**Руководство (v1.0)**

# По работе с драйвером модуля “mPCIe – TTCAN”

Интерфейс ISO-11898
(CAN Bus)

Для драйверов версии 1.0 и ниже

**ОС LINUX**


**14.10.2015**

**ООО “Новомар” 2015 г.**

# Оглавление

# 1. Введение.

Драйвер поддерживает модуль “**mPCIe-TTCAN**”.

Данный драйвер поддерживает одновременную работу не более 100 устройств и присваивает им уникальные символьные имена вида “ttcan_dev_x”, где x — индекс устройства, начиная с 0.

Каждый модуль представляет отдельный файл устройства в файловой системе и отображается в папке “/dev”.

Взаимодействие с драйвером происходит посредством ioctl-команд, перечень которых находится в файле “ioctl_drv_ttcan.h”.

Обращение к независимым каналам платы осуществляется по номеру канала (0 и 1).

# 2. Установка драйвера.

1. Создайте папку /modules в корне файловой системы.
2. Поместите в созданную папку каталог mPCIe-TTCAN_drv, содержащую файлы drv_ttcan.ko, version.txt и ioctl_drv_ttcan.h.
3. Откройте файл /etc/rc.local текстовым редактором.
4. Добавьте в открытый файл перед строкой “exit 0” строку “insmod /modules/mPCIe-TTCAN_drv/drv_ttcan.ko”.
5. Перезагрузите компьютер.

**Если Вы хотите проверить, загружен ли драйвер в ядро:**

1. В терминале введите команду “lsmod”.
2. Найдите в выведенном списке “drv_ttcan”.
3. Если есть — все нормально, если нет — драйвер не работает.

**Для обновления версии драйвера:**

1. Посмотреть версию драйвера (прикладывается в файле version.txt в папке с драйвером).
2. Сравнить с версией текущего драйвера (файл /modules/can_drv/version.txt).
3. Выбрать более позднюю.
4. Заменить файлы drv_ttcan.ko, ioctl_drv_ttcan.h и version.txt в папке /modules mPCIe-TTCAN_drv на новые (сохраняя имена!).
5. Перезагрузить компьютер.

# 3. Подключение файла с командами к разрабатываемому проекту.

1. Скопировать файл “ioctl_drv_ttcan.h” из папки /modules/mPCIe-TTCAN_drv в папку с проектом.
2. В начале проекта добавить строку:
   #include «ioctl_drv_ttcan.h»
3. Использовать команды.

Формат описан ниже; примеры использования приложены.

**ВНИМАНИЕ**

Системная функция **open** открывает модуль “mPCIe-TTCAN” целиком.

Если вызов драйвера не требует номер канала, значит, драйвер работает с модулем как с единым целым устройством.

Если вызов запрашивает номер канала, значит, драйвер работает только с одним конкретным каналом связи.

# 4. Список доступных команд по версиям драйверов.

| Название вызова | Краткое описание |
|---|---|
| Список вызовов, доступных **в драйвере версии до 1.0** | |
| IOCTL_WR_MAINREG_TTCAN | Запись регистров модуля |
| IOCTL_RD_MAINREG_TTCAN | Чтение регистров модуля |
| IOCTL_WR_CANREG_TTCAN | Запись регистров контроллера |
| IOCTL_RD_CANREG_TTCAN | Чтение регистров контроллера |
| IOCTL_VERSION_TTCAN | Информация о плате |
| IOCTL_ENABLE_DMA_TTCAN | Разрешение работы DMA |
| IOCTL_DISABLE_DMA_TTCAN | Запрет работы DMA |
| IOCTL_CLEAR_DMA_TTCAN | Сброс указателя DMA |
| IOCTL_ENABLE_CANONICAL_DMA_TTCAN | Включение режима “CANONICAL” работы с DMA |
| IOCTL_DISABLE_CANONICAL_DMA_TTCAN | Выключение режима “ CANONICAL ” работы с DMA. |
| IOCTL_RESET_TG_TTCAN | Сброс триггера. |
| IOCTL_SET_MODE_TTCAN | Установка режима работы канала |
| IOCTL_GET_MODE_TTCAN | Чтение режима работы канала |
| IOCTL_SET_SPEED_TTCAN | Установка скорости работы канала |
| IOCTL_SET_MASKS_TTCAN | Установка масок и фильтров на прием сообщений. |
| IOCTL_SET_CANn_TIMER_TRSH_TTCAN | Запуск таймера. |
| IOCTL_GET_NBLOCK_RAW_DMA_TTCAN | Кол-во новых блоков данных в DMA |
| IOCTL_RD_RAW_DMA_TTCAN | Чтение данных из DMA |
| IOCTL_RD_CH_RAW_DMA_TTCAN | Поканальное чтение данных из DMA |
| IOCTL_WRITE_DATA_TO_TR_BUF_TTCAN | Запись данных в буфер отправки. |
| IOCTL_SEND_DATA_NOW_TTCAN | Запуск передачи данных из буфера передачи. |
| IOCTL_CHECK_TRANSMIT_TTCAN | Проверка правильности отправки сообщения. |
| IOCTL_SEND_DATA_TG_TTCAN | Запуск однократной передачи данных из буфера передачи по значению триггера. |
| IOCTL_SEND_DATA_LOOP_TTCAN | Запуск зацикленной передачи данных из буфера передачи по значению таймера. |
| IOCTL_CHECK_TG_TTCAN | Проверка триггера. |
| IOCTL_ENABLE_INT_HDAT_TTCAN | Включение прерывания HDAT |
| IOCTL_ENABLE_INT_QDAT_TTCAN | Включение прерывания QDAT |

| | |
|---|---|
| IOCTL_ENABLE_INT_CAN1_TTCAN | Включение прерывания контроллера CAN1 |
| IOCTL_ENABLE_INT_CAN2_TTCAN | Включение прерывания контроллера CAN2 |
| IOCTL_ENABLE_INT_TIM_CAN1_TTCAN | Включение прерывания таймера CAN1 |
| IOCTL_ENABLE_INT_TIM_CAN2_TTCAN | Включение прерывания таймера CAN2 |
| IOCTL_DISABLE_ALL_INT_TTCAN | Отключить все прерывания. |

# 5. Описание использования команд драйвера.

Структура ioctl-команды:

***int ioctl(int fd, IOCTL_..., unsigned long param),***
где:
int fd – дескриптор файла, полученный при вызове функции open для файла устройства;
IOCTL_... – команда из набора, описанного в файле ioctl_drv_ttcan.h;
unsigned long param – параметр, передаваемый с командой. Содержимое параметра зависит от команды (см. описание команд ниже).

Команда, в случае удачного выполнения запроса, возвращает 0.
В случае неудачного выполнения запроса возвращается значение, отличное от 0, что означает:
1. запрос был отклонен ОС (-EBUSY и т.п.);
2. –EACCES - запрос не был выполнен драйвером (команда отсутствует, не может быть выполнена в режиме CANONICAL, некорректные параметры для данной платы/команды).

Примеры вызовов всех функций также есть в прикреплённом файле “**mPCIe_TTCAN_soft.c**”.
Пример работы прерываний находится в прикреплённом файле “ **test_interrupt.c** ”.

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

## 5.1 Режим Canonical

При загрузке драйвера данный режим выключен.
Режим реализован программно в драйвере.
В данном режиме недоступны команды (см. описание команд ниже), способные оказать деструктивное влияние на работу в данном режиме.
В режиме CANONICAL обеспечивается независимое чтение информации из буфера DMA по каналам (режим включается/выключается для всех каналов платы одновременно):

- чтение полученной информации с одного канала платы не оказывает влияние на другие каналы;
- при чтении с канала пользователь получает данные только этого канала.

При включении/выключении режима указатель буфера DMA платы обнуляется.

# 6. Стандартные функции

## 6.1 IOCTL_WR_MAINREG_TTCAN

**Назначение:**

Запись данных в регистровое пространство устройства (BAR) (но не контроллеров!).

**Действие:**

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

**Примечание:**

-

**Входные параметры:**

param – структура типа SADDR_DATA_MAIN_TTCAN:

| Поле структуры | Описание |
|---|---|
| unsigned int **daddr** | Адрес регистра |
| unsigned int **data** | Данные для записи |

**Пример вызова:**

```
SADDR_DATA_MAIN_TTCAN sad;
i = open("/dev/ttcan_dev_0", O_RDWR);
sad.daddr = 0x2000;
sad.data = 0;
retval = ioctl(i, IOCTL_WR_MAINREG_TTCAN, &add1);
if (retval == 0)
  printf("Data 0x%x was written to 0x%x reg\n", add1.data, add1.daddr);
else
  printf("ERROR Written main reg %d \n", retval);
```

## 6.2 IOCTL_RD_MAINREG_TTCAN

**Назначение:**
Чтение данных из регистрового пространства устройства (BAR) (но не контроллеров!).

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

**Примечание:**
-

**Входные параметры:**
param – структура типа SADDR_DATA_MAIN_TTCAN

| Поле структуры | Описание |
|---|---|
| unsigned int **daddr** | идентификатор модуля |
| unsigned int **data** | Не используется |

**Пример вызова:**

```
SADDR_DATA_MAIN_TTCAN add1
i = open("/dev/ttcan_dev_0", O_RDWR);
add1.daddr = 0x2000;
retval = ioctl(i, IOCTL_RD_MAINREG_TTCAN, &add1);
if (retval == 0)
  printf("From 0x%x reg was read 0x%x value\n", add1.daddr, add1.data);
else
  printf("ERROR Reading main reg %d \n", retval);
```

## 6.3 IOCTL_WR_CANREG_TTCAN

**Назначение:**

Запись данных в регистровое пространство контроллера CAN. Размер данных не должен превышать 16 байт.

**Действие:**

Функция записывает данные в **CAN$n^{(*)}$_Buf****. После чего записывает команду в регистр **CAN$n^{(*)}$_ACS***** и дождидается её выполнения.

*n – номер контроллера.

**

См. Раздел 6.1.4 или 6.1.6 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

***

См. Раздел 6.1.3 или 6.1.5 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**

-

**Входные параметры:**

param – структура типа SADDR_DATA_TTCAN:

| Поле структуры | Описание |
|---|---|
| char **daddr** | Адрес регистра |
| char **data** | Данные для записи |
| int **channel** | Номер канала |

**Пример вызова:**

```
SADDR_DATA_TTCAN add1;
i = open("/dev/ttcan_dev_0", O_RDWR);
add1.daddr = CANINTE;
add1.data = 0x0;
add1.channel = 1;
retval = ioctl(i, IOCTL_WR_CANREG_TTCAN, &add1);
if (retval == 0)
  printf("Data 0x%x was written to 0x%x reg of %d ch\n",
add1.data, add1.daddr, add1.channel);
else
  printf("ERROR Written can reg %d \n", retval);
```

## 6.4 IOCTL_RD_CANREG_TTCAN

**Назначение:**

Чтение данных из регистрового пространства контроллера CAN.

**Действие:**

Функция записывает команду в регистр **CAN*n*(*)_ACS**** и дождидается её выполнения. После этого вычитанные данные из **CAN*n*(*)_Buf***** копируются в переменную pDataBuf.

*n – номер контроллера.

** См. Раздел 6.1.3 или 6.1.5 документа "*Руководство по программированию модуля "mPCIe-TTCAN*" в зависимости от выбранного контроллера.

*** См. Раздел 6.1.4 или 6.1.6 документа "*Руководство по программированию модуля "mPCIe-TTCAN*" в зависимости от выбранного контроллера.

**Примечание:**

-

**Входные параметры:**

param – структура типа SADDR_DATA_TTCAN:

| Поле структуры | Описание |
|---|---|
| char **daddr** | Адрес регистра |
| char **data** | Не используется |
| int **channel** | Номер канала |

**Пример вызова:**

```
SADDR_DATA_TTCAN sad;
i = open("/dev/ttcan_dev_0", O_RDWR);
sad.daddr = CAN_CTRL;
sad.channel = 0;
retval = ioctl(i, IOCTL_RD_CANREG_TTCAN, &sad);
if (retval == 0)
  printf("From %d ch 0x%x reg was read 0x%x value\n",
add1.channel, add1.daddr, add1.data);
else
  printf("ERROR Written can reg %d \n", retval);
```

## 6.5 IOCTL_VERSION_TTCAN

**Назначение:**
Чтение информации о модуле.

**Действие:**
Функция заполняет все поля структуры **VERSION_TTCAN**.

**Примечание:**
-

**Входные параметры:**
param – указатель на структуру типа VERSION_TTCAN:

| Поле структуры | Описание |
|---|---|
| unsigned int **device_id** | идентификатор модуля |
| unsigned int **vendor_id** | идентификатор производителя |
| char **revision** | номер ревизии модуля |
| char **dev_name[30]** | имя модуля, назначенное драйвером |
| int **minor** | порядковый номер модуля, присвоенный драйвером |
| char **irq** | номер линии прерываний |
| long **size_dma** | размер буфера DMA |
| void***addr_dma_virt** | адрес буфера DMA |
| long **pci_bars** | номер регистровго пространства (BAR) |

**Пример вызова:**

```
VERSION_TTCAN cVer;
retval = ioctl(i, IOCTL_VERSION_TTCAN, &cVer);
if (retval == 0)
  printf("Device id = 0x%x\nVendor id = 0x%x\nRevision = 0x%x\nDriver
name = %s\nMinor = %d\n", cVer.device_id, cVer.vendor_id,
cVer.revision, cVer.dev_name, cVer.minor);
else{
  printf("ERROR reading device version %d \n", retval);
  return false;
}
```

# 7. Функции конфигурации

## 7.1 IOCTL_ENABLE_DMA_TTCAN

**Назначение:**
Разрешение работы DMA.

**Входные параметры:**

Нулевой бит регистра **DMA_DATA_BASE*** (адрес 1000h) и биты BnBFM, BnBFE и BnBFS регистра **BFPCTRL**** (адрес 0Ch) устанавливается в единицу.
*См. Раздел 5.1.1 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**
После загрузки операционной системы работа DMA не разрешена.

Входные данные

–

**Пример вызова:**

```
i = open("/dev/ttcan_dev_0", O_RDWR);
retval = ioctl(i, IOCTL_ENABLE_DMA_TTCAN, 0);
if (retval == 0)
  printf("DMA work was enabled\n");

else{
  printf("ERROR enable dma\n");
  return false;
}
```

## 7.2 IOCTL_DISABLE_DMA_TTCAN

**Назначение:**
Запрет работы DMA.

**Действие:**
Нулевой бит регистра **DMA_DATA_BASE*** (адрес 1000h) и биты BnBFM, BnBFE и BnBFS регистра **BFPCTRL**** (адрес 0Ch) сбрасываются в ноль.
*См. Раздел 5.1.1 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**
После загрузки операционной системы работа не разрешена.

Входные данные

–

**Пример вызова:**

```
i = open("/dev/ttcan_dev_0", O_RDWR);
retval = ioctl(i, IOCTL_DISABLE_DMA_TTCAN, 0);
if (retval == 0)
  printf("DMA work was enabled\n");

else{
  printf("ERROR enable dma\n");
  return false;
}
```

## 7.3 IOCTL_CLEAR_DMA_TTCAN

**Назначение:**

По вызову данной функции программный счётчик вычитанных пакетов данных DMA уравнивается с регистром **DMA_INDEX***(адрес 1008h).

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

**Действие:**

-

**Примечание:**

**ВНИМАНИЕ!!!**

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

Входные данные

-

**Пример вызова:**

```
i = open("/dev/ttcan_dev_0", O_RDWR);
retval = ioctl(i, IOCTL_CLEAR_DMA_TTCAN, 0);
if (retval == 0)
  printf("DMA counts was reset\n");
else{
  printf("ERROR enable dma\n");
  return false;
}
```

## 7.4 IOCTL_ENABLE_CANONICAL_DMA_TTCAN

**Назначение:**

Включение режима “CanonicalDMA”* работы с DMA.
*см. раздел 5.1 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 соответствующему каналу.
Включается режим “Canonical”, 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 модуля “mPCIe-TTCAN*”.

**Примечание:**

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

**Входные параметры:**

-

**Пример вызова:**

```
i = open("/dev/ttcan_dev_0", O_RDWR);
retval = ioctl(i, IOCTL_ENABLE_CANONICAL_DMA_TTCAN, 0);
if (retval == 0)
  printf("Canonical mode was enabled\n");
else{
  printf("ERROR  setting Canonical mode\n");
  return false;
}
```

## 7.5 IOCTL_DISABLE_CANONICAL_DMA_TTCAN

**Назначение:**

Выключение режима “Canonical”* работы с DMA.
*см. раздел 5.1 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 “Canonical”, 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 модуля “mPCIe-TTCAN*”.

**Примечание:**

Для того чтобы понять работает какая-либо функция без данного режима или нет, смотрите раздел “примечание” просматриваемой функции. Если про данный режим ничего не сказано, значит функция может работать как с данным режимом, так и без него.

**Входные параметры:**

–

**Пример вызова:**

```
fd = open("/dev/ttcan_dev_0", O_RDWR);
retval = ioctl(fd, IOCTL_DISABLE_CANONICAL_DMA_TTCAN,
0);
if (retval == 0)
  printf("Canonical mode was disabled\n");
else{
  printf("ERROR disable Canonical mode\n");
  return false;
}
```

## 7.6 IOCTL_RESET_TG_TTCAN

**Назначение:**
Сброс триггера.

**Действие:**
В биты **TX_TRIG*n*(*)_EN** регистра **CAN*m*(**)_TRIG_CTRL*** записывается значение “10”.
*n – номер триггера
**m – номер канала.
*** См. Раздел 5.3.3 или 5.3.4 документа “*Руководство по программированию модуля “mPCIe-TTCAN*” в зависимости от выбранного контроллера.

**Примечание:**
-

**Входные параметры:**
param – структура типа CH_BUF_TTCAN:

| Поле структуры | Описание |
|---|---|
| uint8 **nCh** | Номер канала |
| uint8 **nBuf** | Номер буфера (равен номеру триггера) |

**Пример вызова:**

```
CH_BUF_TTCAN cCb;
fd = open("/dev/ttcan_dev_0", O_RDWR);
cCb.nCh=1;
cCb.nBuf=2;
retval = ioctl(fd, IOCTL_RESET_TG_TTCAN, &cCb);
if (retval == 0)
  printf("Transmit loop was disabled on %d ch\n", ch);
else{
  printf("ERROR disabling transmit loop - %d\n", retval);
  return false;
}
```

## 7.7 IOCTL_SET_MODE_TTCAN

**Назначение:**

Установка режима работы контроллера.

См. Раздел 6.3 документа "*Руководство по программированию модуля "mPCIe-TTCAN*"

**Действие:**

Значение из переменной mode записывается с 5 по 7 биты регистра **CAN_CTRL*** (адрес Fh). После этого следует проверить, установился ли данный режим работы. Для этого следует воспользоваться вызовом IOCTL_GET_MODE_TTCAN.

*См. Раздел 6.3.1 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**

Будьте внимательны при выставлении режима сна.

Внимательно прочитайте какие действия нужно совершить для перевода устройства в режим сна и для вывода устройства из этого режима в разделе 6.3 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Входные параметры:**

param – структура типа CONF_TTCAN:

| Поле структуры | Описание |
|---|---|
| int **channel** | Номер контроллера CAN |
| char **mode** | Режим работы контроллера который вы хотите задать |

**Пример вызова:**

```
CONF_TTCAN conf;
fd = open("/dev/ttcan_dev_0", O_RDWR);
conf.channel = 0;
conf.mode = CAN_WORK; // CAN_CONF, CAN_WORK,
CAN_MON, CAN_SLEEP, CAN_LOOP
if (ioctl(fd,  IOCTL_SET_MODE_mPCIe_CAN, &conf)) {
//ошибка
}
else {
// Канал 0 установлен в режим CAN_WORK
}
```

## 7.8 IOCTL_GET_MODE_TTCAN

**Назначение:**
Чтение режима работы контроллера.

**Действие:**
Читает значение битов с 5 по 7 регистра **CAN_STAT*** (адрес Eh) в переменную mode.
* См. Раздел 6.3.2 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**
-

**Входные параметры:**
param – структура типа CONF_TTCAN:

| Поле структуры | Описание |
|---|---|
| int **channel** | Номер контроллера CAN |
| char **mode** | Режим работы контроллера который вы хотите задать |

**Пример вызова:**

```
CONF_TTCAN conf;
fd = open("/dev/ttcan_dev_0", O_RDWR);
conf.channel = 0;
if (ioctl(fd, IOCTL_GET_MODE_TTCAN, &conf)) {
//ошибка
}
else {
// в переменной conf.mode находится режим работы канала
}
```

## 7.9 IOCTL_SET_SPEED_TTCAN

**Назначение:**
Функция конфигурации скорости шины CAN.

**Действие:**
В зависимости от значение скорости, определённые значения записываются в регистры **CNF1***(адрес 2ah), **CNF2****(адрес 29h) и **CNF3***** (адрес 28h) выбранного канала.
*См. раздел 6.5.1 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".
**См. раздел 6.5.2 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".
***См. раздел 6.5.3 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**
*До вызова этой функции следует обязательно выставить режим работы контроллера в режим конфигурации.*

**Входные параметры:**
param – структура типа SPEED_TTCAN:

| Поле структуры | Описание |
|---|---|
| int **channel** | Номер канала |
| unsigned int **speed** | Значение скорости |

**Пример вызова:**

```
SPEED_TTCAN cSpeed;
fd = open("/dev/ttcan_dev_0", O_RDWR);
sp.nCh = 0;
sp.speed = WORK_SPEED_125; //WORK_SPEED_125;
WORK_SPEED_250; WORK_SPEED_500; WORK_SPEED_1000
retval = ioctl(fd, IOCTL_SET_SPEED_TTCAN, &cSpeed);
if (retval == 0){
  printf("Speed %d was set on %d ch\n", cSpeed.speed,
cSpeed.channel);
}
else{
  printf("ERROR Set speed %d ch %d \n", cSpeed.channel, retval);
  return false;
}
```

## 7.10 IOCTL_SET_MASKS_TTCAN

**Назначение:**
Установка масок и фильтров на прием сообщений.

**Действие:**
В зависимости от выбранного канала (**channel**) и номера буфера (**buf_num**) функция записывает значение mode_buf в регистр **RXB*n****CTRL**** с 0 бита.

*n – номер буфера.

**См. раздел 6.8.1 и 6.8.2 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

Далее в зависимости от выбранного идентификатора (**ident**) и номера фильтра (**filtr_num**) функция записывает значения масок и фильтров в соответствующие регистры*.

*См. раздел 6.9 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**
Для отключения фильтров и масок переменная **mode_buf** должна быть равна значению 0x60. В этом случае переменные **filtr_num**, **ident**, **identific**, **identific_f** игнорируются и модуль будет принимать из линии все сообщения.

**Входные параметры:**
param – структура типа MASKS_TTCAN:

| Поле структуры | Описание |
|---|---|
| int **channel** | номер канала |
| int **buf_num** | номер буфера приема |
| char **mode_buf** | режим работы буфера |
| int **ident** | 0 – стандартный ID; 1 – расширенный ID |
| unsigned int **identific** | идентификатор маски; 10-0 биты – стандартный ID; 28-11 биты – расширенный ID |
| unsigned int **identific_f** | идентификатор фильтра; 10-0 биты – стандартный ID; 28-11 биты – расширенный ID |
| int **filter_num** | номер фильтра (0, 1, 2, 3, 4 или 5). Первые три фильтра (0, 1, 2) – 0 буфер, 0 маска; следующие три фильтра (3, 4, 5) – 1 буфер, 1 маска |

**Пример вызова:**

```
MASKS_TTCAN cMask;
fd = open("/dev/ttcan_dev_0", O_RDWR);
cMask.channel=nCh;
cMask.buf_num=0;
cMask.mode_buf=0x60; *см. примечание

retval = ioctl(i, IOCTL_SET_MASKS_TTCAN, &cMask);
if (retval == 0)
  printf("Filters and masks was disabled\n");
else{
  printf("ERROR disabling filters and masks. err = %d\n", retval);
  return false;
}
```

## 7.11 IOCTL_SET_CANn_TIMER_TRSH_TTCAN

**Назначение:**
Запуск таймера.

**Действие:**
Значение переменной **nValue**, записывается в регистр **CAN*n*(*)_TIMER_TRSH**** (значение записывается целиком с нулевого бита).

Биты RST, ENABLE и NTU_MODE регистра **CAN*n*(*)_TIMER_CTRL***** устанавливаются в единицу. Значения остальных битов данного регистра не изменяется.

*n – номер канала
**См. раздел 5.2.2 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".
***См. раздел 5.2.4 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**
-

**Входные параметры:**
param – структура типа TIMER_TRSH_TTCAN:

| Поле структуры | Описание |
|---|---|
| uint8 **nCh** | номер канала |
| int **nValue** | номер буфера приема |

**Пример вызова:**

```
TIMER_TRSH_TTCAN cTt;
fd = open("/dev/ttcan_dev_0", O_RDWR);
cTt.nCh = ch;
cTt.nValue = 0x10FFFC;
retval = ioctl(fd, IOCTL_SET_CANn_TIMER_TRSH_TTCAN, &cTt);
if (retval == 0)
  printf("CAN%d_TIMER_TRSH has been set\n", ch);
else{
  printf("ERROR setting CAN%d_TIMER_TRSH - %d\n", ch, retval);
  return false;
}
```

# 8. Функции для чтения принятых данных

## 8.1 IOCTL_GET_NBLOCK_RAW_DMA_TTCAN

**Назначение:**

Чтение количества новых блоков данных, накопленных в буфере DMA устройства по всем каналам (блок - 16 байт, но в буфере данных блоки лежат с выравниванием в 64 байта).

**Действие:**

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

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

**Входные параметры:**

param – переменная типа unsigned int, в которую будет помещено кол-во блоков

**Примечание:**

*Не доступна в режиме "Canonical"*

**Пример вызова:**

```
unsigned int nb;
fd = open("/dev/ttcan_dev_0", O_RDWR);
if (ioctl(fd,  IOCTL_GET_NBLOCK_RAW_DMA_TTCAN, &nb)) {
//ошибка
}
else {
// nb = кол-во непрочитанных блоков в буфере DMA
}
```

## 8.2 IOCTL_RD_RAW_DMA_TTCAN

**Назначение:**
Чтение данных из буфера DMA. Данная функция читает любое требуемое количество новых блоков данных DMA(блок - 16 байт, но в буфере данных блоки лежат с выравниванием в 64 байта).

**Действие:**
Чтобы воспользоваться функцией, рекомендуется узнать количество новых блоков данных с помощью функции IOCTL_GET_NBLOCK_RAW_DMA_TTCAN. Значение возвращенное этой функцией, использовать в качестве входного параметра для переменной **DMA_STR_TTCAN.nBlocks**.
После выполнения функции в переменной nBlocks записывается количество прочитанных блоков (может быть не равно меньше запрашиваемого) и указатель буфера DMA сдвигается на соответствующую количеству прочитанных данных позицию.

**Примечание:**
*Не доступна в режиме "CanonicalDMA".*

**Входные параметры:**

param – структура:

| Поле структуры | Описание |
|---|---|
| unsigned int **nBlocks** | Кол-во запрашиваемых/полученных блоков |
| unsigned int **nCh** | Не используется |
| int **Data[x]** | Указатель на место памяти, куда будут записаны прочитанные данные |

**Пример вызова:**

```
struct {
  unsigned int nBlocks;
  unsigned int nCh;
  int b[16];        //память под один блок данных
} cData;
fd = open("/dev/ttcan_dev_0", O_RDWR);
cData.number_block = 1;
retval = ioctl(i, IOCTL_RD_RAW_DMA_TTCAN, &cData);
if (retval == 0){
  for (j=0; j<16; j++)
    printf("0x%x\n", cData.b[j]);
}
else{
  printf("ERROR reading new data\n");
  return false;
}
```

## 8.3 IOCTL_RD_CH_RAW_DMA_TTCAN

**Назначение:**

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

**Действие:**

Функция проверяет наличие в DMA новых блоков данных. Если они есть, то функция ищет в новых блоках данных, блоки полученные из запрашиваемого канала. При нахождении данных блоков, они складываются в переменную **Data** до тех пор, пока либо не наберётся запрашиваемое количество блоков, либо блоки с новыми данными не закончатся.

**Примечание:**

*Доступна только в режиме "Canonical".*

**Входные параметры:**

param – структура:

| Поле структуры | Описание |
|---|---|
| unsigned int **nBlocks** | Кол-во запрашиваемых/полученных блоков |
| unsigned int **nCh** | Номер канала |
| int **Data[x]** | Указатель на место памяти, куда будут записаны прочитанные данные |

**Пример вызова:**

```
struct {
  unsigned int number_block;
  unsigned int number_channel;
  int b[16];      //memory for 1 block
} cData;

i = open("/dev/ttcan_dev_0", O_RDWR);
cData.number_block = 1;
cData.number_channel = 1;

retval = ioctl(i, IOCTL_RD_CH_RAW_DMA_TTCAN, &cData);
if (retval == 0){
  if (cData.number_block != 0){
    for (j=0; j<16; j++)
      printf("0x%x\n", cData.b[j]);
  }
  else
    printf("No new data in DMA\n");
}
else{
  printf("ERROR reading new data (canonical)\n");
  return false;
}
```

# 9. Функции для передачи данных

В данном драйвере предусмотрено три режима передачи данных:

- 3 буфера передачи данных и 3 значения приоритета. В одном буфере могут быть сообщения только одного приоритета (вызов IOCTL_SEND_DATA_NOW_TTCAN);
- однократная передача данных по триггеру (вызов IOCTL_SEND_DATA_TG_TTCAN);
- передача данных в цикле по таймеру (вызов IOCTL_SEND_DATA_LOOP_TTCAN и IOCTL_SET_CANn_TIMER_TRSH_TTCAN).

До вызова функции отправки данных необходимо записать данные в желаемый буфер с помощью функции IOCTL_WRITE_DATA_TO_TR_BUF_TTCAN. Номера буферов в обеих функциях должны совпадать.

## 9.1 IOCTL_WRITE_DATA_TO_TR_BUF_TTCAN

**Назначение:**

Запись данных в буфер отправки.

**Действие:**

Данные из входных переменных последовательно записываются в регистры **TXB*n*$^{(*)}$SIDH**, **TXB*n*$^{(*)}$SIDL**, **TXB*n*$^{(*)}$EID8**, **TXB*n*$^{(*)}$EID0**, **TXB*n*$^{(*)}$DLC**, **TXB*n*$^{(*)}$Dm** (см. раздел 6.7.2-6.7.7 документа "*Руководство по программированию mPCIe-TTCAN*" в соответствии с выбранным номером буфера).

Значение EID равное -1 говорит функции, что расширенный идентификатор не используется.

*n – номер буфера.

**См. раздел 6.7.1 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**

-

**Входные параметры:**

param – структура WR_TR_BUF:

| Поле структуры | Описание |
|---|---|
| uint8 **nChannel** | Номер контроллера CAN |
| int **nBufNumber** | Номер буфера в который будут записываться данные в ожидании своей очереди для передачи |
| int **SID** | Стандартный индетификатор |
| int **EID** | Расширенный индетификатор |
| int **pData[2]** | Указатель на буфер с данными |
| int **nSize** | Размер передаваемых данных |

**Пример вызова на следующей странице:**

**Пример вызова:**

```
WR_TR_BUF cData;
i = open("/dev/ttcan_dev_0", O_RDWR);
cData.nChannel = ch;
cData.nBufNumber = buf;
cData.SID=SID;
cData.EID=EID;
cData.pData[0]= Data;
cData.nSize=Size;
retval = ioctl(i, IOCTL_WRITE_DATA_TO_TR_BUF_TTCAN,
&cData);
if (retval == 0)
  printf("Data was written to %d buffer\n", buf);
else{
  printf("ERROR writing data to transmit buffer - %d\n", retval);
  return false;
}
```

## 9.2 IOCTL_SEND_DATA_NOW_TTCAN

**Назначение:**
Запуск передачи данных из буфера передачи.

**Действие:**
Приоритет сообщения, находящийся в переменной **nPriority**, записывается в биты 0 и 1 регистра **TXB*n*(*)CTRL****, и также в единицу устанавливается третий бит данного регистра.

*n – номер буфера.
**См. раздел 6.7.1 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**
Значение приоритета (nPriority) должно совпадать с номером буфера (**nBufNumber**), в который были записаны данные с помощью вызова IOCTL_WRITE_DATA_TO_TR_BUF_TTCAN.

**Входные параметры:**
param – структура SEND_DATA_NOW:

| Поле структуры | Описание |
|---|---|
| uint8 nChannel | Номер контроллера CAN |
| uint8 nPriority | Приоритет с которым данные будут ждать передачу |

**Пример вызова:**

```
i = open("/dev/ttcan_dev_0", O_RDWR);
cSend.nChannel = 1;
cSend.nPriority = 0;
retval = ioctl(i, IOCTL_SEND_DATA_NOW_TTCAN, &cSend);
if (retval == 0){
  printf("Command to send data from %d buffer was written\n",
priority);
}
else{
  printf("ERROR writing command\n");
  return false;
}
```

## 9.3 IOCTL_CHECK_TRANSMIT_TTCAN

**Назначение:**
Проверка правильности отправки сообщения.

**Действие:**
Данная функция проверяет 6, 4 и 3 биты регистра **TXB*n*$^{(*)}$CTRL****.
*n – номер буфера отправки
** - См. Раздел 6.7.1 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**
Возвращаемые значения:
0 – Сообщение успешно отправлено;
1 – неверно указан номер канала (**nChannel**);
2 – не верно указан номер буфера(**nPriority**);
3- ошибка на шине;
4 – передача сообщения была прервана;
5 – сообщение проиграло арбитраж во время передачи.

**Входные параметры:**

param – структура SEND_DATA_NOW:

| Поле структуры | Описание |
|---|---|
| uint8 **nChannel** | Номер контроллера CAN |
| uint8 **nPriority** | Номер буфера(приоритета) из которого отправлялось сообщение |

**Пример вызова:**

```
i = open("/dev/ttcan_dev_0", O_RDWR);
cSend.nChannel = 1;
cSend.nPriority = 0;
for (j=0; j<1000; j++){
  retval = ioctl(i, IOCTL_CHECK_TRANSMIT_TTCAN, &cSend);
  if (retval == 0)
    break;
}
if (retval == 0){
  printf("Data was sucessfuly send from %d buffer\n", priority);
}
else{
  printf("ERROR sending data from %d buffer\n", priority);
  return false;
}
```

## 9.4 IOCTL_SEND_DATA_TG_TTCAN

**Назначение:**
Запуск однократной передачи данных из буфера передачи по значению триггера.

**Действие:**
Значение триггера, находящееся в переменной **nTrigger**, записывается в регистр **CAN** $n^{(*)}$**_TX** $m^{(**)}$**_TRIG** *** (значение записывается целиком с нулевого бита).

Значение триггера можно высчитать, прочитав значение таймера **CAN**$n^{(*)}$**_TIMER** и добавив к этому значению нужный период времени. Для увеличения значения на 1 сек нужно добавить к прочитанному значению единицу, начиная с 16 бита. Для увеличения на 2 сек – двойку и т.д.

*n – номер канала
**m – номер буфера.
***См. раздел 5.3.1-5.3.2 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

Также в биты **TX_TRIG**$n^{(*)}$**_EN** регистра **CAN**$m^{(**)}$**_TRIG_CTRL**** записывается значение 01.

*n – номер триггера
**m – номер канала.
***См. раздел 5.3.3-5.3.4 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**
Номер буфера (**nBufNumber**) должен совпадать с номером буфера (**nBufNumber**), в который были записаны данные с помощью функции IOCTL_WRITE_DATA_TO_TR_BUF_TTCAN.

Не забудьте разрешить отправку сообщения по триггерам (см. раздел 6.4.4 документа "*Руководство по программированию модуля "mPCIe-TTCAN*").

**Входные параметры:**

param – структура SEND_DATA_TG:

| Поле структуры | Описание |
|---|---|
| uint8 **nChannel** | Номер контроллера CAN |
| uint8 **nBuf** | Приоритет с которым данные будут ждать передачу |
| int **nTrigger** | Значение триггера |

**Пример вызова:**

```
SEND_DATA_TG cSend;
i = open("/dev/ttcan_dev_0", O_RDWR);
cSend.nChannel = 1;
cSend.nBuf = 2;
cSend.nTrigger = nTimer;
retval = ioctl(i, IOCTL_SEND_DATA_TG_TTCAN, &cSend);
if (retval == 0){
  printf("%d buffer's trigger was enabled\n", buf);
}
else{
  printf("ERROR writing command - %d\n", retval);
  return false;
}
```

## 9.5 IOCTL_SEND_DATA_LOOP_TTCAN

**Назначение:**

Запуск зацикленной передачи данных из буфера передачи по значению таймера.

**Действие:**

Значение таймера, находящееся в переменной **nTimer**, записывается в регистр **CAN** $n^{(*)}$**_TX** $m^{(**)}$**_TRIG*****(значение записывается целиком с нулевого бита).

*n – номер канала

**m – номер буфера.

***См. раздел 5.3.1-5.3.2 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

Также в биты **TX_TRIG**$n^{(*)}$**_EN** регистра **CAN**$m^{(**)}$**_TRIG_CTRL******** записывается значение 01 и бит **TX_TRIG**$n^{(*)}$**_RPT** устанавливается в единицу.

*n – номер триггера

**m – номер канала.

***См. раздел 5.3.3-5.3.4 документа "*Руководство по программированию модуля "mPCIe-TTCAN*".

**Примечание:**

Номер буфера (**nBufNumber**) должен совпадать с номером буфера (**nBufNumber**), в который были записаны данные с помощью функции IOCTL_WRITE_DATA_TO_TR_BUF_TTCAN.

Не забудьте разрешить отправку сообщения по триггерам (см. раздел 6.4.4 документа "*Руководство по программированию модуля "mPCIe-TTCAN*").

**Входные параметры:**

param – структура SEND_DATA_LOOP_TTCAN:

| Поле структуры | Описание |
|---|---|
| uint8 **nCh** | Номер контроллера CAN |
| uint8 **nBuf** | Приоритет с которым данные будут ждать передачу |
| int **nValue** | Значение таймера |

**Пример вызова:**

```
SEND_DATA_LOOP_TTCAN cSdl;
i = open("/dev/ttcan_dev_0", O_RDWR);
cSdl.nCh = ch;
cSdl.nBuf = buf;
cSdl.nValue = 100;
retval = ioctl(i, IOCTL_SEND_DATA_LOOP_TTCAN, &cSdl);
if (retval == 0)
  printf("Transmit loop was enabled on %d ch\n", ch);
else{
  printf("ERROR enabling transmit loop - %d\n", retval);
  return false;
}
```

## 9.6 IOCTL_CHECK_TG_TTCAN

**Назначение:**
Проверка триггера.

**Действие:**
Функция читает регистр **CAN*m*(**)_TRIG_CTRL***** и проверяет биты **TX_TRIG*n*(*)_EN**.
Если биты равны “01”, то триггер занят и функция вернёт значение 2.
Если биты равны “00”, то триггер свободен и функция вернёт значение 0.
*n – номер триггера.
**m – номер канала
***См. раздел 5.3.3-5.3.4 документа “*Руководство по программированию модуля “mPCIe-TTCAN*”.

**Примечание:**
-

**Входные параметры:**
param – структура CH_BUF_TTCAN:

| Поле структуры | Описание |
|---|---|
| uint8 **nCh** | Номер контроллера CAN |
| uint8 **nBuf** | Приоритет с которым данные будут ждать передачу |

**Пример вызова:**

```
CH_BUF_TTCAN cCB;
i = open("/dev/ttcan_dev_0", O_RDWR);
cCb.nCh=ch;
cCb.nBuf=buf;
for (a=0; a<4; a++){
  sleep(1);
  //Проверяем отправились ли данные
  if ((ioctl(i, IOCTL_CHECK_TG_TTCAN, cCB)) == 0){
    bFlag = true;
    break;
  }
}
```

# 10. Команды работы с прерываниями

Механизм прерываний основан на использовании сигналов. На каждое прерывание можно назначить индивидуальные сигнал и процесс. Примеры использования см. в файле **test_interrupt.c**.

Возможные прерывания:
**INT_QDAT_TTCAN** — заполнение 1/8 буфера DMA
**INT_HDAT_TTCAN** — заполнение 1/2 буфера DMA
**INT_CAN1_TTCAN** — прерывание контроллера CAN 1
**INT_CAN2_TTCAN** — прерывание контроллера CAN 2
**INT_TIM_CAN1_TTCAN** — прерывание таймера CAN 1
**INT_TIM_CAN2_TTCAN** — прерывание таймера CAN 2

События, по которым можно получать прерывания:
**RX01E** — буфер приема 0*
**RX11E** — буфер приема 1*
**TX01E** — буфер передачи 0
**TX11E** — буфер передачи 1
**TX21E** — буфер передачи 2
**ERRIR** — ошибка шины
**WAKIE** — прерывание по выходу из спящего режима
**MERRE** — ошибка сообщения

*Если у Вас включен механизм DMA, то прерывания **RX01E** и **RX11E** недоступны.

В вызовы следует передать Pid приложения, которое будет ожидать сигнал и сам сигнал. Для того, что бы узнать как работать с данными параметрами см. файл **test_interrupt.c**.

## 10.1 IOCTL_ENABLE_INT_HDAT_TTCAN

**Назначение:**
Включить ожидание сигнала по заполнению 1/2 буфера DMA.

**Действие:**
Драйвер разрешает прерывание и запоминает pid приложения, которое ожидает сигнал и сам сигнал, который нужно будет отправить по прерыванию.

Для разрешения прерывания функция устанавливает в единицу соответствующий бит регистра **INTERRUPT MASK*** (адрес 1010h). Другие биты данного регистра не изменяются.
* см. Раздел 5.1.4 документа "*Руководство по программированию модуля " mPCIe-TTCAN* ".

**Примечание:**
-

**Входные параметры:**
param – структура типа *SINTHIOSA_TTCAN*

| Поле структуры | Описание |
|---|---|
| uint8 intr | Не используется |
| int proc_pid | Pid процесса |
| int sig_no | Сигнал |

**Пример вызова:**

```
SINTHIOSA_TTCAN ss1;
ss1.proc_pid = pid;
ss1.sig_no = sig;
retval = ioctl(i, IOCTL_ENABLE_INT_HDAT_TTCAN, &ss1);
if (retval == 0){
  printf("signal set: HDAT enabled \n");
}
else{
  printf("error set signal and enabling HDAT %d \n", retval);
}
```

## 10.2 IOCTL_ENABLE_INT_QDAT_TTCAN

**Назначение:**
Включить ожидание сигнала по заполнению 1/8 буфера DMA.

**Действие:**
Драйвер разрешает прерывание и запоминает pid приложения, которое ожидает сигнал и сам сигнал, который нужно будет отправить по прерыванию.

Для разрешения прерывания функция устанавливает в единицу соответствующий бит регистра **INTERRUPT MASK*** (адрес 1010h). Другие биты данного регистра не изменяются.
* см. Раздел 5.1.4 документа "*Руководство по программированию модуля " mPCIe-TTCAN* ".

**Примечание:**
-

**Входные параметры:**
param – структура типа *SINTHIOSA_TTCAN*

| Поле структуры | Описание |
|---|---|
| uint8 intr | Не используется |
| int proc_pid | Pid процесса |
| int sig_no | Сигнал |

**Пример вызова:**

```
SINTHIOSA_TTCAN ss1;
ss1.proc_pid = pid;
ss1.sig_no = sig;
retval = ioctl(i, IOCTL_ENABLE_INT_QDAT_TTCAN, &ss1);
if (retval == 0){
  printf("signal set: QDAT enabled \n");
}
else{
  printf("error set signal and enabling QDAT %d \n", retval);
}
```

## 10.3 IOCTL_ENABLE_INT_CAN1_TTCAN

**Назначение:**

Включить ожидание сигнала по прерыванию контроллера CAN1.

**Действие:**

Драйвер разрешает прерывание и запоминает pid приложения, которое ожидает сигнал и сам сигнал, который нужно будет отправить по прерыванию.

Для разрешения прерывания функция устанавливает в единицу соответствующий бит регистра **INTERRUPT MASK*** (адрес 1010h). В зависимости от значения переменной intr, в единицу устанавливается соответствующий бит регистра, **CANINTE**** (адрес 2Bh). Другие биты данного регистра также не изменяются.

* 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:**

-

**Входные параметры:**

param – структура типа *SINTHIOSA_TTCAN*

| Поле структуры | Описание |
|---|---|
| uint8 **intr** | Прерывание контроллера CAN. (RX01E, RX11E, TX01E, TX11E, TX21E, ERRIR, WAKIE, MERRE) |
| int **proc_pid** | Pid процесса |
| int **sig_no** | Сигнал |

**Пример вызова:**

```
SINTHIOSA_TTCAN ss1;
ss1.proc_pid = pid;
ss1.sig_no = sig;
ss1.intr = TX01E;
retval = ioctl(i, IOCTL_ENABLE_INT_CAN1_TTCAN, &ss1);
if (retval == 0)
  printf("signal set: CAN1 enabled \n");
else
  printf("error set signal and enabling CAN1 %d \n", retval);
```

## 10.4 IOCTL_ENABLE_INT_CAN2_TTCAN

**Назначение:**

Включить ожидание сигнала по прерыванию контроллера CAN2.

**Действие:**

Драйвер разрешает прерывание и запоминает pid приложения, которое ожидает сигнал и сам сигнал, который нужно будет отправить по прерыванию.

Для разрешения прерывания функция устанавливает в единицу соответствующий бит регистра **INTERRUPT MASK*** (адрес 1010h). В зависимости от значения переменной intr, в единицу устанавливается соответствующий бит регистра, **CANINTE**** (адрес 2Bh). Другие биты данного регистра также не изменяются.

* 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:**

-

**Входные параметры:**

param – структура типа *SINTHIOSA_TTCAN*

| Поле структуры | Описание |
|---|---|
| uint8 **intr** | Прерывание контроллера CAN. (RX01E, RX11E, TX01E, TX11E, TX21E, ERRIR, WAKIE, MERRE) |
| int **proc_pid** | Pid процесса |
| int **sig_no** | Сигнал |

**Пример вызова:**

```
SINTHIOSA_TTCAN ss1;
ss1.proc_pid = pid;
ss1.sig_no = sig;
ss1.intr = TX01E;
retval = ioctl(i, IOCTL_ENABLE_INT_CAN2_TTCAN, &ss1);
if (retval == 0)
  printf("signal set: CAN2 enabled \n");
else
  printf("error set signal and enabling CAN2 %d \n", retval);
```

## 10.5 IOCTL_ENABLE_INT_TIM_CAN1_TTCAN

**Назначение:**

Включить ожидание сигнала по Прерыванию таймера CAN1. Порог срабатывания указывается в регистре **CAN1_INT_TRIG***.

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

**Действие:**

Драйвер разрешает прерывание и запоминает pid приложения, которое ожидает сигнал и сам сигнал, который нужно будет отправить по прерыванию.

Для разрешения прерывания функция устанавливает в единицу соответствующий бит регистра **INTERRUPT MASK*** (адрес 1010h). Другие биты данного регистра не изменяются.

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

**Примечание:**

-

**Входные параметры:**

param – структура типа *SINTHIOSA_TTCAN*

| Поле структуры | Описание |
|---|---|
| uint8 **intr** | Не используется |
| int **proc_pid** | Pid процесса |
| int **sig_no** | Сигнал |

**Пример вызова:**

```
SINTHIOSA_TTCAN ss1;
ss1.proc_pid = pid;
ss1.sig_no = sig;
retval = ioctl(i, IOCTL_ENABLE_INT_TIM_CAN1_TTCAN, &ss1);
if (retval == 0){
  printf("signal set: TIM_CAN_1 enabled \n");
}
else{
  printf("error set signal and enabling TIM_CAN_1 %d \n", retval);
}
```

## 10.6 IOCTL_ENABLE_INT_TIM_CAN2_TTCAN

**Назначение:**

Включить ожидание сигнала по Прерыванию таймера CAN2. Порог срабатывания указывается в регистре **CAN2_INT_TRIG***.

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

**Действие:**

Драйвер разрешает прерывание и запоминает pid приложения, которое ожидает сигнал и сам сигнал, который нужно будет отправить по прерыванию.

Для разрешения прерывания функция устанавливает в единицу соответствующий бит регистра **INTERRUPT MASK*** (адрес 1010h). Другие биты данного регистра не изменяются.

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

**Примечание:**

-

**Входные параметры:**

param – структура типа *SINTHIOSA_TTCAN*

| Поле структуры | Описание |
|---|---|
| uint8 **intr** | Не используется |
| int **proc_pid** | Pid процесса |
| int **sig_no** | Сигнал |

**Пример вызова:**

```
SINTHIOSA_TTCAN ss1;
ss1.proc_pid = pid;
ss1.sig_no = sig;
retval = ioctl(i, IOCTL_ENABLE_INT_TIM_CAN2_TTCAN, &ss1);
if (retval == 0){
  printf("signal set: TIM_CAN_2 enabled \n");
}
else{
  printf("error set signal and enabling TIM_CAN_2 %d \n", retval);
}
```

## 10.7 IOCTL_DISABLE_ALL_INT_TTCAN

**Назначение:**

Запрет всех прерываний.

**Действие:**

Все сохранённые в драйверы сигналы и PID процессов удаляются.

Все биты регистров **INTERRUPT MASK*** (адрес 1010h) и **CANINTE**** (адрес 2Bh) сбрасываются в ноль.

* 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:**

-

**Входные параметры:**

Param – не используется

**Пример вызова:**

```
retval = ioctl(i, IOCTL_DISABLE_ALL_INT_TTCAN, 0);
if (retval == 0){
  printf("All interrupts has been disabled \n");
}
else{
  printf("error disabling interrupts \n");
}
```

# 11. Обновление драйвера

| Версия документа | Дата | Изменение |
|---|---|---|
| 1.0 | 14.10.2015 | - Драйвер создан |

# 12. Обновление руководства.

| Версия документа | Дата | Изменение |
|---|---|---|
| 1.0 | 14.10.2015 | - Документ создан |