Руководство (v1.0)

По работе с драйвером модуля “mPCIe–TTCAN”

Интерфейс ISO-11898
(CAN Bus)

Для драйверов версии 1.0

ОС WINDOWS

18.09.2015

ООО “Новомар” 2015 г.

Оглавление

1. Введение

Основой драйвера для модуля “**mPCIe–TTCAN”** является файл TTCAN_YY_VerX.sys, где YY является разрядностью ОС (x86 или x64), для которой разработан драйвер, а X версия драйвера.

Драйвер разрабатывался и тестировался для операционных систем Microsoft Windows XP 32 bit edition, Microsoft Windows 7 32 bit edition и Microsoft Windows 7 64 bit edition.

При установке драйвера на ОС Windows XP, система выдаст предупреждение о том, что данное ПО не прошло сертификацию для Windows XP. Это связано с тем, что компания Microsoft более не поддерживает Windows XP. Нажмите кнопку "Продолжить". Драйвер установится и начнёт свою корректную работу.

Драйвер поддерживает как работу одного, так и одновременную работу нескольких устройств и присваивает им уникальные символьные имена **TTCANx**, где X – индекс устройства, начиная с 0. Т.е. при одновременной работе двух и более устройств, одно будет иметь имя TTCAN0, второе TTCAN1 и т.д.

2. Установка драйвера

Установка драйвера производится стандартными средствами установки оборудования системы Windows.

Для установки драйвера следует открыть "Диспетчер устройств", выбрать устройство с идентификатором **PCI\VEN_A203&DEV_9471&REV_02** и нажать установить драйвер. Идентификатор можно просмотреть в свойствах устройства, во вкладке "Сведения", выбрав пункт "ИД оборудования".

Далее следует выбрать кнопку "Выполнить поиск драйверов на этом компьютере", указать путь к директории драйвера и нажать “Далее”.

Если система отобразит ошибку, что не удалось найти драйвер для этого устройства, значит устройство выбрано неверно. Проверьте идентификатор устройства.

Если система отобразит сообщение, что драйвер установлен, то можно приступать к работе с устройством.

Модуль теперь можно найти в "Диспетчере устройств" в ветке "Multifunction Adapters" под именем "Novomar(R) TTCan Controller".

2.1. Расшифровка названия драйвера.

TTCAN_x64_Ver1_0.sys
1 2 3 4

1	***TTCAN***	Название поддерживаемого модуля
2	***x64***	Разрядность ОС, для которой предназначен драйвер
3	***Ver1_0***	Версия драйвера
4	***sys***	Расширение файла

3. Список доступных IOCTL вызовов по версиям драйверов

Название вызова	Краткое описание
Список вызовов, доступных **в драйвере версии до 1.0**	
IOCTL_TTCANDEV_CLEAR_DMA	Сброс счётчиков DMA
IOCTL_TTCANDEV_DISABLE_CANONICAL_DMA	Выключение режима “CanonicalDMA”
IOCTL_TTCANDEV_ENABLE_CANONICAL_DMA	Включение режима “CanonicalDMA”
IOCTL_TTCANDEV_GET_DEVICE_VER	Номер ревизии устройства
IOCTL_TTCANDEV_DISABLE_INTERRUPT	Запрет прерывания
IOCTL_TTCANDEV_GET_NBLOCK_RAW_DMA	Кол-во новых блоков данных
IOCTL_TTCANDEV_GET_RESOURCE_INFO	Информация о ресурсах устройства
IOCTL_TTCANDEV_READ_BAR	Чтение регистров
IOCTL_TTCANDEV_DMA_COUNT	Программный счётчик DMA
IOCTL_TTCANDEV_READ_DMA_CH_DATA_BLOCKS	Чтение данных из буфера DMA
IOCTL_TTCANDEV_READ_DMA_DATA_BLOCKS	Чтение данных из буфера DMA
IOCTL_TTCANDEV_READ_DMA_DATABUF	Чтение данных из буфера DMA
IOCTL_TTCANDEV_READ_DMA_STATUS	Чтение статуса DMA
IOCTL_TTCANDEV_READ_REG	Чтение регистров
IOCTL_TTCAN_REGISTER_INTERRUPT	Разрешение прерывания
IOCTL_TTCANDEV_WRITE_BAR	Запись регистров
IOCTL_TTCANDEV_WRITE_REG	Запись регистров

4. Режим “CanonicalDMA”

В данном программном обеспечении предусмотрен режим CANONICAL, обеспечивающий независимое чтение информации из буфера DMA по каналам (режим включается/выключается для всех каналов устройства одновременно):

- чтение полученной информации с одного канала устройства не оказывает влияние на другой канал;
- при чтении с канала пользователь получает данные только этого канала.

При включении/выключении режима указатель буфера DMA устройства обнуляется.

Для включения используется вызов IOCTL_TTCANDEV_ENABLE_CANONICAL_DMA, для выключения IOCTL_TTCANDEV_DISABLE_CANONICAL_DMA.

В данном режиме недоступны некоторые функции. Для того чтобы понять доступна ли функция в данном режиме смотрите пункт “примечание” в описании функций. Если об этом ничего не сказано, то функция доступна как в данном режиме, так и вне его.

5. IOCTL вызовы.

В данном описании приняты следующие соглашения и допущения:

1) по умолчанию, смещения указываются в шестнадцатеричном виде (hex);
2) по умолчанию, размеры указываются в десятичном виде;

Во всех примерах вызовов есть три неописанные переменные, это

- m_hDevice - дескриптор устройства на котором должна выполниться операция. Чтобы извлечь данные о дескрипторе устройства, используйте функцию **CreateFile**.
- lpBytesReturned - указатель на переменную, которая получает размер переданных данных.
- xxx – параметр который должен быть равен либо NULL, либо ссылке на структуру OVERLAPPED**,** для того чтобы операция выполняется как перекрывающая (асинхронная) операция.

В случае удачного выполнения функция возвращает ненулевое значение. В случае ошибки возвращаемое значение равно нулю. Чаще всего это происходит из-за некорректных входных данных. Чтобы получить дополнительную информацию об ошибке, вызовите **GetLastError**.

Вызовы сгруппированы по алфавиту.

После загрузки драйвера запрещена работа устройства, работа DMA, все прерывания, указатель DMA сброшен в 0.

Если вы уже работали с предыдушей версией драйвера, то вам необходимо обратить внимание на раздел “Обновление новой версии руководства”, т.к. некоторые обновления драйвера могут нарушить корректную работу вашего программного обеспечения.

5.1. IOCTL_TTCANDEV_CLEAR_DMA

Назначение:

По вызову данной функции программный счётчик вычитанных пакетов данных DMA уравнивается с регистром DMA_INDEX*(адрес 0x1008).

*См. Раздел 5.1.2 документа "*Руководство по программированию mPCIe-TTCAN*".

Действие:

–

Примечание:

ВНИМАНИЕ!!!

ФУНКЦИЮ ИСПОЛЬЗОВАТЬ ОЧЕНЬ АККУРАТНО! ВОЗМОЖНА ПОТЕРЯ ДАННЫХ. ПЕРЕД ВЫЗОВОМ ПРОВЕРЬТЕ ВЫЧИТАНЫ ЛИ ВСЕ ДАННЫЕ.

Вход:

–

Выход:

–

Пример вызова:

```
if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_CLEAR_DMA,
  NULL, 0,
  NULL, 0,
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()
```

5.2. IOCTL_TTCANDEV_DISABLE_CANONICAL_DMA

Назначение:

Выключение режима “CanonicalDMA”* работы с DMA.

*см. раздел 4 данного документа.

Действие:

Два программных счётчика пакетов, содержащие количество прочитанных пакетов по соответствующему каналу обнуляются. Вместо них возвращается один счётчик на оба канала. Он приравнивается к регистровому счётчику пакетов (регистр DMA_INDEX* (адрес 0x1008)).

Выключается режим “CanonicalDMA”, после которого становятся доступны новые функции и недоступны некоторые старые. Режим включается для обоих каналов одновременно.

*См. Раздел 5.1.2 документа “*Руководство по программированию mPCIe-TTCAN* ”.

Примечание:

Для того чтобы понять работает какой-либо вызов с данным режимом или нет, смотрите раздел “примечание” просматриваемого вызова. Если про данный режим ничего не сказано, значит вызов может работать как с данным режимом, так и без него.

Вход:

–

Выход:

–

Пример вызова:

```
if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_DISABLE_CANONICAL_DMA,
  NULL, NULL,
  NULL, NULL,
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()
```

5.3. IOCTL_TTCANDEV_DISABLE_INTERRUPT

Назначение:

Запрет прерывания m_nInt. (Функция разрешения прерывания отсутствует, т.к. прерывание разрешается в вызове IOCTL_TTCANDEV_REGISTER_INTERRUPT).

Действие:

Функция запрещает прерывание m_nInt, записывая данные либо только в регистр INTERRUPT MASK* (адрес 1010h), либо и в регистр INTERRUPT MASK и в регистр CANINTE** (адрес 2Bh).

* см. Раздел 5.1.4 документа "*Руководство по программированию модуля " mPCIe-TTCAN "*.

** см. Раздел 6.4.1 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

Примечание:

Варианты m_nInt:

- **HDAT** – прерывание по заполнению 1/2 буфера данных;
- **QDAT** – прерывание по заполнению 1/8 буфера данных;
- **TIM_CAN1** – Прерывание таймера CAN1. Порог срабатывания указывается в регистре CAN1_INT_TRIG*;
- **TIM_CAN2** – Прерывание таймера CAN2. Порог срабатывания указывается в регистре CAN2_INT_TRIG*;
- **RX0IE** – прерывание по получению сообщения в буфер приёма 0;
- **RX1IE** – прерывание по получению сообщения в буфер приёма 1;
- **TX0IE** – прерывание по отправке сообщения из буфера передачи 0;
- **TX1IE** – прерывание по отправке сообщения из буфера передачи 1;
- **TX2IE** – прерывание по отправке сообщения из буфера передачи 2;
- **ERRIE** – прерывание по событию ошибки шины или переполнению буферов (конкретный источник - в регистре EFLG);
- **WAKIE** – прерывание по событию выхода из режима сна;
- **MERRE** – прерывание по ошибке приёма либо передачи сообщения.

Вход – структура TTCAN_INT_NTFCN_IN:

Смещение	Размер	Назначение	Примечание
0x00	4	*m_nCh:* номер канала. Для прерываний HDAT, QDAT, TIM_CAN1 и TIM_CAN2 не используется. Для остальных прерываний должен быть равен 1 или 2.	
0x00	4	*m_nInt:* номер прерывания	См. п. примечание
0x04	4	*m_hEvt:*	Не используется

Выход:

-

Пример вызова на следующей странице:

Пример вызова:

```
TTCAN_INT_NTFCN_IN InputBuf;
InputBuf.m_nInt = HDAT;

if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_DISABLE_INTERRUPT,
  &InputBuf, sizeof(InputBuf),
  NULL, 0,
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()
```

5.4. IOCTL_TTCANDEV_ENABLE_CANONICAL_DMA

Назначение:

Включение режима “CanonicalDMA”* работы с DMA.

*см. раздел 4 данного документа.

Действие:

Один программный счётчик вычитанных пакетов данных DMA обнуляется. Он заменяется на два, каждый из которых приравнивается к регистровому счётчику пакетов (регистр DMA_INDEX* (адрес 0x1008)) и содержит количество пакетов по соответствующей шине.

Включается режим “CanonicalDMA”, после которого становятся доступны новые функции и недоступны некоторые старые. Режим включается для обоих каналов одновременно.

*См. Раздел 5.1.2 документа “*Руководство по программированию mPCIe-TTCAN*”.

Примечание:

Для того чтобы понять работает какой-либо вызов с данным режимом или нет, смотрите раздел “примечание” просматриваемого вызова. Если про данный режим ничего не сказано, значит вызов может работать как с данным режимом, так и без него.

Вход:

–

Выход:

–

Пример вызова:

```
if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_ENABLE_CANONICAL_DMA,
  NULL, NULL,
  NULL, NULL,
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()
```

5.5. IOCTL_TTCANDEV_GET_DEVICE_VER

Назначение:
Чтение номера ревизии устройства.

Действие:
Функция читает значение ревизии устройства из конфигурационного адресного пространства PCI.

Примечание:

–

Вход:

–

Выход:

Смещение	Размер	Назначение	Примечание
0x00	4	Номер ревизии устройства	

Пример вызова:

```
UINT32 Output;

if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_GET_DEVICE_VER,
  NULL,NULL,
  &Output, sizeof(Output),
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()

Revision = Output;
```

5.6. IOCTL_TTCANDEV_GET_NBLOCK_RAW_DMA

Назначение:

Чтение количества новых блоков данных, накопленных в буфере DMA устройства по всем каналам (блок - 64 байта).

Действие:

Функция вычитает из значения регистра DMA_INDEX*(адрес 0x1008) значение программного счётчика вычитанных пакетов. Полученное количество блоков возвращается в переменной nValue.

*См. Раздел 5.1.2 документа "*Руководство по программированию mPCIe-TTCAN*".

Примечание:

Не доступна в режиме "CanonicalDMA". Воспользуйтесь вызовом IOCTL_TTCANDEV_DISABLE_CANONICAL_DMA.

Вход:

–

Выход – структура TTCAN_GET_NBLOCK_DMA_OUT:

Смещение	Размер	Назначение	Примечание
0x00	4	*m_nValue:* Кол-во новых блоков	

Пример вызова:

```
TTCAN_GET_NBLOCK_DMA_OUT Output;

if(::DeviceIoControl(
  m_hDevice,
IOCTL_TTCANDEV_GET_NBLOCK_RAW_DMA,
  NULL,NULL,
  &Output, sizeof(Output),
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()

nValue = Output.m_nValue;
```

5.7. IOCTL_TTCANDEV_GET_RESOURCE_INFO

Назначение:
Получение информации о ресурсах и состоянии.

Действие:
Функция читает значение ревизии устройства из конфигурационного адресного пространства PCI.

Примечание:
-

Вход:

–

Выход – структура TTCAN_GET_RESOURCE_INFO_OUT:

Смещение	Размер	Назначение	Примечание
0x00	4	*m_nBus* : Номер шины	
0x04	4	*m_nSlot* : Номер слота	
0x08	4	*m_nFunction* : Номер функции	
0x0C	4	*m_nDataBufPhysAddr* : Физический адрес буфера DMA	
0x10	4	*m_nDataBufSize* : Размер буфера DMA	В байтах.
0x14	4	*m_BarPhysAddr*: Физический адрес пространства регистра BAR	
0x18	128	*m_szCompileDate* : Текстовая строка, заканчивающаяся нулем, с информацией о времени и дате компиляции драйвера	

Пример вызова:

```
TTCAN_GET_RESOURCE_INFO_OUT Output;

if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_GET_DEVICE_VER,
  NULL,NULL,
  &Output, sizeof(Output),
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()

pBus = Output.m_nBus;
pSlot = Output.m_nSlot;
nFunction = Output.m_nFunction;
pBufAddr = Output.m_nDataBufPhysAddr;
pBufSize = Output.m_nDataBufSize;
pBarAddr = Output.m_BarPhysAddr;
pStr = (char*)Output.m_szCompileDate;
```

5.8. IOCTL_TTCANDEV_READ_BAR

Назначение:
Чтение данных из регистрового пространства устройства (BAR).

Действие:
Функция читает необходимое количество данных из желаемого адреса регистрового пространства устройства (BAR) в переменную pData.

Примечание:
Рекомендуется использовать если данные больше 32 бит. Если меньше или равны 32 битам, то рекомендуется использовать вызов IOCTL_TTCANDEV_READ_REG.

Вход – структура TTCAN_READ_BAR_IN:

Смещение	Размер	Назначение	Примечание
0x00	4	*m_nOffset*: Смещение блока данных на чтение	
0x04	4	*m_nSize*: Размер блока данных	

Выход:

Смещение	Размер	Назначение	Примечание
0x00	*m_nSize*	Считанный блок данных	

Пример вызова:

```
TTCAN_READ_BAR_IN Input;
Input.m_nOffset = nAddr;
Input.m_nSize = nSize;
Input.m_nBar = nBar;

if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_READ_BAR,
  &Input, sizeof(Input),
  &Data, sizeof(Data),
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()
```

5.9. IOCTL_TTCANDEV_DMA_COUNT

Назначение:
Чтение программного счётчика записанных блоков данных в DMA.

Действие:
Функция забирает значение счётчика из внутренних переменных драйвера.

Примечание:
Не доступна в режиме "CanonicalDMA". Воспользуйтесь вызовом IOCTL_TTCANDEV_DISABLE_CANONICAL_DMA.

Вход:

–

Выход – TTCAN_DMA_COUNT_OUT:

Смещение	Размер	Назначение	Примечание
0x00	4	*m_nData:* Значение счётчика	В байтах

Пример вызова:

```
TTCAN_DMA_COUNT_OUT OutputBuf;

if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_DMA_COUNT,
  NULL, NULL,
  &OutputBuf, sizeof(OutputBuf),
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()

nCount = OutputBuf.m_nCount;
```

5.10. IOCTL_TTCANDEV_READ_DMA_CH_DATA_BLOCKS

Назначение:

Чтение данных из буфера DMA. Данная функция читает любое требуемое значение новых блоков данных DMA по запрашиваемому каналу.

Действие:

По номеру запрашиваемого канала функция перебирает блоки данных в DMA, проверяя из какого канала пришли данные. Если из нужного то она кладёт данные в переменную pData до тех пор, пока либо не наберёт запрашиваемое количество блоков, либо блоки с новыми данными не закончатся.

Примечание:

Доступна только в режиме "CanonicalDMA". Воспользуйтесь вызовом IOCTL_TTCANDEV_ENABLE_CANONICAL_DMA.

Вход – структура TTCAN_READ_DMA_CH_DATA_BLOCKS_IN:

Смещение	Размер	Назначение	Примечание
0x00	4	*m_nBlocks*: Запрашиваемое кол-во блоков данных для чтения	
0x04	4	*m_nCh*: Номер шины, пришедшие данные с которой следует вычитать.	0-1

Выход:

Смещение	Размер	Назначение	Примечание
0x00	*m_nBlocks*64*	Считанный блок данных	

Пример вызова:

```
TTCAN_READ_DMA_CH_DATA_BLOCKS_IN InputBuf;
//Размер запрашиваемых данных
UINT32 nOutputSize = nBlocks*128;
InputBuf.m_nBlocks = nBlocks;
InputBuf.m_nCh = nCh;

if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_READ_DMA_CH_DATA_BLOCKS,
  &InputBuf, sizeof(InputBuf),
  &Data, nOutputSize,
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()

//размер прочитанных данных
pBlocks = lpBytesReturned/64;
```

5.11. IOCTL_TTCANDEV_READ_DMA_DATA_BLOCKS

Назначение:

Чтение данных из буфера DMA. Данная функция читает любое требуемое количество новых блоков данных DMA.

Действие:

Чтобы воспользоваться функцией, рекомендуется узнать количество новых блоков данных с помощью вызова IOCTL_TTCANDEV_GET_NBLOCK_RAW_DMA. Значение возвращенное этой функцией, использовать в качестве входного параметра для переменной nBlocks.

После выполнения функции указатель буфера DMA сдвигается на соответствующую количеству прочитанных данных позицию.

Примечание:

Не доступна в режиме "CanonicalDMA". Воспользуйтесь вызовом IOCTL_TTCANDEV_DISABLE_CANONICAL_DMA.

Вход – структура TTCAN_READ_DMA_DATA_BLOCKS_IN:

Смещение	Размер	Назначение	Примечание
0x00	4	*m_nBlocks*: Запрашиваемое кол-во блоков данных для чтения	

Выход:

Смещение	Размер	Назначение	Примечание
0x00	*m_nBlocks*64*	Считанный блок данных	

Пример вызова:

```
TTCAN_READ_DMA_DATA_BLOCKS_IN InputBuf;
//Размер запрашиваемых данных
UINT32 nOutputSize = nBlocks*64;
InputBuf.m_nBlocks = nBlocks;

if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_DMA_COUNT,
  NULL, NULL,
  &OutputBuf, sizeof(OutputBuf),
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()

//размер прочитанных данных
pBlocks = lpBytesReturned/64;
```

5.12. IOCTL_TTCANDEV_READ_DMA_DATABUF

Назначение:

Чтение данных из буфера DMA. Данная функция вычитывает только один новый блок данных.

Действие:

Данная функция читает данные из буфера DMA только по одному блоку (блок - 64 байт). Для того, чтобы проверить есть ли готовые данные надо вызвать функцию IOCTL_TTCAN_READ_DMA_STATUS.

Если возвращаемое ей значение равно ‘1’, значит данные есть и их следует вычитать вызвав данную функцию, и снова проверить статус готовности данных DMA. И так пока статус не станет равен нулю.

Примечание:

Не доступна в режиме “CanonicalDMA”. Воспользуйтесь вызовом IOCTL_TTCANDEV_DISABLE_CANONICAL_DMA.

Вход:

–

Выход:

Смещение	Размер	Назначение	Примечание
0x00	64	Считанный блок данных	В байтах

Пример вызова:

```
if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_READ_DMA_DATABUF,
  NULL, NULL,
  &OutputBuf, 64,
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()
```

5.13. IOCTL_TTCANDEV_READ_DMA_STATUS

Назначение:
Чтение статуса DMA.

Действие:
Функция сравнивает программный счётчик вычитанных пакетов данных из DMA со значением регистра DMA_INDEX*(адрес 0x1008), если значения не равны, то новые данные появились, их следует вычитать и в OutputBuf вернётся 1. Если равны, то новых данных нет и в OutputBuf вернётся 0.
*См. Раздел 5.1.2 документа "*Руководство по программированию mPCIe-TTCAN*".

Примечание:
Не доступна в режиме "CanonicalDMA". Воспользуйтесь вызовом IOCTL_TTCANDEV_DISABLE_CANONICAL_DMA.

Вход:
–

Выход – TTCAN_READ_DMA_STATUS:

Смещение	Размер	Назначение	Примечание
0x00	4	*m_nStatus*: Статус новых данных	

Пример вызова:

```
TTCAN_READ_DMA_STATUS OutputBuf;
if(::DeviceIoControl(
  m_hDevice,
IOCTL_TTCANDEV_READ_DMA_STATUS,
  NULL, NULL,
  &OutputBuf, sizeof(OutputBuf),
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()
```

5.14. IOCTL_TTCANDEV_READ_REG

Назначение:
Чтение данных из регистрового пространства устройства (BAR).

Действие:
Функция читает данные из желаемого адреса регистрового пространства устройства (BAR) в переменную pData.

Примечание:
Рекомендуется использовать если данные меньше, либо равны 32 битам. Если больше 32 бит, то рекомендуется использовать функцию IOCTL_TTCANDEV_READ_BAR.

Вход – структура TTCAN_READ_REG_IN:

Смещение	Размер	Назначение	Примечание
0x00	*4*	*m_nAddress*: Адрес регистра	

Выход :

Смещение	Размер	Назначение	Примечание
0x00	4	Считанный блок данных	

Пример вызова:

```
TTCAN_READ_REG_IN InputBuf;
InputBuf.m_nAdress = nAddr;

if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_READ_REG,
  &InputBuf, sizeof(InputBuf),
  &pData, sizeof(pData),
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()
```

5.15. IOCTL_TTCAN_REGISTER_INTERRUPT

Назначение:

Разрешение и ожидание прерывания m_nInt.

Действие:

Приложение передаёт драйверу событие (event) m_hEvt и информацию об ожидаемом драйвере.

Драйвер разрешает прерывание m_nInt и ожидает его.

Для разрешения прерывания функция записывает данные либо только в регистр INTERRUPT MASK* (адрес 1010h), либо и в регистр INTERRUPT MASK и в регистр CANINTE** (адрес 2Bh).

Как только ожидаемое прерывание обрабатывается драйвером, hEvent переводится в сигнальное состояние.

* см. Раздел 5.1.4 документа "*Руководство по программированию модуля " mPCIe-TTCAN* ".

** см. Раздел 6.4.1 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

Примечание:

Варианты m_nInt:

- **HDAT** – прерывание по заполнению 1/2 буфера данных;
- **QDAT** – прерывание по заполнению 1/8 буфера данных;
- **TIM_CAN1** – Прерывание таймера CAN1. Порог срабатывания указывается в регистре CAN1_INT_TRIG*;
- **TIM_CAN2** – Прерывание таймера CAN2. Порог срабатывания указывается в регистре CAN2_INT_TRIG*;
- **RX0IE** – прерывание по получению сообщения в буфер приёма 0**;
- **RX1IE** – прерывание по получению сообщения в буфер приёма 1**;
- **TX0IE** – прерывание по отправке сообщения из буфера передачи 0;
- **TX1IE** – прерывание по отправке сообщения из буфера передачи 1;
- **TX2IE** – прерывание по отправке сообщения из буфера передачи 2;
- **ERRIE** – прерывание по событию ошибки шины или переполнению буферов (конкретный источник - в регистре EFLG);
- **WAKIE** – прерывание по событию выхода из режима сна;
- **MERRE** – прерывание по ошибке приёма либо передачи сообщения.

* см. Раздел 5.3.5 документа "*Руководство по программированию модуля " mPCIe-TTCAN* ".

** Прерывания RX0IF, RX1IF автоматически сбрасываются контроллером DMA сразу после вычитывания данных из буфера контроллера CAN и копирования их в память ПК.

Вход – структура TTCAN_INT_NTFCN_IN:

Смещение	Размер	Назначение	Примечание
0x00	4	*m_nCh:* номер канала. Для прерываний HDAT, QDAT, TIM_CAN1 и TIM_CAN2 не используется. Для остальных прерываний должен быть равен 1 или 2.	
0x04	4	*m_nInt:* номер прерывания	См. п. примечание
0x08	4	*m_hEvt:* Event для ожидания прерывания	

Выход:

–

Пример вызова на следующей странице:

Пример вызова:

```
TTCAN_INT_NTFCN_IN InputBuf;
InputBuf.m_nInt = HDAT;
InputBuf.m_hEvt = hEvent;

HANDLE Event= CreateEvent(NULL, TRUE, FALSE, name);

if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_REGISTER_INTERRUPT,
  &InputBuf, sizeof(InputBuf),
  NULL, NULL,
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()
```

5.16. IOCTL_TTCANDEV_WRITE_BAR

Назначение:
Запись данных в регистровое пространство устройства (BAR).

Действие:
Функция записывает необходимое количество данных по желаемому адресу в регистровое пространство устройства (BAR).

Примечание:
Рекомендуется использовать если данные больше 32 бит. Если меньше или равны 32 битам, то рекомендуется использовать функцию IOCTL_TTCANDEV_WRITE_REG.

Вход – структура TTCAN_WRITE_BAR_IN:

Смещение	Размер	Назначение	Примечание
0x00	4	*m_nOffset*: Смещение блока данных на запись	В байтах
0x04	4	*m_nSize*: Размер блока данных	В байтах
0x08	*m_nSize*	Блок данных на запись	

Выход:
–

Пример вызова:

```
TTCAN_WRITE_BAR_IN nInputSize;
pInput->m_nOffset = nAddr;
pInput->m_nSize = nSize;
pInput->m_nBar = nBar;
pInput->m_Data = Data;

if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_WRITE_BAR
  &InputBuf, sizeof(InputBuf),
  NULL, NULL,
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()
```

5.17. IOCTL_TTCANDEV_WRITE_REG

Назначение:
Запись данных в регистровое пространство устройства (BAR).

Действие:
Функция записывает данные по желаемому адресу в регистровое пространство устройства (BAR).

Примечание:
Рекомендуется использовать если данные меньше, либо равны 32 битам. Если больше 32 бит, то рекомендуется использовать функцию IOCTL_TTCANDEV_WRITE_BAR.

Вход – структура TTCAN_READ_BAR_IN:

Смещение	Размер	Назначение	Примечание
0x00	4	*m_nAddress:* адрес регистра	
0x04	4	*m_nData:* Записываемый блок данных	В байтах

Выход:
–

Пример вызова:

```
TTCAN_WRITE_REG_IN InputBuf;

if(::DeviceIoControl(
  m_hDevice,
  IOCTL_TTCANDEV_WRITE_REG,
  &InputBuf, sizeof(InputBuf),
  NULL, NULL,
  &lpBytesReturned,
  xxx))
else
  dw = GetLastError()
```

6. Обновление драйвера

Версия драйвера	Дата	Изменение
1.0	18.09.2015	Драйвер создан.

7. Обновление руководства

Версия документа	Дата	Изменение
1	18.09.2015	Документ создан.