NIM351

# Программное обеспечение поддержки прикладного программирования

Руководство программиста

**Версия 1.20**

## СОДЕРЖАНИЕ

Настоящее Руководство содержит указания по установке и использованию драйверов и библиотек прикладного программирования модуля NIM351 производства ЗАО "НПФ "ДОЛОМАНТ".

**Торговые марки**

Логотип «Fastwel» является торговой маркой, принадлежащей ЗАО «НПФ «ДОЛОМАНТ», Москва, Российская Федерация.

Кроме того, настоящий документ может содержать наименования, фирменные логотипы и торговые марки, являющиеся зарегистрированными торговыми марками, а следовательно, права собственности на них принадлежат их законным владельцам.

**Права собственности**

Настоящий документ содержит информацию, которая является собственностью ЗАО «НПФ «ДОЛОМАНТ». Он не может быть скопирован или передан с использованием известных средств, а также не может храниться в системах хранения и поиска информации без предварительного письменного согласия ЗАО «НПФ «ДОЛОМАНТ» или одного из ее уполномоченных агентов. Информация, содержащаяся в настоящем документе, насколько нам известно, не содержит ошибок, однако, ЗАО «НПФ «ДОЛОМАНТ» не может принять на себя ответственность за какие-либо неточности и их последствия, а также ответственность, возникающую в результате использования или применения любой схемы, продукта или примера, приведенного в настоящем документе. ЗАО «НПФ «ДОЛОМАНТ» оставляет за собой право изменять и усовершенствовать как настоящий документ, так и представленный в нем продукт по своему усмотрению без дополнительно извещения.

**Контактная информация**

Производитель ЗАО «НПФ «ДОЛОМАНТ»:

Почтовый адрес: Российская Федерация, 117437, Москва, Профсоюзная ул., 108

Телефон: (495) 232-2033

Факс: (495) 232-1654

Электронная почта: info@fastwel.ru

Для получения информации о других продуктах, выпускаемых под торговой маркой «Fastwel», посетите наш Интернет-сайт по адресу

http://www.fastwel.ru/

Эксклюзивный дистрибьютор компания «Прософт»

Электронная почта: info@prosoft.ru

Web: http://www.prosoft.ru/

Телефон: (495) 234-0636

Факс: (495) 234-0640

**Авторское право**

# 1. ВВЕДЕНИЕ

## 1.1. Сведения о документе

Настоящий документ содержит указания по установке и использованию программного обеспечения пакета поддержки прикладного программирования для модуля NIM351 производства ЗАО "НПФ "ДОЛОМАНТ".

## 1.2. Информация о поставляемом программном обеспечении

Программное обеспечение (ПО) пакета поддержки реализует программный интерфейс доступа к функциям контроллера CAN-интерфейсов модуля NIM351 из прикладных программ и включает в себя следующие основные компоненты:

1. Драйвер контроллера CAN-интерфейсов.
2. Библиотеку прикладного программирования контроллера CAN-интерфейсов.
3. Пример программирования.

Кроме того, в комплект поставки включены командные файлы для построения из командной строки бинарных модулей тестовых примеров, использующие компиляторы и редакторы связей сред разработки из числа приведенных, установленных на инструментальном персональном компьютере (ПК) пользователя:

1. Microsoft Visual C++ 2008, или
2. Microsoft Visual C++ 2005, или
3. Microsoft Visual C++ .NET 2003, или
4. Microsoft Visual C++ .NET 2002, или
5. Microsoft Visual C++ 6.0 SP1.

Предполагается, что пользователь пакета поддержки имеет навыки программирования на языках C и/или C++ для операционных систем семейства Windows и знаком с настольными операционными системами Windows XP/Vista/7 на уровне продвинутого пользователя.

В настоящий комплект поставки ПО включены версии модулей драйверов, библиотек и исходных текстов только для операционной системы Microsoft Windows XP. Бинарные файлы драйверов и библиотек прикладного программирования находятся в подкаталоге \bin\xp каталога установки:

1. nim351.sys – драйвер контроллера CAN-интерфейсов.
2. nim351.inf – inf-файл драйвера контроллера CAN-интерфейсов.
3. fwcan.dll – библиотека, реализующая программный интерфейс доступа к функциям адаптеров сети CAN.

Файлы статических библиотек связи с динамическими библиотеками поддержки находятся в подкаталоге \lib\xp:

1. fwcan.lib – библиотека связи приложения с динамической библиотекой доступа к функциям адаптера сети CAN.

Заголовочные файлы и модули исходных текстов драйверов, библиотек и тестовых примеров находятся в подкаталоге \src:

1. \src\eal – включает заголовочные файлы и модули исходных текстов слоя абстракции окружения, используемые для доступа к некоторым функциям ОС.
2. \src\can – включает заголовочные файлы и модули исходных текстов драйвера и библиотеки поддержки адаптеров сети CAN.
3. \src\test – включает исходные тексты тестовых примеров:

   \src\test\can_test.cpp – тестовый пример программирования CAN-адаптеров.

Командные файлы построения исполняемых модулей драйвероа, библиотек прикладного программирования и тестовых примеров находятся в подкаталоге \build:

1. \build\build_test.bat – предназначен для построения всех тестовых примеров. Результирующие исполняемые файлы тестовых примеров будут расположены в подкаталоге \intdir\test каталога установки пакета поддержки.
2. \build\xp\vc_build_test.bat – основной командный файл для построения одного исполняемого файла примера.
3. \build\xp\vc_chk.bat – командный файл проверки наличия установленных инструментальных средств Microsoft Visual C++, вызываемый из vc_build_lib.bat и vc_build_test.bat.
4. \build\xp\vccc.bat – командный файл вызова компилятора.
5. \build\xp*.cfg – файлы общих параметров командной строки, используемых при вызове компилятора и редактора связей среды разработки.

Файлы конфигураций построения, используемых при построении тестовых примеров, находятся в подкаталоге \configs\xp\test и содержат специфические для каждого тестового примера параметры командной строки, используемые при вызове компилятора и редактора связей.

# 2. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОНТРОЛЛЕРА ШИНЫ CAN

## 2.1. Общие сведения

Контроллер CAN-интерфейсов модуля NIM351 имеет в своем составе два CAN-адаптера, каждый из которых рассматривается программным обеспечением независимо друг от друга, как отдельное устройство.

Функционально ПО контроллера шины CAN реализует программный интерфейс доступа к сети CAN на канальном уровне эталонной модели ISO/OSI, предоставляя механизмы регистрации и обработки событий в CAN-адаптере, способы управления CAN-кадрами и режимом работы CAN-адаптера.

Поддерживаются:

- стандартный и расширенный протоколы CAN (11 и 29-битовый идентификаторы кадров);
- скорость передачи до 1000 Мбит/сек;
- операции синхронного и асинхронного доступа к CAN-адаптеру для приема и передачи кадров;
- приемный фильтр сообщений;
- регистрация времени приема сообщений;
- регистрация ошибок в CAN-адаптере и определение его текущего статуса.

Для доступа к аппаратным функциям устройства ПО использует драйвер nim351.sys. Указания по добавлению устройства (CAN-адаптера) в систему приведены в п. 2.2.2 настоящего руководства. Интерфейс доступа к функциям CAN-адаптера из прикладных программ обеспечивается библиотекой динамической загрузки fwcan.dll и статической библиотекой fwcan.lib. Бинарные файлы драйвера (nim351.sys) и динамической библиотеки (fwcan.dll) находятся в подкаталоге \bin\xp каталога установки пакета поддержки. Бинарный файл статической библиотеки (fwcan.lib) в подкаталоге \lib\xp. Интерфейс библиотеки прикладного программирования определен в заголовочном файле \src\can\lib\can_lib.h.

## 2.2. Драйвер CAN-адаптера

### 2.2.1. Общие сведения

Драйвер CAN-адаптеров nim351.sys реализован в соответствии с моделью драйверов режима ядра Windows с поддержкой PnP. Так как контроллер CAN-интерфейсов NIM351 не поддерживает перемещение своих ресурсов (номеров прерываний, области ввода/вывода), то PnP-функциональность реализована в ограниченной форме.

Каждый из CAN-адаптеров контроллера CAN-интерфейсов NIM351рассматривается как отдельное устройство и устанавливается независимо от другого.

### 2.2.2. Установка CAN-адаптера

Для добавления CAN-адаптера в список устройств Windows XP:

1. Откройте **Панель управления** (**Start**–**Control Panel**).
2. Дважды щелкните **Add Hardware**, нажмите **Next**
3. В появившемся после сканирования окне выберите пункт *Yes, I have already connected the hardware*, как показано на рис. 1, и нажмите **Next**.


**Рис. 1. Диалог мастера установки устройств**

4. В появившемся окне прокрутите список вниз до конца и выберите пункт *Add a new hardware device.*

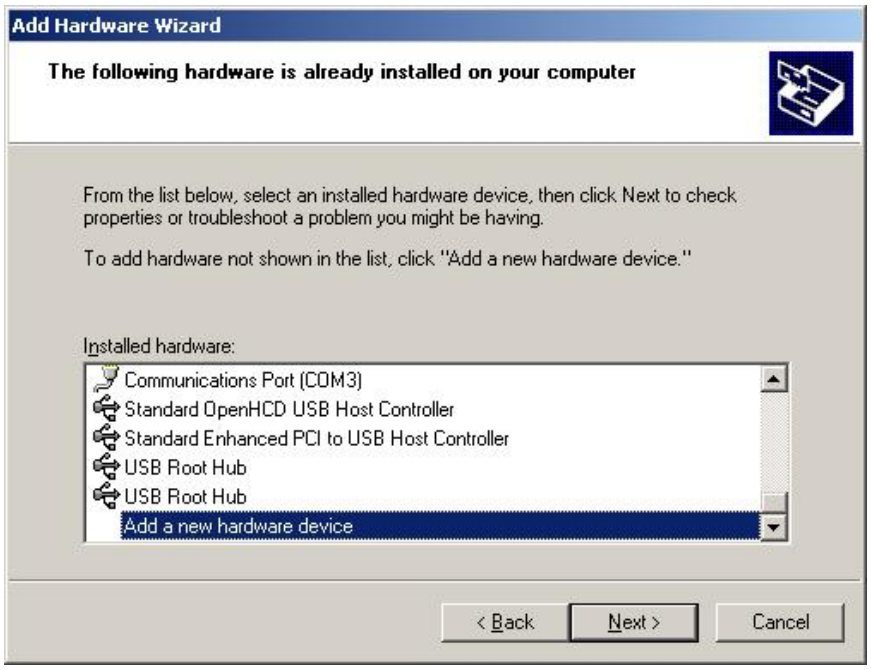


**Рис. 2. Выбор типа устройства**

5. В появившемся окне выберите установку устройства вручную, как показано на рис. 3, и нажмите **Next**.

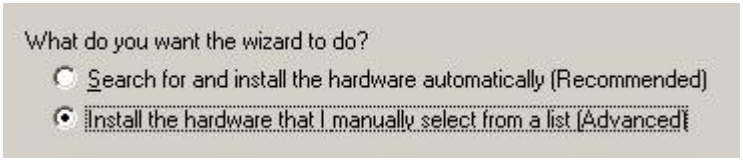


**Рис. 3. Выбор типа установки**

6. В следующем диалоге выберите пункт *Show All Devices*, как показано на рис. 4.

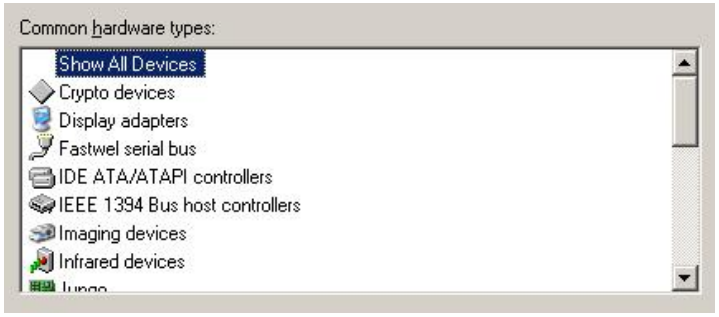


**Рис. 4. Выбор типа добавляемого устройства**

7. Нажмите **Next**, а затем, когда будет отображен список всех устройств, нажмите **Have Disk**.


**Рис. 5. Выбор драйвера для установки**

8. Укажите путь к файлу nim351.inf и выберите соответствующую строку, как показано на рис. 6.

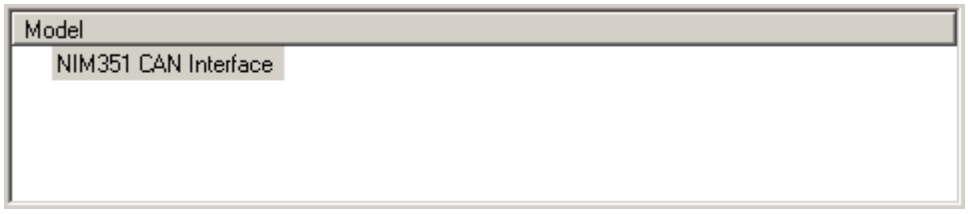


**Рис. 6. Выбор типа устанавливаемого устройства**

9. Нажмите **Next**, на все вопросы о копировании отвечайте утвердительно, в случае появления диалога об отсутствии у драйвера цифровой подписи выберите пункт *Все равно продолжить* (*Continue anyway*).
10. Если не возникло никаких сообщений о конфликтах, появится окно с информацией о том, что установка ПО завершена, показанное на рис. 7.
11. В случае возникновения каких-либо ошибок при установке драйвера, а также конфликтов с другими устройствами, обратитесь к системному администратору.


**Рис. 7. Диалоговая панель информации о завершении установки**

### 2.2.3. Конфигурирование CAN-адаптера

Конфигурирование ресурсов, используемых устройством, производится с помощью **Менеджера устройств** (**Device Manager**) ОС Windows. Для конфигурирования необходимо выбрать устройство в списке **Менеджера устройств**:

1. Щелкните правой кнопкой мыши над пиктограммой *My Computer* и выберите команду **Properties** в появившемся контекстном меню.
2. Откройте вкладку **Hardware**.
3. Нажмите кнопку **Device Manager**.

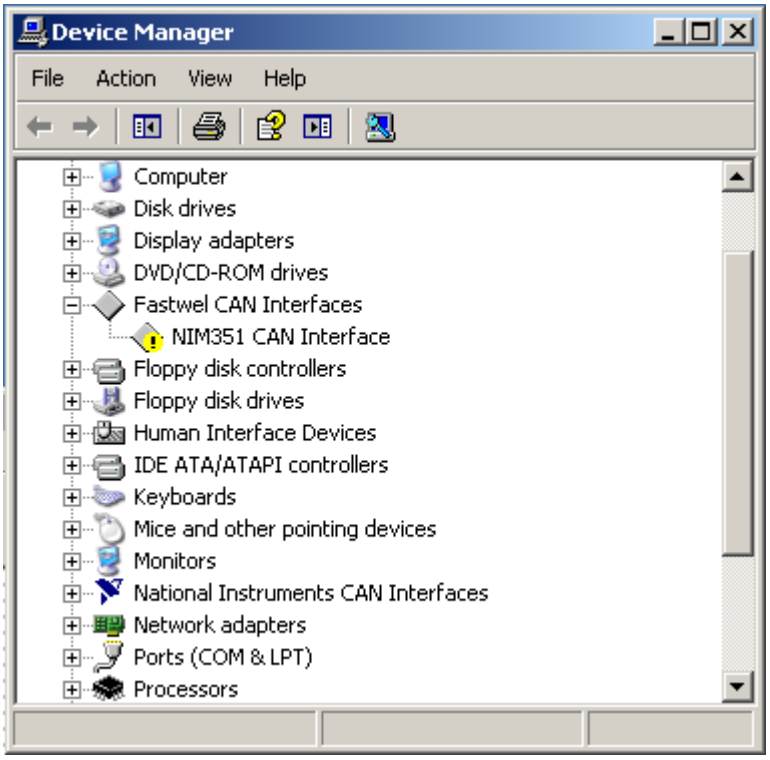


**Рис. 8. Менеджер устройств Windows**

4. Откройте окно свойств устройства, для чего раскройте элемент дерева устройств *Fastwel CAN Interfaces*, дважды щелкните на его дочернем элементе *NIM351 CAN Interface*, после чего откройте вкладку Resources, как показано на рис. 9.

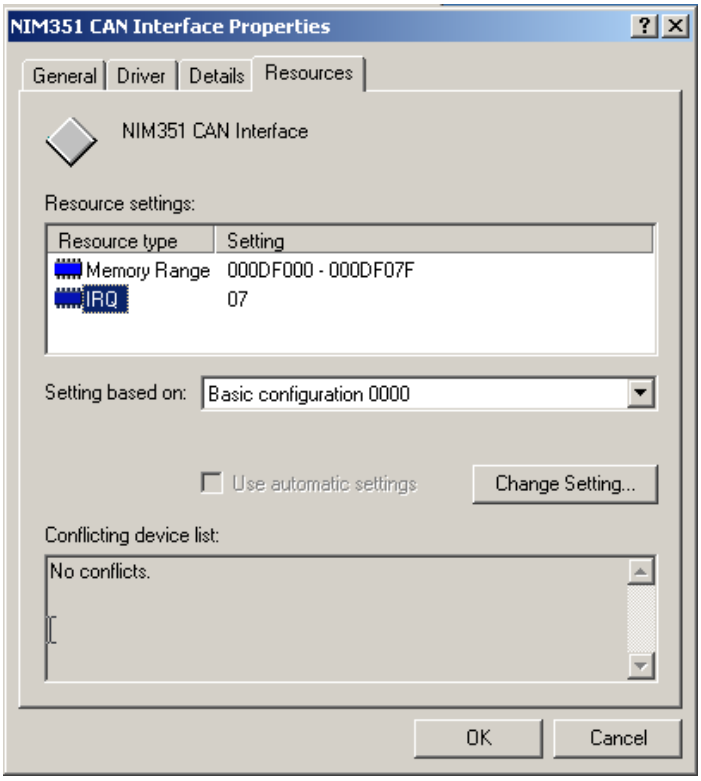


**Рис. 9. Редактирование ресурсов**

Отредактируйте значения ресурсов CAN-адаптера в соответствии с текущей заданной аппаратной конфигурацией.

Для устройства CAN1 NIM351используйте конфигурацию *Basic configuration 0000*, а для устройства CAN2 используйте конфигурацию *Basic configuration 0001*.

Для установки значений следует выбрать тип ресурса и нажать кнопку **Change Settings** в окне свойств адаптера, после чего в появившемся окне редактора ресурса задать требуемое значение, и, убедившись в отсутствии конфликтов, нажать **ОК**.

## 2.3. Библиотека прикладного программирования

### 2.3.1. Общие сведения

Интерфейс доступа к функциям CAN-адаптера из прикладных программ обеспечивается библиотекой динамической загрузки fwcan.dll и статической библиотекой fwcan.lib. Каждая программа, работающая с библиотекой должна включать заголовочный файл can_lib.h, который содержит объявления типов данных и прототипы функций, экспортируемых библиотекой. Интерфейс библиотеки представлен в табл. 1.

**Таблица 1. Функции библиотеки интерфейса прикладного программирования fwcan.dll**

| Имя функции | Описание |
|---|---|
| fw_can_init | Инициализировать внутренние структуры данных библиотеки. Данная функция должна быть вызвана первой до вызова любой другой функции библиотеки. |
| fw_can_open | Открыть CAN-адаптер и получить его системный идентификатор. |
| fw_can_close | Закрыть открытый CAN-адаптер. |
| fw_can_get_controller_config | Получить текущие настройки CAN-адаптера. |
| fw_can_set_controller_config | Задать новые значения параметров настройки CAN-адаптера. |
| fw_can_get_timeouts | Получить текущие значения таймаутов опреций чтения, записи и автоматического сброса состояния BUS_OFF. |
| fw_can_set_timeouts | Установить новые значения таймаутов опреций чтения, записи и автоматического сброса состояния BUS_OFF. |
| fw_can_start | Запустить CAN-адаптер (перевести в рабочее состояние, CAN-адаптер подключен к сети) |
| fw_can_stop | Остановить CAN-адаптер (перевести в состояние инициализации, CAN-адаптер |

| Имя функции | Описание |
|---|---|
| | отключен от сети) |
| fw_can_get_controller_state | Получить текущее состояние CAN-адаптера. |
| fw_can_get_stats | Получить статистическую информацию CAN-адаптера. |
| fw_can_clear_stats | Сбросить в 0 статистическую информацию CAN-адаптера. |
| fw_can_send | Передать CAN-кадры. Функция блокируется до тех пор, пока все кадры не будут реально переданы в сеть, либо пока не произойдет таймаут передачи. |
| fw_can_recv | Прочитать полученные CAN-кадры из приемного буфера. Если буфер пуст, функция блокируется до тех пор, пока не будет принят первый кадр или не истечет таймаут приема. |
| fw_can_post_message | Поместить CAN-кадр во внутренний буфер передачи CAN-адаптера. Функция не блокируется и не ожидает завершения передачи кадра в сеть. |
| fw_can_peek_message | Прочитать кадр из приемного буфера. Функция не блокируется при любом состоянии буфера. |
| fw_can_purge | Прервать отложенные обращения к адаптеру по записи и/или прервать отложенные обращения к адаптеру по чтению, и/или сбросить в адаптере текущий запрос на передачу, и/или очистить приемный буфер, и/или выполнить аппаратный сброс адаптера. |
| fw_can_wait | Приостановить (блокировать) выполнение вызывающего потока исполнения (нити, на контексте которой вызывается данная функция) до получения адаптером заданного статуса (получен кадр, свободный внутренний буфер передачи, произошла ошибка) или до наступления таймаута. |
| fw_can_get_clear_errors | Получить и сбросить счетчики ошибок CAN-адаптера |

Каждая из функций в качестве результата выполнения возвращает код ошибки перечислимого типа **F_CAN_RESULT**, возможные значения которого перечислены в табл. 2.

**Таблица 2. Значения перечислимого типа F_CAN_RESULT**

| Значение | Описание |
|---|---|
| **CAN_RES_OK** | успех |
| **CAN_RES_HARDWARE** | Аппаратная ошибка |
| **CAN_RES_INVALID_HANDLE** | Недействительный (неправильный) системный идентификатор (хэндл) адаптера |
| **CAN_RES_INVALID_POINTER** | Некорректный указатель |
| **CAN_RES_INVALID_PARAMETER** | Неправильный параметр (один или несколько) |
| **CAN_RES_INSUFFICIENT_RESOURCES** | Не хватило системных ресурсов |
| **CAN_RES_OPEN_DEVICE** | Не удалось открыть устройство |
| **CAN_RES_UNEXPECTED** | Внутренняя ошибка в драйвере устройства |
| **CAN_RES_FAILURE** | Ошибка обращения к драйверу или к устройству |
| **CAN_RES_RXQUEUE_EMPTY** | Буфер приема пуст |
| **CAN_RES_NOT_SUPPORTED** | Операция не поддерживается |
| **CAN_RES_TIMEOUT** | Таймаут |

Проверка успешности результата операции (**result**) может быть выполнена макросом **F_CAN_SUCCESS(result)** имеющего значение **true** в случае успеха.

### 2.3.2. Работа с библиотекой

Типичная последовательность работы с библиотекой выглядит следующим образом:

1. Инициализация библиотеки (**fw_can_init**).
2. Получение доступа к CAN-адаптеру (**fw_can_open**).
3. Конфигурирование (**fw_can_get_controller_config, fw_can_set_controller_config, fw_can_get_timeouts, fw_can_set_timeouts**).
4. Запуск CAN-адаптера/подключение к сети (**fw_can_start**).
5. Работа с сетью посредством операций ввода/вывода (**fw_can_send, fw_can_recv; fw_can_wait, fw_can_post_message, fw_can_peek_message**).

6. Обработка ошибок CAN-интерфейса (**fw_can_wait, fw_can_get_controller_state, fw_can_purge, fw_can_get_clear_errors**).
7. Останов CAN-адаптера/отключение от сети (**fw_can_stop**).
8. Закрытие/освобождение CAN-адаптера (**fw_can_close**).

### 2.3.3. Описание функций и типов данных

#### 2.3.3.1. fw_can_int

**Прототип**

```
F_CAN_RESULT fw_can_init(void)
```

**Параметры**

Нет.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция инициализирует внутренние структуры данных библиотеки. Должна вызываться только один единственный раз в самом начале пользовательского приложения до вызова любой другой функции библиотеки.

#### 2.3.3.2. fw_can_open

**Прототип**

```
F_CAN_RESULT fw_can_open(F_CAN_DEVID id, PF_CAN_HANDLE phDev)
```

**Параметры**

**F_CAN_DEVID id**

Идентификатор (номер CAN-адаптера), начиная с 1.

**PF_CAN_HANDLE phDev**

Указатель на переменную типа **F_CAN_HANDLE**, в которой, при успешном открытии адаптера, окажется хэндл устройства. Корректность хэндла можно проверить при помощи макроса **fw_can_is_handle_valid**:

```
PF_CAN_HANDLE handle;
if(CAN_RES_OK == fw_can_open(1, &handle) && fw_can_is_handle_valid(handle))
{
  //устройство с индексом 1 открыто успешно
  fw_can_close(handle);
}
```

**Возвращаемое значение**

**CAN_RES_OK –** успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция открывает CAN-адаптер с заданным идентификатором (номером) и возвращает хэндл программного доступа к нему. После успешного выполнения устройство находится в начальном состоянии: CAN-адаптер – в состоянии **CAN_STATE_INIT**, конфигурация адаптера задана значениями по умолчанию (см. пп. 2.3.3.20–2.3.3.21).

#### 2.3.3.3. fw_can_close

**Прототип**

```
F_CAN_RESULT fw_can_close(F_CAN_HANDLE hDev)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл закрываемого адаптера.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция закрывает и освобождает устройство, ранее открытое при успешном вызове функции **fw_can_open**.

#### 2.3.3.4. fw_can_get_controller_config

**Прототип**

```
F_CAN_RESULT fw_can_get_controller_config(F_CAN_HANDLE hDev,
                                          PF_CAN_SETTINGS pDcb)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**PF_CAN_SETTINGS pDcb**

Указатель на переменную типа **F_CAN_SETTINGS**, в которую, при успешном вызове, будут записаны текущие значения параметров конфигурации CAN-адаптера. Описание полей структуры и их назначение приведены в п. 2.3.3.20.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция получает текущие значения параметров конфигурации CAN-адаптера.

#### 2.3.3.5. fw_can_set_controller_config

**Прототип**

```
F_CAN_RESULT fw_can_set_controller_config(F_CAN_HANDLE hDev,
                                          PF_CAN_SETTINGS pDcb)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**PF_CAN_SETTINGS pDcb**

Указатель на переменную типа **F_CAN_SETTINGS**, содержащую требуемые значения параметров конфигурации CAN-адаптера. Описание полей структуры и их назначение приведены в п. 2.3.3.20.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция задает новые значения параметров конфигурации CAN-адаптера. Установка параметров возможна только в состоянии адаптера **CAN_STATE_INIT**.

### 2.3.3.6. fw_can_get_timeouts

**Прототип**

```
F_CAN_RESULT fw_can_get_timeouts(F_CAN_HANDLE hDev,
                                 PF_CAN_TIMEOUTS pTimeouts)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**PF_CAN_TIMEOUTS pTimeouts**

Указатель на переменную типа **F_CAN_TIMEOUTS**, в которую, при успешном вызове, будут записаны текущие значения параметров. Описание полей структуры и их назначение приведены в п. 2.3.3.21.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция получает текущие значения таймаутов, используемых в синхронных операциях чтения, записи и в операции автоматического восстановления CAN-адаптера из состояния **CAN_STATE_BUS_OFF.**

### 2.3.3.7. fw_can_set_timeouts

**Прототип**

```
F_CAN_RESULT fw_can_set_timeouts(F_CAN_HANDLE hDev,
                                 PF_CAN_TIMEOUTS pTimeouts)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**PF_CAN_TIMEOUTS pTimeouts**

Указатель на переменную типа **F_CAN_TIMEOUTS**, содержащую требуемые значения параметров. Описание полей структуры и их назначение приведены в п. 2.3.3.21.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция задает новые значения таймаутов, используемых в синхронных операциях чтения, записи и в операции автоматического восстановления CAN-адаптера из состояния **CAN_STATE_BUS_OFF.** Установка параметров возможна только в состоянии адаптера **CAN_STATE_INIT.**

#### 2.3.3.8. fw_can_get_controller_state

**Прототип**

```
F_CAN_RESULT fw_can_get_controller_state(F_CAN_HANDLE hDev,
                                        PF_CAN_STATE pState)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**PF_CAN_STATE pState**

Указатель на переменную перечислимого типа **F_CAN_STATE**, в которую, при успешном вызове, будет записано текущие значения состояния CAN-адаптера:

**CAN_STATE_INIT** – начальное состояние адаптера; допускается конфигурирование интерфейса.
**CAN_STATE_ERROR_ACTIVE** – кол-во ошибок приема/передачи не более 96 (состояние определено в спецификации CAN 2.0B).
**CAN_STATE_ERROR_WARNING** – кол-во ошибок приема/передачи от 96 до 127 (CAN 2.0B).
**CAN_STATE_ERROR_PASSIVE** – кол-во ошибок приема/передачи от 128 до 255 (CAN 2.0B).
**CAN_STATE_BUS_OFF** – кол-во ошибок передачи ≥256 (BUSOFF); адаптер отключен от сети (CAN 2.0B).
**CAN_STATE_STOPPED** – адаптер остановлен; промежуточное состояние, устанавливаемое при конфигурировании, запуске, останове, перезапуске CAN-адаптера.
**CAN_STATE_SLEEPING** – состояние сна; не используется в текущей версии.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция получает текущее значение состояния CAN-адаптера.

#### 2.3.3.9. fw_can_start

**Прототип**

```
F_CAN_RESULT fw_can_start(F_CAN_HANDLE hDev)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция переводит CAN-адаптер в рабочее состояние. Адаптер подключается к сети, можно посылать и принимать кадры.

#### 2.3.3.10. fw_can_stop

**Прототип**

```
F_CAN_RESULT fw_can_stop(F_CAN_HANDLE hDev)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция переводит CAN-адаптер в состояние **CAN_STATE_INIT**. Адаптер отключается от сети (не участвует в обмене данными). Возможно конфигурирование интерфейса.

### 2.3.3.11. fw_can_send

**Прототип**

```
F_CAN_RESULT fw_can_send(  F_CAN_HANDLE hDev,
                           PF_CAN_TX pTx,
                           size_t nTx,
                           size_t* nSend)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**PF_CAN_TX pTx**

Указатель на буфер передаваемых сообщений.

**size_t nTx**

Количество сообщений в буфере **pTx** .

**size_t* nSend**

Указатель на переменную, в которую будет помещено количество реально переданных сообщений при возврате из данной функции.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция передает сообщения в шину CAN.

Алгоритм отправки выглядит следующим образом:

1. Если очередь заданий на отправку не пуста, задание добавляется в конец очереди.
2. Если очередь пуста (аналогично, после того как будут выполнены все задания впереди по очереди), задание начинает выполняться драйвером: первый кадр загружается в регистры CAN-адаптера и выставляется запрос на передачу кадра.
3. После успешной отправки кадра (получено подтверждение конца операции), драйвер начинает передачу следующего кадра.
4. И так далее, до тех пор, пока не будет отправлен последний кадр задания или не истечет интервал времени определенный таймаутом передачи, который вычисляется для задания по формуле:

   **Tsend = (nTx * WriteTotalTimeoutMultiplier + WriteTotalTimeoutConstant) (мс)**

Значения конфигурационных параметров **WriteTotalTimeoutMultiplier** и **WriteTotalTimeoutConstant** могут быть получены функцией **fw_can_get_timeouts()** и заданы функцией **fw_can_set_timeouts()**. При нулевых значениях параметров таймаут передачи будет бесконечным (значения по умолчанию равны 0!).

Функция блокируется на время ожидания в очереди и время передачи кадров задания или до тех пор, пока задание не будет отменено функцией **fw_can_purge()** (см. п. 2.3.3.17). Из-за возможных ошибок передачи (например, обрыв кабеля, нет других контроллеров в сети, несоответствие выставленных скоростей передачи у контроллеров сети) и истечения интервала ожидания, количество реально переданных сообщений может отличаться от числа сообщений в задании и должно быть проанализировано при завершении данной функции, даже если она вернула **CAN_RES_OK.**

### 2.3.3.12. fw_can_recv

#### Прототип

```
F_CAN_RESULT fw_can_recv(F_CAN_HANDLE hDev, PF_CAN_RX pRx,
   size_t szRx, size_t* nRecv)
```

#### Параметры

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**PF_CAN_TX pRx**

Указатель на буфер приема в приложении.

**size_t nRx**

Запрашиваемое количество кадров (емкость буфера приема **pRx**).

**size_t* nSend**

Указатель на переменную, в которую будет помещено количество сообщений, записанных в буфер приема при возврате из данной функции.

#### Возвращаемое значение

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

#### Комментарии

Функции считывает сообщения из приемного буфера CAN-адаптера.

Алгоритм работы функции выглядит следующим образом:

1. Если при вызове **fw_can_recv()** в приемном буфере имеются сообщения, **fw_can_recv()** прочитает их и немедленно завершится. Если количество кадров в приемном буфере было меньшим, чем **nRx**, записывается столько кадров, сколько их было в приемном буфере.
2. Если внутренний буфер пуст и очередь заданий на чтение сообщений не пуста, задание добавляется в конец очереди.
3. Если очередь пуста (аналогично, после того как будут выполнены все задания на прием впереди по очереди), задание начнет выполняться драйвером. Задание будет выполняться до тех пор пока не будет принят первый же кадр или не истечет интервал времени определенный таймаутом приема:

   **Trecv = ReadTotalTimeout (мс)**

Значение конфигурационного параметра **ReadTotalTimeout** может быть получено функцией **fw_can_get_timeouts()** и задано функцией **fw_can_set_timeouts().** При нулевом значении параметра таймаут приема равен 0 (значение по умолчанию равно 0!).

Функция блокируется на время ожидания в очереди и время приема или до тех пор, пока задание не будет отменено функцией **fw_can_purge()** (см. п. 2.3.3.17).

#### 2.3.3.13. fw_can_post_message

**Прототип**

```
F_CAN_RESULT fw_can_post_message (F_CAN_HANDLE hDev, PF_CAN_TX pTx)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**PF_CAN_TX pTx**

Указатель на буфер передаваемого сообщения.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция помещает сообщение в аппаратный буфер передачи CAN-адаптера.

Функция не блокируется и не ожидает завершения передачи сообщения.

Алгоритм работы функции выглядит следующим образом. Сначала проверяется, свободен ли аппаратный буфер передачи. Если буфер занят, возвращается код ошибки **CAN_RES_FAILURE**. Если буфер свободен, то кадр загружается в регистры CAN-адаптера, выставляется запрос на передачу кадра, и функция, не дожидаясь реальной отправки, завершается с кодом успеха.

Результат реальной отправки или не отправки кадра может быть получен приложением позже путем вызова функцией **fw_can_wait().**

#### 2.3.3.14. fw_can_peek_message

**Прототип**

```
F_CAN_RESULT fw_can_peek_message(F_CAN_HANDLE hDev, PF_CAN_RX pRx)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**PF_CAN_TX pRx**

Указатель на буфер приема сообшения в приложении.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функции считывает одно сообщение из приемного буфера CAN-адаптера. Функция не блокируется и немедленно завершается при любом состоянии буфера.

#### 2.3.3.15. fw_can_wait

**Прототип**

```
F_CAN_RESULT fw_can_wait(F_CAN_HANDLE hDev, PF_CAN_WAIT pWait, size_t msec)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**PF_CAN_WAIT pWait**

Указатель на объект типа **F_CAN_WAIT**, определенный следующим образом:

```
typedef struct _F_CAN_WAIT
{
  F_CAN_STATUS  waitMask;
  F_CAN_STATUS  status;
} F_CAN_WAIT, *PF_CAN_WAIT;
```

Поле **waitMask** содержит маску ожидаемого статуса. В поле **status** будет возвращено текущее значения статуса CAN-адаптера. Статус CAN-адаптера определяется битовой маской флагов:

**CAN_STATUS_RXBUF** – (=1: одно и более принятых сообщений доступны для чтения; =0: нет принятых сообщений);
**CAN_STATUS_TXBUF** – (=1: внутренний буфер передачи CAN-адаптера свободен, возможна запись в буфер передачи; =0: внутренний буфер передачи CAN-адаптера занят, некоторое сообщение ожидает отправки или находится в процессе отправки).
**CAN_STATUS_ERR** – (=1: произошла ошибка с момента последнего сброса счетчика ошибок; =0: с момента последнего сброса счетчиков ошибок новых ошибок не произошло).

**size_t msec**

Таймаут ожидания в миллисекундах.

#### Возвращаемое значение

**CAN_RES_OK** – успех;

**CAN_RES_TIMEOUT** – таймаут. После истечения времени ожидания текущий статус не соответствует ожидаемому;

значение, отличное от **CAN_RES_OK** и **CAN_RES_TIMEOUT**, свидетельствует об ошибке.

#### Комментарии

Функция блокирует выполнение вызывающего потока до тех пор, пока статус CAN-адаптера не будет соответствовать одному из значений, определенных маской **pWait->waitMask** (в приемном буфере есть сообщения, аппаратный буфер передачи свободен, ошибка), или не истечет таймаут заданный параметром **msec**. При завершении данной функции с кодом **CAN_RES_OK** или **CAN_RES_TIMEOUT** поле **pWait->status** будет содержать текущее значения статуса.

### 2.3.3.16. fw_can_get_clear_errors

#### Прототип

```
F_CAN_RESULT fw_can_get_timeouts(F_CAN_HANDLE hDev, PF_CAN_ERRORS pErrors)
```

#### Параметры

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**PF_CAN_ERRORS pErrors**

Указатель на переменную типа **F_CAN_ERRORS**, в которую, при успешном вызове, будут записаны текущие значения счетчиков ошибок. Описание полей структуры и их назначение приведены в п. 2.3.3.23.

#### Возвращаемое значение

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

#### Комментарии

Функция помещает значения счетчиков ошибок данного адаптера в объект по указателю **pErrors** и сбрасывает счетчики. Если в качестве **pErrors** передать **NULL**, будет произведен только сброс счетчиков.

#### 2.3.3.17. fw_can_purge

**Прототип**

```
F_CAN_RESULT fw_can_purge(F_CAN_HANDLE hDev, F_CAN_PURGE_MASK flags)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**F_CAN_PURGE_MASK flags**

Флаги требуемых операций:

**CAN_PURGE_TXABORT** – очистить очередь заданий на отправку сообщений.
**CAN_PURGE_TXCLEAR** – сбросить в адаптере текущий запрос на передачу.
**CAN_PURGE_RXABORT** – очистить очередь заданий на чтение входящих сообщений.
**CAN_PURGE_RXCLEAR** – очистить внутренний буфер приема.
**CAN_PURGE_HWRESET** – выполнить аппаратный сброс CAN-адаптера.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция выполняет отмеченные операции: очищает буферы приема и/или передачи и/или выполняет аппаратный сброс CAN-адаптера.

#### 2.3.3.18. fw_can_get_stats

**Прототип**

```
F_CAN_RESULT fw_can_get_stats(F_CAN_HANDLE hDev, PF_CAN_STATS pStats)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**PF_CAN_STATS pStats**

Указатель на переменную типа **F_CAN_STATS**, в которую, при успешном вызове, будет записана статистическая информация CAN-адаптера. Описание полей структуры и их назначение приведены в п. 2.3.3.22.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция возвращает текущую статистическую информацию CAN-адаптера.

#### 2.3.3.19. fw_can_clear_stats

**Прототип**

```
F_CAN_RESULT fw_can_clear_stats(F_CAN_HANDLE hDev)
```

**Параметры**

**F_CAN_HANDLE hDev**

Хэндл адаптера.

**Возвращаемое значение**

**CAN_RES_OK** – успех;

значение, отличное от **CAN_RES_OK**, свидетельствует об ошибке.

**Комментарии**

Функция сбрасывает в 0 статистическую информацию CAN-адаптера.

### 2.3.3.20. F_CAN_SETTINGS

Структура **F_CAN_SETTINGS** описывает конфигурацию CAN-адаптера и используется в функциях **fw_can_get_controller_config**, **fw_can_set_controller_config**.

```
typedef struct _F_CAN_SETTINGS
{
  F_CAN_CONTROLLER  controller_type;
  F_CAN_BAUDRATE    baud_rate;
  u16               opmode;
  u32               acceptance_code;
  u32               acceptance_mask;
  u32               error_mask;
} F_CAN_SETTINGS, *PF_CAN_SETTINGS;
```

Поле **controller_type** является справочным и определяет тип CAN-контроллера, используемого аппаратурой CAN-адаптера. Значение устанавливается при вызове **fw_can_get_controller_config** и не используется при вызове **fw_can_set_controller_config**.

Поле **baud_rate** определяет скорость передачи данных в сети. В заголовочном файле can.h определены стандартные значения скорости передачи:

**CANBR_1Mbaud** – 1 MBit/sec,

**CANBR_800kBaud** – 800 kBit/sec,

**CANBR_500kBaud** – 500 kBit/sec,

**CANBR_250kBaud** – 250 kBit/sec,

**CANBR_125kBaud** – 125 kBit/sec,

**CANBR_100kBaud** – 100 kBit/sec,

**CANBR_50kBaud** – 50 kBit/sec,

**CANBR_20kBaud** – 20 kBit/sec,

**CANBR_10kBaud** – 10 kBit/sec.

Для задания нестандартной скорости передачи используется макроопределение

**CANBR_Btr0Btr1(btr0, btr1)**

Параметры **btr0, btr1** определяют значения регистров BTR0, BTR1 CAN контроллера Philips SJA 1000 с частотой задающего генератора 16 Мгц. Более подробная информация по конфигурированию скорости передачи и описание битовых полей регистров приведены в разделе 6.5 спецификации Philips SJA 1000. В заголовочном файле can.h можно также найти таблицы соответствия значений указанных регистров для набора стандартных и нескольких нестандартных скоростей.

Поле **opmode** определяет режим работы CAN-адаптера. Режим задается битовой маской флагов:

**CAN_OPMODE_STANDARD** – принимаются кадры стандартного формата (11-битовый идентификатор).
**CAN_OPMODE_EXTENDED** – принимаются кадры расширенного формата (29-битовый идентификатор).
**CAN_OPMODE_ERRFRAME** – используется механизм индикация ошибок посредством отправки драйвером в приемный буфер специального CAN-сообщения. Формат специального сообщения определен в заголовочном файле can.h.
**CAN_OPMODE_LSTNONLY** – функционирование в режиме "Listen Only" (мониторинг шины). Адаптер не передает сообщения в шину и не устанавливает признаки подтверждений или ошибок. Программа может использовать данный режим для “горячего подключения к шине” с автоопределением скорости передачи.

**CAN_OPMODE_SELFTEST** – в этом режиме возможно выполнение полного тестирования адаптера без наличия на шине других активных устройств. Адаптер считает передачу кадра выполненной, даже при отсутствии сигнала подтверждения на шине.
**CAN_OPMODE_SELFRECV** – адаптер принимает собственные передаваемые сообщения (при условии, что они удовлетворяют заданному фильтру).

При определении режима значения флагов комбинируются побитовым ИЛИ. Например, сочетание **CAN_OPMODE_STANDARD | CAN_OPMODE_EXTENDED** установит для адаптера режим работы с обоими форматами кадров.

Поля **acceptance_code** и **acceptance_mask** определяют значение фильтра принимаемых кадров. Входные сообщения фильтруются в соответствии со значениями их идентификаторов.

Сообщение, удовлетворяющее фильтру, добавляется в приемный буфер, в противном случае игнорируется. Значение параметра **acceptance_code** определяет битовый шаблон идентификаторов сообщений, подлежащих приему. Значение параметра **acceptance_mask** определяет значимые при сравнении битовые поля. Если бит в **acceptance_mask** установлен в 0, соответствующий ему бит не используется при сравнении. В противном случае, если бит, равен 1, значение соответствующего ему бита в **acceptance_code** является значимым.

В табл. 3 и 4 представлена связь между битовыми полями в параметрах фильтра (**acceptance_code** и **acceptance_mask**) и битовыми полями идентификатора сообщения.

**Таблица 3. Значения битовых полей фильтра для CAN-кадров стандартного формата**

| Бит | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ID10 | ID9 | ID8 | ID7 | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | RTR |

**Таблица 4. Значения битовых полей фильтра для CAN-кадров расширенного формата**

| Бит | 29 | 28 | 27 | 26 | 25 | … | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ID28 | ID27 | ID26 | ID25 | ID24 | … | ID4 | ID3 | ID2 | ID1 | ID0 | RTR |

Биты 11–1 (29–1) фильтра соответствуют битам 10–0 (28–0) CAN_ID сообщения.

Бит 0 фильтра соответствует биту RTR сообщения.

Для сообщений стандартного формата являются значимыми только 12 младших бит фильтра. Для сообщений расширенного формата являются значимыми 30 младших бит фильтра.

Следующий пример иллюстрирует работу фильтра для сообщений стандартного формата:

| | |
|---|---|
| acceptance_code | 010 0000 0000 0 |
| acceptance_mask | 111 1111 1100 1 |
| Допустимые ID: | 010 0000 00xx 0 |
| ID 200h, RTR = 0 | 010 0000 0000 0 |
| ID 201h, RTR = 0 | 010 0000 0001 0 |
| ID 202h, RTR = 0 | 010 0000 0010 0 |
| ID 203h, RTR = 0 | 010 0000 0011 0 |

Поле **error_mask** определяет фильтр ошибок, передаваемых специальными сообщениями об ошибках при работе в режиме **CAN_OPMODE_ERRFRAME.**

После открытия устройства функцией **fw_can_open()** параметры конфигурации CAN-адаптера принимают следующие начальные значения:

режим работы – **CAN_OPMODE_STANDARD|CAN_OPMODE_EXTENDED**

скорость передачи – **CANBR_250kBaud**

фильтра принимаемых кадров – настроен на прием всех возможных сообщений

фильтр ошибок – **CAN_ERR_CRTL|CAN_ERR_BUSOFF.**

### 2.3.3.21. F_CAN_TIMEOUTS

Структура **F_CAN_TIMEOUTS** используется в функциях **fw_can_get_controller_config()** и **fw_can_set_controller_config()** для получения текущих и установки новых значений временных параметров CAN-интерфейса. Значения всех временных параметров представлены в миллисекундах.

```
typedef struct _F_CAN_TIMEOUTS
{
  u32 WriteTotalTimeoutMultiplier;
  u32 WriteTotalTimeoutConstant;
  u32 ReadTotalTimeout;
  u32 RestartBusoffTimeout;
} F_CAN_TIMEOUTS, *PF_CAN_TIMEOUTS;
```

Поля **WriteTotalTimeoutMultiplier** и **WriteTotalTimeoutConstant** определяют значение таймаута передачи CAN-кадров функцией **fw_can_send()**, который вычисляется по формуле:

```
Tsend = N * WriteTotalTimeoutMultiplier + WriteTotalTimeoutConstant (мс),
```

где N – количество отправляемых сообщений. Алгоритм отправки кадров функцией **fw_can_send()** описан в п. 2.3.3.11.

Поле **ReadTotalTimeout** определяет значение таймаута, используемого функцией **fw_can_recv()** при чтении данных. Алгоритм работы функции **fw_can_recv()** описан в п. 2.3.3.12.

Поле **RestartBusoffTimeout** определяет значение таймаута, используемого алгоритмом автоматического восстановления CAN-адаптера из состояния **CAN_STATE_BUS_OFF**. Если таймаут равен 0, то восстановление не производится. В противном случае, после истечения временного интервала определенного таймаутом будет произведен аппаратный сброс адаптера.

Непосредственно после открытия устройства функцией **fw_can_open()** таймауты имеют нулевые значения.

### 2.3.3.22. F_CAN_STATS

Структура **F_CAN_STATS** используется для описания статистики функционирования CAN-адаптера.

```
typedef struct _F_CAN_STATS
{
  // кол-во принятых кадров
  u32 rx_packets;
  // кол-во принятых байт
  u32 rx_bytes;
  // кол-во переданных кадров
  u32 tx_packets;
  // кол-во переданных байт
  u32 tx_bytes;
  // кол-во ошибок приема
  u32 rx_errors;
  // кол-во ошибок передачи
  u32 tx_errors;
  // кол-во переполнений буфера передачи
  u32 tx_over_errors;
  // кол-во ошибок протокола (кадрирование, CRC, и т.д.)
  u32 bus_errors;
  // кол-во переходов в CAN_STATE_ERROR_WARNING
  u32 error_warning;
  // кол-во переходов в CAN_STATE_ERROR_PASSIVE
  u32 error_passive;
  // кол-во переходов в CAN_STATE_BUS_OFF
  u32 bus_off;
  // кол-во "проигрышей" арбитража
  u32 arbitration_lost;
  // кол-во перезапусков адаптера
  u32 restarts;
} F_CAN_STATS, *PF_CAN_STATS;
```

### 2.3.3.23. F_CAN_ERRORS

Структура **F_CAN_ERRORS** используется в функции **fw_can_get_clear_errors()** для получения текущих значений счетчиков ошибок адаптера.

```
typedef struct _F_CAN_ERRORS
{
  // кол-во таймаутов передачи
```

```
  size_t tx_timeouts;
  // кол-во переполнений буфера приема
  size_t data_overrun;
  // кол-во переходов в CAN_STATE_ERROR_PASSIVE
  size_t error_passive;
  // кол-во переходов в CAN_STATE_BUS_OFF
  size_t bus_off;
} F_CAN_ERRORS, *PF_CAN_ERRORS;
```

Вызов функции **fw_can_get_clear_errors()** обнуляет значения счетчиков.

### 2.4. Пример программирования

Исходный текст тестового примера программирования контроллера CAN-интерфейсов (can_test.cpp) находится в подкаталоге \src\test каталога установки пакета поддержки. Кроме того, в подкаталоге \src расположены вспомогательные заголовочные файлы и модули исходных текстов. Статическая библиотека связи приложения со вспомогательной динамической библиотекой fwcan.dll расположена в подкаталоге \lib\xp каталога установки пакета поддержки.

Для компиляции и компоновки исполняемого модуля примера из командной строки с использованием среды разработки, установленной на инструментальном ПК, могут быть использованы включенные в комплект поставки пакетные (командные) и конфигурационные файлы построения. Подробное их описание приведено в п. 1.2 настоящего руководства.

## ПРИЛОЖЕНИЕ 1. ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

| Версия | Дата | Ссылка | Статус | Примечания |
|---|---|---|---|---|
| 1.00 | | Документ | создан | |
| 1.10 | 04.10.2011 | Документ | изменен | Реализован механизм конфигурирования контроллера на нестандартные скорости обмена |
| 1.20 | 26.09.2012 | Документ | изменен | Для служебных сообщений добавлен признак в идентификаторе сообщения.<br>Устранены дефекты:<br>1. некорректная работа счетчика tx_timeout в структуре F_CAN_ERRORS;<br>2. ранняя установка статуса ошибки адаптера уже при переходе в состояние CAN_STATE_ERROR_WARNING, до перехода в CAN_STATE_ERROR_PASSIVE.<br>Версия драйвера 1.0.0.4. |
| | | | | |
| | | | | |
| | | | | |