



Руководство (v3.2)

**По работе с библиотекой модулей
“mPCIE – TTCAN”, “PCIE – TTCAN”**

Интерфейс ISO-11898-4
(TTCAN)

Для библиотек версии 3.1

ОС LINUX



08.12.2021

ООО “Новомар”

Оглавление

1	Введение.....	4
2	Сборка библиотеки.....	4
3	Описание использования функций библиотек.....	7
4	Стандартные функции.....	8
4.1	NM_TTCAN_GetDevName.....	8
4.2	NM_TTCAN_GetDevPath.....	9
4.3	NM_TTCAN_WriteReg.....	10
4.4	NM_TTCAN_ReadReg.....	11
4.5	NM_TTCAN_WriteCanReg.....	12
4.6	NM_TTCAN_ReadCanReg.....	13
4.7	NM_TTCAN_ModifyCanReg.....	14
4.8	NM_TTCAN_WriteCanRegs.....	15
4.9	NM_TTCAN_ReadCanRegs.....	16
4.10	NM_TTCAN_GetVersion.....	17
4.11	NM_TTCAN_GetDriverVersion.....	18
5	Функции конфигурации.....	19
5.1	NM_TTCAN_EnableDMA.....	19
5.2	NM_TTCAN_DisableDMA.....	20
5.3	NM_TTCAN_ResetTrigger.....	21
5.4	NM_TTCAN_SetMode.....	22
5.5	NM_TTCAN_GetMode.....	23
5.6	NM_TTCAN_SetOneshotMode.....	24
5.7	NM_TTCAN_SetSpeed.....	25
5.8	NM_TTCAN_SetSpeedByParams.....	26
5.9	NM_TTCAN_GetErrors.....	27
5.10	NM_TTCAN_SetMasks.....	28
5.11	NM_TTCAN_SetTimerThreshold.....	30
5.12	NM_TTCAN_SetTimerCEED.....	31
5.13	NM_TTCAN_SetTimerFree.....	32
5.14	NM_TTCAN_SetTimerResetOnRXB.....	33
5.15	NM_TTCAN_StopTimer.....	34
5.16	NM_TTCAN_GetTimer.....	35
5.17	NM_TTCAN_StartTimerInt.....	36
5.18	NM_TTCAN_StopTimerInt.....	37
5.19	NM_TTCAN_WaitTimerInt.....	38
5.20	NM_TTCAN_SetTimeouts.....	39
5.21	NM_TTCAN_Reset.....	40
5.22	NM_TTCAN_ResetCAN.....	41
5.23	NM_TTCAN_SetTxPause.....	42
6	Функции для чтения принятых данных.....	43
6.1	NM_TTCAN_ReadChannelRawDMA.....	43
6.2	NM_TTCAN_ReadDMA.....	44
6.3	NM_TTCAN_DecodeBuf.....	45
7	Функции для передачи данных.....	46
7.1	NM_TTCAN_SetSendMode.....	48
7.2	NM_TTCAN_GetCountMsgInFifo.....	49
7.3	NM_TTCAN_GetCountMsgInHpFifo.....	50
7.4	NM_TTCAN_WriteDataToFifo.....	51
7.5	NM_TTCAN_WriteDataToHpFifo.....	52
7.6	NM_TTCAN_WriteDataToTgFifo.....	53
7.7	NM_TTCAN_WriteDataToSendBuf.....	54
7.8	NM_TTCAN_SendData.....	55
7.9	NM_TTCAN_SendDataRightNow.....	57

<u>Описание функций библиотеки</u>	<u>xPCIe-TTCAN</u>
7.10 NM_TTCAN_CheckTransmit	58
7.11 NM_TTCAN_WaitTransmit.....	59
7.12 NM_TTCAN_EndTransmit	60
7.13 NM_TTCAN_SendDataByTrigger.....	61
7.14 NM_TTCAN_SendDataByTriggerAuto.....	62
7.15 NM_TTCAN_CheckTrigger.....	63
7.16 NM_TTCAN_ABAT	64
8 Обновление библиотеки	65
9 Обновление руководства	66

1 Введение.

Библиотека поддерживает модули: “**mPCIe-TTCAN**”, “**PCIe-TTCAN**”(далее **xPCIe-TTCAN**).
Взаимодействие с библиотекой происходит посредством функций, перечень которых находится в файле “libnmttcn.h”.

2 Сборка библиотеки.

1. Создайте отдельную папку.
2. Скачайте в эту папку архив с исходными текстами драйвера и распакуйте его.
3. Скачайте в эту папку архив с исходными текстами библиотеки и распакуйте его.
4. Выполните команду “cd mPCIe-TTCAN/lib” для перехода в каталог с исходными текстами библиотеки.
5. Выполните команду “make”.

1. Список доступных функций по версиям библиотек.

Название вызова	Краткое описание
Список функций, доступных в библиотеке версии 2.0	
NM_TTCAN_GetDevName	Получить имя устройства по его номеру.
NM_TTCAN_GetDevPath	Получить полный маршрут устройства по его номеру.
NM_TTCAN_WriteReg	Запись регистра модуля.
NM_TTCAN_ReadReg	Чтение регистра модуля.
NM_TTCAN_WriteCanReg	Запись регистра контроллера канала.
NM_TTCAN_ReadCanReg	Чтение регистра контроллера канала.
NM_TTCAN_ModifyCanReg	Модификация регистра контроллера канала.
NM_TTCAN_WriteCanRegs	Запись регистров контроллера канала.
NM_TTCAN_ReadCanRegs	Чтение регистров контроллера канала.
NM_TTCAN_GetVersion	Информация о плате.
NM_TTCAN_GetDriverVersion	Информация о драйвере.
NM_TTCAN_EnableDMA	Разрешение работы DMA.
NM_TTCAN_DisableDMA	Запрет работы DMA.
NM_TTCAN_ResetTrigger	Сброс триггера канала.
NM_TTCAN_SetMode	Установка режима работы канала.
NM_TTCAN_GetMode	Чтение режима работы канала.
NM_TTCAN_SetOneshotMode	Установка/снятие режима однократной попытки отправки сообщения для канала.
NM_TTCAN_SetSpeed	Установка скорости работы канала.
NM_TTCAN_SetSpeedByParams	Установка специальной скорости работы канала.
NM_TTCAN_GetErrors	Чтение регистров ошибок канала.
NM_TTCAN_SetMasks	Установка масок и фильтров на прием сообщений канала.
NM_TTCAN_SetTimerThreshold	Запуск таймера канала с ограничением счёта.
NM_TTCAN_SetTimerCEED	Установка начального значения таймера канала при сбросе.
NM_TTCAN_SetTimerFree	Запуск таймера канала в режиме свободного счёта.
NM_TTCAN_SetTimerResetOnRXB	Установка сброса таймера канала по приёму сообщения в буфер RXBn.
NM_TTCAN_StopTimer	Остановка таймера канала.
NM_TTCAN_GetTimer	Чтение текущего значения таймера канала.
NM_TTCAN_StartTimerInt	Установка прерывания по таймеру канала.
NM_TTCAN_StopTimerInt	Остановка прерывания по таймеру канала.

NM_TTCAN_WaitTimerInt	Ожидание прерывания по таймеру канала.
NM_TTCAN_SetTimeouts	Установка абсолютного и интервального таймера прерываний канала.
NM_TTCAN_Reset	Сброс модуля.
NM_TTCAN_ResetCAN	Сброс канала.
NM_TTCAN_ReadChannelRawDMA	Чтение данных из DMA канала.
NM_TTCAN_WriteDataToSendBuf	Запись данных в буфер отправки канала.
NM_TTCAN_SendData	Запись данных в буфер отправки канала, запуск передачи сообщения и ожидание завершения передачи.
NM_TTCAN_SendDataRightNow	Запуск передачи сообщения из буфера отправки канала.
NM_TTCAN_CheckTransmit	Проверка завершения и правильности передачи сообщения из буфера отправки канала.
NM_TTCAN_WaitTransmit	Ожидание завершения и проверка правильности передачи сообщения из буфера отправки канала.
NM_TTCAN_EndTransmit	Снятие запроса на передачу сообщения из буфера отправки канала.
NM_TTCAN_SendDataByTrigger	Запуск однократной передачи сообщения из буфера отправки канала по значению триггера.
NM_TTCAN_SendDataByTriggerAuto	Запуск повторяющейся передачи сообщения из буфера отправки канала по значению триггера.
NM_TTCAN_CheckTrigger	Проверка срабатывания триггера.
NM_TTCAN_ABAT	Установка/снятие режима остановки всех активных передач.
NM_TTCAN_DecodeBuf	Декодирование пакета, полученного из буфера DMA.
Список функций, добавленных в библиотеке версии 3.0	
NM_TTCAN_ReadDMA	Чтение данных из буфера DMA
NM_TTCAN_SetSendMode	Устанавливает режим передачи канала
NM_TTCAN_GetCountMsgInFifo	Получает текущее кол-во сообщений в FIFO
NM_TTCAN_GetCountMsgInHpFifo	Получает текущее кол-во сообщений в HPFIFO
NM_TTCAN_WriteDataToFifo	Записать сообщение в FIFO на отправку
NM_TTCAN_WriteDataToHpFifo	Записать сообщение в HPFIFO на отправку
NM_TTCAN_SetTxPause	Управление флагом TX_Pause
NM_TTCAN_WriteDataToTgFifo	Записать данные о времени отправки в TGFIFO на отправку

3 Описание использования функций библиотек.

Функции [NM_TTCAN_GetDevName](#) и [NM_TTCAN_GetDevPath](#) возвращают число байт, записанных в буфер.

Остальные функции, в случае удачного выполнения запроса, возвращают 0. Функция, 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.

4 Стандартные функции

4.1 NM_TTCAN_GetDevName

Назначение:

Получение имени устройства по его номеру.

Действие:

Функция записывает в **psz** имя устройства, не переходя за границу буфера **strsize**.

Примечание:

-

Аргументы функции:

Аргумент	Описание
uint32_t devnum	Номер устройства
char* psz	Указатель на буфер, в который будет записано имя устройства
uint32_t strsize	Размер буфера

Пример вызова:

```
char devname[32];
uint32_t devnamelen;
devnamelen = NM_TTCAN_GetDevName(0, devname, sizeof(devname));
if (devnamelen > 0)
    printf("devname: '%s'\n", devname);
```

4.2 NM_TTCAN_GetDevPath

Назначение:

Получение полного маршрута устройства по его номеру.

Действие:

Функция записывает в **psz** полный маршрут к устройству, не переходя за границу буфера **strsize**.

Примечание:

-

Аргументы функции:

Аргумент	Описание
uint32_t devnum	Номер устройства
char* psz	Указатель на буфер, в который будет записан полный маршрут к устройству
uint32_t strsize	Размер буфера

Пример вызова:

```
char devpath[32];
uint32_t devpathlen;
devpathlen = NM_TTCAN_GetDevPath(0, devpath, sizeof(devpath));
if (devpathlen > 0)
    printf("devpath: '%s'\n", devpath);
```

4.3 NM_TTCAN_WriteReg

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!!!

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t addr	Адрес регистра
uint32_t data	Данные для записи

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_WriteReg(fd, 0x2000, 0);
if (ret == -1)
printf("error: %d (%s)\n", errno, strerror(errno));
```

4.4 NM_TTCAN_ReadReg

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!

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t addr	Адрес регистра
uint32_t* pdata	Указатель на возвращаемый результат

Пример вызова:

```
int fd, ret;
uint32_t data;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_ReadReg(fd, 0x2000, &data);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
```

4.5 NM_TTCAN_WriteCanReg

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 в драйвере 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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint8_t addr	Адрес регистра
uint8_t data	Данные для записи

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_WriteCanReg(fd, 1, CANINTE, 0);
if (ret == -1)
printf("error: %d (%s)\n", errno, strerror(errno));
```

4.6 NM_TTCAN_ReadCanReg

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 в драйвере 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:

-

Аргументы функции:

Аргумент	Описание
<code>int fd</code>	Дескриптор устройства, полученный от функции <code>open()</code>
<code>uint32_t channel</code>	Номер канала (допустимые значения – 1...2)
<code>uint8_t addr</code>	Адрес регистра
<code>uint8_t* pdata</code>	Указатель на возвращаемый результат

Пример вызова:

```
int fd, ret;
uint8_t data;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_ReadCanReg(fd, 1, CAN_CTRL, &data);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
```

4.7 NM_TTCAN_ModifyCanReg

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 в драйвере 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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint8_t addr	Адрес регистра
uint8_t mask	Маска для указания модифицируемых битов регистра
uint8_t data	Новое значение для модифицируемых битов регистра

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_ModifyReg(fd, 1, TXB0CTRL, 0x08, 0x08);
if (ret == -1)
printf("error: %d (%s)\n", errno, strerror(errno));
```

4.8 NM_TTCAN_WriteCanRegs

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 в драйвере 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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint8_t addr	Адрес первого регистра
uint8_t* pdata	Указатель на записываемые данные
uint32_t datasize	Количество записываемых данных (допустимые значения – 1...16)

Пример вызова:

```
int fd, ret;
uint8_t data[] = { 0x00, 0x00, 0x00, 0x00, 0x01, 0xFF };
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_WriteCanRegs(fd, 1, TXB0CTRL + 1, data, sizeof(data));
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
```

4.9 NM_TTCAN_ReadCanRegs

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 в драйвере 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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint8_t addr	Адрес первого регистра
uint8_t* pdata	Указатель на считываемые данные
uint8_t datasize	Количество считываемых данных (допустимые значения – 1...16)

Пример вызова:

```
int fd, ret;
uint8_t data[6];
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_ReadCanRegs(fd, 1, TXB0CTRL + 1, data, sizeof(data));
if (ret == -1)
printf("error: %d (%s)\n", errno, strerror(errno));
```

4.10 NM_TTCAN_GetVersion

Назначение:

Чтение информации о модуле.

Действие:

Функция заполняет все поля структуры **VERSION_TTCAN**.

Примечание:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции <code>open()</code>
VERSION_TTCAN* p version	Указатель на возвращаемую структуру VERSION_TTCAN

Пример вызова:

```
int fd, ret;  
VERSION_TTCAN ver;  
fd = open("/dev/ttcan_dev_0", O_RDWR);  
ret = NM_TTCAN_GetVersion(fd, &ver);  
if (ret == -1)  
printf("error: %d (%s)\n", errno, strerror(errno));
```

4.11 NM_TTCAN_GetDriverVersion

Назначение:

Чтение версии и даты драйвера.

Действие:

Функция заполняет 32-битную переменную.

Примечание:

-

Аргументы функции:

Аргумент	Описание
<code>int fd</code>	Дескриптор устройства, полученный от функции <code>open()</code>
<code>uint32_t* pdrv_version</code>	Указатель на возвращаемое значение, в котором 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 = NM_TTCAN_GetDriverVersion(fd, &drv_ver_date);
if (ret == -1)
    printf("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);
```

5 Функции конфигурации

5.1 NM_TTCAN_EnableDMA

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.

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()

Пример вызова:

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

5.2 NM_TTCAN_DisableDMA

Назначение:

Запрет работы DMA.

Действие:

Нулевой бит регистра **DMA_DATA_BASE*** (адрес 1000h) и биты **VnBFM**, **VnBFE** и **VnBFS** регистра **BFCTRL**** (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.

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()

Пример вызова:

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

5.3 NM_TTCAN_ResetTrigger

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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
CAN_BUF nbuf	Номер буфера (равен номеру триггера)

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_ResetTrigger(fd, 1, 2);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("Trigger %u reset on CAN%u \n", 2, 1);
```

5.4 NM_TTCAN_SetMode

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 [NM_TTCAN_GetMode](#).

* 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”.

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
CAN_MODE mode	Новый режим работы контроллера

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_SetMode (fd, 1, CAN_WORK);
if (ret == -1)
printf("error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u mode set to %u\n", 1, CAN_WORK);
```

5.5 NM_TTCAN_GetMode

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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
CAN_MODE* pmode	Указатель на текущий режим работы контроллера

Пример вызова:

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

5.6 NM_TTCAN_SetOneshotMode

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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
int osm	0 – снятие однократного режима. Не 0 – установка однократного режима

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_SetOneshotMode (fd, 1, 1);
if (ret == -1)
printf("error: %d (%s)\n", errno, strerror(errno));
else
printf("CAN%u oneshot mode set\n", 1);
```

5.7 NM_TTCAN_SetSpeed

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.

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
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;

```
fd = open("/dev/ttcan_dev_0", O_RDWR);
```

```
ret = NM_TTCAN_SetSpeed (fd, 1, WORK_SPEED_125);
```

```
if (ret == -1)
```

```
printf("error: %d (%s)\n", errno, strerror(errno));
```

```
else
```

```
printf("CAN%u speed was set to %u kHz\n", 1, WORK_SPEED_125);
```

5.8 NM_TTCAN_SetSpeedByParams

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.

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
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;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_SetSpeedByParams(fd, 1, 1, 1, 1, 1, 3, 4, 2, 0);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u speed was set\n", 1);
```

5.9 NM_TTCAN_GetErrors

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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint8_t* pEFLG	Указатель на флаги ошибок шины
uint8_t* pTEC	Указатель на счётчик ошибок передачи
uint8_t* pREC	Указатель на счётчик ошибок приёма

Пример вызова:

```
int fd, ret;
uint8_t nEFLG, nTEC, nREC;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_GetErrors (fd, 1, &nEFLG, &nTEC, &nREC);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u EFLG=%02XH TEC=%u REC=%u \n", 1, nEFLG, nTEC, nREC);
```

5.10 NM_TTCAN_SetMasks

Назначение:

Установка масок и фильтров на прием сообщений.

Действие:

В зависимости от выбранного канала (**channel**) и номера фильтра (**filter**) функция записывает значение **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 (**eid**) 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 **mode** должна быть равно **RXB_MODE_OFF**. В этом случае параметры **filter**, **eid**, **sidm**, **eidm_or_2bytes**, **sidf**, **eidf_or_2bytes** игнорируются и модуль будет принимать из линии все сообщения.

Если **mode** равно **RXB_MODE_ALL** и **eid** равно нулю, то **eidm_or_2bytes** и **eidf_or_2bytes** должны содержать маску и фильтр для двух первых байт данных.

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
RXB_MODE mode	Режим работы буфера (допустимые значения – RXB_MODE_ALL , RXB_MODE_SID , RXB_MODE_EID , RXB_MODE_OFF)
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
int eid	0 – фильтр применяется для сообщений только с SID; Не 0 – фильтр применяется для сообщений только с EID
uint32_t sidm	идентификатор маски SID (допустимые значения – 0...7FFFH)
uint32_t eidm_or_2bytes	В зависимости от mode и eid – идентификатор маски EID (допустимые значения – 0...3FFFFH) или маска первых 2х байт данных (биты 15-8 – маска 0го байта данных; биты 7-0 – маска 1го байта данных)
uint32_t sidf	идентификатор фильтра EID (допустимые значения – 0...3FFFFH)
uint32_t eidf_or_2bytes	В зависимости от mode и eid – идентификатор фильтра EID (допустимые значения – 0...3FFFFH) или первые 2 байта данных (биты 15-8 – 0й байт данных; биты 7-0 – 1й байт данных)

Пример вызова:

```
int fd, ret;
uint32_t sidm, sidf, eidm, eidf;
fd = open("/dev/ttcan_dev_0", O_RDWR);
retval = NM_TTCAN_SetMasks(fd, 1, RXB_MODE_EID, 1, 1, sidm, eidm, sidf, eidf);
```

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

5.11 NM_TTCAN_SetTimerThreshold

Назначение:

Запуск таймера с указанным периодом.

Действие:

Из параметров **ntu_trsh** и **ntu_div** вычитается единица и полученные значения записываются в соответствующие поля регистра **CANn*_TIMER_TRSH****.

Биты RST, ENABLE и NTU_MODE регистра **CANn*_TIMER_CTRL***** устанавливаются в единицу, а битовая маска EPOCH_MASK устанавливается в соответствии с параметром **epoch_bits**. 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:

-

Аргументы функции:

Аргумент	Описание
<code>int fd</code>	Дескриптор устройства, полученный от функции <code>open()</code>
<code>uint32_t channel</code>	Номер канала (допустимые значения – 1...2)
<code>uint8_t epoch_bits</code>	Число бит в счётчике циклов (допустимые значения – 0...8)
<code>uint32_t ntu_trsh</code>	Новое значение для поля NTU_TRSH регистра CANn*_TIMER_TRSH (допустимые значения – 1...65536)
<code>uint32_t div_trsh</code>	Новое значение для поля DIV_TRSH регистра CANn*_TIMER_TRSH (допустимые значения – 1...16384)

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_SetTimerThreshold(fd, 1, 2, 4 * 16 * 8, 1000 / 50);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u_TIMER_TRSH has been set\n", 1);
```

5.12 NM_TTCAN_SetTimerCEED

Назначение:

Установка режима и значений корректировки начального значения таймера.

Действие:

Значения параметров **ntu_ceed** и **div_ceed** записываются в соответствующие поля регистра **CANn*_TIMER_CEED****.

Если параметр **epoch** отличен от нуля, то значение поля **epoch_ceed** записывается в регистр **CANn*_EPOCH_CEED*****, 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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции <code>open()</code>
uint32_t channel	Номер канала (допустимые значения – 1...2)
int epoch	Флаг использования счётчика циклов
uint32_t epoch_ceed	Новое значение для регистра CANn*_EPOCH_CEED*** (допустимые значения – 0...FFH)
uint32_t ntu_ceed	Новое значение для поля NTU_CEED регистра CANn*_TIMER_CEED (допустимые значения – 0...65535)
uint32_t div_ceed	Новое значение для поля DIV_CEED регистра CANn*_TIMER_CEED (допустимые значения – 0...16383)

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_SetTimerCEED(fd, 1, 1, 1, 1 * 16 * 8, 0);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u_TIMER_CEED&CAN%u_EPOCH_CEED has been set\n", 1);
```

5.13 NM_TTCAN_SetTimerFree

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 **epoch_bits**. Значения остальных битов данного регистра не изменяются.

* n – номер канала.

** См. раздел 5.2.4 документа “Руководство по программированию модуля “xPCIe-TTCAN””.

Примечание:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции <code>open()</code>
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint8_t epoch_bits	Число бит в счётчике циклов (допустимые значения – 0...8)

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_SetTimerFree(fd, 1, 2);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u TIMER set FREE \n", 1);
```

5.14 NM_TTCAN_SetTimerResetOnRXB

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 параметра **rxb0**, а бит **RST_ON_RXB1** устанавливается в соответствии со значением параметра **rxb1**. Значения остальных битов данного регистра не изменяются.

* n – номер канала.

** См. раздел 5.2.4 документа “Руководство по программированию модуля “xPCIE-TTCAN”.

Примечание:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции <code>open()</code>
uint32_t channel	Номер канала (допустимые значения – 1...2)
int rxb0	0 – снять режим RST_ON_RXB0 ; не 0 – установить режим RST_ON_RXB0
int rxb1	0 – снять режим RST_ON_RXB1 ; не 0 – установить режим RST_ON_RXB1

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_SetTimerResetOnRXB(fd, 1, 1, 0);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u RST_ON_RXB0 has been set\n", 1);
```

5.15 NM_TTCAN_StopTimer

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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)

Пример вызова:

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

5.16 NM_TTCAN_GetTimer

Назначение:

Чтение текущего значения таймера.

Действие:

Поля CUR_NTU и CUR_DIV регистр **CANn*_TIMER**** читаются в параметры **pcur_ntu** и **pcur_div**, а регистр **CANn*_TIMER_EPOCH***** читается в параметр **pepoch**.

* 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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint32_t* pepoch	Указатель на значение регистра CANn*_TIMER_EPOCH***
uint32_t* pcur_ntu	Указатель на значение поля CUR_NTU регистра CANn*_TIMER**
uint32_t* pcur_div	Указатель на значение поля CUR_DIV регистра CANn*_TIMER**

Пример вызова:

```
int fd, ret;
uint32_t cur_epoch, cur_ntu, cur_div
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_GetTimer(fd, 1, &cur_epoch, &cur_ntu, &cur_div);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u TIMER EPOCH=%u NTU=%u DIV=%u\n", 1, cur_epoch, cur_ntu, cur_div);
```

5.17 NM_TTCAN_StartTimerInt

Назначение:

Запуск триггера прерывания по значению таймера.

Действие:

Значения параметров `ntu_trig` и `div_trig` записываются в поля `NTU_TRIG` и `DIV_TRIG` регистра `CANn*_INT_TRIG**`.

Если значение `epoch` отлично от нуля, то значение поля `epoch_val` записывается в регистр `CANn*_INT_TRIG_EPOCH***`, 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:

-

Аргументы функции:

Аргумент	Описание
<code>int fd</code>	Дескриптор устройства, полученный от функции <code>open()</code>
<code>uint32_t channel</code>	Номер канала (допустимые значения – 1...2)
<code>int epoch</code>	Флаг учёта счётчика циклов
<code>uint32_t epoch_val</code>	Новое значение для регистра <code>CANn*_INT_TRIG_EPOCH****</code> (допустимые значения – 0...FFH)
<code>uint32_t ntu_trig</code>	Новое значение для поля <code>NTU_TRIG</code> регистра <code>CANn*_INT_TRIG**</code> (допустимые значения – 0...65535)
<code>uint32_t div_trig</code>	Новое значение для поля <code>DIV_TRIG</code> регистра <code>CANn*_INT_TRIG**</code> (допустимые значения – 0...16383)

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_StartTimerInt(fd, 1, 1, 1, 1 * 16 * 8, 0);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u_TIMER INT started\n", 1);
```

5.18 NM_TTCAN_StopTimerInt

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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции <code>open()</code>
uint32_t channel	Номер канала (допустимые значения – 1...2)

Пример вызова:

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

5.19 NM_TTCAN_WaitTimerInt

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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint32_t timeout	Таймаут ожидания прерывания, в миллисекундах

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_WaitTimerInt(fd, 1, 10);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u TIMER triggered\n", 1);
```

5.20 NM_TTCAN_SetTimeouts

Назначение:

Установка значений абсолютного и интервального таймера.

Действие:

Поле **abs_timeout** пишется в регистр **CANn*_TIMEOUT_ABSOLUTE****, а поле **itv_timeout** 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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint32_t abs_timeout	Значение абсолютного таймера, в микросекундах (допустимые значения – 0...3FFFFFFH)
uint32_t itv_timeout	Значение интервального таймера, в микросекундах (допустимые значения – 0...3FFFFFFH)

Пример вызова:

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

5.21 NM_TTCAN_Reset

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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()

Пример вызова:

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

5.22 NM_TTCAN_ResetCAN

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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)

Пример вызова:

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

5.23 NM_TTCAN_SetTxPause

Назначение:

Управление флагом TX_Pause.

Действие:

Управление битом TX_Pause регистра REG_CANx_FIFO_CONSTAT.

Примечание:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint8_t fifo	0 — fifo, 1- hpfifo
uint8_t value	Бит TX_Pause

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_SetTxPause(fd, 1, 0, 1);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u SetTxPause OK \n", 1);
```

6 Функции для чтения принятых данных

6.1 NM_TTCAN_ReadChannelRawDMA

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 параметре **pnblocks** запрашивается меньше блоков, чем накопилось в буфере DMA, то функция копирует эти блоки данных в параметр **pdata** 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 параметр **pdata** и немедленно возвратит результат. А при отличном от нуля времени ожидания процесс перейдет в состояние ожидания и будет возобновлен 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 параметр **pdata**. В любом случае после возврата из функции в параметре **pnblocks** будет указано количество скопированных блоков данных в параметр **pdata**.

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 “nmttc.h” и в разделе 6.8 документа “Руководство по программированию модуля “xPCIe-TTCAN”.

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
DMA_SLOT_TTCAN* pdata	Указатель на область, в которую будут записаны накопившиеся блоки данных
uint32_t* pnblocks	Кол-во запрашиваемых/полученных блоков (допустимые значения при вызове – 1...400). После вызова значение может быть 0...<исходное значение при вызове>
uint32_t timeout	Максимальное время ожидания требуемого количества блоков данных, в миллисекундах

Пример вызова:

```
int fd, ret;
DMA_SLOT_TTCAN data[64];
uint32_t nblocks = sizeof(data) / sizeof(*data);
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_ReadChannelRawDMA(fd, 1, &data, &nblocks, 1);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    if (nblocks > 0)
        printf("Received %u blocks from DMA\n", nblocks);
    else
        printf("No new data in DMA\n");
}
```

6.2 NM_TTCAN_ReadDMA

Назначение:

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

Действие:

Функция проверяет в буфере DMA количество новых блоков данных, копирует эти блоки данных в параметр **pdata** 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 “mPCIe-CAN”.

Формат структуры DMA_READ_BLOCK см. в файле “nmscan.h” и в разделе 6.8 документа “Руководство по программированию модуля “mPCIe-CAN”.

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
DMA_READ_BLOCK * pdata	Указатель на область, в которую будут записаны накопившиеся блоки данных

Пример вызова:

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

6.3 NM_TTCAN_DecodeBuf

Назначение:

Распаковка данных, полученных из DMA.

Действие:

Функция декодирует данные из параметра **psrcdata** и устанавливает все остальные параметры соответствующими значениями.

Примечание:

-

Аргументы функции:

Аргумент	Описание
DMA_SLOT_TTCAN* psrcdata	Указатель на данные, полученные из DMA
uint32_t* pbuf	Указатель на номер буфера RXB, в который был записан пакет
uint32_t* pch	Указатель на номер канала, в который пришёл пакет
uint32_t* ptmr_epoch	Указатель на EPOCH таймера на момент старта получения пакета
uint32_t* ptmr_ntu	Указатель на NTU таймера на момент старта получения пакета
uint32_t* ptmr_div	Указатель на DIV таймера на момент старта получения пакета
uint32_t* psid	Указатель на SID пакета
uint32_t* peid	Указатель на EID пакета (в случае если там будет FFFFFFFFH, то пакет был передан без EID)
uint8_t ponlydata[15]	Указатель на данные пакета
uint32_t* pdatasize	Указатель на размер данных полученного пакета

Пример вызова:

```
DMA_SLOT_TTCAN slot;
uint32_t nbuf, channel, tmr_epoch, tmr_ntu, tmr_div, sid, eid, datasize;
uint8_t data[15];
NM_TTCAN_DecodeBuf(&slot, &nbuf, &channel, &tmr_epoch, &tmr_ntu, &tmr_div, &sid, &eid,
data, &datasize);
```

7 Функции для передачи данных

В данной библиотеке предусмотрено пять методов передачи данных:

четыре для режима **Native**:

- синхронный метод передачи (вызов [NM_TTCAN_SendData](#));
 - асинхронный метод передачи (вызов [NM_TTCAN_SendDataRightNow](#));
 - асинхронный метод передачи по однократному срабатыванию триггера (вызов [NM_TTCAN_SendDataByTrigger](#));
 - асинхронный метод передачи по повторяющимся срабатываниям триггера (вызов [NM_TTCAN_SendDataByTriggerAuto](#));
- и передача в режиме **FIFO** с помощью вызовов: [NM_TTCAN_WriteDataToFifo](#), [NM_TTCAN_WriteDataToHpFifo](#), [NM_TTCAN_WriteDataToTgFifo](#).

До вызова функции асинхронной отправки данных необходимо записать данные в желаемый буфер с помощью функции [NM_TTCAN_WriteDataToSendBuf](#). Номера буферов в обеих функциях должны совпадать.

После вызова асинхронной отправки данных [NM_TTCAN_SendDataRightNow](#) необходимо дождаться отправки сообщения либо путём периодического опроса вызовом [NM_TTCAN_CheckTransmit](#), либо путём синхронного вызова [NM_TTCAN_WaitTransmit](#).

После вызова асинхронной отправки данных [NM_TTCAN_SendDataByTrigger](#) необходимо дождаться отправки сообщения либо путём периодического опроса сперва о срабатывании триггера вызовом [NM_TTCAN_CheckTrigger](#), а потом об окончании передачи вызовом [NM_TTCAN_CheckTransmit](#), либо путём синхронного вызова [NM_TTCAN_WaitTransmit](#).

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*.

Для начала работы в режиме **FIFO** необходимо установить режим передачи вызовом [NM_TTCAN_SetSendMode](#). Предварительно конфигурация CAN контроллера должна быть уже установлена. Доступ к регистрам CAN контроллера будет закрыт при включении режима **FIFO**.

Проверка количества сообщений в FIFO производится через вызовы [NM_TTCAN_GetCountMsgInFifo](#) и [NM_TTCAN_GetCountMsgInHpFifo](#). Записать в FIFO можно (63-count) сообщений.

Сообщения, передаваемые через обычное FIFO, могут сопровождаться временем отправки. В этом случае сначала записывается триггер вызовом [NM_TTCAN_WriteDataToTgFifo](#), затем само сообщение вызовом [NM_TTCAN_WriteDataToFifo](#). Пример: *nmttcantesttxtgfifo_m*.

Передача сообщений высокоприоритетного FIFO (вызов [NM_TTCAN_WriteDataToHpFifo](#)) происходит в обход обычного FIFO. Пока все высокоприоритетные сообщения не будут отправлены, передача из обычного FIFO приостанавливается. Пример: *nmttcantesttxfhpf_m*.

Для минимизации задержек при работе с шиной CAN может применяться следующая техника приостановки передачи: включается флаг TX_PAUSE (вызов [NM_TTCAN_SetTxPause](#) с value = 1), сообщения записываются в FIFO, в нужный момент передача разрешается (вызов [NM_TTCAN_SetTxPause](#) с value = 0). Пример: *nmttcantesttxfifops_m*.

Внимание! Пустое FIFO (count=0) не означает, что передача всех сообщений завершена. Для подтверждения окончания передачи проверьте регистр CANx_FIFO_CONSTAT.RECENT_ID или CANx_FIFO_HP_CONSTAT.RECENT_ID – он должен быть равен msgID последнего вызова [NM_TTCAN_WriteDataToFifo](#) или [NM_TTCAN_WriteDataToHpFifo](#). Подробнее см. 5.5.1 «Руководство по программированию xPCIe-TTCAN».

7.1 NM_TTCAN_SetSendMode

Назначение:

Устанавливает режим передачи канала.

Действие:

Включает режим FIFO. После этого должны использоваться ф-ции п.7.2 — 7.6. Подробнее см. примеры реализации тестов.

Примечание:

- Доступ к регистрам CAN контроллера возможен только в режиме Native.

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint8_t mode	режим (Native - 0; FIFO - 1)

Пример вызова:

```
int fd, ret;
fd = open("/dev/can_dev_0", O_RDWR);
ret = NM_TTCAN_SetSendMode (fd, 1, 0);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u data buffer %u was filled with data\n", 1, 0);
```

7.2 NM_TTCAN_GetCountMsgInFifo

Назначение:

Получает текущее кол-во сообщений в FIFO.

Действие:

-

Примечание:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
int* count	текущее кол-во сообщений в FIFO

Пример вызова:

```
int fd, ret;
int count;
fd = open("/dev/can_dev_0", O_RDWR);
ret = NM_TTCAN_GetCountMsgInFifo (fd, 1, & count);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u data buffer %u was filled with data\n", 1, 0);
```

7.3 NM_TTCAN_GetCountMsgInHpFifo

Назначение:

Получает текущее кол-во сообщений в HPFIFO.

Действие:

-

Примечание:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
int* count	текущее кол-во сообщений в HPFIFO

Пример вызова:

```
int fd, ret;
int count;
fd = open("/dev/can_dev_0", O_RDWR);
ret = NM_TTCAN_GetCountMsgInHpFifo (fd, 1, & count);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u data buffer %u was filled with data\n", 1, 0);
```

7.4 NM_TTCAN_WriteDataToFifo

Назначение:

Записать сообщение в FIFO на отправку.

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 в драйвере общим семафором (индивидуальным для каждого канала и буфера) для возможности их корректного использования из параллельных процессов.

Примечание:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint8_t isExtId	признак расширенного сообщения
uint32_t sid	SID
uint32_t eid	EID
uint8_t* pdata	Массив данных
uint32_t datasize	Размер данных массива
uint8_t msgId	уникальный идентификатор сообщения
uint8_t rtr	Rtr бит

Пример вызова:

```
int fd, ret;
int count;
uint8_t pdata[8];
fd = open("/dev/can_dev_0", O_RDWR);
ret = NM_TTCAN_WriteDataToFifo (fd, 1, 1, 0x1, 0x1, pdata, 8, 0, 0);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u data buffer %u was filled with data\n", 1, 0);
```

7.5 NM_TTCAN_WriteDataToHpFifo

Назначение:

Записать сообщение в HPFIFO на отправку.

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 в драйвере общим семафором (индивидуальным для каждого канала и буфера) для возможности их корректного использования из параллельных процессов. .

Примечание:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint8_t isExtId	признак расширенного сообщения
uint32_t sid	SID
uint32_t eid	EID
uint8_t* pdata	Массив данных
uint32_t datasize	Размер данных массива
uint8_t msgId	уникальный идентификатор сообщения
uint8_t rtr	Rtr бит

Пример вызова:

```
int fd, ret;
int count;
uint8_t pdata[8];
fd = open("/dev/can_dev_0", O_RDWR);
ret = NM_TTCAN_WriteDataToHpFifo (fd, 1, 1, 0x1, 0x1, pdata, 8, 0, 0);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u data buffer %u was filled with data\n", 1, 0);
```

7.6 NM_TTCAN_WriteDataToTgFifo

Назначение:

Записать данные о времени отправки в TGFIFO на отправку.

Действие:

Записывает данные о времени отправки в буфер TGFIFO, активирует триггер отправки.

Примечание:

Надо использовать перед ф-циями NM_TTCAN_WriteDataToFifo, NM_TTCAN_WriteDataToHpFifo

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
uint8_t isUseEpoch	флаг учёта использования эпохи (0 – не использовать)
uint8_t epoch	данные о времени отправки epoch
uint8_t ntu	данные о времени отправки ntu
uint8_t div	данные о времени отправки div

Пример вызова:

```
int fd, ret;
int count;
uint8_t pdata[8];
fd = open("/dev/can_dev_0", O_RDWR);
ret = NM_TTCAN_WriteDataToTgFifo (fd, 1, 1, 1, 0);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u data buffer %u was filled with data\n", 1, 0);
```

7.7 NM_TTCAN_WriteDataToSendBuf

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 в драйвере 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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
CAN_BUF nbuf	Номер буфера, в который будут записываться данные (допустимые значения – 0...2)
uint32_t prio	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* pdata	Указатель на данные
uint32_t datasize	Размер передаваемых данных (допустимые значения – 0...8)

Пример вызова:

```
int fd, ret;
uint8_t data[] = { 0xFF };
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_WriteDataToSendBuf(fd, 1, 0, 0, 0x7FF, 0x3FFFF, data, sizeof(data));
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u data buffer %u was filled with data\n", 1, 0);
```

7.8 NM_TTCAN_SendData

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 перейдет 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 в драйвере 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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
CAN_BUF nbuf	Номер буфера, в который будут записываться данные (допустимые значения – 0...2)
uint32_t prio	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...3FFFFFFH и специальное значение FFFFFFFFH, которое означает, что расширенный идентификатор не используется)
uint8_t* pdata	Указатель на данные
uint32_t datasize	Размер передаваемых данных (допустимые значения – 0...8)
uint32_t timeout	Максимальное время ожидания отправки сообщения, в миллисекундах
TXBCTRL* ptxb_ctrl	Указатель на значение регистра TXBn*CTRL в случае истечения времени ожидания отправки сообщения

Пример вызова:

```
int fd, ret;
uint8_t data[] = { 0xFF };
uint8_t txb_ctrl;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_SendData (fd, 1, 0, 0, 0x7FF, 0x3FFFF, data, sizeof(data), 100, &txb_ctrl);
if (ret == -1)
{
```

```
printf("error: %d (%s); TXB%uCTRL=%02XH\n", errno, strerror(errno), 0, txb_ctrl);  
}  
else  
printf("CAN%u data buffer %u was filled with data\n", 1, 0);
```

7.9 NM_TTCAN_SendDataRightNow

Назначение:

Запуск асинхронной передачи сообщения.

Действие:

Бит TXREQ регистра **TXBn*CTRL**** устанавливается в единицу.

Все функции работы с буферами контроллера CAN защищены в драйвере 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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
CAN_BUF nbuf	Номер буфера, в который будут записываться данные (допустимые значения – 0...2)

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_SendDataRightNow(fd, 1, 0);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u data buffer %u was requested to send\n", 1, 0);
```

7.10 NM_TTCAN_CheckTransmit

Назначение:

Проверка правильности отправки сообщения.

Действие:

Данная функция читает регистр **TXBn*CTRL**** в параметр **ptxb_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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
CAN_BUF nbuf	Номер буфера, который будет проверен (допустимые значения – 0...2)
TXBCTRL* ptxb_ctrl	Указатель на значение регистра TXBn*CTRL

Пример вызова:

```
int fd, ret;
uint8_t txb_ctrl
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_CheckTransmit(fd, 1, 0, &txb_ctrl);
if (ret == -1)
{
printf("error: %d (%s); TXB%uCTRL=%02XH\n", errno, strerror(errno), 0, txb_ctrl);
}
else
printf("CAN%u data buffer %u was sended\n", 1, 0);
```

7.11 NM_TTCAN_WaitTransmit

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 параметр **ptxb_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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
CAN_BUF nbuf	Номер буфера, чья отправка будет ожидаться (допустимые значения – 0...2)
uint32_t timeout	Максимальное время ожидания отправки сообщения, в миллисекундах
TXBCTRL* ptxb_ctrl	Указатель на значение регистра TXBn*CTRL

Пример вызова:

```
int fd, ret;
uint8_t txb_ctrl;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_WaitTansmit(fd, 1, 0, 100);
if (ret == -1)
{
printf("error: %d (%s); TXB%uCTRL=%02XH\n", errno, strerror(errno), 0, txb_ctrl);
}
else
printf("CAN%u data buffer %u was sended\n", 1, 0);
```

7.12 NM_TTCAN_EndTransmit

Назначение:

Снятие запроса отправки сообщения.

Действие:

Данная функция записывает ноль в бит TXREQ регистра **TXBn*CTRL****.

* n – номер буфера.

** См. Раздел 6.7.1 документа “Руководство по программированию модуля “xPCIE-TTCAN”.

Примечание:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
CAN_BUF nbuf	Номер буфера, с которого будет снят запрос на отправку (допустимые значения – 0...2)

Пример вызова:

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

7.13 NM_TTCAN_SendDataByTrigger

Назначение:

Запуск однократной передачи данных из буфера передачи по срабатыванию триггера.

Действие:

Значения триггера, находящиеся в параметрах **ntu_trig** и **div_trig**, записываются в соответствующие поля регистра **CANn*_TXm**_TRIG *****.

Если параметр **epoch** не равен нулю, то значение из параметра **epoch_val** записывается в регистр **CANn*_TXm**_TRIG_EPOCH ******, 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 (параметр **nbuf**) в этом вызове должен совпадать с номером буфера, в который были записаны данные с помощью функции [NM_TTCAN_WriteDataToSendBuf](#).

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции <code>open()</code>
uint32_t channel	Номер канала (допустимые значения – 1...2)
CAN_BUF nbuf	Номер буфера, который будет установлен на отправку (допустимые значения – 0...2)
int epoch	Флаг использования счётчика циклов
uint32_t epoch_val	Значение регистра CANn*_TXm**_TRIG_EPOCH ****
uint32_t ntu_trig	Значение поля NTU_TRIG триггера в регистре CANn*_TXm**_TRIG ***
uint32_t div_trig	Значение поля DIV_TRIG в регистре CANn*_TXm**_TRIG ***

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_SendDataByTrigger(fd, 1, 0, 1, 1, 1 * 16 * 8, 0);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u data buffer %u was requested to send by trigger\n", 1, 0);
```

7.14 NM_TTCAN_SendDataByTriggerAuto

Назначение:

Запуск многократной передачи данных из буфера передачи по срабатыванию триггера.

Действие:

Значения триггера, находящиеся в параметрах **ntu_trig** и **div_trig**, записываются в соответствующие поля регистра **CANn*_TXm**_TRIG *****.

Если параметр **epoch** не равен нулю, то значение из параметра **epoch_val** записывается в регистр **CANn*_TXm**_TRIG_EPOCH ******, 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 [NM_TTCAN_WriteDataToSendBuf](#).

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции <code>open()</code>
uint32_t channel	Номер канала (допустимые значения – 1...2)
CAN_BUF nbuf	Номер буфера, который будет установлен на отправку (допустимые значения – 0...2)
int epoch	Флаг использования счётчика циклов
uint32_t epoch_val	Значение регистра CANn*_TXm**_TRIG_EPOCH ****
uint32_t ntu_trig	Значение поля NTU_TRIG триггера в регистре CANn*_TXm**_TRIG ***
uint32_t div_trig	Значение поля DIV_TRIG в регистре CANn*_TXm**_TRIG ***

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_SendDataByTriggerAuto(fd, 1, 0, 1, 1, 1 * 16 * 8, 0);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u data buffer %u was requested to send by trigger\n", 1, 0);
```

7.15 NM_TTCAN_CheckTrigger

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:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
CAN_BUF nbuf	Номер триггера, который будет проверен (допустимые значения – 0...2)

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_CheckTrigger(fd, 1, 0);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u trigger %u is free\n", 1, 0);
```

7.16 NM_TTCAN_ABAT

Назначение:

Остановка всех активных передач.

Действие:

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

* См. Раздел 6.3.1 документа “Руководство по программированию модуля “xPCIE-TTCAN”.

Примечание:

-

Аргументы функции:

Аргумент	Описание
int fd	Дескриптор устройства, полученный от функции open()
uint32_t channel	Номер канала (допустимые значения – 1...2)
int abat	0 – снятие запроса на остановку всех активных передач. Не 0 - запрос на остановку всех активных передач

Пример вызова:

```
int fd, ret;
fd = open("/dev/ttcan_dev_0", O_RDWR);
ret = NM_TTCAN_ABAT(fd, 1, 1);
if (ret == -1)
    printf("error: %d (%s)\n", errno, strerror(errno));
else
    printf("CAN%u ABAT was set\n", 1);
```

8 Обновление библиотеки

Версия библиотеки	Дата	Изменение
2.0	31.05.2018	- Библиотека создана
3.0	29.07.2020	- Добавлен режим FIFO
3.1	14.10.2021	- Добавлена обработка бита RTR в функциях передачи.

9 Обновление руководства.

Версия документа	Дата	Изменение
2.0	31.05.2018	- Документ создан
3.0	29.07.2020	- Добавлен режим FIFO
3.1	21.10.2021	- Добавлена обработка бита RTR в функциях передачи.
3.2	08.12.2021	- Исправлены незначительные ошибки.