Руководство (v3.2)

По работе с драйвером модулей “mPCIe – TTCAN”, “PCIe – TTCAN”

Интерфейс ISO-11898-4
(TT CAN)

Для драйверов версии 3.1

ОС LINUX

08.12.2021

ООО “Новомар”

Оглавление

1. Введение.

Драйвер поддерживает модули: “**mPCIe-TTCAN**”, “**PCIe-TTCAN**”(далее **xPCIe-TTCAN**).
Данный драйвер присваивает устройствам 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 “nmttcan.h”.
Обращение к независимым каналам платы осуществляется по номеру канала (1 и 2).

2. Сборка и установка драйвера.

1. Создайте отдельную папку.
2. Скачайте в эту папку архив с исходными текстами драйвера и распакуйте его.
3. Выполните команду “cd mPCIe-TTCAN/drv” для перехода в каталог с исходными текстами драйвера.
4. Выполните команду “make”.
5. Выполните команду “sudo make install”. Или выполните шаги 6…9.
6. Создайте папку “/modules” в корне файловой системы.
7. Поместите в созданную папку файл “nmttcan.ko”.
8. Откройте файл “/etc/rc.local” текстовым редактором.
9. Добавьте в открытый файл перед строкой “exit 0” строку “insmod /modules/nmttcan.ko”. После этого драйвер будет автоматически загружаться при старте ОС Linux.
10. Для загрузки драйвера вручную используйте команду “insmod /modules/nmttcan.ko”.
1`. Для выгрузки драйвера вручную используйте команду “rmmod nmttcan.ko”.

Если Вы хотите проверить, загружен ли драйвер в ядро:
1. В терминале введите команду “lsmod”.
2. Найдите в выведенном списке “nmttcan”.
3. Если есть — все нормально, если нет — драйвер не работает.

Внимание! Для работы с драйвером версии 3.0 и выше в режиме FIFO, со счётчиками статистики и таблицей Timemark необходимо Firmware модуля не ниже «07» от 16.07.2020 (См. поле: «Дата предпродажной проверки» на этикетке).

Для обновления Firmware обратитесь к производителю.

Для обновления версии драйвера:
1. Посмотреть текущую версию драйвера в файле “/var/log/kern.log”, поискав в нём последнюю строку вида “Novomar, Ltd. mPCIe-TTCAN driver version: ?.? date: ??.??.????”.
2. Сравнить с версией текущего драйвера.
3. Выбрать более позднюю.
4. Заменить файл “nmttcan.ko” в папке /modules на новый.
5. Выгрузить старый драйвер командой “rmmod nmtcan.ko”.
6. Загрузить новый драйвер командой “insmod /modules/nmttcan.ko”.

3. Подключение файла с командами к разрабатываемому проекту.

1. Скопировать файл “nmttcan.h” из папки “/modules” в папку с проектом.
2. В начале проекта добавить строку:
   #include "nmttcan.h"
3. Использовать команды.

Формат описан ниже; примеры использования приложены.

ВНИМАНИЕ

Системная функция **open** открывает модуль “xPCIe-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 функций, добавленных в библиотеке версии 2.0	
IOCTL_WR_MAINREG_TTCAN	Запись регистра модуля.
IOCTL_RD_MAINREG_TTCAN	Чтение регистра модуля.
IOCTL_WR_CANREG_TTCAN	Запись регистра контроллера канала.
IOCTL_RD_CANREG_TTCAN	Чтение регистра контроллера канала.
IOCTL_MODIFY_CANREG_TTCAN	Модификация регистра контроллера канала.
IOCTL_WR_CANREGS_TTCAN	Запись регистров контроллера канала.
IOCTL_RD_CANREGS_TTCAN	Чтение регистров контроллера канала.
IOCTL_VERSION_TTCAN	Информация о плате.
IOCTL_VERSION_DRIVER_TTCAN	Информация о драйвере.
IOCTL_ENABLE_DMA_TTCAN	Разрешение работы DMA.
IOCTL_DISABLE_DMA_TTCAN	Запрет работы DMA.
IOCTL_RESET_TG_TTCAN	Сброс триггера канала.
IOCTL_SET_MODE_TTCAN	Установка режима работы канала.
IOCTL_GET_MODE_TTCAN	Чтение режима работы канала.
IOCTL_SET_ONESHOT_MODE_TTCAN	Установка/снятие режима однократной попытки отправки сообщения для канала.
IOCTL_SET_SPEED_TTCAN	Установка скорости работы канала.
IOCTL_SET_SPEED_PARAMS_TTCAN	Установка специальной скорости работы канала.
IOCTL_GET_ERRORS_TTCAN	Чтение регистров ошибок канала.
IOCTL_SET_MASKS_TTCAN	Установка масок и фильтров на прием сообщений канала.
IOCTL_SET_CANn_TIMER_TRSH_TTCAN	Запуск таймера канала с ограничением счёта.
IOCTL_SET_CANn_TIMER_CEED_TTCAN	Установка начального значения таймера канала при сбросе.
IOCTL_SET_CANn_TIMER_FREE_TTCAN	Запуск таймера канала в режиме свободного счёта.
IOCTL_SET_CANn_TIMER_RST_ON_TXB_TTCAN	Установка сброса таймера канала по приёму сообщения в буфер RXBn.
IOCTL_STOP_CANn_TIMER_TTCAN	Остановка таймера канала.
IOCTL_GET_CANn_TIMER_TTCAN	Чтение текущего значения таймера канала.
IOCTL_START_CANn_TIMER_INT_TTCAN	Установка прерывания по таймеру канала.
IOCTL_STOP_CANn_TIMER_INT_TTCAN	Остановка прерывания по таймеру

	канала.
IOCTL_WAIT_CANn_TIMER_INT_TTCAN	Ожидание прерывания по таймеру канала.
IOCTL_SET_CANn_TIMEOUTS_TTCAN	Установка абсолютного и интервального таймера прерываний канала.
IOCTL_RESET_TTCAN	Сброс модуля.
IOCTL_RESET_CANn_TTCAN	Сброс канала.
IOCTL_RD_CH_RAW_DMA_TTCAN	Чтение данных из DMA канала.
IOCTL_WRITE_DATA_TO_TR_BUF_TTCAN	Запись данных в буфер отправки канала.
IOCTL_SEND_DATA_TTCAN	Запись данных в буфер отправки канала, запуск передачи сообщения и ожидание завершения передачи.
IOCTL_SEND_DATA_NOW_TTCAN	Запуск передачи сообщения из буфера отправки канала.
IOCTL_CHECK_TRANSMIT_TTCAN	Проверка завершения и правильности передачи сообщения из буфера отправки канала.
IOCTL_WAIT_TRANSMIT_TTCAN	Ожидание завершения и проверка правильности передачи сообщения из буфера отправки канала.
IOCTL_END_TRANSMIT_TTCAN	Снятие запроса на передачу сообщения из буфера отправки канала.
IOCTL_SEND_DATA_TG_TTCAN	Запуск однократной передачи сообщения из буфера отправки канала по значению триггера.
IOCTL_SEND_DATA_LOOP_TTCAN	Запуск повторяющейся передачи сообщения из буфера отправки канала по значению триггера.
IOCTL_CHECK_TG_TTCAN	Проверка срабатывания триггера.
IOCTL_ABAT_TTCAN	Установка/снятие режима остановки всех активных передач.
Список функций, добавленных в библиотеке версии 3.0	
IOCTL_READ_DMA_BLOCKS_CAN	Чтение данных из буфера DMA
IOCTL_WRITE_DATA_TO_FIFO_CAN1	Отправка данных в режиме FIFO для канала 1
IOCTL_WRITE_DATA_TO_FIFO_CAN2	Отправка данных в режиме FIFO для канала 2
Список функций, добавленных в библиотеке версии 3.1	
IOCTL_WRITE_DATA_TO_FIFO_CAN1_V2	Отправка данных в режиме FIFO для канала 1
IOCTL_WRITE_DATA_TO_FIFO_CAN2_V2	Отправка данных в режиме FIFO для канала 2

5. Описание использования команд драйвера.

Структура ioctl-команды:

#include <sys/ioctl.h>
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 nmttcan.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 -1. Причину ошибки можно узнать из значения переменной errno:
1. EINVAL – ошибки в параметрах запроса;
2. ETIME – таймаут выполнения запроса;
3. EBUSY – запрос ещё не выполнен;
4. EFAULT – ошибка обращения к памяти пользовательского процесса.
5. ENOTTY – неизвестный запрос.
6. EPERM – запрос не поддерживается.

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

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:
Недопустимо обращение к устройству по адресам, не кратным четырём!!!

Входные параметры:
param – указатель на структуру типа SADDR_DATA_MAIN_TTCAN:

Поле структуры	Описание
uint32_t **daddr**	Адрес регистра
uint32_t **data**	Данные для записи

Пример вызова:

```
int fd, ret;
    SADDR_DATA_MAIN_TTCAN saddr;
fd = open("/dev/ttcan_dev_0", O_RDWR);
saddr.daddr = 0x2000;
saddr.data = 0;
ret = ioctl(fd, IOCTL_WR_MAINREG_TTCAN, &saddr);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("Data %08XH was written to BAR register %08XH \n", saddr.data, saddr.daddr);
```

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 поле **data**.

Примечание:
Недопустимо обращение к устройству по адресам, не кратным четырём!

Входные параметры:
param – указатель на структуру типа SADDR_DATA_MAIN_TTCAN

Поле структуры	Описание
uint32_t **daddr**	Адрес регистра
uint32_t **data**	Возвращаемый результат

Пример вызова:

```
int fd, ret;
    SADDR_DATA_MAIN_TTCAN saddr;
fd = open("/dev/ttcan_dev_0", O_RDWR);
saddr.daddr = 0x2000;
ret = ioctl(fd, IOCTL_RD_MAINREG_TTCAN, &saddr);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("Data %08XH was read from BAR register %08XH\n", saddr.data, saddr.daddr);
```

6.3 IOCTL_WR_CANREG_TTCAN

Назначение:

Запись данных в одиночный регистр контроллера CAN.

Действие:

Функция записывает данные в **CANn*_BUF****. После чего записывает команду “Write” в регистр **CANn*_ACS***** и дожидается её выполнения путём перевода процесса в состояние ожидания, из которого процесс будет выведен после прихода прерывания, означающего окончание операции или по истечении 10 миллисекунд. В последнем случае функция возвратит значение -1, а в errno будет установлено значение ETIME.
Все функции работы с регистрами контроллера CAN защищены общим семафором (индивидуальным для каждого канала) для возможности их корректного использования из параллельных процессов.

* n – номер контроллера.
** См. Раздел 6.1.4 или 6.1.6 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.
*** См. Раздел 6.1.3 или 6.1.5 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.

Примечание:

-

Входные параметры:

param – указатель на структуру типа SADDR_DATA_TTCAN:

Поле структуры	Описание
uint8_t **daddr**	Адрес регистра
uint8_t **data**	Данные для записи
uint32_t **channel**	Номер канала (допустимые значения – 1…2)

Пример вызова:

```
int fd, ret;
    SADDR_DATA_TTCAN saddr;
fd = open("/dev/ttcan_dev_0", O_RDWR);
saddr.daddr = CANINTE;
saddr.data = 0x0;
saddr.channel = 1;
ret = ioctl(fd, IOCTL_WR_CANREG_TTCAN, &saddr);
if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("Data %02XH was written to register %02XH of CAN%u\n", saddr.data, saddr.daddr,
    saddr.channel);
```

6.4 IOCTL_RD_CANREG_TTCAN

Назначение:
Чтение данных из одиночного регистра контроллера CAN.

Действие:
Функция записывает команду “Read” в регистр **CANn*_ACS**** и дожидается её выполнения путём перевода процесса в состояние ожидания, из которого процесс будет выведен после прихода прерывания, означающего окончание операции или по истечении 10 миллисекунд. В последнем случае функция возвратит значение -1, а в errno будет установлено значение ETIME. В случае получения прерывания о завершении операции вычитанные данные из **CANn*_BUF***** копируются в поле **data**.
Все функции работы с регистрами контроллера CAN защищены общим семафором (индивидуальным для каждого канала) для возможности их корректного использования из параллельных процессов.

* n – номер контроллера.
** См. Раздел 6.1.4 или 6.1.6 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.
*** См. Раздел 6.1.3 или 6.1.5 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.

Примечание:
-

Входные параметры:
param – указатель на структуру типа SADDR_DATA_TTCAN:

Поле структуры	Описание
uint8_t **daddr**	Адрес регистра
uint8_t **data**	Возвращаемый результат
uint32_t **channel**	Номер канала (допустимые значения – 1…2)

Пример вызова:

```
int fd, ret;
    SADDR_DATA_TTCAN saddr;
fd = open("/dev/ttcan_dev_0", O_RDWR);
saddr.daddr = CAN_CTRL;
saddr.channel = 1;
ret = ioctl(fd, IOCTL_RD_CANREG_TTCAN, &saddr);
if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("Data %02XH was read from register %02XH of CAN%u \n", saddr.data, saddr.daddr,
    saddr.channel);
```

6.5 IOCTL_MODIFY_CANREG_TTCAN

Назначение:
Модификация регистра контроллера CAN.

Действие:
Функция записывает данные в **CANn*_BUF****. После чего записывает команду “Bit Modify” в регистр **CANn*_ACS***** и дожидается её выполнения путём перевода процесса в состояние ожидания, из которого процесс будет выведен после прихода прерывания, означающего окончание операции или по истечении 10 миллисекунд. В последнем случае функция возвратит значение -1, а в errno будет установлено значение ETIME.
Все функции работы с регистрами контроллера CAN защищены общим семафором (индивидуальным для каждого канала) для возможности их корректного использования из параллельных процессов.

* n – номер контроллера.
** См. Раздел 6.1.4 или 6.1.6 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.
*** См. Раздел 6.1.3 или 6.1.5 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.

Примечание:
-

Входные параметры:
param – указатель на структуру типа SADDR_DATA_MODIFY_TTCAN:

Поле структуры	Описание
uint8_t **daddr**	Адрес регистра
uint16_t **data**	Данные для записи (младшие 8 бит – маска для указания модифицируемых битов регистра, старшие 8 бит – новое значение для модифицируемых битов регистра)
uint32_t **channel**	Номер канала (допустимые значения – 1…2)

Пример вызова:

```
int fd, ret;
    SADDR_DATA_MODIFY_TTCAN saddr;
fd = open("/dev/ttcan_dev_0", O_RDWR);
saddr.daddr = TXB0CTRL;
saddr.data = 0x0808;
saddr.channel = 1;
ret = ioctl(fd, IOCTL_MODIFY_CANREG_TTCAN, &saddr);
if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
    else
printf("Data %02H was written to bits %02XH of register %02XH of CAN%u\n", saddr.data >> 8,
    saddr.data & 0xFF, saddr.daddr, saddr.channel);
```

6.6 IOCTL_WR_CANREGS_TTCAN

Назначение:
Запись блока данных в регистры контроллера CAN.

Действие:
Функция записывает данные в **CANn*_BUF****. После чего записывает команду “Write” в регистр **CANn*_ACS***** и дожидается её выполнения путём перевода процесса в состояние ожидания, из которого процесс будет выведен после прихода прерывания, означающего окончание операции или по истечении 10 миллисекунд. В последнем случае функция возвратит значение -1, а в errno будет установлено значение ETIME.
Все функции работы с регистрами контроллера CAN защищены общим семафором (индивидуальным для каждого канала) для возможности их корректного использования из параллельных процессов.

* n – номер контроллера.
** См. Раздел 6.1.4 или 6.1.6 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.
*** См. Раздел 6.1.3 или 6.1.5 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.

Примечание:
-

Входные параметры:
param – указатель на структуру типа SADDR_DATAS_TTCAN:

Поле структуры	Описание
uint8_t **channel**	Номер канала (допустимые значения – 1…2)
uint8_t **addr**	Адрес первого регистра
uint8_t **nsize**	Количество записываемых данных (допустимые значения – 1…16)
uint8_t **data**[16]	Записываемые данные

Пример вызова:

```
int fd, ret;
    SADDR_DATAS_TTCAN saddr;
fd = open("/dev/ttcan_dev_0", O_RDWR);
saddr.channel = 1;
saddr.addr = TXB0CTRL + 1;
saddr.nsize = 6;
saddr.data[0] = 0x00;
saddr.data[1] = 0x00; // SID=000, EID absent
saddr.data[2] = 0x00;
saddr.data[3] = 0x00;
saddr.data[4] = 0x01; // DLC=1
saddr.data[5] = 0xFF; //TXB0D0=FFH
ret = ioctl(fd, IOCTL_WR_CANREGS_TTCAN, &saddr);
if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u registers was filled with data starting from address %02XH\n", saddr.channel);
```

6.7 IOCTL_RD_CANREGS_TTCAN

Назначение:
Чтение блока данных из регистров контроллера CAN.

Действие:
Функция записывает команду “Read” в регистр **CANn*_ACS**** и дожидается её выполнения путём перевода процесса в состояние ожидания, из которого процесс будет выведен после прихода прерывания, означающего окончание операции или по истечении 10 миллисекунд. В последнем случае функция возвратит значение -1, а в errno будет установлено значение ETIME. В случае получения прерывания о завершении операции вычитанные данные из **CANn*_BUF***** копируются в поле **data**.
Все функции работы с регистрами контроллера CAN защищены общим семафором (индивидуальным для каждого канала) для возможности их корректного использования из параллельных процессов.

* n – номер контроллера.
** См. Раздел 6.1.3 или 6.1.5 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.
*** См. Раздел 6.1.4 или 6.1.6 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.

Примечание:
-

Входные параметры:
param – указатель на структуру типа SADDR_DATAS_TTCAN:

Поле структуры	Описание
uint8_t **channel**	Номер канала (допустимые значения – 1…2)
uint8_t **addr**	Адрес первого регистра
uint8_t **nsize**	Количество считываемых данных (допустимые значения – 1…16)
uint8_t **data**[16]	Возвращаемый результат

Пример вызова:

```
int fd, ret;
    SADDR_DATAS_TTCAN saddr;
fd = open("/dev/ttcan_dev_0", O_RDWR);
saddr.channel = 1;
saddr.addr = TXB0CTRL + 1;
saddr.nsize = 6;
ret = ioctl(fd, IOCTL_RD_CANREGS_TTCAN, &saddr);
if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("Data %02XH %02XH %02XH %02XH %02XH %02XH was read from registers starting
    from %02XH of CAN%u\n", saddr.data[0], saddr.data[1], saddr.data[2], saddr.data[3],
    saddr.data[4], saddr.data[5], saddr.addr, saddr.channel);
```

6.8 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
uint32_t **device_id**	Идентификатор модуля
uint32_t **vendor_id**	Идентификатор производителя
uint8_t **revision**	Номер ревизии модуля
char **dev_name[30]**	Имя модуля, назначенное драйвером
uint32_t **minor**	Порядковый номер модуля, присвоенный драйвером
uint32_t **irq**	Номер линии прерываний
uint64_t **size_dma**	Размер буфера DMA
uint64_t **addr_dma_virt**	Виртуальный адрес в ядре ОС буфера DMA
uint64_t **pci_bars**	Виртуальный адрес в ядре ОС регистрового пространства (BAR)

Пример вызова:

```
int fd, ret;
    VERSION_TTCAN ver;
fd = open("/dev/ttcan_dev_0", O_RDWR);
    ret = ioctl(fd, IOCTL_VERSION_TTCAN, &ver);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
  printf("Vendor ID = %04XH\nDevice ID = %04XH\nRevision ID = %02XH\nDevice name =
    '%s'\nMinor = %u\nInterrupt=%u\nDMA bufsize=%u\n", ver.vendor_id, ver.device_id,
    ver.revision, ver.dev_name, ver.minor, ver.irq, ver.size_dma);
```

6.9 IOCTL_VERSION_DRIVER_TTCAN

Назначение:

Чтение версии и даты драйвера.

Действие:

Функция заполняет 32-битную переменную.

Примечание:

-

Входные параметры:

param – указатель на 32-битную переменную, в которую будет записано в BCD формате значение версии и даты создания драйвера. В старших 8 битах – старшая и младшая цифры версии драйвера. В младших 24 битах шесть цифр - дата создания драйвера (две цифры день, две цифры месяц и младшие две цифры года).

Пример вызова:

```
int fd, ret;
    uint32_t drv_ver_date;
fd = open("/dev/ttcan_dev_0", O_RDWR);
    ret = ioctl(fd, IOCTL_VERSION_DRIVER_TTCAN, &drv_ver_date);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("Driver Version: %X.%X, Date: %02X.%02X.%02X\n", drv_ver_date >> 28, (drv_ver_date
    >> 24) & 0x0F, (drv_ver_date >> 16) & 0xFF, (drv_ver_date >> 8) & 0xFF, drv_ver_date &
    0xFF);
```

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 "xPCIe-TTCAN*".
** См. Раздел 6.4.3 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

Примечание:
После загрузки операционной системы работа DMA не разрешена.

Входные данные

–

Пример вызова:

```
int fd, ret;
    fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = ioctl(fd, IOCTL_ENABLE_DMA_TTCAN);
if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
  printf("DMA work was enabled\n");
```

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 "xPCIe-TTCAN*".
** См. Раздел 6.4.3 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

Примечание:
После загрузки операционной системы работа не разрешена.

Входные данные

–

Пример вызова:

```
int fd, ret;
    fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = ioctl(fd, IOCTL_DISABLE_DMA_TTCAN);
if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
  printf("DMA work was disabled\n");
```

7.3 IOCTL_RESET_TG_TTCAN

Назначение:
Сброс триггера.

Действие:
В биты **TX_TRIGn*_EN** регистра **CANm*_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 “xPCIe-TTCAN*”.

Примечание:
-

Входные параметры:
param – указатель на структуру типа CH_BUF_TTCAN:

Поле структуры	Описание
uint8 **nCh**	Номер канала (допустимые значения – 1…2)
uint8 **nBuf**	Номер буфера (равен номеру триггера)

Пример вызова:

```
int fd, ret;
    CH_BUF_TTCAN cb;
fd = open("/dev/ttcan_dev_0", O_RDWR);
cb.nCh = 1;
cb.nBuf = 2;
ret = ioctl(fd, IOCTL_RESET_TG_TTCAN, &cb);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
  printf("Trigger %u reset on CAN%u \n", cb.nBuf, cb.nCh);
```

7.4 IOCTL_SET_MODE_TTCAN

Назначение:

Установка нового режима работы контроллера.
См. Раздел 6.3 документа "*Руководство по программированию модуля "xPCIe-TTCAN*"

Действие:

Значение из поля **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 "xPCIe-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 "xPCIe-TTCAN*".

Входные параметры:

param – указатель на структуру типа CONF_TTCAN:

Поле структуры	Описание
uint32_t **channel**	Номер канала (допустимые значения – 1…2)
uint8_t **mode**	Новый режим работы контроллера

Пример вызова:

```
int fd, ret;
    CONF_TTCAN conf;
fd = open("/dev/ttcan_dev_0", O_RDWR);
conf.channel = 1;
conf.mode = CAN_WORK; // CAN_CONF, CAN_WORK, CAN_MON, CAN_SLEEP, CAN_LOOP
    ret = ioctl(fd,  IOCTL_SET_MODE_TTCAN, &conf);
    if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
 printf("CAN%u mode set to %u\n", conf.channel, conf.mode);
```

7.5 IOCTL_GET_MODE_TTCAN

Назначение:

Чтение текущего режима работы контроллера.

Действие:

Читает значение битов с 5 по 7 регистра **CAN_STAT*** (адрес Eh) в поле **mode**.

* См. Раздел 6.3.2 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

Примечание:

-

Входные параметры:

param – указатель на структуру типа CONF_TTCAN:

Поле структуры	Описание
uint32_t **channel**	Номер канала (допустимые значения – 1…2)
uint8_t **mode**	Текущий режим работы контроллера

Пример вызова:

```
int fd, ret;
    CONF_TTCAN conf;
fd = open("/dev/ttcan_dev_0", O_RDWR);
conf.channel = 1;
ret = ioctl(fd, IOCTL_GET_MODE_TTCAN, &conf);
if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
 printf("CAN%u mode is %u\n", conf.channel, conf.mode);
```

7.6 IOCTL_SET_ONESHOT_MODE_TTCAN

Назначение:

Установка или снятие режима однократной попытки передачи сообщения контроллером.
См. Раздел 6.3.1 документа "*Руководство по программированию модуля "xPCIe-TTCAN*"

Действие:

Бит 0 из поля **mode** записывается в бит 3 регистра **CAN_CTRL*** (адрес Fh).

* См. раздел 6.5.1 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

Примечание:

-

Входные параметры:

param – указатель на структуру типа CONF_TTCAN:

Поле структуры	Описание
uint32_t **channel**	Номер канала (допустимые значения – 1…2)
uint8_t **mode**	0 – снятие однократного режима. Не 0 – установка однократного режима

Пример вызова:

```
int fd, ret;
    CONF_TTCAN conf;
fd = open("/dev/ttcan_dev_0", O_RDWR);
conf.channel = 1;
conf.mode = 1;
    ret = ioctl(fd,  IOCTL_SET_ONESHOT_MODE_TTCAN, &conf);
    if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
 printf("CAN%u oneshot mode %s\n", conf.channel, conf.mode ? "set" : "reset");
```

7.7 IOCTL_SET_SPEED_TTCAN

Назначение:
Функция конфигурации скорости шины CAN.

Действие:
В зависимости от значения 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 "xPCIe-TTCAN*".
** См. раздел 6.5.2 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".
*** См. раздел 6.5.3 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

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

Входные параметры:
param – указатель на структуру типа SPEED_TTCAN:

Поле структуры	Описание
uint32_t **channel**	Номер канала (допустимые значения – 1…2)
uint32_t **speed**	Значение скорости (допустимые значения – 125, 250, 500 или 1000)

Пример вызова:

```
int fd, ret;
    SPEED_TTCAN spd;
fd = open("/dev/ttcan_dev_0", O_RDWR);
spd.channel = 1;
spd.speed = WORK_SPEED_125;
ret = ioctl(fd, IOCTL_SET_SPEED_TTCAN, &spd);
if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u speed was set to %u kHz\n", spd.channel, spd.speed);
```

7.8 IOCTL_SET_SPEED_PARAMS_TTCAN

Назначение:
Функция конфигурации скорости шины CAN для нестандартных скоростей.

Действие:
В соответствии со входными параметрами 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 "xPCIe-TTCAN*".
** См. раздел 6.5.2 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".
*** См. раздел 6.5.3 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

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

Входные параметры:
param – указатель на структуру типа SPEED_PARAMS_TTCAN:

Поле структуры	Описание
uint32_t **channel**	Номер канала (допустимые значения – 1…2)
uint8_t **brp**	Коэффициент деления частоты опорного генератора (допустимые значения – 1…32)
uint8_t **sjw**	Шаг перестройки синхронизации (допустимые значения – 1…4)
uint8_t **sam**	Конфигурация точки сэмплирования (допустимые значения – 0…1)
uint8_t **btlmode**	Выбор величины PS2 (допустимые значения – 0…1)
uint8_t **phseg1**	Длительность сегмента фазы 1 (допустимые значения – 1…8)
uint8_t **phseg2**	Длительность сегмента фазы 2 (допустимые значения – 1…8)
uint8_t **prseg**	Длительность сегмента фазы распространения (допустимые значения – 1…8)
uint8_t **wakfil**	ФНЧ шины для детектора активности шины в режиме сна (допустимые значения – 0…1)

Пример вызова:

```
int fd, ret;
    SPEED_PARAMS_TTCAN sp;
fd = open("/dev/ttcan_dev_0", O_RDWR);
sp.channel = 1;
sp.brp = 1; sp.sjw = 1;
    sp.btlmode = 1; sp.sam = 1; sp.phseg1 = 3; sp.prseg = 2;
    sp.phseg2 = 4; sp.wakfil = 0;
ret = ioctl(fd, IOCTL_SET_SPEED_PARAMS_TTCAN, &sp);
if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
     printf("CAN%u speed was set\n", spd.channel);
```

7.9 IOCTL_GET_ERRORS_TTCAN

Назначение:
Функция чтения регистров ошибок контроллера CAN.

Действие:
Читаются регистры **TEC***(адрес 1CH), **REC****(адрес 1DH) и **EFLG***** (адрес 2DH) выбранного канала.

* См. раздел 6.6.1 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".
** См. раздел 6.6.2 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".
*** См. раздел 6.6.3 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

Примечание:
-

Входные параметры:
param – указатель на структуру типа ERRORS_TTCAN:

Поле структуры	Описание
uint8_t **channel**	Номер канала (допустимые значения – 1…2)
uint8_t **nEFLG**	Флаги ошибок шины
uint8_t **nTEC**	Счётчик ошибок передачи
uint8_t **nREC**	Счётчик ошибок приёма

Пример вызова:

```
int fd, ret;
    ERRORS_TTCAN er;
fd = open("/dev/ttcan_dev_0", O_RDWR);
er.channel = 1;
ret = ioctl(fd, IOCTL_GET_ERRORS_TTCAN, &er);
if (ret == -1)
 printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u EFLG=%02XH TEC=%u REC=%u \n", er.channel, er.nEFLG, er.nTEC,
    er.nREC);
```

7.10 IOCTL_SET_MASKS_TTCAN

Назначение:
Установка масок и фильтров на прием сообщений.

Действие:
В зависимости от выбранного канала (**channel**) и номера фильтра (**filter**) функция записывает значение **rxb_mode** в биты RXM** регистра **RXBn*CTRL****.

* n – номер буфера.
** См. раздел 6.8.1 и 6.8.2 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

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

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

Примечание:
Для отключения фильтров и масок поле **rxb_mode** должна быть равна 3. В этом случае поля **filter**, **ident**, **identific_m**, **identific_f** игнорируются и модуль будет принимать из линии все сообщения.

Если **rxb_mode** и **ident** равны 0, то 26-19 биты **identific_m** и **identific_f** должны содержать 0й байт данных, а 18-11 биты **identific_m** и **identific_f** должны содержать 1й байт данных.

Входные параметры:
param – указатель на структуру типа MASKS_TTCAN:

Поле структуры	Описание
uint8_t **channel**	Номер канала (допустимые значения – 1…2)
uint8_t **rxb_mode**	Режим работы буфера (допустимые значения – 0…3)
uint8_t **filter**	Номер фильтра (допустимые значения – 0…5). Первые два фильтра (0…1) – 0 буфер, 0 маска; следующие четыре фильтра (2…5) – 1 буфер, 1 маска
uint8_t **ident**	0 – фильтр применяется для сообщений только с SID; 1 – фильтр применяется для сообщений только с EID
uint32_t **identific_m**	идентификатор маски; 10-0 биты – SID; 28-11 биты – EID или первые 2 байта данных
uint32_t **identific_f**	идентификатор фильтра; 10-0 биты – SID; 28-11 биты – EID или первые 2 байта данных

Пример вызова:

```
int fd, ret;
    MASKS_TTCAN mask;
    uint16_t sidmask, sidfilter;
    uint32_t eidmask, eidfilter;
fd = open("/dev/ttcan_dev_0", O_RDWR);
mask.channel = 1;
mask.rxb_mode = 2;
mask.filter = 1;
mask.ident = 1;
    mask.identific_m = (eidmask << 11) | sidmask;
mask.identific_f = (eidfilter << 11) | sidfilter;
retval = ioctl(fd, IOCTL_SET_MASKS_TTCAN, &mask);
```

```
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("Filters and masks for CAN%u was enabled\n", mask.channel);
```

7.11 IOCTL_SET_CANn_TIMER_TRSH_TTCAN

Назначение:
Запуск таймера с указанным периодом.

Действие:
Значение поля **nValue** записывается в регистр **CANn*_TIMER_TRSH**** (значение записывается целиком с нулевого бита).
Биты RST, ENABLE и NTU_MODE регистра **CANn*_TIMER_CTRL***** устанавливаются в единицу, а битовая маска EPOCH_MASK устанавливается в соответствии с параметром **nEpochBits**. Значения остальных битов данного регистра не изменяются.

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

Примечание:
-

Входные параметры:
param – указатель на структуру типа TIMER_TRSH_TTCAN:

Поле структуры	Описание
uint8_t **nCh**	Номер канала (допустимые значения – 1…2)
uint8_t **nEpochBits**	Число бит в счётчике циклов (допустимые значения – 0…8)
uint32_t **nValue**	Новое значение для регистра **CANn*_TIMER_TRSH** (допустимые значения – 4…FFFFFFFCH)

Пример вызова:

```
int fd, ret;
    TIMER_TRSH_TTCAN tt;
fd = open("/dev/ttcan_dev_0", O_RDWR);
tt.nCh = 1;
    tt.nEpochBits = 2;
tt.nValue = ((4 * 16 * 8 - 1) << 16) | ((1000 / 50 - 1) << 2);
ret = ioctl(fd, IOCTL_SET_CANn_TIMER_TRSH_TTCAN, &tt);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u_TIMER_TRSH has been set\n", tt.nCh);
```

7.12 IOCTL_SET_CANn_TIMER_CEED_TTCAN

Назначение:
Установка режима и значений корректировки начального значения таймера.

Действие:
Значение поля **nValue** записывается в регистр **CANn*_TIMER_CEED**** (значение записывается целиком с нулевого бита).
Если значение **bEpoch** отлично от нуля, то значение поля **nEpoch** записывается в регистр **CANn*_EPOCH_CEED***** (значение записывается целиком с нулевого бита), а бит EPOCH_CEED_EN регистра **CANn*_TIMER_CTRL****** устанавливается в единицу
Бит CEED_EN регистра **CANn*_TIMER_CTRL****** устанавливается в единицу. Значения остальных битов данного регистра не изменяются.

* n – номер канала.
** См. раздел 5.2.3 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".
*** См. раздел 5.2.6 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".
**** См. раздел 5.2.4 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

Примечание:
-

Входные параметры:
param – указатель на структуру типа TIMER_TRSH_TTCAN:

Поле структуры	Описание
uint8_t **nCh**	Номер канала (допустимые значения – 1…2)
uint8_t **bEpoch**	Флаг использования счётчика циклов
uint32_t **nEpoch**	Новое значение для регистра **CANn*_EPOCH_CEED***** (допустимые значения – 0…FFH)
uint32_t **nValue**	Новое значение для регистра **CANn*_TIMER_CEED** (допустимые значения – 0…FFFFFFFCH)

Пример вызова:

```
int fd, ret;
    TIMER_TRSH_TTCAN tt;
fd = open("/dev/ttcan_dev_0", O_RDWR);
tt.nCh = 1;
tt.bEpoch = 1;
    tt.nEpoch = 1;
tt.nValue = (1 * 16 * 8) << 16;
ret = ioctl(fd, IOCTL_SET_CANn_TIMER_CEED_TTCAN, &tt);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u_TIMER_CEED&CAN%u_EPOCH_CEED has been set\n", tt.nCh);
```

7.13 IOCTL_SET_CANn_TIMER_FREE_TTCAN

Назначение:
Запуск таймера в режиме свободного счёта.

Действие:
В регистре **CANn*_TIMER_CTRL**** биты EPOCH_CEED_EN, RST_ON_RXB0, RST_ON_RXB1, NTU_MODE и CEED_EN устанавливаются в ноль, биты RST и ENABLE устанавливаются в единицу, а битовая маска EPOCH_MASK устанавливается в соответствии с параметром **nEpochBits**. Значения остальных битов данного регистра не изменяются.

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

Примечание:
-

Входные параметры:
param – указатель на структуру типа TIMER_TRSH_TTCAN:

Поле структуры	Описание
uint8_t **nCh**	Номер канала (допустимые значения – 1…2)
uint8_t **nEpochBits**	Число бит в счётчике циклов (допустимые значения – 0…8)

Пример вызова:

```
int fd, ret;
    TIMER_TRSH_TTCAN tt;
fd = open("/dev/ttcan_dev_0", O_RDWR);
tt.nCh = 1;
    tt.nEpochBits = 2;
ret = ioctl(fd, IOCTL_SET_CANn_TIMER_FREE_TTCAN, &tt);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u TIMER set FREE \n", tt.nCh);
```

7.14 IOCTL_SET_CANn_TIMER_RST_ON_RXB_TTCAN

Назначение:

Установка/снятие режима сброса таймера по приёму сообщения в буферы RXB0 и RXB1.

Действие:

В регистре **CANn*_TIMER_CTRL**** бит RST_ON_RXB0 устанавливается в соответствии со значением бита 0 параметра **nValue**, а бит RST_ON_RXB1 устанавливается в соответствии со значением бита 1 параметра **nValue**. Значения остальных битов данного регистра не изменяются.

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

Примечание:

-

Входные параметры:

param – указатель на структуру типа TIMER_TRSH_TTCAN:

Поле структуры	Описание
uint8_t **nCh**	Номер канала (допустимые значения – 1…2)
uint32_t **nValue**	Бит 0 управляет режимом RST_ON_RXB0, а бит 1 управляет режимом RST_ON_RXB1

Пример вызова:

```
int fd, ret;
    TIMER_TRSH_TTCAN tt;
fd = open("/dev/ttcan_dev_0", O_RDWR);
tt.nCh = 1;
    tt.nValue = 1 << 0;
ret = ioctl(fd, IOCTL_SET_CANn_TIMER_RST_ON_RXB_TTCAN, &tt);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u RST_ON_RXB0 has been set\n", tt.nCh);
```

7.15 IOCTL_STOP_CANn_TIMER_TTCAN

Назначение:
Запрещение работы таймера.

Действие:
В регистре **CANn*_TIMER_CTRL**** бит ENABLE устанавливается в ноль. Значения остальных битов данного регистра не изменяются.

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

Примечание:
-

Входные параметры:

param – указатель на 32-битную переменную, в которой указан номер канала. Допустимые значения – 1...2.

Пример вызова:

```
int fd, ret;
    uint32_t channel;
fd = open("/dev/ttcan_dev_0", O_RDWR);
channel = 1;
    ret = ioctl(fd, IOCTL_STOP_CANn_TIMER _TTCAN, &channel);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u TIMER stopped\n", channel);
```

7.16 IOCTL_GET_CANn_TIMER_TTCAN

Назначение:
Чтение текущего значения таймера.

Действие:
Регистр **CANn*_TIMER**** читается в поле **nValue**, а регистр **CANn*_TIMER_EPOCH***** читается в поле **nEpoch**.

* n – номер канала.
** См. раздел 5.2.1 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".
*** См. раздел 5.2.5 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

Примечание:
-

Входные параметры:
param – указатель на структуру типа TIMER_TRSH_TTCAN:

Поле структуры	Описание
uint8_t **nCh**	Номер канала (допустимые значения – 1…2)
uint32_t **nValue**	Значение регистра **CANn*_TIMER****
uint32_t **nEpoch**	Значение регистра **CANn*_TIMER_EPOCH*****

Пример вызова:

```
int fd, ret;
    TIMER_TRSH_TTCAN tt;
fd = open("/dev/ttcan_dev_0", O_RDWR);
tt.nCh = 1;
    ret = ioctl(fd, IOCTL_GET_CANn_TIMER _TTCAN, &tt);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u TIMER EPOCH=%u NTU=%u DIV=%u\n", tt.nCh, tt.nEpoch, tt.nValue >> 16,
    (tt.nValue >> 2) & 0x3FFF);
```

7.17 IOCTL_START_CANn_TIMER_INT_TTCAN

Назначение:
Запуск триггера прерывания по значению таймера.

Действие:
Значение поля **nValue** записывается в регистр **CANn*_INT_TRIG**** (значение записывается целиком с нулевого бита).
Если значение **bEpoch** отлично от нуля, то значение поля **nEpoch** записывается в регистр **CANn*_INT_TRIG_EPOCH***** (значение записывается целиком с нулевого бита), а бит INT_TRIG_EPOCH_EN регистра **CANn*_TRIG_CTRL****** устанавливается в единицу. Иначе бит INT_TRIG_EPOCH_EN регистра **CANn*_TRIG_CTRL****** устанавливается в ноль.
В регистре **CANn*_TRIG_CTRL****** бит INT_TRIG_RPT устанавливается в единицу, а биты INT_TRIG_EN регистра устанавливаются в “01”. Значения остальных битов данного регистра не изменяются.

* n – номер канала.
** См. раздел 5.3.5 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.
*** См. раздел 5.3.8 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.
**** См. раздел 5.3.3 или 5.3.4 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.

Примечание:
-

Входные параметры:
param – указатель на структуру типа TIMER_TRSH_TTCAN:

Поле структуры	Описание
uint8_t **nCh**	Номер канала (допустимые значения – 1…2)
uint8_t **bEpoch**	Флаг учёта счётчика циклов
uint32_t **nEpoch**	Новое значение для регистра **CANn*_INT_TRIG_EPOCH***** (допустимые значения – 0…FFH)
uint32_t **nValue**	Новое значение для регистра **CANn*_INT_TRI G **** (допустимые значения – 0…FFFFFFFCH)

Пример вызова:

```
int fd, ret;
    TIMER_TRSH_TTCAN tt;
fd = open("/dev/ttcan_dev_0", O_RDWR);
tt.nCh = 1;
tt.bEpoch = 1;
    tt.nEpoch = 1;
tt.nValue = (1 * 16 * 8) << 16;
ret = ioctl(fd, IOCTL_START_CANn_TIMER_INT_TTCAN, &tt);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u_TIMER INT started\n", tt.nCh);
```

7.18 IOCTL_STOP_CANn_TIMER_INT_TTCAN

Назначение:
Останов триггера прерывания по значению таймера.

Действие:
В регистре **CANn*_TRIG_CTRL**** биты INT_TRIG_EPOCH_EN и INT_TRIG_RPT устанавливаются в ноль, а биты INT_TRIG_EN устанавливаются в “10”. Значения остальных битов данного регистра не изменяются.

* n – номер канала.
** См. раздел 5.3.3 или 5.3.4 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.

Примечание:
-

Входные параметры:
param – указатель на 32-битную переменную, в которой указан номер канала. Допустимые значения – 1...2.

Пример вызова:

```
int fd, ret;
    uint32_t channel;
fd = open("/dev/ttcan_dev_0", O_RDWR);
channel = 1;
    ret = ioctl(fd, IOCTL_STOP_CANn_TIMER_INT _TTCAN, &channel);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u TIMER INT stopped \n", channel);
```

7.19 IOCTL_WAIT_CANn_TIMER_INT_TTCAN

Назначение:
Ожидание срабатывания триггера прерывания по значению таймера.

Действие:
Функция переводит вызвавший процесс в ждущий режим. Процесс возобновляется после срабатывания прерывания или по истечении таймаута. В случае таймаута ожидания функция возвращает -1, а errno будет равно ETIME.

Примечание:
-

Входные параметры:
param – указатель на структуру типа TIMER_TRSH_TTCAN:

Поле структуры	Описание
uint8_t **nCh**	Номер канала (допустимые значения – 1…2)
uint32_t **nValue**	Таймаут ожидания прерывания, в миллисекундах

Пример вызова:

```
int fd, ret;
    TIMER_TRSH_TTCAN tt;
fd = open("/dev/ttcan_dev_0", O_RDWR);
tt.nCh = 1;
    tt.nValue = 10;
ret = ioctl(fd, IOCTL_WAIT_CANn_TIMER _TTCAN, &tt);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u TIMER triggered\n", tt.nCh);
```

7.20 IOCTL_SET_CANn_TIMEOUTS_TTCAN

Назначение:
Установка значений абсолютного и интервального таймера.

Действие:
Поле **absolute** пишется в регистр **CANn*_TIMEOUT_ABSOLUTE****, а поле **interval** пишется в регистр **CANn*_TIMEOUT_INTERVAL*****.

* n – номер канала.
** См. раздел 5.1.5 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".
*** См. раздел 5.1.6 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

Примечание:
-

Входные параметры:
param – указатель на структуру типа TIMEOUTS _TTCAN:

Поле структуры	Описание
uint8_t **channel**	Номер канала (допустимые значения – 1…2)
uint32_t **absolute**	Значение абсолютного таймера, в микросекундах (допустимые значения – 0…3FFFFFH)
uint32_t **interval**	Значение интервального таймера, в микросекундах (допустимые значения – 0…3FFFFFH)

Пример вызова:

```
int fd, ret;
    TIMEOUTS_TTCAN t;
fd = open("/dev/ttcan_dev_0", O_RDWR);
t.channel = 1;
    t.absolute = 4 * 16 * 8;
t.interval = 1 * 16 * 8;
ret = ioctl(fd, IOCTL_SET_CANn_TIMEOUTS _TTCAN, &t);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u timeouts set: absolute=%u us; interval = %u us\n", t.channel, t.absolute,
    t.interval);
```

7.21 IOCTL_RESET_TTCAN

Назначение:
Программный сброс модуля и аппаратный сброс всех контроллеров CAN.

Действие:
Запрещается DMA, очищается буфер DMA, сбрасываются в исходные значения все регистры, таймеры и триггеры модуля. Происходит аппаратный сброс всех контроллеров CAN. Все регистры всех контроллеров CAN возвращаются к значениям по умолчанию.

Примечание:
-

Входные параметры:
-

Пример вызова:

```
int fd, ret;
    fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = ioctl(fd, IOCTL_RESET_TTCAN);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("TTCAN module reset OK\n");
```

7.22 IOCTL_RESET_CANn_TTCAN

Назначение:

Аппаратный сброс контроллера CAN. Все регистры контроллера CAN возвращаются к значениям по умолчанию.

Действие:

В бит RST_CANn* регистра **CANn*_CTRL**** записывается единица.

* n – номер канала.
** См. раздел 6.1.1 или 6.1.2 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

Примечание:

-

Входные параметры:

param – указатель на 32-битную переменную, в которой указан номер канала. Допустимые значения – 1...2.

Пример вызова:

```
int fd, ret;
    uint32_t channel;
fd = open("/dev/ttcan_dev_0", O_RDWR);
channel = 1;
    ret = ioctl(fd, IOCTL_RESET_CANn _TTCAN, &channel);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u reset OK \n", channel);
```

8. Функции для чтения принятых данных

8.1 IOCTL_RD_CH_RAW_DMA_TTCAN

Назначение:
Чтение данных из буфера DMA. Данная функция пытается считать требуемое количество новых блоков данных по запрашиваемому каналу.

Действие:
Функция проверяет в буфере DMA количество новых блоков данных. Если в поле **number_block** запрашивается меньше блоков, чем накопилось в буфере DMA, то функция копирует эти блоки данных в поле **buf** и немедленно возвращает результат. Если же запрашивается больше блоков, чем накопилось на данный момент в буфере DMA, то при нулевом времени ожидания функция скопирует имеющиеся блоки данных в поле **buf** и немедленно возвратит результат. А при отличном от нуля времени ожидания процесс перейдёт в состояние ожидания и будет возобновлён при накоплении в DMA требуемого количества блоков данных, истечении указанного времени ожидания, или срабатывании абсолютного или интервального таймера, после чего скопирует накопившееся количество блоков данных (но не больше, чем запрошенное) в поле **buf**. В любом случае после возврата из функции в поле **number_block** будет указано количество скопированных блоков данных в поле **buf**.
При включённом режиме аппаратного контроля переполнения буфера DMA в случае переполнения буфера произойдёт временная остановка записи в буфер DMA до освобождения свободного места. В этом случае в пакете, который будет записан после этого события в буфер DMA, бит OVF поля RXBnCTRL структуры DMA_SLOT _TTCAN будет установлен в единицу.

Примечание:
О реакции на переполнение буфера DMA см. разделы 5.1.1, 5.1.2 и 6.8 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.
Формат структуры DMA_SLOT_TTCAN см. в файле “nmttcan.h” и в разделе 6.8 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.

Входные параметры:
param – указатель на структуру типа DMA_STR_TTCAN:

Поле структуры	Описание
uint32_t **number_block**	Кол-во запрашиваемых/полученных блоков (допустимые значения при вызове – 1…400). После вызова значение может быть 0…<исходное значение при вызове>
uint32_t **number_channel**	Номер канала (допустимые значения – 1…2)
uint32_t **timeout**	Максимальное время ожидания требуемого количества блоков данных, в миллисекундах
DMA_SLOT_TTCAN **buf[400]**	Область, в которую будут записаны накопившиеся блоки данных

Пример вызова:

```
int fd, ret;
DMA_STR_TTCAN dma;
fd = open("/dev/ttcan_dev_0", O_RDWR);
dma.number_block = 64;
dma.number_channel = 1;
dma.timeout = 1;
ret = ioctl(fd, IOCTL_RD_CH_RAW_DMA_TTCAN, &dma);
if (ret == -1)
```

```
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else{
  if (dma.number_block > 0)
    printf("Received %u blocks from DMA\n", dma.number_block);
  else
    printf("No new data in DMA\n");
     }
```

8.2 IOCTL_READ_DMA_BLOCKS_TTCAN

Назначение:
Чтение данных из буфера DMA. Данная функция пытается считать доступное количество новых блоков данных по запрашиваемому каналу, но не более 64 за раз.

Действие:
Функция проверяет в буфере DMA количество новых блоков данных, копирует эти блоки данных в поле **buf** и немедленно возвращает результат.

Примечание:
О реакции на переполнение буфера DMA см. разделы 5.1.1, 5.1.2 и 6.8 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".
Формат структуры DMA_READ_BLOCK см. в файле "nmttcan.h.

Входные параметры:
param – указатель на структуру типа DMA_READ_BLOCK:

Поле структуры	Описание
int numChannel	Номер канала (допустимые значения – 1…2)
unsigned int countBlocks	Фактическое считанное кол-во блоков DMA
DMA_RAW_BLOCK_64 blocks	Массив блоков DMA
STAT_INFO info	Информация о состоянии циклического буфера DMA

Пример вызова:

```
int fd, ret;
DMA_READ_BLOCK dma;
fd = open("/dev/can_dev_0", O_RDWR);
dma. numChannel = 1;
ret = ioctl(fd, IOCTL_READ_DMA_BLOCKS_CAN, &dma);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else{
  if (dma.number_block > 0)
    printf("Received %u blocks from DMA\n", dma.number_block);
  else
    printf("No new data in DMA\n");
     }
```

9. Функции для передачи данных

В данном драйвере предусмотрено пять режимов передачи данных:
четыре для режима **Native**:
- синхронный режим передачи (вызов IOCTL_SEND_DATA_TTCAN);
- асинхронный режим передачи (вызов IOCTL_SEND_DATA_NOW_TTCAN);
- асинхронный режим передачи по однократному срабатыванию триггера (вызов IOCTL_SEND_DATA_TG_TTCAN);
- асинхронный режим передачи по повторяющимся срабатываниям триггера (вызов IOCTL_SEND_DATA_LOOP_TTCAN);
- и передача в режиме **FIFO** с помощью вызовов:
IOCTL_WRITE_DATA_TO_FIFO_TTCAN_V2, IOCTL_WRITE_DATA_TO_FIFO_TTCAN2_V2.

До вызова функции асинхронной 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.

После вызова асинхронной отправки данных IOCTL_SEND_DATA_NOW_TTCAN необходимо дождаться отправки сообщения либо путём периодического опроса вызовом IOCTL_CHECK_TRANSMIT_TTCAN, либо путём синхронного вызова IOCTL_WAIT_TRANSMIT _TTCAN.

После вызова асинхронной отправки данных IOCTL_SEND_DATA_TG_TTCAN необходимо дождаться отправки сообщения либо путём периодического опроса сперва о срабатывании триггера вызовом IOCTL_CHECK_TG_TTCAN, а потом об окончании передачи вызовом IOCTL_CHECK_TRANSMIT_TTCAN, либо путём синхронного вызова IOCTL_WAIT_TRANSMIT _TTCAN.

Исключение: в случае асинхронной передачи с одинаковыми приоритетами одновременно из трёх буферов необходимо использовать следующий алгоритм:
1. Загрузить данные в буфер 2 и отправить его на передачу.
2. Загрузить данные в буфер 1 и отправить его на передачу.
3. Загрузить данные в буфер 0 и отправить его на передачу.
4. Дождаться окончания передачи из буфера 2.
5. Дождаться окончания передачи из буфера 1.
6. Дождаться окончания передачи из буфера 0.
7. Перейти к шагу 1.

Пример реализации алгоритма – тест *nmttcantesttrx*.

9.1 IOCTL_WRITE_DATA_TO_FIFO_TTCAN1_V2

Назначение:

Отправка данных в режиме FIFO для канала 1.

Действие:

Данные попадают в очередь на отправку.

Примечание:

-

Входные параметры:

param – указатель на структуру типа CAN_WRT_MSG_V2:

Поле структуры	Описание
unsigned char mRtrBit	наличие RTR бита
unsigned char mTypeFifo	тип FIFO (0 –FIFO, 1-HPFIFO)
unsigned char mIsExtMsg	признак расширенного сообщения
unsigned int mSID	Стандартный идентификатор (допустимые значения – 0…7FFH)
unsigned int mEID	Расширенный идентификатор (допустимые значения – 0…3FFFFH и специальное значение FFFFFFFFH, которое означает, что расширенный идентификатор не используется)
unsigned char mDataLength	фактический размер данных (0-8)
unsigned char mData	Данные пакета
unsigned char mMsgId	уникальный идентификатор сообщения

Пример вызова:

```
int fd, ret;
    CAN_WRT_MSG_V2 sd;
fd = open("/dev/can_dev_0", O_RDWR);
sd. mRtrBit = 0;
sd. mTypeFifo = 0;
sd. mIsExtMsg = 1;
sd.mSID = 0x7FF;
sd.mEID = 0x3FFFF;
sd.mData[0] = 0xFF;
sd. mDataLength =1;
ret = ioctl(fd, IOCTL_WRITE_DATA_TO_FIFO_CAN1_V2, &sd);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u data buffer %u was filled with data\n", sd.nChannel, sd.nBufNumber);
```

9.2 IOCTL_WRITE_DATA_TO_FIFO_TTCAN2_V2

Назначение:
Отправка данных в режиме FIFO для канала 2.

Действие:
Данные попадают в очередь на отправку.

Примечание:
-

Входные параметры:
param – указатель на структуру типа CAN_WRT_MSG_V2:

Поле структуры	Описание
unsigned char mRtrBit	наличие RTR бита
unsigned char mTypeFifo	тип FIFO (0 –FIFO, 1-HPFIFO)
unsigned char mIsExtMsg	признак расширенного сообщения
unsigned int mSID	Стандартный идентификатор (допустимые значения – 0…7FFH)
unsigned int mEID	Расширенный идентификатор (допустимые значения – 0…3FFFFH и специальное значение FFFFFFFFH, которое означает, что расширенный идентификатор не используется)
unsigned char mDataLength	фактический размер данных (0-8)
unsigned char mData	Данные пакета
unsigned char mMsgId	уникальный идентификатор сообщения

Пример вызова:

```
int fd, ret;
    CAN_WRT_MSG_V2 sd;
fd = open("/dev/can_dev_0", O_RDWR);
sd. mRtrBit = 0;
sd. mTypeFifo = 0;
sd. mIsExtMsg = 1;
sd.mSID = 0x7FF;
sd.mEID = 0x3FFFF;
sd.mData[0] = 0xFF;
sd. mDataLength =1;
ret = ioctl(fd, IOCTL_WRITE_DATA_TO_FIFO_CAN1_V2, &sd);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u data buffer %u was filled with data\n", sd.nChannel, sd.nBufNumber);
```

9.3IOCTL_WRITE_DATA_TO_FIFO_TTCAN1

Назначение:
Отправка данных в режиме FIFO для канала 1.

Действие:
Данные попадают в очередь на отправку.

Примечание:
- Функция сохранена для совместимости с предыдущей версией. Для нового ПО используйте IOCTL_WRITE_DATA_TO_FIFO_TTCAN1_V2, где добавлен бит RTR.

Входные параметры:
param – указатель на структуру типа CAN_WRT_MSG:

Поле структуры	Описание
unsigned char mTypeFifo	тип FIFO (0 –FIFO, 1-HPFIFO)
unsigned char mIsExtMsg	признак расширенного сообщения
unsigned int mSID	Стандартный идентификатор (допустимые значения – 0…7FFH)
unsigned int mEID	Расширенный идентификатор (допустимые значения – 0…3FFFFH и специальное значение FFFFFFFFH, которое означает, что расширенный идентификатор не используется)
unsigned char mDataLength	фактический размер данных (0-8)
unsigned char mData	Данные пакета
unsigned char mMsgId	уникальный идентификатор сообщения

Пример вызова:

```
int fd, ret;
    CAN_WRT_MSG sd;
fd = open("/dev/can_dev_0", O_RDWR);
sd. mTypeFifo = 0;
sd. mIsExtMsg = 1;
sd.mSID = 0x7FF;
sd.mEID = 0x3FFFF;
sd.mData[0] = 0xFF;
sd. mDataLength =1;
ret = ioctl(fd, IOCTL_WRITE_DATA_TO_FIFO_CAN1, &sd);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u data buffer %u was filled with data\n", sd.nChannel, sd.nBufNumber);
```

9.4 IOCTL_WRITE_DATA_TO_FIFO_TTCAN2

Назначение:
Отправка данных в режиме FIFO для канала 2.

Действие:
Данные попадают в очередь на отправку.

Примечание:
- Функция сохранена для совместимости с предыдущей версией. Для нового ПО используйте IOCTL_WRITE_DATA_TO_FIFO_TTCAN2_V2, где добавлен бит RTR.

Входные параметры:
param – указатель на структуру типа CAN_WRT_MSG:

Поле структуры	Описание
unsigned char mTypeFifo	тип FIFO (0 –FIFO, 1-HPFIFO)
unsigned char mIsExtMsg	признак расширенного сообщения
unsigned int mSID	Стандартный идентификатор (допустимые значения – 0…7FFH)
unsigned int mEID	Расширенный идентификатор (допустимые значения – 0…3FFFFH и специальное значение FFFFFFFFH, которое означает, что расширенный идентификатор не используется)
unsigned char mDataLength	фактический размер данных (0-8)
unsigned char mData	Данные пакета
unsigned char mMsgId	уникальный идентификатор сообщения

Пример вызова:

```
int fd, ret;
    CAN_WRT_MSG sd;
fd = open("/dev/can_dev_0", O_RDWR);
sd. mTypeFifo = 0;
sd. mIsExtMsg = 1;
sd.mSID = 0x7FF;
sd.mEID = 0x3FFFF;
sd.mData[0] = 0xFF;
sd. mDataLength =1;
ret = ioctl(fd, IOCTL_WRITE_DATA_TO_FIFO_CAN1, &sd);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u data buffer %u was filled with data\n", sd.nChannel, sd.nBufNumber);
```

9.5 IOCTL_WRITE_DATA_TO_TR_BUF_TTCAN

Назначение:
Запись данных в буфер отправки.

Действие:
Данные из входных полей последовательно записываются в регистры **TXBn*CTRL**, **TXBn*SIDH**, **TXBn*SIDL**, **TXBn*EID8**, **TXBn*EID0**, **TXBn*DLC**, **TXBn*Dm** **.
Все функции работы с буферами контроллера CAN защищены общим семафором (индивидуальным для каждого канала и буфера) для возможности их корректного использования из параллельных процессов.

* n – номер буфера.
** См. раздел 6.7.1-6.7.7 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

Примечание:
-

Входные параметры:
param – указатель на структуру типа SEND_DATA:

Поле структуры	Описание
uint8_t **nChannel**	Номер канала (допустимые значения – 1…2)
uint8_t **nBufNumber**	Номер буфера, в который будут записываться данные (допустимые значения – 0…2)
uint8_t **nPriority**	Приоритет буфера передачи (допустимые значения – 0…3; 0 – низший, 3 – наивысший)
uint32_t **SID**	Стандартный идентификатор (допустимые значения – 0…7FFH)
uint32_t **EID**	Расширенный идентификатор (допустимые значения – 0…3FFFFH и специальное значение FFFFFFFFH, которое означает, что расширенный идентификатор не используется)
uint8_t **nData[8]**	Буфер с данными
uint32_t **nSize**	Размер передаваемых данных

Пример вызова:

```
int fd, ret;
    SEND_DATA sd;
fd = open("/dev/ttcan_dev_0", O_RDWR);
sd.nChannel = 1;
sd.nBufNumber = 0;
sd.nPriority = 0;
sd.SID = 0x7FF;
sd.EID = 0x3FFFF;
sd.nData[0] = 0xFF;
sd.nSize=Size;
ret = ioctl(fd, IOCTL_WRITE_DATA_TO_TR_BUF_TTCAN, &sd);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u data buffer %u was filled with data\n", sd.nChannel, sd.nBufNumber);
```

9.6 IOCTL_SEND_DATA_TTCAN

Назначение:
Синхронная отправка сообщения.

Действие:
Данные из входных полей последовательно записываются в регистры **TXBn*CTRL**, **TXBn*SIDH**, **TXBn*SIDL**, **TXBn*EID8**, **TXBn*EID0**, **TXBn*DLC**, **TXBn*Dm** **. После этого в бит TXREQ регистра **TXBn*CTRL**** будет записана единица и процесс перейдёт в состояние ожидания, из которого он будет выведен либо после прерывания, сигнализирующего об окончании отправки данных, либо по истечении максимального времени ожидания отправки. В последнем случае функция возвратит -1, в errno будет установлено значение ETIME, а в поле **txb_ctrl** будет записано значение регистра **TXBn*CTRL**.
Все функции работы с буферами контроллера CAN защищены общим семафором (индивидуальным для каждого канала и буфера) для возможности их корректного использования из параллельных процессов.

* n – номер буфера.
** См. раздел 6.7.1-6.7.7 документа "*Руководство по программированию модуля "xPCIe-TTCAN*".

Примечание:
-

Входные параметры:
param – указатель на структуру типа SEND_DATA:

Поле структуры	Описание
uint8_t **nChannel**	Номер канала (допустимые значения – 1…2)
uint8_t **nBufNumber**	Номер буфера, в который будут записываться данные (допустимые значения – 0…2)
uint8_t **nPriority**	Приоритет буфера передачи (допустимые значения – 0…3; 0 – низший, 3 – наивысший)
TXBCTRL **txb_ctrl**	Значение регистра **TXBn*CTRL** в случае истечения времени ожидания отправки сообщения
uint32_t **timeout**	Максимальное время ожидания отправки сообщения, в миллисекундах
uint32_t **SID**	Стандартный идентификатор (допустимые значения – 0…7FFH)
uint32_t **EID**	Расширенный идентификатор (допустимые значения – 0…3FFFFH и специальное значение FFFFFFFFH, которое означает, что расширенный идентификатор не используется)
uint8_t **nData[8]**	Буфер с данными
uint32_t **nSize**	Размер передаваемых данных

Пример вызова:

```
int fd, ret;
    SEND_DATA sd;
fd = open("/dev/ttcan_dev_0", O_RDWR);
sd.nChannel = 1;
sd.nBufNumber = 0;
sd.nPriority = 0;
    sd.timeout = 100;
```

```
sd.SID = 0x7FF;
sd.EID = 0x3FFFF;
sd.nData[0] = 0xFF;
sd.nSize=Size;
ret = ioctl(fd, IOCTL_SEND_DATA _TTCAN, &sd);
if (ret == -1)
{
printf("ioctl error: %d (%s); TXB%uCTRL=%02XH\n", errno, strerror(errno), sd.nBufNumber,
    sd.txb_ctrl);
}
else
printf("CAN%u data buffer %u was filled with data\n", sd.nChannel, sd.nBufNumber);
```

9.7 IOCTL_SEND_DATA_NOW_TTCAN

Назначение:
Запуск асинхронной передачи сообщения.

Действие:
Бит TXREQ регистра **TXBn*CTRL**** устанавливается в единицу.
Все функции работы с буферами контроллера CAN защищены общим семафором (индивидуальным для каждого канала и буфера) для возможности их корректного использования из параллельных процессов.

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

Примечание:
-

Входные параметры:
param – указатель на структуру типа SEND_DATA_NOW:

Поле структуры	Описание
uint8_t **nChannel**	Номер канала (допустимые значения – 1…2)
uint8_t **nBuf**	Номер буфера, который будет установлен на отправку (допустимые значения – 0…2)

Пример вызова:

```
int fd, ret;
    SEND_DATA_NOW sdn;
fd = open("/dev/ttcan_dev_0", O_RDWR);
sdn.nChannel = 1;
sdn.nBuf = 0;
ret = ioctl(fd, IOCTL_SEND_DATA_NOW _TTCAN, &sdn);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u data buffer %u was requested to send\n", sdn.nChannel, sdn.nBuf);
```

9.8 IOCTL_CHECK_TRANSMIT_TTCAN

Назначение:
Проверка правильности отправки сообщения.

Действие:
Данная функция читает регистр **TXBn*CTRL**** в поле **txb_ctrl**. Если бит TXREQ окажется равен нулю, то функция возвращает 0. В противном случае функция возвратит значение -1, а в errno будет установлено значение EBUSY.

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

Примечание:
-

Входные параметры:
param – указатель на структуру типа SEND_DATA_NOW:

Поле структуры	Описание
uint8_t **nChannel**	Номер канала (допустимые значения – 1…2)
uint8_t **nBuf**	Номер буфера, который будет проверен (допустимые значения – 0…2)
TXBCTRL **txb_ctrl**	Значение регистра **TXBn*CTRL**

Пример вызова:

```
int fd, ret;
SEND_DATA_NOW sdn;
fd = open("/dev/ttcan_dev_0", O_RDWR);
sdn.nChannel = 1;
sdn.nBuf = 0;
ret = ioctl(fd, IOCTL_CHECK_TRANSMIT_TTCAN, &sdn);
if (ret == -1)
{
printf("ioctl error: %d (%s); TXB%uCTRL=%02XH\n", errno, strerror(errno), sdn.nBuf,
    sdn.txb_ctrl);
}
else
printf("CAN%u data buffer %u was sended\n", sdn.nChannel, sdn.nBuf);
```

9.9 IOCTL_WAIT_TRANSMIT_TTCAN

Назначение:
Ожидание отправки сообщения.

Действие:
Данная функция переводит процесс в состояние ожидания, из которого он будет выведен либо после прерывания, сигнализирующего об окончании отправки данных, либо по истечении максимального времени ожидания отправки. В последнем случае функция возвратит -1, в errno будет установлено значение ETIME, а в поле **txb_ctrl** будет записано значение регистра **TXBn*CTRL**.
Функция защищена общим с функциями работы с буферами контроллера CAN семафором (индивидуальным для каждого канала и буфера) для возможности их корректного использования из параллельных процессов.

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

Примечание:
-

Входные параметры:
param – указатель на структуру типа SEND_DATA_NOW:

Поле структуры	Описание
uint8_t **nChannel**	Номер канала (допустимые значения – 1…2)
uint8_t **nBuf**	Номер буфера, чья отправка будет ожидаться (допустимые значения – 0…2)
TXBCTRL **txb_ctrl**	Значение регистра **TXBn*CTRL**
uint32_t **timeout**	Максимальное время ожидания отправки сообщения, в миллисекундах

Пример вызова:

```
int fd, ret;
SEND_DATA_NOW sdn;
fd = open("/dev/ttcan_dev_0", O_RDWR);
sdn.nChannel = 1;
sdn.nBuf = 0;
sdn.timeout = 100;
ret = ioctl(fd, IOCTL_WAIT_TRANSMIT_TTCAN, &sdn);
if (ret == -1)
{
printf("ioctl error: %d (%s); TXB%uCTRL=%02XH\n", errno, strerror(errno), sdn.nBuf,
    sdn.txb_ctrl);
}
else
printf("CAN%u data buffer %u was sended\n", sdn.nChannel, sdn.nBuf);
```

9.10 IOCTL_END_TRANSMIT_TTCAN

Назначение:
Снятие запроса отправки сообщения.

Действие:
Данная функция записывать ноль в бит TXREQ регистра **TXBn*CTRL****.

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

Примечание:
-

Входные параметры:
param – указатель на структуру типа SEND_DATA_NOW:

Поле структуры	Описание
uint8_t **nChannel**	Номер канала (допустимые значения – 1…2)
uint8_t **nBuf**	Номер буфера, с которого будет снят запрос на отправку (допустимые значения – 0…2)

Пример вызова:

```
int fd, ret;
SEND_DATA_NOW sdn;
fd = open("/dev/ttcan_dev_0", O_RDWR);
sdn.nChannel = 1;
sdn.nBuf = 0;
ret = ioctl(fd, IOCTL_END_TRANSMIT_TTCAN, &sdn);
if (ret == -1)
printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u data buffer %u request to send was cleared\n", sdn.nChannel, sdn.nBuf);
```

9.11 IOCTL_SEND_DATA_TG_TTCAN

Назначение:
Запуск однократной передачи данных из буфера передачи по срабатыванию триггера.

Действие:
Значение триггера, находящееся в поле **nTrigger**, записывается в регистр **CANn*_TXm**_TRIG** *** (значение записывается целиком с нулевого бита).
Если поле **bEpoch** не равно нулю, то значение из поля **nEpoch** записывается в регистр **CANn*_TXm**_TRIG_EPOCH** **** (значение записывается целиком с нулевого бита), а бит TX_TRIGm**_EPOCH_EN регистра **CANn*_TRIG_CTRL** ***** устанавливается в единицу. Бит TX_TRIGm**_RPT регистра **CANn*_TRIG_CTRL** ***** устанавливается в ноль, а биты TX_TRIGm**_EN устанавливаются в “01”.

* n – номер канала.
** m – номер буфера.
*** См. раздел 5.3.1 или 5.3.2 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.
**** См. раздел 5.3.6 или 5.3.7 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.
***** См. раздел 5.3.3 или 5.3.4 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.

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

Входные параметры:
param – указатель на структуру типа SEND_DATA_TG:

Поле структуры	Описание
uint8_t **nChannel**	Номер канала (допустимые значения – 1…2)
uint8_t **nBuf**	Номер буфера, который будет установлен на отправку (допустимые значения – 0…2)
uint8_t **bEpoch**	Флаг использования счётчика циклов
uint32_t **nEpoch**	Значение регистра **CANn*_TXm**_TRIG_EPOCH** ****
uint32_t **nTrigger**	Значение триггера в регистре **CANn*_TXm**_TRIG** ***

Пример вызова:

```
int fd, ret;
SEND_DATA_TG sdt;
fd = open("/dev/ttcan_dev_0", O_RDWR);
sdt.nChannel = 1;
sdt.nBuf = 0;
sdt.bEpoch = 1;
sdt.nEpoch = 1;
sdt.nTrigger = (1 * 16 * 8) << 16;
ret = ioctl(fd, IOCTL_SEND_DATA_TG_TTCAN, &sdt);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
  printf("CAN%u data buffer %u was requested to send by trigger\n", sdt.nChannel, sdt.nBuf);
```

9.12 IOCTL_SEND_DATA_LOOP_TTCAN

Назначение:
Запуск многократной передачи данных из буфера передачи по срабатыванию триггера.

Действие:
Значение триггера, находящееся в поле **nTrigger**, записывается в регистр **CANn*_TXm**_TRIG** *** (значение записывается целиком с нулевого бита).
Если поле **bEpoch** не равно нулю, то значение из поля **nEpoch** записывается в регистр **CANn*_TXm**_TRIG_EPOCH** **** (значение записывается целиком с нулевого бита), а бит TX_TRIGm**_EPOCH_EN регистра **CANn*_TRIG_CTRL** ***** устанавливается в единицу.
Бит TX_TRIGm**_RPT регистра **CANn*_TRIG_CTRL** ***** устанавливается в единицу, а биты TX_TRIGm**_EN устанавливаются в “01”.

* n – номер канала.
** m – номер буфера.
*** См. раздел 5.3.1 или 5.3.2 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.
**** См. раздел 5.3.6 или 5.3.7 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.
***** См. раздел 5.3.3 или 5.3.4 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.

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

Входные параметры:
param – указатель на структуру типа SEND_DATA_TG:

Поле структуры	Описание
uint8_t **nChannel**	Номер канала (допустимые значения – 1…2)
uint8_t **nBuf**	Номер буфера, который будет установлен на отправку (допустимые значения – 0…2)
uint8_t **bEpoch**	Флаг использования счётчика циклов
uint32_t **nEpoch**	Значение регистра **CANn*_TXm**_TRIG_EPOCH** ****
uint32_t **nTrigger**	Значение триггера в регистре **CANn*_TXm**_TRIG** ***

Пример вызова:

```
int fd, ret;
SEND_DATA_TG sdt;
fd = open("/dev/ttcan_dev_0", O_RDWR);
sdt.nChannel = 1;
sdt.nBuf = 0;
sdt.bEpoch = 1;
sdt.nEpoch = 1;
sdt.nTrigger = (1 * 16 * 8) << 16;
ret = ioctl(fd, IOCTL_SEND_DATA_LOOP_TTCAN, &sdt);
if (ret == -1)
  printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
  printf("CAN%u data buffer %u was requested to send by trigger\n", sdt.nChannel, sdt.nBuf);
```

9.13 IOCTL_CHECK_TG_TTCAN

Назначение:
Проверка триггера.

Действие:
Функция читает регистр **CANn*_TRIG_CTRL***** и проверяет биты **TX_TRIGm**_EN**.
Если биты равны “01”, то триггер занят, функция вернёт значение -1, а в errno будет установлено значение EBUSY.
Если биты равны “00”, то триггер свободен и функция вернёт значение 0.

* n – номер канала.
** m – номер триггера.
***См. раздел 5.3.3-5.3.4 документа “*Руководство по программированию модуля “xPCIe-TTCAN*”.

Примечание:
-

Входные параметры:
param – указатель на структуру типа CH_BUF_TTCAN:

Поле структуры	Описание
uint8_t **nCh**	Номер канала (допустимые значения – 1…2)
uint8_t **nBuf**	Номер триггера, который будет проверен (допустимые значения – 0…2)

Пример вызова:

```
int fd, ret;
CH_BUF_TTCAN cb;
fd = open("/dev/ttcan_dev_0", O_RDWR);
cb.nCh = 1;
cb.nBuf = 0;
ret = ioctl(fd, IOCTL_CHECK_TG_TTCAN, &cb);
if (ret == -1)
   printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
  printf("CAN%u trigger %u is free\n", cb.nCh, cb.nBuf);
```

9.14 IOCTL_ABAT_TTCAN

Назначение:
Остановка всех активных передач.

Действие:
Функция устанавливает бит ABAT регистра **CAN_CTRL*** (адрес Fh) в соответствии со значением поля **mode**.

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

Примечание:
-

Входные параметры:
param – указатель на структуру типа CONF_TTCAN:

Поле структуры	Описание
uint32_t **channel**	Номер канала (допустимые значения – 1…2)
uint8_t **mode**	0 – снятие запроса на остановку всех активных передач. Не 0 - запрос на остановку всех активных передач

Пример вызова:

```
int fd, ret;
CONF_TTCAN conf;
fd = open("/dev/ttcan_dev_0", O_RDWR);
conf.channel = 1;
conf.mode = 1;
ret = ioctl(fd, IOCTL_ABAT_TTCAN, &conf);
if (ret == -1)
   printf("ioctl error: %d (%s)\n", errno, strerror(errno));
else
  printf("CAN%u ABAT was %s\n", conf.channel, (conf.mode != 0) ? "set" : "reset");
```

10. Обновление драйвера

Версия библиотеки	Дата	Изменение
2.0	31.05.2018	- Драйвер создан
3.0	29.07.2020	- Добавлен режим FIFO.
3.1	19.07.2021	- Добавлена обработка бита RTR в функциях передачи.

11. Обновление руководства.

Версия документа	Дата	Изменение
2.0	31.05.2018	- Документ создан
3.0	29.07.2020	- Добавлено описание функций для режима FIFO.
3.1	21.07.2021	- Добавлена обработка бита RTR в функциях передачи.
3.2	08.12.2021	- Исправлены незначительные ошибки.