe............|
00000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00000020 1c 00 00 00 01 00 00 00 aa aa 88 88 00 00 00 00 |................|
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00000040 00 00 26 00 00 20 00 00 40 4b 4c 00 00 00 00 00 |..&.. ..@KL.....|
00000050 00 00 00 00 f2 10 1b 05 fa cb 50 06 00 00 00 00 |..........P.....|
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
000000b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
000000c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
000000d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
000000e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
000000f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00000100
```

Указатель на строку, возвращаемый методом, должен при необходимости быть освобожден методом free() в пользовательской программе.

Прототип метода GetDump выглядит следующим образом:

```
const char* GetDump(enum DumpSources source);
```

4.2.4.8. Метод WriteTable

Данный метод является членом класса AnBDriverAPI.

Данный метод возвращает значение типа bool.

Данный метод принимает три аргумента:

- const char* table – указатель на массив байт, который представляет таблицу, сформированный в пользовательском приложении;
- unsigned int size – размер массива table в байтах;
- enum DestTables dest – значение, представляющее таблицу, в которую осуществляется запись (см таблицу 1).

В случае успешной записи данный метод вернет значение true. В таблицу назначения dest будет записано size байт из массива table.

В случае ошибки будет возвращено значение false.

Прототип данного метода выглядит следующим образом:

```
bool WriteTable(const char* table, unsigned int size, enum DestTables dest);
```

4.2.4.9. Метод ReadTable

Данный метод является членом класса AnBDriverAPI.

Данный метод возвращает значение типа bool.

Данный метод принимает три аргумента:

- char*& table – ссылка на указатель на массив байт;
- unsigned int& size – ссылка на размер массива table в байтах;
- enum DestTables dest – значение, представляющее таблицу, из которую осуществляется чтение (см таблицу 1).

В случае успешной записи данный метод вернет значение true. Указатель, переданный по ссылке, table будет инициализирован значением указателя на выделенную в методе область памяти размером, которым будет инициализирован аргумент size. Данный размер будет соответствовать значениям массива AnBTableMemorySize (см. пункт 4.1.3). В массив table будет записано содержимое указанной таблицы.

В случае ошибки будет возвращено значение false. Значения аргументов table и size будут не определены.

Прототип данного метода выглядит следующим образом:

```
bool ReadTable(char*& table, unsigned int& size, enum DestTables dest);
```

4.2.4.10. Метод SetDev

Данный метод является членом класса AnBDriverAPI.

Данный метод возвращает значение типа bool.

Данный метод в качестве аргумента принимает значение типа AnBDevices (см. Таблицу 1.)

В случае успеха данный метод возвращает значение true. А регистр, представленный значением AnBRegs::RegMode, будет соответствующим образом обновлен (обновляется бит, соответствующий режиму работы контроллера A&B).

В случае ошибки данный метод вернет значение false.

Прототип данного метода выглядит следующим образом:

```
bool SetDev(enum AnBDevices mode);
```

4.2.4.11. Метод SetTable

Данный метод является членом класса AnBDriverAPI.

Данный метод возвращает значение типа bool.

Данный метод принимает два аргумента:

- enum AnBTableModes mode – режим работы таблицы (см. таблицу 1);
- unsigned int size – размер таблицы.

В случае успеха данный метод возвращает значение true. А регистр, представленный значением AnBRegs::RegTable, будет соответствующим образом обновлен.

В случае ошибки данный метод вернет значение false.

Прототип данного метода выглядит следующим образом:

```
bool SetTable(enum AnBTableModes mode, unsigned int size);
```

4.2.4.12. Метод SetBuffersCount

Данный метод является членом класса AnBDriverAPI.

Данный метод возвращает значение типа bool.

Данный метод в качестве аргумента принимает значение типа unsigned int.

В случае успеха данный метод вернет true, а количество буферов будет установлено равным значению аргумента.

В случае ошибки метод вернет значение false.

Прототип данного метода выглядит следующим образом:

```
bool SetBuffersCount(unsigned int count);
```

4.2.4.13. Метод DMAEnable

Данный метод является членом класса AnBDriverAPI.

Данный метод возвращает значение типа bool.

Данный метод запускает подсистему драйвера, обслуживающую прямой доступ к памяти.

В случае успешного запуска метод возвращает значение true.

В случае возникновения ошибки данный метод возвращает значение false (в некоторых случаях дальнейшая работа системы считается непредсказуемой — см. пункт 5).

Прототип данного метода выглядит следующим образом:

```
bool DMAEnable();
```

4.2.4.14. Метод DMADisable

Данный метод является членом класса AnBDriverAPI.

Данный метод возвращает значение типа bool.

Данный метод останавливает подсистему драйвера, обслуживающую прямой доступ к памяти.

В случае успешного запуска метод возвращает значение true.

В случае возникновения ошибки данный метод возвращает значение false (в некоторых случаях дальнейшая работа системы считается непредсказуемой — см. пункт 5).

Прототип данного метода выглядит следующим образом:

```
bool DMADisable();
```

4.2.4.15. Метод DMARead

Данный метод является членом класса AnBDriverAPI.

Данный метод возвращает значение типа bool.

Данный метод в качестве аргумента принимает ссылку на указатель на массив байт.

В случае успеха данный метод вернет значение true. Аргумент будет проинициализирован значением указателя на область памяти, которая будет выделена методом. Размер этой области будет соответствовать значению константы AnBBufferMinSize (см. пункт 4.1.1). Массив будет проинициализирован значениями, полученными от контроллера A&B за очередной цикл прямого доступа к памяти.

В случае ошибки данный метод вернет false. Значение аргумента должно считаться неопределенным.

Данный метод является блокирующим. Блокировка происходит до тех пор, пока очередная порция данных не будет готова (пока хотя бы один цикл прямого доступа к памяти не будет завершен на стороне драйвера).

Указатель, получаемый посредством данного метода, должен быть освобожден (оператором delete[]) при необходимости в пользовательской программе.

Прототип данного метода выглядит следующим образом:

```
bool DMARead(char*& buffer);
```

4.2.4.16. Метод DMAIsReady

Данный метод является членом класса AnBDriverAPI.

Данный метод возвращает значение true, когда существует завершенный цикл прямого доступа к памяти, данные которого доступны для чтения (вызов метода DMARead не будет блокирующим).

Данный метод возвращает значение false, когда нет ни одного завершенного цикла прямого доступа к памяти, данные которого доступны для чтения (вызов метода DMARead заблокирует вызвавший его поток исполнения).

Прототип данного метода выглядит следующим образом:

```
bool DMAIsReady();
```

5. Описание кодов ошибок

В таблице 3 приведены описания кодов ошибок, которые соответствуют значениям структуры error (см. пункт 4.2.1).

Для поля errno вместо значения может приведен системный вызов ОС, в документации которого следует искать дополнительную информацию об ошибке.

Колонка «X» соответствует маркировке ошибок, при возникновении которых дальнейшая работа системы считается непредсказуемой.

При возникновении ошибок, не описанных в таблице 3, поведение системы должно считаться непредсказуемым, а код ошибки должен быть зафиксирован.

Метод	**Код ошибки**		**X**	**Описание**
	type	**errno**		
GetDevicesList	1	opendir		Невозможно получить доступ к директории /dev
	2	realloc		Невозможно выделить память
	3	0		Устройств не обнаружено
	5	0	X	Приватному методу драйвера был передан nullptr в качестве аргумента
	4	open		Не удается открыть файл, ассоциированный с устройством

Метод	Код ошибки		X	Описание
	type	errno		
		ENODEV		Данные устройства не были инициализированы
	6	ioclt		Не удается выполнить ioctl
		EINVAL		Версия модуля отличается от версии библиотеки
RegRawRead	14	ioctl		Не удается выполнить ioctl
		EINVAL		Неверно задан адрес регистра. Адрес находится вне пределов (см. пункт 4.1.1), либо не кратен четырем.
		EBUSY		Доступ к регистрам при запущенном прямом доступе к памяти
RegRawWrite	15	ioctl		Не удается выполнить ioctl
		EINVAL		Неверно задан адрес регистра. Адрес находится вне пределов (см. пункт 4.1.1), либо не кратен четырем.
		EBUSY		Доступ к регистрам при запущенном прямом доступе к памяти
GetDump	16	ioctl		Не удается выполнить ioctl
		EINVAL		Неверно задан аргумент
	24	0		Невозможно прочитать дамп
WriteTable	5	0		Методу был передан nullptr в качестве первого аргумента
	17	ioctl		Не удается выполнить ioctl

Метод	Код ошибки		X	Описание
	type	errno		
		EINVAL		Неверно задан третий аргумент
	20	write		Не удается выполнить write
		ENODEV	X	Данные устройства не были инициализированы
		EBUSY		Доступ к таблице при запущенном прямом доступе к памяти
ReadTable	17	ioctl		Не удается выполнить ioctl
		EINVAL		Неверно задан третий аргумент
	19	read		Не удается выполнить read
		ENODEV	X	Данные устройства не были инициализированы
		EBUSY		Доступ к таблице при запущенном прямом доступе к памяти
SetDev	18	ioctl		Не удается выполнить ioctl
		EBUSY		Изменение режима работы устройства при запущенном прямом доступе к памяти
		EINVAL		Неверно задан аргумент
SetTable	21	ioctl		Не удается выполнить ioctl
		EINVAL		Неверно задан второй аргумент
		EBUSY		Изменение параметров устройства при запущенном прямом доступе к памяти
	22	ioctl		Не удается выполнить ioctl

Метод	Код ошибки		X	Описание
	type	errno		
		EINVAL		Неверно задан первый аргумент
		EBUSY		Изменение параметров устройства при запущенном прямом доступе к памяти
SetBuffersCount	23	ioctl		Не удается выполнить ioctl
		EBUSY		Изенение количества буферов при запущенном прямом доступе к памяти
		EINVAL		Неверно задан аргумент
		ENOMEM		Недостаточно памяти
DMAEnable	7	ioctl		Не удается выполнить ioctl
		EINVAL		Подсистема уже запущена
	10	mmap		Не удается выполнить mmap
		BUSY	X	Подсистема уже запущена
	11	pthread_create		Не удается создать поток для подсистемы
	8	ioctl	X	BUG
DMADisable	9	ioclt		Не удается выполнить ioctl
		EINVAL		Подстистема уже остановлена
	12	pthread_join	X	Не удается завершить поток
DMARead	13	0		Подсистема не запущена

Таблица 3. Коды ошибок.