Оптоизолированный USB-CAN-адаптер 2.0

Руководство пользователя

ООО КАСКОД-ЭЛЕКТРО
2008
Санкт-Петербург

Содержание

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Описание USB-CAN-адаптера 2.0

USB-CAN-адаптер представляет собой оптоизолированный одноканальный CAN-адаптер, подключаемый к PC-компьютеру через USB-порт. USB-CAN-адаптер предназначен для создания цифровых распределенных систем реального времени, отладки протоколов управления, а также для мониторинга CAN-сети.

Благодаря технологии Plug & Play USB-CAN-адаптер можно подключать и отключать при включенном компьютере. Определение подключенного адаптера к компьютеру производится автоматически, что упрощает настройку системы.

На рис.1 представлен внешний вид USB-CAN-адаптера.

Рис. 1 Внешний вид USB-CAN-адаптера.

USB-CAN-адаптер производит прием сообщений (только Data Frames, Remote Frames не принимаются) по 32-м каналам сообщений с сохранением принятых сообщений либо в общий, либо в отдельный буфера пользователя. При приеме может производится дополнительная программная фильтрация сообщений с сохранением сообщений, прошедших через указанные фильтры.

Передача сообщений производится по 32-м каналам сообщений из буферов, указанных пользователем.

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Технические характеристики и комплект поставки

Технические характеристики:

- CAN-интерфейс 2.0A, 2.0B;
- ISO 11898;
- один оптоизолированный CAN-канал с производительностью до 1 Мбит/с;
- напряжение изоляции – 1000 В (постоянное);
- интерфейс с PC-компьютером – USB 2.0, режим Full Speed 12 Мбит/с;
- поддержка технологии Plug & Play;
- размер ОЗУ – 256 Кбайт;
- питание – от USB;
- потребляемая мощность не более 1.5 Вт;
- рабочий диапазон температур:
  - стандартный от 0 до +70 °C;
  - расширенный от –40 до +85 °C.

Комплект поставки:

стандарт:
- USB-CAN-адаптер;
- компакт-диск с пакетом программного обеспечения:
  - драйвер USB-CAN-адаптера для операционных систем Windows 98SE/ME/2000/XP;
  - статическая (lib, для Visual C++ 6.0 и Borland C++ Builder 6.0) и динамическая (dll) библиотеки функций для работы с драйвером;
  - примеры программ использования библиотеки функций на языке Visual C++ 6.0 и Borland C++ Builder 6.0;
  - документация.

по заказу:
- USB-кабель;
- CAN-кабель.

Краткое описание интерфейсов – CAN и USB

CAN-интерфейс

Сетевой интерфейс CAN (Controller Area Network) был разработан в 1987 г. (версия 1.0) фирмами BOSCH и INTEL для создания бортовых мультипроцессорных систем реального времени. Последняя спецификация интерфейса 2.0, разработанная фирмой BOSCH в 1992 г., является дополнением предыдущей версии. В международной организации по стандартизации зарегистрирован ISO 11898 (для высокоскоростных приложений) и ISO 11519-2 (для низкоскоростных приложений).

Принцип работы

CAN является высокоинтегрированным сетевым интерфейсом передачи данных со скоростью до 1 Мбит/с. Устройства в CAN-сети соединяются по 3-проводной шине (2 сигнальных и один общий, см рис.2).

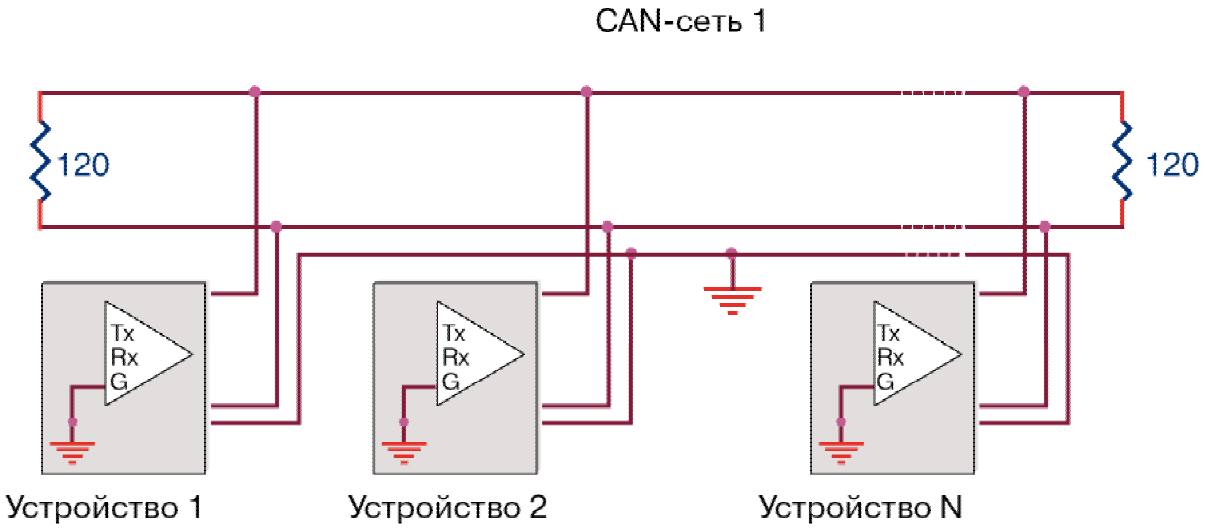

Рис. 2 Устройство CAN-сети.

Сообщения данных, передаваемые из любого узла по CAN-шине, могут содержать от 1 до 8 байт. Каждое сообщение помечено идентификатором, который в сети является уникальным (например: “Нагрев до 240”, “Отказ нагрева”, ”Бункер загружен”, и т.д.). При передаче другие узлы сети получают сообщение, и каждый из них проверяет идентификатор. Если сообщение имеет отношение к данному узлу, то оно обрабатывается, в противном случае – игнорируется. CAN-контроллер каждого из устройств может обрабатывать одновременно несколько идентификаторов (например, контроллеры SIEMENS и INTEL могут обрабатывать

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 ООО КАСКОД-ЭЛЕКТРО
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

до 15). Таким образом, в каждом из устройств можно легко организовать несколько “виртуальных” каналов обмена информацией с различными устройствами, включая каналы одновременного получения сообщений.

Идентификаторы

Идентификатор определяет тип и приоритет сообщения. Более низкому числовому значению идентификатора соответствует более высокое значение приоритета. Сообщение, имеющее более высокий приоритет, передается раньше сообщения, имеющего более низкий приоритет. После сообщения с высоким приоритетом передаётся сообщение с более низким приоритетом, если во время передачи не появится сообщение с более высоким приоритетом, затем передаётся сообщение с еще более низким приоритетом и т. д.

Физическая шина

Представляет собой витую пару (экранированную или неэкранированную) и общий провод. Плоская пара (телефонный тип кабеля) также работает хорошо, но более чувствительна к внешним источникам шума.

Высокая надёжность

Для обеспечения безотказной работы в тяжёлых условиях по стандарту ISO 11898 CAN-контроллер обеспечивает работу в сети в следующих случаях:

- любой из 3-х проводов в шине оборван;
- любой провод – замкнут на питание;
- любой провод – замкнут на общий провод.

При обрыве 2-х проводов часть функций основной системы может быть реализована в каждой из подсистем, созданных обрывом.

Сетевая гибкость и лёгкость расширения

Принятая в CAN-сети схема передачи сообщений обеспечивает большие возможности при создании, расширении и модернизации систем.

Новые устройства, предназначенные для приёма данных, могут добавляться к сети без изменения уже существующих программных средств, если их подключение не приводит к превышению нагрузочной способности и максимальной длины шины. При этом новые сетевые устройства способны обмениваться информацией между собой, не нарушая работоспособность старой системы, если в протоколе обмена были использованы новые идентификаторы.

В CAN-сети имеется возможность одновременной передачи сообщений сразу нескольким устройствам. Эта особенность позволяет передавать по ней синхросигналы.

Арбитраж CAN-шины

В любой системе некоторые из параметров изменяются быстрее, чем другие. Например, скорость ротора двигателя, как правило, изменяется за меньший промежуток времени, чем температура его корпуса или положение заслонки. Быстро изменяющиеся параметры должны передаваться более часто и, следовательно, требуют более высокого приоритета. Во время работы также возможно появление аварийных сообщений, которые должны передаваться с наивысшим приоритетом (например, превышение допустимой температуры, обрыв управляющего соленоида, короткое замыкание в цепи и т.д.).

Узлы CAN-сети являются равноправными при обмене, и каждый из них в любой момент времени может иметь сообщение, требующее безотлагательной передачи. Вероятность одновременного требования передачи от различных устройств не является чем-то необычайным, а случается регулярно. Для разрешения подобного конфликта требуется быстродействующий механизм распределения очередности передачи сообщений. Для этого в CAN-системе используется *неразрушающий поразрядный арбитраж*.

Приоритет CAN-сообщения определяется двоичным значением его идентификатора.

Числовое значение каждого идентификатора сообщения назначается в начальной фазе проектирования системы. Идентификатор с самым низким числовым значением идентификатора имеет самый высокий приоритет. Передача логического нуля по CAN-шине осуществляется токовой посылкой, а состояние логической единицы определяется по отсутствию тока. В процессе передачи каждый из источников сообщений, который имеет необходимость в передаче, начинает передавать свой идентификатор, одновременно проверяя его на линии. Если в процессе передачи обнаруживается несовпадение (т.е. “лишний” ноль), то передатчик, обнаруживший это несоответствие, прекращает передачу своего идентификатора и переключается на прием. Конфликта на шине при этом нет, так как значение бита с уровнем логической единицы фактически не передается, и в результате сообщение с наивысшим приоритетом проходит по шине так, как будто оно единственное. В следующем цикле шины будет передано сообщение с более низким приоритетом, и т.д. Таким образом достигается максимальная пропускная способность шины и минимальная задержка для “горячих” сообщений.

Обнаружение ошибок

CAN содержит 5-ступенчатый механизм обнаружения ошибок:

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

- циклический контроль по избыточности (CRC);
- контроль передаваемого поля битов;
- контроль сигнала “подтверждение приема”;
- текущий контроль логического уровня битов;
- контроль заполнения битов.

Циклический контроль по избыточности (CRC)

Каждое переданное сообщение содержит контрольный код (CRC), вычисленный передатчиком на основе содержания передаваемого сообщения. Приёмные узлы выполняют аналогичную операцию, помечают обнаруженные ошибки и устанавливают соответствующие флаги.

Текущий контроль логического уровня битов

Любой передатчик автоматически контролирует и сравнивает фактический логический уровень битов на шине с уровнем, который он передает. Если уровни не совпадают, помечается ошибка логического уровня битов.

Контроль передаваемого поля битов

В составе CAN-сообщения передаются предопределенные битовые комбинации, которые контролируются при приёме. Если приемник обнаруживает недопустимый бит в одной из этих комбинаций, то устанавливается флаг ошибки формата.

Контроль заполнения битов

CAN использует методику добавления заполняющего бита для дополнительного контроля передаваемых сообщений. После передачи пяти последовательных битов с одинаковым уровнем передатчик автоматически вводит в разрядный поток бит противоположного значения. Приемники сообщения автоматически удаляют такие биты перед обработкой сообщения. Если обнаруживается шестой бит одинаковой полярности, то помечается ошибка заполнения битов.

Контроль сигнала “подтверждение приема”

Каждое переданное сообщение подтверждается приемником, и если этого не произошло, тогда устанавливается флаг ошибки подтверждения приема.

Флаг ошибки

В случае, если обнаружена ошибка, то узел, обнаруживший ошибку, прерывает передачу посылкой флага ошибки. При этом передатчик автоматически повторяет передачу сообщения, что предотвращает все узлы от возникновения ошибок и гарантирует непротиворечивость данных в сети.

Формат CAN-сообщения

Стандартный CAN-протокол (версия 2.0A) поддерживает формат сообщения с 11-разрядными идентификаторами (стандартное сообщение). Расширенный CAN-протокол (версия 2.0B) поддерживает 11-битовый и 29-битовый форматы идентификаторов (расширенное сообщение). Большинство контроллеров версии 2.0A передают и принимают только сообщения стандартного формата, хотя часть из них могут только получать сообщения расширенного формата. Контроллеры версии 2.0B могут посылать и получать сообщения в обоих форматах.

Различия форматов

В версии 2.0B поле битов идентификатора состоит из двух частей:

- **первая часть** (основная часть идентификатора) имеет длину одиннадцать битов для совместимости с версией 2.0A;
- **вторая часть** имеет длину восемнадцать битов (расширение идентификатора), что дает общую длину идентификатора в двадцать девять бит.

Для различения форматов используются специальные биты в поле арбитража.

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

USB-интерфейс

USB (Universal Serial Bus – универсальная последовательная шина) является промышленным стандартом расширения архитектуры PC-компьютера. Версия 1.0 была опубликована в январе 1996 г. Последующие две версии 1.1 и 2.0, опубликованные соответственно в 1998 г и 2000 г, наиболее распространены.

Архитектура USB определяется следующими критериями:

- легко реализуемое расширение периферии PC-компьютера;
- скорость передачи до 12 Мбит/с (версия 1.1) или до 480 Мбит/с (версия 2.0);
- гибкость протокола смешенной передачи изохронных данных и асинхронных сообщений;
- возможность интеграции в PC-компьютерах любых размеров и конфигураций;
- легкое создание устройств-расширений PC-компьютеров.

С точки зрения пользователя важными параметрами USB являются следующие:

- простота подключения к PC-компьютеру, т.е. невозможно неправильно подключить устройство;
- не требуется выключать питание перед подключением из-за особенностей конструкции разъемов;
- скрытие подробностей электрического подключения от конечного пользователя;
- самоидентифицирующиеся периферийные устройства (Plug & Play);
- возможность динамического подключения периферийных устройств;
- малопотребляющие устройства (до 500 ма) могут получать питание прямо от USB-шины.

Структура USB

Физическое соединение устройств осуществляется по топологии многоярусной звезды. Центром каждой звезды является **хаб** (обеспечивает дополнительные точки подключения). Каждый кабельный сегмент соединяет две точки – хаб с другим хабом или **функцией** (представляет собой конечное периферийное устройство). В системе имеется, причем только один, **хост-контроллер**, расположенный в вершине пирамиды функций и хабов и управляющий работой всей системой. Хост-контроллер интегрируется с корневым хабом (Root Hub), обеспечивающим одну или несколько точек подключения – **портов**. Контроллер USB, входящий в состав чипсетов, обычно имеет встроенный двухпортовый корневой хаб.

Логически устройство, подключенное к любому порту хаба USB может рассматриваться как непосредственно подключенное к хост-контроллеру. Таким образом, точка подключения устройства не важна.

Хост-контроллер производит распределение пропускной способности шины между устройствами. USB-шина позволяет подключать, конфигурировать, использовать и отключать устройства во время работы хоста и самих устройств.

Функции представляют собой устройства, способные передавать или принимать данные или управляющую информацию по шине. Типично функции представляют собой отдельные периферийные устройства, подключенные к порту хаба USB кабелем. Каждая функция предоставляет конфигурационную информацию, описывающую возможности устройства и требования к ресурсам. Перед использованием функция должна быть сконфигурирована хостом – ей должна быть выделена полоса в канале и выбраны опции конфигурации.

Хаб представляет собой кабельный концентратор. Точки подключения называются портами хаба. Каждый хаб преобразует одну точку подключения в их множество. Архитектура допускает соединение нескольких хабов.

У каждого хаба имеется один **восходящий** порт (Upstream Port), предназначенный для подключения к хабу верхнего уровня и один или несколько **нисходящих** портов (Downstream Port), предназначенных для подключения функций или хабов нижнего уровня. Хаб распознает подключение и отключение устройств и управляет подачей питания на нисходящие сегменты.

Физический интерфейс

Для передачи данных по USB-шине используется дифференциальный способ. Информационные сигналы и питающее напряжение 5 В передаются по четырех проводному кабелю.

Версия USB 1.1 имеет два режима передачи данных:

- низкая скорость передачи данных (Low Speed) до 1.5 Мбит/с;
- полная скорость передачи данных (Full Speed) до 12 Мбит/с.

Версия 2.0 расширяет версию 1.1 режимом высокой скорости передачи данных (High Speed) до 480 Мбит/с.

Для режимов полной скорости используется экранированная витая пара с импедансом 90 Ом и длиной кабеля до 5 м, для режима низкой скорости – неэкранированная витая пара и длиной кабеля до 3 м. В одной системе можно использовать одновременно все режимы, причем переключение скоростей для устройств осуществляется прозрачно. Выбор режима скорости передачи данных

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

устройством, подключенным к определенному порту, определяется хабом по уровням информационных сигналов, смещаемых нагрузочными резисторами приемопередатчиков.

Разъемы

Стандарт USB определяет два типа разъемов – A и B.

Разъемы типа «A» (Upstream Connector) применяются для подключения к хабам. Вилки устанавливаются на кабелях, а гнезда – на нисходящих портах хабов.

Разъемы типа «B» (Downstream Connector) устанавливаются на устройствах. Ответная часть (вилка) устанавливается на кабеле, противоположный конец которого имеет вилку типа «A».

Разъемы типов «A» и «B» различаются механически (вилка типа «A» прямоугольная, а вилка «B» - квадратная), что исключает недопустимые петлевые соединения хабов. Четырехконтактные разъемы имеют ключи, исключающие неправильное соединение. Конструкция разъемов обеспечивает позднее соединение и раннее отсоединение сигнальных цепей по сравнению с питающими.

Питание

Питание устройств возможно от USB-шины через кабель (Bus-Powered Devices) или от собственного блока питания (Self-Powered Devices). Хост обеспечивает питанием непосредственно подключенные к нему устройства. Каждый хаб, в свою очередь, обеспечивает питание устройств, подключенных к его нисходящим портам.

Установка пакета программного обеспечения

В данной главе описан процесс установки пакета программного обеспечения для USB-CAN-адаптера на жестком диске из инсталляционного дистрибутива. Перед началом установки необходимо убедиться в том, что компьютер и управляемая им операционная система отвечают необходимым системным требованиям.

Системные требования

Минимальные системные требования, которым должен соответствовать компьютер и управляемая им операционная система:

- PC-совместимый компьютер с процессором Pentium-II и выше или с другим совместимым процессором;
- свободный USB-порт с поддержкой стандарта USB версии 2.0;
- операционная система Windows 98/ME/2000/XP;
- 32 Мбайт ОЗУ;
- 10 Мбайт свободного дискового пространства.

Установка драйвера USB-CAN-адаптера

Для установки драйвера необходимо произвести следующие действия:

- подключить адаптер;
- вставить компакт-диск с пакетом программного обеспечения в устройство считывания компакт-дисков;
- выбрать автоматический поиск дрйвера устройства или указать программе установки устройств местоположение inf-файла драйвера адаптера на компакт-диске – usbcan2.inf.

Для установки примеров программ, библиотек и документации для USB-CAN-адаптера запустите файл *setup.exe* из каталога *usbcan2* с инсталляционного диска и следуйте инструкциям программы установки.

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Структура каталогов после установки

Программа установки скопирует все необходимые файлы в указанный при установке каталог или в каталог, предлагаемый по умолчанию. В результате получится следующая структура (при установке в каталог по умолчанию):

C:\USBCAN2	Документация и файлы библиотек
C:\USBCAN2\INCLUDE	Заголовочные файлы библиотеки
C:\USBCAN2\SAMPLES	Примеры программ работы с библиотекой
C:\USBCAN2\SAMPLES\FILTER	Пример приема сообщений с использованием фильтров
C:\USBCAN2\SAMPLES\RECEIVEVC	Пример приема сообщений без использования фильтров (для Visual C++ 6.0)
C:\USBCAN2\SAMPLES\RECEIVEBB6	Пример приема сообщений без использования фильтров (для Borland C++ Builder)
C:\USBCAN2\SAMPLES\SENDVC	Пример отправки сообщений (для Visual C++ 6.0)
C:\USBCAN2\SAMPLES\SENDBB6	Пример отправки сообщений (для Borland C++ Builder)
C:\USBCAN2\SAMPLES\MIRRORVC	Пример режима «зеркало» (для Visual C++ 6.0)
C:\USBCAN2\SAMPLES\MIRRORBB6	Пример режима «зеркало» (для Borland C++ Builder)
C:\USBCAN2\SAMPLES\CYCLEVC	Пример режима циклического приемного буфера (для Visual C++ 6.0)

Подключение USB-CAN-адаптера

Подключение к PC-компьютеру

Подключение USB-CAN-адаптера к PC-компьютеру осуществляется USB-кабелем. Если USB-CAN-адаптер подключен к PC-компьютеру правильно, то должен загореться красный индикатор на боковой стороне корпуса адаптера. На рис.3 представлена боковая сторона корпуса адаптера с USB-разъемом.

Рис. 3 USB-разъем типа «B» на боковой стороне корпуса адаптера.

Подключение к CAN-сети

Подключение USB-CAN-адаптера к CAN-сети осуществляется CAN-кабелем. На рис.4 представлена боковая сторона корпуса адаптера с CAN-разъемом и переключателем нагрузочного резистора.

Рис. 4 CAN-разъем и переключатель нагрузочного резистора на боковой стороне корпуса адаптера.

Переключатель 1, находящийся на боковой стороне корпуса адаптера, предназначен для включения нагрузочного резистора 120 Ом между сигналами BUS_L (нижнее активное значение шины) и BUS_H (верхнее активное значение шины). Положение переключателя вниз (ON) – резистор подключен, положение переключателя вверх – резистор отключен.

При подключении устройств к CAN-сети необходимо подключить нагрузочные резисторы на двух концевых устройствах.

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Контакты разъема для подключения к CAN-сети представлены в таблице 1.

Таблица 1. Описание контактов CAN-разъема.

Номер контакта	Сигнал
3	GND общий шины
2	BUS_L (нижнее активное значение шины)
7	BUS_H (верхнее активное значение шины)
1,4,5,6,8,9	Не используются

Количество устройств на CAN-шине, подключенной к адаптеру, не должно превышать 110 (это связано с электрическими параметрами используемого буфера CAN-шины).

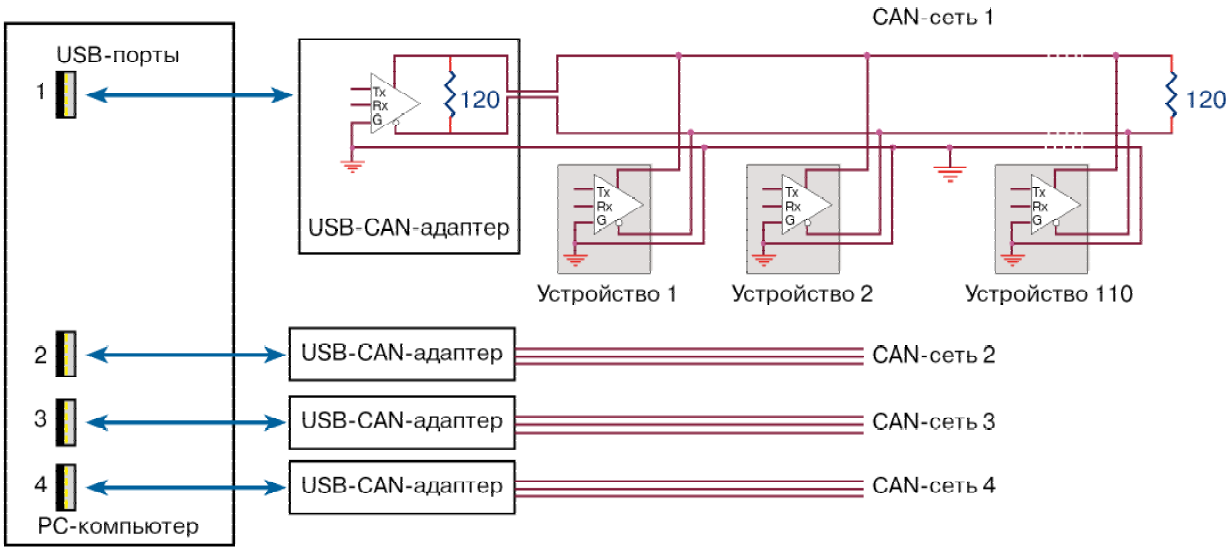

Рис. 5 Подключение USB-CAN-адаптера к PC-компьютеру и CAN-сети.

Работа с драйвером адаптера

Перед началом работы с драйвером адаптера необходимо внимательно изучить спецификацию CAN-сети ISO 11898.

Для работы с драйвером адаптера используется библиотека функций. Подробное описание функций библиотеки, а также способы подключения библиотеки к программе пользователя можно найти в главе «Описание функций библиотеки» данного руководства.

Нумерация адаптеров

Драйвер адаптера поддерживает работу до 10 подключенных адаптеров к USB-шине одного компьютера. При работе со всеми функциями библиотеки требуется указывать номер адаптера (далее номер CAN-канала) в качестве первого параметра.

Нумерация адаптеров (CAN-каналов) начинается с 0 и производится следующим образом:

- каждому вновь подключенному адаптеру присваивается наименьший свободный номер;
- после отключения адаптера присвоенный ему номер освобождается.

Присвоенные номера адаптеров действительны до следующей загрузки операционной системы. При использовании нескольких адаптеров рекомендуется подключать адаптеры после загрузки операционной системы в необходимом порядке.

Открытие и закрытие CAN-канала

В начале работы с каждым CAN-каналом необходимо вызвать функцию открытия UsbCan_Open (UINT Channel, PUSBCAN_INFO DrvInfo). Эта функция производит начальную инициализацию подключенного адаптера перед работой. В качестве входных параметров в функцию передаются номер CAN-канала (от 0 до 9) и адрес структуры USBCAN_INFO. После успешного выполнения функции поля структуры будут заполнены следующими значениями:

- версия драйвера адаптера;
- идентификатор производителя адаптера;
- идентификатор адаптера;
- версия адаптера;
- размер ОЗУ адаптера в байтах.

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Значения всех полей могут быть использованы по усмотрению пользователя. Значение последнего следует использовать в качестве максимального размера всех пользовательских буферов для приема и передачи. Значение этого поля рекомендуется сохранить в глобальной переменной пользовательской программы для дальнейшего определения суммарного размера выделяемой памяти под приемные и передающие буфера. В случае, если адаптер не подключен к PC-компьютеру или CAN-канал уже открыт, а также в других ошибочных ситуациях функция вернет соответствующий код системной ошибки. После завершения работы с CAN-каналом необходимо вызвать функцию закрытия UsbCan_Close (UINT Channel). Ниже представлен фрагмент программы на языке Visual C++ 6.0, иллюстрирующий использование функций открытия и закрытия CAN-канала.

```
#include "stdafx.h"
#include <stdlib.h>
#include "..\..\include\usbcan2.h"

// Указываем использование динамически подключаемой библиотеки
#define __USBCAN2_SHARED_LIBRARY__
#include "..\..\include\usbcanlib2.h"

int     main(int argc, char* argv[])
{
        USBCAN_INFO  info;
        UINT result;
        UINT channel = 0; // Номер канала

        // Открываем канал 0
        result = UsbCan_Open(channel, &info);
        if ( result != ERROR_SUCCESS )  {
            printf("Error to open\n");
            return 1;
        }

        //
        // Выводим на экран информацию об адаптере и драйвере
        //
        printf("Ram size       = %X\n", info.BoardRamSize);
        printf("Driver version = %X\n", info.DriverVersion);
        printf("Vendor ID      = %X\n", info.VendorID);
        printf("Product ID     = %X\n", info.ProductID);
        printf("Device version = %X\n", info.DeviceVersion);

        //...

        //
        // Закрываем канал
        //
        UsbCan_Close(channel);

        return 0;
}
```

Инициализация CAN-контроллера

После успешного открытия CAN-канала и перед дальнейшей работой необходимо инициализировать CAN-контроллер. При вызове функции инициализации ей передаются аргументы: значения глобальных масок (стандартной и расширенной), и значения регистров скорости обмена.

Значения глобальных масок используются в том случае, когда индивидуальное значение маски для канала сообщения равно минус 1, т.е. не указано (более подробно см. Разделе Функция запуска приема сообщений).

При указании значений регистров скорости обмена необходимо указывать непосредственное значение регистров:

Bit time 0:

- биты 0–5 – поле BRP (Baud Rate Prescaler). Значение делителя скорости. Значения от 1 до 63;
- биты 6–7 – поле SJW (Synchronization Jump Width). Значения от 1 до 3.

Bit time 1:

- биты 0–3 – поле Tseg1 (Time segment before the sample point). Значения от 1 до 15;
- биты 4–6 – поле Tseg2 (Time segment after the sample point). Значения от 1 до 3.

Для определения значений регистров можно воспользоваться файлом canbt.mcd. Это файл в формате Matchcad 7 с примером вычисления скорости обмена.

При максимальной скорости обмена равной 1 Мбит/с, значения регистров скорости обмена равны соответственно 0x41 и 0x34. Для изменения скорости обмена достаточно изменить значение битовго поля BRP.

Фрагмент программы на языке Visual C++ 6.0, иллюстрирующий использование функции инициализации CAN-контроллера.

```
#include "stdafx.h"
#include <stdlib.h>
#include "..\..\include\usbcan2.h"

// Указываем использование динамически подключаемой библиотеки
#define __USBCAN2_SHARED_LIBRARY__
#include "..\..\include\usbcanlib2.h"

int     main(int argc, char* argv[])
{
```


Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

```
        // Открываем канал ...

        // Инициализируем CAN-контроллер
        // Значения регистров масок устанавливаем в 0
        // Значения нулевого и первого регистров скорости обмена – 1 Мбит/с
        result = UsbCan_Init(channel, 0, 0, 0x41, 0x34);
        if ( result != ERROR_SUCCESS )  {
            printf("Error to initialize CAN channel\n");
            return 1;
        }
        //...

        // Закрываем канал ...

        return 0;
}
```

Инициализация размеров буферов

После того, как был инициализирован CAN-контроллер, необходимо произвести инициализацию размеров буферов под принятые сообщения и сообщения на передачу. Суммарный размер всех буферов не должен превышать размер ОЗУ адаптера, величина которого возвращается после вызова функции UsbCan_Open.

Для принятых сообщений указывается значение суммы размеров всех используемых приемных буферов (общего и отдельных буферов).

Для сообщений на передачу указывается значение размера буфера каждого канала сообщения. Если канал сообщения не используется для передачи сообщений, то соответствующее ему значение размера буфера должно быть равно нулю.

Фрагмент программы на языке Visual C++ 6.0, иллюстрирующий использование функции инициализации размеров буферов.

```
#include "stdafx.h"
#include <stdlib.h>

#include "..\..\include\usbcan2.h"
// Указываем использование динамически подключаемой библиотеки
#define __USBCAN2_SHARED_LIBRARY__
#include "..\..\include\usbcanlib2.h"

int     main(int argc, char* argv[])
{
        UINT result;
        UINT TxBufSize = 10*USBCAN_MSG_SIZE, RxBufSize = 10*USBCAN_MSG_SIZE;
        UINT Channel = 0;

        // ...
```

```
        // Инициализируем размеры буферов
        USBCAN_BUFFERS_SIZE  BuffersSize;
        memset(&BuffersSize, 0, sizeof(USBCAN_BUFFERS_SIZE));
        BuffersSize.SendBufSize[TxMsgObj-1] = TxBufSize;
        BuffersSize.ReceiveBufSize = RxBufSize;
        result = UsbCan_InitBuffersSize(Channel, &BuffersSize);
        if ( result != ERROR_SUCCESS )  {
            printf("Invalid parameters. Error=%d\n", result);
            return 1;
        }

        //...
        return 0;
}
```

Прием и передача сообщений

После того, как была произведена инициализация размеров буферов, т.е. распределение памяти под буфера, можно выполнять прием и передачу сообщений.

Прием сообщений может осуществляться как в общий буфер, так и в отдельные буфера указанных каналов сообщений.

Последовательность вызовов функций для приема сообщений

1. Указание адреса и размера общего буфера для принимаемых сообщений.

Если используется общий буфер для принятых сообщений, то перед запуском приема сообщений необходимо сначала указать адрес и размер буфера для принимаемых сообщений при помощи вызова функции UsbCan_SetReceiveBuf. Если требуется принятые сообщения по указанному каналу сообщения сохранять в отдельный буфер, то адрес буфера указывается при вызове функции запуска приема сообщений.

2. Запуск часов реального времени.

Если необходимо, чтобы с каждым принятым сообщением сохранялась временная метка нужно вызвать функцию UsbCan_StartRTC. После этого каждое принятое сообщение будет хранить временную метку, равную интервалу между принятыми сообщениями. Шаг интервала временной метки равен 3.2 мкс.

3. Запуск приема сообщений.

Для запуска приема сообщений необходимо вызвать функцию UsbCan_StartReceive. Параметрами этой функции являются номер CAN-канала, номер канала сообщения, значение арбитража, индивидуальная маска, число сообщений, которое необходимое принять, флаги, фильтр для приема сообщений

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

(опционально), адрес и размер буфера для сохранения сообщений (если необходимо принятые сообщения, по указанному каналу сообщения, сохранять в отдельный буфер).

4. Ожидание завершения приема сообщений.

После запуска приема сообщений необходимо в цикле вызывать функцию ожидания завершения приема сообщений UsbCan_Wait.

5. Останов приема сообщений.

После завершения приема сообщений или для прерывания приема сообщений необходимо вызвать функцию UsbCan_StopReceive.

6. Подготовка под повторный прием сообщений.

Если после успешного приема сообщений необходимо повторить прием в этот же буфер, то необходимо вызвать функцию UsbCan_ResetReceiveBuf, которая произведет подготовку для повторного приема.

Фрагмент программы на языке Visual C++ 6.0, иллюстрирующий прием сообщений.

```
#include "stdafx.h"
#include <stdlib.h>

#include "..\..\include\usbcan2.h"

// Указываем использование динамически подключаемой библиотеки
#define __USBCAN2_SHARED_LIBRARY__

#include "..\..\include\usbcanlib2.h"

int     main(int argc, char* argv[])
{

        // Открываем канал ...
        // Инициализируем CAN-контроллер ...
        // Инициализируем размеры буферов ...

        //
        // указваем адрес и размер буфера для приема
        //
        result = UsbCan_SetReceiveBuf(Channel, Buf, BufSize);
        if ( result != ERROR_SUCCESS )  {
            printf("Invalid parameters. Error=%d\n", result);
            free(Buf);
            return 1;
        }
```

```
        //
        // Повтор пприем сообщений в буфер 3 раза
        //
        for (int k=0; k<3; k++)  {

             printf("Step=%d\tStart receiving messages\n", k);


             // Запускаем часы реального времени
             printf("Start RTC\n");
             result = UsbCan_StartRTC(Channel);
             if ( result != ERROR_SUCCESS )  {
                 printf("Error start RTC\n");
                 return 1;
             }
             //
             // Запускаем прием
             //
             result = UsbCan_StartReceive(Channel, MSGOBJ_1, MSGNUM, 0x20,
                                    -1, USBCAN_ID_STD, NULL, NULL, NULL);
             if ( result != ERROR_SUCCESS )  {
                 printf("Error start receiving\n");
                 return 1;
             }

             //
             // Ожидаем завершения приема в течении 30 секунд
             //
             BOOL Receive = FALSE;
             for (i=0; i<=300; i++)  {
                 if ( UsbCan_Wait(Channel, 0, 100, USBCAN_DIR_RECEIVE) ==
                                    ERROR_SUCCESS )  {
                    Receive=TRUE;
                    break;
                 }
             }

             if ( Receive )
                 printf("OK\n");
             else
                 printf("Time-out. Messages not received\n");

             // Останов приема
             UsbCan_StopReceive(Channel, MsgObj);

             // Подготовка под повторный прием
             UsbCan_ResetReceiveBuf(Channel, MsgObj);

        }

        // Закрываем канал ...

        return 0;
}
```


Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Последовательность вызовов функций для передачи сообщений

1. Указание адреса и размера общего буфера с сообщениями на передачу.

Для каждого канала сообщения, используемого для передачи, необходимо создать буфер с сообщениями и указать его при помощи вызова функции UsbCan_SetSendBuf.

2. Запуск передачи сообщений.

Для запуска передачи сообщений необходимо вызвать функцию UsbCan_StartSend. Параметрами этой функции являются номер CAN-канала и номер канала сообщения.

3. Ожидание завершения передачи сообщений.

После запуска передачи сообщений необходимо в цикле вызывать функцию ожидания завершения передачи сообщений UsbCan_Wait.

4. Останов передачи сообщений.

После завершения передачи сообщений или для прерывания передачи сообщений необходимо вызвать функцию UsbCan_StopSend.

5. Подготовка под повторную передачу сообщений.

Если после успешной передачи сообщений необходимо повторить передачу из того же буфера, то необходимо вызвать функцию UsbCan_ResetSendBuf, которая произведет подготовку для повторной передачи.

Фрагмент программы на языке Visual C++ 6.0, иллюстрирующий передачу сообщений.

```
#include "stdafx.h"
#include <stdlib.h>

#include "..\..\include\usbcan2.h"

// Указываем использование динамически подключаемой библиотеки
#define __USBCAN2_SHARED_LIBRARY__
#include "..\..\include\usbcanlib2.h"

int     main(int argc, char* argv[])
{

        // Открываем канал ...
        // Инициализируем CAN-контроллер ...
        // Инициализируем размеры буферов ...
```

```
        //
        // Указваем адрес и размер буфера для передачи
        //
        result = UsbCan_SetSendBuf(Channel, MsgObj, Buf, BufSize);
        if ( result != ERROR_SUCCESS )  {
            printf("Invalid parameters. Error=%d\n", result);
            free(Buf);
            return 1;
        }

        //
        // Повтор передачи сообщений из буфера 3 раза
        //
        for (int k=0; k<3; k++)  {

             printf("Step=%d\tStart sending messages\n", k);

             //
             // Запускаем передачу
             //
             result = UsbCan_StartSend(Channel, MsgObj);
             if ( result != ERROR_SUCCESS )  {
                 printf("Error start sending by msgobj=%d\n", MsgObj);
                 return 1;
             }

             //
             // Ожидаем завершения передачи в течении 30 секунд
             //
             BOOL Send = FALSE;
             for (i=0; i<=300; i++)  {
                 if ( UsbCan_Wait(Channel, MsgObj, 100, USBCAN_DIR_SEND)
                       == ERROR_SUCCESS )  {
                    Send=TRUE;
                    break;
                 }
             }

             if ( Send )
                 printf("OK\n");
             else
                 printf("Time-out. Messages not send\n");

             // Останов передачи
             UsbCan_StopSend(Channel, MsgObj);

             // Подготовка под повторную передачу
             UsbCan_ResetSendBuf(Channel, MsgObj);

        }

        // Закрываем канал ...

        return 0;
}
```


Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Прием сообщений в режиме циклического буфера

Для приема сообщений в режиме циклического буфера необходимо также, как и в нормальном режиме произвести инициализацию размера приемного буфера. Запуск на прием сообщений производится аналогичным образом с единственным отличием – в качестве входного параметра числа сообщений, которое необходимо принять, указывается значение –1. После этого в цикле производить вызов функции UsbCan_GetReceived(…), для получения числа принятых сообщений. Также в цикле выполнять вызов функции UsbCan_Wait(…) для определения момента переполнения буфера (признаком переполнения является успешное выполнение функции).

Фрагмент программы на языке Visual C++ 6.0, иллюстрирующий прием сообщений.

```
#include "stdafx.h"
#include <stdlib.h>
#include "..\..\include\usbcan2.h"

// Указываем использование динамически подключаемой библиотеки
#define __USBCAN2_SHARED_LIBRARY__
#include "..\..\include\usbcanlib2.h"

int     main(int argc, char* argv[])
{

        // Открываем канал ...
        // Инициализируем CAN-контроллер ...
        // Инициализируем размеры буферов ...

        //
        // указваем адрес и размер буфера для приема
        //
        result = UsbCan_SetReceiveBuf(Channel, Buf, BufSize);
        if ( result != ERROR_SUCCESS )  {
            printf("Invalid parameters. Error=%d\n", result);
            free(Buf);
            return 1;
        }
        //
        // Запускаем прием
        //
        result = UsbCan_StartReceive(Channel, MSGOBJ_1, -1, 0x20, 0x15,
                                     USBCAN_ID_STD, NULL, NULL, NULL);
        if ( result != ERROR_SUCCESS )  {
            printf("Error start receiving\n");
            return 1;
        }
```

```
        //
        // Ожидаем завершения приема в течении 30 секунд
        //
        while (1)  {
             if ( UsbCan_Wait(Channel, 0, 100, USBCAN_DIR_RECEIVE) ==
                              ERROR_SUCCESS )  {
                 break;
             }
             UINT i;
             UsbCan_GetReceived(Channel, 0, i);
             printf("received %d messages\n", i);
        }

        // Останов приема
        UsbCan_StopReceive(Channel, MsgObj);
        // Подготовка под повторный прием
        UsbCan_ResetReceiveBuf(Channel, MsgObj);

        // Закрываем канал ...
        return 0;
}
```


Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Описание функций библиотеки

Для работы с USB-CAN-адаптером используется библиотека функций драйвера. В комплект поставки включены примеры программ работы с драйвером с использованием библиотеки функций.

При использовании статической версии библиотеки (lib) необходимо выполнить следующие действия:
в среде **Visual C++ 6.0**:

- добавить строки в исходный текст программы на языке Visual C++ 6.0

#include “usbcan2.h”
#include “usbcanlib2.h”

- добавить имя файла статической библиотеки usbcan2.lib к списку библиотек линковщика.

в среде **Borland C++ Builder 6.0**:

- добавить строки в исходный текст программы на языке Borland C++ Builder 6.0

#include “usbcan2.h”
#include “usbcanlib2.h”

- добавить имя файла статической библиотеки usbcan2_bb6.lib к списку библиотек линковщика.

При использовании динамической версии библиотеки (dll) необходимо выполнить следующие действия:
в среде Visual C++ 6.0 и Borland C++ Builder 6.0:

- добавить строки в исходный текст программы на языке Visual C++ 6.0

#include “usbcan2.h”
#define __USBCAN2_SHARED_LIBRARY__
#include “usbcanlib2.h”

- добавить имя файла библиотеки импорта usbcani2.lib к списку библиотек линковщика.

Функция открытия CAN-канала

Производит открытие CAN-канала и начальную инициализацию устройства.

Формат функции:

- UINT UsbCan_Open (UINT Channel, PUSBCAN_INFO DrvInfo)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- DrvInfo – адрес структуры, в которую сохраняется информация об устройстве и драйвере.

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал уже открыт;
- ERROR_BAD_ARGUMENTS – неправильно указан номер CAN-канала;
- код системной ошибки, получаемый функцией GetLastError().

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Функция закрытия CAN-канала

Производит закрытие указанного CAN-канала.

Формат функции:

- UINT UsbCan_Close (UINT Channel)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9.

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_BAD_ARGUMENTS – неправильно указан номер CAN-канала;
- код системной ошибки, получаемый функцией GetLastError().

Функция инициализации CAN-контроллера

Производит начальную инициализацию CAN-контроллера с установкой глобальных масок и регистров скорости обмена. Все каналы сообщений устанавливаются в неактивное состояние. Инициализация CAN-контроллера должна быть произведена после открытия CAN-канала и перед вызовом всех остальных функций.

Формат функции:

- UINT UsbCan_Init (UINT Channel, UINT SMask, UINT EMask, UINT BitTim0, UINT BitTim1)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- SMask – значение регистра стандартной 11-битовой маски;
- EMask – значение регистра расширенной 29-битовой маски;
- BitTim0 – значение регистра скорости обмена Bit Timing 0;
- BitTim1 – значение регистра скорости обмена Bit Timing 1.

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_BAD_ARGUMENTS – неправильно указан номер CAN-канала;
- код системной ошибки, получаемый функцией GetLastError().

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Функция инициализации размеров приемного и передающих буферов

Производит инициализацию размеров приемного и/или передающих буферов сообщений. Суммарный размер всех указанных буферов не должен превышать размер ОЗУ USB-CAN-адаптера. Размер ОЗУ хранится в возвращаемой структуре функции открытия канала и равен 256 Кб.

Формат функции:

- UINT UsbCan_InitBuffersSize (UINT Channel, PUSBCAN_BUFFERS_SIZE Buffers)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- Buffers – адрес структуры размеров приемного и/или передающих буферов.

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_NOT_ENOUGH_MEMORY – суммарный размер всех буферов больше размера ОЗУ адаптера;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - адрес структуры размеров буферов равен NULL;
  - CAN-контроллер не был инициализирован;
  - размер одного из буферов не кратен размеру структуры сообщения.
- код системной ошибки, получаемый функцией GetLastError().

Функция установки приемного буфера сообщений

Указывает драйверу адаптера адрес буфера и его размер для сохранения принятых сообщений.

Формат функции:

- UINT UsbCan_SetReceiveBuf (UINT Channel, PUSBCAN_MSG MsgBuf, UINT MsgBufSize)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- MsgBuf – адрес буфера для принимаемых сообщений;
- MsgBufSize – размер буфера в байтах (должен быть кратен размеру структуры параметров сообщения).

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_NOT_ENOUGH_MEMORY – размер указанного буфера больше размера, заявленного для приемного буфера вызовом функции UsbCan_SetupBuffersSize;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - не была произведена установка размера приемного буфера;
  - адрес буфера для принимаемых сообщений равен NULL;
  - размер буфера равен 0 или не кратен размеру структуры параметров сообщения;
  - CAN-контроллер не был инициализирован;
- код системной ошибки, получаемый функцией GetLastError().

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Функция сброса приемного буфера сообщений

Производит подготовку приемного буфера для проведения повторного приема сообщений.

Формат функции:

- UINT UsbCan_ResetReceiveBuf (UINT Channel)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9.

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_NOT_ENOUGH_MEMORY – размер используемого буфера для приема больше размера, заявленного для приемного буфера вызовом функции UsbCan_SetupBuffersSize;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - не была произведена установка размера приемного буфера;
  - CAN-контроллер не был инициализирован;
- код системной ошибки, получаемый функцией GetLastError().

Функция установки передающего буфера сообщений

Указывает драйверу адаптера адрес буфера и его размер для передачи.

Формат функции:

- UINT UsbCan_SetSendBuf (UINT Channel, UINT MsgObj, PUSBCAN_MSG MsgBuf, UINT MsgBufSize)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- MsgObj – номер канала сообщения для передачи (от 1 до 32);
- MsgBuf – адрес буфера сообщений для передачи;
- MsgBufSize – размер буфера сообщений в байтах (должен быть кратен размеру структуры параметров сообщения).

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_NOT_ENOUGH_MEMORY – если размер указанного буфера больше размера, заявленного для передающего буфера вызовом функции UsbCan_SetupBuffersSize;
- ERROR_BUSY – занят указанный канал сообщения;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - не была произведена установка размера передающего буфера;
  - адрес буфера сообщений на передачу равен NULL;
  - размер буфера равен 0 или не кратен размеру структуры параметров сообщения;
  - недопустимый номер канала сообщения;
  - CAN-контроллер не был инициализирован;
- код системной ошибки, получаемый функцией GetLastError().

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Функция сброса передающего буфера сообщений

Производит подготовку передающего буфера для проведения повторной передачи сообщений.

Формат функции:

- UINT UsbCan_ResetSendBuf (UINT Channel, UINT MsgObj)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- MsgObj – номер канала сообщения для сброса (от 1 до 32).

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - не была произведена установка размера передающего буфера;
  - недопустимый номер канала сообщения;
  - CAN-контроллер не был инициализирован;
- код системной ошибки, получаемый функцией GetLastError().

Функция запуска часов реального времени адаптера

Производит запуск часов реального времени адаптера. Данная функция вызывается перед запуском адаптера на прием сообщений для сохранения временной метки момента принятия каждого сообщения. Временная метка принятого сообщения представляет собой число, размер которого равен 6 байтам. Значение шага часов реального времени равно 3.2 мкс. Добавляя значение временной метки каждого принятого сообщения к значению системного времени PC-компьютера, которое должно быть получено и сохранено перед вызовом функции запуска часов реального времени адаптера, можно получить время приема сообщения.

Формат функции:

- UINT UsbCan_StartRTC (UINT Channel)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9.

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - CAN-контроллер не был инициализирован;
- код системной ошибки, получаемый функцией GetLastError().

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Функция запуска приема сообщений

Производит запуск CAN-контроллера на прием сообщений по указанному каналу сообщения.

Формат функции:

- UINT UsbCan_StartReceive (UINT Channel, UINT MsgObj, UINT Msg, UINT Arbitr, UINT Mask, BOOL Flags, PUSBCAN_RECEIVE_FILTER Filter, PUSBCAN_MSG MsgBuf, UINT MsgBufSize)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- MsgObj – номер канала сообщения для приема (от 1 до 32);
- Msg – число сообщений для приема (-1 для режима циклического буфера);
- Arbitr – значение арбитража сообщения для приема;
- Mask – значение маски для приема сообщения (если указано значение –1, то будет использоваться значение глобальной маски);
- Flags – флаги канала сообщения:
  - USBCAN_ID_EXT – прием сообщений с расширенным 29-битовым идентификатором;
  - USBCAN_ID_STD – прием сообщений со стандартным 11-битовым идентификатором.
- Filter – адрес структуры параметров фильтров принимаемых сообщений или NULL, если фильтры не используются;
- MsgBuf – адрес буфера для принятых сообщений (если требуется принятые сообщения по указанному каналу сообщения сохранять отдельно) или NULL (если требуется принятые сообщения сохранять в общем буфере);
- MsgBufSize - размер буфера.

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_BUSY – указанный канал сообщения занят;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - не была произведена установка размера приемного буфера;
  - недопустимый номер канала сообщения;
  - CAN-контроллер не был инициализирован;
  - значение параметра фильтра «размер данных» больше 8;
  - одновременно установлены флаги:
    - USBCAN_FILTER_SAVE_FIRST_PASSED;

        - USBCAN_FILTER_SAVE_ONLY_PASSED;
    - не установлен ни один из флагов:
        - USBCAN_FILTER_SAVE_FIRST_PASSED;
        - USBCAN_FILTER_SAVE_ONLY_PASSED;
    - не установлен ни один из флагов:
        - USBCAN_FILTER_ID;
        - USBCAN_FILTER_DATA;
        - USBCAN_FILTER_DATA_SIZE;
    - одновременно установлены флаги:
        - USBCAN_FILTER_OR;
        - USBCAN_FILTER_AND;
    - установлены любые два флага из трех:
        - USBCAN_FILTER_ID;
        - USBCAN_FILTER_DATA;
        - USBCAN_FILTER_DATA_SIZE;

      а флаги USBCAN_FILTER_OR или USBCAN_FILTER_AND не установлены;
    - установлен один из флагов:
        - USBCAN_FILTER_ID;
        - USBCAN_FILTER_DATA;
        - USBCAN_FILTER_DATA_SIZE;

      и одновременно установлен один из флагов USBCAN_FILTER_OR или USBCAN_FILTER_AND;
- код системной ошибки, получаемый функцией GetLastError().

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Функция остановки приема сообщений

Производит остановку приема сообщений по указанному каналу.

Формат функции:

- UINT UsbCan_StopReceive (UINT Channel, UINT MsgObj)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- MsgObj – номер канала сообщения для приема (от 1 до 32);

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - канал сообщения не был запущен на прием;
  - недопустимый номер канала сообщения;
  - CAN-контроллер не был инициализирован;
- код системной ошибки, получаемый функцией GetLastError().

Функция получения числа принятых сообщений

Возвращает число сохраненных в приемном буфере сообщений.

Формат функции:

- UINT UsbCan_GetReceived (UINT Channel, UINT MsgObj, UINT& Received)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- MsgObj – номер канала сообщения (от 1 до 32), если требуется получить число принятых сообщений по определенному каналу сообщения, или 0, если требуется получить общее число принятых сообщений;
- Received – ссылка на переменную, в которую будет записано число сообщений принятых в приемный буфер.

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - неправильно указан номер канала сообщения;
  - канал сообщения не был запущен на прием;
  - недопустимый номер канала сообщения;
  - CAN-контроллер не был инициализирован;
- код системной ошибки, получаемый функцией GetLastError().

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Функция запуска передачи сообщений

Производит запуск на передачу сообщений из передающего буфера.

Формат функции:

- UINT UsbCan_StartSend (UINT Channel, UINT MsgObj)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- MsgObj – номер канала сообщения для передачи (от 1 до 32).

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_BUSY – указанный канал сообщения занят;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - не была произведена установка размера передающего буфера;
  - недопустимый номер канала сообщения;
  - CAN-контроллер не был инициализирован;
- код системной ошибки, получаемый функцией GetLastError().

Функция остановки передачи сообщений

Производит остановку передачи сообщений по указанному каналу.

Формат функции:

- UINT UsbCan_StopSend (UINT Channel, UINT MsgObj)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- MsgObj – номер канала сообщения для остановки передачи (от 1 до 32).

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - не была произведена установка размера передающего буфера;
  - недопустимый номер канала сообщения;
  - CAN-контроллер не был инициализирован;
- код системной ошибки, получаемый функцией GetLastError().

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Функция получения числа переданных сообщений

Возвращает число переданных сообщений по указанному каналу.

Формат функции:

- UINT UsbCan_GetSend (UINT Channel, UINT MsgObj, UINT& Send)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- MsgObj – номер канала сообщения для получения числа переданных сообщений (от 1 до 32);
- Send – ссылка на переменную, в которую будет записано число переданных сообщений из передающего буфера.

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - канал сообщения не был запущен на передачу;
  - недопустимый номер канала сообщения;
  - CAN-контроллер не был инициализирован;
- код системной ошибки, получаемый функцией GetLastError().

Функция ожидания завершения приема или передачи

Производит ожидание завершения выполнения операции приема или передачи сообщений. В режиме циклического буфера для принимаемых сообщений используется для определения момента переполнения.

Формат функции:

- UINT UsbCan_Wait (UINT Channel, UINT MsgObj, UINT Interval, UINT Dir)

Входные параметры:

- Channel – номер CAN-канала (адаптера) от 0 до 9;
- MsgObj – номер канала сообщения для остановки передачи (от 1 до 32) или остановки приема (0 – в случае завершения приема в общий буфер и от 1 до 32 – в случае завершения приема в отдельный буфер);
- Interval – интервал ожидания в мс;
- Dir – направление обмена для ожидания, т.е. ожидание приема или передачи.

Возвращаемые значения:

- ERROR_SUCCESS – успешное выполнение функции;
- ERROR_INVALID_HANDLE – CAN-канал не был открыт;
- ERROR_TIMEOUT – время ожидания завершилось, а операция не завершилась;
- ERROR_BAD_ARGUMENTS – в следующих случаях:
  - неправильно указан номер CAN-канала;
  - не была произведена установка размера приемного или передающего буфера;
  - недопустимый номер канала сообщения;
  - CAN-контроллер не был инициализирован;
- код системной ошибки, получаемый функцией GetLastError().

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3

Описание структур данных библиотеки

Для работы с библиотекой функций драйвера USB-CAN-адаптера используются следующие структуры данных.

Структура USBCAN_INFO

Структура используется при открытии драйвера и заполняется драйвером соответствующими значениями:

- UINT DriverVersion – версия драйвера адаптера;
- UINT VendorID – ID производителя адаптера;
- UINT ProductID – ID устройства;
- UINT DeviceVersion – версия адаптера;
- UINT BoardRamSize – размер ОЗУ адаптера.

Структура USBCAN_MSG

Структура используется для создания буферов для принятых сообщений и сообщений на передачу. Структура состоит из следующих полей:

- WORD MsgObj – номер канала сообщения (от 0 до 32);
- BYTE Arbitr[4] – 11-битное или 29-битное значение арбитража канала сообщения;
- WORD Flags – флаги канала сообщения. Используются следующие биты:
  - 0 бит – признак потерянного сообщения (соответствует биту MsgLst). Используется только для принятых сообщений;
  - 1 бит – не используется;
  - 2 бит – тип идентификатора: расширенный (=1) или стандартный (=0);
  - 3 бит – направление обмена: передача (=1) или прием (=0);
  - 4-7 биты – размер данных сообщения;
- BYTE Data[8] – данные сообщения;
- WORD Status – значение статуса после приема сообщения;
- BYTE Time[6] – значение временной метки для принятых сообщений. Представляет собой интервал между принятыми сообщениями. Значение шага интервала между принятыми сообщениями равно 3.2 мкс.

Структура USBCAN_BUFFERS_SIZE

Структура используется при инициализации размеров буферов. Инициализация размеров буферов необходима для проведения распределения памяти адаптера и должна производиться перед операциями приема или передачи.

- DWORD ReceiveBufSize – размер для приемного буфера;
- DWORD SendBufSize[USBCAN_MSGOBJ_MAX-1] – размер буфера на передачу для соответствующего канала сообщения (от 0 до 31).

Структура USBCAN_RECEIVE_FILTER

Структура используется для указания фильтра при приеме сообщений.

- BYTE Data[8] – значение фильтра данных;
- WORD DataSize – значение фильтра размера данных;
- DWORD Id – значение фильтра идентификатора;
- BYTE DataFlags – маска фильтра данных. Номер бита, установленный в 1, соответствует номеру сравниваемых байта фильтра данных и байта данных принятого сообщения;
- BYTE FilterFlags – флаги фильтров:
    - 0 бит – используется фильтр идентификатора;
    - 1 бит – используется фильтр данных;
    - 2 бит – используется фильтр размера данных;
    - 3 бит – применять операцию И к результатам прохождения указанных фильтров;
    - 4 бит – применять операцию ИЛИ к результатам прохождения указанных фильтров;
    - 5 бит – после первого успешного прохождения через фильтры производится сохранение всех принятых сообщений без прохождения фильтров;
    - 6 бит – производить сохранение только успешно прошедших фильтры сообщений.

Email : support@kaskod.ru
http://www.kaskod.ru
тел.: (812) 476-0795
тел.: (812) 466-5784
тел.: (812) 466-5420
факс: (812) 466-5401
196625 **ООО КАСКОД-ЭЛЕКТРО**
Санкт-Петербург, Павловск,
Фильтровское шоссе, 3