« МАРАФОН »

Библиотека CHAI 2.14.0

Руководство по разработке программного обеспечения

Версия документа 0.28

2020 год, Марафон, Москва, 117330, Мосфильмовская ул., 17б
Тел.: +7 (495)-988-27-26, 939-13-24 Факс: +7 (495)-939-56-59

Гарантийные обязательства МАРАФОНА.

ГАРАНТИЙНЫЕ ОБЯЗАТЕЛЬСТВА
(Ограниченная гарантия на продукцию МАРАФОНА)

Программное обеспечение:

Гарантийное обслуживание по программному обеспечению можно получить, связавшись с офисом МАРАФОНА в оговоренный гарантийный период. Адрес офиса МАРАФОНА приведен на первой странице Руководства по эксплуатации устройства, а также приложен вместе с Регистрационной карточкой.

МАРАФОН гарантирует, что его программное обеспечение будет работать в строгом соответствии с прилагаемой к нему МАРАФОНОМ документацией в течении девяноста (90) дней с момента его приобретения у МАРАФОНА или Авторизованного Реселлера. МАРАФОН предоставляет гарантию на носитель, на котором поставляется программное обеспечение, в виде отсутствия потери им информации на тот же гарантийный срок. Данная гарантия имеет отношение только к приобретенному программному обеспечению или его замене по гарантии, и не касается любых обновлений или замен, которые получены через Internet или бесплатно.

Ответственность МАРАФОНА по обеспечению гарантии программного обеспечения состоит в замене его на новое, которое выполняет перечисленные в прилагаемой документации функции. Ответственность Заказчика состоит в выборе соответствующего приложения, программной платформы/системы и дополнительных материалов. МАРАФОН не отвечает за работоспособность программного обеспечения вместе с любыми аппаратными средствами и/или программными платформами/системами, которые поставляются третьими сторонами, если совместимость с ними не оговорена в прилагаемой к продукции МАРАФОН документации. Согласно данной гарантии, МАРАФОН старается обеспечить разумную совместимость своей продукции, но МАРАФОН не несет ответственность, если с аппаратными или программными средствами третьих фирм происходят сбои. МАРАФОН не гарантирует, что работа программного обеспечения будет непрерывна и в процессе не будут происходить ошибки, а также то, что все дефекты в программном продукте с или без учета документации на него, будут исправлены.

Ограничения гарантий

Вышеупомянутые гарантии и замечания являются исключительными и соответствуют всем прочим гарантиям, объявленным или подразумеваемым, которые даются в явном виде или в соответствии с законодательством, установленными законами или в другом виде, включая гарантии на сам товар и его пригодность для стандартных целей. МАРАФОН никогда не допускает и не принимает на себя прочую ответственность, связанную с продажами, поддержкой инсталляции или использования продукции МАРАФОНА

МАРАФОН никогда не несет ответственность по гарантии, если проводимое им тестирование и анализ определяет, что заявленный дефект в изделии не был обнаружен, или он был вызван неверным использованием заказчиком, или третьей стороной, невнимательной или неправильной инсталляцией или тестированием, попыткой ремонта неавторизованными лицами, или чем-либо еще, не предусмотренным в назначении изделия, типа несчастного случая, огня, пожара и других бедствий.

Ограничения ответственности

Ни в каком случае МАРАФОН не несет ответственность за любые убытки, включая потерю данных, потерю прибыли, стоимости покрытия или других случайных, последовательных или непрямых убытков, являющихся следствием инсталляции, сопровождения, использования, производительности, неисправности или временной неработоспособности изделий производства МАРАФОНА. Эти ограничения действуют, даже если МАРАФОН был предупрежден о возможности такого убытка.

Регистрационная карточка, прилагаемая на обратной стороне Руководства, должна быть отправлена в офис МАРАФОН по факсу, электронной почте или почтовым отправлением. Список адресов/ телефонов/ факсов офисов МАРАФОНА содержится на первой странице данного Руководства.

Юр. адрес: 117330 Москва, ул. Мосфильмовская, дом 17Б.
Факт. адрес: 119899 Москва, Ленинске горы, МГУ, НИИЯФ, д.1. стр.5.
Тел. (495)-988-27-26, 939-56-59, 939-13-24
Факс. (495)-939-56-59

E-mail: support@marathon.ru
WEB: www.marathon.ru
По техническим вопросам звоните по тел. +7 (495)-988-27-26, 939-56-59, 939-13-24 или свяжитесь с нами по email support@marathon.ru.

Лицензионное соглашение на программное обеспечение, поставляемое с CAN-интерфейсами производства МАРАФОН

Все права на программное обеспечение, аппаратное обеспечение и данное руководство принадлежат фирме Марафон и защищены законодательством Российской Федерации.

ПЕРЕД ИСПОЛЬЗОВАНИЕМ ПРИЛАГАЕМОГО ИЗДЕЛИЯ ПОКУПАТЕЛЬ ДОЛЖЕН ВНИМАТЕЛЬНО ОЗНАКОМИТЬСЯ С УСЛОВИЯМИ НАСТОЯЩЕГО СОГЛАШЕНИЯ. ИСПОЛЬЗОВАНИЕ ДАННОГО ИЗДЕЛИЯ ПОДРАЗУМЕВАЕТ ПРИНЯТИЕ ЭТИХ ПОСТАНОВЛЕНИЙ И УСЛОВИЙ. ЕСЛИ ОГОВОРЕННЫЕ УСЛОВИЯ ЯВЛЯЮТСЯ ДЛЯ ПОКУПАТЕЛЯ НЕПРИЕМЛЕМЫМИ, ОН ДОЛЖЕН НЕЗАМЕДЛИТЕЛЬНО ВЕРНУТЬ НЕИСПОЛЬЗОВАННЫЙ КОМПЛЕКТ, ПРИ ЭТОМ ЗАТРАТЫ ПОКУПАТЕЛЯ БУДУТ ВОЗМЕЩЕНЫ.

ДАННЫЙ ДОКУМЕНТ ЯВЛЯЕТСЯ ЛИЦЕНЗИОННЫМ СОГЛАШЕНИЕМ, НО НЕ СОГЛАШЕНИЕМ О ПРОДАЖЕ. МАРАФОН ЯВЛЯЕТСЯ ВЛАДЕЛЬЦЕМ ИЛИ ИМЕЕТ ЛИЦЕНЗИОННЫЕ СОГЛАШЕНИЯ С ДРУГИМИ ВЛАДЕЛЬЦАМИ АВТОРСКИХ ПРАВ НА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ВХОДЯЩЕЕ В КОМПЛЕКТ ПОСТАВКИ СВОИХ ИЗДЕЛИЙ И ПО. ПОКУПАТЕЛЬ НЕ ПРИОБРЕТАЕТ НИКАКИХ ПРАВ НА ИНТЕЛЛЕКТУАЛЬНУЮ СОБСТВЕННОСТЬ, СОДЕРЖАЩУЮСЯ В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ, ЗА ИСКЛЮЧЕНИЕМ ТЕХ, КОТОРЫЕ НАСТОЯЩЕЕ СОГЛАШЕНИЕ ПРЕДОСТАВЛЯЕТ ЕМУ В ОТНОШЕНИИ ЭТОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. ПРАВО СОБСТВЕННОСТИ НА ПРИЛАГАЕМУЮ КОПИЮ УПОМЯНУТОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, А ТАКЖЕ НА ВСЕ КОПИИ, СДЕЛАННЫЕ С НЕЕ, СОХРАНЯЕТСЯ ЗА МАРАФОНОМ ИЛИ ДРУГИМИ ВЛАДЕЛЬЦАМИ АВТОРСКИХ ПРАВ. ПОКУПАТЕЛЬ ПРИНИМАЕТ НА СЕБЯ ВСЮ ОТВЕТСТВЕННОСТЬ В ОТНОШЕНИИ ВЫБОРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ДОСТИЖЕНИЯ СВОИХ ЦЕЛЕЙ, А ТАКЖЕ ЗА УСТАНОВКУ, ИСПОЛЬЗОВАНИЕ И РЕЗУЛЬТАТЫ ИСПОЛЬЗОВАНИЯ ДАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.

Покупатель имеет право:

1. копировать Программное обеспечение исключительно для создания резервных копий или при установке для подразумеваемого обычного использования Программного обеспечения при условии, что в любой копии упомянутого Программного обеспечения будут воспроизведены все уведомления об авторских правах и торговых марках, содержащиеся в данном Программном обеспечении;

2. передавать право владения копиями Программного обеспечения другому юридическому или физическому лицу путем передачи данной копии настоящего Соглашения и всей прочей документации, а также по меньшей мере одной полной и не претерпевший изменений копии Программного обеспечения, при условии, что (1) все сделанные Покупателем копии Программного обеспечения будут переданы означенному лицу или уничтожены, (2) такая передача права владения прекращает лицензионное соглашение Покупателя с МАРАФОНОМ, и (3) означенное лицо примет на себя и будет соблюдать постановления данного лицензионного соглашения с момента начала пользования Программным обеспечением; и

3. использовать торговые марки, связанные с Программным обеспечением, исключительно в соответствии с существующей практикой использования торговых марок, включая ссылки на имена владельцев торговых марок.

Без письменного согласия МАРАФОНА запрещается:

1. использовать, копировать, изменять, объединять или передавать копии данного Программного обеспечения при условиях, отличных от оговоренных в данном соглашении;

2. деассемблировать или декомпилировать Программное обеспечение;

3. выдавать сублицензию, сдавать в аренду и лизинг, передавать в пользование данное Программное обеспечение или любую его копию.

Копирование этого руководства возможно только при получении письменного разрешения у фирмы Марафон.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

Библиотека CHAI (CAN Hardware Abstraction Interface) реализует программный интерфейс (API) доступа к сети CAN на канальном уровне (Data Link Layer) эталонной модели ISO/OSI. Интерфейс библиотеки по возможности не зависит от используемой аппаратуры CAN и операционной системы. Библиотека разработана для применения в среде операционных систем общего назначения, таких как Windows 8.1/10, Linux.

1.1 Возможности CHAI

- стандартный и расширенный протокол CAN (11 и 29-битовые идентификаторы кадров);
- битовая скорость до 1000 Кбит/с;
- настройка приемного аппаратного фильтра CAN-контроллера;
- прием CAN-кадров с использованием очереди сообщений;
- отправка кадров в сеть напрямую через регистры CAN-контроллера, либо через очередь кадров;
- регистрация времени приема кадра (отметка времени);
- ожидание наступления того или иного события (получение кадра, ошибка сети и т.п.);
- несколько CAN-контроллеров в одном устройстве;
- несколько CAN-устройств в одном компьютере.

1.2 Управление CAN-кадрами

Библиотека предоставляет собственные способы управления CAN-кадрами независимые от типа используемого контроллера.

Принятые из сети кадры помещаются в приемную очередь (FIFO), с сохранением последовательности и времени (timestamp) получения.

Отправляемые CAN-кадры помещаются в регистры CAN контроллера и отправляются в сеть, либо помещаются в очередь на отправку (FIFO) для последующей автоматической обработки контроллером посредством прерываний.

RTR-кадры обрабатываются одинаковым образом с кадрами данных.

CHAI предоставляет функцию ожидания следующих событий:

- количество кадров в приемной очереди достигло или превысило установленный порог,
- количество пустых слотов в очереди на отправку достигло или превысило установленный порог,

- произошла ошибка (Bus-off, Error Warning Limit, аппаратное переполнение контроллера, программное переполнение приемной очереди, тайм-аут при отправке кадра).

1.3 Поддержка Windows на платформе x64 (amd64)

CHAI поддерживает 64-разрядные версии Windows 8.1/10. Поддерживаются как 64-разрядные пользовательские приложения, так и 32-разрядные приложения с использованием встроенного в эти ОС механизма WoW (Windows on Windows).

2. ОБЗОР ФУНКЦИОНАЛА И ПРОГРАММНОГО ИНТЕРФЕЙСА

Каждому CAN-контроллеру ставится в соответствие один канал ввода-вывода. Каналы идентифицируются целым числом начиная с 0. Каждый канал (CAN-контроллер) рассматривается библиотекой как отдельное устройство. Например, плата CAN-bus-PCI имеет в своем составе два контроллера SJA1000 и соответственно два канала ввода-вывода. Обнаружение и перечисление (enumaration - назначение номеров) каналов производится библиотекой в автоматическом режиме во время загрузки и инициализации.

Каждый канал ввода-вывода имеет одну приемную очередь и одну очередь на передачу (примечание: устройства семейства CAN-bus-USBnp не имеют очереди на передачу до версии прошивки 1.3 включительно).

В каждом CAN-контроллере существует аппаратный приемный фильтр, позволяющий выбирать для приема кадры на основе значений их идентификаторов, разгружая тем самым центральный процессор. Кадры не прошедшие фильтр сбрасываются контроллером без оповещения.

2.1 Состояния канала ввода-вывода и ошибки сети.

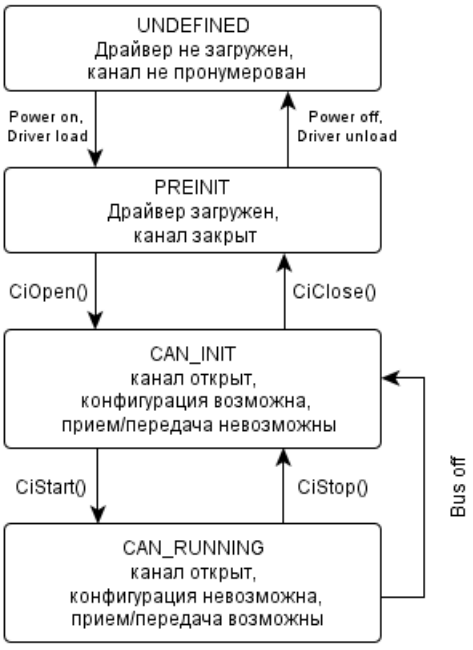

Рис. 1. Состояния и жизненный цикл канала ввода-вывода.

В процессе работы с сетью канал ввода-ввода может находится в двух состояниях CAN_INIT и CAN_RUNNING (см. рисунок 1). В состоянии CAN_INIT канал отключен от сети (прием и отправка кадров данных невозможны, кадры ошибок и подтверждения не посылаются) и может быть сконфигурирован (настройка скорости передачи, приемного фильтра, параметры очередей и т. п.). Для передачи-приема данных канал должен быть переведен в состояние CAN_RUNNING, в котором конфигурация канала невозможна.

Переключение канала из одного состояния в другое производится вызовами CiStart [3.4] и CiStop [3.5]. Кроме того, канал автоматически переходит из состояния CAN_RUNNING в CAN_INIT при наступлении события Bus off (одна из возможных ошибок CAN-сети).

Сеть CAN имеет алгоритм обнаружения и ограничения ошибок, который встроен на аппаратном уровне в каждый CAN-контроллер. Контроллер имеет два аппаратных счетчика ошибок: один на передачу, другой на прием. В зависимости от поведения сети и контроллера эти счетчики увеличиваются или уменьшаются и при пересечении определенных границ могут вызывать аппаратное прерывание (оповещение) и переводить контроллер в состояние Bus off (отключен от сети). Подробно ошибки протокола CAN и алгоритм их обнаружения описаны в спецификации CAN 2.0 Bosch [Лит. 3].

Ниже перечислены ошибки сети и ошибки канала ввода-вывода, которые могут возникать в процессе работы библиотеки:

- EWL (Error Warning Level) - один из аппаратных счетчиков превысил уровень 96 (как правило говорит о несовпадении скоростей передачи узлов в сети, или возникновении помех в сети, так же возникает при попытке передачи кадра в сеть, где отсутствуют другие узлы);
- BOFF (Bus off) - один из аппаратных счетчиков достиг уровня 255, CAN-контроллер отключен от сети и переведен в состояние CAN_INIT;
- HOVR – переполнение аппаратного приемного буфера контроллера, один или несколько кадров были потеряны (кадры поступают из сети слишком быстро — драйвер не успевает вынимать их из аппаратного буфера);
- SOVR – переполнение программной приемной очереди драйвера, один или несколько кадров были потеряны (приложение вынимает кадры из приемной очереди слишком медленно);
- WTOUT – тайм-аут передачи кадра в сеть при помощи функции CiWrite;

2.2 Программный интерфейс.

Перед началом работы библиотека должна быть инициализирована функцией CiInit, которая должна вызываться один раз перед любым другим вызовом CHAI.

Для работы с сетью канал ввода-ввода должен быть открыт (CiOpen [3.2]), после окончания работы — канал должен быть закрыт (CiClose [3.3]).

В целях переносимости в заголовочном файле chai.h определены следующие типы данных:

- _u8 - беззнаковое целое длины 8 бит (1 байт)
- _s8 - знаковое целое длины 8 бит (1 байт)
- _u16 - беззнаковое целое длины 16 бит (2 байта)
- _s16 - знаковое целое длины 16 бит (2 байта)
- _u32 - беззнаковое целое длины 32 бит (4 байта)
- _s32 - знаковое целое длины 32 бит (4 байта)

Функции программного интерфейса библиотеки, как правило, возвращают ноль или положительное целое (_s16) в случае успешного выполнения; в случае ошибки – отрицательное целое, модуль которого равен коду ошибки определенной в заголовочном файле chai.h:

- ECIGEN - generic (not specified) error
- ECIBUSY - device or resource busy
- ECIMFAULT - memory fault
- ECISTATE - function can't be called for object in current state
- ECIINCALL - invalid call, function can't be called for this object
- ECIINVAL - invalid parameter
- ECIACCES - can not access resource
- ECINORES - no resources
- ECINOSYS - function or feature not implemented
- ECIIO - input/output error
- ECINODEV - no such device
- ECIINTR - call was interrupted by event
- ECITOUT - time out occurred

Структура данных для представления CAN-кадра имеет следующий прототип:

```
typedef struct {
  _u32 id;          /* идентификатор кадра */
  _u8 data[8];      /* данные */
  _u8 len;          /* фактическая длина поля данных,
                       от 0 до 8 байт */
  _u16 flags;       /* bit 0 - RTR,
                       bit 2 - EFF */
  _u32 ts;          /* отметка времени получения
                      (timestamp) в микросекундах */
} canmsg_t;


Примечания:

если бит 0 поля flags выставлен в 1 – кадр помечен как RTR

если бит 2 поля flags выставлен в 1 – кадр помечен как EFF (Extended
Frame Format, идентификатор - 29 бит)

время, указываемое в поле ts, измеряется от момента открытия канала в
микросекундах,  переполнение этого поля наступает примерно через 71
минуту после открытия канала; после переполнения время вновь
отсчитывается от нуля.
```

Для удобства работы с типом данных canmsg_t в библиотеке определены вспомогательные функции:

- `void msg_zero (canmsg_t *msg)` - обнуляет кадр msg; после вызова msg представляет собой кадр стандартного формата (SFF - standart frame format, идентификатор - 11 бит), длина поля данных - ноль, данные и все остальные поля равны нулю;
- `_s16 msg_isrtr (canmsg_t *msg)` - возвращает не ноль (true), если msg - RTR кадр, в противном случае msg - кадр данных;
- `void msg_setrtr (canmsg_t *msg)` - помечает msg как RTR кадр;
- `_s16 msg_iseff (canmsg_t *msg)` - возвращает не ноль (true), если msg - кадр расширенного формата (EFF - extended frame format, идентификатор - 29 бит), в противном случае msg - кадр стандартного формата (SFF - standart frame format, идентификатор - 11 бит);
- `void msg_seteff (canmsg_t *msg)` - помечает msg как кадр расширенного формата.

2.3 Типичная последовательность вызовов.

- инициализация:
  CiInit [3.1];
- получение списка обнаруженных устройств CAN
  CiBoardInfo [3.29];
- открытие канала:
  CiOpen [3.2];
- конфигурирование канала и очередей:
  CiSetBaud [3.7],
  CiSetFilter [3.6],
  CiRcQueResize [3.17],
  CiRcQueTreshold [3.16],
  CiTrQueTreshold [3.15];
- запуск канала:
  CiStart [3.4];
- ожидание наступления события:
  CiWaitEvent [3.14],
  работа с сетью через операции ввода-вывода:
  CiRead [3.12], CiTransmit [3.8],
  чтение ошибок сети:
  CiErrsGetClear [3.13],
  чтение состояний процессов отправки/приема:
  CiTrStat [3.11], CiRcQueGetCnt [3.19];
- остановка канала:
  CiStop [3.5];
- закрытие канала:
  CiClose [3.3].

2.4 Использование библиотеки.

Каждая программа, работающая с CHAI должна включать заголовочный файл CHAI_BASE\chai.h, который содержит необходимые объявления типов данных и прототипов функций библиотеки. При сборке пользовательское приложение должно прилинковывать файл CHAI_BASE\lib\chai.lib для ОС Windows; и файл CHAI_BASE/lib/libchai.so для ОС Linux. Здесь CHAI_BASE – установочная директория программного пакета CHAI. Например, для компиляции вашей программы в ОС Linux вы можете использовать команду (предполагается, что библиотека CHAI установлена в /opt/chai-2.14.0):

```
gcc -I/opt/chai-2.14.0/include -L/opt/chai-2.14.0/lib -o your_app your_app.c -lchai
```

Для ОС Windows текущая версия библиотеки разрабатывается в среде MS Visual C++ 2017 Community, поэтому формат файла chai.lib соответствует формату этой версии среды разработки.

Примеры параметров сборки и исходные коды программ использующих CHAI находятся в директории CHAI_BASE/ex (can-echowt.c), и в директории CHAI_BASE/src (chaitest.c и canmon.c).

3. СПИСОК ФУНКЦИЙ

3.1 _s16 CiInit(void)

Описание:

Инициализирует внутренние структуры данных библиотеки. Должна вызываться **один раз** в самом начале пользовательского приложения до вызова любой другой функции библиотеки CHAI.

Параметры:

Нет.

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

3.2 _s16 CiOpen(_u8 chan, _u8 flags)

Описание:

Открывает канал ввода-вывода для дальнейшего использования. После успешного выполнения (в случае если параметр flags равен нулю) канал находится в следующем **начальном состоянии**:

- CAN-контроллер находится в состоянии CAN_INIT,
- скорость передачи — 500 Кбит/сек,
- формат CAN-кадра - SFF (11 бит),
- фильтр контроллера настроен на прием всех возможных кадров,
- значение тайм-аута на передачу CI_WRITE_TIMEOUT_DEF (определено в chai.h),
- размеры очередей на прием/передачу равны CIQUE_DEFSIZE_RC/CIQUE_DEFSIZE_TR (определено в chai.h),
- пороги очередей на прием/передачу равны CIQUE_RC_THRESHOLD_DEF/ CIQUE_TR_THRESHOLD_DEF (определено в chai.h).

Параметры:

- chan - номер открываемого канала
- flags - флаги, возможные значения:
  CIO_CAN11 - стандартный формат кадра (идентификатор - 11 бит, по умолчанию),
  CIO_CAN29 - расширенный формат кадра (идентификатор - 29 бит).
  Значения флагов можно комбинировать побитовым ИЛИ, например:
  CIO_CAN11|CIO_CAN29 - канал будет открыт в режиме работы с обоими форматами кадра.

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов;
- ECINODEV - номеру канала, переданному в качестве параметра, не соответствует ни один из обнаруженных CAN-контроллеров;
- ECIBUSY - канал уже открыт (занят);
- ECIMFAULT - не удается выделить память для структур данных канала;

3.3 _s16 CiClose(_u8 chan)

Описание:

Закрывает канал ввода-вывода. После этого вызова аннулируются все функции - обработчики событий этого канала, CAN-контроллер переводится в состояние CAN_INIT (остановлен).

Параметры:

- chan - номер закрываемого канала

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал уже закрыт (не был открыт);
- ECINODEV - номеру канала, переданному в качестве параметра, не соответствует ни один из обнаруженных CAN-контроллеров;

3.4 _s16 CiStart(_u8 chan)

Описание:

Переводит канал ввода-вывода в рабочее состояние CAN_RUNNING (можно отправлять и принимать кадры).

Параметры:

- chan - номер канала

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;

3.5 _s16 CiStop(_u8 chan)

Описание:

Переводит канал ввода-вывода в состояние CAN_INIT (разрешено конфигурирование канала, CAN-контроллер не участвует в обмене данными сети).

Параметры:

- chan - номер канала

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;

3.6 _s16 CiSetFilter(_u8 chan, _u32 acode, _u32 amask)

Описание:

Устанавливает приемный фильтр (acceptance filter) CAN-контроллера. Приемный фильтр представляется двумя значениями: acode и amask, которые накладываются на идентификатор принимаемого кадра. В amask единицами помечаются те позиции, в которых биты идентификатора кадра должны равняться битам acode в тех же позициях для успешного прохождения фильтра, и как следствие попадания в приемную очередь. Кадры не прошедшие фильтр сбрасываются без оповещения.

Действие функции CiSetFilter зависит от значения параметра flags передаваемого в функцию CiOpen.

Если канал открыт в режиме CIO_CAN11 или CIO_CAN11|CIO_CAN29 (только 11-битовые идентификаторы или совместно 11-битовые и 29-битовые идентификаторы), тогда CiSetFilter устанавливает значение фильтра только для 11-битовых идентификаторов, при этом действие фильтра на кадры с 29-битовыми идентификаторами не специфицируется. Если требуется установить фильтр для 29-битовых идентификаторов, необходимо открывать канал в режиме только CIO_CAN29.

Требуемое состояние канала ввода-вывода: CAN_INIT

Параметры:

- acode - значение фильтра,
- amask - маска фильтра.

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;
- ECISTATE - канал не находится в состоянии CAN_INIT;
- ECIMFAULT - системная ошибка памяти (не удается скопировать параметры или результаты);

3.7 _s16 CiSetBaud(_u8 chan, _u8 bt0, _u8 bt1)

Описание:

Устанавливает скорость передачи CAN-контроллера канала.

Требуемое состояние канала ввода-вывода: CAN_INIT

Параметры:

- bt0 - значение регистра bt0 CAN-контроллера,
- bt1 - значение регистра bt1 CAN-контроллера.

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;
- ECISTATE - канал не находится в состоянии CAN_INIT;
- ECIMFAULT - системная ошибка памяти (не удается скопировать параметры или результаты);

Примечание:

В заголовочном файле chai.h определены стандартные (рекомендованные CiA) скорости передач: BCI_1M, BCI_800K, BCI_500K, BCI_250K, BCI_125K, BCI_50K, BCI_20K, BCI_10K.

Эти мнемонические значения содержат в себе сразу значения bt0 и bt1. Поэтому при их использовании вместо двух параметров bt0 и bt1 указывается одно мнемоническое значение скорости. Пример, выставить скорость передачи канала 1 в 500 Кбит/сек:

```
CiSetBaud(1, CI_500K);
```

Подробно расчет скорости передачи CAN-контроллера описан в документе «Determination of Bit Timing Parameters for the CAN Controller SJA 1000 AN97046» [Лит. 2].

3.8 _s16 CiTransmit(_u8 chan, canmsg_t * mbuf)

Описание:

Отправляет кадр в сеть через очередь на отправку.

Функция не доступна для устройств CAN-bus-USBnp с версией прошивки меньше 1.3 включительно.

Требуемое состояние канала ввода-вывода: CAN_RUNNING

Параметры:

- chan - номер канала;
- mbuf - указатель на буфер в котором находится CAN-кадр;

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- EUCNORES - нет места в очереди;
- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;
- ECISTATE - канал не находится в состоянии CAN_RUNNING, или попытка отправки кадра неподходящего формата (например, канал был открыт в режиме «только CIO_CAN11» и произошла попытка отправки кадра расширенного формата);
- ECIMFAULT - системная ошибка памяти (не удается скопировать параметры или результаты);

Примечание:

Тайм-аут на передачу (функция CiWriteTout [3.24]) не влияет на работу CiTransmit!

Алгоритм отправки кадра.

Если очередь на передачу не пуста, то кадр помещается в конец очереди, возвращается код успеха. В противном случае, если аппаратный буфер на отправку свободен, то кадр загружается в регистры CAN-контроллера и выставляется запрос на передачу кадра, если буфер занят, кадр помещается в очередь.

3.9 _s16 CiTransmitSeries(_u8 chan, canmsg_t * mbuf, int cnt, int *chaierr)

Описание:

Отправляет серию кадров в сеть через очередь на отправку.

Функция не доступна для устройств CAN-bus-USBnp с версией прошивки меньше 1.4 включительно.

Требуемое состояние канала ввода-вывода: CAN_RUNNING

Параметры:

- chan - номер канала;
- mbuf - указатель на буфер (массив) в котором находится серия CAN-кадров;
- cnt – кол-во кадров в буфере;
- chaierr – указатель на переменную куда будет сохранен результат выполнения (ошибка), возможные ошибки аналогичны функции CiTransmit;

Возвращаемое значение:

>=0 — количество успешно записанных в очередь кадров, может быть меньше чем запрошенное количество cnt;

Примечание:

Тайм-аут на передачу (функция CiWriteTout [3.24]) не влияет на работу CiTransmitSeries!

Основная цель данной функции уменьшить задержки между посылками отдельных кадров для устройства CAN-bus-USB (поддерживается начиная с версии прошивки 1.5).

Алгоритм отправки кадра.

В цикле для каждого кадра вызывается функция CiTrasnmit(). В случае возникновения ошибки очередного вызова CiTrasnmit(), цикл прерывается, ошибка записывается в переменную chaierr, возвращается количество успешно записанных в очередь кадров.

3.10 _s16 CiTrCancel(_u8 chan, _u16 * trqcnt)

Описание:

Сбрасывает текущий запрос на передачу и очищает очередь на отправку.

Функция не доступна для устройств CAN-bus-USBnp с версией прошивки меньше 1.3 включительно.

Параметры:

- chan - номер канала;
- trqcnt - указатель на переменную, куда записывается количество стертых кадров в очереди на передачу;

Возвращаемое значение:

< 0 - ошибка

> 0 - успешное выполнение; коды успешного выполнения

- CI_TRCANCEL_NOTRANSMISSION – нет текущего запроса на передачу;
- CI_TRCANCEL_ABORTED – текущий запрос на передачу был сброшен (кадр не отправлен в сеть);
- CI_TRCANCEL_TRANSMITTED – текущий запрос на передачу не был сброшен (кадр отправлен в сеть).

3.11 _s16 CiTrStat(_u8 chan, _u16 * trqcnt)

Описание:

Возвращает текущее состояние процесса отправки кадров канала ввода-вывода.

Параметры:

- chan - номер канала
- trqcnt - указатель на переменную, куда записывается количество кадров находящихся в очереди на отправку;

Возвращаемое значение:

< 0 - ошибка

>0 – успешное выполнение; коды успешного выполнения:

- CI_TR_INCOMPLETE – контроллер передает кадр в сеть;
- CI_TR_COMPLETE_OK – последняя передача в сеть была успешной;
- CI_TR_COMPLETE_ABORT – последняя передача в сеть была сброшена.

3.12 _s16 CiRead(_u8 chan, canmsg_t *mbuf, _s16 cnt)

Описание:

Вынимает cnt кадров из очереди на прием и сохраняет их в буфере mbuf. Если в приемной очереди кадров меньше чем cnt, сохраняет столько кадров сколько есть в очереди. Функция возвращает управление сразу (не ожидает приема из сети запрошенного количества кадров cnt).

Требуемое состояние канала ввода-вывода: CAN_RUNNING

Параметры:

- chan - номер канала
- mbuf - указатель на буфер в который будут скопированы кадры
- cnt – запрашиваемое количество CAN-кадров

Возвращаемое значение:

>=0 - успешное выполнение, количество прочитанных кадров

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;
- ECISTATE - канал не находится в состоянии CAN_RUNNING;
- ECIMFAULT - системная ошибка памяти (не удается скопировать параметры или результаты);

3.13 _s16 CiErrsGetClear(_u8 chan, canerrs_t * errs)

Описание:

Возвращает значение программных счетчиков ошибок канала ввода-вывода и сбрасывает их в ноль. Счетчики ошибок возвращаются в структуре canerrs_t которая имеет следующий вид:

```
typedef struct {
  _u16 ewl;    // кол-во ошибок EWL
  _u16 boff;   // кол-во ошибок BOFF
  _u16 hwovr;  // кол-во ошибок HOVR
  _u16 swovr;  // кол-во ошибок SOVR
  _u16 wtout;  // кол-во ошибок WTOUT
} canerrs_t;
```

При открытии канала счетчики равны нулю и непрерывно увеличиваются на 1 при возникновении соответствующей ошибки.

Параметры:

- chan - номер канала
- errs - указатель на структуру canerrs_t, куда будут записаны данные.

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;
- ECIMFAULT - системная ошибка памяти (не удается скопировать параметры или результаты);

3.14 _s16 CiWaitEvent(canwait_t * cw, int cwcount, int tout)

Описание:

Блокирует работу потока выполнения до наступления заданного события или до наступления тайм-аута в одном из указанных каналов ввода-вывода. Каналы ввода-вывода и интересующие события задаются с помощью массива структур canwait_t, которая определена следующим образом:

```
typedef struct {
  _u8 chan;   // номер открытого канала
  _u8 wflags; // флаги интересующих нас событий
  _u8 rflags; // флаги наступивших событий (результат выполнения)
} canwait_t;
```

Возможные флаги событий:

- CI_WAIT_RC - количество **кадров** в приемной очереди стало больше или равно значению порога (см. CiRcQueTreshold [3.16]);
- CI_WAIT_ER – ошибка сети EWL, BOFF, HOVR, SOVR, или WTOUT (см. пункт [2.1] и CiErrsGetClear [3.13]);
- CI_WAIT_TR - количество **пустых слотов** в очереди на отправку стало больше или равно значению порога (см. примечание и CiTrQueTreshold [3.15]).

Функционал флага CI_WAIT_TR не доступен для устройств CAN-bus-USBnp с версией прошивки меньше 1.3 включительно.

Пример использования:

```
canwait_t cw[2];

/* открываем и инициализируем два канала */
...
cw[0].chan = chan1;
cw[0].wflags = CI_WAIT_RC | CI_WAIT_ER;
cw[1].chan = chan2;
cw[1].wflags = CI_WAIT_RC | CI_WAIT_ER;

CiStart(chan1);
CiStart(chan2);

while (1) {
  ret = CiWaitEvent(cw, 2, 1000); // timeout = 1000 миллисекунд
  if (ret > 0) {
    for (i=0; i<2; i++) {
      if (cw[i].rflags & CI_WAIT_RC) {
        // получен кадр в канале cw[i].chan
        // читаем кадры с помощью CiRead();
      }
      if (cw[i].rflags & CI_WAIT_ER) {
        // в канале cw[i].chan произошла ошибка
        // читаем ошибки с помощью CiErrsGetClear()
      }
    }
  } else if (ret < 0) {
     // ошибка CiWaitEvent()
  }  else { // ret == 0
    // timeout
  }
}
```

Параметры:

- cw – указатель на массив структур canwait_t пользователя;
- cwcount – количество элементов в массиве;
- tout – значение тайм-аута в миллисекундах, если значение равно -1 (минус единица), то тайм-аут принимается равным бесконечности;

Возвращаемое значение:

> 0 - успешное выполнение: произошло одно из указанных событий;

0 - тайм-аут;

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;
- ECIMFAULT - системная ошибка памяти (не удается скопировать параметры или результаты);
- ECIGEN – ошибка операционной системы;

Примечание:

Функционал флага CI_WAIT_TR имеет особенность при значении порога очереди на передачу равному размеру этой очереди (значение по умолчанию). В этом случае функция CiWaitEvent дожидается не только полного освобождения программной очереди на передачу, но и освобождения аппаратного буфера в CAN-контроллере. Эта особенность используется в реализации функции CiWrite (см. примечание к описанию этой функции [3.23]).

3.15 _s16 CiTrQueThreshold(_u8 chan, _s16 getset, _u16 * thres)

Описание:

Возвращает или устанавливает значение порога очереди на отправку. Порог измеряется в количестве **пустых слотов** в очереди на отправку (размер очереди минус количество кадров в очереди).

Функция не доступна для устройств CAN-bus-USBnp с версией прошивки меньше 1.3 включительно.

Требуемое состояние канала ввода-вывода: CAN_INIT

Параметры:

- chan - номер канала,
- getset - CI_CMD_SET установить значение, CI_CMD_GET прочесть,
- msec - указатель на переменную значения порога.

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов; канал не был открыт; запрошенное значение тайм-аута выходит за допустимые пределы;

3.16 _s16 CiRcQueThreshold(_u8 chan, _s16 getset, _u16 * thres)

Возвращает или устанавливает значение порога приемной очереди. Порог измеряется в количестве **кадров** в приемной очереди.

Требуемое состояние канала ввода-вывода: CAN_INIT

Параметры:

- chan - номер канала,
- getset - CI_CMD_SET установить значение, CI_CMD_GET прочесть,
- msec - указатель на переменную значения порога.

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов; канал не был открыт; запрошенное значение тайм-аута выходит за допустимые пределы;

3.17 _s16 CiRcQueResize(_u8 chan, _u16 size)

Изменяет размер приемной очереди.

Требуемое состояние канала ввода-вывода: CAN_INIT

Параметры:

- chan - номер канала,
- size - новый размер очереди.

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов; канал не был открыт; запрошенное значение тайм-аута выходит за допустимые пределы;

3.18 _s16 CiRcQueCancel(_u8 chan, _u16 * rcqcnt)

Описание:

Принудительно очищает (стирает) содержимое приемной очереди канала.

Параметры:

- chan - номер канала;
- rcqcnt – указатель на переменную пользователя, куда сохраняется количество стертых кадров.

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;

3.19 _s16 CiRcQueGetCnt(_u8 chan, _u16 * rcqcnt)

Описание:

Возвращает количество кадров находящихся в приемной очереди драйвера.

Параметры:

- chan - номер канала,
- rcqcnt – указатель на переменную пользователя, куда сохраняется количество кадров в очереди.

Возвращаемое значение:

0 - успешное выполнение,

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;
- ECIMFAULT - системная ошибка памяти (не удается скопировать параметры или результаты);

3.20 _s16 CiBoardGetSerial(_u8 brdnum, char *sbuf, _u16 bufsize)

Описание:

Возвращает серийный номер платы в виде строки.

Примечание: в настоящий момент серийные номера поддерживаются только для устройства CAN-bus-USBnp, для всех остальных типов устройств вызов возвращает ошибку ECINOSYS.

Параметры:

- brdnum — номер платы (аналогично binfo.brdnum вызова CiBoardInfo [3.29]);
- sbuf – указатель на массив пользователя, в который будет скопирован серийный номер;
- bufsize – длина массива sbuf;

Возвращаемое значение:

0 - успешное выполнение

< 0 – ошибка

возможные ошибки:

- ECIINVAL - номер платы, переданный в качестве параметра, выходит за пределы поддерживаемого числа плат;
- ECINODEV - номеру платы, переданному в качестве параметра, не соответствует ни одна из обнаруженных плат;
- ECINOSYS — данный тип платы не поддерживает серийный номер;
- ECIBUSY - вызов не может быть выполнен, поскольку все каналы заняты другими процессами;

3.21 _s16 CiHwReset(_u8 chan)

Описание:

Выполняет аппаратный сброс CAN-контроллера. При этом сбрасываются аппаратные счетчики ошибок CAN-контроллера, а также счетчики аппаратных и программных переполнений возвращаемые вызовом CiChipStat [3.27]; счетчики ошибок возвращаемые вызовом CiErrsGetClear [3.13] не изменяются.

Значения аппаратного фильтра и скорости передачи сохраняются. Состояние канала ввода-вывода CAN_INIT или CAN_RUNNING сохраняется.

Параметры:

- chan - номер канала

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;
- ECIMFAULT - системная ошибка памяти (не удается скопировать параметры или результаты);

3.22 _s16 CiSetLom(_u8 chan, _u8 mode)

Описание:

Переключает CAN-контроллер в режим Listen Only Mode и обратно. В режиме Listen Only Mode контроллер принимает все кадры, но не посылает кадры подтверждения и ошибок, то есть не участвует в работе сети. В этом режиме контроллер невидим для остальных узлов и никак не влияет на работу сети в целом. Режим Listen Only Mode используется для тестирующих и следящих за работой сети приложений. Посылка кадров в этом режиме невозможна.

Требуемое состояние канала ввода-вывода: CAN_INIT

Параметры:

- chan - номер канала;
- mode – возможные значения: CI_LOM_ON включает режим, CI_LOM_OFF выключает режим; CI_LOM_ON/CI_LOM_OFF определены в заголовочном файле chai.h.

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;
- ECIMFAULT - системная ошибка памяти (не удается скопировать параметры или результаты);
- ECISTATE - контроллер не находится в состоянии CAN_INIT;

3.23 _s16 CiWrite(_u8 chan, canmsg_t *mbuf, _s16 cnt)

Описание:

Устаревшая функция обратной совместимости - не рекомендуется использовать в новом коде.

Отправляет один кадр в сеть через регистры CAN-контроллера минуя очередь на отправку. Очередь на отправку должна быть пуста.

Примечание: **для корректной работы CiWrite порог очереди на отправку (см. CiTrQueTreshold [3.15]) должен быть равен размеру этой очереди (значение по умолчанию). Если значение порога другое, функция CiWrite возвращает ошибку ECIINVAL**. То есть, при использовании функции CiWrite нельзя изменять значение порога очереди на передачу со значения по умолчанию.

Требуемое состояние канала ввода-вывода: CAN_RUNNING

Параметры:

- chan - номер канала;
- mbuf - указатель на буфер в котором находится CAN-кадр;
- cnt – должно равняться единице начиная с версии библиотеки 2.0.0 (этот параметр оставлен для совместимости со старыми версиями).

Возвращаемое значение:

1 - успешное выполнение, (один отправленный кадр)

< 0 - ошибка

возможные ошибки:

- ECIBUSY – очередь на отправку не пуста,
- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт, **либо порог очереди на передачу не равен размеру очереди (см. примечание выше)**;
- ECISTATE - канал не находится в состоянии CAN_RUNNING, или попытка отправки кадра неподходящего формата (например, канал был открыт в режиме «только CIO_CAN11» и произошла попытка отправки кадра расширенного формата);
- ECIMFAULT - системная ошибка памяти (не удается скопировать параметры или результаты);
- ECIIO - ошибка ввода-вывода, регистры CAN-контроллера на отправку кадра не освобождаются за положенное время (timeout);

Примечание:

Алгоритм отправки кадра.

Если аппаратный буфер на отправку свободен, то кадр загружается в регистры CAN-контроллера и выставляется запрос на передачу кадра. Если буфер занят возвращается ошибка ECIIO.

Далее, если тайм-аут на передачу выставлен в ноль возвращается код успеха (1), иначе в течение времени тайм-аута проверяется успешное окончание отправки. Если отправка успешна, то возвращается код успеха (1), в противном случае попытки передачи прекращаются, запрос на передачу и кадр в аппаратном буфере контроллера сбрасываются, возвращается ошибка ECIIO.

Из приведенного алгоритма видно, что поведение драйвера отличается в двух случаях:

- тайм-аут **не равен** нулю: результат реальной отправки кадра в сеть драйвер возвращает сразу в качестве кода возврата CiWrite(); функция тратит время на ожидание конца операции;
- тайм-аут **равен** нулю: результат реальной отправки или не отправки кадра драйвер возвращает при следующем вызове CiWrite(); функция не тратит время на ожидание конца операции;

Значение тайм-аута на отправку может быть изменено при помощи функции CiWriteTout [3.24].

3.24 _s16 CiWriteTout(_u8 chan, _s16 getset, _u16 * msec)

Описание:

Устаревшая функция обратной совместимости - не рекомендуется использовать в новом коде.

Возвращает или устанавливает текущее значение тайм-аута в миллисекундах на отправку кадра в сеть с помощью вызова CiWrite [3.23].

Примечание: **Тайм-аут на передачу не влияет на работу функции CiTransmit [3.8]!**

Параметры:

- chan - номер канала,
- getset - CI_CMD_SET установить значение, CI_CMD_GET прочесть,
- msec - указатель на переменную значения тайм-аута.

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов; канал не был открыт; запрошенное значение тайм-аута выходит за допустимые пределы;

3.25 _u32 CiGetLibVer(void)

Описание:

Возвращает версию библиотеки CHAI как беззнаковое четырехбайтное целое число. Номер версии состоит из трех цифр: major (второй байт), minor (первый байт), subminor (нулевой байт). Для получения из 4-байтного числа major, minor и subminor чисел в заголовочном файле chai.h определены макросы: VERMAJ(ver), VERMIN(ver), VERSUB(ver). Эти макросы возвращают знаковое 4-байтное целое.

Параметры:

Нет.

Возвращаемое значение:

>0 - успех, версия библиотеки

0 - ошибка

3.26 _u32 CiGetDrvVer(void)

Описание:

Возвращает версию драйвера unican как беззнаковое четырехбайтное целое число. Номер версии состоит из трех цифр: major (второй байт), minor (первый байт), subminor (нулевой байт). Для получения из 4-байтного числа major, minor и subminor чисел в заголовочном файле chai.h определены макросы: VERMAJ(ver), VERMIN(ver), VERSUB(ver). Эти макросы возвращают знаковое 4-байтное целое.

Параметры:

Нет.

Возвращаемое значение:

> 0 - успешное выполнение, версия драйвера

0 - ошибка

3.27 _s16 CiChipStat(_u8 chan, chipstat_t *stat)

Описание:

Возвращает текущий статус CAN-контроллера. Структура chipstat_t имеет следующий вид:

```
typedef struct {
  int type;               /* тип контроллера */
  int brdnum;             /* номер платы на котором
                             находится контроллер */
  int irq;                /* линия прерывания
                          используемая контроллером */
  unsigned long baddr;    /* базовый адрес
                             контроллера */
  unsigned long hovr_cnt; /* счетчик аппаратных
                             переполнений */
  unsigned long sovr_cnt; /* счетчик программных
                             переполнений */
  char _pad[32];          /* padding для приведения
                      к типу текущего контроллера */
} chipstat_t;
```

В действительности используются другие структуры соответствующие данному типу контроллера, с возможностью вывода значений регистров этого контроллера. Например, для SJA1000 используется структура sja1000stat_t:

```
typedef struct {
  int type;
  int brdnum;
  int irq;
  unsigned long baddr;
  unsigned long hovr_cnt;
  unsigned long sovr_cnt;

  /* this registers are readable in all mode */
  unsigned char mode;
  unsigned char stat;
  unsigned char inten;
  unsigned char clkdiv;
  unsigned char ecc;
  unsigned char ewl;
  unsigned char rxec;
  unsigned char txec;
  unsigned char rxmc;
  /* this registers are readable in init(reset) mode
only */
  unsigned int acode;
  unsigned int amask;
  unsigned char btr0;
  unsigned char btr1;
  unsigned char outctl;
  char _pad[8];
} sja1000stat_t;
```

При использовании функции CiChipStat необходимо использовать структуру конкретного типа, и при вызове CiChipStat приводить ее к типу chipstat_t. Например, для SJA1000:

```
_u8 chan = 0;
sja1000stat_t stat;

... /* CiOpen() CiRead() ... etc. */
CiChipStat(chan, (chipstat_t *) &stat);
```

. . .

Следует иметь ввиду, что значения регистров зависят от состояния в котором находится контроллер (CAN_INIT или CAN_RUNNING). Например, для SJA1000 значения полей acode и amask отражают значение фильтра, только если контроллер находится в состоянии CAN_INIT.

Параметры:

- chan - номер канала
- stat - указатель на структуру chipstat_t, куда будут записаны данные.

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

возможные ошибки:

- ECIINVAL - номер канала, переданный в качестве параметра, выходит за пределы поддерживаемого числа каналов, либо канал не был открыт;
- ECIMFAULT - системная ошибка памяти (не удается скопировать параметры или результаты);

3.28 _s16 CiChipStatToStr(chipstat_t * status, chstat_desc_t * desc)

Описание:

Преобразует бинарное значение структуры данных статуса контроллера в текстовое описание статуса в формате ASCII. Структура данных chstat_desc_t имеет вид:

```
typedef struct {
  char name[CI_CHSTAT_STRNUM][CI_CHSTAT_MAXLEN];
  char val[CI_CHSTAT_STRNUM][CI_CHSTAT_MAXLEN];
} chstat_desc_t;
```

Функция заполняет эту структуру размещая в полях name название соответствующего поля структуры chipstat_t, а в полях val соответствующее значение. Пример использования:

```
chstat_desc_t desc;
chipstat_t st;
int i;
...
CiChipStat(chan, &st);
CiChipStatToStr(&st, &desc);
i = 0;
while (desc.name[i][0] != '\0') {
  printf("%-12s: %s\n", desc.name[i], desc.val[i]);
  i++;
}
```

Параметры:

- status – входной параметр, указатель на структуру статуса CAN контроллера заполненную функцией CiChipStat [3.27];
- desc – выходной параметр, указатель на структуру данных chstat_desc_t;

Возвращаемое значение:

0 - успешное выполнение

< 0 - ошибка

3.29 _s16 CiBoardInfo(canboard_t *binfo)

Описание:

Выдает информацию о плате. Структура canboard_t имеет следующий вид:

```
typedef struct {
  _u8 brdnum;        /* номер платы (от 0 до CI_BRD_NUMS-1)*/
  _u32 hwver;        /* номер версии железа (аналогичен по структуре
номеру версии библиотеки */
  _s16 chip[4];      /* массив номеров каналов (например chip[0]
содержит номер канала к которому
                      привязан первый чип платы, если номер <0 -
чип отсутствует)*/
  char name[64];     /* текстовая строка названия платы */
  char manufact[64]; /* текстовая строка - имя производителя */
} canboard_t;
```

Поле brdnum является входным параметром и определяет для какой платы (платы нумеруются начиная с нуля) выдать информацию. Пример:

```
canboard_t binfo;
...
binfo.brdnum = 1;
CiBoardInfo(&binfo);
...
```

Параметры:

- binfo - указатель на структуру canboard_t

Возвращаемое значение:

>=0 - успешное выполнение

< 0 – ошибка

возможные ошибки:

- ECIINVAL - номер платы, переданный в качестве параметра, выходит за пределы поддерживаемого числа плат;
- ECINODEV - номеру платы, переданному в качестве параметра, не соответствует ни одна из обнаруженных плат;
- ECIMFAULT - системная ошибка памяти (не удается скопировать параметры или результаты);
- ECIBUSY - вызов не может быть выполнен, поскольку все каналы заняты другими процессами;

Примечание: данная функция должна использоваться для определения доступных системе аппаратных CAN-интерфейсов и соответствующих им каналов ввода-вывода CAN. Например:

```
int print_binfo(void)
{
  _s16 i, j, ret;
  canboard_t binfo;
  int cnt = 0;

  for (i = 0; i < CI_BRD_NUMS; i++) {
    binfo.brdnum = (_u8) i;
    ret = CiBoardInfo(&binfo);
    if (ret < 0)
```

```
      continue;
    printf("%s (%s): \n", binfo.name, binfo.manufact);
    for (j = 0; j < 4; j++) {
      if (binfo.chip[j] >= 0) {
          printf("   channel %d\n", binfo.chip[j]);
          cnt++;
      }
    }
  }
  return cnt;
}
```

Приведенная выше функция print_binfo() распечатывает на экране все обнаруженные библиотекой CHAI аппаратные CAN-интерфейсы и каналы ввода-вывода им соответствующие. Количество обнаруженных каналов ввода-вывода возвращается в качестве результата работы функции.

3.30 void CiStrError(_s16 cierrno, char *buf, _s16 n)

Описание:

Возвращает текстовое описание ошибки CHAI.

Параметры:

- cierrno - номер ошибки CHAI (может быть как положительным так и отрицательным числом, модуль которого равен коду ошибки);
- buf - указатель на буфер в котором будет сохранено текстовое описание ошибки;
- n - длина буфера в байтах (символах).

Возвращаемое значение:

нет

3.31 void CiPerror(_s16 cierrno, const char *s)

Описание:

Печатает в стандартный поток вывода ошибок сообщение s, а затем текстовое описание ошибки cierrno.

Параметры:

- cierrno - номер ошибки CHAI (может быть отрицательным числом);
- s - сообщение.

Возвращаемое значение:

нет

4. ИЗМЕНЕНИЯ В ПРОГРАММНОМ ИНТЕРФЕЙСЕ CHAI 2.X.X

В версиях 2.x.x библиотеки CHAI внесены следующие основные изменения в программный интерфейс по сравнению с версиями 1.x.x:

- удален блокирующий режим работы с каналом ввода-ввода; флаг CIO_BLOCK вызова CiOpen оставлен для обратной совместимости, но полностью игнорируется;
- функция CiWrite отправляет только один кадр, параметр cnt вызова CiWrite() сохранен для совместимости, должен быть всегда равен 1;
- функция CiRead работает в режиме совместимом с неблокирующим режимом версии 1.x.x, то есть возвращает столько кадров сколько есть в очереди, но не больше запрошенного количества;
- значения MSG_HOVR, MSG_SOVR поля flags структуры данных кадра canmsg_t удалены; функции msg_ishovr и msg_issovr оставлены для совместимости, всегда возвращают FALSE;
- добавлена функция CiErrsGetClear для чтения ошибок CAN в режиме опроса без использования функций обратного вызова;
- добавлена функция CiWaitEvent позволяющая блокировать работу потока выполнения до наступления события CAN без использования режима занятого ожидания (busy wait);
- добавлена функция CiSetLom вместо CiSJA1000SetLom и CiSJA1000ClearLom, которые сохранены для обратной совместимости;
- добавлены функции: CiChipStatToStr, CiGetWriteTout;
- в версии 2.3.0 добавлена функция CiBoardGetSerial;
- в версии 2.5.0 добавлена функция CiSetCBex;
- в версии 2.6.0 добавлены функции: CiCB_lock и CiCB_unlock;
- в версии 2.10.0 добавлены функции CiTrStat, CiTransmit (недоступна для CAN-bus-USBnp с версией прошивки до 1.3 включительно), CiTrCancel (недоступна для CAN-bus-USBnp с версией прошивки до 1.3 включительно), CiTrQueThreshold (недоступна для CAN-bus-USBnp с версией прошивки до 1.3 включительно), CiRcQueThreshold, CiGetFirmwareVer, CiRcQueCancel (вместо CiRcQueEmpty), CiRcQueResize (вместо CiQueResize), CiRcQueGetCnt (вместо CiRcGetCnt), CiWriteTout (вместо CiSetWriteTout и CiGetWriteTout);
  механизм функций обратного вызова объявлен устаревшим и не рекомендуется для использования в новом коде (функции CiSetCB, CiSetCBex, CiCB_lock, CiCB_unlock);
- в версии 2.11.0 добавлена функция CiTransmitSeries (недоступна для CAN-bus-USBnp с версией прошивки до 1.4 включительно);
- в версии 2.12.0 удален механизм функций обратного вызова (функции CiSetCB, CiSetCBex, CiCB_lock, CiCB_unlock и связанные с ними структуры данных и определения);
  механизм отправки кадров через функции CiWrite, CiWriteTout объявлен устаревшим,

не рекомендуется для использования в новом коде, и может быть удален в следующих версиях.

5. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

5.1 ОС Linux

Для реализации функции CiWaitEvent используется системный вызов poll.

5.2 ОС Windows

Для реализации функции CiWaitEvent используется системный вызов WaitForMultipleObjects.

6. СПИСОК ЛИТЕРАТУРЫ.

[Лит. 1] Спецификация CAN-контроллера NXP (Philips) SJA 1000 [http://can.marathon.ru/files/SJA1000.pdf];

[Лит. 2] Определение параметров скорости передачи CAN-контроллера SJA 1000 - «Determination of Bit Timing Parameters for the CAN Controller SJA 1000 AN97046» [http://can.marathon.ru/files/can_timing.pdf];

[Лит. 3] Спецификация CAN 2.0 Bosch [http://can.marathon.ru/files/can2spec.pdf];