

CAN bus

полное руководство

Ошибки CAN | J1939 | OBD2 | UDS | CANopen | CAN FD | LIN | DBC | CAN

NMEA 2000

ИЗОБУС

CCP / XCP на CAN

Содержание

CANedge - 2 x регистратор данных шины CAN / LIN	2...
Объяснение шины CAN - простое введение	2...
Что такое CAN bus?	2...
4 главных преимущества CAN bus	9 9
Краткая история шины CAN	7...
Будущее шины CAN	7...
Что такое фрейм CAN?	
Протоколирование данных CAN - примеры использования	9...
Как протоколировать данные шины CAN	9...
Как декодировать необработанные данные CAN в "физические значения"	1 1
Какова связь между CAN, J1939, OBD2, CANopen ...?	13.
J1939 Объяснени - простое вступление	14.
Что такое J1939?	14.
История и будущее J1939	14.
4 ключевые характеристики J1939	15...
Разъем J1939 (9-контактный)	17.
PGN и SPN J1939	18.
Образцы данных для грузовиков J1939: исходные и физические значения	20.
Сообщения с запросами J1939	23...
Транспортный протокол J1939 (TP)	24...
Регистрация данных J1939 - примеры использования	26 26
6 практических советов по регистрации данных J1939	26 26
Объяснение OBD2 - простое введение	28 28
Что такое OBD2?	28 _
Разъем OBD2	28 _
Есть ли в моей машине OBD2?	29...
Связь между OBD2 и шиной CAN	29...
История и будущее OBD2	30...
Идентификаторы параметров OBD2 (PID)	32...
Как регистрировать данные OBD2?	33 33
Необработанные сведения о кадре OBD2	34...
Регистрация данных OBD2 - примеры использования	36 36
Объяснена UDS (унифицированные диагностические службы)	36 36
Что такое протокол UDS?	36 36
Структура сообщений UDS	37...
UDS против шины CAN: стандарты и модель OSI	42...
CAN ISO-TP - транспортный протокол (ISO 15765-2)	46...
UDS против OBD2 против WWH-OBD против OBDonUDS	47...
Вопросы и ответы: Как запросить / записать данные UDS	51...
Пример 1: Запись однокадровых данных UDS (скорость через WWH-OBD)	55...
Пример 2: запись и декодирование многоkadровых данных UDS (SoC)	56...
Пример 3: запись идентификационного номера транспортного средства	59...

Ведение журнала данных UDS - приложения	60...
Объяснение CANopen - Простое введение	61...
Что такое CANopen?	61...
Шесть основных концепций CANopen	63...
Основы коммуникации CANopen	63...
Словарь объектов CANopen	68
SDO - настройка сети CANopen	69...
PDO - управление сетью CANopen	70...
Ведение журнала данных CANopen - примеры использования	72...
МОЖНО ли объяснить - Простое вступление	73...
Почему МОЖЕТ FD?	73...
Что такое CAN FD?	73...
Как работает CAN FD?	74...
Накладные расходы и эффективность передачи данных CAN FD по сравнению с CAN	76...
Примеры: Приложения CAN FD	78...
Ведение журнала МОЖЕТ ПРИВЕСТИ к удалению данных - примеры использования	79...
CAN FD - outlook	80...
LIN Bus Объяснил - Простое вступление	80...
Что такое LIN bus?	80...
Приложения для шины LIN	81...
Как работает LIN bus?	83...
Шесть типов каркасов LIN	85...
Расширенные разделы LIN	866
Файл описания LIN (LDF) по сравнению с Файлами DBC	87...
Регистрация данных шины LIN - примеры использования	89...
Практические рекомендации по ведению журнала данных LIN	90...
МОЖЕТ ли DBC-файл быть объяснен - Простое вступление	91...
Что такое файл CAN DBC?	91...
Пример: Извлечение физического значения сигнала о скорости двигателя	92...
CAN DBC editor игровая площадка	96
J1939/Данные OBD2 и образцы DBC	96 96
Дополнительно: Метаинформация, атрибуты и мультиплексирование	96 96
Программные инструменты DBC (редактирование и обработка)	98
Объяснение ошибок шины CAN - Простое введение	99 99
Что такое ошибки шины CAN?	99...
Кадр ошибки шины CAN	100
Типы ошибок CAN	103
Состояния узла CAN и счетчики ошибок	106
Примеры: Генерация и протоколирование кадров ошибок	109
Ошибки шины LIN	112 113
Примеры использования для ведения журнала кадров ошибок CAN	
Вопросы и ответы	114
Объяснение NMEA 2000 - Простое вступление	114
Что такое NMEA 2000?	114
NMEA 2000 против NMEA 0183	115

Разъемы NMEA 2000 и топология сети	121
Быстрый пакет NMEA 2000	122
N2K PGN и поля данных	122
Регистрация морских данных NMEA 2000	127
Описание ISOBUS (ISO 11783) - простое введение	128
Что такое ISOBUS?	128
История ISOBUS и AEF	128
Модель и стандарты ISOBUS OSI	130
Функциональные возможности ISOBUS	132
ISOBUS против SAE J1939	134
Соединители ISOBUS	135
ISOBUS PGN и SPN [+ DBC]	136
Данные о лесозаготовительном тракторе / орудии	139
Примеры использования регистрации данных	141
CCP / XCP на CAN Explained - простое введение	141
Что такое CCP / XCP?	142
Типы сообщений CCP	144
Как записывать данные ECU через CCP	148
Декодирование данных сигнала CCP с ECU	154
Файлы описания A2L - ECU	157
Начальная авторизация и авторизация по ключу	159
XCP на CAN - основы	160
Использование CANedge для сбора данных CCP / XCP	164
Регистрация данных CCP / XCP - приложения	165

CANedge - 2 x регистратор данных шины CAN / LIN

Интересуетесь шиной CAN и связанными с ней протоколами? Тогда обязательно ознакомьтесь с нашей серией CANedge!

CANedge представляет собой 2-кратный регистратор данных CAN / LIN с промышленной SD-картой емкостью 8-32 ГБ. В серии дополнительно предусмотрены GPS / IMU, Wi-Fi и / или 3G / 4G - идеально подходит для OEM-диагностики, телематики автопарка, профилактического обслуживания и многого другого:



ПОДКЛЮЧИ И ИГРАЙ

Технические ХАРАКТЕРИСТИКИ

Регистрируй данные CAN
поставляется из коробки.
Автономный.
Автоматическое определение скорости
передачи данных . Энергосбережение. Кадры ошибок. MF4

COMPACT

Всего 8 x 5 x 2 см.
100 г. Прочный
алюминиевый корпус
корпус. 5+ Светодиоды. Мощность
5 В
выход (CH2)

Wi-Fi / LTE Передача

данных через Wi-Fi
или 3G / 4G на ваш
сервер. E2E
Безопасность. OTA
Обновления

GPS + 3D IMU

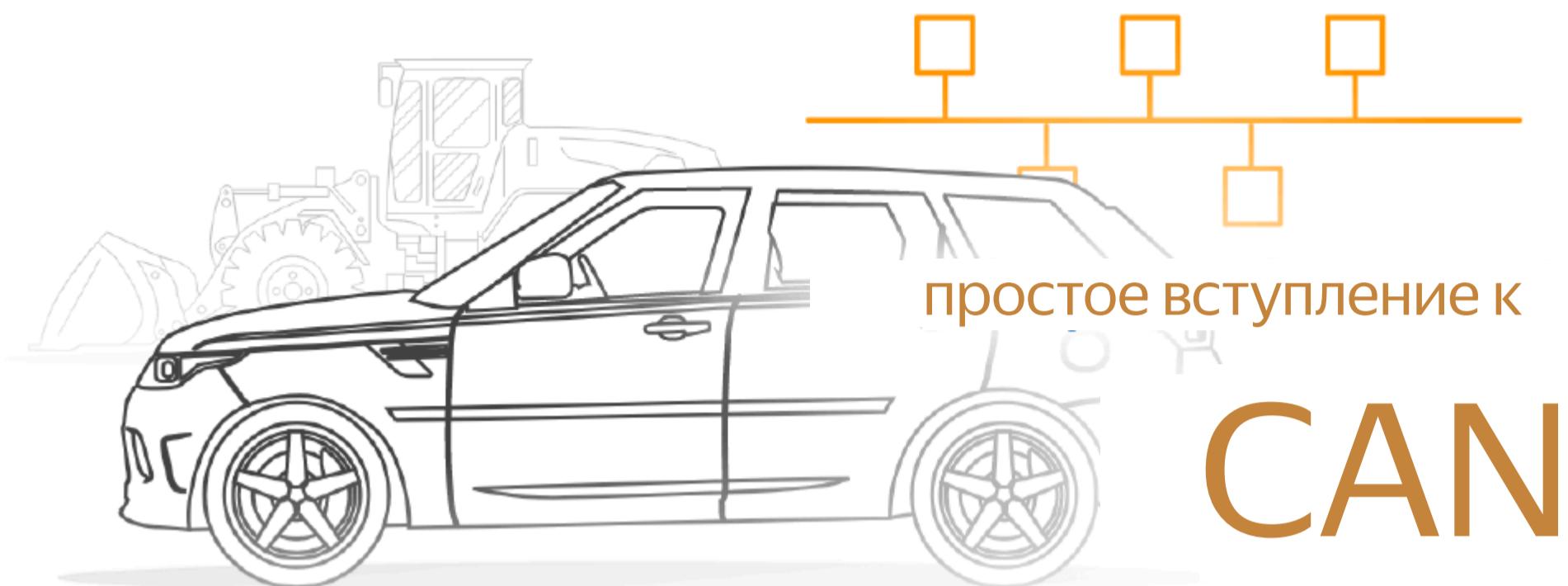
Встроенный GPS/IMU.
3-кратная точность
благодаря слиянию датчиков.
Положение, скорость,
расстояние и многое другое

СОВМЕСТИМОСТЬ

Бесплатный открытый исходный
код программное
обеспечение / API. От MF4 до
ASC / CSV. Декодирование DBC.
Информационные панели

[Посмотрите 5-минутное вступительное видео CANedge](#)





Объяснение шины CAN - простое вступление

Нужно простое, практическое введение в CAN bus? В этом руководстве мы объясняем, что такое локальная сеть (CAN bus) "для чайников" вкл. интерпретация сообщения, регистрация CAN - и ссылка на OBD2, J1939 и CANopen.

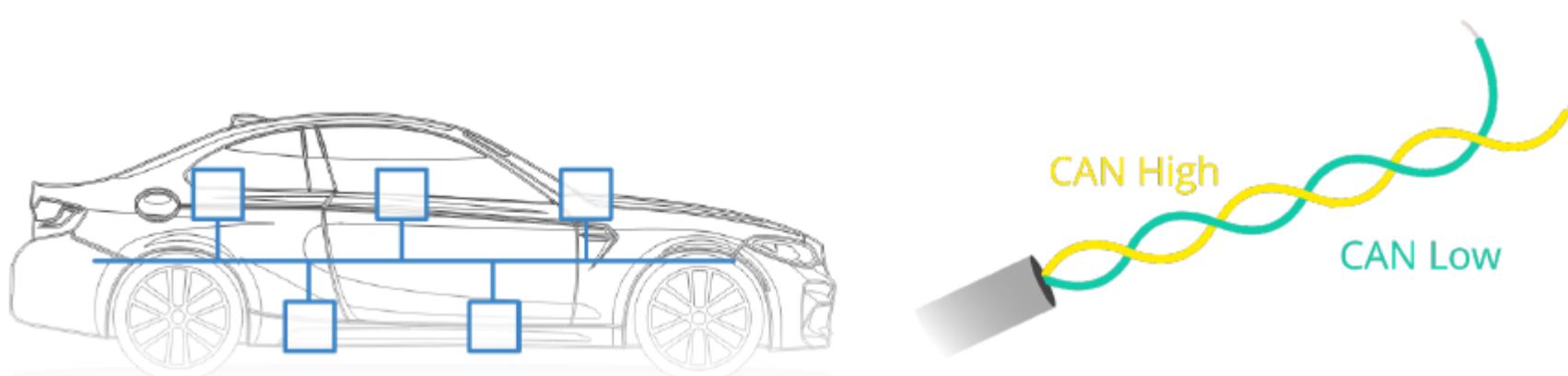
Читайте дальше, чтобы узнать, почему это руководство стало руководством № 1 по CAN bus.

Что такое CAN bus?

Ваш автомобиль подобен человеческому телу:

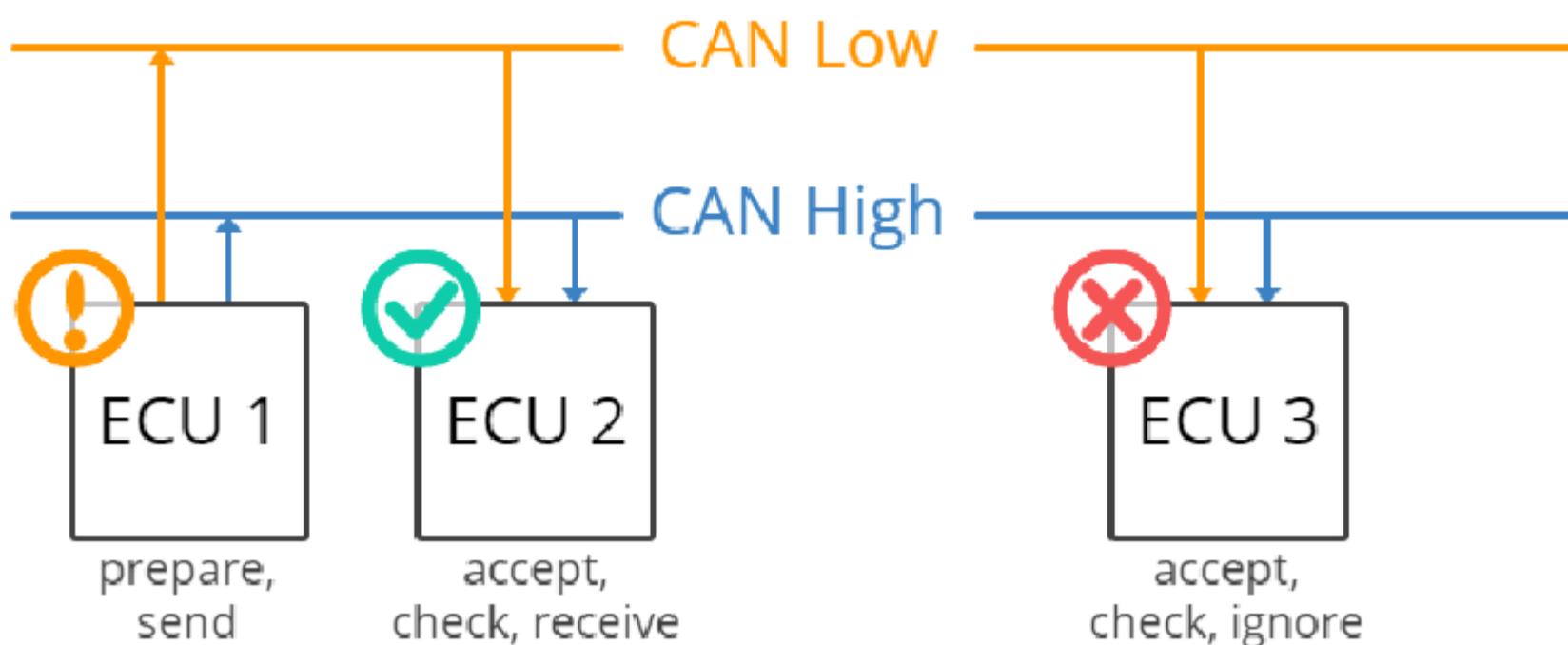
Локальная сеть управления (CAN bus) - это нервная система, обеспечивающая коммуникацию.

В свою очередь, "узлы" или "электронные блоки управления" (ECU) подобны частям корпуса, соединенным между собой через шину CAN. Информация ощущаемое одной частью может быть передано другой.



Итак, что такое ECU?

В автомобильной системе CAN bus ECU могут быть, например, блоком управления двигателем, подушками безопасности, аудиосистемой и т.д. Современный автомобиль может иметь до 70 блоков управления - и каждый из них может содержать информацию, которой необходимо поделиться с другими частями сети.



Вот где пригодится стандарт CAN.:

Система шины CAN позволяет каждому ЭБУ взаимодействовать со всеми другими ЭБУ - без сложной специальной проводки.

В частности, блок управления может подготавливать и передавать информацию (например, данные датчика) по шине CAN (состоящей из двух проводов, CAN low и CAN high). Переданные данные принимаются всеми другими блоками управления в сети CAN - и каждый блок управления может затем проверить данные и решить, принимать их или игнорировать.

Физический уровень шины CAN и канала передачи данных (OSI)

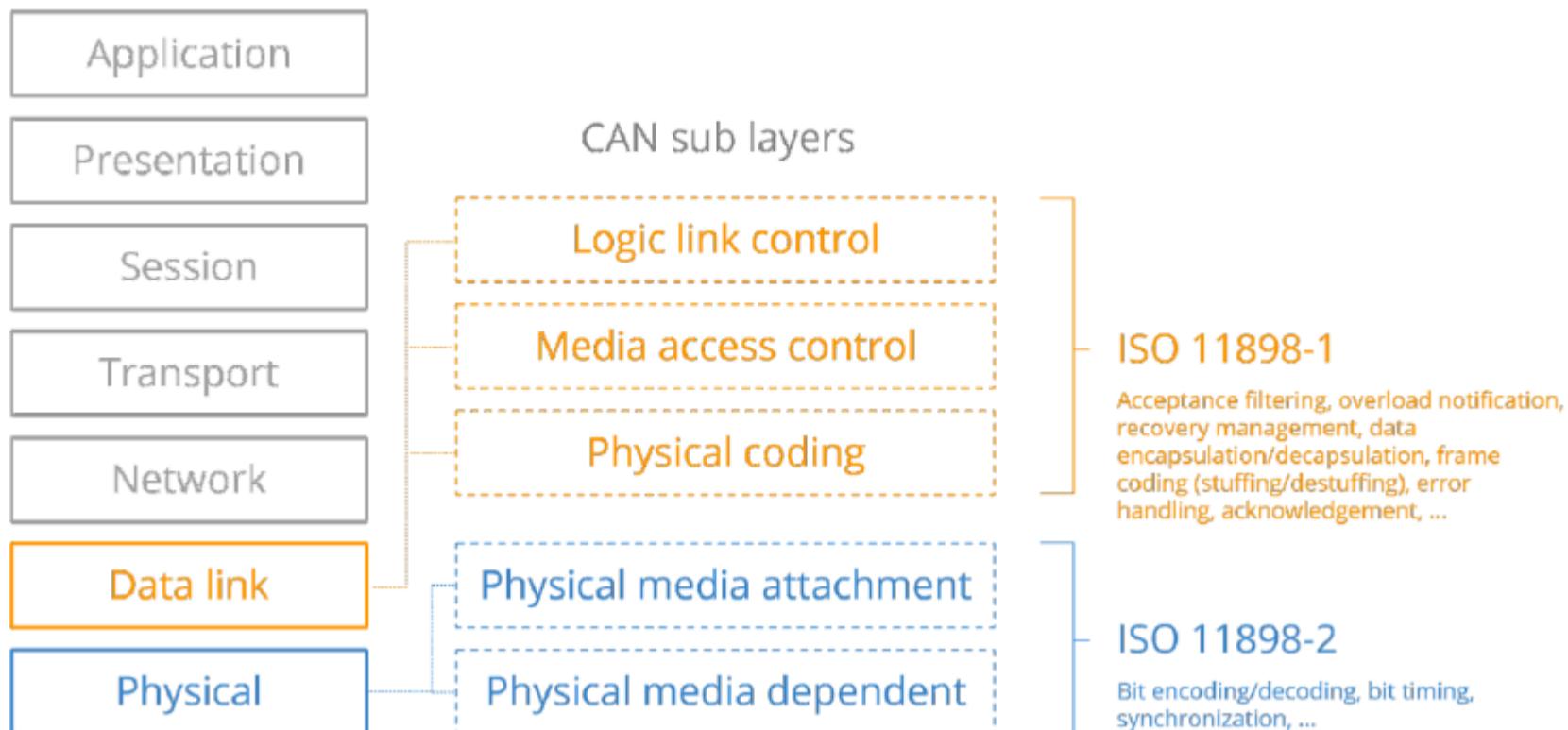
В более технических терминах локальная сеть контроллера описывается уровнем канала передачи данных и физическим уровнем. В случае высокоскоростной CAN, ISO 11898-1 описывает уровень канала передачи данных, в то время как ISO 11898-2 описывает физический уровень. Роль CAN часто представлен в 7-слойной модели OSI, как показано на рисунке.

Физический уровень шины CAN определяет такие вещи, как типы кабелей, уровни электрического сигнала, требования к узлу, полное сопротивление кабеля и т.д. Например, ISO 11898-2 предписывает ряд вещей, в том числе нижеприведенные:

Скорость передачи данных в бодах: Узлы CAN должны подключаться по двухпроводнойшине со скоростью передачи данных до 1 Мбит/с (классическая CAN) или 5 Мбит/с (CAN FD)

Длина кабеля: Максимальная длина кабеля CAN должна составлять от 500 метров (125 Кбит /с) до 40 метров (1 Мбит /с)
Подключение: Шина CAN должна быть подключена надлежащим образом с использованием резистора для подключения шины CAN на 120 Ом на каждый конец шины

7-слойная модель OSI



Высокоскоростная шина CAN, низкоскоростная шина CAN, шина LIN,...

В контексте автомобильных сетей вы часто будете сталкиваться с несколькими различными типами сетей. Ниже мы приводим очень краткое описание:

Высокоскоростная шина CAN: Основное внимание в этой статье уделяется высокоскоростной шине CAN (ISO 11898). Это, безусловно, самая популярный стандарт CAN для физического уровня, поддерживающий скорости передачи данных от 40 Кбит / с до 1 Мбит / с (классический CAN). Он обеспечивает простую прокладку кабелей и сегодня используется практически во всех автомобильных приложениях. Он также служит основой для нескольких протоколов более высокого уровня, таких как OBD2, J1939, NMEA 2000, CANopen и т.д. Второе поколение CAN называется CAN FD (CAN с гибкой скоростью передачи данных)

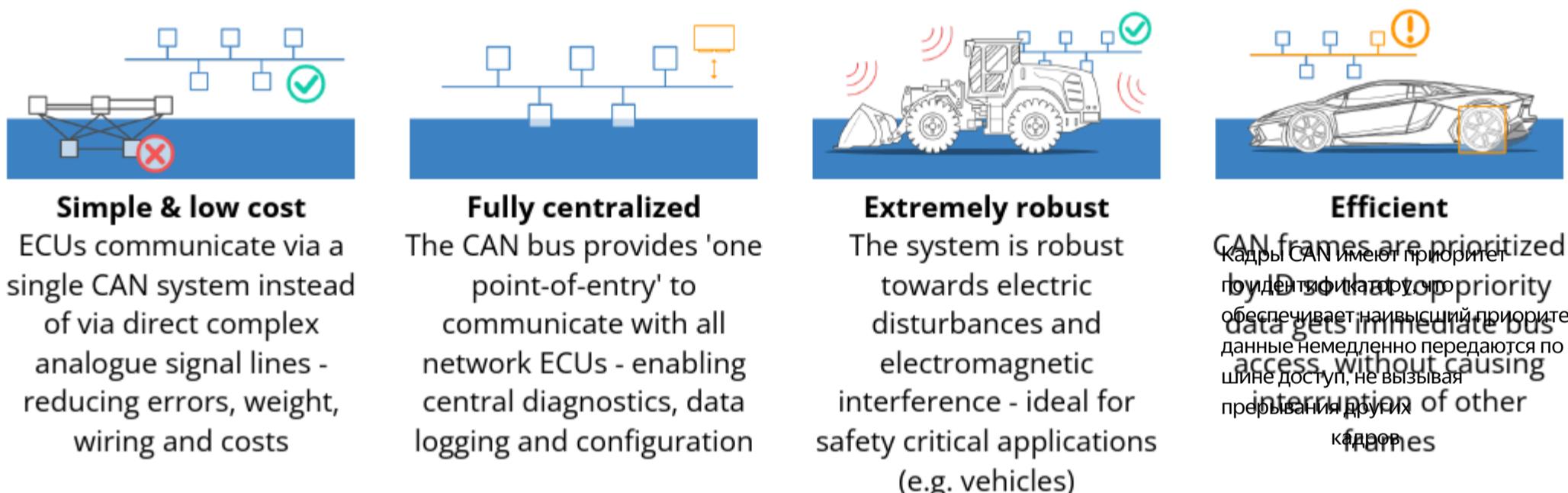
Низкоскоростная шина CAN: этот стандарт обеспечивает скорость передачи данных от 40 кбит / с до 125 кбит / с и позволяет шине CAN связь будет продолжаться даже при неисправности одного из двух проводов - следовательно, это также называется "отказоустойчивый CAN". В этой системе каждый узел CAN имеет собственное завершение CAN

Шина LIN: Шина LIN - это более дешевое дополнение к сетям CAN bus с меньшим количеством проводов и более дешевыми узлами. LIN кластеры шин обычно состоят из ведущего узла LIN, действующего как шлюз, и до 16 подчиненных узлов. Типичные варианты использования включают, например, некритичные функции автомобиля, такие как кондиционер, функциональность дверей и т.д. - подробнее смотрите в нашей статье о LIN bus [введение или регистратор данных шины LIN](#)

Автомобильный ethernet: Он все чаще используется в автомобильном секторе для поддержки высоких требований к пропускной способности ADAS (передовых систем помощи водителю), информационно-развлекательных систем, камер и т.д. Автомобильный ethernet обеспечивает гораздо более высокую скорость передачи данных по сравнению с CAN bus, но ему не хватает некоторых характеристик безопасности / производительности классических CAN и CAN FD. Скорее всего, в ближайшие годы появятся и те, и другие автомобильный ethernet, CAN FD и CAN XL будут использоваться в новых автомобилестроительных и промышленных разработках

4 главных преимущества CAN bus

Стандарт шины CAN используется практически во всех автомобилях и многих машинах благодаря нижеперечисленным ключевым преимуществам:

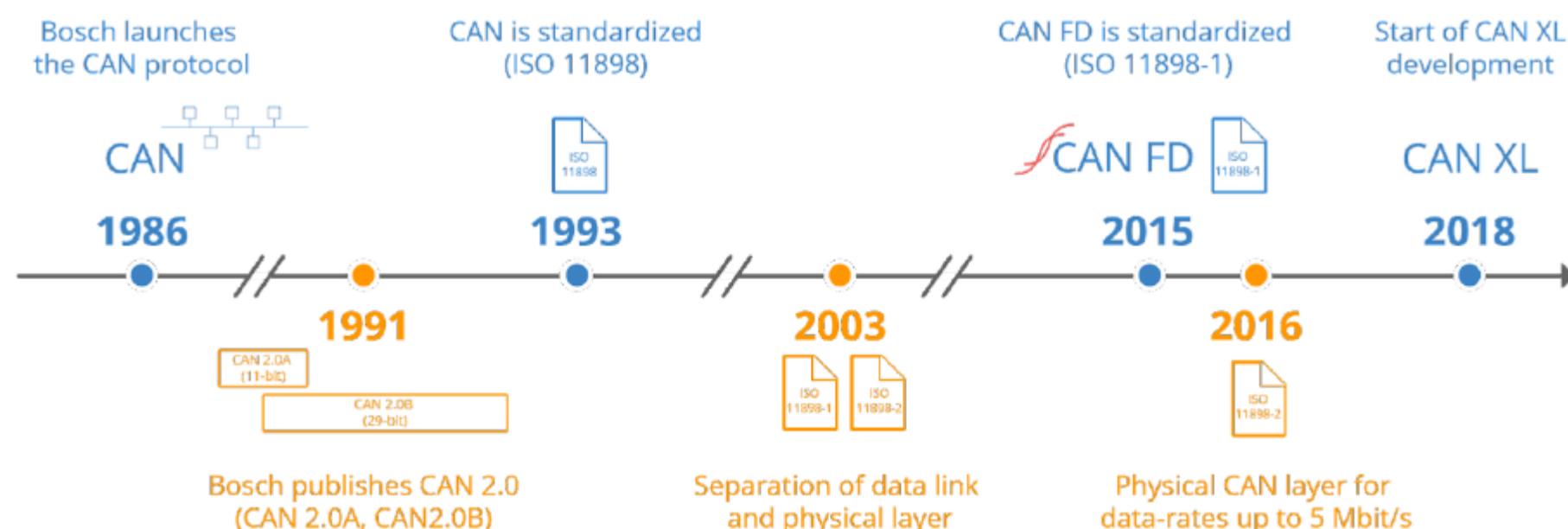


Краткая история шины CAN

До CAN: автомобильные ЭБУ опирались на сложную двухточечную проводку

- 1986: Bosch разработала протокол CAN в качестве решения
- 1991: Bosch опубликовала CAN 2.0 (CAN 2.0A: 11 бит, 2.0B: 29 бит)
- 1993: CAN принят в качестве международного стандарта (ISO 11898)
- 2003: стандарт ISO 11898 становится стандартом серии
- 2012: Bosch выпустила CAN FD 1.0 (гибкая скорость передачи данных)
- 2015: Протокол CAN FD стандартизирован (ISO 11898-1)
- 2016: Физический уровень CAN для передачи данных со скоростью до 5 Мбит / с, стандартизированный по ISO 11898-2

Сегодня CAN входит в стандартную комплектацию автомобилей (легковых, грузовых, автобусов, тракторов, ...), кораблей, самолетов, аккумуляторов для электромобилей, техники и многое другое.



Будущее CAN bus

Заглядывая в будущее, можно сказать, что протокол шины CAN останется актуальным, хотя на него будут влиять основные тенденции.:

Потребность во все более расширенной функциональности транспортных средств.

[Рост облачных вычислений.](#)

Рост Интернета вещей (IoT) и подключенных транспортных средств.

[Влияние автономных транспортных средств](#)

В частности, рост числа подключенных транспортных средств (V2X) и облачных технологий приведет к быстрому развитию телематики транспортных средств и IoT CAN регистраторы. В свою очередь, перевод сети CAN bus в режим "онлайн" также подвергает транспортные средства угрозам безопасности - и может потребовать перехода на новые протоколы CAN, такие как CAN FD.

Появление CAN FD

По мере расширения функциональных возможностей автомобиля увеличивается и нагрузка на CANbus. Для поддержки этого была разработана CAN FD (гибкая скорость передачи данных) шина CAN "следующего поколения". В частности, CAN FD обладает тремя преимуществами (по сравнению с классическим CAN):

Он обеспечивает скорость передачи данных до 8 Мбит / с (по сравнению с 1 Мбит / с).

Он позволяет загружать данные объемом до 64 байт (по сравнению с 8 байтами)

Он обеспечивает повышенную безопасность за счет аутентификации.

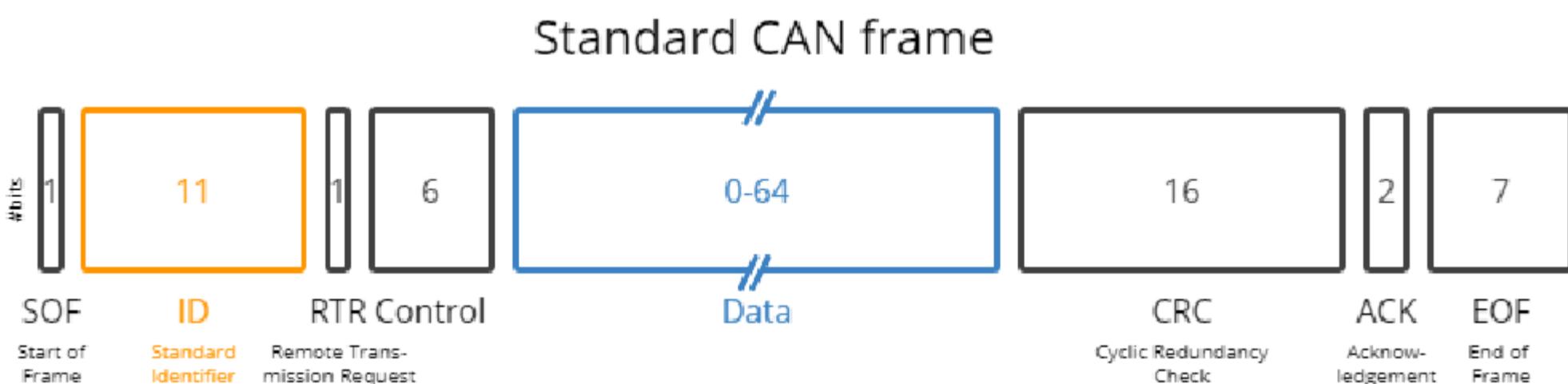
Короче говоря, CAN FD повышает скорость и эффективность - и поэтому внедряется в новых автомобилях. Это также приведет к [возрастающей потребности в регистрациях IoT CAN FD](#).

"Первые автомобили, использующие CAN FD, появятся в 2019/2020 годах, и CAN FD постепенно заменит классический CAN" - CAN в области автоматизации (CiA), "CAN 2020: будущее технологии CAN"

Что такое CAN-фрейм?

Связь по шине CAN осуществляется с помощью CAN-фреймов.

Ниже приведена стандартная рамка CAN с 11-битным идентификатором (CAN 2.0A), который используется в большинстве автомобилей. Расширенный 29-битный рамка идентификатора (CAN 2.0B) идентична, за исключением более длинного идентификатора. Например, он используется в протоколе | 1939 для транспортных средств большой грузоподъемности. Обратите внимание, что идентификатор CAN и данные выделены - они важны при записи данных шины CAN, как мы увидим ниже.



8 полей сообщений протокола шины CAN

SOF: начало кадра - это "доминирующий 0", сообщающий другим узлам, что узел CAN намеревается подключиться. ID: ID является идентификатором кадра - меньшие значения имеют более высокий приоритет.

RTR: запрос удаленной передачи указывает, отправляет ли узел данные или запрашивает выделенные данные у другого узла

Элемент управления: Элемент управления содержит бит расширения идентификатора (IDE), который является "доминирующим 0" для 11-разрядного. Он также содержит 4-разрядный код длины данных (DLC), который определяет длину передаваемых байтов данных (от 0 до 8 байт).

Данные: Данные содержат байты данных, или полезную нагрузку, которая включает сигналы CAN, которые могут быть извлечены и декодированы для получения информации

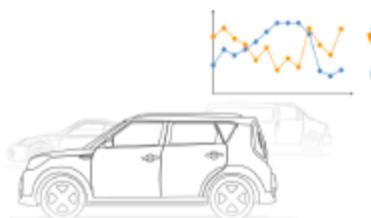
CRC: проверка циклического резервирования используется для обеспечения целостности данных

Подтверждение: слот подтверждения указывает, правильно ли узел подтвердил и получил данные

EOF: EOF отмечает конец кадра CAN

Регистрация данных CAN - примеры использования

Существует несколько распространенных вариантов использования для записи кадров данных шины CAN:



Logging/streaming data from cars

OBD2 data from cars can e.g. be used to reduce fuel costs, improve driving, test prototype parts and insurance

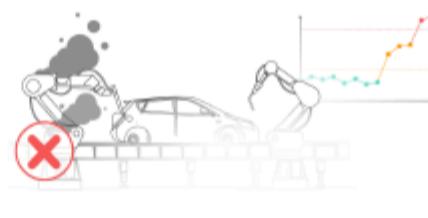
[learn more](#)



Heavy duty fleet telematics

J1939 data from trucks, buses, tractors etc. can be used in fleet management to reduce costs or improve safety

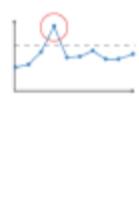
[learn more](#)



Predictive maintenance

Vehicles and machinery can be monitored via IoT CAN loggers in the cloud to predict and avoid breakdowns

[learn more](#)



Vehicle/machine blackbox

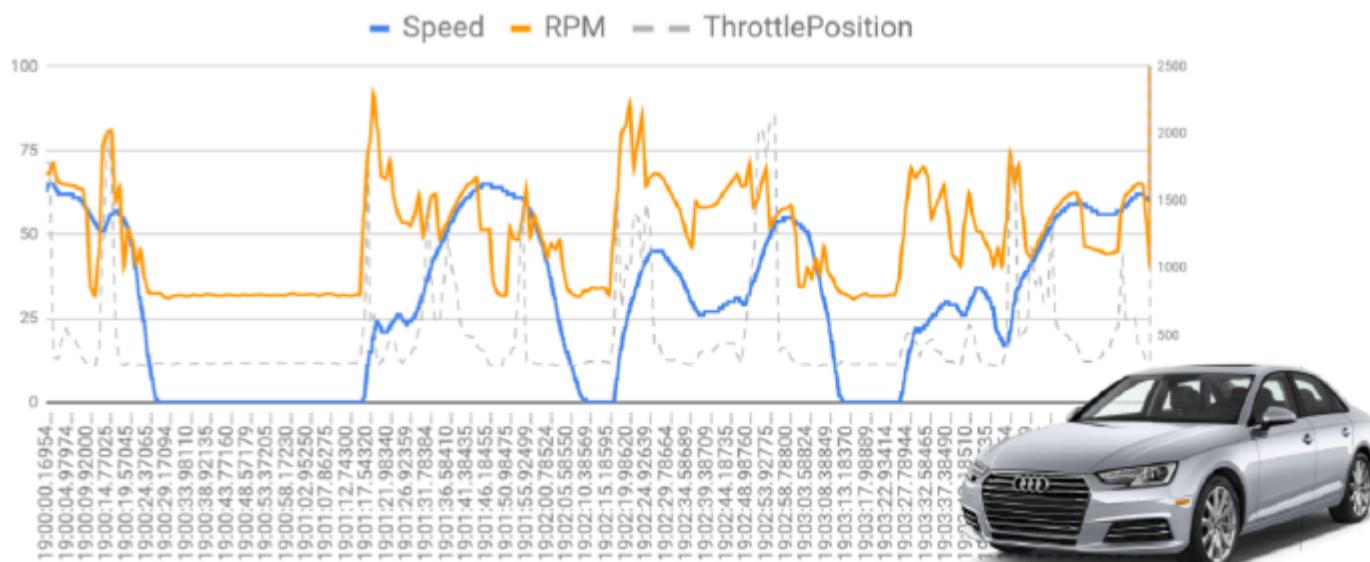
A CAN logger can serve as a 'blackbox' for vehicles or equipment providing data for legal disputes or diagnostics

[learn more](#)

Как зарегистрировать данные шины CAN

Как упоминалось, для регистрации CAN важны два поля CAN: идентификатор CAN и данные. Для записи данных CAN вам понадобится Регистратор CAN. Это позволяет записывать данные CAN с отметкой времени на SD-карту. В некоторых случаях для потоковой передачи требуется интерфейс CAN данные на ПК - например, для взлома автомобиля.

Данные OBD2 - скорость, об/мин, ThrottlePos (Audi A4, CANedge2)



Подключение к шине CAN

Первым шагом является подключение регистратора CAN к шине CAN. Обычно для этого используется кабель-адаптер.:

Автомобили: В большинстве автомобилей для подключения используется адаптер OBD2. В большинстве автомобилей это позволит вам регистрировать необработанные данные CAN, а также выполнять запросы для регистрации данных OBD2 или UDS (Unified Diagnostic Services)

Автомобили большой грузоподъемности: для регистрации данных J1939 с грузовиков, экскаваторов, тракторов и т.д. Обычно можно подключиться к [CAN-шина J1939 через стандартный соединительный кабель J1939 \(9-контактный deutsch\)](#).

Морские перевозки: Большинство судов используют протокол NMEA 2000 и позволяют подключаться через адаптер M12 для регистрации морские данные

CANopen: Для ведения журнала CANopen часто можно напрямую использовать разъем CiA 303-1 DB9 (т.е. Используемый по умолчанию [разъем для наших регистраторов CAN](#), опционально с удлинителем шины CAN

Бесконтактный: При отсутствии разъема типичным решением является использование бесконтактного считывателя CAN, например, CANCrocodile. Это позволяет записывать данные непосредственно с необработанного жгута проводов CAN twisted, без прямого подключение к шине CAN (часто полезно в целях гарантии)

Другое: на практике, используется множество других разъемов, и часто вам потребуется создать собственную шину CAN [адаптер - здесь полезен универсальный адаптер с открытым проводом](#)

После того, как вы определили правильный разъем и подтвердили вывод, вы можете подключить регистратор CAN для начала записи данных. Для CANedge/CLX000, может скорость передачи определяется автоматически и устройство начнет записывать необработанные данные немедленно.

Например: сырье может выбирать данных (J1939)

При желании вы можете загрузить необработанные образцы OBD2 и J1939 из CANedge2 в наших вводных документах. Вы можете, например, загрузить это данные в бесплатных программных средствах декодирования шины CAN. Данные с CANedge записываются в популярном двоичном формате MF4, но может быть преобразован в любой формат файла с помощью наших простых конвертеров MF4 (например, в CSV, ASC, TRC, ...).

Ниже приведен CSV-пример необработанных кадров CAN, зарегистрированных с большегрузного грузовика с использованием протокола J1939. Обратите внимание, что Идентификаторы CAN и байты данных представлены в шестнадцатеричном формате:

Временная метка; BusChannel; ИДЕНТИФИКАТОР; IDE; DLC; длина данных; Dir; EDL; BRS; Байты данных
1578922367.777150;1;14FEF131;1;8;8;0;0;0; CFFFFF300FFF30
1578922367.777750;1;10F01A01;1;8;8;0;0;0; 2448FFFFFFFFFF
1578922367.778300;1;CF00400;1;8;8;0;0;0; 107D82BD1200F482
1578922367.778900;1;14FF0121;1;8;8;0;0;0; FFFFFFFFFF

Пример: CANedge CAN logger

CANedge1 позволяет легко записывать данные с любой шины CAN на SD-карту объемом 8-32 ГБ. Просто подключите его, например, к легковому или грузовому автомобилю, чтобы начать запись - и декодируйте данные с помощью бесплатного программного обеспечения / API. Кроме того, CANedge2 (Wi-Fi) а CANedge3 (3G / 4G) позволяет автоматически передавать данные на ваш собственный сервер - и обновлять устройства по воздуху.



Как декодировать необработанные данные CAN в "физические значения"

Если вы просмотрите приведенный выше пример необработанных данных шины CAN, вы, вероятно, заметите: необработанные данные шины CAN не доступны для чтения человеком.

Чтобы интерпретировать это, вам необходимо декодировать кадры CAN в масштабированном виде технические значения, также известные как физические значения (км / ч, degC, ...).

Ниже мы пошагово покажем, как это работает.



Извлечение сигналов CAN из необработанных кадров CAN

Каждый кадр CAN на шине содержит некоторое количество сигналов CAN (параметров) в байтах данных CAN. Например, Кадр CAN с определенным идентификатором CAN может содержать данные, например, для 2 сигналов CAN.



Для извлечения физического значения сигнала CAN требуется следующая информация:

Начало бита: с какого бита начинается сигнал

Длина бита: длина сигнала в битах

Смещение: значение для смещения значения сигнала на

Масштаб: значение для умножения значения сигнала на

Чтобы извлечь сигнал CAN, вы "вырезаете" соответствующие биты, принимаете десятичное значение и выполняете линейное масштабирование: $\text{physical_value} = \text{смещение} + \text{масштаб} * \text{raw_value_decimal}$

Проблема, связанная с проприетарными данными CAN

Чаще всего "правила декодирования" шины CAN являются проприетарными и нелегкодоступными (за исключением OEM, то есть оригинального Производителя оборудования). Для решения этой проблемы, если вы не являетесь производителем, существует ряд решений.:

Запись данных J1939: если вы регистрируете необработанные данные с транспортных средств большой грузоподъемности (грузовиков, тракторов, ...), вы обычно записываете данные J1939. Это стандартизовано для разных брендов, и вы можете использовать наш файл J1939 DBC для декодирования данных. [Смотрите также наше введение о регистраторе данных J1939](#)

Запись данных OBD2/UDS: Если вам нужно записать данные из автомобилей, вы обычно можете запросить данные OBD2 / UDS, которые стандартизированы для всех автомобилей. Подробнее смотрите наше введение к регистратору данных OBD2 и наш бесплатный файл DBC OBD2.

Используйте общедоступные файлы DBC: для автомобилей существуют онлайн-базы данных, некоторые из которых были переработаны другими пользователями данных CAN. Мы храним список таких баз данных в нашем файле DBC intro

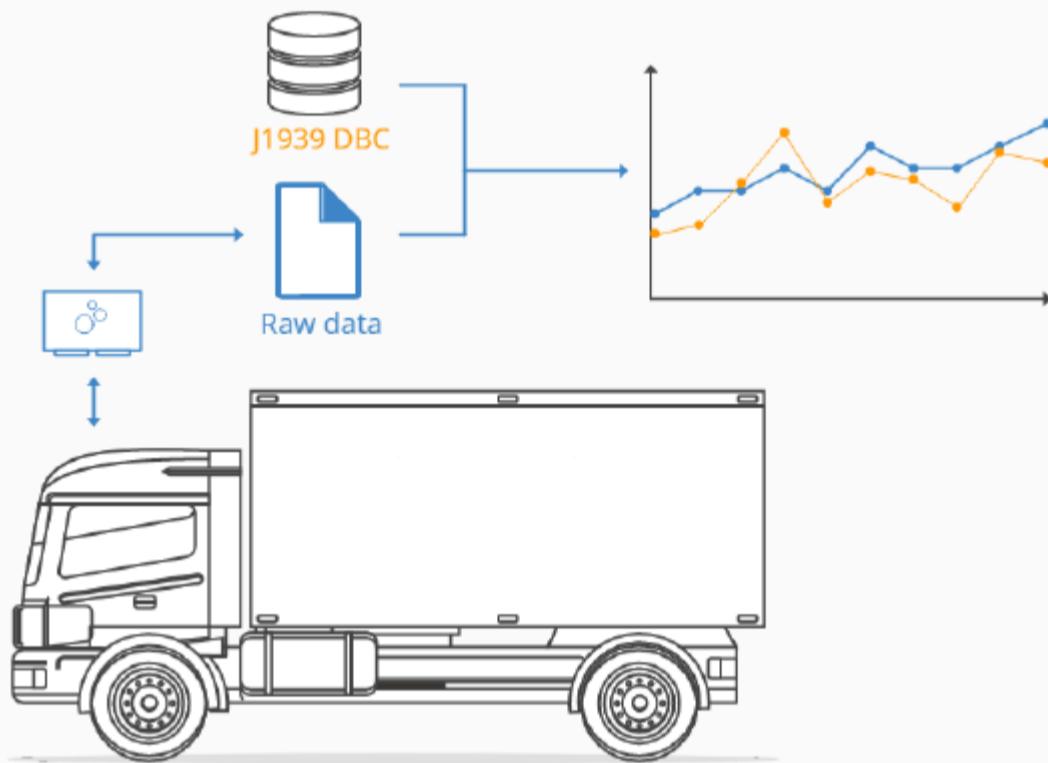
Обратная инженерия данных: вы также можете попытаться выполнить обратную инженерию данных самостоятельно с помощью анализатора шины CAN, хотя это может занять много времени и быть сложным.

Используйте модули датчиков: В некоторых случаях вам могут понадобиться данные датчиков, которые недоступны на шине CAN (или которые сложно реконструировать). Здесь могут использоваться модули типа sensor-to-CAN, такие как серия CANmod. Вы можете интегрировать такие модули с вашей шиной CAN или используйте их в качестве дополнений к вашему регистратору CAN для добавления таких данных, как [GNSS / IMU](#) или [данные о температуре](#)

Партнер с OEM: В некоторых случаях OEM предоставит правила декодирования как часть системы CAN bus технические спецификации. В других случаях вы сможете получить информацию, например, через партнерские отношения

Файлы базы данных CAN (DBC) - пример J1939

В некоторых случаях правила преобразования являются стандартными для всех производителей - например, в протоколе J1939 для тяжелых условий эксплуатации. Это означает, что вы можете использовать правила преобразования параметров J1939 практически на любом транспортном средстве большой грузоподъемности для преобразования значительной части ваших данных. Чтобы сделать это практическим, вам нужен формат для хранения правил преобразования. Здесь CAN формат базы данных (DBC) является отраслевым стандартом и поддерживается большинством программного обеспечения для декодирования шины CAN, включая вспомогательные инструменты для наших регистраторов CAN, asammfd и CANvas. Мы также предлагаем недорогой DBC-файл J1939, который вы можете приобрести в цифровом виде. С его помощью вы сможете быстро преобразовать необработанные данные J1939 в удобочитаемую форму. Учиться [подробнее!](#)



Пример: Декодированные данные выборки CAN (физические значения)

Чтобы проиллюстрировать, как можно извлечь сигналы CAN из необработанных кадров данных CAN, мы приводим ниже предыдущий пример J1939 данные - но теперь расшифрованы с помощью файла J1939 DBC с использованием инструмента asammfd GUI. Как видно, результатом являются данные временной серии с такими параметрами, как температура масла, обороты двигателя, GPS, расход топлива и скорость:

```
>timestamps, ActualEnginePercentTorque, EngineSpeed, EngineCoolantTemperature, EngineOilTemperature1, EngineFuelRate, EngineTotalIdleHours, FuelLevel1, Latitude, Longitude, WheelBasedVehicleSpeed
2020-01-13 16:00:13.259449959+01:00, 0,1520.13,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.268850088+01:00, 0,1522.88,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.270649910+01:00, 0,1523.34,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.271549940+01:00, 0,1523.58,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.278949976+01:00, 0,1525.5,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
```

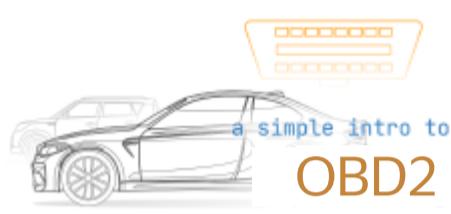
<Подробнее о регистрации данных J1939 читайте в наших статьях о регистраторе данных J1939 и телематике интеллектуального анализа данных. Вы также можете узнать, как анализируйте / визуализируйте свои данные CAN с помощью бесплатного графического интерфейса asammfd или информационных панелей телематики.

Какова связь между CAN, J1939, OBD2, CANopen ...?

Локальная сеть контроллера обеспечивает основу для связи, но не намного больше.

Например, стандарт CAN не определяет, как обрабатывать сообщения размером более 8 байт - или как декодировать необработанные данные. Поэтому существует набор стандартизованных протоколов для дальнейшего определения способа передачи данных между узлами CAN данной сети.

Некоторые из наиболее распространенных стандартов включают SAE J1939, OBD2 и CANopen. Кроме того, эти протоколы более высокого уровня будут все чаще основываться на "следующем поколении" CAN, CAN FD (например, CANopen FD и J1939-17/22).



SAE J1939

J1939 является стандартом бортовая сеть для транспортных средств большой грузоподъемности (например, грузовиков и автобусов). J1939 параметры (например, число оборотов в минуту, скорость, ...) идентифицируются с помощью подозрительного параметра число (SPN), которые сгруппированы в параметре группы, обозначенные PG число (PGN).

[j1939 введение](#)
[телеинформатика j1939](#)

OBD2

Бортовая диагностика (OBD, ISO 15765) является самодиагностика и возможность создания отчетов, которые например, механики используют для выявления проблем с автомобилем. OBD2 указывает на диагностическую неисправность коды (DTCS) и данные в режиме реального времени (например, скорость, об/мин), которые могут быть записаны через Регистраторы OBD2.

[введение в obd2](#)
[протоколирование obd2](#)

CANopen

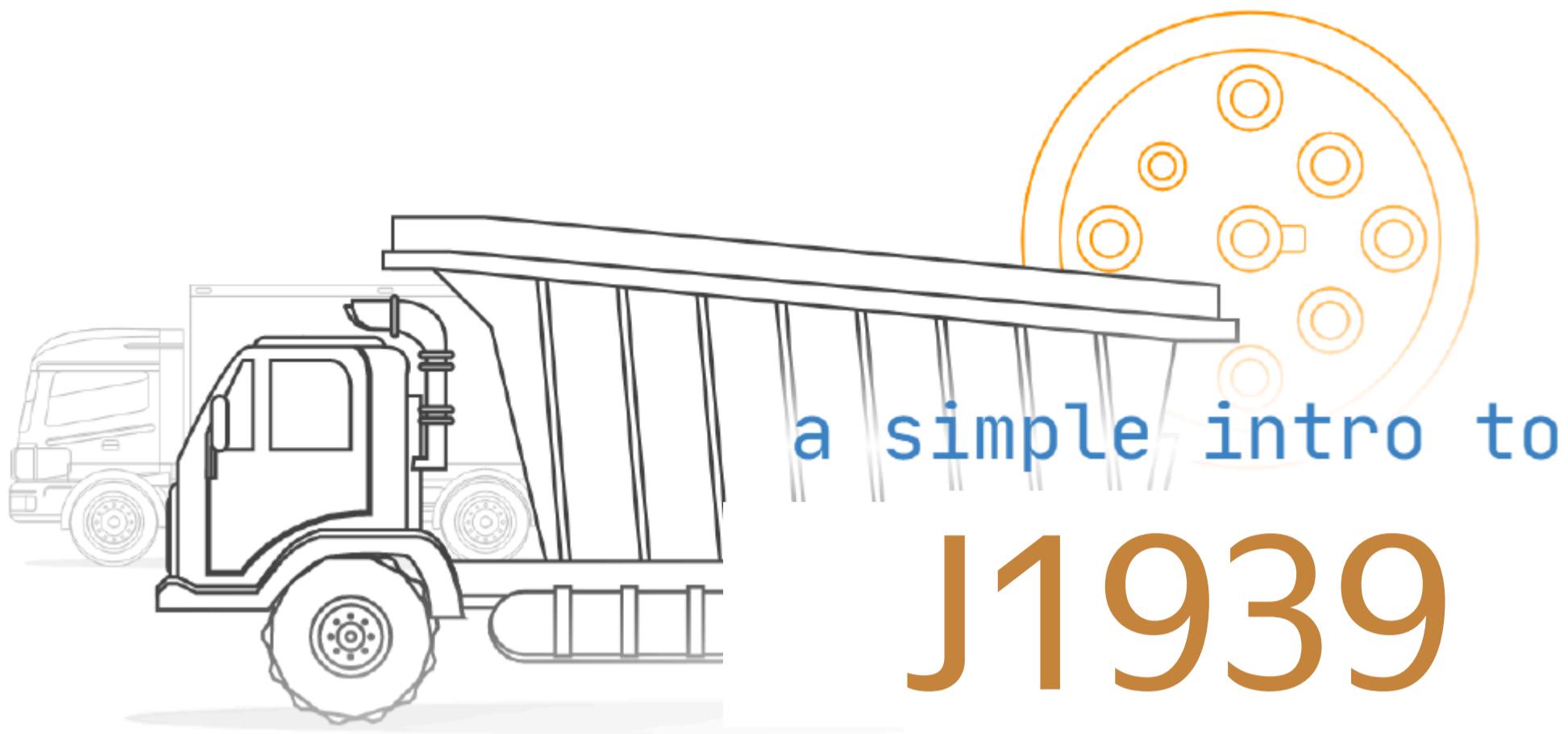
CANopen широко используется встроенном управлении приложениях, включая, например, промышленную автоматизацию. Он основан на CAN, что означает что регистратор данных шины CAN также может регистрировать данные CANopen . Это ключевой момент, например, в диагностике оборудования или оптимизации производства.

[введение в canopen](#)
[регистратор canopen](#)

CAN FD

Шина CAN с гибким управлением скорость передачи данных (CAN FD) является расширением классического Канального уровня CAN. IT увеличивает полезную нагрузку с 8 до 64 байт и обеспечивает более высокую скорость передачи данных, в зависимости от CAN трансивер. Это позволяет использовать все более интенсивные данные варианты использования, такие как EVs.

[can fd intro](#)
[can fd logger](#)



J1939 объяснено - простое введение

В этом руководстве мы знакомим с основами протокола J1939, включая PGN и SPN. Это практическое введение, поэтому вы также узнаете, как декодировать данные J1939 с помощью файлов DBC, как работает ведение журнала J1939, основные варианты использования и практические советы.

Что такое J1939?

Короче говоря, SAE J1939 - это набор стандартов, которые определяют, как ЭБУ взаимодействуют по шине CAN в автомобилях большой грузоподъемности. Как объясняется во введении к шине CAN, сегодня большинство автомобилей используют локальную сеть контроллера (CAN) для связи с ЭБУ. Однако CAN-шина обеспечивает только "основу" для общения (подобно телефону), а не "язык" для разговора.

В большинстве автомобилей большой грузоподъемности этот язык соответствует стандарту SAE J1939, определенному Обществом инженеров автомобильной промышленности (SAE). С технической точки зрения, J1939 предоставляет протокол более высокого уровня (HLP), основанный на CAN как "физическом уровне". Что это означает?

Единый стандарт для автомобилей большой грузоподъемности

Простыми словами, J1939 предлагает стандартизованный метод связи между блоками управления, или, другими словами: J1939 обеспечивает общий язык между производителями. Напротив, например, в автомобилях используются проприетарные протоколы, специфичные для OEM-производителей.

Примеры применения J1939

Транспортные средства большой грузоподъемности (например, грузовики и автобусы) являются одним из наиболее известных применений. Однако несколько других ключевых отрасли промышленности сегодня используют SAE J1939 либо напрямую, либо через производные стандарты (например, ISO 11783, MilCAN, NMEA 2000, FMS):

Лесозаготовительные машины (например, рубители, форвардеры, трелевочные машины)

Горнодобывающая техника (например, бульдозеры, драглайны, экскаваторы, ...)

Военная техника (например, танки, транспортные средства, ...)

Сельское хозяйство (например, тракторы, комбайны, ...)

Строительство (например, мобильная гидравлика, краны, ...)

Пожарно-спасательные работы (например, машины скорой помощи, пожарные машины, ...)

Прочее (например, суда, насосы, электронные автобусы, производство электроэнергии, ...)

История и будущее J1939

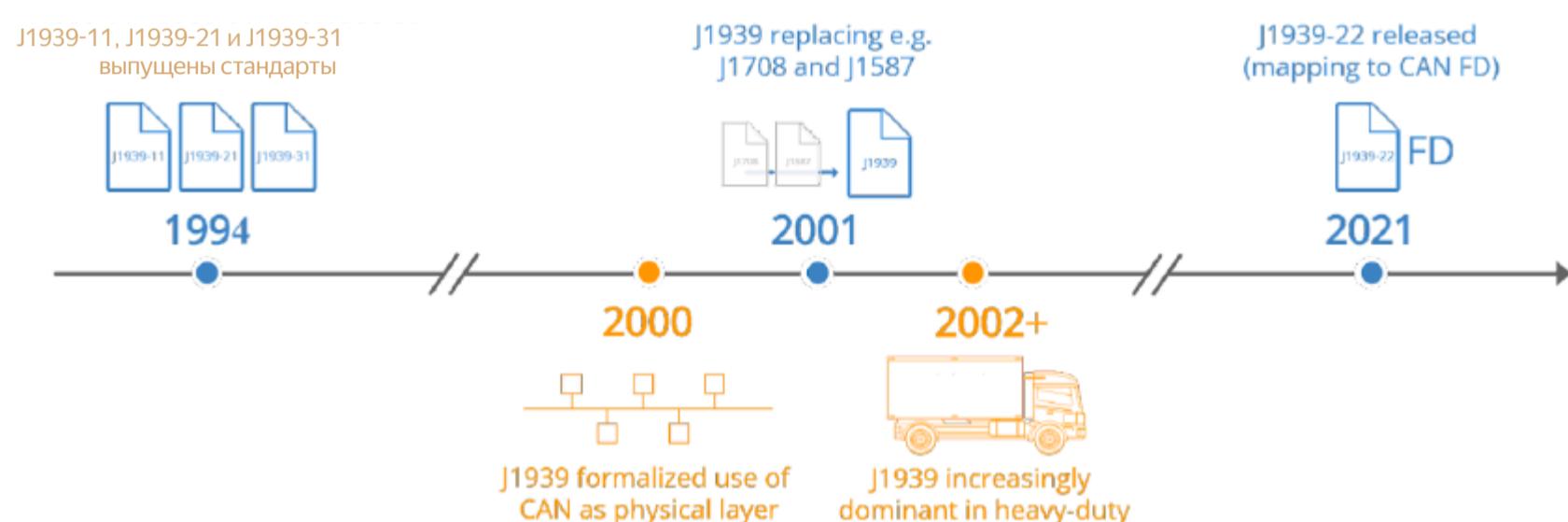
История

1994: Выпущены первые документы (J1939-11, J1939-21, J1939-31)

2000: Опубликован первоначальный документ верхнего уровня

2000: CAN официально включен как часть стандарта J1939

2001: J1939 начинает замену бывших стандартов SAE J1708 / J1587



Будущее

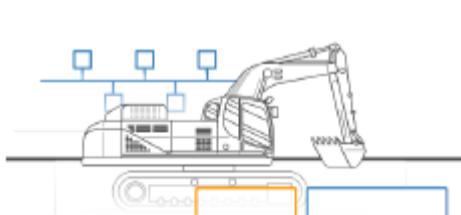
С развитием телематики для тяжелых условий эксплуатации J1939 будет играть все большую роль на рынке подключенных транспортных средств. В свою очередь, это увеличит потребность в безопасных регистрациях партий J1939. Параллельно OEM-производители будут все чаще переходить с классического CAN на CAN FD в рамках перехода на J1939 с гибкой скоростью передачи данных. В свою очередь, это увеличит потребность в регистрациях данных J1939 FD.

"Рынок подключаемых устройств в автомобиле - оборудования и услуг, предоставляющих всевозможные новые функциональные возможности водителям и владельцам автопарка - ожидается, что к 2020 году он достигнет 120 миллиардов евро".

- Boston Consulting Group, Подключенные транспортные средства и путь к доходам

4 ключевые характеристики J1939

Протокол J1939 имеет набор определяющих характеристик, описанных ниже:



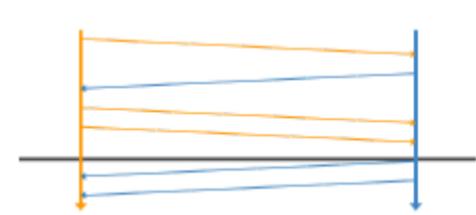
Скорость передачи данных 250 Тыс. бод и 29-битный расширенный идентификатор



Большинство сообщений J1939 широковещательная передача осуществляется по CAN-шине, хотя некоторые данные доступны только доступно путем запроса



Идентификаторы PGN и SPN параметры Сообщения J1939 идентифицируются с помощью 18-разрядного Номера групп параметров (PGN), в то время как сигналы J1939 называются подозрительными Номера параметров (SPN)



Отправляются многобайтовые переменные сначала младший по значению байт (Порядок байтов Intel). PGNS с объемом до 1785 байт поддерживается с помощью J1939 транспортный протокол

Дополнительные характеристики J1939

Ниже приведен набор дополнительных характеристик протокола J1939:

Зарезервировано: J1939 включает в себя широкий спектр стандартных PGN, хотя PGN с 00FF00 по 00FFFF зарезервированы для частного использования

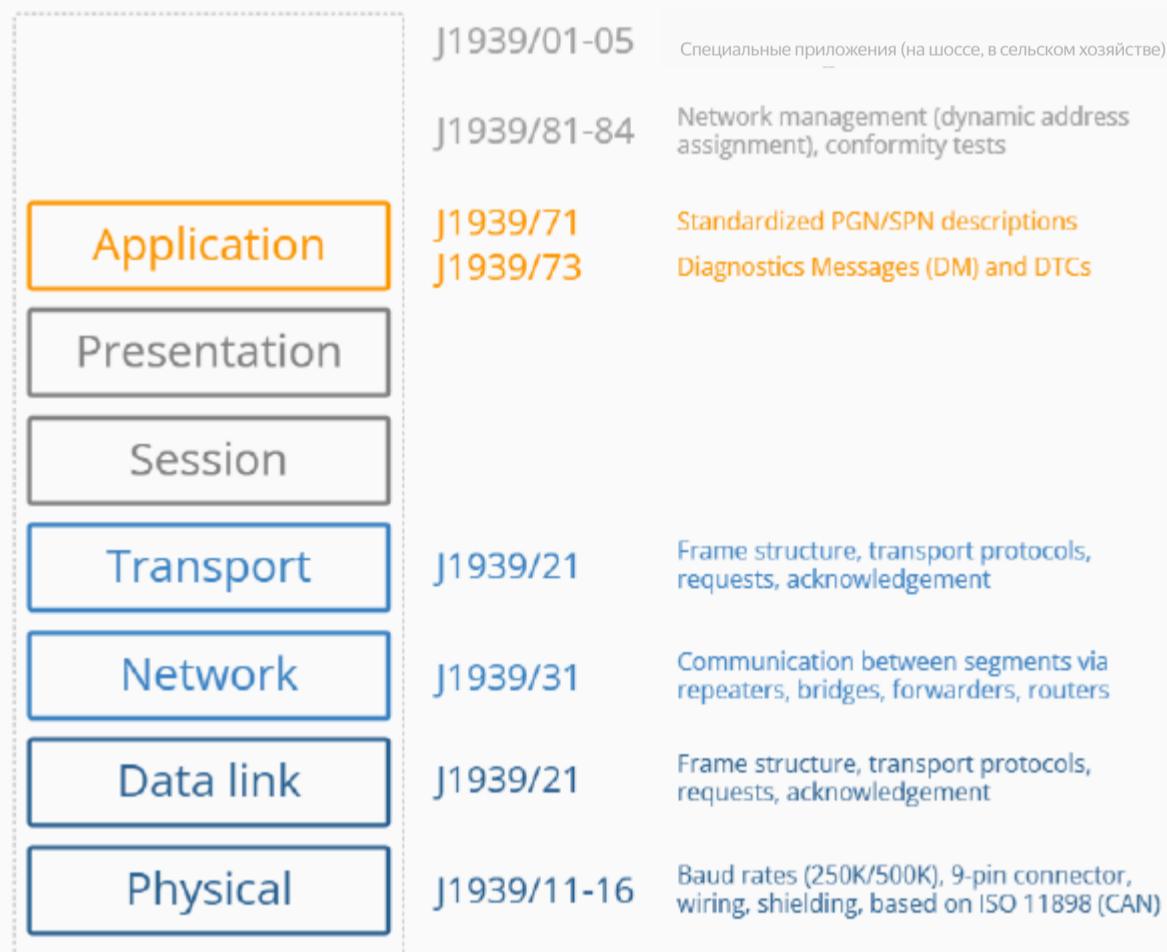
Особые значения: байт данных 0xFF (255) отражает отсутствие данных, в то время как 0xFE (254) отражает ошибку

Запрос адреса J1939: Стандарт SAE J1939 определяет процедуру присвоения адресов источников J1939 ECU после сетевой инициализации с помощью 8-разрядного адреса динамическим способом

Технические характеристики: J1939 объясняет "протокол более высокого уровня"

J1939 основан на CAN, который обеспечивает базовый "физический уровень" и "канальный уровень", самые низкие уровни в OSI Модель. В принципе, CAN позволяет передавать небольшие пакеты по шине CAN, но не намного больше. Здесь, J1939 служит протоколом более высокого уровня поверх, обеспечивая более сложную связь.

7-уровневая модель OSI | стандарты J1939



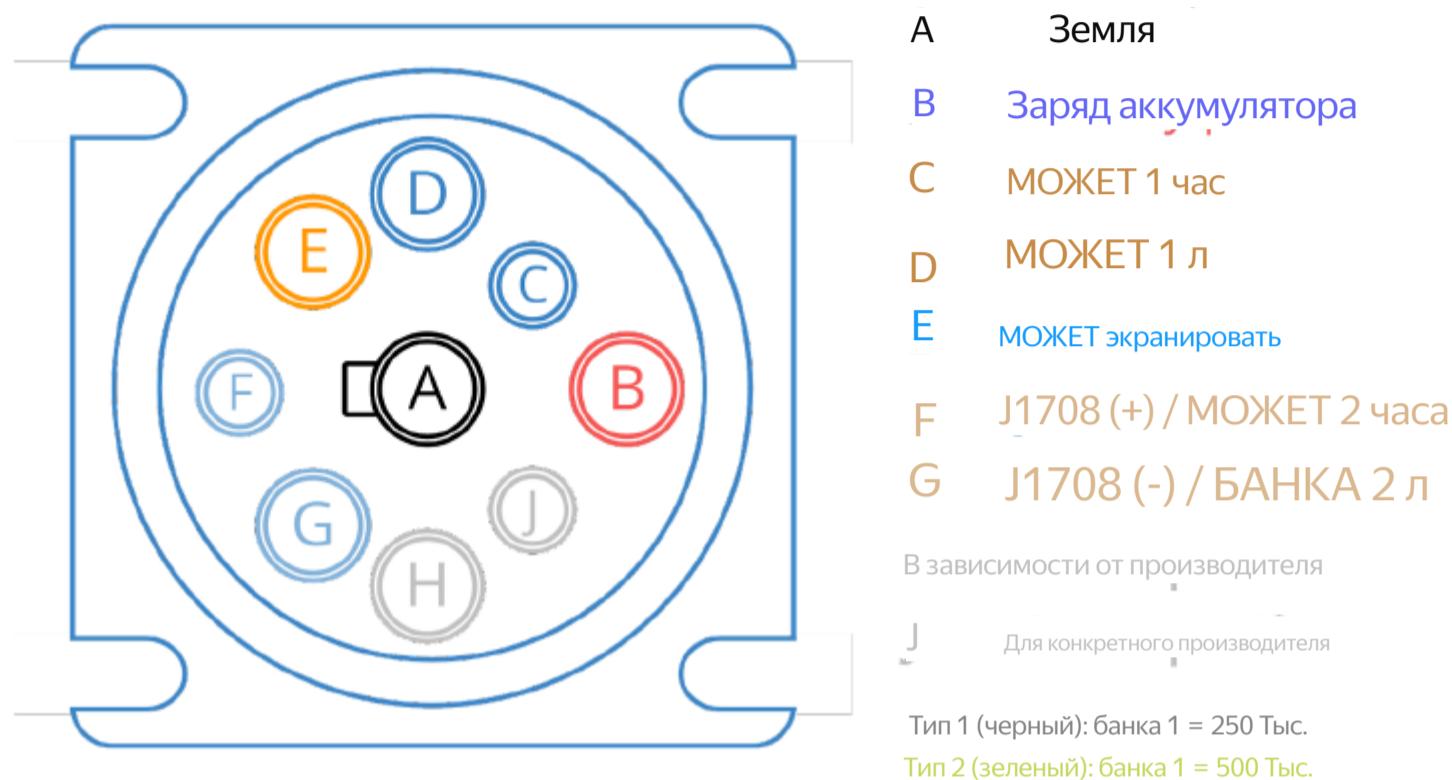
Протокол более высокого уровня обеспечивает связь по большим сложным сетям, например, производителей транспортных средств.

Например, протокол SAE J1939 определяет, как обрабатывать "многопакетные сообщения", то есть когда объем данных превышает 8 необходимо передать байты. Аналогично, он определяет, как данные должны быть преобразованы в данные, доступные для чтения человеком. Он делает это, предоставляем семейство стандартов. Например, J1939-71 - это документ, детализирующий информацию, необходимую для преобразования большого набора стандартизованных сообщений J1939 от разных производителей в понятные человеку данные (подробнее об этом ниже). Существует множество других протоколов более высокого уровня на основе CAN, например CANopen, DeviceNet, Unified Diagnostic Services. Обычно они предлагают определенный уровень стандартизации в соответствующих отраслях, хотя все они могут быть расширены производителями.

Для сравнения, вышеупомянутые легковые автомобили имеют уникальные стандарты для каждого производителя. Другими словами, вы можете использовать один и тот же файл базы данных J1939 для преобразования, например, частоты вращения двигателя двух грузовиков разных производителей, но вы невозможно, например, прочитать данные с Audi A4, используя те же идентификаторы и параметры масштабирования, что и для Peugeot 207.

Разъем J1939 (9-контактный)

Стандарт J1939-13 определяет "встроенный диагностический разъем", также известный как разъем J1939 или 9-контактный deutsch разъем. Это стандартизированный метод для подключения к сети J1939 большинства автомобилей большой грузоподъемности - см. на иллюстрации показана схема подключения разъема J1939.



Черный тип 1 против зеленого типа 2

Обратите внимание, что разъем J1939 deutsch поставляется в двух вариантах: оригинальный черный разъем (тип 1) и более новый зеленый разъем (тип 2), который начал выпускаться примерно в 2013-14 годах.

Разъем J1939 (9-контактный deutsch)



самец



Тип 1 женский



самец



Тип 2

Женский

Разъемы-розетки J1939 типа 2 физически имеют обратную совместимость, то время как разъемы-розетки типа 1 подходят только для типа 1 разъемы-розетки. Разъем типа 2 был разработан для стандарта SAE J1939-14, который добавляет поддержку 500 Тыс. бит скорости передачи данных. Целью "блокирования" разъемов типа 1 является убедитесь, что старое оборудование (предположительно использующее скорость передачи данных 250 Тыс. бит) не подключено к сетям типа 2 со скоростью передачи данных 500 тыс. бит J1939. В частности, физический блок подключается через меньшее отверстие для контакта F в штекерных разъемах типа 2. Смотрите также пример кабеля-адаптера DB9-J1939 (тип 2).



Несколько сетей J1939

Как видно, разъем J1939 deutsch обеспечивает доступ к сети J1939 через контакты C (CAN high) и D (CAN low). Это упрощает взаимодействие с сетью J1939 на большинстве автомобилей большой грузоподъемности.

Однако в некоторых случаях вы также можете получить доступ к вторичной сети J1939 через контакты F и G или контакты H и J (где H означает CAN H, а J - CAN L).

Многие современные автомобили большой грузоподъемности имеют 2 или более параллельных сетей CAN-шин, а в некоторых случаях по крайней мере две из них будут доступны через один и тот же разъем J1939. Это также означает, что вы не обязательно получите доступ ко всем доступным данным J1939, если вы пытались подключиться только через "стандартные" контакты C и D.

Другие мощные разъемы

Хотя разъем J1939 deutsch является наиболее распространенным способом подключения к сети J1939 автомобилей большой грузоподъемности, конечно, существуют и другие разъемы. Ниже приведены несколько примеров:

Магистральный разъем J1939: В этом 3-контактном разъеме deutsch предусмотрены контакты для подключения CAN H / L к экрану CAN (без питания / заземления)

Разъем CAT: Промышленный разъем Caterpillar представляет собой 9-контактный разъем deutsch серого цвета. Однако вывод отличается от разъема J1939 (A: питание, B: заземление, F: CAN L, G: CAN H) и физически блокирует доступ к разъему доступ к стандартным разъемам типа 1 и 2 J1939

Разъем OBD2 типа B: Разъем OBD2 типа B (SAE J1962) в автомобилях большой грузоподъемности иногда обеспечивает прямой доступ к сети J1939 через контакты 6 и 14

Разъем OBD2 Volvo 2013: Этот вариант соответствует разъему OBD2 типа B, но также предоставляет доступ к J1939 high через контакт 3 и J1939 low через контакт 11

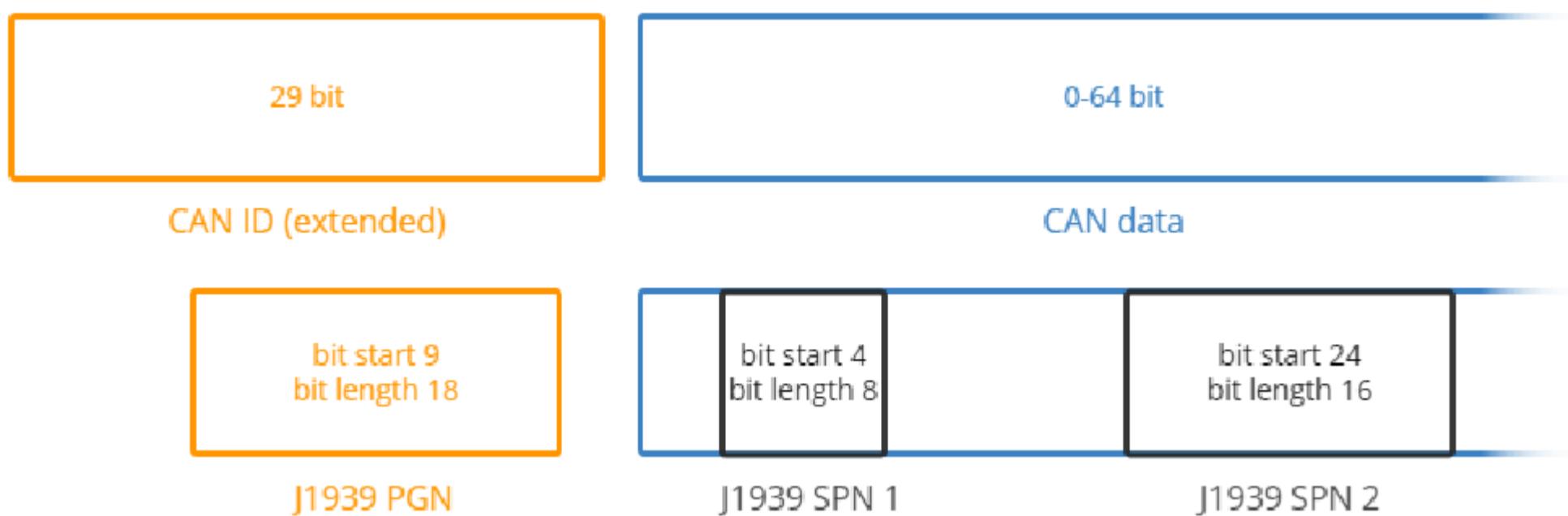
PGN и SPN J1939

В следующем разделе мы расскажем о PGN и SPN J1939.

Номер группы параметров (PGN)

J1939 PGN содержит 18-разрядное подмножество 29-разрядного расширенного идентификатора CAN. Проще говоря, PGN служит уникальным идентификатором кадра в стандарте J1939. Например, вы можете посмотреть это в стандартной документации J1939-71, в которой перечислены PGN / SPNs.

Сообщение J1939 (PGN & SPNS)



Пример: J1939 PGN 61444 (EEC1)

Предположим, вы записали сообщение J1939 с шестнадцатеричным идентификатором 0CF00401. Здесь PGN начинается с бита 9 и имеет длину 18 (индексируется с 1). Результирующий PGN равен 0F004 или в десятичной системе счисления 61444. Просмотрев это в документации SAE J1939-71, вы обнаружите, что это "Электронный контроллер двигателя 1 - EEC1". Кроме того, документ будет содержать подробную информацию о PGN, включая приоритет, скорость передачи и список связанных SPN - см. иллюстрацию. Для этого PGN существует семь SPN (например, двигатель Частота вращения, об / мин), каждый из которых можно посмотреть в документации J1939-71 для получения дополнительной информации.

PGN61444 - Электронный контроллер двигателя 1 - EEC1

Частота повторения передачи: зависит от частоты вращения двигателя
Длина данных: 8 байт
Страница данных: 0
Формат PDU: 240
Для конкретного PDU: 4
Приоритет по умолчанию: 3
Номер группы параметров: 61444 (0x00F004)

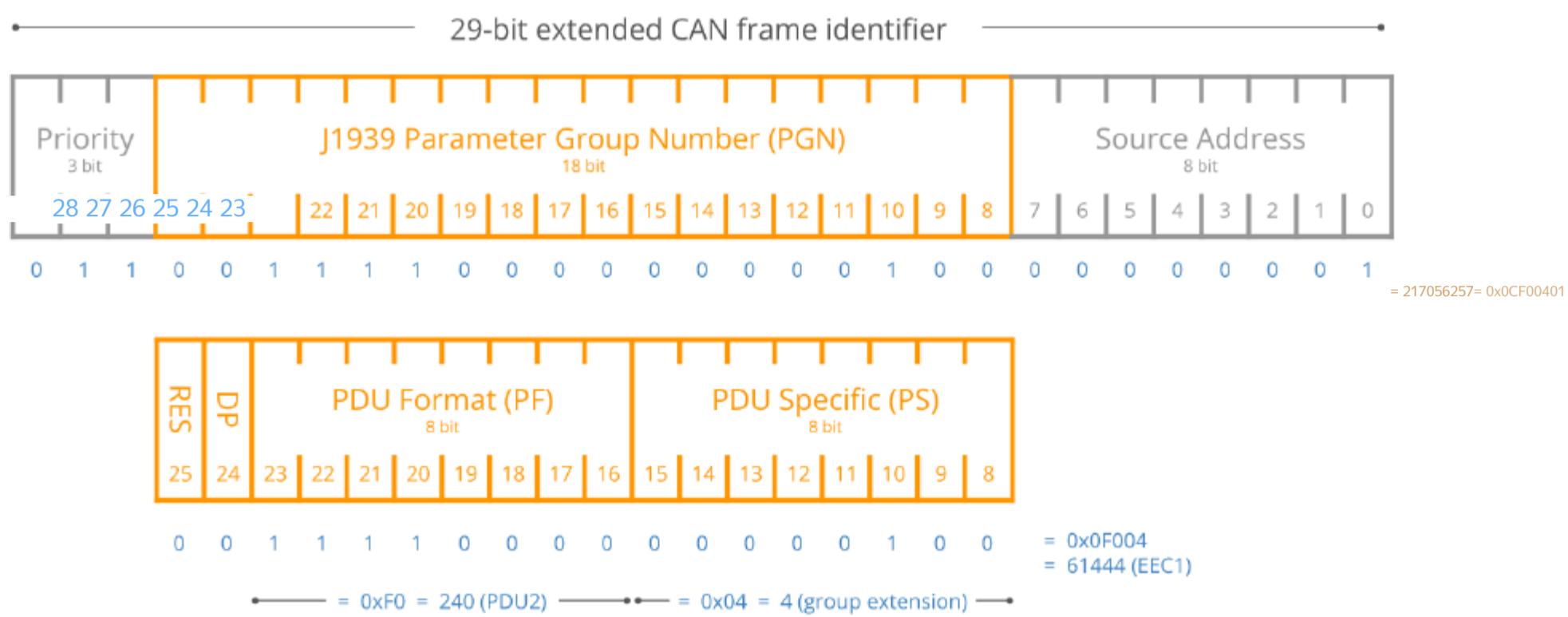
Начало бита / байт	Длина	Идентификатор SPN	Описание SPN
1.1	4 бита	899	Режим крутящего момента двигателя
2...	1 байт	512	Двигатель по требованию водителя - % крутящего момента
3...	1 байт	513	Фактический крутящий момент двигателя в процентах
4-5	2 байта	190	Частота вращения двигателя
6	1 байт	1483	Описание Управляющего устройства для управления двигателем
7.1	4 бита	1675	Режим запуска двигателя
	1 байт	2432	Потребляемый двигателем крутящий момент в процентах

Подробная разбивка J1939 PGN

Давайте подробно рассмотрим переход с CAN ID на PGN. В частности, 29-битный идентификатор CAN включает приоритет (3 бита), J1939 PGN (18 бит) и адрес источника (8 бит). В свою очередь, PGN может быть разделен на зарезервированный бит (1 бит), страницу данных (1 бит), Формат PDU (8 бит) и специфичный для PDU (8 бит).

Подробная иллюстрация PGN также включает примеры значений для каждого поля в двоичной, десятичной и шестнадцатеричной форме.

Чтобы узнать больше о переходе с 29-разрядного CAN ID на 18-разрядный J1939 PGN, ознакомьтесь также с нашим онлайн-конвертером CAN ID в J1939 PGN . Конвертер также включает полный список J1939 PGN для PGN, включенных в наш файл J1939 DBC.



blue: Example values

RES: Reserved | DP: Data Page | PDU: Protocol Data Unit (message format)

PF < 240: Message is PDU1 (addressable message, PS contains destination address)

PF >= 240: Message is PDU2 (broadcast message, PS contains group extension)

Номер подозрительного параметра (SPN)

SPN J1939 служит идентификатором сигналов (параметров) CAN, содержащихся в байтах данных. SPN сгруппированы по PGN и могут быть описаны в терминах их начальной позиции в битах, длины в битах, масштаба, смещения и единицы измерения - информации, необходимой для извлечения и масштабирования данных SPN до физических значений.



Пример: Извлечение J1939 SPN 190 (частота вращения двигателя)

Предположим, что вы записали необработанный кадр J1939, как показано ниже:

```
>CAN ID           <>Data bytes
<>0CF00401       <>FF FF FF 68 13 FF FF FF
```

Преобразуя идентификатор CAN в J1939 PGN, вы определяете, что это PGN 61444 из предыдущих. Из J1939-71 в документе вы видите, что одним из SPN в этом PGN является частота вращения двигателя (SPN 190) с подробностями, как показано на иллюстрации ниже.

Используя эти сведения, можно извлечь данные о физических значениях частоты вращения двигателя, например, для построения графика. Для этого обратите внимание на информацию SPN. Следует, что соответствующие данные представлены в байтах 4 и 5, то есть в байтах 68 и 13 шестнадцатеричных данных. Принимая десятичную форму из шестнадцатеричного значения 1368 (порядок байтов Intel) мы получаем 4968 в десятичной системе счисления. Чтобы получить число оборотов в минуту, мы проводим масштабирование этого значения используя смещение 0 и шкалу 0,125 об / мин / бит. Физическое значение (также известное как масштабированное инженерное значение) составляет 621 об/мин.

Обратите внимание, что некоторые байты данных в приведенном выше виде имеют десятичную дробь FF или 255, т.е. недоступны. Хотя PGN теоретически может поддерживать SPN в этом диапазоне, заполнение FF означает, что данное конкретное приложение не поддерживает эти параметры.

J1939 SPN 190 | Engine Speed

Actual engine speed which is calculated over a minimum crankshaft angle of 720 degrees divided by the number of cylinders.

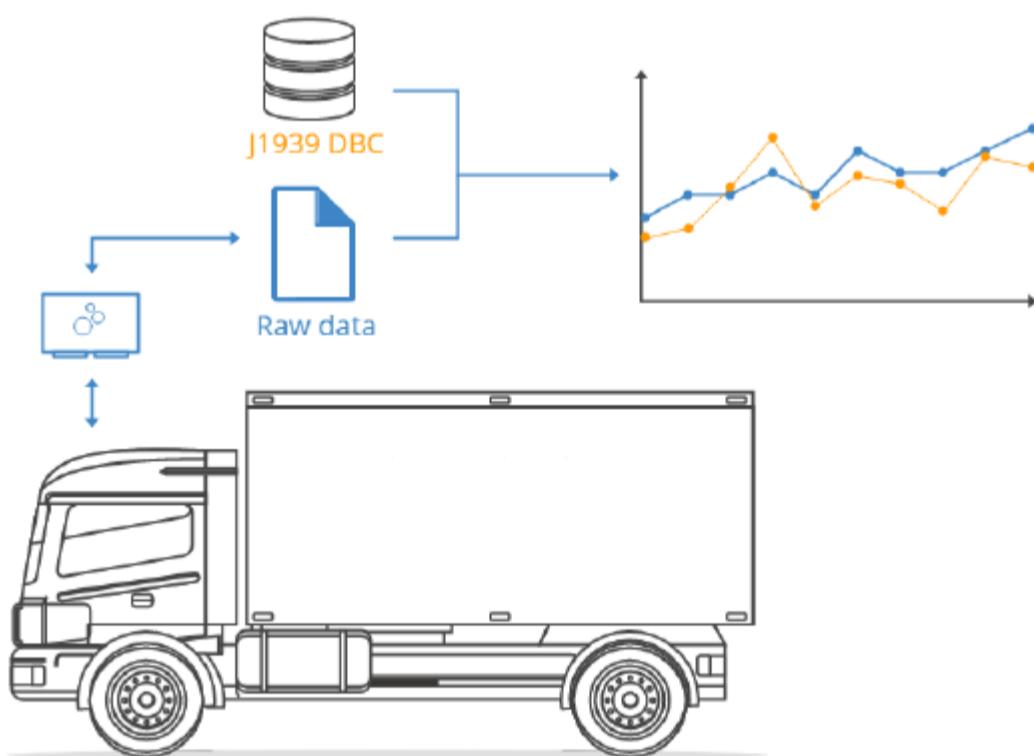
Bit start: 24
Length: 2 bytes
Scale: 0.125
Offset: 0
Unit: rpm
min-max: 0-8031.875
PGN reference: 61444



На практике вы не применяете правила поиска в формате PDF для данных J1939 - вместо этого эта информация хранится в файле базы данных DBC.

Пример: файл J1939 DBC

Файл J1939 DBC может использоваться для декодирования данных на большинстве транспортных средств большой грузоподъемности. Например, необработанные данные J1939 могут записываться с помощью регистратора данных шины CAN и анализируется с помощью программного средства CAN, поддерживающего преобразование DBC (например, asammfd). Обычно это приводит к преобразованию 40-60% данных о транспортном средстве, а остальные данные являются собственноностью производителя. [для этого требуется обратный инжиниринг](#).



Примерные данные грузовика J1939: исходные и физические значения

Ниже мы проиллюстрируем, как выглядят реальные данные J1939. "Необработанные" данные J1939 были записаны с большегрузного грузовика с помощью CANedge2, в то время как "физические значения" отражают выходные данные после декодирования необработанных данных с помощью бесплатного программного обеспечения asammfd и [DBC J1939](#).

Пример: Необработанные данные грузовика J1939 (CSV)

Данные из CANedge записаны в стандартном двоичном формате MDF4, который может быть преобразован в любой формат файла с помощью наших конвертеров MDF4 (например, в CSV, ASC, TRC, ...). Ниже приведена CSV-версия необработанных кадров J1939. Обратите внимание, что CAN Идентификаторы и байты данных представлены в шестнадцатеричном формате:

```
>TimestampEpoch;BusChannel;ID;IDE;DLC;DataLength;Dir;EDL;BRS;DataBytes
1578922367.777150;1;14FEF131;1;8;8;0;0;0;CFFFFFFF300FFFF30
1578922367.777750;1;10F01A01;1;8;8;0;0;0;2448FFFFFFFF
1578922367.778300;1;CF00400;1;8;8;0;0;0;107D82BD1200F482
1578922367.778900;1;14FF0121;1;8;8;0;0;0;FFFFFF
1578922367.779500;1;18F0000F;1;8;8;0;0;0;007DFFFF0F7DFFFF
1578922367.780050;1;18FFA03D;1;8;8;0;0;0;2228240019001AFF
1578922367.780600;1;10FCFD01;1;8;8;0;0;0;FFFFFF1623FFFF
1578922367.781200;1;18FD9401;1;8;8;0;0;0;A835FFFFA9168F03
1578922367.781750;1;18FDA101;1;8;8;0;0;0;1224FFFFFF00FF
1578922367.782350;1;18F00E3D;1;8;8;0;0;0;741DFFFFFF
1578922367.782950;1;18F00F3D;1;8;8;0;0;0;B40FFFFFF
1578922367.783500;1;10FDA301;1;8;8;0;0;0;FFFFFF
```

<При желании вы можете загрузить полные образцы raw J1939 MDF4 из CANedge2 в наших вводных документах. Образцы данных также включает демонстрационный J1939 DBC, чтобы вы могли повторить шаги преобразования через asammdf.

Пример: Расшифрованные физические значения данных грузовика J1939 (CSV).

После декодирования и экспорта необработанных данных J1939 результатом являются данные временных рядов с такими параметрами, как температура масла, частота вращения двигателя, GPS, расход топлива и скорость:

```
>timestamps, ActualEnginePercentTorque, EngineSpeed, EngineCoolantTemperature, EngineOilTemperature1, EngineFuelRate, EngineTotalIdleHours, FuelLevel1, Latitude, Longitude, WheelBasedVehicleSpeed
2020-01-13 16:00:13.259449959+01:00,0,1520.13,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.268850088+01:00,0,1522.88,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.270649910+01:00,0,1523.34,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.271549940+01:00,0,1523.58,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.278949976+01:00,0,1525.5,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.289050102+01:00,0,1527.88,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.299000025+01:00,0,1528.13,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.308300018+01:00,0,1526.86,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.309099913+01:00,0,1526.75,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.317199945+01:00,0,1526.45,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
```

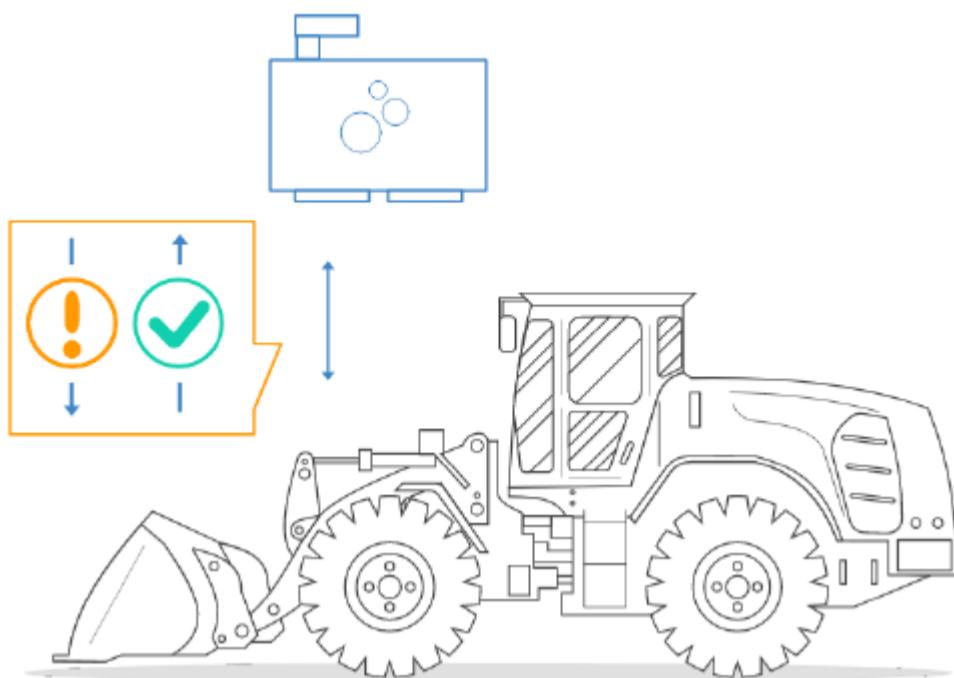
<Чтобы узнать больше о регистрации данных J1939, ознакомьтесь с нашими статьями о регистраторе данных J1939 и телематике интеллектуального анализа данных.

О регистраторе CANedge J1939

CANedge позволяет легко записывать данные J1939 на SD-карту объемом 8-32 ГБ. Просто подключите его, например, к грузовику, чтобы начать регистрацию - [и декодируйте данные с помощью бесплатного программного обеспечения / API](#) и нашего J1939 DBC. Узнать больше.

J1939 запрашивать сообщения

Большинство сообщений J1939 передаются по шине CAN, но некоторые отправляются только "по запросу" (например, при опросе а J1939 регистратор данных). Данные по запросу часто включают диагностические коды неисправностей J1939 (DTCS), что делает их важными в автомобиле диагностика. Ниже мы кратко опишем, как это работает:



Отправка сообщений с запросом J1939

Для отправки запроса J1939 по шине CAN используется специальное "сообщение запроса" (PGN 59904), которое является единственным J1939 сообщением, содержащим всего 3 байта данных. Он имеет приоритет 6, переменную скорость передачи и может быть отправлен как глобальный, так и конкретный запрос адреса. Байты данных 1-3 должны содержать запрошенный PGN (порядок байтов Intel). Примеры запрошенных сообщений J1939 включают диагностические сообщения (например, J1939 DM2).

Регистратор данных шины CAN, такой как CANedge, может быть настроен для отправки сообщений с запросом J1939 - смотрите, например, Наше введение в CANedge для подробное пошаговое руководство.

Запросы кода J1939 против соответствия гарантии

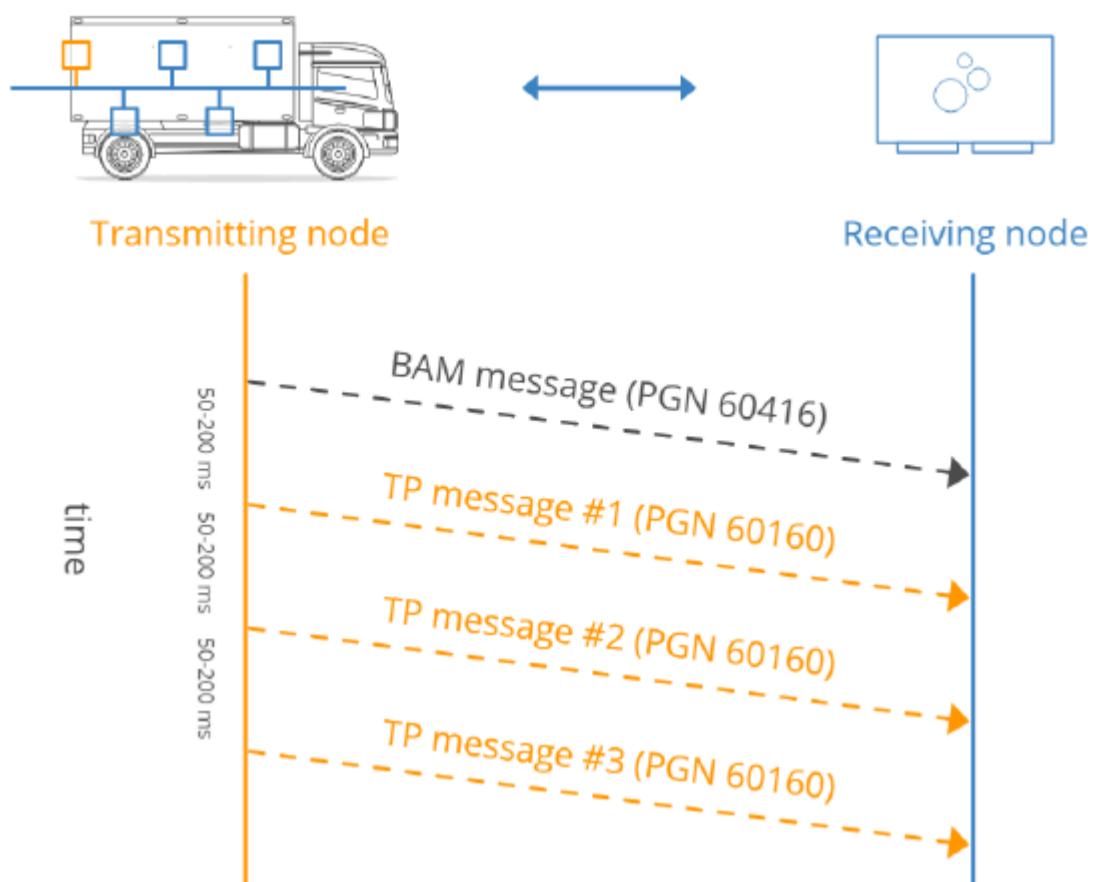
Отправка сообщений с запросом обычно является ключом к запросу кодов J1939 и, следовательно, к диагностике J1939. Одна из проблем, однако от регистраторов J1939 часто требуется бесконтактное подключение к каналу передачи данных J1939, что означает, что они не могут взаимодействовать с шиной CAN и передавать кадры запроса J1939. Ограничение часто связано с соблюдение гарантий, поскольку некоторые производители транспортных средств не допускают прямого доступа сторонних устройств через разъем J1939.

В некоторых случаях требуется, чтобы анализатор J1939 был "физически" бесконтактным, например, с помощью адаптера CANCrocodile. В других в некоторых случаях достаточно, чтобы регистратор J1939 работал в "настраиваемом" бесшумном режиме. Последний упрощает выполнение специальные запросы кодов неисправностей J1939 либо через обновление конфигурации вручную, либо через обновление по воздуху для CANedge2.

Транспортный протокол J1939 (TP)

Предыдущие примеры PGN и SPN основаны на сообщениях J1939 с 8 байтами данных. Хотя они наиболее распространены, Многокадровые сообщения J1939 также существуют с объемом данных >8 байт - передаются по транспортному протоколу J1939.

Ниже мы описываем, как работает транспортный протокол J1939, практический пример данных TP J1939 и как декодировать многокадровые сообщения J1939 с помощью файлов DBC:



Как работает транспортный протокол J1939?

Протокол J1939 определяет, как деконструировать, передавать и повторно собирать пакеты по нескольким кадрам - процесс упоминается как транспортный протокол J1939 (см. J1939-21). Существуют два типа ТР J1939:

1. Режим подключения (предназначен для конкретного устройства).
2. ВАМ (широковещательное сообщение об объявлении), предназначенное для всей сети

Например, передающий блок управления может отправить начальный пакет ВАМ для настройки передачи данных. ВАМ определяет PGN идентификатор для многопакетного сообщения, а также количество байтов данных и пакетов, подлежащих отправке. Затем следует до 255 пакетов /кадров данных. Каждый из 255 пакетов использует первый байт данных для указания порядкового номера (от 1 до 255), за которым следуют 7 байтов данных. Следовательно, максимальное количество байт в многопакетном сообщении составляет 7 байт x 255 = 1785 байт.

В Последний пакет содержит по крайней мере один байт данных, за которым следуют неиспользуемые байты, имеющие значение FF. В сценарии типа ВАМ время между сообщениями составляет 50-200 мс. При последующей обработке программный инструмент преобразования может повторно собрать несколько вводит 7 байт данных в одну полезную нагрузку и обрабатывает ее в соответствии со спецификациями многопакетных PGN и SPN, как найдено, например, в файле DBC J1939.

Практический пример транспортного протокола J1939

Декодирование многокадровых данных J1939 сложнее, чем декодирование стандартных кадров J1939. Чтобы понять почему, рассмотрим приведенный ниже пример ответа транспортного протокола J1939, записанного с помощью CANedge2:

```
>TimestampEpoch;BusChannel;ID;IDE;DLC;DataLength;Dir;EDL;BRS;DataBytes
1605078459.438750;1;1CECFF00;1;8;8;0;0;0;20270006FFE3FE00
1605078459.498750;1;1CEBF00;1;8;8;0;0;0;013011B2A041B240
1605078459.559750;1;1CEBF00;1;8;8;0;0;0;021FB2102CB2603B
1605078459.618750;1;1CEBF00;1;8;8;0;0;0;03B230430000D309
1605078459.678750;1;1CEBF00;1;8;8;0;0;0;04C0441E37967DE1
1605078459.738750;1;1CEBF00;1;8;8;0;0;0;05E02E7B02FFFF80
1605078459.799850;1;1CEBF00;1;8;8;0;0;0;06E0FFFFFFFFF
```

<Приведенная выше последовательность состоит из двух типов сообщений J1939:

PGN 60416 - ВАМ транспортного протокола
J1939
(Управление подключением)

Длина данных: 8 байт
Приоритет по умолчанию: 7
Номер группы параметров: 60416 (0xEC00)

Байт	Описание
1...	Зафиксировано на уровне 32
2-3	Размер сообщения в байтах
4...	Количество пакетов
5...	Зарезервировано (заполнено FF)
6-8	PGN

Транспортный протокол PGN
60160 - J1939
(Передача данных)

Длина данных: 8 байт
Приоритет по умолчанию: 7
Номер группы параметров: 60160 (0xEB00)

Байт	Описание
1	Порядковый номер (от 1 до 255) Полезная нагрузка данных (неиспользуемые местоположения в для последнего кадра установлено значение FF)

Сообщение J1939 ВАМ с идентификатором 1CECFF00 (PGN 60416 или EC00), которое содержит
длину данных ответа и J1939 PGN - и сообщения о передаче данных J1939 с идентификатором
1CEBFF00 (PGN 60160 или EB00). Они содержат полезную нагрузку в нескольких кадрах.

Ниже мы разберем пример транспортного протокола J1939 с акцентом на интерпретацию байтов данных:

Временная метка (эпоха)	ИДЕНТИФИКАТОР CAN	PGN (шестнадцатеричный)	PGN (десятичный)	Байты данных	условные обозначения
1605078459.438750	1CECFF00	EC00	60416	20270006FFE3FE00	управляющий байт (0x20-E)
1605078459.498750	1CEBFF00	EB00	60160	013011B2A041B240	#байты данных (0x0027=3)
1605078459.559750	1CEBFF00	EB00	60160	021FB2102CB2603B	#пакеты (0x06=6)
1605078459.618750	1CEBFF00	EB00	60160	03B230430000D309	зарезервировано J1939 PGN
1605078459.678750	1CEBFF00	EB00	60160	04C0441E37967DE1	(0x00FEE3=65251)
1605078459.738750	1CEBFF00	EB00	60160	05E02E7B02FFFF80	порядковые номера
1605078459.799850	1CEBFF00	EB00	60160	06E0FFFFFFFFFFFF	данные полезной нагрузки неиспользуемые (3 байта)
1605078459.438750	18FEE3FE	FEE3	65251	3011B2A041B240...	E0FFFFFF

Как правило, последовательность ответов транспортного протокола J1939 может обрабатываться следующим образом:

Идентифицируйте кадр ВАМ, указывающий на инициирование новой последовательности (через PGN 60416)

Извлеките PGN J1939 из байтов 6-8 полезной нагрузки ВАМ для использования в качестве идентификатора нового кадра

Создайте новую полезную нагрузку данных путем объединения 2-8 байтов кадров передачи данных (т.е. Кроме 1-го байта)

Выше последние 3 байта ВАМ равны E3FE00. При изменении порядка они равны PGN FEE3, или конфигурации двигателя 1 (EC1). Далее, полезная нагрузка определяется путем объединения первых 39 байт в 6 пакетах / кадрах передачи данных.

Примечание: Последние 3 байта полезной нагрузки данных в этом практическом примере равны FF, но мы все равно включаем их в полезную нагрузку как в сообщении ВАМ длина данных указана равной 39. Последние 3 FF байта в 6-м пакете не используются.

Как декодировать сообщение транспортного протокола J1939

С помощью описанного выше метода мы создали "сконструированный" фрейм данных J1939 с длиной данных, превышающей 8 байт. Этот фрейм может быть декодирован с использованием файла J1939 DBC, точно так же, как обычный фрейм данных J1939. Для PGN EC1 J1939 DBC определяет длину данных, равную 40, с сигналами, определенными для полной полезной нагрузки.

Таким образом, как только программное обеспечение / API J1939 реконструирует многокадровый ответ в один кадр J1939, DBC декодирование может выполняться как обычно. Одна небольшая настройка заключается в том, что большинство файлов DBC J1939 ожидают, что необработанный файл журнала данных J1939 будет содержать 29-разрядные идентификаторы CAN (не 18-разрядные PGN J1939). Таким образом, если программное обеспечение встраивает восстановленный кадр J1939 TP в исходные необработанные данные, возможно, потребуется сначала преобразовать извлеченный J1939 PGN в 29-битный CAN ID. Вы также можете ознакомиться с нашими Таблицей Google J1939, в которой описывается, как PGN J1939 может быть преобразован в 29-битный идентификатор CAN.

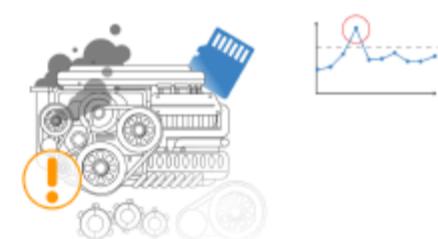
Декодирование данных TP J1939 и Python API.

CANedge позволяет запрашивать и записывать данные транспортного протокола J1939. Для декодирования данных TP вы можете либо преобразовать файлы необработанных журналов в другой формат (например, векторный ASC), или вы можете использовать наш Python API. В нашей библиотеке api-примеров на на github мы приводим базовый пример того, как данные транспортного протокола J1939 могут быть восстановлены и декодированы DBC, включая.

примеры данных. Поскольку CANedge Python API позволяет декодировать данные транспортного протокола J1939, сигналы J1939 из многокадровые сообщения могут, например, визуализироваться на информационных панелях телематики J1939.

Регистрация данных J1939 - примеры использования

Существует несколько распространенных вариантов использования для записи данных J1939:



Heavy duty fleet telematics

J1939 data from trucks, buses, tractors etc. can be used in fleet management to reduce costs or improve safety

[learn more](#)

Live stream diagnostics

By streaming decoded J1939 data to a PC, technicians can perform real-time J1939 diagnostics on vehicles

[learn more](#)

Predictive maintenance

Vehicles can be monitored via [WiFi CAN loggers](#) in the cloud to predict breakdowns based on the J1939 data

[learn more](#)

Heavy-duty vehicle blackbox

A [CAN logger](#) can serve as a 'blackbox' for heavy-duty vehicles, providing data for legal disputes or J1939 diagnostics

[learn more](#)

6 практических советов по ведению журнала данных J1939

Многие наши конечные пользователи работают с журналированием J1939 в полевых условиях - и ниже мы делимся 6 практическими советами по ведению журнала:

J1939 logger vs J1939 streaming interface

Автономные регистраторы данных J1939 с SD-картами идеально подходят для регистрации данных, например, из автопарка за неделю или месяцы. А WiFi J1939 logger также позволяет использовать варианты телематики. В отличие от этого, интерфейс J1939 USB-PC требует наличия ПК для потоковой передачи данных по шине CAN в режиме реального времени. Это, например, полезно для целей диагностики или анализа физических событий. CLX000 поддерживает оба режима работы, в то время как CANedge2 идеально подходит для телематики.

Прямой кабель-адаптер вместо бесконтактного считывания показаний

Для подключения анализатора CAN к оборудованию J1939 (например, грузовику) обычно можно использовать 9-контактный разъем J1939. Мы предлагаем Адаптер DB9-J1939, который подходит к 9-контактному разъему deutsch, который используется во многих автомобилях большой грузоподъемности. В качестве альтернативы вы можете предпочитать подключать свой регистратор CAN напрямую к шине CAN, например, через CANCrocodile. Этот метод использует индукцию для бесшумной записи данных без обрезки проводов CAN.

Загрузка данных через Wi-Fi или сотовую связь (3G / 4G)

Для управления автопарком и телематики вы обычно загружаете данные либо через Wi-Fi, либо через 3G / 4G. CANedge2 позволяет передавать данные, подключаясь к точке доступа Wi-Fi, которая может быть как маршрутизатором WLAN, так и точкой доступа 3G / 4G. Если вам нужны данные с грузовика в дороге, вы можете установить CANedge2 и использовать его для питания точки доступа 3G / 4G USB. В преимуществом этого является то, что у вас будет постоянный доступ к устройству, если только оно не находится вне зоны действия сети. Однако в случаях, когда

данные необходимо загружать только периодически альтернативой может быть загрузка данных через маршрутизаторы WLAN, когда транспортные средства посещают, например, определенные места (гаражи, ремонтные мастерские и т.д.), что позволяет снизить затраты на передачу данных.

Выбор программного обеспечения и файл DBC J1939

При регистрации или потоковой передаче данных J1939 ключевым является программное обеспечение для последующей обработки. В частности, программное обеспечение должно поддерживать Преобразование J1939 на основе DBC для упрощения преобразования в данные, понятные человеку. Бесплатное программное обеспечение / API для наши регистраторы CAN поддерживают это. Для потоковой передачи по USB наш бесплатный плагин Wireshark позволяет конвертировать DBC в реальном времени. Кроме того, мы предлагаем загрузить файл DBC J1939 в цифровом формате в сотрудничестве с SAE.

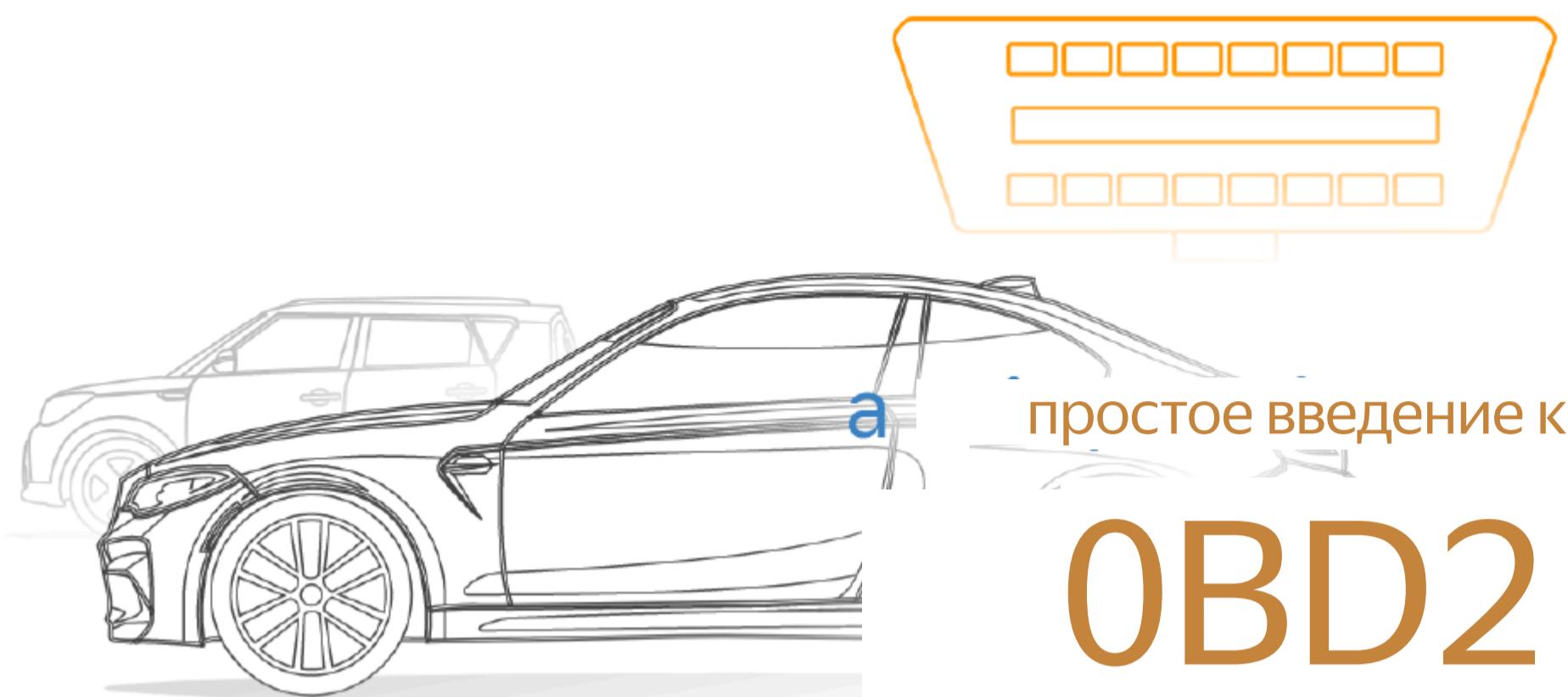
Рассмотрите необходимость в PGN запроса

Некоторые PGN J1939 доступны только по запросу, что означает, что вам необходимо "опросить" шину CAN, чтобы зарегистрировать их. В CANedge и CLX000 способны передавать пользовательские сообщения CAN, которые могут использоваться для отправки периодических запросов PGN. Обратите внимание, что это невозможно в "автоматическом режиме" (т.е. Это невозможно, если регистратор подключен, например, через CANCrocodile).

Фильтруйте, скимайте и шифруйте данные.

Для оптимизации ведения журнала данных J1939 может быть полезен ряд расширенных конфигураций. В частности, CANedge расширенные фильтры и параметры частоты дискретизации помогают оптимизировать объем регистрируемых данных - например, для минимизации использования полосы пропускания сотовой связи . Другие опции включают бесшумный режим и циклическое ведение журнала, причем последнее позволяет регистратору всегда определять приоритетность последних данных (полезно, например, при ведении журнала в черном ящике).

Поскольку J1939 стандартизирован, крайне важно шифровать ваши данные "в состоянии покоя" (например, на SD-карте) и "при передаче" (во время загрузки). Невыполнение этого требования подвергает обработку ваших данных различным рискам безопасности, включая. Штрафы по GDPR / CCPA и потерю конфиденциальности и целостности данных. Для получения подробной информации о защите ведения журнала данных J1939 ознакомьтесь с нашим руководством по безопасному ведению журнала CAN.

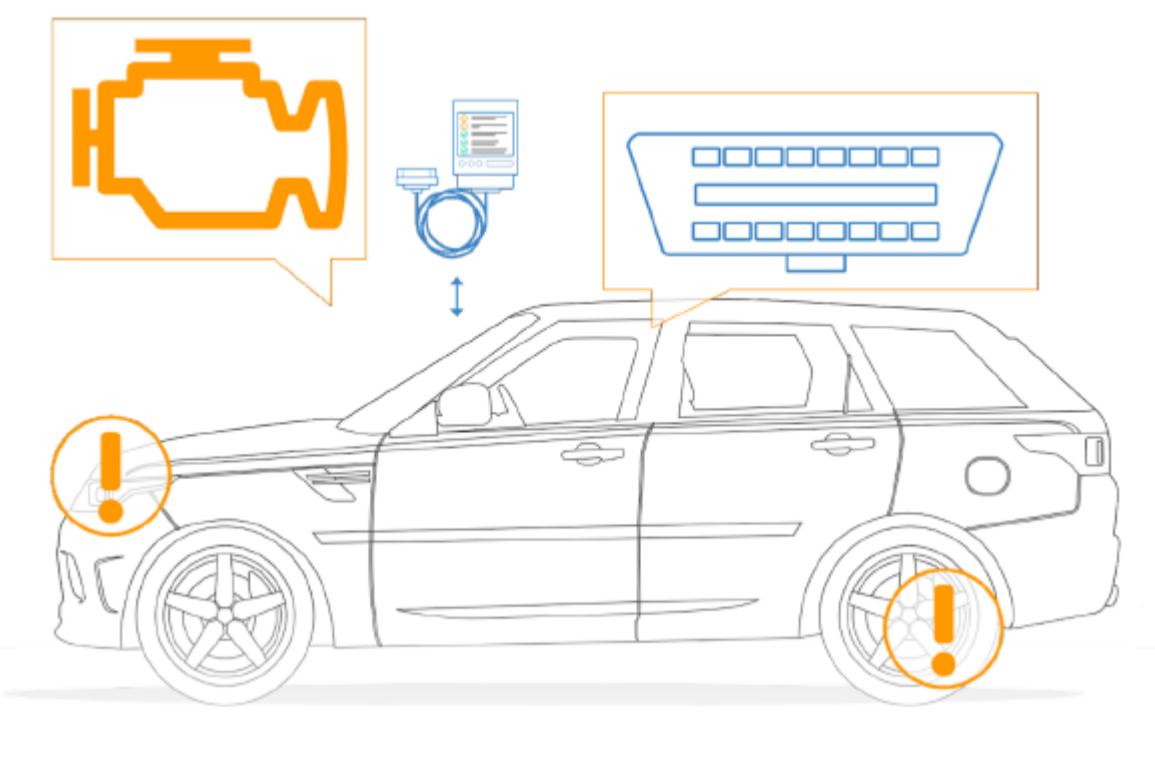


Объяснение OBD2 - простое введение

В этом руководстве мы представляем протокол встроенной диагностики (OBD2), включая разъем OBD2, идентификаторы параметров OBD2 (PID) и связь с CAN bus. Это практическое введение, поэтому вы также узнаете, как запрашивать и декодировать данные OBD2, регистрировать ключи примеры использования и практические советы.

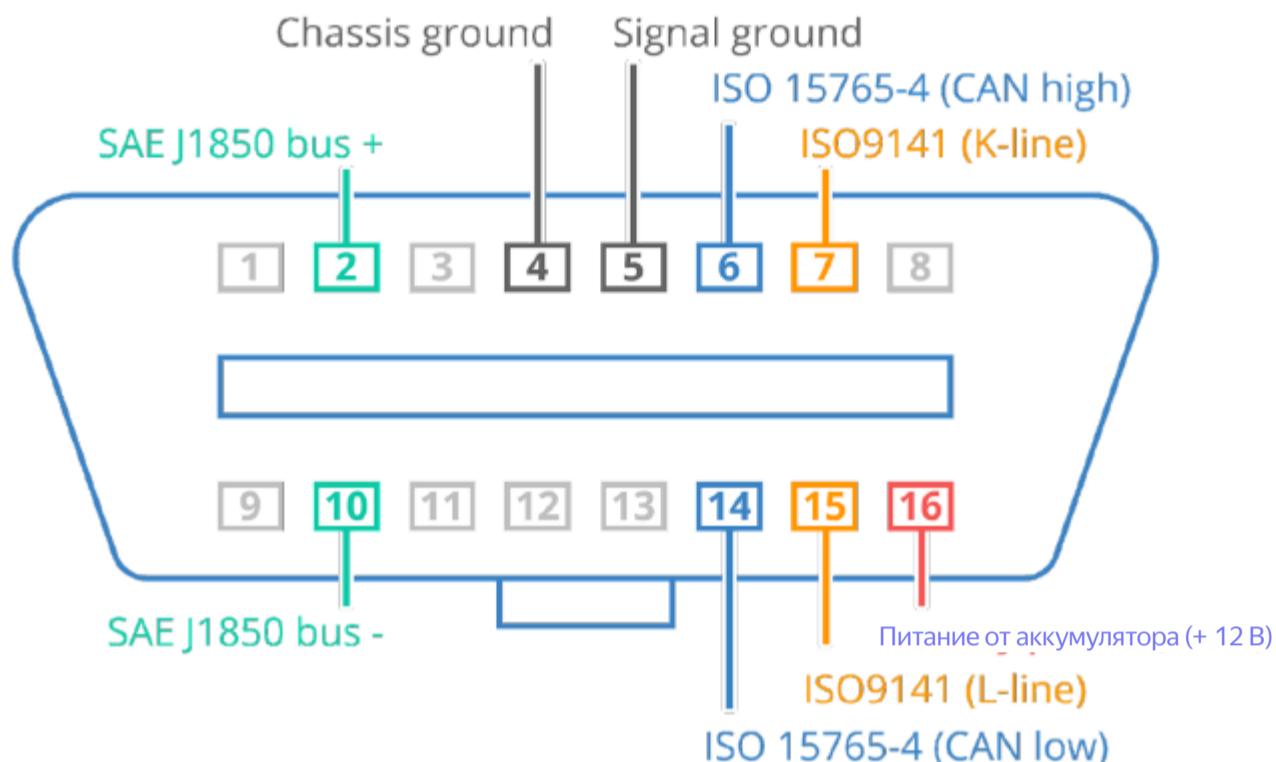
Что такое OBD2?

Короче говоря, OBD2 - это встроенная в ваш автомобиль система самодиагностики. Вы, вероятно, уже сталкивались с OBD2: когда-нибудь замечали световой индикатор неисправности на вашей приборной панели? Это ваш автомобиль сообщает вам, что есть проблема. Если вы обратитесь к механику, он использует сканер OBD2 для диагностики проблемы. Для этого он подключит считыватель OBD2 к 16-контактному разъему OBD2 рядом с рулевым колесом. Это позволяет ему считывать коды OBD2, также известные как диагностические коды неисправностей (DTCs), для просмотра и устранения неполадок проблема.



Разъем OBD2

Разъем OBD2 позволяет легко получать доступ к данным из вашего автомобиля. Стандарт SAE J1962 определяет два 16-контактных разъема OBD2 типы разъемов (A и B). На иллюстрации приведен пример контактного разъема OBD2 типа А (также иногда называемого разъем канала передачи данных, DLC).



Следует отметить несколько моментов.:

Разъем OBD2 находится рядом с рулевым колесом, но может быть скрыт за крышками / панелями.

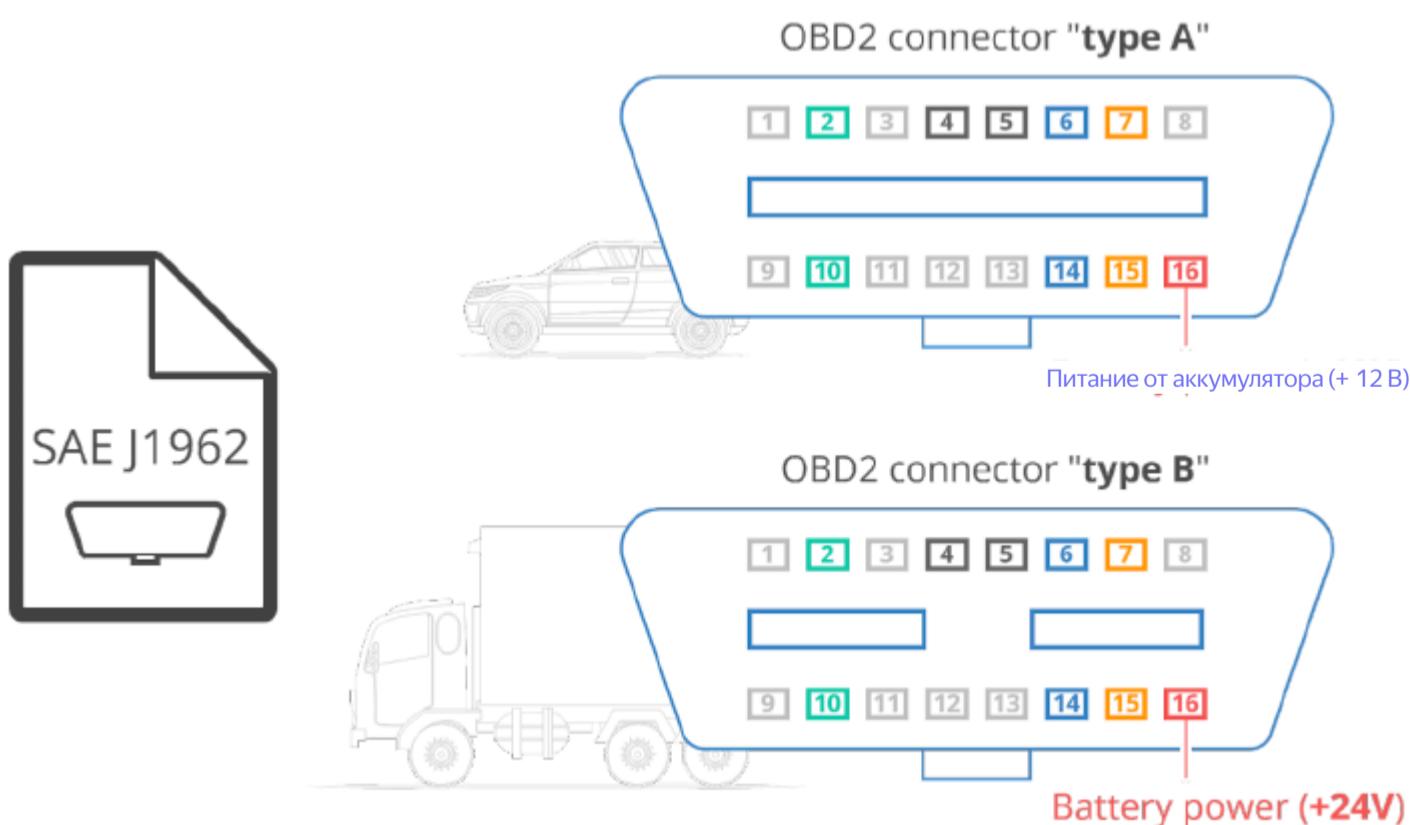
Вывод 16 обеспечивает питание аккумулятора (часто при выключенном зажигании).

Вывод OBD2 зависит от протокола связи

Наиболее распространенным протоколом является CAN (через ISO 15765), что означает, что обычно будут подключены контакты 6 (CAN-H) и 14 (CAN-L) подключен

Разъем OBD2 - тип А или В

На практике вы можете встретить разъем OBD2 как типа А, так и типа В. Обычно тип А встречается в автомобилях, в то время как тип В распространен в автомобилях средней и тяжелой грузоподъемности.



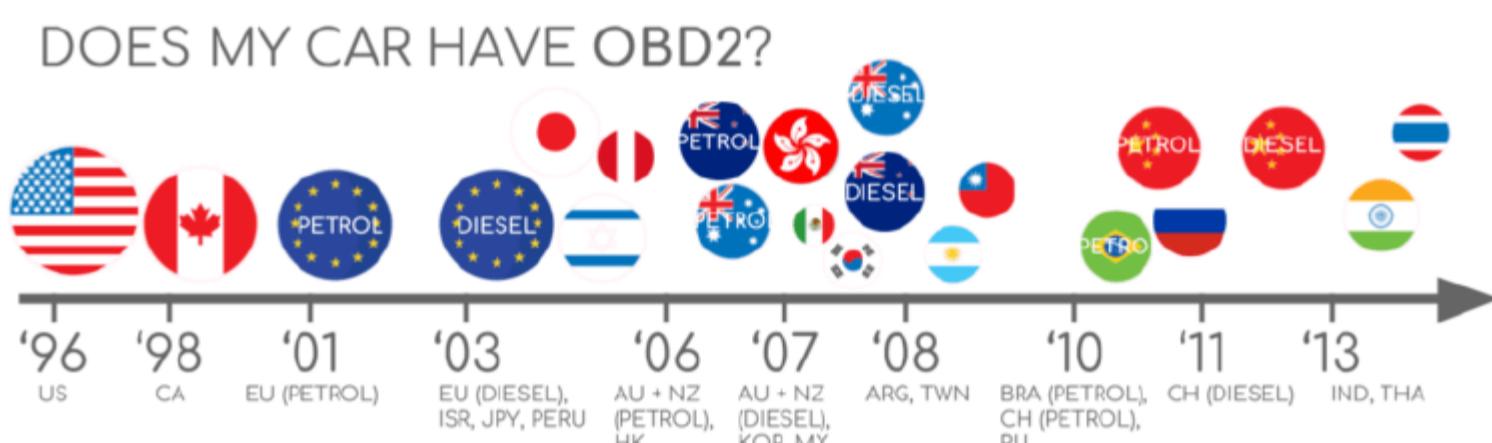
Различия в разъемах типа А и В

Как видно из иллюстрации, два типа имеют схожие распиновки OBD2, но обеспечивают два разных источника питания выходы (12 В для типа А и 24 В для типа В). Часто скорость передачи данных также будет отличаться, поскольку автомобили обычно используют 500К, в то время как в большинстве автомобилей большой грузоподъемности используется 250К (в последнее время появилась поддержка 500К).

Чтобы легче было физически различать два типа разъемов OBD2, обратите внимание, что разъем OBD2 типа В имеет прерывистый паз посередине. В результате кабель-адаптер OBD2 типа В будет совместим как с типами А, так и с В, в то время как кабель типа А не будет вставляться в гнездо типа В.

Есть ли в моей машине OBD2?

Вкратце: возможно! Почти все новые автомобили поддерживают OBD2, и большинство работает на CAN (ISO 15765). Что касается старых автомобилей, имейте в виду, что даже при наличии 16-контактного разъема OBD2 он все равно может не поддерживать OBD2. Один из способов определить соответствие - это определить где и когда он был куплен новым:



Связь между OBD2 и CAN bus

Встроенная диагностика, OBD2, представляет собой "протокол более высокого уровня" (как язык). CAN - это метод связи (как телефон). В частности, стандарт OBD2 определяет разъем OBD2, включая набор из пяти протоколов, по которым он может работать (см. ниже). Кроме того, с 2008 года CAN bus (ISO 15765) является обязательным протоколом для OBD2 во всех автомобилях, продаваемых в США.

Что такое стандарт ISO 15765?

ISO 15765 относится к набору ограничений, применяемых к CAN стандарт (который сам по себе определен в ISO 11898). Можно скажем, что ISO 15765 похож на "консервную банку для автомобилей". В частности, ISO 15765-4 описывает физический уровень канала передачи данных и сетевые уровни, стремящиеся стандартизировать шину CAN интерфейс для внешнего испытательного оборудования.

ISO 15765-2, в свою очередь, описывает транспортный уровень (ISO TP) для отправки кадров CAN с полезной нагрузкой, превышающей 8 байт. Этот вспомогательный стандарт также иногда называют Диагностическая связь по CAN (или DoCAN). Смотрите также иллюстрация 7-уровневой модели OSI. OBD2 также можно сравнить с другими протоколами более высокого уровня (например, J1939, [CANopen](#)).

7-слойная модель OSI



Пять протоколов OBD2

Как объяснялось выше, шина CAN сегодня служит основой для связи OBD2 в подавляющем большинстве автомобилей через ISO 15765. Однако, если вы осматриваете более старый автомобиль (до 2008 года), полезно знать остальные четыре протокола, которые были использованы в качестве основы для OBD2. Обратите также внимание на распиновки, по которым можно определить, какой протокол может использоваться в вашем автомобиле.

- ISO 15765 (CAN bus): обязательный для легковых автомобилей США с 2008 года и сегодня используется в подавляющем большинстве автомобилей.

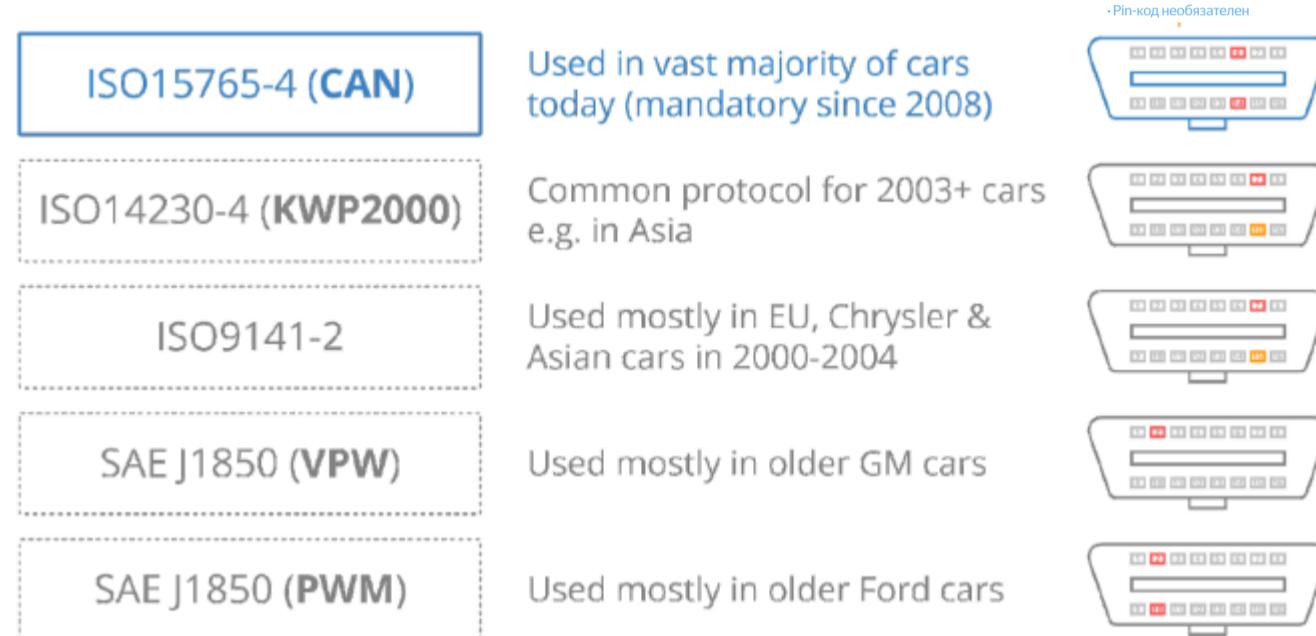
ISO14230-4 (KWP2000): ключевое слово Protocol 2000 было общим протоколом для автомобилей 2003+ года выпуска, например, в Азии

ISO9141-2: использовался в автомобилях ЕС, Chrysler и Азии в 2000-04 годах.

SAE J1850 (VPW): используется в основном в старых автомобилях GM

SAE J1850 (PWM): используется в основном в старых автомобилях Ford

Пять сигнальных протоколов, совместимых с OBD2

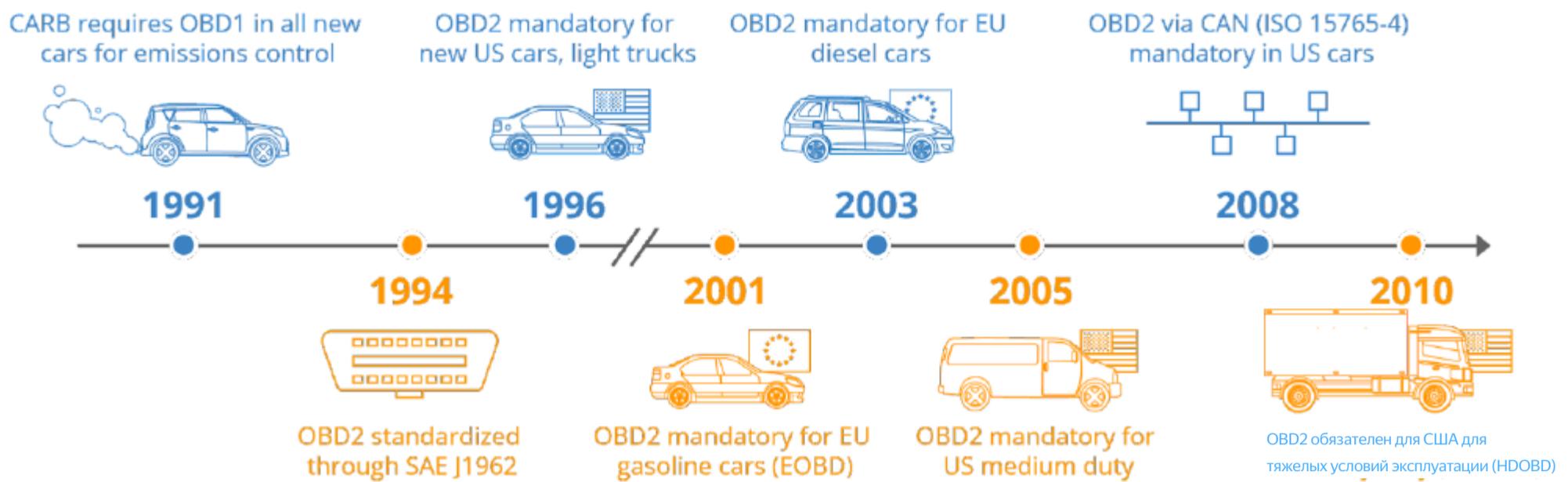


История и будущее OBD2

История

OBD2 происходит из Калифорнии, где Калифорнийский совет по воздушным ресурсам (CARB) требовал наличия OBD во всех новых автомобилях из 1991+ для целей контроля выбросов. Стандарт OBD2 был рекомендован Обществом инженеров автомобильной промышленности (SAE) и стандартизировал коды неисправностей и разъем OBD у всех производителей (SAE J1962). С этого момента стандарт OBD2 внедрялся поэтапно:

- 1996: OBD2 стал обязательным в США для легковых автомобилей / легких грузовиков
- 2001: Требуется в ЕС для бензиновых автомобилей
- 2003: Требуется в ЕС также для дизельных автомобилей (EOBD)
- 2005: OBD2 был необходим в США для автомобилей средней грузоподъемности
- 2008: Автомобили в США должны использовать ISO 15765-4 (CAN) в качестве основы OBD2
- 2010: Наконец, OBD2 потребовался в автомобилях большой грузоподъемности в США



Будущее

OBD2 никуда не денется - но в какой форме? Два потенциальных маршрута могут радикально изменить OBD2:

OBD3 / OBD-III - беспроводное тестирование выбросов

В современном мире подключенных автомобилей тесты OBD2 могут показаться громоздкими: ручные проверки контроля выбросов отнимают много времени и стоят дорого. Решение? OBD3 - добавление телематики ко всем автомобилям. По сути, OBD3 добавляет небольшое радио транспондер (например, для оплаты проезда на мосту) ко всем автомобилям. Используя это, идентификационный номер автомобиля (VIN) и коды неисправностей могут быть отправлены через Wi-Fi на центральный сервер для проверки.

Многие устройства сегодня уже облегчают передачу данных CAN или OBD2 через Wi-Fi / сотовую связь - например, CANedge2 WiFi CAN регистратор. Это экономит средства и удобно, но также является политической проблемой из-за проблем с надзором.

Устранение сторонних сервисов OBD2

Протокол OBD2 изначально был разработан для стационарного контроля выбросов. Тем не менее, сегодня OBD2 широко используется для генерации данных в режиме реального времени сторонними компаниями - с помощью ключей OBD2, CAN-регистраторов и т.д. Однако немецкая автомобильная промышленность [хотите это изменить](#):

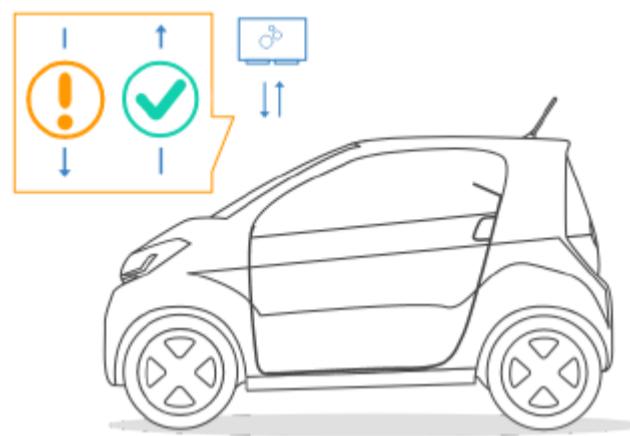
"OBD был разработан для обслуживания автомобилей в ремонтных мастерских. Никоим образом не предполагалось, что он позволит третьим сторонам создавать основанная на данных экономия при доступе через этот интерфейс"

- Кристоф Гроте, старший вице-президент BMW по электронике (2017) Предлагается "отключить" функциональность OBD2 во время вождения и вместо этого собирать данные на центральном сервере. Это фактически дало бы производителям контроль над автомобильными "большими данными". Аргументация основана на принципах безопасности (например, устранение риска взлома автомобиля), хотя многие рассматривают это как коммерческий ход. Станет ли это реальной тенденцией, неизвестно будет видно - но это может действительно подорвать рынок сторонних сервисов OBD2.

Идентификаторы параметров OBD2 (PID)

Почему вас должны волновать данные OBD2? Механикам, очевидно, не все равно о кодах безопасности OBD2 (возможно, у вас тоже есть), в то время как регулирующим органам OBD2 нужен для контроля выбросов. Но протокол OBD2 также поддерживает широкий диапазон стандартных идентификаторов параметров (PID), которые могут регистрироваться в большинстве автомобилей.

Это означает, что вы можете легко получать удобочитаемые данные OBD2 от вашего автомобиля по скорости, оборотам в минуту, положению дроссельной заслонки и многому другому. Другими словами, OBD2 позволяет легко анализировать данные с вашего автомобиля - в отличие от к специфичным для производителя запатентованным данным raw CAN.



Декодирование данных OBD2 по сравнению с данными шины CAN

В принципе, записать необработанные кадры CAN с вашего автомобиля просто. Например, если вы подключите регистратор CAN к разъему OBD2, вы начнете записывать данные, передаваемые по шине CAN, "из коробки". Однако необработанные сообщения CAN должны быть декодированы с помощью базы данных правил преобразования (DBC) и подходящего программного обеспечения CAN, поддерживающего декодирование DBC (например, asammfd). Проблема в том, что эти файлы CAN DBC, как правило, являются проприетарными, что делает необработанные данные CAN нечитаемыми если только вы не производитель автомобилей.

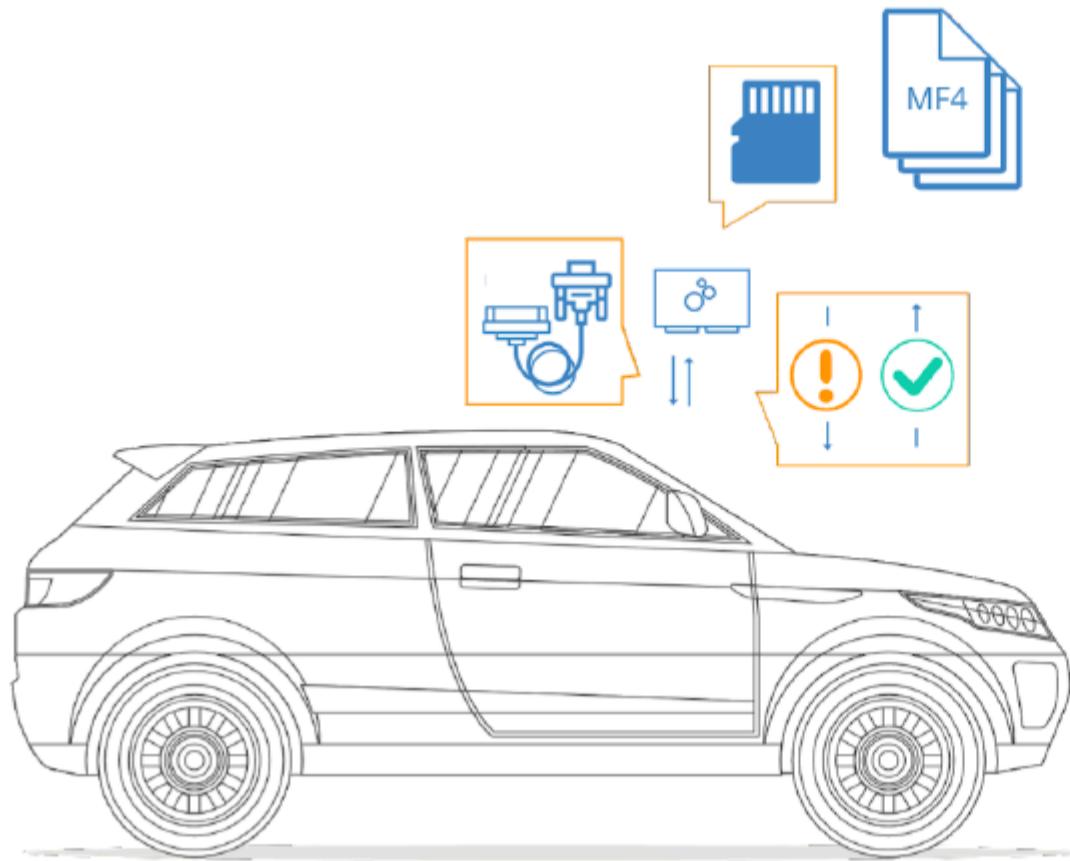
Автомобильные хакеры могут попытаться перепроектировать правила, хотя это может быть сложно. Однако CAN по-прежнему остается единственным методом для получения "полного доступа" к данным вашего автомобиля, в то время как OBD2 предоставляет доступ только к ограниченному набору данных.

Как зарегистрировать данные OBD2?

Регистрация данных OBD2 работает следующим образом:

- Вы подключите регистратор OBD2 к разъему OBD2
- С помощью инструмента, вы отправляете "кадры запроса" через CAN
- Соответствующие ECU отправляют "кадры ответа" через CAN
- Декодируют необработанные ответы OBD2, например, через OBD2 DBC

Другими словами, регистратор CAN, способный передавать пользовательские кадры CAN, также может использоваться в качестве регистратора OBD2. Обратите внимание, что автомобили различаются в зависимости от модели / года выпуска по тому, какие PID OBD2 они поддерживают. Подробнее смотрите в нашем руководстве по регистратору данных OBD2.



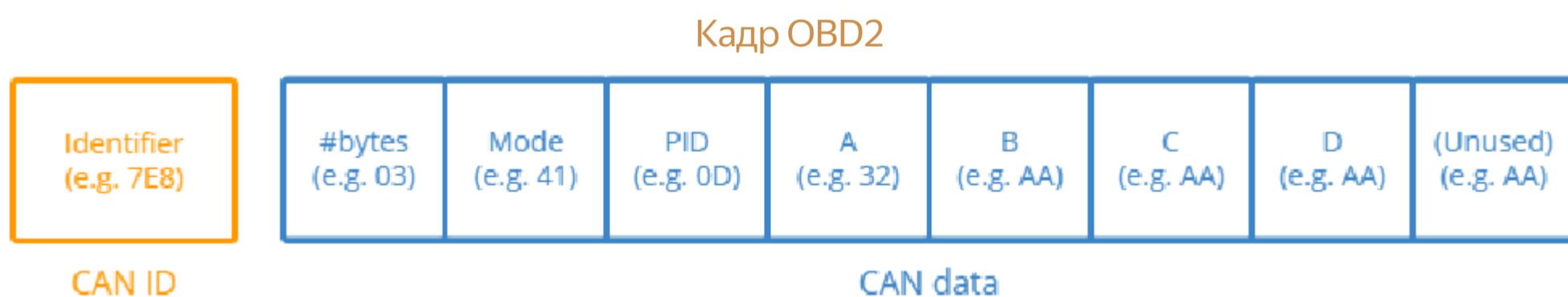
124	22	28	24	00	19	00	1A	FF
2BA	CF	FF	FF	F3	00			
AA2	10	7D	82	BD	12	00	F4	82
15B	AA	2F	F1	13	22	91	AB	
7E8	03	41	0D	32	AA	AA	AA	AA
A21	FF	FF	FF	12	FF	81	43	22

Регистратор данных CANedge OBD2

CANedge позволяет легко записывать данные OBD2 на SD-карту объемом 8-32 ГБ. Просто укажите, какие PID OBD2 вы хотите запросить, затем подключите его к своему автомобилю через адаптер OBD2, чтобы начать регистрацию. Обработайте данные с помощью бесплатного программного обеспечения / API и нашего OBD2 DBC.

Исходные данные рамки OBD2

Чтобы начать запись данных OBD2, полезно понять основы структуры необработанных сообщений OBD2. В упрощенно сообщение OBD2 состоит из идентификатора и данных. Далее данные разделяются на режим, PID и данные байты (A, B, C, D), как показано ниже.



Поля сообщений OBD2, описанные ниже.

Идентификатор: Для сообщений OBD2 идентификатор является стандартным 11-разрядным и используется для различия "сообщений запроса" (ID 7DF) и "сообщений ответа" (ID 7E8 - 7EF). Обратите внимание, что 7E8 обычно находится там, где находится основной двигатель или блок управления.

Длина: это просто отражает длину в количестве байтов оставшихся данных (от 03 до 06). Для транспортного средства Например, скорость равна 02 для запроса (поскольку следуют только 01 и 0D), в то время как для ответа это 03 как 41, Следуют 0D и 32.

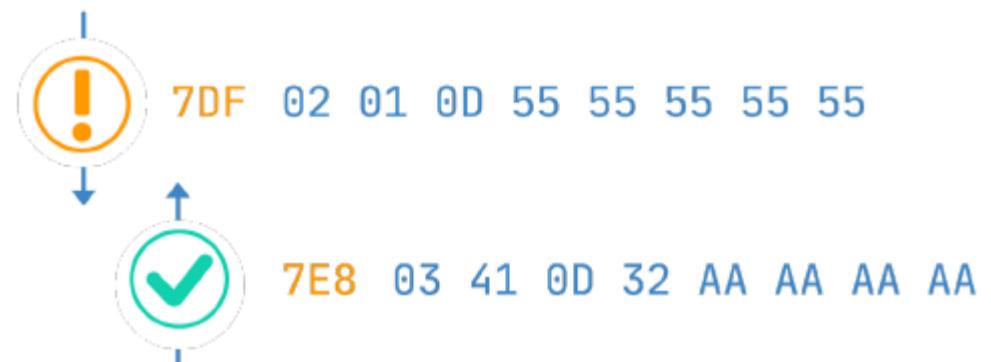
Режим: Для запросов это будет между 01-0A. Для ответов 0 заменяется на 4 (т.е. 41, 42, ..., 4A). Там существует 10 режимов, описанных в стандарте SAE J1979 OBD2. Режим 1 отображает текущие данные и используется, например, для просмотра скорости автомобиля, оборотов в минуту и т.д. В режиме реального времени. Другие режимы используются, например, для отображения или очистки сохраненной диагностики коды неисправностей и отображения данных стоп-кадра.

PID: Для каждого режима существует список стандартных PID OBD2 - например, в режиме 01 PID 0D - это скорость транспортного средства. Для полного списка, ознакомьтесь с нашим обзором PID OBD2. У каждого PID есть описание, а у некоторых указаны минимальные / максимальные значения и формула преобразования. Формула для скорости - это, например, просто A, означающая, что байт данных A (который находится в шестнадцатеричном формате) равен преобразуется в десятичную дробь, чтобы получить преобразованное значение км / ч (т. е. 32 становится 50 км / ч выше). Например, для оборотов в минуту (PID 0C), формула имеет вид $(256 * A+B) / 4$

A, B, C, D: Это байты данных в шестнадцатеричном формате, которые необходимо преобразовать в десятичную форму перед их использованием в вычислениях по формуле PID. Обратите внимание, что последний байт данных (после Dh) не используется.

Пример запроса / ответа OBD2

Пример запроса / ответа МОЖЕТ содержать сообщение для PID "Скорость транспортного средства" со значением 50 км / ч можно увидеть на иллюстрации. Обратите особое внимание на то, как формула для OBD2 PID 0D (скорость транспортного средства) просто включает в себя 4-й байт (0x32) и преобразование его в десятичную форму (50).



Расширенный запрос / ответ OBD2 PID

В некоторых транспортных средствах (например, фургонах и автомобилях малой / средней / тяжелой грузоподъемности) вы можете обнаружить, что в необработанных данных CAN используются расширенные 29-разрядные идентификаторы CAN вместо 11-разрядных идентификаторов CAN.

В этом случае, обычно вам нужно будет изменить PID-запросы OBD2, чтобы использовать идентификатор CAN 18DB33F1 вместо 7DF. структура полезной нагрузки данных остается идентичной примерам для 11-битных идентификаторов CAN.

Если транспортное средство отвечает на запросы, вы обычно увидите ответы с идентификаторами CAN от 18DAF100 до 18DAF1FF (на практике, обычно 18DAF110 и 18DAF11E). Идентификатор ответа также иногда отображается в форме 'J1939 PGN', в частности, PGN 0xDA00 (55808), который в стандарте J1939-71 помечен как "Зарезервирован для ISO 15765-2".

Мы предоставляем файл OBD2 DBC как для 11-разрядных, так и для 29-разрядных ответов, что обеспечивает простое декодирование данных в большинстве Программных средств CAN.

10 служб OBD2 (они же режимы)

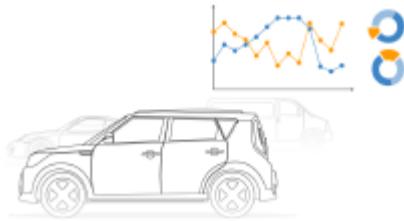
Существует 10 диагностических служб OBD2 (или режимов), как описано в стандарте SAE J1979 OBD2. Режим 1 показывает текущие данные и используется для просмотра в режиме реального времени такие параметры, как скорость автомобиля, обороты в минуту, положение дроссельной заслонки и т.д. Другие режимы, например, используются для отображения / очистки диагностики коды неисправностей (DTCS) и отображают данные стоп-кадра. Производителям не обязательно поддерживать все диагностические службы они могут поддерживать режимы, выходящие за рамки этих 10 услуг (т.е. услуги OBD2, специфичные для конкретного производителя).

Службы/режимы диагностики OBD2 (SAE J1979)

- | | |
|----|--|
| 01 | Отображать текущие данные (например, данные в режиме реального времени) |
| 02 | Отображать данные стоп-кадра (как указано выше, но во время стоп-кадра) |
| 03 | Отображать сохраненные диагностические коды неисправностей (DTCS) |
| 04 | Очистить коды неисправностей и сохраненные значения |
| 05 | Результаты тестов датчиков кислорода (не только CAN) |
| 06 | Результаты тестов для мониторинга системы (и датчиков кислорода для CAN) |
| 07 | Показывают ожидающие проверки коды неисправности |
| 08 | Контролируют работу бортовой системы |
| 09 | Запрашивают информацию о транспортном средстве (например, VIN) |
| 0A | Постоянные коды неисправностей (также известные как очищенные коды неисправностей) |

Регистрация данных OBD2 - примеры вариантов использования

Данные OBD2 от легковых автомобилей и легких грузовиков можно использовать в различных вариантах использования:



Регистрация данных от автомобилей

Данные OBD2 с автомобилей могут например, использоваться для снижения затрат на топливо, улучшения вождения, тестирования прототипов запчастей и страхования

[Узнать больше](#)



Диагностика автомобиля в режиме реального времени

Интерфейсы OBD2 могут использоваться для потоковой передачи удобочитаемых данных OBD2 в режиме реального времени, например, для диагностики проблем с транспортным средством

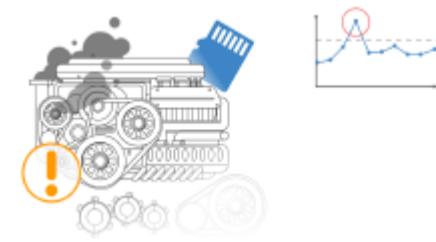
[Узнать больше](#)



Профилактическое обслуживание

Легковые автомобили и легкие грузовики можно контролировать с помощью IoT OBD2 регистраторы в облаке для прогнозировать и избегать сбои

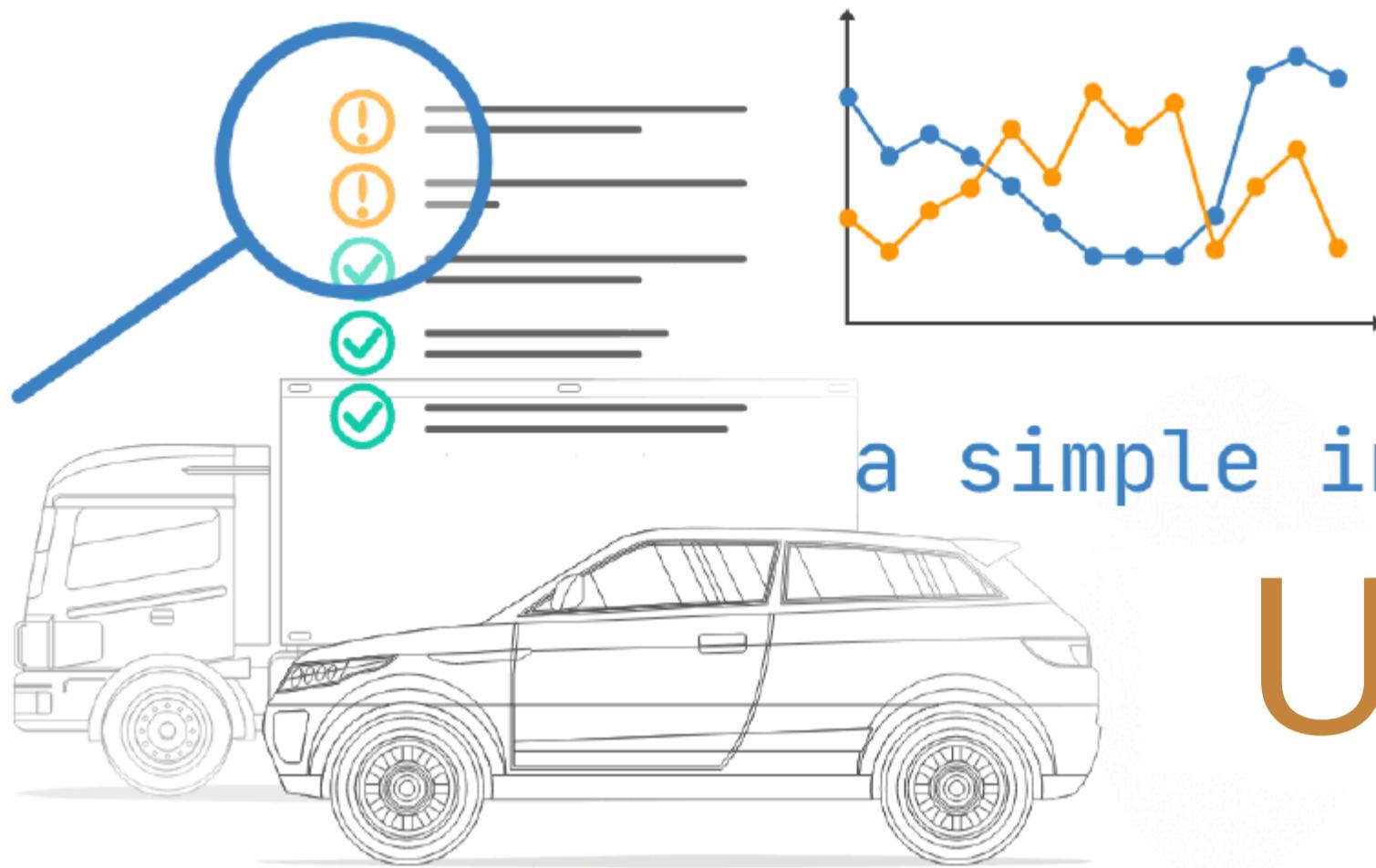
[Узнать больше](#)



Автомобильный регистратор черного ящика

Регистратор OBD2 может служить в качестве "черного ящика" для транспортных средств или оборудования, обеспечивающего данные, например, для споры или диагностика

[Учиться Еще](#)



Объяснение UDS (унифицированные диагностические службы)

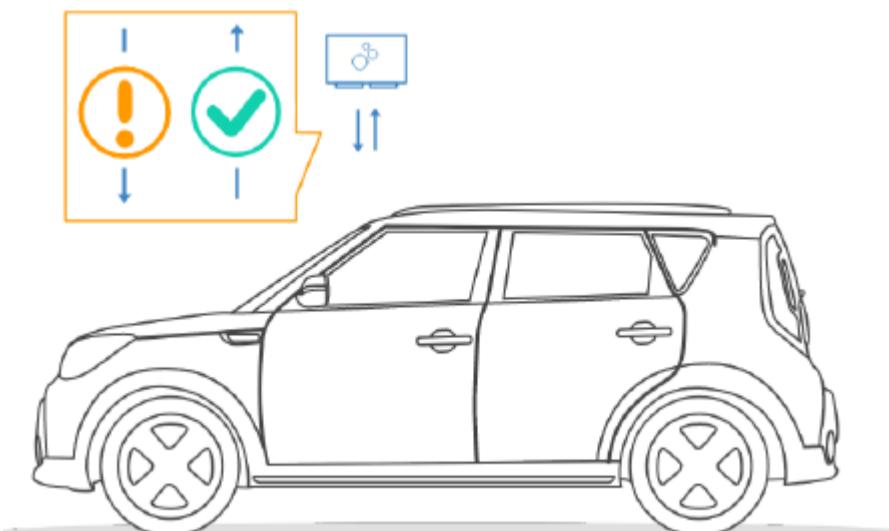
В этом практическом руководстве мы знакомим с основами UDS с акцентом на UDS на шине CAN (UDSonCAN) и диагностике через CAN (DoCAN). Мы также представляем протокол ISO-TP и объясняем разницу между UDS, OBD2, WWH-OBD и OBDOnUDS. Наконец, мы объясним, как запрашивать, записывать и декодировать сообщения UDS - с практическими примерами регистрации EV Состояние заряда и идентификационный номер транспортного средства (VIN).

Что такое протокол UDS?

Унифицированные диагностические службы (UDS) - это протокол связи, используемый в автомобильных электронных блоках управления (ECU) для обеспечения возможности диагностики, обновления встроенного программного обеспечения, рутинного тестирования и многое другое.

Протокол UDS (ISO 14229) стандартизирован как производителями, так и стандартами (такими как CAN, KWP 2000, Ethernet, LIN). Кроме того, UDS сегодня используется в ECU всеми производителями оригинального оборудования уровня 1 (OEM-производители).

На практике обмен данными UDS осуществляется во взаимодействии клиент-сервер, при этом клиент является инструментом тестирования, а сервер является блоком управления автомобилем. Например, вы можете подключить интерфейс шины CAN к разъему OBD2 автомобиля и отправлять Запросы UDS в автомобиль. Предполагая, что целевой блок управления поддерживает службы UDS, он будет реагировать соответствующим образом.



В свою очередь, это обеспечивает различные варианты использования:

Считывание / очистка диагностических кодов неисправностей (DTC) для устранения неполадок автомобиля

Извлечение значений параметров, таких как температура, уровень заряда, VIN и т. Д

Инициируйте сеансы диагностики, например, для тестирования критически важных функций безопасности

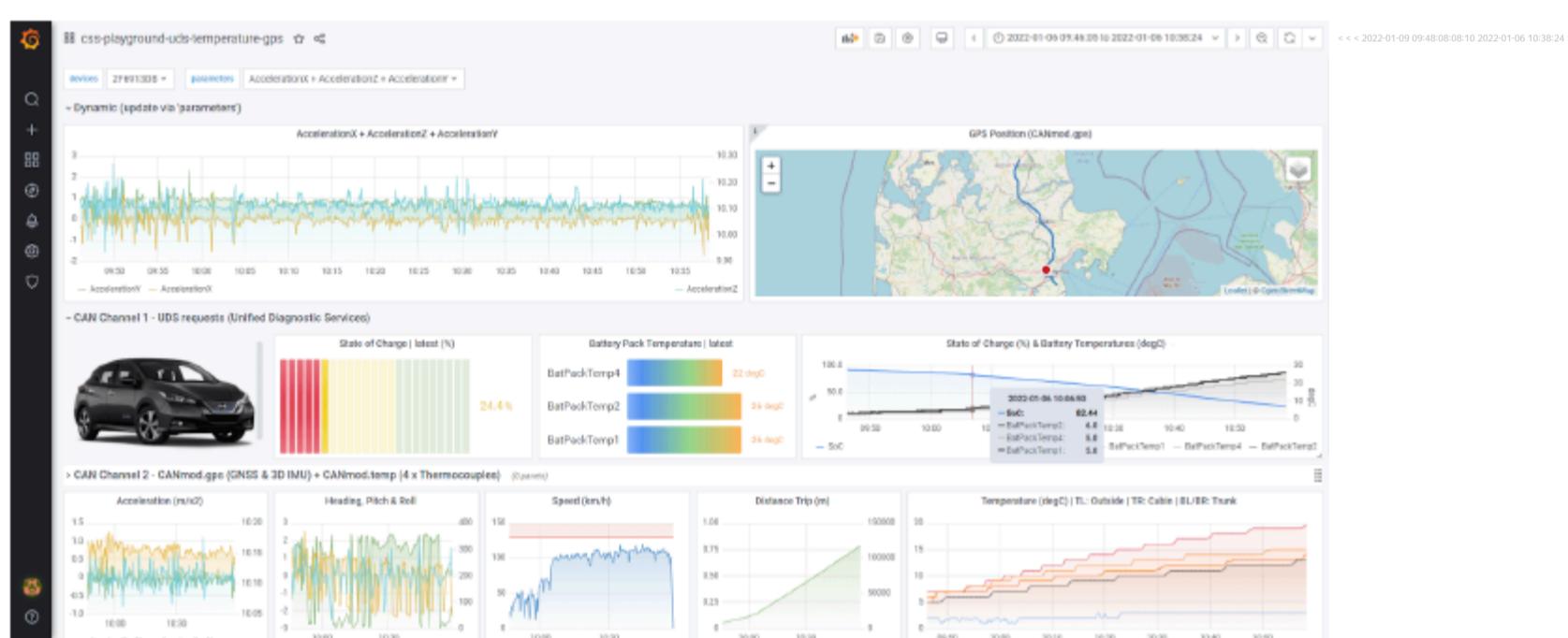
Измените поведение ЭБУ с помощью сброса настроек, прошивки встроенного программного обеспечения и модификации настроек.

UDS часто называют "расширенной диагностикой производителя автомобиля" или "расширенной диагностикой" - подробнее об этом ниже.

Пример: Nissan Leaf SoC%

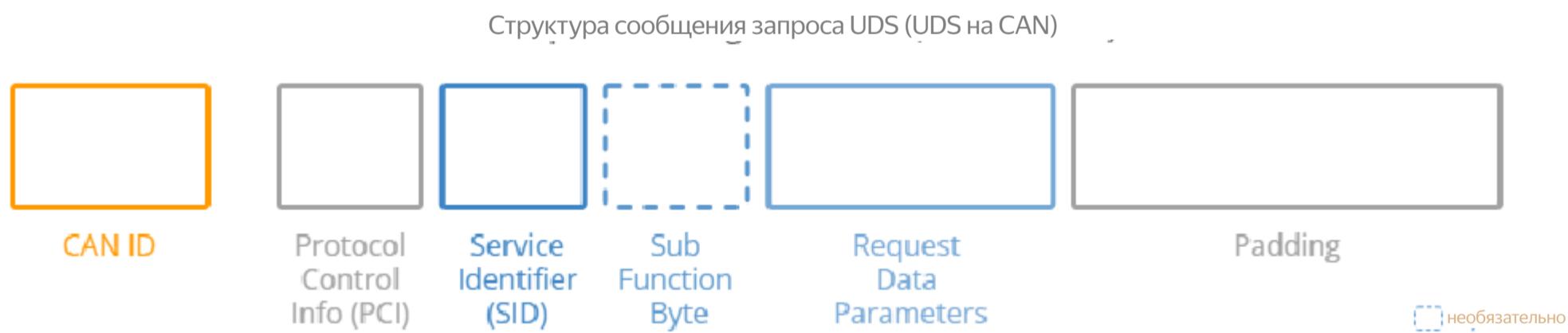
UDS и CAN ISO-TP являются сложными темами. В качестве мотивации мы провели тематическое исследование, чтобы показать, как это может быть полезно. В частности, мы используем CANedge2 для запроса данных о состоянии заряда (SoC%) и температуре аккумулятора от Nissan Leaf EV. В этом примере мы также добавляем CANmod.gps и CANmod.temp для добавления данных GNSS, IMU и температуры.

Многокадровые ответы ISO-TP и сигналы CANmod декодируются DBC с помощью нашего Python API и записываются в базу данных для [визуализации на информационных панелях Grafana](#). Ознакомьтесь с этим: [тематическое исследование playground](#)



Структура сообщений UDS

UDS - это протокол, основанный на запросах. На рисунке мы обрисовали в общих чертах пример фрейма запроса UDS (с использованием шины CAN):



Диагностический запрос UDS на CAN содержит различные поля, которые мы подробно описываем ниже:

Информация об управлении протоколом (PCI)

Поле PCI само по себе не связано с запросом UDS, но требуется для диагностических запросов UDS, выполняемых по шине CAN. Короче говоря, поле PCI может иметь длину 1-3 байта и содержать информацию, относящуюся к передаче сообщений, которые не умещаются в одном кадре CAN. Мы подробнее рассмотрим это в разделе, посвященном транспортному протоколу шины CAN (ISO-TP).

Идентификатор службы UDS (SID)

Описанные выше варианты использования относятся к различным службам UDS. Когда вы хотите воспользоваться определенной услугой UDS, UDS сообщение запроса должно содержать идентификатор службы UDS (SID) в полезной нагрузке данных. Обратите внимание, что идентификаторы разделены между SID запроса (например, 0x22) и SID ответа (например, 0x62). Как и в OBD2, SID ответа обычно добавляют 0x40 к SID запроса. Смотрите также обзор всех стандартизованных служб UDS и SIDS. В основном мы сосредоточимся на службе UDS 0x22 в эта статья, которая используется для считывания данных (например, скорости, SoC, температуры, VIN) с ЭБУ.

Идентификаторы службы UDS (SID)			
	UDS SID	UDS SID	Обслуживание
	(запрос)	(ответ)	
18	0x10	0x50	Контроль сеанса диагностики
	0x11	0x51	Сброс ECU
	0x27	0x67	Доступ безопасности
	0x28	0x68	Контроль связи
	0x29	69x0	Аутентификация
8	0x2E	0x7E	Присутствует тестер
		0xC3	Параметры синхронизации доступа
	0x84	0xC4	Защищенная передача данных
	0x85	0xC5	Управление настройками DTC
	0x86	0xC6	Реакция на событие
	0x87	0xC7	Управление связью
	0x22	0x62	Считывание данных по идентификатору
§	0x23	0x63	Считывание памяти по адресу
	0x24	0x64	Считывание масштабируемых данных по идентификатору
	0x2A	0x6A	Периодическое считывание данных по идентификатору
9	0x2C	0x6C	Динамически Определять идентификатор данных
8	0x2E	0x6E	Запись данных по идентификатору
	0x3D	0x7D	Запись памяти по адресу
Коды неисправностей 6	0x14	0x54	Очистить диагностическую информацию
	0x19	0x59	Прочитать информацию DTC
	0x2F	0x6F	Управление вводом-выводом по идентификатору
	0x31	0x71	Обычное управление
	0x34	0x74	Просьба Скачать
	0x35	0x75	Загрузка запроса
	0x36	0x76	Передача данных
	0x37	0x77	Завершите передачу запроса
	0x38	0x78	Запросить передачу файла
	0x7F		Отрицательный ответ

UDS SIDS по сравнению с другими диагностическими службами

Стандартизованные службы UDS, показанные выше, на практике являются подмножеством более широкого набора диагностических служб - см. Ниже Обзор. Обратите внимание, что идентификаторы SID от 0x00 до 0x0F зарезервированы для разрешенных служб OBD (подробнее об этом позже).

>Diagnostic service identifiers - overview (0x00 - 0xFF)

<Идентификатор службы (SID)	Тип службы	Дополнительные сведения
0x00 - 0x0F	Запросы на обслуживание OBD	ISO 15031-5
0x00 - 0x3E	ISO 14229 (запросы)	ISO 14229
0x3F	Неприменимо	Зарезервировано
0x40 - 0x4F	Ответы службы OBD	ISO 15031-5
0x50 - 0x7E	ISO 14229 (ответы)	ISO 14229
0x7F	SID отрицательного ответа	ISO 14229
0x80	Неприменимо	ISO 14229 (зарезервирован)
0x81 - 0x82	Неприменимо	ISO 14229 (зарезервирован)
0x83 - 0x88	ISO 14229 (запросы)	ISO 14229
0x89 - 0x9F	Запросы на обслуживание	Зарезервированы
0xA0 - 0xB9	Запросы на обслуживание	Определяются OEM-производителем транспортн
0xBA - 0xBE	Запросы на обслуживание	Определяются OEM-производителем систем
0xBF	Не применимо	Зарезервировано
0xC0	Неприменимо	ISO 14229 (зарезервировано)
0xC1 - 0xC2	Неприменимо	ISO 14229 (зарезервировано)
0xC3 - 0xC8	ISO 14229 (ответы)	ISO 14229
0xC9 - 0xDF	Сервисные ответы	Зарезервированы
0xE0 - 0xF9	Сервисные ответы	Определяются производителем автомобиля
0xFA - 0xFE	Сервисные ответы	Определяются производителем систем
0xFF	Неприменимо	Зарезервировано

>UDS request SIDs

<>UDS positive response SIDs

<>UDS negative response SID

<Защита UDS посредством сеансового контроля (аутентификации)

Как очевидно, UDS обеспечивает полный контроль над ЭБУ автомобиля. По соображениям безопасности критически важные службы UDS поэтому ограничены процессом аутентификации. По сути, ECU отправляет "исходное значение" тестировщику, который, в свою очередь, должен создать "ключ" для получения доступа к критически важным службам безопасности. Чтобы сохранить этот доступ, тестировщику необходимо периодически отправлять сообщение "тестировщик присутствует".

На практике этот процесс аутентификации позволяет производителям транспортных средств ограничивать доступ к UDS для пользователей вторичного рынка и гарантировать, что только назначенные инструменты смогут использовать критически важные для безопасности сервисы UDS.

Обратите внимание, что переключение между уровнями аутентификации осуществляется с помощью управления сессиями диагностики, которое является одной из Доступных сервисов UDS. Производители транспортных средств могут решать, какие сеансы поддерживаются, хотя они всегда должны поддерживать "сеанс по умолчанию" (т. Е. Который не требует какой-либо аутентификации). С учетом сказанного, они решают, какие службы также поддерживаются в сеансе по умолчанию. Если инструмент тестирования переключается на сеанс, отличный от сеанса по умолчанию, он должен периодически отправлять сообщение "тестер присутствует", чтобы избежать возврата к сеансу по умолчанию.

Байт вспомогательной функции UDS

Байт вспомогательной функции используется в некоторых кадрах запроса UDS, как описано ниже. Однако обратите внимание, что в некоторых службах UDS, как и 0x22, байт вспомогательной функции не используется. Как правило, когда запрос отправляется в ECU, ECU может ответить положительно или отрицательно. В случае положительного ответа тестировщик может захотеть подавить ответ (поскольку он может быть неактуальным). Это делается путем установки 1-го бита равным 1 в байте вспомогательной функции. Отрицательные ответы не могут быть подавлены.

Оставшиеся 7 битов могут использоваться для определения до 128 значений вспомогательной функции. Например, при считывании информации DTC через SID 0x19 (считывание диагностической информации), вспомогательная функция может использоваться для управления типом отчета -смотрите Также таблицу ниже.

>UDS services - sub function types

<SID UDS (запрос)	UDS SID (ответ)	Обслуживание	Типы подфункций
0x10	0x50	Управление сеансом диагностики	Тип сеанса диагностики
0x11	0x51	Сброс ECU	Тип сброса
0x27	0x67	Доступ по безопасности	Тип доступа по безопасности
0x28	0x68	Управление связью	Тип управления
0x3E	0x7E	Присутствует тестер	"Нулевая вспомогательная функция"
0x83	0xC3	Параметры синхронизации доступа	Тип доступа к параметру синхронизации
0x85	0xC5	Настройки управляющего DTC	Тип настройки DTC
0x86	0xC6	Реакция на событие	Тип события
0x87	0xC7	Управление связью	Тип управления связью
0x2C	0x6C	Динамически определяемый идентификатор данных	Тип определения
0x19	0x59	Считывание информации DTC	Тип отчета
0x31	0x71	Рутинный контроль	Тип рутинного управления

Пример: вспомогательные функции службы 0x19

Если мы посмотрим конкретно на службу 0x19, мы можем увидеть примеры различных вспомогательных функций ниже:

>UDS service 0x19 - sub function byte values & types			
<UDS SID (запрос)	Идентификатор UDS SID (ответ)	Тип отчета (значение вспомогательной функции)	Сообщить
0x19	0x59	0x01	Номер DTC по маске состояния
		0x02	DTC по маске состояния
		0x03	Идентификация моментального снимка DTC
		0x04	Запись моментального снимка DTC по номеру DTC
		0x05	Сохраненные данные DTC по номеру записи
		0x06	Расширенная запись данных DTC по номеру DTC
		0x07	Номер DTC по записи маски серьезности
		0x08	DTC по записи маски серьезности
		0x09	Информация о серьезности DTC
		0x0A	Поддерживаемый DTC
		0x0B	Первый сбойный DTC
		0x0C	Первый подтвержденный DTC
		0x0D	Последний сбойный DTC
		0x0E	Последний подтвержденный DTC
		0x10	Зеркальное отображение DTC памяти по маске состояния
		0x11	DTC зеркальной памяти по номеру DTC
		0x12	Количество DTC зеркальной памяти по маске состояния
		0x13	Количество выбросов OBD DTC по маске состояния
		0x14	Выбросы OBD DTC по маске состояния
		0x15	Счетчик обнаружения неисправностей DTC
		0x16	DTC с постоянным состоянием
		0x17	Расширенная запись данных DTC по номеру записи
		0x18	Определяемый пользователем DTC памяти по маске состояния
		0x19	Определяемый пользователем снимок DTC памяти по номеру
		0x42	Определяемая пользователем запись DTC памяти по номеру
		0x55	WWH-OBD DTC по записи маски состояния
			WWH-OBD коды неисправностей с постоянным состоянием

UDS "Параметры данных запроса" - вкл. Идентификатор данных (DID)

В большинстве служб запросов UDS, различные типы параметров данных запроса используются для обеспечения дальнейшей конфигурации запроса помимо SID и необязательного байта вспомогательной функции. Здесь мы приводим два примера.

Служба 0x19 (считывание информации DTC) - конфигурация запроса

Например, служба 0x19 позволяет считывать информацию DTC. Запрос UDS для SID 0x19 включает вспомогательную функцию byte - например, 0x02 позволяет считывать коды неисправностей с помощью маски состояния. В этом конкретном случае за байтом вспомогательной функции следует 1-байтовый параметр, называемый маской состояния DTC, для предоставления дополнительной информации о запросе. Аналогично, другие типы вспомогательные функции в 0x19 имеют определенные способы настройки запроса.

Служба 0x22 (чтение данных по идентификатору) - Идентификаторы данных

Другим примером является служба 0x22 (чтение данных по идентификатору). Эта служба использует идентификатор данных (DID), который составляет 2 байта значение от 0 до 65535 (0xFFFF). DID служит идентификатором параметра для обоих запросов /ответов (аналогично как идентификатор параметра, или PID, используется в OBD2).

Например, запрос на чтение данных через один DID в UDS через CAN будет включать поле PCI, службу UDS 0x22 и 2-байтовый DID . В качестве альтернативы, можно запросить данные для дополнительных DID, добавив их после начального DID в запросе. Мы подробнее рассмотрим это в разделе о том, как записывать и декодировать обмен данными UDS.

Идентификаторы данных могут быть частными и известны только производителям оборудования, хотя некоторые DID стандартизированы. Это, например, в случае с WWH-OBD DIDS (подробнее об этом позже) идентификационный номер транспортного средства (VIN) равен 0xF190. Смотрите отдельная таблица для списка стандартизованных DID в UDS.

>UDS - standardized data identifiers (DID)

<UDS DID (идентификатор данных)	Описание
0xF180	Идентификация загрузочного программного обеспечения
0xF181	Идентификация прикладного программного обеспечения
0xF182	Идентификация данных приложения
0xF183	Отпечаток пальца загрузочного программного обеспечения
0xF184	Отпечаток пальца прикладного программного обеспечения
0xF185	Отпечаток пальца данных приложения
0xF186	Активный сеанс диагностики
0xF187	Номер запасной части производителя
0xF188	Номер программного обеспечения ECU производителя
0xF189	Версия программного обеспечения ECU производителя
0xF18A	Идентификатор поставщика системы
0xF18B	Дата изготовления ECU
0xF18C	Серийный номер ECU
0xF18D	Поддерживаемые функциональные блоки
0xF18E	Номер детали в сборе от производителя
0xF190	Идентификационный номер транспортного средства (VIN)
0xF192	Номер оборудования системного поставщика ECU
0xF193	Номер версии оборудования системного поставщика ECU
0xF194	Номер программного обеспечения системного поставщика ECU
0xF195	Номер версии программного обеспечения ECU поставщика системы
0xF196	Правила выхлопа / номер официального утверждения типа
0xF197	Название системы / тип двигателя
0xF198	Код ремонтной мастерской / серийный номер тестера
0xF199	Дата программирования
0xF19D	Дата установки ECU
0xF19E	Файл ODX

Положительные или отрицательные ответы UDS

Когда ECU положительно реагирует на запрос UDS, структура фрейма ответа состоит из аналогичных элементов, что и структура запроса. Например, "положительный" ответ на запрос службы 0x22 будет содержать SID ответа 0x62 (0x22 + 0x40) и 2-байтовый DID, за которым следует фактическая полезная нагрузка данных для запрошенного DID. Как правило, структура положительного Ответное сообщение UDS зависит от сервиса. Однако в некоторых случаях ECU может выдать отрицательный ответ на Запрос UDS - например, если услуга не поддерживается. Отрицательный ответ структурирован следующим образом: пример:

Пример отрицательного ответа UDS (UDS для CAN)



Подробности + Таблица кодов отрицательного ответа

Ниже мы кратко описываем фрейм отрицательного ответа с акцентом на NRC:

- 1-й байт - это поле PCI
- 2-й байт - это код отрицательного ответа SID, 0x7F
- 3-й байт - это SID отклоненного запроса
- 4-й байт - это код отрицательного ответа (NRC)

В отрицательном ответе UDS NRC предоставляет информацию о причине отклонения в соответствии с таблицей ниже.

>UDS SID 0x7F - Negative Response Codes (NRC)

<NRC UDS
Описание
0x10
Общее отклонение
0x11
Служба не поддерживается
0x12
Подфункция не поддерживается
0x13
Недопустимая длина / формат сообщения
0x14
Слишком длинный ответ
0x21
Повторный запрос занят
0x22
Неверные условия
0x24
Ошибка последовательности запроса
0x25
Нет ответа от компонента подсети
0x26
Сбой предотвращает выполнение запрошенного действия
0x31
Запрос вне зоны действия сети
0x33
Отказано в доступе безопасности
0x35
Недопустимый ключ
0x36
Превышено количество попыток
0x37
Требуемое время задержки не истекло
0x70
Загрузка не принята
0x71
Передача данных приостановлена

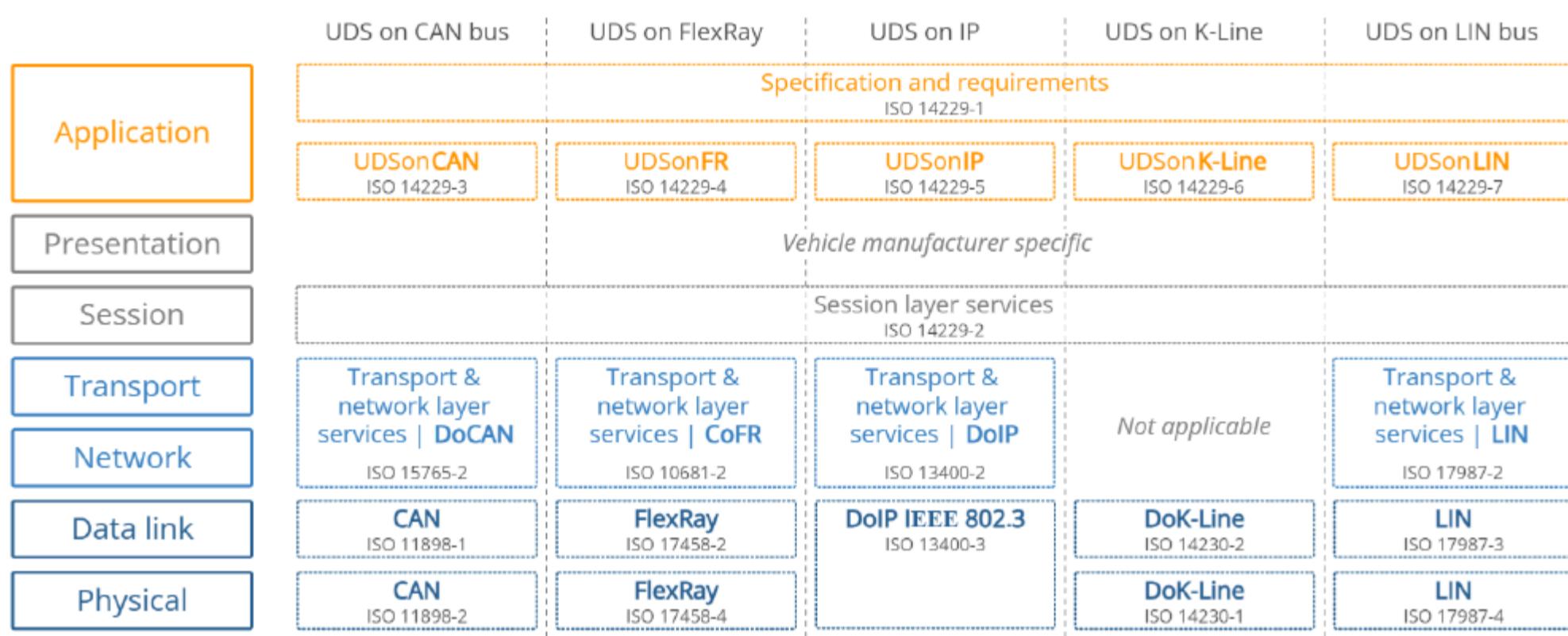
0x72
Сбой программирования
0x73
Неверный счетчик последовательности блоков
0x78
Запрос получен - ожидание ответа
0x7E
Подфункция не поддерживается в активном сеансе.
0x7F
Служба не поддерживается в активном сеансе.
0x81 / 0x82 слишком высокие / низкие обороты в минуту.
0x83 / 0x84 Двигатель запущен / не запущен.
0x85
Слишком низкое время работы двигателя.
0x86 / 0x87 слишком высокая / низкая температура.
0x88 / 0x89 слишком высокая / низкая скорость.
0x8A / 0x8B слишком высокая / низкая педаль газа.
0x8C / 0x8D Диапазон передачи не в нейтральном /уважаемый,
0x8F
Выключатели тормозов не замкнуты,
0x90
Рычаг переключения передач не в стояночном положении,
0x91
Сцепление гидротрансформатора заблокировано,
0x92 / 0x93 слишком высокое / низкое напряжение,
0xF0-0xFE Неверны условия, определенные производителем

UDS против шины CAN: стандарты и модель OSI

Чтобы лучше понять UDS, мы рассмотрим, как это соотносится с шиной CAN и моделью OSI.

Как объясняется в нашем руководстве по шине CAN, локальная сеть контроллера служит основой для связи. В частности, CAN описывается уровнем канала передачи данных и физическим уровнем в модели OSI (согласно ISO 11898). В отличие от CAN, UDS (ISO 14229) является "протоколом более высокого уровня" и использует как сеансовый уровень (5-й), так и прикладной уровень (7-й) в модели OSI, как показано ниже:

7-уровневая модель OSI | Унифицированные диагностические службы (UDS)



Обзор стандартов и концепций UDS

UDS относится к большому количеству стандартов / концепций, что означает, что это может быть немного затруднительно. Чтобы дать вам обзор, ниже приводится высокоуровневое объяснение наиболее важных из них (с акцентом на CAN в качестве основы).

Краткий обзор слоев модели UDS OSI.

Ниже мы приводим краткое описание каждого уровня модели OSI:

Применение: Это описано в стандарте ISO 14229-1 (для различных уровней последовательного канала передачи данных). Далее, отдельный стандарт ISO описывает прикладной уровень UDS для различных протоколов нижнего уровня - например, ISO 14229-3 для CAN bus (он же UDSonCAN)

Презентация: Это зависит от производителя транспортного средства

Сеанс: Это описано в стандарте ISO 14229-2

Транспорт + сеть: для CAN используется стандарт ISO 15765-2 (он же ISO-TP)

Канал передачи данных: В случае CAN это описано в стандарте ISO 11898-1

Физический: в случае CAN это описано в стандарте ISO 11898-2

Как показано, в качестве основы для связи UDS может использоваться множество стандартов, отличных от CAN, включая FlexRay, Ethernet, LIN bus и K-line. В этом руководстве мы сосредоточимся на CAN, который также является наиболее распространенным протоколом нижнего уровня.

ISO 14229-1 (прикладной уровень)

Стандарт ISO 14229-1 описывает требования прикладного уровня к UDS (независимо от того, какой нижний уровень используется протокол). В частности, в нем излагается следующее:

Потоки связи клиент-сервер (запросы, ответы, ...)
Службы UDS (согласно описанному ранее обзору)
Коды положительных ответов и отрицательных ответов (NRCS)
Различные определения (например, коды неисправностей, идентификаторы данных параметров, также известные как DID, ...)

ISO 14229-3 (прикладной уровень для CAN)

Целью 14229-3 является обеспечение возможности внедрения унифицированных диагностических служб (UDS) в зоне управления Сети (CAN), также известные как UDSonCAN. Этот стандарт описывает требования прикладного уровня для UDSonCAN.

Этот стандарт не описывает никаких требований к реализации архитектуры шины CAN для автомобиля. Вместо этого он основное внимание уделяется некоторым дополнительным требованиям / ограничениям для UDS, специфичным для UDSonCAN.

В частности, в 14229-3 описывается, к каким службам UDS предъявляются особые требования CAN. Затронутыми службами UDS являются ResponseOnEvent и ReadDataByPeriodicIdentifier, для которых конкретные требования CAN подробно описаны в 14229-3. Все остальные сервисы UDS реализованы в соответствии с ISO 14229-1 и ISO 14229-2.

Стандарт ISO 14229-3 также описывает набор сопоставлений между ISO 14229-2 и ISO 15765-2 (ISO-TP) и описывает требования, относящиеся к 11-разрядным и 29-разрядным идентификаторам CAN, когда они используются для UDS и законодательно закрепленных OBD в соответствии с ISO 15765-4.

ISO 14229-2 (сеансовый уровень)

Это описывает сеансовый уровень в модели UDS OSI. В частности, он описывает запрос на обслуживание / подтверждение / индикацию примитивы. Они обеспечивают интерфейс для реализации UDS (ISO 14229-1) с любым из коммуникационных протоколов (например, CAN).

ISO 15765-2 (Транспортный + сетевой уровень для CAN)

Для UDS на CAN ISO 15765-2 описывает, как передавать диагностические запросы и ответы. В частности, стандарт описывает, как структурировать кадры CAN для обеспечения передачи многокадровых полезных нагрузок. Поскольку это жизненно важно часть понимания UDS на CAN, мы углубимся в следующий раздел.

ISO 11898 (физический уровень + уровень канала передачи данных для CAN)

Когда UDS основан на шине CAN, физический уровень и уровень канала передачи данных описаны в ISO 11898-1 и ISO 11898-2. Когда UDS основан на CAN, его можно сравнить с протоколами более высокого уровня, такими как J1939, OBD2, CANopen, NMEA 2000 и т.д. Однако, в отличие от этих протоколов, UDS может быть альтернативно основан на других протоколах связи, таких как FlexRay, Ethernet, LIN и т.д.

UDSonCAN против DoCAN

Когда говорят об UDS на базе шины CAN, часто используются два термина: UDSonCAN (UDS на шине CAN) и DoCAN (Диагностика на шине CAN). В некоторых руководствах UDS эти термины используются как взаимозаменяемые, что может привести к путанице.

В стандарте ISO 14229-1 термины используются так, как показано на нашей иллюстрации модели OSI. Другими словами, UDSonCAN используется для обозначения стандарта ISO 14229-3, в то время как DoCAN используется для обозначения стандарта ISO 15765-2, также известного как ISO-TP.

Однако часть путаницы может возникнуть из-за того, что ISO 14229-3 также предоставляет модель OSI, в которой используются оба параметра DoCAN по отношению к ISO 15765-2 и в качестве наложения на слои модели OSI 2-7. В ISO 14229-2 DoCAN упоминается как

протокол связи, на котором реализован UDS (ISO 14229-1). Это соответствует иллюстрации из ISO 14229-3. В этом контексте DoCAN можно рассматривать как более общий термин для обозначения внедрения UDS в CAN, в то время как UDSonCAN, по-видимому, последовательно ссылается только на ISO 14229-3.

ISO 15765-3 против ISO 14229-3

UDS на шине CAN (UDSonCAN) иногда упоминается через ISO 15765-3. Однако в настоящее время этот стандарт устарел и был заменен на ISO 14229-3.

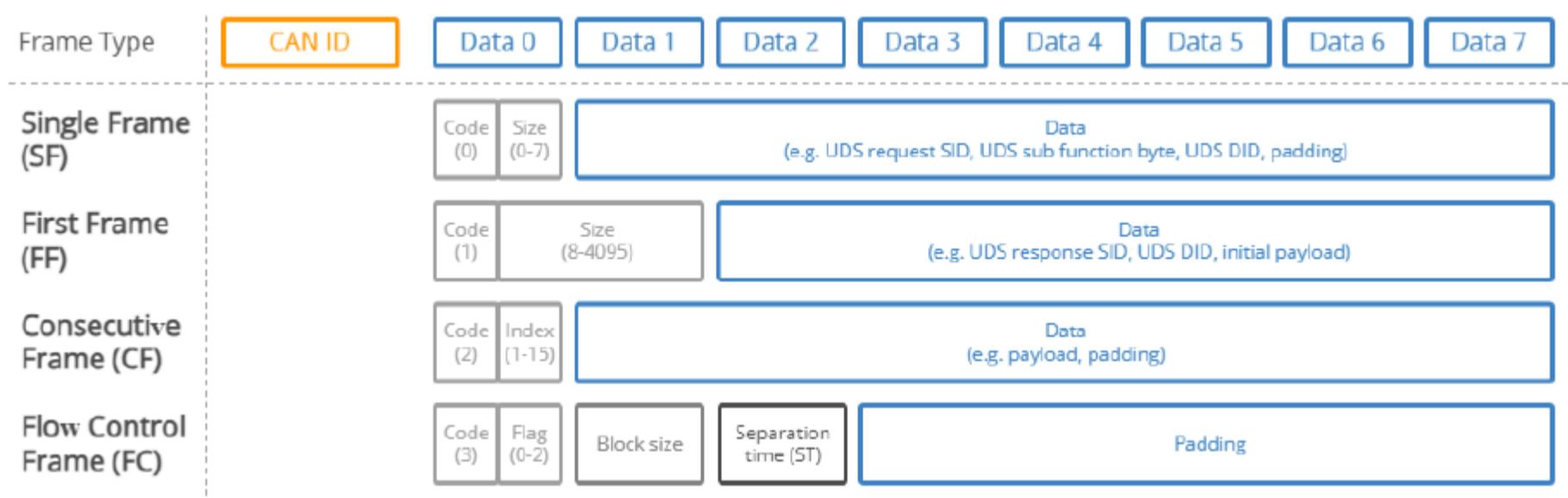
CAN ISO-TP - транспортный протокол (ISO 15765-2)

При реализации диагностики на CAN одной из проблем является размер полезной нагрузки кадра CAN: для классических кадров CAN, это ограничено 8 байтами, а для CAN FD полезная нагрузка ограничена 64 байтами. Диагностика транспортных средств часто требует передачи данных о гораздо более крупных полезных нагрузках.

Стандарт ISO 15765-2 был разработан для решения проблемы больших полезных нагрузок при диагностике транспортных средств на основе CAN. Стандарт определяет транспортный протокол и службы сетевого уровня для использования в транспортных сетях на базе CAN. Наиболее общие варианты использования включают UDS (ISO 14229-1), OBD (SAE J1979, ISO 15031-5) и согласованный во всем мире OBD, также известный как WWH-OBD (ISO 27145).

Стандарт ISO-TP описывает, как передавать полезные данные CAN объемом до 4095 байт посредством сегментации, потока контроля и повторной сборки. ISO-TP определяет конкретные фреймы CAN для обеспечения такой связи, как показано ниже:

Типы фреймов ISO TP (транспортный протокол шины CAN, ISO 15765-2)



Что касается фрейма управления потоком

Фрейм управления потоком используется для "настройки" последующей связи. Его можно сконструировать, как показано ниже.:

ISO TP Flow Control (FC) frame types (CAN Bus Transport Protocol, ISO 15765-2)



Flag 0x00 = Continue To Send
0x01 = Wait
0x02 = Overflow/abort

Block size 0x00 = remaining "frames" to be sent without flow control or delay
> 0x00 = send number of "frames" before waiting for the next flow control frame

Separation time (ST) 0x00 - 0x7F = separation time in milliseconds (0 - 127 ms)
0xF1 - 0xF9 = separation time in even multiples of 100 microseconds (0xF1 = 100 µs, 0xF2 = 200 µs, ... 0xF9 = 900 µs)

Несколько комментариев:

В простейшем случае полезная нагрузка FC может быть установлена на 30 00 00 00 00 00 00 00 00 00 00 00 (все оставшиеся кадры должны быть отправлены без задержки)

В качестве альтернативы, можно принять решение о более детальном контроле связи, например, чередуя команды ожидания и продолжения, а также указание конкретного времени разделения (в миллисекундах) между кадрами

тации к кадрам ISO-TP

Тип кадра ISO-TP можно определить по первому фрагменту первого байта (0x0, 0x1, 0x2, 0x3).

Общий размер кадра может составлять до 4095 байт (0xFFFF), как видно из кадра FF

CF изменяется от 1 до 15 (0xF) и затем сбрасывается, если требуется отправить больше данных

(например, 0x00, 0xAA, ...) используется для обеспечения того, чтобы полезная нагрузка кадра составляла 8 байт в длину

исываем, как работает протокол ISO-TP для однокадровой и многокадровой связи:

Однокадровая связь

Сообщения связи инициируются тестирующим инструментом, отправляющим запрос. Этот фрейм запроса называется (SF). В простейшем случае инструмент тестирования отправляет один кадр для запроса данных от ECU. Ответ может быть содержаться в 7-байтовой полезной нагрузке, ECU предоставляет ответ в виде одного кадра.

Примерного запроса / ответа UDS: считывание данных по идентификатору (UDS на CAN)



Запрос
ответ

Пр
со
EC

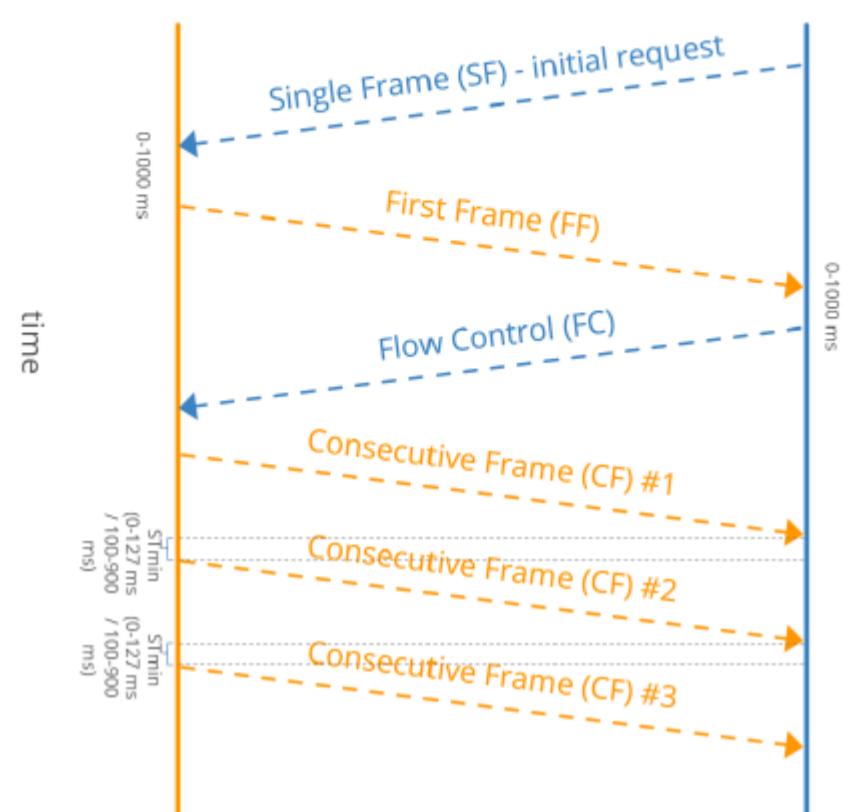
ISO-TP: многокадровая связь

Когда полезная нагрузка превышает 7 байт, ее необходимо разделить через несколько кадров CAN.

Как и прежде, тестировщик начинает с отправки одного кадра (SF) запроса к блоку управления (отправителю). Однако в этом случае ответ превышает 7 байт. Из-за этого ECU отправляет первый кадр (FF), который содержит информацию о общая длина пакета (от 8 до 4095 байт), а также начальный фрагмент данных. Когда тестировщик получает FF, он отправит кадр управления потоком (FC), который сообщает ECU как должна передаваться остальная часть передачи данных.

После этого ECU отправит последовательные кадры (CF), которые содержат оставшуюся полезную нагрузку данных.

ISO-TP играет важную роль в большинстве протоколов диагностики на основе CAN . Прежде чем мы покажем практические полезно привести примеры таких коммуникационных потоков ознакомиться с наиболее распространенными протоколами диагностики транспортных средств .

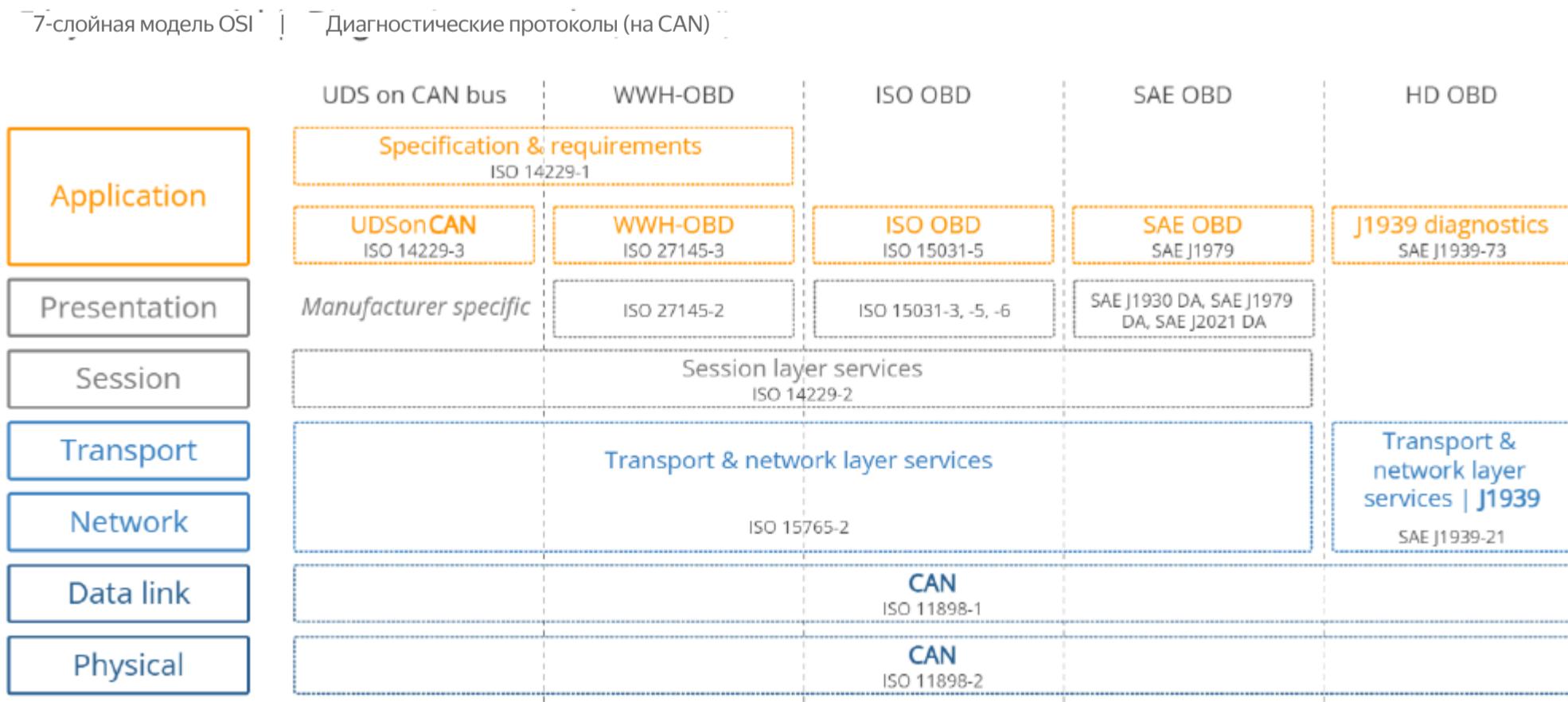


UDS против OBD2 против WWH-OBD против OBDonUDS

Распространенный вопрос заключается в том, как UDS соотносится с бортовой диагностикой (OBD2), согласованной во всем мире системой OBD (WWH-OBD) и OBDonUDS. Чтобы понять это, важно сначала отметить следующее:

OBD (бортовая диагностика) сегодня реализована по-разному в разных странах и на разных транспортных средствах.

Это иллюстрируется приведенной ниже моделью OSI, включающей используемые сегодня протоколы диагностики транспортных средств на основе CAN:



Давайте рассмотрим каждый диагностический протокол:

ISO OBD (или EOBD) относится к спецификации протокола OBD, разрешенной для использования в автомобилях ЕС, в то время как SAE OBD относится к спецификации протокола OBD, разрешенной для использования в США. Эти два устройства технически эквивалентны и, следовательно, часто называются просто OBD или OBD2

HD OBD (SAE J1939) обычно относится к БД для тяжелых условий эксплуатации и обычно реализуется с помощью J1939 протокол как в автомобилях, произведенных в США, так и в ЕС, с J1939-73, определяющим диагностические сообщения

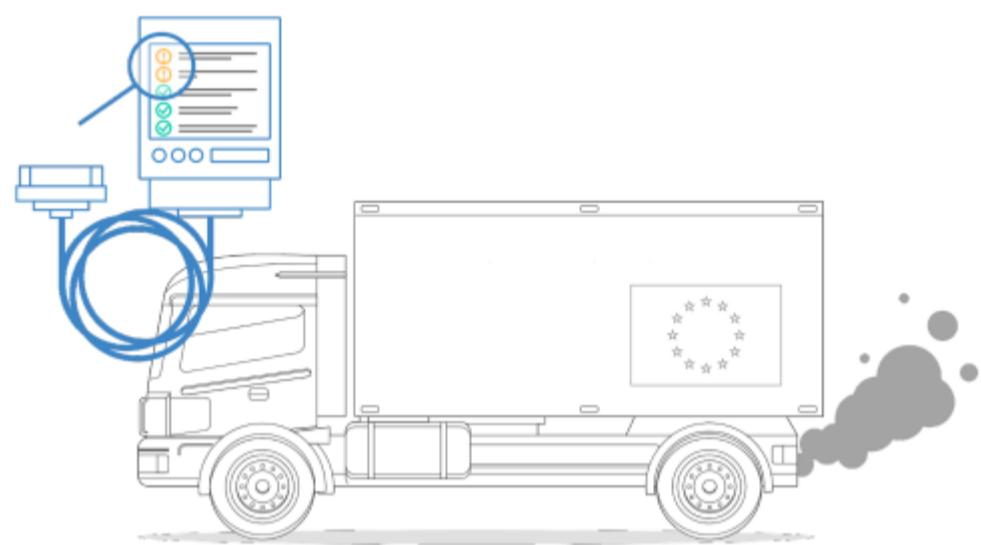
UDS (ISO 14229) был внедрен производителями транспортных средств для удовлетворения потребностей в расширенной диагностике данные / функциональность - за пределами протоколов OBD, ориентированных на выбросы. Он реализован в большинстве ЭБУ сегодня, на разных рынках и для разных типов транспортных средств, хотя сам по себе UDS не обеспечивает необходимой стандартизации требуется, чтобы служить альтернативой OBD

WWH-OBD (и / или, возможно, OBDonUDS) предоставляют обновленную версию OBD2 для диагностики, связанной с выбросами - на основе UDS

Чтобы разобраться в UDS, полезно лучше разобраться в WWH-OBD и OBDonUDS:

Что такое WWH-OBD (ISO 27145)?

WWH-OBD - это глобальный стандарт диагностики транспортных средств, разработанный ООН в соответствии с мандатом Глобальных технических Правил (GTR). Его цель - обеспечить единый, перспективная альтернатива существующим протоколам OBD (ISO OBD, SAE OBD, HD OBD). Кроме того, WWH-OBD является на базе UDS для расширенной диагностики функциональность, уже реализованная большинством производителей автомобильной техники Сегодня.



Преимущества WWH-OBD

Переход с OBD2 на WWH-OBD принесет ряд преимуществ, в первую очередь связанных с использованием протокола UDS в качестве основы. Прежде всего, можно увеличить объем данных. Идентификаторы параметров OBD2 (PID) ограничены 1 байтом, ограничение количества уникальных типов данных 255, в то время как идентификатор данных UDS (DID) равен 2 байтам, что позволяет использовать 65535 параметры.

Для диагностических кодов неисправностей (DTCS) OBD2 допускает 2-байтовые коды неисправностей. Здесь WWH-OBD допускает "расширенные коды неисправностей" в 3 байта. Это позволяет группировать коды неисправностей по 2-байтовым типам и использовать 3-й байт в качестве индикатора режима сбоя для указания Подтипа DTC. Кроме того, WWH-OBD позволяет классифицировать коды неисправностей в зависимости от того, насколько серьезной является проблема в отношении качества выхлопных газов.

WWH-OBD также стремится учитывать потенциальные будущие требования, позволяя использовать интернет-протокол (IP) используется как альтернатива CAN, что означает, что UDsonIP также будет возможен в будущих реализациях WWH-OBD. Одним из потенциальных преимуществ этого будет возможность в один прекрасный день выполнять удаленную диагностику по тому же протоколу.

Каково состояние развертывания WWH-OBD?

Цель WWH-OBD - служить глобальным стандартом на всех рынках и для всех типов транспортных средств (легковые, грузовые, автобусы, ...). Кроме того, цель состоит в том, чтобы потенциально расширить стандартизированную функциональность, выходящую за рамки простого контроля выбросов.

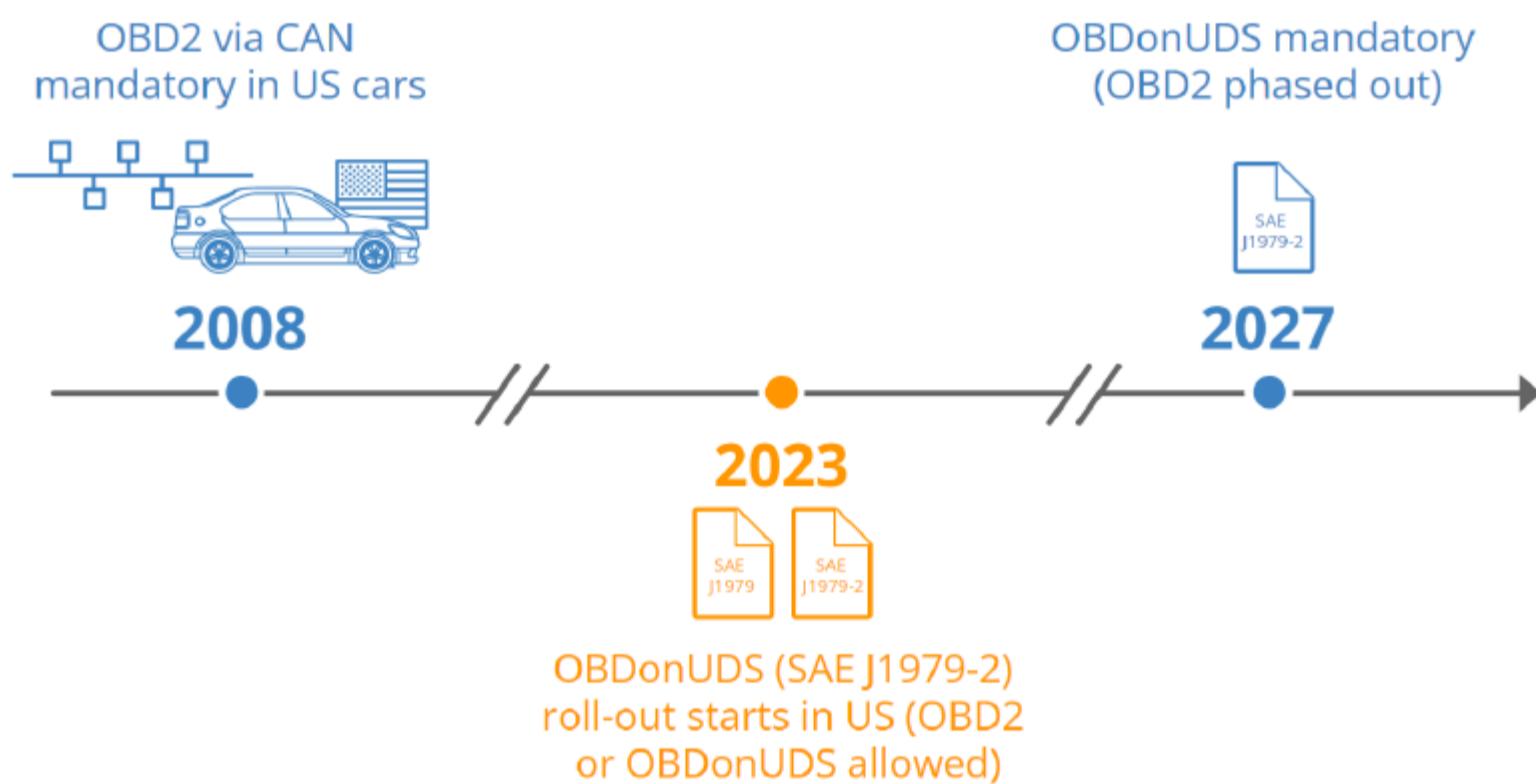
На практике WWH-OBD требуется в ЕС с 2014 года для новых транспортных средств большой грузоподъемности (в соответствии со стандартами Euro-VI). В этом случае обратите внимание, что HD OBD (согласно J1939) по-прежнему разрешен для этих транспортных средств.

Однако, помимо этого, внедрение WWH-OBD было ограничено. Одна из проблем заключается в том, что WWH-OBD в настоящее время не принят EPA / CARB в США. Смотрите, Например, Это обсуждение, чтобы узнать о возможных мотивах. Однако недавно в США был принят OBDonUDS (SAE J1979-2).

Что такое OBDonUDS (SAE J1979-2)?

Аналогично тому, как OBD2 был разделен на "SAE OBD" (SAE J1979) для США и "ISO OBD" (ISO 15031) для ЕС, "следующий поколение OBD2 снова может быть разделено по регионам.

В частности, WWH-OBD (ISO 21745) уже принят в ЕС для автомобилей большой грузоподъемности, но еще не принят в США. Вместо этого недавно было решено внедрить OBD на UDS в автомобилях США в форме стандарта SAE J1979-2, который служит обновлением SAE J1979. Новый стандарт SAE J1979-2 также называется OBDonUDS. Цель состоит в том, чтобы инициировать переходный этап, начинаящийся в 2023 году, когда ЭБУ разрешено поддерживать либо OBD2, либо OBDonUDS. С 2027 года, OBDonUDS станет обязательным требованием для всех автомобилей, произведенных в США.



Забегая вперед: WWH-OBD против OBDonUDS

Напомним, что WWH-OBD и OBDonUDS являются возможными решениями для создания протокола "следующего поколения" для бортовой диагностики, связанной с выбросами. Еще неизвестно, будут ли они существовать параллельно (как ISO / SAE OBD), или один из протоколов станет стандартом де-факто как в США, так и в ЕС и за его пределами.

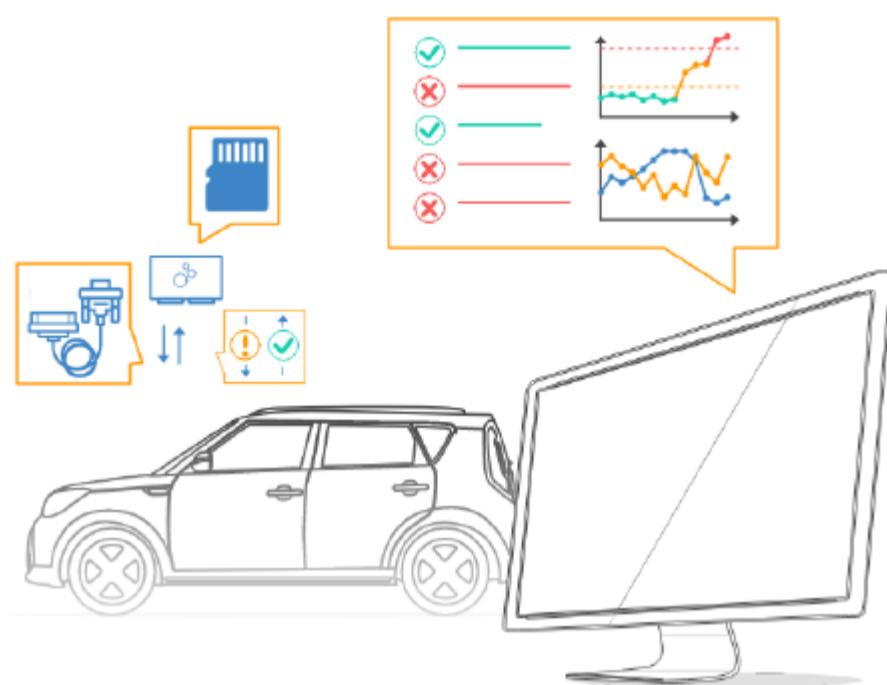
В любом случае основой для диагностики, связанной с выбросами, будет UDS, которая упростит программирование ECU, поскольку диагностика, связанная с выбросами, все чаще может реализовываться в рамках той же структуры на основе UDS, что и расширенная диагностика, специфичная для конкретного производителя.

Вопросы и ответы: Как запросить / записать данные UDS

Теперь мы рассмотрели основы UDS и Транспортный протокол на основе CAN.

Имея это в виду, мы можем дать некоторые конкретные рекомендации о том, как вы можете работать с данными UDS на практике. В частности, мы сосредоточимся на том, как UDS можно использовать для регистрации различных параметров данных, такие как уровень заряда (SoC) в электромобилях.

Перед примерами мы рассмотрим часто задаваемые вопросы о регистрации данных UDS.:

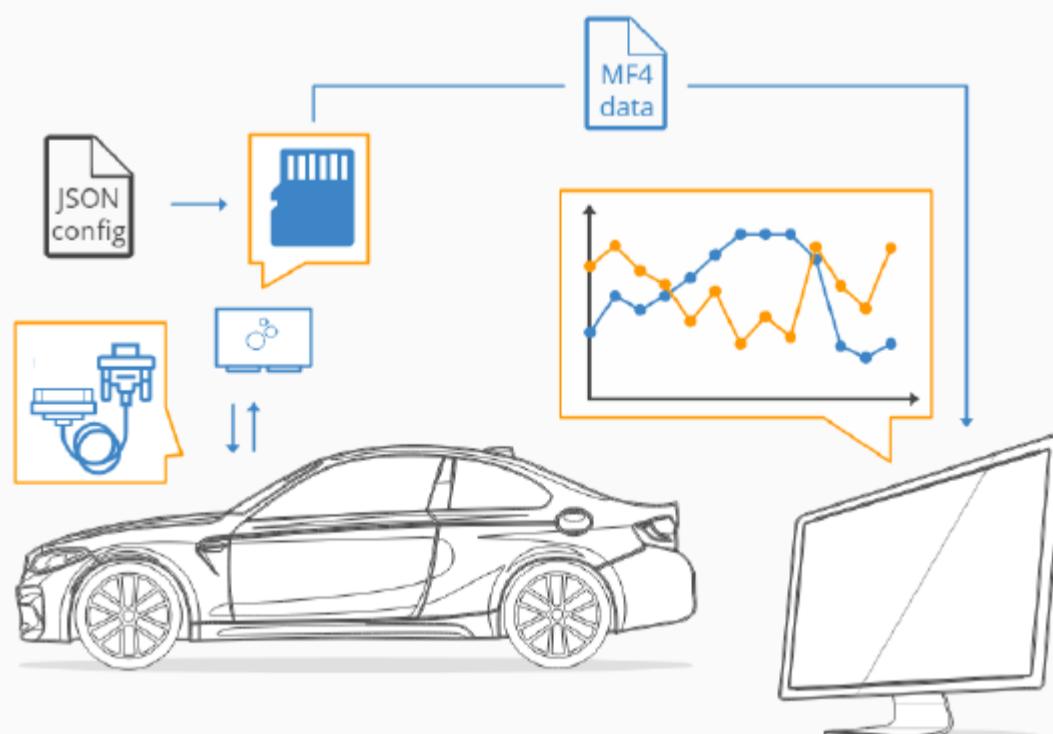


Может ли CANedge записывать данные UDS?

Да, как мы покажем ниже, CANedge можно настроить для запроса данных UDS. Фактически, устройство может быть настроен для передачи до 64 пользовательских кадров CAN в виде периодических или одиночных кадров. Вы можете управлять идентификатором CAN, CAN полезной нагрузкой данных, периодом времени и задержкой.

Для однокадровой связи UDS вы просто настраиваете CANedge с помощью фрейма запроса, который запускает единый фрейм ответа от ECU.

Для многокадровой связи вы можете снова настроить CANedge с помощью фрейма запроса, а затем добавить поток Управляющий фрейм в виде отдельного фрейма, который будет передаваться через X миллисекунд после фрейма запроса. Регулируя синхронизацию, вы можете настроить это так, чтобы управление потоком отправлялось после того, как блок управления отправит ответ первого кадра.



Обратите внимание, что CANedge запишет ответы UDS в виде необработанных кадров CAN. Затем вы можете обработать и декодировать эти данные с помощью вашего предпочтительного программного обеспечения (например, векторных инструментов) или нашего CAN bus Python API для повторной сборки и декодирования кадров.

Примечание: В будущих обновлениях встроенного ПО мы можем улучшить функциональность передачи, чтобы CANedge мог передавать настраиваемые кадры CAN на основе условий запуска, таких как получение определенного кадра. Это позволит отправить Управление потоком кадров с задержкой после приема первого кадра, обеспечивая более простую и надежную

реализация. С учетом сказанного, текущая функциональность будет служить для поддержки большинства передаваемых UD, связанных с сервисы 0x22 (считывание данных по идентификатору) и 0x19 (считывание информации DTC).

Какие данные я могу регистрировать через UDS?

Очень распространенным вариантом использования записи данных UDS с помощью "автономных" регистраторов данных будет получение диагностического кода неисправности значения, например для использования при диагностике проблем.

В дополнение к кодам неисправностей, UDS позволяет запрашивать "текущее значение" различных датчиков, связанных с каждым блоком управления. Это позволяет, например, руководителям автопарка, исследователям и т.д. собирать интересующие данные, такие как скорость, обороты в минуту, уровень заряда, температуры и т.д. Из автомобиля (при условии, что они знают, как запрашивать / декодировать данные, как описано ниже).

Помимо вышесказанного, UDS, конечно, также может использоваться для более низкоуровневого управления ЭБУ, например, для сброса настроек и перепрошивки встроенное ПО, хотя такие варианты использования чаще выполняются с использованием интерфейса шины CAN, а не "автономного" устройства.

Данные UDS являются собственностью - или существуют общедоступные параметры?

Важно отметить, что UDS является диагностическим протоколом конкретного производителя. Другими словами, в то время как UDS обеспечивает стандартизированной структура диагностического сообщения фактическое "содержание" сообщения остается собственностью компании и в большинстве случаев известно только производителю конкретного транспортного средства / блока управления.

Например, UDS стандартизирует способ запроса данных параметров от ECU через службу "Считывания данных по идентификатору" (0x22). Но в нем не указан список стандартизованных идентификаторов и правил интерпретации. Этим UDS отличается от OBD2, где общедоступный список PID-кодов OBD2 позволяет практически любому пользователю интерпретировать данные OBD2 со своего автомобиля.

С учетом сказанного, автомобили, поддерживающие WWH-OBD или OBDonUDS, могут поддерживать некоторые из обычных PID OBD2, таких как speed, RPM и т.д. Через обычные значения PID - но с префиксом 0xF4, как показано в примере 1 ниже.

Как правило, только производитель (OEM) будет знать, как запросить собственные параметры через UDS - и как интерпретировать результат. Конечно, одним из исключений из этого правила являются случаи, когда компаниям или частным лицам удается выполнить обратный анализ спроектировать эту информацию. Участие в таком обратном проектировании - очень сложная задача, но иногда вы можете найти общедоступная информация и файлы DBC, в которых другие выполняли это упражнение. Наше введение к файлам DBC содержит список общедоступных Базы данных DBC / декодирования.

Кому выгодно регистрировать данные UDS?

Из-за запатентованного характера связи UDS, она, как правило, наиболее актуальна для инженеров-автомобилестроителей, работающих в OEM-производителях. Такие инструменты, как регистратор данных CANedge CAN bus, позволяют этим пользователям записывать необработанный трафик CAN от транспортного средства - и в то же время запрашивать диагностические коды неисправностей и значения конкретных параметров через UDS.

Кроме того, некоторые пользователи вторичного рынка, такие как менеджеры автопарка и даже частные лица, могут извлечь выгоду из UDS, предполагая, что они способны идентифицировать информацию обратного проектирования, необходимую для запроса и декодирования интересующих параметров UDS.

Регистрация данных UDS также будет становиться все более актуальной при условии, что WWH-OBD будет внедрен должным образом. Уже сегодня WWH-OBD используется в автомобилях большой грузоподъемности ЕС, выпущенных после 2014 года, что означает, что связь UDS будет актуальна для случаев использования, связанных с бортовой диагностикой в этих автомобилях.

Центральный шлюз (CGW): Зачем регистрировать данные датчиков через UDS, а не через CAN?

Если вы хотите запросить диагностические коды неисправностей (DTC) на основе UDS, вам, конечно, придется использовать UDS communication для этой цели. Однако, если ваша цель - записать текущие значения датчика, это менее понятно. Как правило, параметры данных, представляющие интерес, например, для телематики автомобиля (скорость, уровень заряда и т.д.), уже будут передана данных между блоками управления по шине CAN в виде необработанных кадров CAN - без использования диагностического инструмента запрос этой информации. Это связано с тем, что ECU полагаются на передачу этой информации как часть своей работы (как объясняется во введении к CAN bus).

Таким образом, если у вас есть прямой доступ к шине CAN, проще просто зарегистрировать этот необработанный трафик CAN и обработать его. Если вы производитель автомобиля, вы будете знать, как интерпретировать эти необработанные данные CAN в любом случае - и это будет проще сделать в большинстве случаев выполните настройку вашего устройства и последующую обработку. Если вы работаете на вторичном рынке, вам также будет проще реконструировать необработанные кадры CAN, поскольку вы можете сосредоточиться на отдельных кадрах и избежать уровня сложности запроса / ответа.

Однако, несмотря на вышесказанное, одна из ключевых причин, по которой UDS часто используется для извлечения значений датчиков, связана с "шлюзами". В частности, все большая доля современных автомобилей начала блокировать доступ к необработанным данным шины CAN через разъем OBD2. Особенно часто это касается немецких автомобилей, а также электромобилей. Чтобы записать существующий трафик CAN в таком автомобиле, вам потребуется, например, использовать адаптер CANCrocodile и "привязать" его к жгуту проводов CAN low / high. Это, в свою очередь, потребует обнажить жгут проводов путем снятия панели, что часто является непомерно большим для многих случаев использования. В отличие от этого, OBD2 в разъеме шлюзы, как правило, по-прежнему позволяют УДС коммуникации - ВКЛ. датчик общения.

Вторая - и более тонкая - причина заключается в том, что большая часть работ по обратному проектированию выполняется "производителями ключей OBD2". Они разрабатывают инструменты, которые позволяют извлекать данные из множества различных автомобилей через разъем OBD2. Все чаще единственный способ для этих ключей получать полезную информацию через разъем OBD2 - это связь UDS, что обеспечивает пропорционально более высокую доступность информации / баз данных, связанных с параметрами UDS, по сравнению с параметрами raw CAN параметры.

Поддерживает ли мой автомобиль UDS?

Поскольку сегодня большинство ECU поддерживают связь UDS, ответом будет "да, скорее всего".

Если вы производитель транспортного средства, у вас в большинстве случаев будет информация, необходимая для построения запросов UDS для каких бы данные вам ни были нужны - вы также будете знать, как их расшифровать.

Для конкретного случая записи данных WWH-OBD в грузовых автомобилях EC стандартизированная информация DID может быть записана обоими OEM-производителями и пользователями вторичного рынка - аналогично тому, как вы можете записывать общедоступные PID OBD2 для легковых / грузовых автомобилей.

Помимо вышесказанного, если вы работаете на вторичном рынке и хотите записать служебную информацию UDS с легкового / грузового автомобиля, это будет затруднительно. В этом случае вам придется либо связаться с OEM-производителем (например, в качестве системного интегратора / партнера), либо идентифицировать информацию о запросе / декодировании посредством обратного проектирования. Последнее на практике невозможно для большинства людей.

В отдельных случаях вы можете использовать общедоступную информацию, размещенную, например, на github поможете в создании запросов UDS - и расшифровке ответов. Просто имейте в виду, что общедоступные ресурсы основаны на обратном проектировании - и может быть некорректной или устаревшей. Поэтому вы должны относиться ко всей информации со значительной долей серьезности.

Почему UDS приобретает все большее значение?

Если ваш вариант использования предполагает запись данных с автомобилей, выпущенных в период с 2008 по 2018 год, вас чаще всего будут интересовать данные, которые можно собрать с помощью OBD2 data logging. Это связано с тем, что большинство автомобилей ICE после 2008 года поддерживают значительную долю общедоступные идентификаторы параметров OBD2, такие как скорость, обороты в минуту, положение дроссельной заслонки, уровень топлива и т.д. Однако ожидается, что доступность данных OBD2 со временем будет снижаться по некоторым причинам.

Прежде всего, как мы объясняли в предыдущем разделе, ожидается, что WWH-OBD (на основе UDS) или OBDonUDS будут постепенно заменять OBD2 в качестве стандарта де-факто для диагностики транспортных средств.

Во-вторых, с появлением электромобилей законодательно установленный OBD2 вообще не обязательно поддерживается. И даже если EV поддерживает некоторые PID OBD2, вы заметите из нашего списка PID OBD2, что некоторые из наиболее важных параметров EV, таких как состояние заряда (SoC) и состояние работоспособности (SoH) недоступны через OBD2. Напротив, UDS остается поддерживаемым в большинстве электромобилей и будет предоставлять доступ к гораздо более широкому спектру данных - хотя и без удобства использования после выхода на рынок общедоступного списка UDS параметры (по крайней мере, пока). Ожидается, что продажи электромобилей обгонят продажи автомобилей ICE в период с 2030 по 2040 год - и, таким образом, UDS коммуникация будет становиться все более актуальной.

О CANedge CAN logger

CANedge позволяет легко записывать данные CAN / UDS на 8-32 ГБ SD-карту. Вы можете настроить, какие кадры CAN отправлять, включая пользовательские запросы UDS и фреймы управления потоком. Данные могут быть обработаны с помощью бесплатного программного обеспечения / инструментов API.



CANedge2 (Wi-Fi) и CANedge3 (3G / 4G) также позволяют вам автоматически загружать данные на ваш собственный сервер / облако, в то время как все CANedge доступны устройства со встроенным GPS / IMU для добавления более 40 сигналов, таких как местоположение, скорость, расстояние поездки и других, в ваши журналы.

Пример 1: Запись данных UDS за один кадр (скорость через WWH-OBD)

Чтобы показать, как UDS работает на практике, мы начнем с базового примера. Как указывалось ранее, WWH-OBD основан на UDS - и будет установлен на всех грузовых автомобилях ЕС после 2014 года.

В рамках этого многие грузовики большой грузоподъемности ЕС позволят вам запрашивать такие параметры, как скорость, обороты в минуту, уровень топлива и т.д., способом, аналогичным тому, как вы запрашивали бы эту информацию через Запросы PID OBD2 в автомобиле - смотрите наше введение к OBD2 и Введение к регистратору данных OBD2 для получения подробной информации. Однако в разделе WWH-OBD (ISO 21745-2), PID OBD2 заменены на DID WWH-OBD.



Для службы 01 PID WWH-OBD эквивалентны PID OBD2, за исключением того, что впереди добавлен 0xF4. Например, Скорость транспортного средства с PID OBD2 равна 0x0D, которая становится 0xF40D в контексте WWH-OBD.

В этом случае мы будем использовать регистратор данных CANedge2 по шине CAN в качестве нашего инструмента "тестирования". Этот инструмент позволяет записывать необработанные данные по шине CAN, а также передавать пользовательские кадры CAN. Для запроса скорости автомобиля WWH-OBD, мы будем использовать службу UDS 0x22 (считывание данных по идентификатору) и идентификатор данных 0xF40D. Запрос / ответ выглядит следующим образом:

>UDS on CAN - single frame communication example				
<ВремяИДЕНТИФИКАТОР CAN (ШЕСТНАДЦАТЬБИТНЫЙ)Байты данных (ШЕСТНАДЦАТЬБИТНЫЕ)		Отправитель	Тип фрейма	Условные обозначения
4.0435 18DB33F1	03 22 F4 0D AA AA AA AA AA	CANedge (клиент)	Одиночный кадр (SF)	Поле PCI
4.0468 18DAF100	04 62 F4 0D 32 AA AA AA AA	Блок управления (сервер)	Первый кадр (FF)	Идентификатор UDS SID (запрос)
				uds sid (response) padding/unused payload (1 byte)

Сведения о потоке связи

Обратите внимание, что запрос отправляется с идентификатором CAN 0x18DB33F1. Это тот же 29-битный идентификатор CAN, который использовался бы для выполнения функционально адресуемый запрос OBD2 PID в автомобилях большой грузоподъемности, который можно сравнить с 11-разрядным идентификатором CAN 0x7DF используется для запросов OBD2 в автомобилях.

Ответ имеет идентификатор CAN 0x18DAF100, пример физического идентификатора ответа, соответствующего идентификаторам, которые вы видите в обычных Ответы OBD2 от автомобилей большой грузоподъемности.

Давайте разберем полезную нагрузку сообщений потока связи:

Сначала CANedge2 отправляет запрос с одним кадром (SF):

Начальные 4 бита поля PCI равны типу кадра (0x0 для SF).

Следующие 4 бита поля PCI равны длине запроса (3 байта)

2-й байт содержит идентификатор службы (SID), в данном случае 0x22

Третий и четвертый байты содержат значение DID для скорости транспортного средства в WWH-OBD (0xF40D)

Остальные байты дополнены.

В ответ на этот запрос грузовик отвечает ответом в один кадр (SF):

1-й байт снова отражает поле PCI (теперь длиной 4 байта)

2-й байт - это SID ответа для считывания данных по идентификатору (0x62, т.е. 0x22 + 0x40)

3-й и 4-й байты снова содержат значение DID 0xF40D

5-й байт содержит значение скорости транспортного средства, 0x32

Здесь мы можем использовать те же правила декодирования, что и для ISO / SAE OBD2, что означает, что физическое значение скорости транспортного средства - это просто десятичная форма 0x32, то есть 50 км / ч. Смотрите также наш инструмент преобразования PID OBD2. Если вы знакомы с протоколированием PID OBD2, должно быть очевидно, что запросы WWH-OBD очень похожи, за исключением использование структуры полезной нагрузки UDSonCAN для запросов / ответов.

Пример 2: Запись и декодирование многокадровых данных UDS (SoC)

В этом разделе мы проиллюстрируем, как многокадровые UDS коммуникация работает в контексте CAN ISO-TP.

В частности, мы будем использовать CANedge2 и сервис UDS SID 0x22 (читывает данные по идентификатору) для запроса текущего значения значение уровня заряда (SoC%) Hyundai Kona электромобиль.

Сначала CANedge настраивается на отправку двух кадров CAN:



1. Запрос с одним кадром (SF) (период: 5000 мс, задержка: 0 мс)
2. Кадр управления потоком (FC) (период: 5000 мс, задержка: 100 мс)

Подмножество результирующего потока связи выглядит следующим образом:

1.0135	7E4	03 22 01 01 AA AA AA	AA	CANedge (клиент)	Одиночный кадр (SF)
1.0228	7EC	10 3E 62 01 01 FF F7	E7	Блок управления (сервер)	Первый кадр (FF)
1.0235	7E4	30 00 00 00 00 00 00	00 00	CANedge (клиент)	Управление потоком (FC)
1.0426	7EC	21 И ДАЛЕЕ 96 3E 11 42 90	83	Блок управления (сервер)	Последовательный кадр (CF)
1.0486	7EC	22 00 12 0E F3 17 16	16	Блок управления (сервер)	Последовательный кадр (CF)
1.0586	7EC	23 16 17 16 00 00 18	C3	Блок управления (сервер)	Последовательный кадр (CF)
1.0687	7EC	24 02 C2 0B 00 00 92	00	Блок управления (сервер)	Последовательный кадр (CF)
1.0787	7EC	25 01 04 50 00 01 02	0E	Блок управления (сервер)	Последовательный кадр (CF)
1.0886	7EC	26 00 00 63 F3 00 00	60	Блок управления (сервер)	Последовательный кадр (CF)
1.0986	7EC	27 74 00 3D A5 07 0D	01	Блок управления (сервер)	Последовательный кадр (CF)
1.1086	7EC	28 7F 00 00 00 00 03 E8		Блок управления (сервер)	Последовательный кадр (CF)

→ Reassembled UDS frame
1.0135 7EC 62 01 01 FF F7 E7 00 00 00 ... 7F 00 00 00 00 03 E8

Сведения о потоке передачи данных

Далее мы подробно рассмотрим эту связь между CANedge и ECU. Прежде всего, начальный одиночный Запрос фрейма (SF) создается по той же логике, что и в нашем предыдущем примере - содержит поле PCI, SID 0x22 и DID. В этом случае мы используем DID 0x0101. В ответ на первоначальный запрос SF целевой блок управления блоком управления отправляет первый Кадровый (FF) ответ:

Начальные 4 бита равны типу кадра (0x1 для FF)

Следующие 12 битов равны размеру полезной нагрузки данных, в данном случае 62 байта (0x3E).

3-й байт - это SID ответа для считывания данных по идентификатору (0x62, т.е. 0x22 + 0x40)

4-й и 5-й байты содержат идентификатор данных (DID) 0x0101

Остальные байты содержат начальную часть полезных данных они 0x0101

После ФФ, тестер инструмент направляет поток управления (ФК) рамка:

В начальные 4 бита равны типу кадра (0x3 для FC)

Следующие 4 бита указывают, что ECU должен "Продолжить отправку" (0x0)

2-й байт устанавливает оставшиеся кадры, которые будут отправлены без управления потоком или задержка

3-й байт устанавливает минимальное время разделения последовательных кадров (ST) равным 0.

Как только ECU получает FC, он отправляет оставшиеся последовательные кадры (CF):

Первые 4 бита равны типу кадра (0x2 для CF)

Следующие 4 бита равны значениям счетчика, увеличиваются от 1 до 8. в этом случае

В оставшиеся 7 байт каждого CF содержат остальную полезную нагрузку для DID

Что касается проприетарных данных UDS

Часть используемой здесь информации является проприетарной. В частности, как правило, неизвестно, какой Идентификатор данных (DID) использовать для того, чтобы запросить, например, уровень заряда данного электромобиля, если только вы не являетесь производителем автомобиля (OEM). Кроме того, как объясняется в следующем разделе, неизвестно, как декодировать полезную нагрузку ответа.

Однако существуют различные онлайн-ресурсы, например, на [github](#), где энтузиасты создают базы данных с открытым исходным кодом для конкретных параметров и определенных автомобилей (на основе обратного проектирования). Информация, которую мы используем для этого конкретного сообщения, взята из одной такой базы данных.

Относительно используемых идентификаторов CAN

В этом случае мы используем идентификатор CAN 0x7E4 для запроса данных от определенного блока управления, который, в свою очередь, отвечает идентификатором CAN 0x7EC. Это известно как физически адресованный запрос.

Напротив, функционально адресуемый запрос будет использовать идентификатор CAN 0x7DF (или 0x18DB33F1 в автомобилях большой грузоподъемности). Обычно, идентификаторы CAN запроса и ответа сопоставляются (согласно таблице ниже), и вы можете идентифицировать физический идентификатор запроса соответствующий конкретному физическому идентификатору ответа, вычитая значение 8 из идентификатора ответа. Другими словами, если ECU отвечает через CAN ID 0x7EC, идентификатор физического запроса, предназначенный для этого ECU, будет 0x7E4 (как в нашем примере EV).

Поскольку вы можете не знать, на какой адрес ориентироваться изначально, в некоторых случаях вы можете начать с отправки функционального запроса с использованием идентификатора CAN 0x7DF, и в этом случае соответствующий блок управления должен предоставить положительный ответ на первый кадр, если полезная нагрузка начального запроса структурирована правильно. В некоторых транспортных средствах вы можете также отправлять последующий поток Управляющий кадр, использующий тот же идентификатор CAN, 0x7DF, чтобы запустить ECU для отправки оставшихся последовательных кадров. Однако некоторые реализации могут потребовать, чтобы вы вместо этого использовали идентификатор запроса физической адресации для потока Фрейм управления.

Реализация структуры запроса с динамическим обновлением идентификаторов CAN может быть сложной. Если вы являетесь производителем, вы, конечно, знаете соответствующие идентификаторы CAN, которые следует использовать для отправки запросов на обслуживание с физическим адресом. Если нет, вы можете выполните анализ, используя, например, интерфейс шины CAN, чтобы определить, какой ответ могут выдавать идентификаторы CAN при функциональной отправке адресуемые запросы на обслуживание - и используйте эту информацию для построения вашей конфигурации.

Отдельно отметим, что в стандарте ISO 15765-4 указано, что расширенные запросы / ответы на диагностику могут использовать установленный законом диапазон идентификаторов OBD2 CAN до тех пор, пока это не мешает - что мы и наблюдаем в этом конкретном примере Hyundai Kona, где Идентификаторы 0x7EC / 0x7E4 используются для личных данных. Смотрите также таблицу из ISO 15765-4 для получения обзора разрешенных идентификаторов OBD CAN для использования в функциональных и физических запросах OBD PID.

>Legislated OBD CAN identifiers (11-bit)		
<CAN ID	Описание	
0x7DF от 0x7E0 до 0x7E7	Функционально адресованный запрос, отправленный тестером Физический запрос от тестера к ECU с # 1 по # 8	
От 0x7E8 до 0x7EF Установленный законом OBD	Идентификаторы CAN (29 бит) Физический отклик от ECU # 1 до # 8 на тестер	
ИДЕНТИФИКАТОР CAN	Описание	
0x18DB33F1 0x18DAxxF1	Функционально адресованный запрос, отправленный тестером Физический запрос от тестера к ECU # xx	
0x18DAF1xx	Физический ответ от ECU к тестеру	

Что касается временных параметров

В приведенном выше примере мы обычно фокусируемся на последовательности кадров CAN. Важна последовательность: например, если ваш инструмент тестирования отправляет кадр управления потоком до получения первого кадра, кадр управления потоком будет либо проигнорирован (таким образом, не запускаются последовательные кадры), либо вызовет ошибку.

Однако, в дополнение к этому, также необходимо будет соблюдать определенные пороговые значения синхронизации. Например, если ваш инструмент тестирования получает первый кадр из многокадрового ответа от ECU., блок управления отключится, если кадр управления потоком не будет отправлен в течение установленного периода времени.

Как правило, вы должны настроить свой тестер (например, CANedge) таким образом, чтобы всегда отправлялся кадр управления потоком после получения ответа на первый кадр от ECU (обычно это происходит в течение 10-50 мс с момента отправки начального запроса) - но таким образом, чтобы он был отправлен в течение установленного времени после получения первого кадра (например, в течение 0-50 мс). Для получения подробной информации по этому поводу, не стесняйтесь обращаться к нам.

Как повторно собрать и декодировать многокадровые данные UDS?

Теперь мы показали, как вы можете запросить / записать многокадровый ответ UDS для сбора данных фирменного датчика ECU. В чтобы извлечь "физические значения", такие как состояние заряда, вам нужно знать, как интерпретировать кадры отклика CAN. Как поясняется, что информация о "декодировании", как правило, является частной и известна только производителю. Однако, в конкретном случае с Hyundai Kona EV, мы знаем следующее о сигнале SoC из онлайн-ресурсов.:

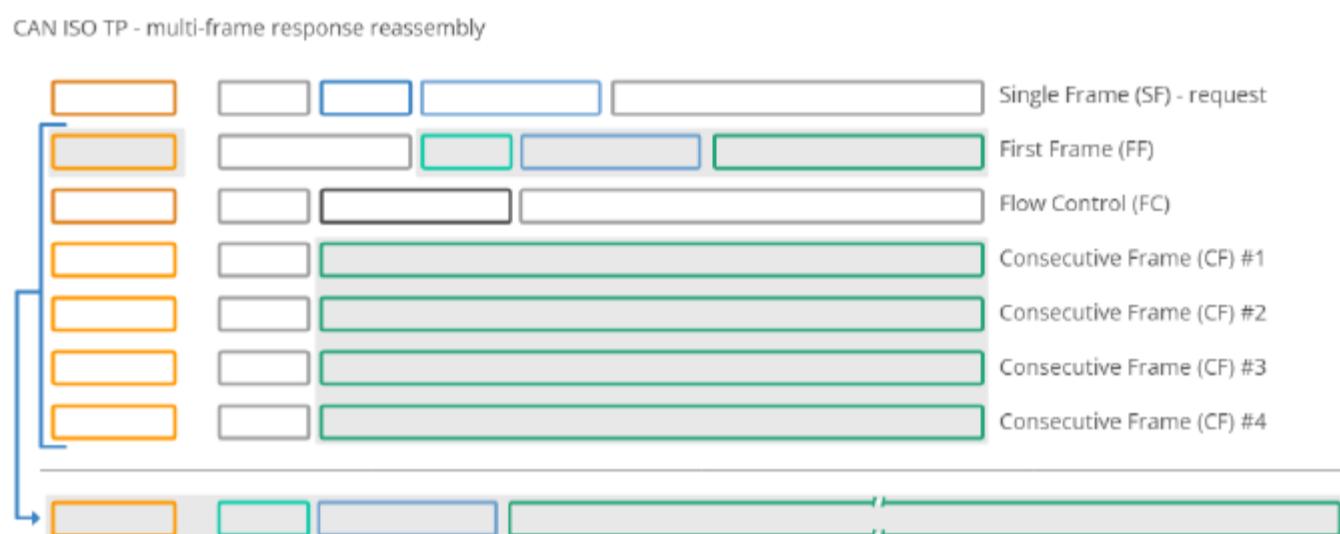
Сигнал находится в 8-м байте полезной нагрузки данных

Сигнал без знака

Сигнал имеет масштаб 0,5, смещение 0 и единицу измерения "%"

Итак, как нам использовать эти знания для декодирования сигнала?

Сначала нам нужно собрать сегментированные рамки CAN. Результат этого показан в предыдущем сообщении пример. Посредством повторной сборки мы получаем "CAN-фрейм" с идентификатором 0x7EC и полезной нагрузкой, превышающей 8 байт. Полезная нагрузка в данном случае содержит SID в 1-м байте и DID во 2-м и 3-м байтах.



Повторно собранный кадр CAN можно обработать вручную, например, в Excel. Однако обычно мы рекомендуем использовать CAN базы данных (DBC-файлы) для хранения правил декодирования. В этом конкретном случае вы можете рассматривать повторно собранный CAN-фрейм как случай расширенного мультиплексирования. Мы предоставляем пример файла UDS DBC для Hyundai Kona, включая. Состояние заряда и сигналы температуры, которые могут быть полезны для вдохновения.

Наши шины CAN API-интерфейс Python позволяет сборку & ДБН декодирование мульти-кадр УДС ответы - смотрите в нашем примеров API хранилище для получения более подробной информации ВКЛ. образец сведения Хендай Кона.

```
>BO_2028 Battery:62 Vector_XXX
SG_M_SID_0x220101_StateOfChargeBMS m257 : 5618@1+(0.5,0)[0|0] "&" Vector_XXX
SG_response m98M : 15|16@0+(1.0)[0|0] "unit" Vector_XXX
SG_service M : 7|8||0+(1.0)[0|0] """" Vector_XXX
BO_1979 Temperature: 54 Vector_XXX
SG_M_SID_0x220100_IndoorTemp m256 : 64|881+(0.5,-40)[0|0] "degC" Vector_XXX
SG_response m98 : 15|16@0+(1.0)[0|0] "unit" Vector_XXX
SG service M : 7|8@0+(1.0)[0|0] "" Vector XXX
SG_M_SID_0x220100_OutdoorTemp : 79|8@0+(0.5,-40)[0|0] "" Vector_XXX
SG_M_SID_0x220100_Speed : 255|8||0+(0.5,0)[0|0] "" Vector_XXX
BA_DEF_BO_ "VFrameFormat" ENUM "StandardCAN", "ExtendedCAN", "StandardCAN_FD", "ExtendedCAN_FD", "J1939PG";
BA_DEF_ "ProtocolType" STRING :
BA_DEF_DEF_ "VFrameFormat" "";
BA_DEF_DEF_ "ProtocolType" "";
BA_ "ProtocolType"
BA_ "VFrameFormat" BO_2028 0;
SG_MUL_VAL_2028 M_SID_0x220101_StateOfChargeBMS response 257-257;
SG_MUL_VAL_2028 response service 98-98:
- - -
```

<Пример 3: Запишите идентификационный номер транспортного средства

Идентификационный номер транспортного средства (он же VIN, номер шасси, номер рамы) - это уникальный идентификационный код, используемый для дорожного Автомобили. Номер был стандартизирован и требуемой по закону, с 1980 года - см. VIN-номера страницы на [Википедия](#).

По VIN - идентификационный номер транспортного средства

Где корабль
построен

Vehicle brand,
engine, size, type

Vehicle
year

Vehicle serial
number

2T3RFREV7DW108177

Vehicle
manufacturer
identifier

Vehicle
check
number

Which plant
assembled
the vehicle

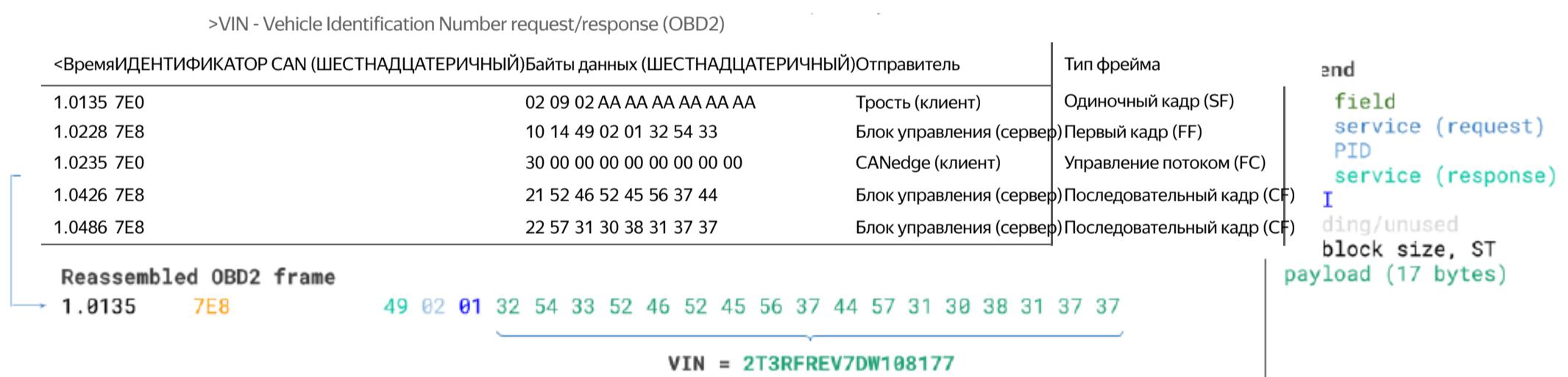
VIN состоит из 17 символов ASCII и может быть извлечен по запросу из транспортного средства. Это полезно, например, для регистрации данных или в телематике используются случаи, когда требуется уникальный идентификатор для связи, например, с файлами журналов шины CAN.

Байты данных CAN могут быть преобразованы из HEX в ASCII с помощью таблиц, онлайн-преобразователей HEX в ASCII, пакетов Python и т.д. Для примера, байт 0x47 соответствует букве "G". Поскольку VIN имеет длину 17 байт (17 символов ASCII), он не помещается в один кадр CAN, но должен быть извлечен с помощью многокадрового диагностического запроса / ответа, как в примере 2. Кроме того, VIN извлекается по-разному в зависимости от используемого протокола.

Ниже мы приводим три примера записи VIN.

3.1: Как записать VIN через OBD2 (SAE J1979)

Чтобы извлечь идентификационный номер транспортного средства, например, из легкового автомобиля, используя запросы OBD2, вы используете сервис 0x09 и PID 0x02:



Сведения о потоке связи

Логика структуры фрейма идентична примеру 2, при этом инструмент тестирования отправляет запрос на один фрейм с Поле PCI (0x02), идентификатор службы запроса (0x09) и идентификатор данных (0x02).

Транспортное средство отвечает Первым кадром, содержащим PCI, длину (0x014 = 20 байт), SID ответа. 0x49 т.е. 0x09 + 0x40) и идентификатор данных (0x02). После идентификатора данных следует байт 0x01, который представляет собой количество элементов данных (NODI), в данном случае 1 (подробнее см. SAE J1979 или ISO 15031-5).

Оставшиеся 17 байт равны VIN и могут быть переведены из HEX в ASCII с помощью методов, рассмотренных ранее.

3.2: Как записать VIN через UDS (ISO 14229-2)

Чтобы прочитать идентификационный номер транспортного средства через UDS, вы можете использовать SID UDS 0x22 и DID 0xF190:

>VIN - Vehicle Identification Number request/response (UDS on CAN)			
<ВремяИДЕНТИФИКАТОР CAN (шестнадцатеричный)	Байты данных (шестнадцатеричный)	Отправитель	Тип фрейма
1.0135 7E0	03 22 F1 90 AA AA AA AA AA	CANedge (клиент)	Одиночный кадр (SF)
1.0228 7E8	10 14 62 F1 90 32 54 33	Блок управления (сервер)	Первый кадр (FF)
1.0235 7E0	30 00 00 00 00 00 00 00 00	CANedge (клиент)	Управление потоком (FC)
1.0426 7E8	21 52 46 52 45 56 37 44	Блок управления (сервер)	Последовательный кадр (CF)
1.0486 7E8	22 57 31 30 38 31 37 37	Блок управления (сервер)	Последовательный кадр (CF)

reassembled UDS frame
→ 1.0135 7E8 62 F1 90 32 54 33 52 46 52 45 56 37 44 57 31 30 38 31 37 37

VIN = 2T3RFREV7DW108177

end
 field
 SID (request)
 DID
 SID (response)
 length/unused
 clock size, ST
 load (17 bytes)

Сведения о потоке связи

Как видно, поток связи запрос / ответ выглядит аналогично приведенному выше варианту OBD2. Основные изменения касаются к использованию службы UDS 0x22 вместо службы OBD2 0x09 - и использование 2-байтового UDS ПРИВЕЛО К 0xF190 вместо 1-байтового OBD2 PID 0x02. Кроме того, фрейм ответа UDS не включает количество элементов данных Поле (NODI) после DID, в отличие от того, что мы видели в случае OBD2.

3.3: Как записать VIN через WWH-OBD (ISO 21745-3)

Если вам необходимо запросить идентификационный номер транспортного средства у грузовика из ЕС после 2014 года, вы можете использовать протокол WWH-OBD . Структура идентична примеру UDS, за исключением того, что WWH-OBD определяет использование DID 0xF802 для VIN.

>VIN - Vehicle Identification Number request/response (WWH-OBD on CAN)			
<ВремяИДЕНТИФИКАТОР CAN (ШЕСТНАДЦАТЕРИЧНЫЙ)	Байты данных (ШЕСТНАДЦАТЕРИЧНЫЕ)	Отправитель	Тип фрейма
1.0135 18DB33F1	03 22 F8 02 AA AA AA AA AA	CANedge (клиент)	Одиночный кадр (SF)
1.0228 18DAF100	10 14 62 F8 02 32 54 33	Блок управления (сервер)	Первый кадр (FF)
1.0235 18DB33F1	30 00 00 00 00 00 00 00 00	CANedge (клиент)	Управление потоком (FC)
1.0426 18DAF100	21 52 46 52 45 56 37 44	Блок управления (сервер)	Последовательный кадр (CF)
1.0486 18DAF100	22 57 31 30 38 31 37 37	Блок управления (сервер)	Последовательный кадр (CF)

reassembled WWH-OBD frame
→ 1.0135 18DAF100 49 F8 02 32 54 33 52 46 52 45 56 37 44 57 31 30 38 31 37 37

>VIN = 2T3RFREV7DW108177

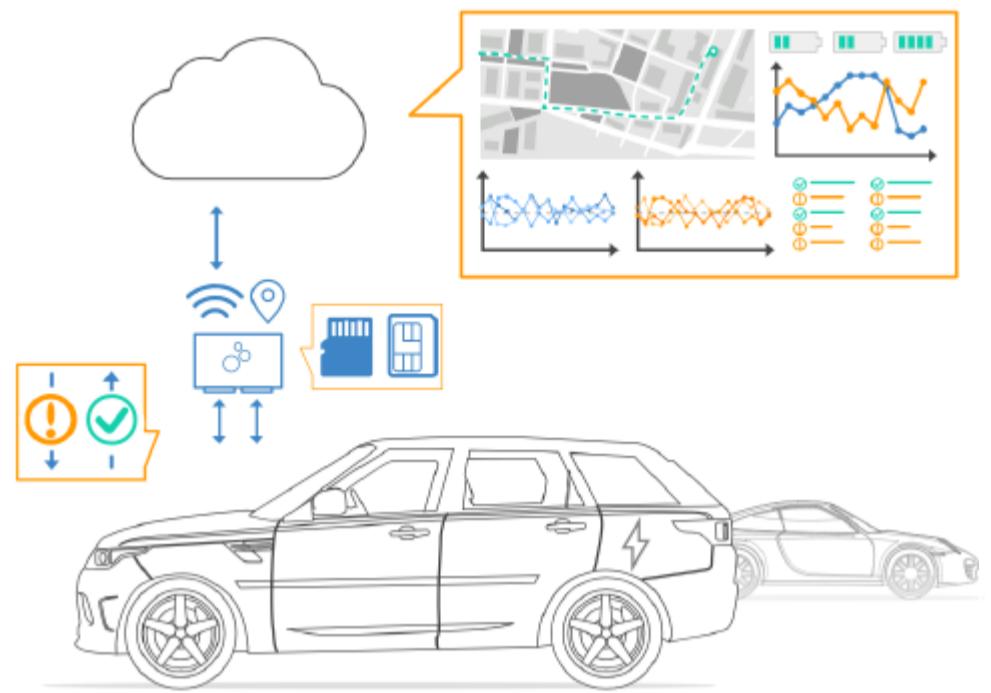
end
 field
 SID (request)
 DID
 SID (response)
 length/unused
 clock size, ST
 load (17 bytes)

<Регистрация данных UDS - приложения

В этом разделе мы описываем примеры использования для записи данных UDS.

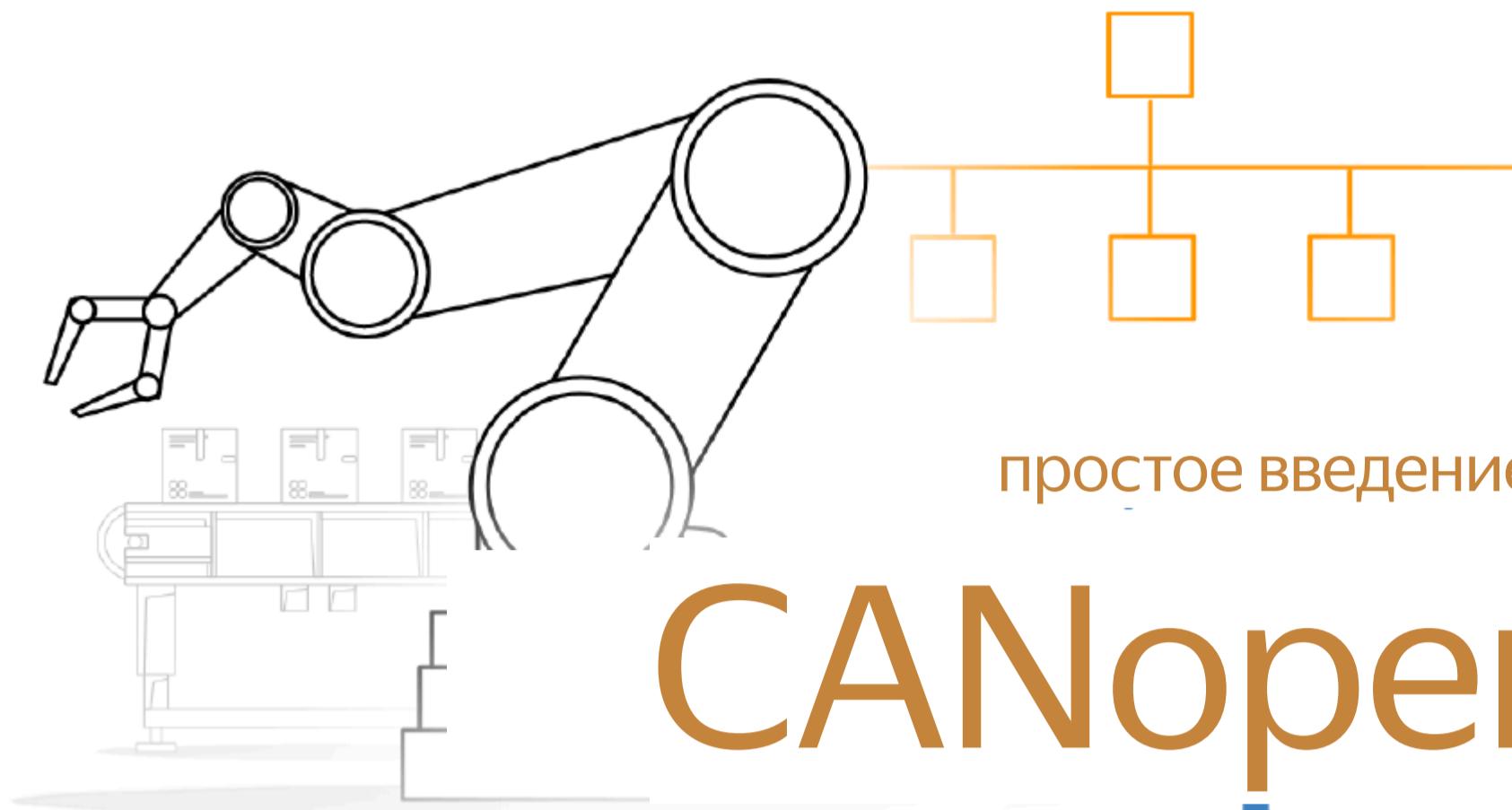
UDS telematics для прототипов электромобилей

Как производителю, вам может потребоваться получать данные о различных датчиках параметры от прототипов электромобилей во время их работы в полевых условиях. Здесь CANedge3 может быть развернут для запроса данных, например, о состоянии заряда, работоспособности, температуре и других данных путем отправки запроса UDS периодические кадры и управление потоком кадров. Данные объединяются с данными, сгенерированными внутри системы GPS / IMU, и отправляется через 3G / 4G на ваш собственный облачный сервер для анализа с помощью Векторных инструментов, Python или MATLAB.



Обучение модели прогнозируемого обслуживания

Если вы хотите реализовать прогнозируемое обслуживание в парках большегрузных автомобилей первым шагом является обычно это "обучение вашей модели". Для этого требуется собрать большие объемы обучающих данных, включая оба данные датчика (скорость, об/мин, положение дроссельной заслонки, давление в шинах и т.д.) и "результаты классификации" (неисправность / отсутствие неисправности). В одну сторону последнее можно получить, периодически запрашивая диагностические данные коды неисправностей автомобиля, предоставляя вам файлы журналов которые объединяют оба типа данных с течением времени. Вы можете использовать CANedge1 для сбора этих данных в автономном режиме на SD-карты или CANedge2 для автоматической выгрузки данных - например, когда автомобили вернутся к стационарным маршрутизаторам Wi-Fi в гаражах.



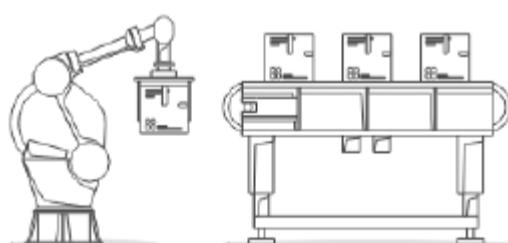
Объяснение CANopen - простое введение

В этом руководстве мы представляем CANopen с основами протокол ВКЛ. объект словарь, сервисы, SDO, PDO и master / slave узлы. CANopen может показаться сложным - поэтому для непрофессионала это руководство является наглядным введением.

Что такое CANopen?

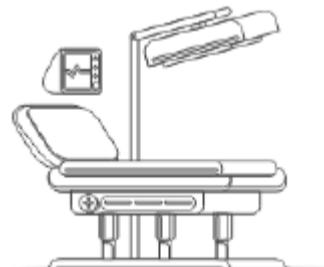
CANopen - это протокол связи на основе CAN.

Стандарт CANopen полезен, поскольку он обеспечивает готовую совместимость между устройствами (узлами), например, в промышленном оборудовании. Кроме того, он предоставляет стандартные методы настройки устройств - также после установки. Изначально CANopen был разработан для систем управления машинами, ориентированных на движение. Сегодня CANopen широко используется в управлении двигателями (шаговые двигатели / серводвигатели), а также в широком спектре других применений:



Робототехника

Автоматизированная робототехника, конвейерные ленты и другое промышленное оборудование



Медицина

Генераторы рентгеновского излучения, инъекторы для пациентов
столы и устройства для диализа



Автомобилестроение

Сельское хозяйство, железная дорога, прицепы, тяжеловесы
добыва промышленных полезных ископаемых, морской транспорт и многое другое

CANopen - протокол более высокого уровня

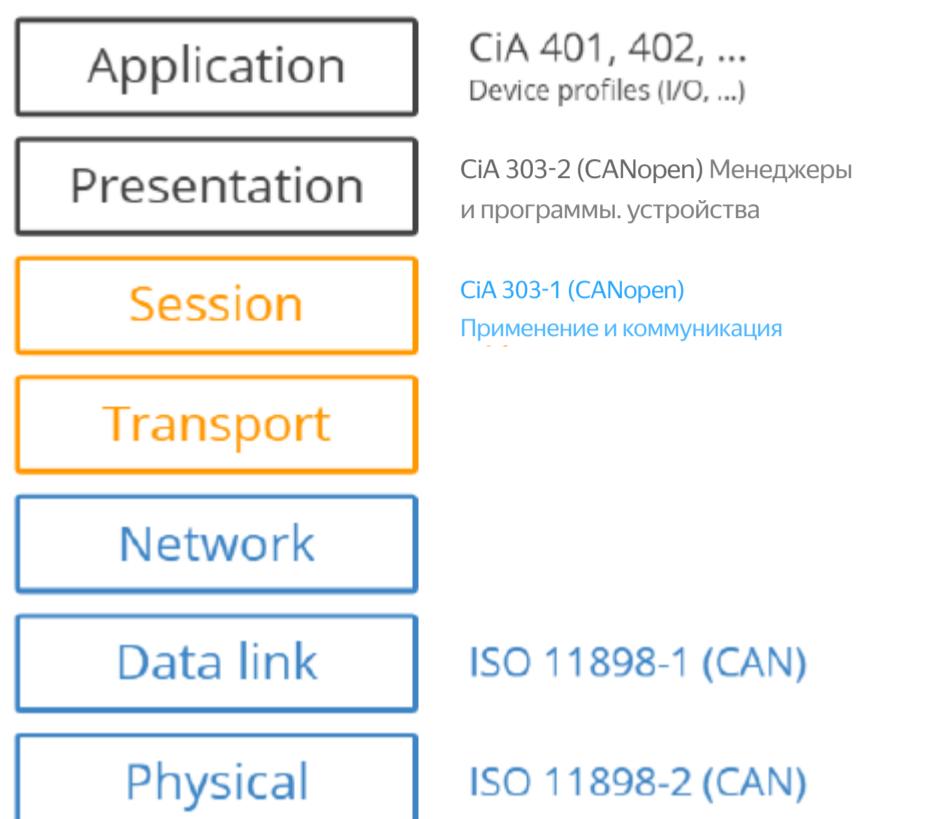
Важно понимать следующее:

CANopen - это "протокол более высокого уровня", основанный на CAN Автобус.

Это означает, что шина CAN (ISO 11898) служит в качестве "транспортного средства" (например, грузовика) для сообщений CANopen (например, контейнеров).

Вы можете просматривать CANopen из 7-слойной модели OSI.

7-уровневая модель OSI



CANopen в контексте модели OSI

Модель OSI - это концептуальная модель, стандартизирующая коммуникационные функции в различных коммуникациях Технологии. Нижние уровни описывают базовую коммуникацию (например, необработанные битовые потоки), в то время как более высокие уровни описывают такие вещи, как сегментация длинных сообщений и сервисов, таких как инициирование, индикация, ответ и подтверждение сообщений. Шина CAN представляет два низких уровня (1: физический, 2: канал передачи данных). Это означает, что CAN просто включает передачу кадров с 11-разрядным идентификатором CAN, битом удаленной передачи (RTR) и 64 битами данных (поля, относящиеся к протоколы более высокого уровня). Другими словами, шина CAN играет в CANopen ту же роль, что и, например, в протоколе J1939. Как видно выше, CANopen реализует 7-й уровень модели OSI (приложения) с помощью набора стандартов. Как часть этого, он добавляет несколько важных концепций, которые мы подробно описываем ниже. Стоит отметить, что CANopen также может быть адаптирован к другим протоколам канального уровня, отличным от CAN (например, EtherCAT, Modbus, Powerlink). Чтобы полностью понять разницу между CAN bus и CANopen, [смотрите также наш вводный урок по шине CAN](#).

CANopen FD

В этой статье мы в первую очередь сосредоточимся на CANopen на основе классического CAN. Однако стоит отметить, что по мере развертывания CAN FD CANopen FD может играть все более важную роль в качестве следующего поколения CANopen. Для получения подробной информации ознакомьтесь с обзором CAN в области автоматизации на CANopen FD.

Шесть основных концепций CANopen

Даже если вы знакомы с CAN bus и, например, J1939, CANopen добавляет ряд важных новых концепций:



Communication Models
Информационные материалы

Communication Protocols

Device States

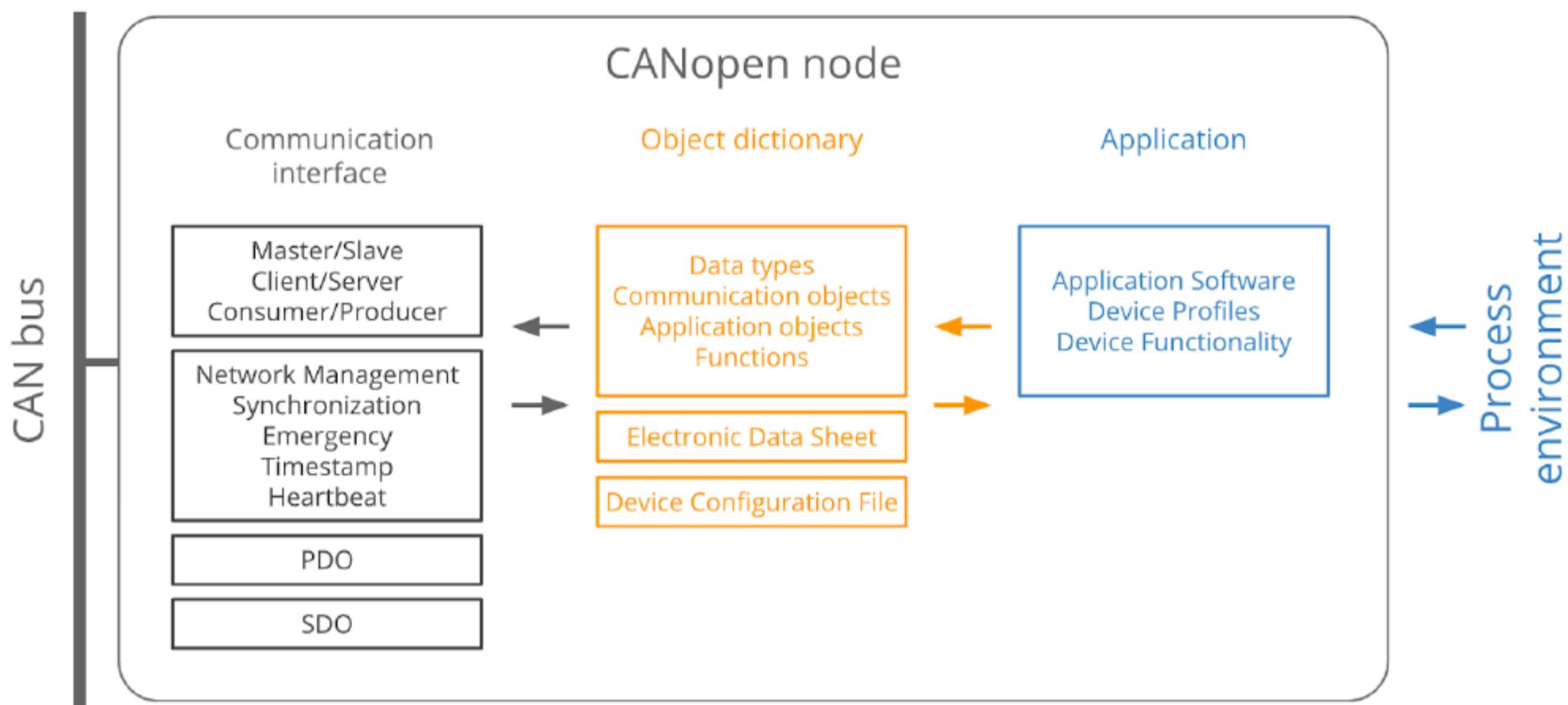
Object Dictionary

Electronic Data Sheet

Device Profiles Standards
Стандарты
описите, например, ввод-вывод

Существует 3 модели для устройства/узла связи:	протоколы используются в информационные материалы, например , настройка узлы (SDOS) или передающие данные в реальном времени (PDOs)	Устройство поддерживает разные состояния. "Главный" узел может изменять состояние "ведомого" узла - например, его сброс	У каждого устройства есть OD с записями, которые указывают, например, конфигурацию устройства. IT может быть доступен через SDOs	EDS представляет собой стандартный файл формат для OD записи, разрешающие например, сервисные инструменты для обновления устройств	модули (CiA 401) и управление движением (CiA 402) для поставщик независимость
Ведущий / ведомый, клиент/сервер и производитель/потребитель	mer				

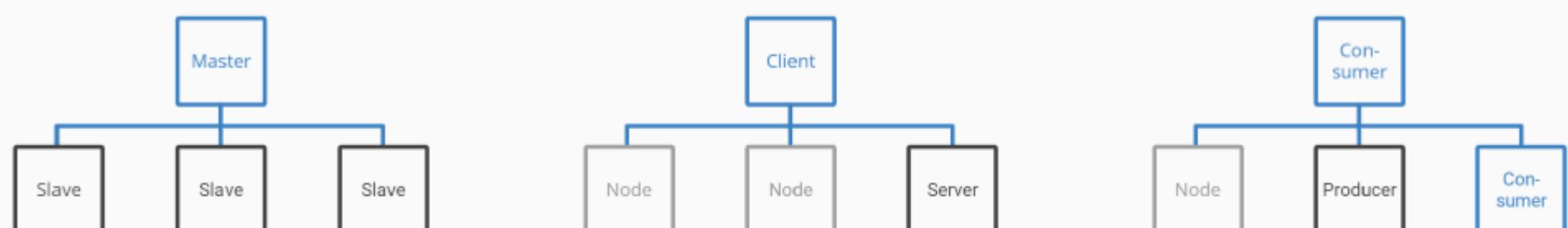
На приведенном ниже рисунке показано, как концепции CANopen связаны друг с другом, и мы подробно рассмотрим каждую из них ниже.:



Основы связи CANopen

В сети CANopen необходимо взаимодействовать нескольким устройствам. Например, при установке промышленной автоматизации у вас может быть манипулятор робота с несколькими узлами серводвигателя и узлом интерфейса управления / ПК. Для облегчения связи используются три в CANopen существуют модели, каждая из которых тесно связана с протоколами CANopen, которые мы рассмотрим вкратце. Краткое описание смотрите ниже Введение:

CANopen communication models



Master/Slave

One node (e.g. the control interface) acts as application master or host controller. It sends/requests data from the slaves (e.g. the servo motors). This is used in e.g. diagnostics or state management.

Client/Server

A client sends a data request to a server, which replies with the requested data. Used e.g. when an application master needs data from the OD of a slave. A read from a server is an "upload", while a write is

Consumer/Producer

Here, the producer node broadcasts data to the network, which is consumed by the consumer node. The producer either sends this data on request (pull model) or without a specific request (push model).

В может быть 0-127 подчиненных устройств
стандартные приложения. Обратите внимание, что в
единой сети CANopen могут
использоваться разные хост-контроллеры, совместно использующие
один и тот же уровень канала передачи
данных.

Пример сервиса: NMT

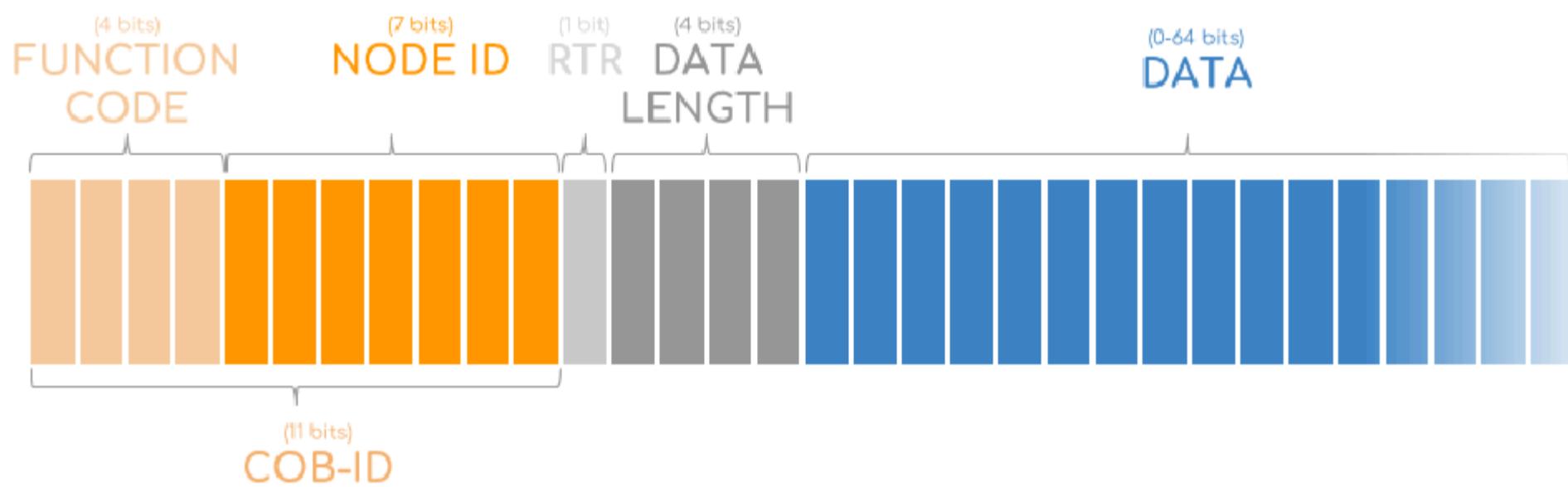
"загрузка" (терминология использует
перспективу "на стороне сервера").
Пример сервиса: SDO

пример сервиса: сердцебиение

Как видно, модели практически идентичны, но мы различаем их для согласованности терминологии.

Фрейм CANopen

Чтобы понять коммуникацию CANopen, необходимо разобрать фрейм CANopen CAN:



11-разрядный идентификатор CAN называется идентификатором объекта связи (COB-ID) и разделен на две части: По умолчанию первые 4 бита равны коду функции, а следующие 7 бит содержат идентификатор узла.

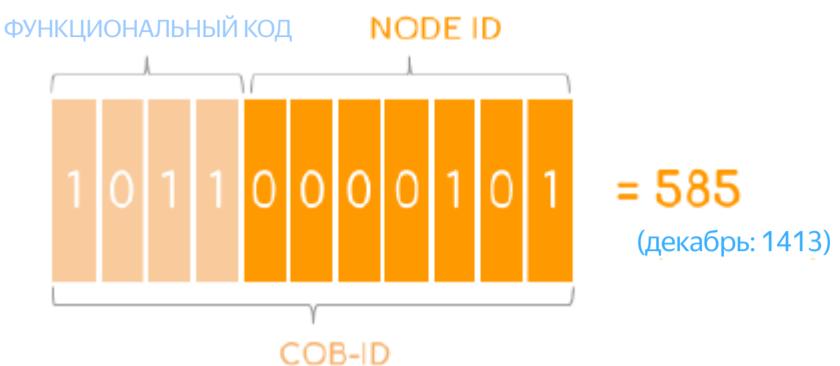
Чтобы понять, как работает COB-ID, давайте для начала рассмотрим предопределенное распределение идентификаторов, используемых в простых сетях CANopen (см. Таблицу). Примечание: Ниже мы будем ссылаться на COB-идентификаторы и идентификаторы узлов в шестнадцатеричном формате. Как очевидно, COB-идентификаторы (например, 381, 581, ...) связаны со службами связи (передают PDO 3, передают SDO, ...). Таким образом, COB-ID детализирует, какой узел является отправкой / получение данных - и какая услуга используется.

КОММУНИКАЦИОННЫЙ ОБЪЕКТ	ФУНКЦИОНАЛЬНЫЙ КОД (4 бита, bin)	ИДЕНТИФИКАТОРЫ УЗЛОВ (7 бит, bin)	СОВ-идентификаторы (шестнадцатеричный)(dec)	СОВ-идентификаторы (шестнадцатеричный)(dec)
1 НМТ		0000000	0...	
2 СИНХРОНИЗАЦИЯ	0001	0000000	80...	128
3 ВЕДУЩИЙ	0001	0000001-1111111	81 - ФФ	129 - 255
4 ВРЕМЯ	0010	0000000	100	256
5 Передача PDO 1	0011	0000001-111111118181	- 1ФФ	385 - 511
Получить PDO 1	0100	0000001-1111111 201 -	27F	513 - 639
Передающий PDO 2	0101	0000001-1111111 281 -	2FF	641 - 767
Получить PDO 2	0110	0000001-1111111 301 -	37F	769 - 895
Передающий PDO 3	0111	0000001-1111111 381 -	3FF	897 -
Получить PDO 3	1000	0000001-11111111 401 -	47F	1025 -
Передающий PDO 4	1001	0000001-11111111 481 -	4FF	1153 -
Получить PDO 4	1010	0000001-11111111 501 -	57F	1281 -
6 Передача SDO	1011	0000001-11111111 581 -	5FF	1409 -
Получение SDO	1100	0000001-11111111 601 -	67F	1537 -
7 СЕРДЦЕБИЕНИЕ	1110	0000001-11111111 701 -	77F	1793 -

Пример

Устройство CANopen с идентификатором узла 5 будет передавать SDO через 11-разрядный идентификатор CAN 585.

Это соответствует двоичному функциональному коду 1011 и идентификатор узла 5 (двоичный код: 0000101) -смотрите иллюстрацию.



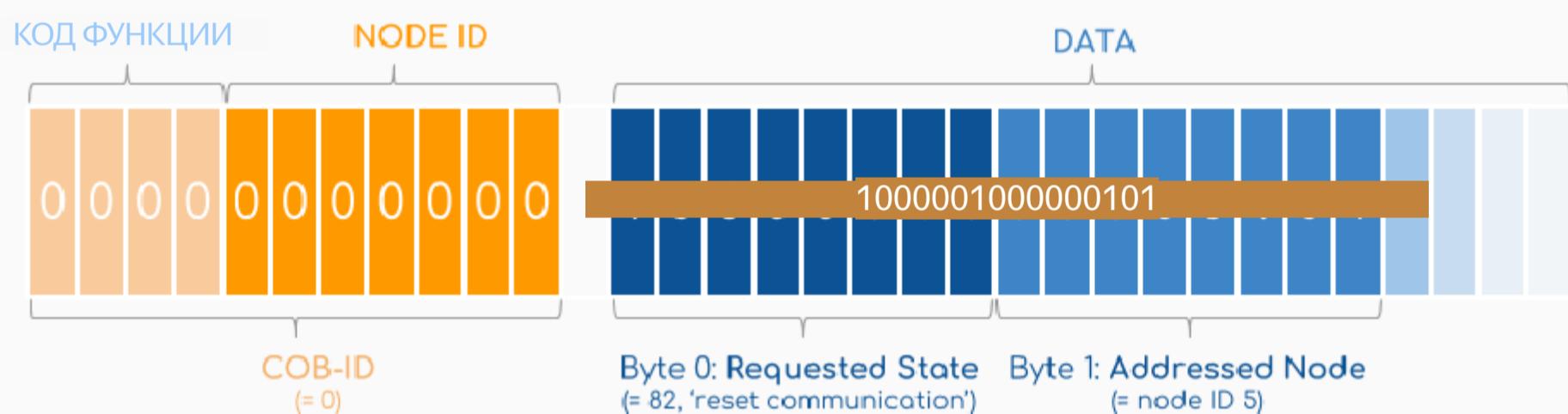
Протоколы связи CANopen / сервисы

Ниже мы кратко опишем 7 упомянутых типов сервисов, включая, как они используют 8 байтов данных кадра CAN.

1 Сетевое управление (NMT)

Что это? Служба NMT используется для контроля состояния устройств CANopen (например, предоперационного, эксплуатационного, остановлен) с помощью команд NMT (например, start, stop, reset).

Как это работает? Чтобы изменить состояние, ведущий NMT отправляет 2-байтовое сообщение с идентификатором CAN 0 (т.е. функциональный код 0 и идентификатор узла 0). Это сообщение обрабатывают все подчиненные узлы. 1-й байт данных CAN содержит запрошенное состояние, а 2-й Байт данных CAN содержит идентификатор узла целевого узла. Идентификатор узла 0 указывает широковещательную команду.



Возможные команды включают также переход в рабочее состояние (состояние 01), в остановленное состояние (состояние 02), в предоперационное состояние (состояние 80) как сбросить приложение (81) и сбросить связь (82).

Синхронизация # 2 (SYNC)

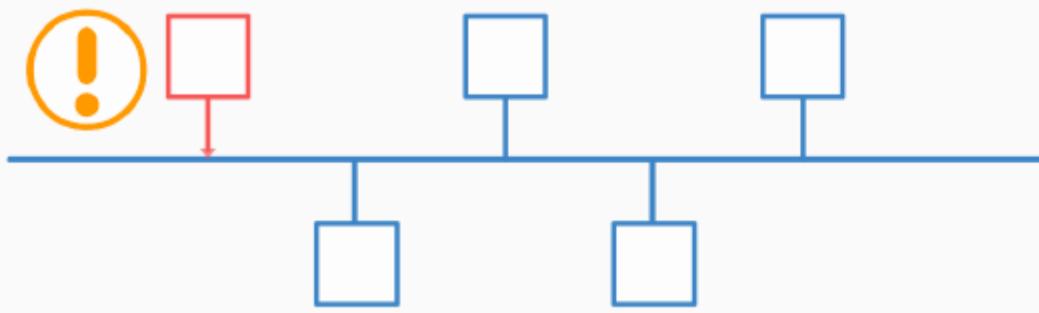
Что это? Сообщение СИНХРОНИЗАЦИИ используется, например, для синхронизации считывания входных данных и приведения в действие нескольких CANopen устройства - обычно запускаются главным приложением.

Как это работает? Мастер приложения отправляет сообщение СИНХРОНИЗАЦИИ (COB ID 080) в сеть CANopen (со счетчиком синхронизации или без него). Несколько подчиненных узлов могут быть сконфигурированы так, чтобы реагировать на СИНХРОНИЗАЦИЮ и отвечать передачей входных данных данные, полученные в одно и то же время, или путем настройки выходных данных в то же время, что и узлы, участвующие в синхронной работе. С помощью счетчика СИНХРОНИЗАЦИИ можно настроить несколько групп синхронно работающих устройств.

3 Аварийная ситуация (EMCY)

Что это? Служба экстренной помощи используется в случае, если устройство выдает неустранимую ошибку (например, сбой датчика), позволяя ему сообщить об этом остальной сети.

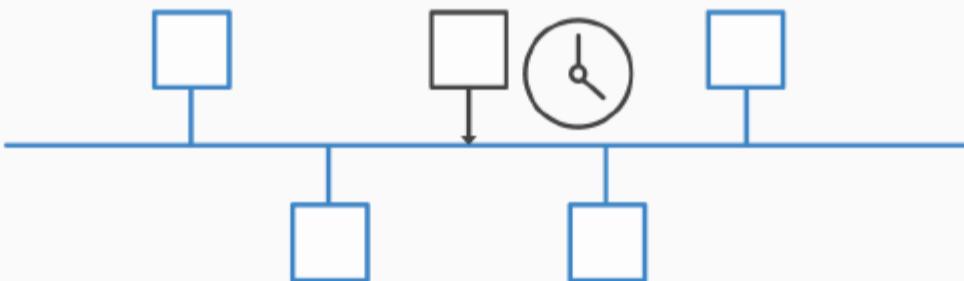
Как это работает? Затронутый узел отправляет в сеть одно сообщение EMCY (например, с COB-ID 085 для узла 5) с высоким приоритетом. Байты данных содержат информацию об ошибке, которую можно просмотреть для получения подробной информации.



4 Временная метка (TIME) [PDO]

Что это? С помощью этой службы связи можно распределять время по глобальной сети. Служба ВРЕМЕНИ содержит 6-байтовую информацию о дате и времени.

Как это работает? Мастер приложения отправляет сообщение о ВРЕМЕНИ с идентификатором CAN 100, где начальные 4 байта данных укажите время в мс после полуночи, а следующие 2 байта содержат количество дней с 1 января 1984 года.



#5 Объект обработки данных [PDO]

Что это? Служба PDO используется для передачи данных в режиме реального времени между устройствами, например, измеренных данных, таких как положение или командных данных, таких как запросы крутящего момента. В этом отношении он похож, например, на параметры широковещательных данных в J1939.

Как это работает? Мы подробно рассмотрим это ниже.

6 Объект служебных данных [SDO]

Что это? Службы SDO используются для доступа / изменения значений в словаре объектов устройства CANopen - например, когда мастеру приложения необходимо изменить определенные конфигурации устройства CANopen.

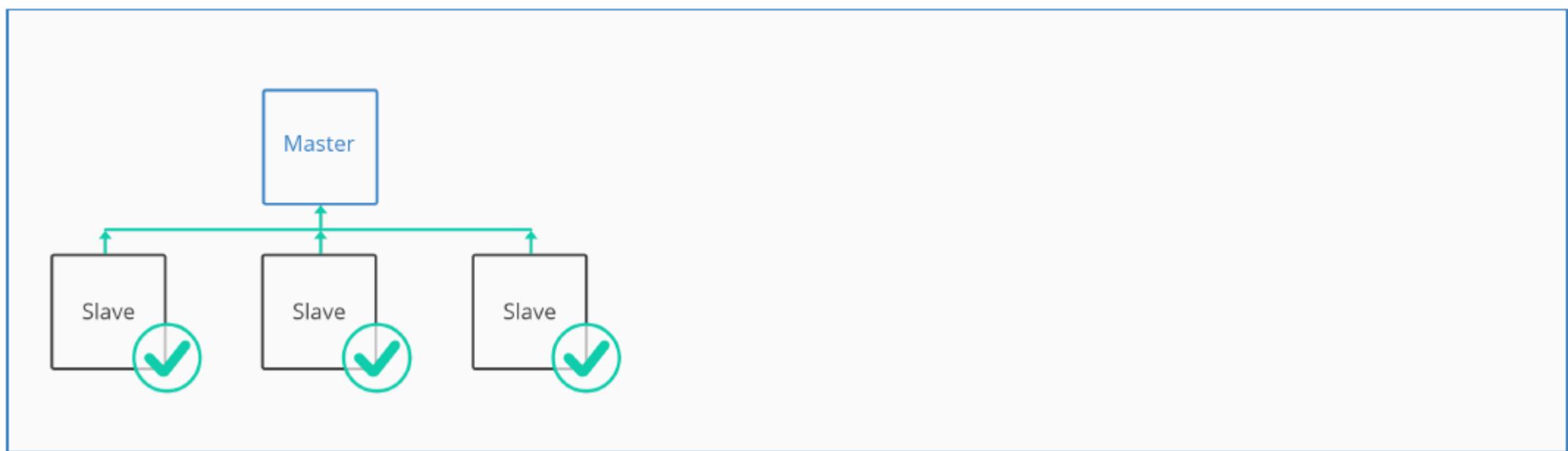
Как это работает? Мы подробно рассмотрим это ниже.

7 Мониторинг узлов (сердцебиение) [SDO]

Что это? Служба Heartbeat предназначена для двух целей: предоставления "активного" сообщения и подтверждения команды NMT.

Как это работает? Подчиненное устройство NMT периодически отправляет (например, каждые 100 мс) сообщение о пульсе (например, с идентификатором CAN 705 для узла 5) с указанием "состояния" узла в 1-м байте данных

Затем "потребитель" сообщения о сердцебиении (например, мастер NMT и, возможно, любое другое устройство) реагирует, если в течение определенного срока сообщение не получено .



Услуги PDO и SDO особенно важны, поскольку они формируют основу для большинства коммуникаций CANopen. Ниже мы подробно рассмотрим каждый из них, но сначала нам нужно представить основную концепцию CANopen: объектный словарь.

Объектный словарь CANopen

Все узлы CANopen должны иметь объектный словарь (OD) - но что это? Объектный словарь представляет собой стандартизированную структуру содержащую все параметры, описывающие поведение узла CANopen. Записи OD просматриваются с помощью 16-битного индекса и 8-битного субиндекса. Например, индекс 1008 (субиндекс 0) узла OD, совместимого с CANopen, содержит имя устройства узла.

В частности, запись в словаре объектов определяется атрибутами:

Индекс: 16-разрядный базовый адрес объекта

Имя объекта: название устройства производителя

Код объекта: массив, переменная или запись

Тип данных: например, VISIBLE_STRING, или UNSIGNED32, или имя записи

Доступ: rw (чтение / запись), ro (только для чтения), wo (только для записи)

Категория: указывает, является ли этот параметр обязательным / необязательным (M / O)

Стандартные разделы OD

Объектный словарь разделен на стандартные разделы, где некоторые записи обязательны, а другие доступны полностью настраиваемый. Важно отметить, что записи OD устройства (например, ведомого) могут быть доступны другому устройству (например, ведущему) через CAN используя, например, SDOs. Например, это может позволить ведущему приложению изменять, регистрирует ли подчиненный узел данные через определенный входной датчик - или как часто подчиненный узел отправляет сердцебиение.

ИНДЕКС OD (16 бит, шестнадцатеричный)Описание

	Зарезервировано
0001 - 025F	Типы данных
0260 - 0FFF	Зарезервировано
1000 - 1FFF	Область объекта связи
2000 - 5FFF	Область, специфичная для производителя
6000 - 9FFF	Область, специфичная для профиля устройства
A000 - BFFF	Область, специфичная для профиля интерфейса
C000 - FFFF	Зарезервировано

Ссылка на электронный паспорт данных и файл конфигурации устройства

Чтобы понять OD, полезно взглянуть на "удобочитаемую форму": Электронный паспорт данных и устройство
Файл конфигурации.



Объектный словарь



Electronic Data Sheet



Файл конфигурации устройства

Электронный паспорт данных (EDS)

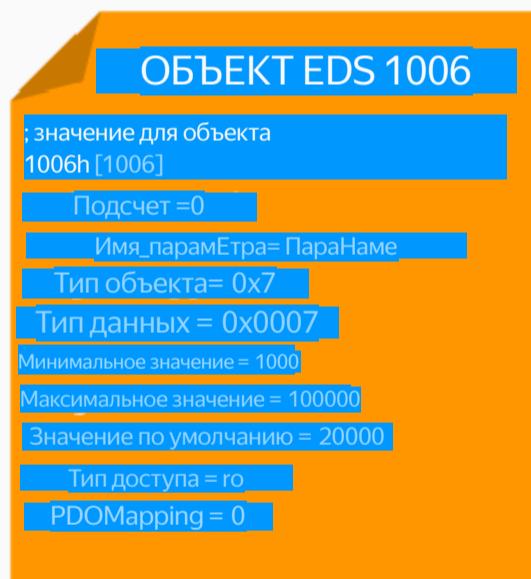
На практике настройка / управление сложными сетями CANopen будет осуществляться с использованием соответствующих программных средств. Чтобы упростить это, стандарт CiA 306 определяет удобный для чтения человеком (и машиной) формат файла INI, действующий как "шаблон" для OD устройства - например, "ServoMotor3000". Этот EDS обычно предоставляется поставщиком и содержит информацию обо всех объектах устройства (но не о значениях).

Файл конфигурации устройства (DCF)

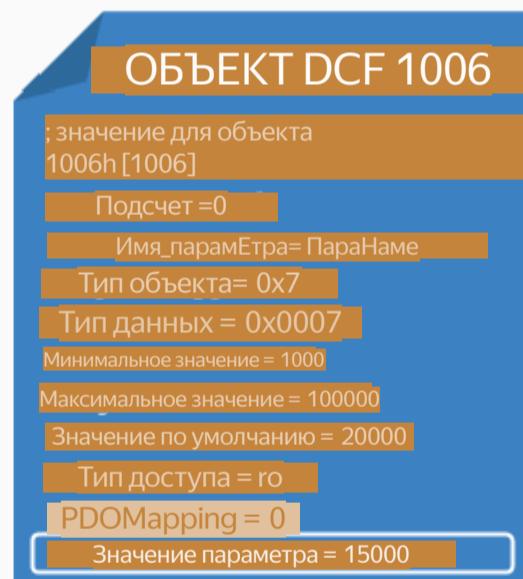
Предположим, что завод приобрел ServoMotor3000 для интеграции в свою конвейерную ленту. При этом оператор редактирует ЭЦП устройства и добавляет определенные значения параметров и/или изменяет названия каждого объекта, описанного в ЭЦП. При этом таким образом, оператор эффективно создает то, что известно как файл конфигурации устройства (DCF). С учетом этого ServoMotor3000 готов к интеграции в конкретную сеть CANopen на месте установки.

Пример EDS и DCF

Просмотр реальных примеров EDS / DCF - один из лучших способов по-настоящему понять объектный словарь CANopen - см. например, разница между записью объекта EDS и DCF ниже. Мы рекомендуем ознакомиться со стандартом CiA 306, чтобы получить более глубокое представление об OD, EDS и DCF на практических примерах.



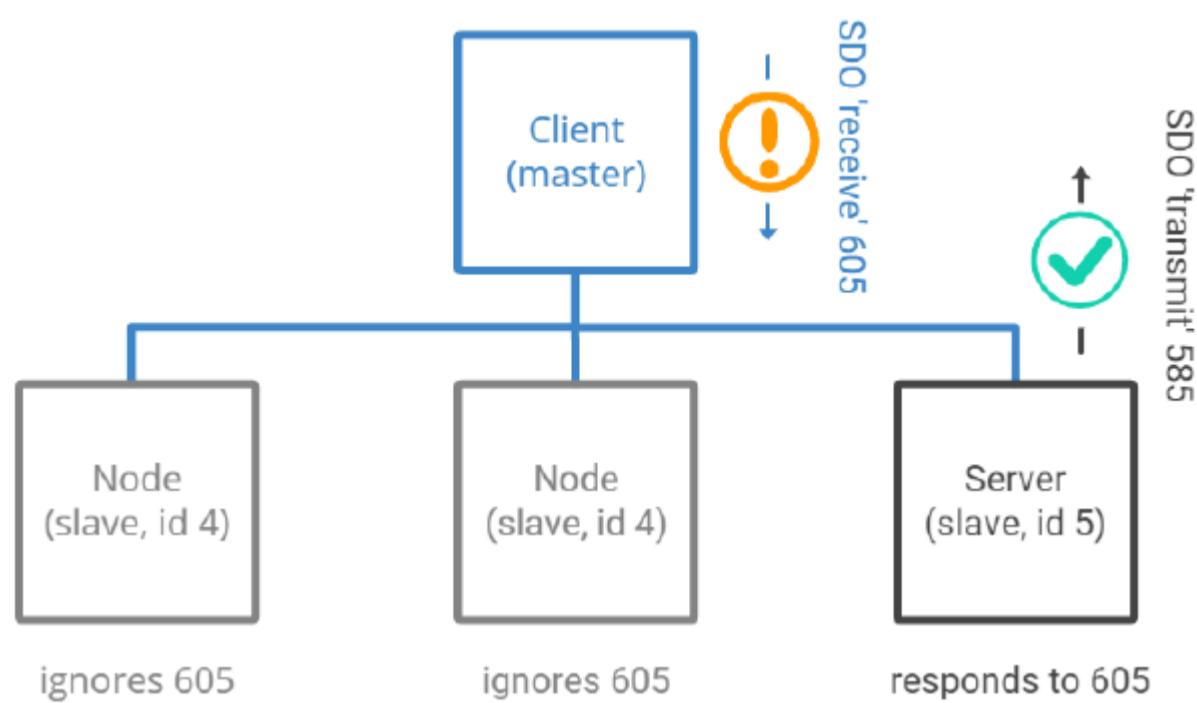
Комментарии
Поле для комментария
16-разрядный индекс объекта
Количество подиндексов
Индекс OD типа данных
Минимальный (если применимо)
Максимальный (если применимо)
Используется, если значение не указано
Доступ: ro = только для чтения PDO
можно отображать? 0 = n_O
Конкретное значение параметра



Как упоминалось, DCF обычно создается при интеграции устройства. Однако часто бывает необходимо прочитать и / или измените значения объектов узла после первоначальной настройки - именно здесь вступает в действие служба CANopen SDO.

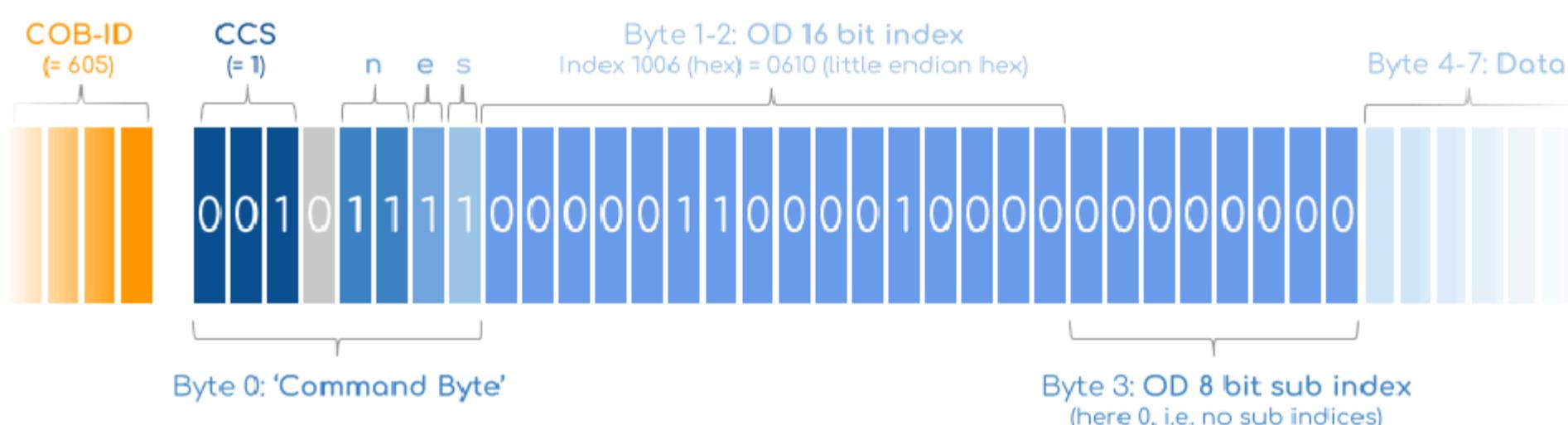
SDO - настройка сети CANopen

Что такое служба SDO? Служба SDO позволяет узлу CANopen считывать / редактировать значения объектного словаря другого узла по сети CAN. Как упоминалось в разделе "Модели связи", службы CANopen SDO используют "клиент / сервер" поведение. В частности, "клиент" SDO инициирует обмен данными с одним выделенным "сервером" SDO. Целью может быть обновление записи OD (называемой "загрузка SDO") или чтение записи ("Загрузка SDO"). В простых сетях master / slave узел с функциями NMT master действует как клиент для всех подчиненных узлов NMT, считающих или записывающих данные в свои ODS.



Пример: Загрузка SDO клиентского узла.

Клиентский узел может инициировать загрузку SDO на узел 5, передав приведенный ниже кадр CAN, который запустит узел 5 (и будет проигнорирован другими узлами, см. Иллюстрацию выше). "Получение" SDO (т. е. запрос) МОЖЕТ выглядеть следующим образом:



Объясняются переменные сообщения SDO

Во-первых, COB-ID 605 отражает использование "приема SDO" (COB-ID 600 + идентификатор узла).

CCS (спецификатор клиентской команды) - это тип передачи (например, 1: Загрузка, 2: Выгрузка).

н является ли #байт в байтах данных 4-7, которые не содержат данных (допустимо, если e & s установлено)

Если установлено, e указывает на "ускоренную передачу" (все данные находятся в одном кадре CAN)

Если установлено, s указывает, что размер данных указан в н

Индекс (16 бит) и субиндекс (8 бит) отражают адрес OD, к которому необходимо получить доступ.

Наконец, байты 4-7 содержат данные, которые необходимо загрузить на узел 5.

Комментарии к примерам CANopen SDO

Как только главный (клиент) отправляет кадр CAN, подчиненный узел 5 (сервер) отвечает через "SDO transmit" с COB-ID 585. Ответ содержит индекс/субиндекс и 4 пустых байта данных. Естественно, если клиентский узел запросил вместо загрузки (т. Е. считывания данных с узла 5 OD) узел 5 будет отвечать соответствующими данными, содержащимися в байтах 4-7. Несколько комментариев:

Как видно, каждый SDO использует 2 идентификатора, создавая "канал SDO"

Пример упрощен, поскольку он "ускорен" (данные содержатся в 4 байтах).

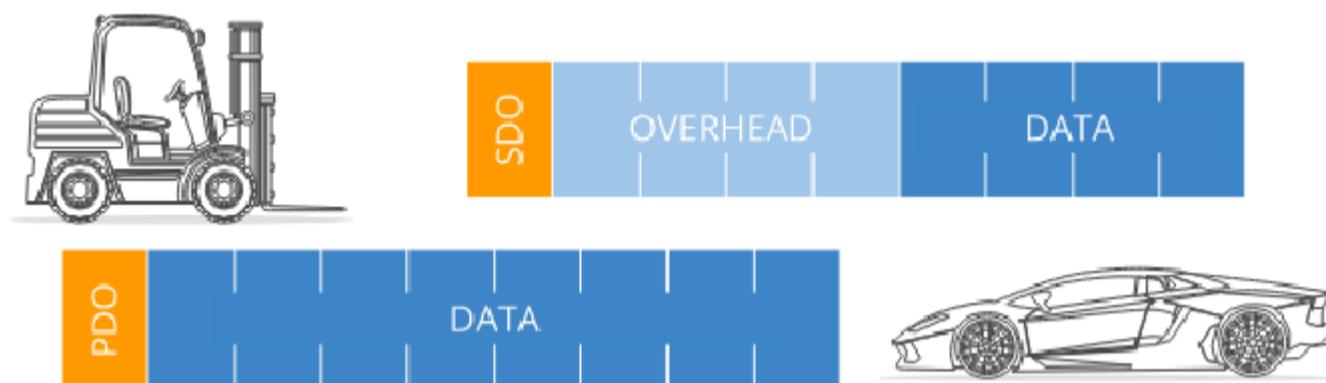
Для сценариев с большими данными можно использовать сегментацию / передачу блоков SDO.

SDO являются гибкими, но сопряжены с большими накладными расходами, что делает их менее идеальными для обработки оперативных данных в режиме реального времени. Вот где Поступает PDO.

PDO - управление сетью CANopen

Прежде всего: что такое служба CANopen PDO? Служба CANopen PDO используется для эффективного обмена данными в режиме реального времени рабочие данные между узлами CANopen. Например, PDO будет передавать данные о давлении от датчика давления - или данные о температуре с датчика температуры. Но подождите: разве служба SDO не может просто сделать это?

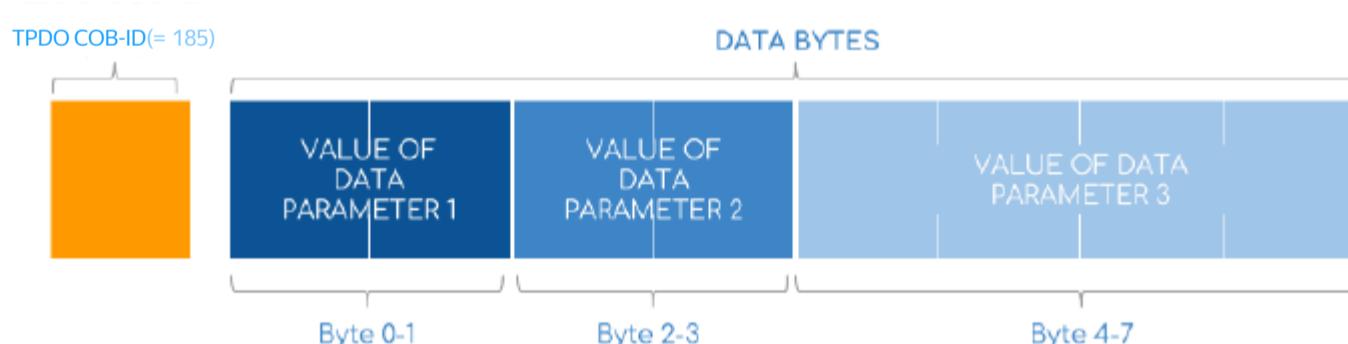
Да, в принципе, для этого можно использовать службу SDO. Однако один ответ SDO может содержать только 4 байта данных из-за накладных расходов (командный байт и адреса OD). Далее, предположим, что главному узлу требуются два значения параметра (например, "SensTemp2" и "Torque5") из узла 5 - чтобы получить это через SDO, потребовалось бы 4 полных кадра CAN (2 запроса, 2 ответа).



В отличие от этого, сообщение PDO может содержать 8 полных байт данных - и оно может содержать несколько значений параметров объекта в пределах одного кадра. Таким образом, то, что потребовало бы по крайней мере 4 кадра с SDO, потенциально может быть выполнено с помощью 1 кадра в PDO. Обслуживание. PDO часто рассматривается как наиболее важный протокол CANopen, поскольку он передает основную часть информации.

Как работает служба CANopen PDO?

Для PDO, используется терминология потребителя / производителя. Таким образом, производитель "производит данные", которые он передает "потребитель" (master) использует передающий PDO (TPDO). И наоборот, он может получать данные от потребителя через PDO приема (RPDO). Узлы производителя могут, например, быть сконфигурированы для ответа на триггер СИНХРОНИЗАЦИИ, транслируемый потребителем каждые 100 мс. Затем узел 5 может, например, транслировать ниже, передавать PDO с COB-ID 185:



Обратите внимание, как байты данных упакованы со значениями 3 параметров. Эти значения отражают данные в реальном времени о конкретных записях OD узел 5. Узлы, которые используют эту информацию (потребители), конечно, должны знать, как интерпретировать данные PDO байты.

Служба PDO по сравнению с PGN и SPN J1939.

Разве служба PDO не похожа на PGN и SPN J1939? Да, в какой-то степени это похоже на то, как конкретный J1939 группа параметров (PG) будет содержать несколько SPN / сигналов (также известных как параметры данных) в 8 байтах данных. J1939 МОЖЕТ фрейму не нужно тратить байты данных на "декодирование" информации, потому что это известно соответствующим узлам (и внешними инструментами, например, через файлы J1939 DBC или стандарты J1939 PDF). Проблема в том, что в CANopen эти "PDO" сопоставления часто настраиваются и могут быть изменены во время создания DCF и / или через службу SDO. Для более подробной информации о PDO смотрите в этой статье (стр. 5).

Ведение журнала данных CANopen - примеры использования

Поскольку CANopen - это протокол на основе CAN, можно записывать raw Кадры CANopen с использованием регистратора данных шины CAN. В качестве примера, CANedge позволяет записывать данные CANopen на SD-карту объемом 8-32 ГБ.

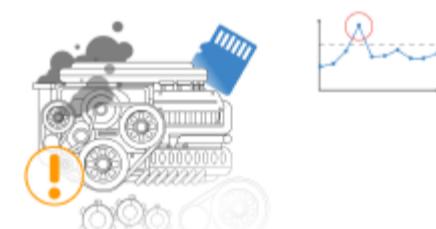
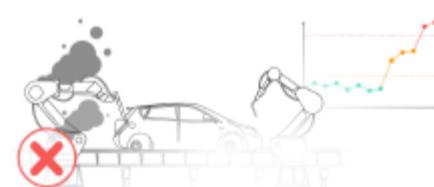
Просто подключите его к своему приложению, чтобы начать ведение журнала - и обработать данные с помощью бесплатного программного обеспечения / API. Узнайте больше!



Декодирование необработанных данных с помощью файлов CANopen DBC

Поскольку CANopen основан на шине CAN, вы можете хранить свои правила декодирования CANopen в стандартизированной базе данных CAN формат, файл CAN DBC. Благодаря этому вы можете напрямую декодировать необработанные данные CANopen в программном обеспечении с открытым исходным кодом / инструментах API для CANedge, включая графический интерфейс asammfdf, информационные панели телематики и инструменты Python API. Обратите внимание, что у вас может не быть файла CANopen DBC, который легко доступен, но ваши правила декодирования фреймов CANopen могут храниться в вашем EDS / DCF или в формате PDF. В таком случае вы можете начать с нашего вступления к DBC, чтобы узнать, как создать файл CANopen DBC с нуля используя различные бесплатные инструменты редактирования.

Решения, подобные CANedge, позволяют использовать несколько вариантов ведения журнала CANopen:



Ведение журнала узла CANopen данные

Как правило, ведение журнала
Могут использоваться данные CANopen
например, для анализа оперативных
[данных. Регистраторы Wi-Fi CAN могут](#)
также использоваться, например, для
беспроводных SDOS

[Узнать больше](#)

Складской парк Руководство

CANopen часто используется в
вилочных погрузчиках для электромобилей/AGV на
складах, где мониторинг, например
SoC помогает
сократить количество поломок и
увеличить срок службы батареи

[Узнать больше](#)

Прогнозирующее техническое обслуживание

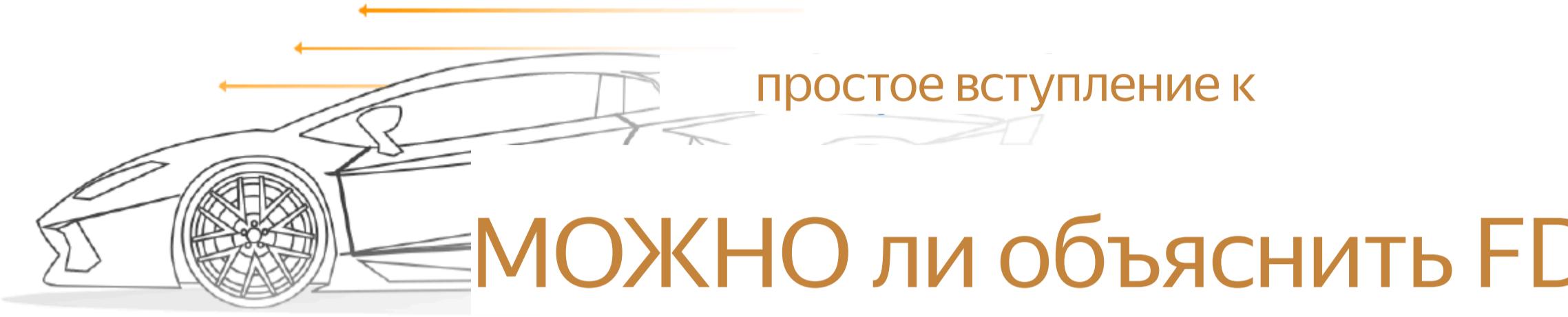
Промышленное оборудование может
мониторинг осуществляется с помощью IIoT
CAN регистраторы в облаке для
прогнозирования и предотвращения
сбоев на основе
данных CANopen

[Узнать больше](#)

Диагностика оборудования черный ящик

Регистратор CAN может служить в качестве
"черного ящика" для промышленного
оборудования, предоставляющего данные
наприимер, при возникновении спорных ситуаций или редких случаях
диагностика проблем

[Узнать больше](#)



МОЖНО ли объяснить FD - Простое вступление

В этом руководстве мы представляем CAN FD (CAN Flexible Data-rate) - вкл. Кадры CAN FD, накладные расходы и эффективность, пример приложения и варианты использования ведения журнала.

Почему CAN FD?

Протокол CAN существует с 1986 года и пользуется популярностью: сегодня практически любая движущаяся машина использует CAN - будь то легковые автомобили, грузовики, лодки, самолеты или роботы.

Но с развитием современных технологий "Классический" протокол CAN (официальный термин, используемый в ISO 11898-1: 2015) подвергается давлению.:

Расширение функциональных возможностей транспортных средств приводит к резкому увеличению объема данных

Пропускная способность сетей все чаще ограничивается 1 Мбит / с

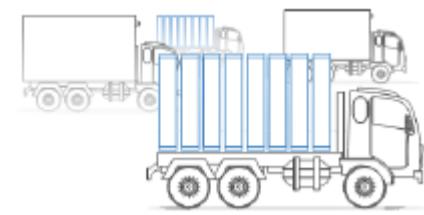
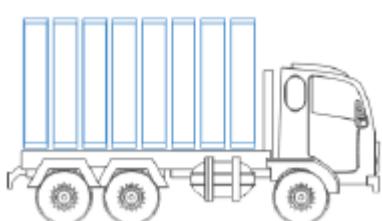
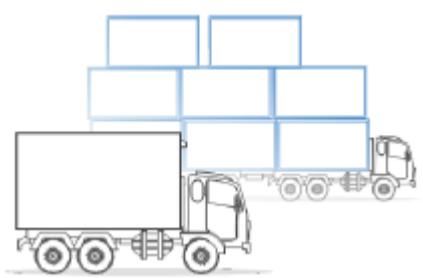
Чтобы справиться с этим, OEM-производители создают сложные и дорогостоящие обходные пути

В частности, классический CAN сталкивается со значительными накладными расходами (> 50%), поскольку каждый кадр данных CAN может содержать только 8 данных байт. Кроме того, скорость сети ограничена 1 Мбит / с, что ограничивает реализацию функций обработки данных. CAN FD решает эти проблемы, делая его перспективным.

Что такое CAN FD?

Протокол CAN FD был предварительно разработан Bosch (совместно с отраслевыми экспертами) и выпущен в 2012 году. Он был улучшен с помощью стандартизации и теперь соответствует стандарту ISO 11898-1: 2015. Оригинальная версия Bosch CAN FD (отличная от ISO CAN FD) несовместима с ISO CAN FD.

CAN FD предлагает четыре основных преимущества:



1 Увеличенная длина

CAN FD поддерживает до 64 байт данных на кадр данных по сравнению с 8 байтами данных для Классической CAN. Это снижает нагрузку на протокол и приводит к повышению эффективности протокола.

2 Увеличенная скорость

CAN FD поддерживает двухразрядную скорость передачи данных: номинальная (арбитражная) скорость передачи данных ограничена до 1 Мбит / с, как в классической CAN - и скорость передачи данных в битах, которая зависит от сети топологии/приемопередатчиков. В на практике скорость передачи данных может быть увеличена до 5 Мбит / с.

3 Повышение надежности

CAN FD использует улучшенную проверку циклического резервирования (CRC) и "защищенный счетчик битов содержимого", которые снижают риск необнаруженных ошибок. Это например, жизненно важный для обеспечения безопасности такие приложения, как транспортные средства и промышленная автоматизация.

№ 4 Плавный переход

CAN FD и классический CAN смешивать можно только ECU при определенных условиях. Это позволяет осуществлять постепенное внедрение CAN FD узлов, что значительно сокращает затраты и сложность для OEM-производителей.

На практике CAN FD может повысить пропускную способность сети в 3-8 раз по сравнению с классическим CAN, создавая простое решение для увеличения объема данных.

Как работает CAN FD?

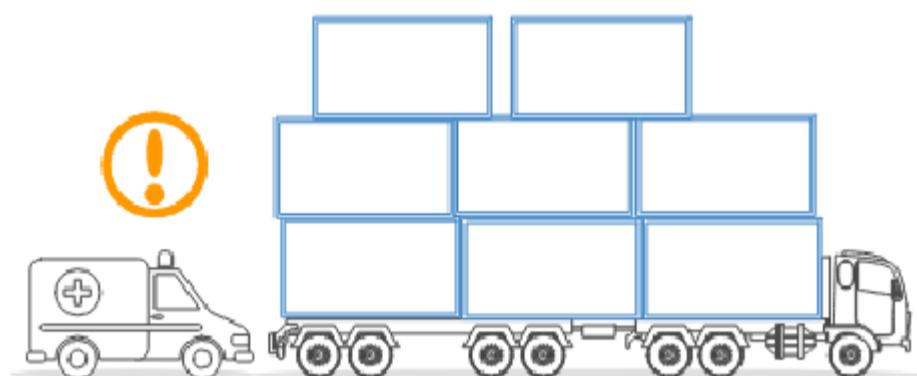
Итак, CAN FD кажется довольно простым: Ускорить передачу данных и поместить больше данных в каждое сообщение, верно? На самом деле это не так просто. Ниже мы описываем основные проблемы, которые пришлось решить решению CAN FD .

Две ключевые проблемы

Прежде чем рассматривать фрейм данных CAN FD, важно понять две основные части классического CAN, которые мы хотим поддерживать:

1 Избегайте критических задержек сообщений

Почему бы просто не упаковать классические фреймы CAN объемом 64 байта данных? Это снизило бы накладные расходы и упростило интерпретацию сообщений. Однако, если скорость передачи данных равна без изменений это также заблокировало бы шину CAN на более длительный срок, потенциально задерживая критически важные данные с более высоким приоритетом кадры.



2 Поддерживайте практическую длину проводов CAN.

Таким образом, требуется большая скорость для отправки большего количества данных за Сообщение. Но почему бы не ускорить все сообщение CAN (а не только этап передачи данных)? Это связано с "арбитражем": если 2+ узла передают данные одновременно, арбитраж определяет, какой узел имеет приоритет. "Победитель" продолжает отправку (без задержки), в то время как другие узлы "отключаются" во время передачи данных.

Что касается времени передачи данных.

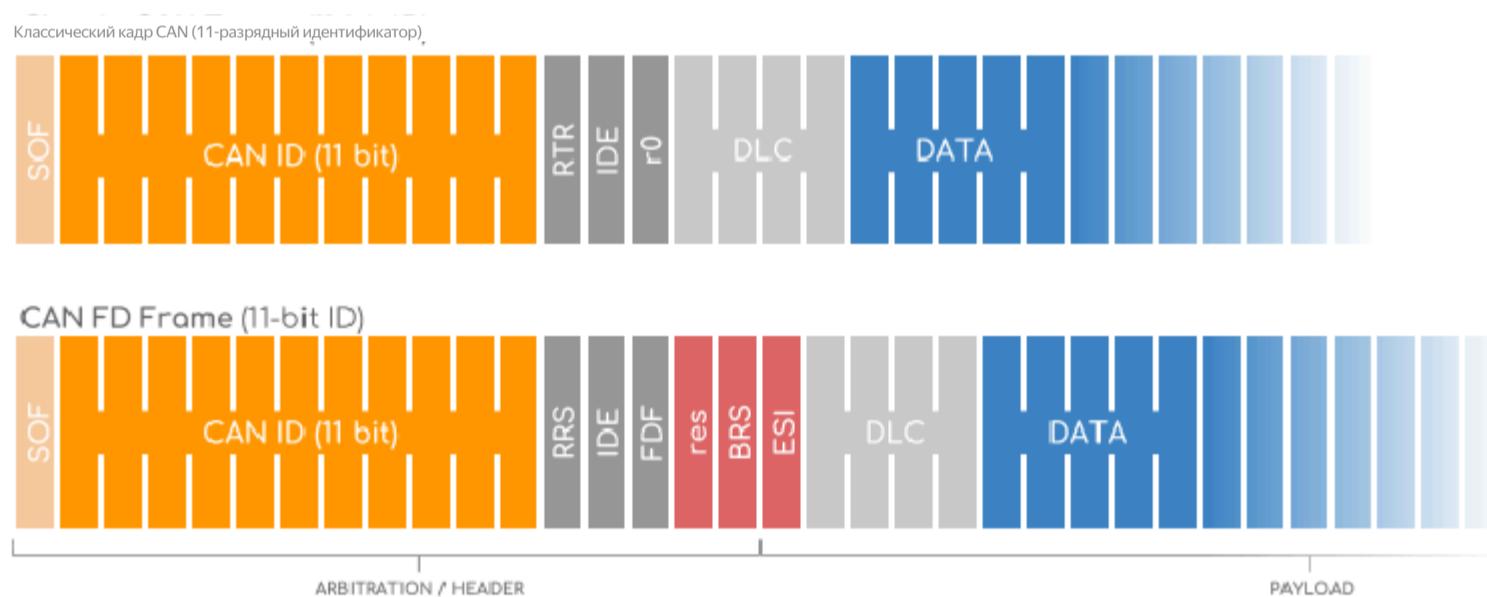
Во время арбитража "битовое время" обеспечивает достаточную задержку между каждым битом, чтобы позволить каждому узлу в сети отреагировать. Чтобы быть уверенным, что каждый узел будет достигнут в течение установленного времени, сеть CAN, работающая со скоростью 1 Мбит / с, должна иметь максимальную длину 40 метров (на практике 25 метров). Ускорение арбитражного процесса привело бы к сокращению максимальной продолжительности до неприемлемого уровня.

С другой стороны, после арбитража остается "пустое шоссе", обеспечивающее высокую скорость передачи данных (когда только один узел управляет шинами). Перед слотом подтверждения - когда несколько узлов подтверждают правильность приема кадра данных - скорость необходимо снизить до номинальной скорости передачи данных.

Короче говоря, необходимо найти способ увеличивать скорость только во время передачи данных.

Решение: Кадр CAN FD

Протокол CAN FD вводит скорректированный кадр данных CAN, позволяющий использовать дополнительные байты данных и гибкую скорость передачи данных. Ниже мы сравниваем 11-разрядный классический кадр CAN и 11-разрядный кадр CAN FD (также поддерживается 29-разрядный код):



Ниже мы шаг за шагом рассмотрим различия:

Объяснены поля кадра CAN FD

RTR и RRS: Запрос удаленной передачи (RTR) используется в классическом CAN для идентификации кадров данных и соответствующие удаленные кадры. В CAN FD удаленные фреймы вообще не поддерживаются - замена удаленного запроса (RRS) всегда доминирует (0).

r0 против FDF: в классическом CAN r0 является зарезервированным и доминирующим (0). В CAN FD он называется FDF и является рецессивным (1). После бита r0 / FDF протокол CAN FD добавляет "3 новых бита". Обратите внимание, что узлы, которые не поддерживают CAN FD, выдают кадр ошибки после бита FDF.

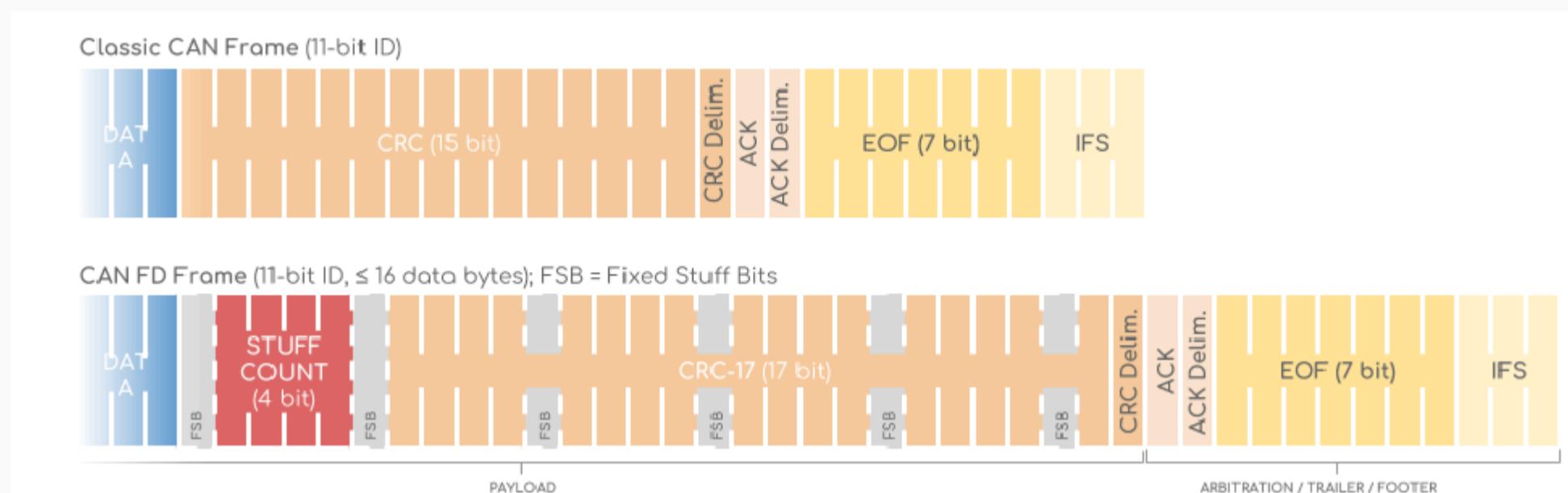
res: Этот новый зарезервированный бит играет ту же роль, что и r0, т. е. В будущем ему может быть присвоено значение рецессивного (1) для обозначения нового протокола.

BRS: Переключатель скорости передачи данных (BRS) может быть доминирующим (0), что означает, что кадр данных CAN FD отправляется со скоростью арбитража (т.е. максимум до 1 Мбит/с). Установка значения в рецессивное (1) означает, что оставшаяся часть кадра данных отправляется с более высокой скоростью передачи данных (до 5 Мбит/с).

ESI: бит индикатора состояния ошибки (ESI) по умолчанию является доминирующим (0), т.е. "ошибка активна". Если передатчик становится "ошибкой" пассивным (1), чтобы указать, что он находится в пассивном режиме ошибки.

DLC (bin)	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111			
DLC (декабрь)	1...	2...	3...	4...	5...	6...	7	8	9					10	11	12	13	14	15
Классическая БАНКА	0	1	2	3	4	5	6	7	8	8	8	8	8	8	8	8	8	8	
МОЖЕТ FD		2...	3...	4...	5...	6...	7	8		12	16	20	24	32	48		64...		

DLC: Как и в классическом CAN, DLC CAN FD имеет размер 4 бита и обозначает количество байтов данных в кадре. Указанное выше в таблице показано, как два протокола последовательно используют DLC до 8 байт данных. Для поддержки 4-разрядного DLC CAN FD использует остальные 7 значений от 9 до 15 для обозначения количества используемых байтов данных (12, 16, 20, 24, 32, 48, 64).



SBC: количество битов заполнения (SBC) предшествует CRC и состоит из 3 битов, закодированных серым цветом, и бита четности. Исправлено следующее Бит Stuff-Bit можно рассматривать как второй бит четности. SBC добавлен для повышения надежности связи.

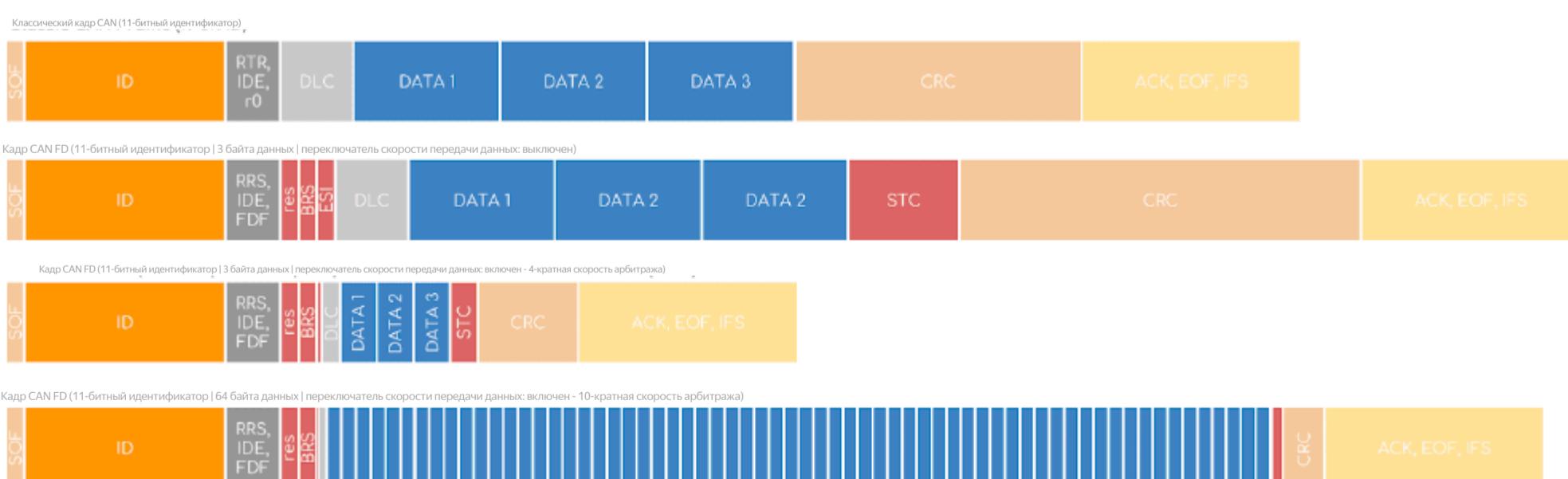
CRC: проверка циклической избыточности (CRC) составляет 15 бит в классическом CAN, в то время как в CAN FD она составляет 17 бит (до 16 данных байт) или 21 бит (для 20-64 байт данных). В классическом CAN в CRC может быть от 0 до 3 бит заполнения, в то время как в CAN FD есть всегда есть четыре фиксированных исходных бита для повышения надежности связи.

ACK: фаза передачи данных (она же полезная нагрузка) кадра данных CAN FD заканчивается на бите ACK, который также отмечает конец потенциальное увеличение скорости передачи данных.

Накладные расходы и эффективность данных может ФД и может

Как видно, дополнительной функциональности можете ФД добавляет много лишних битов и Классическая можете - как это может привести к менее Накладные расходы?

В Ответ в том, что это не так -смотрите Приведенную Ниже визуализацию классического CAN против CAN FD для 3 байт данных. Фактически, эффективность CAN FD не превышает классического CAN до тех пор, пока не превысит 8 байт данных. Однако при переходе к 64 байтам данных байт, эффективность может варьироваться от ~ 50% до ~ 90% (подробнее об этом ниже).



Need for speed: включение переключения скорости передачи данных

Как уже упоминалось, отправка 64 байт данных с обычной скоростью привела бы к блокировке шины CAN, что снизило бы производительность в реальном времени. Для решения этой проблемы можно включить переключение скорости передачи данных, чтобы разрешить отправку полезной нагрузки с более высокой скоростью по сравнению со скоростью арбитража (например, 5 Мбит / с против 1 Мбит / с). Выше мы наглядно визуализировали эффект для сценариев с 3 байтами данных и 64 байтами данных. Обратите внимание, что более высокая скорость применяется к разделу кадра данных, начинающемуся с бита BRS и заканчивающемуся разделителем CRC. Кроме того, большинство сегодня транспортные средства используют скорость 0,25-0,5 Мбит / с., это означает, что при скорости 5 Мбит / с CAN FD увеличит скорость передачи полезной нагрузки в 10 раз.

Смешивание классических узлов CAN и CAN FD

Как упоминалось ранее, классические узлы CAN и CAN FD могут быть смешаны при определенных условиях. Это позволяет использовать поэтапный переход на CAN FD вместо необходимости переключать все блоки управления за один раз. Существуют два сценария.:

100% Система CAN FD: Здесь контроллеры CAN FD могут свободно смешивать классические кадры данных CAN и CAN FD FD.

Некоторые узлы являются устаревшими классическими узлами CAN: здесь контроллеры CAN FD могут переключаться на классическую связь CAN, чтобы избежать реакции на фреймы ошибок от классических узлов CAN. Кроме того, во время, например, перепрошивки ECU, классические узлы CAN могут быть отключены, чтобы разрешить временный переход на связь CAN FD.

Какова максимальная скорость передачи данных CAN FD?

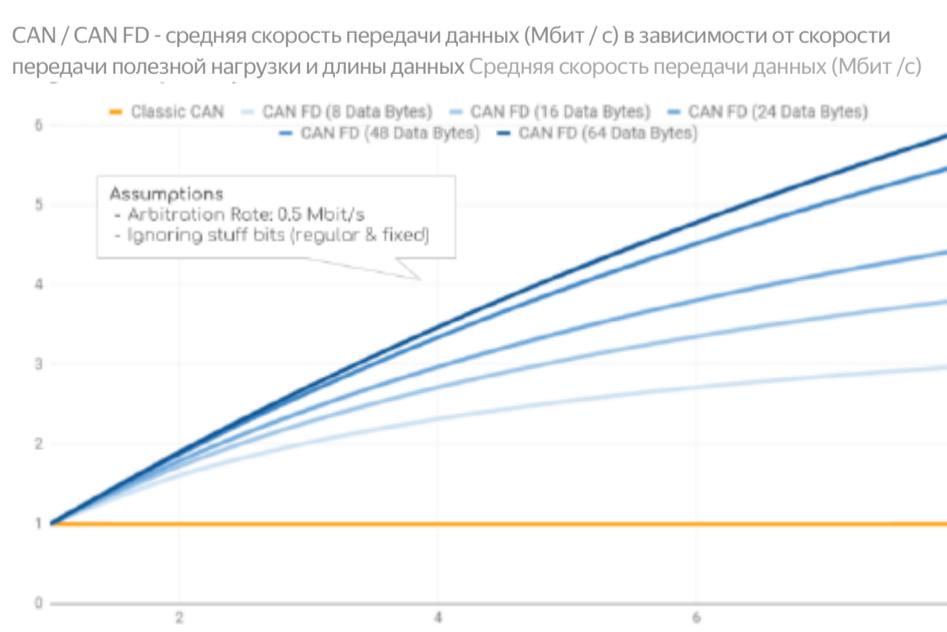
Сбивающим с толку аспектом CAN FD является максимальная скорость передачи данных на этапе загрузки, поскольку в разных статьях упоминаются разные уровни. Некоторые утверждают, что практические приложения обеспечивают скорость до 8 Мбит / с и теоретически 15 Мбит / с. Другие утверждают, что до 12 Мбит / с. Кроме того, Daimler заявляет, что скорость выше 5 Мбит / с сомнительна - как из-за отсутствия стандарта для этого, так и потому, что ожидается, что недорогой автомобильный Ethernet (10 BASE-T1) ограничит спрос на CAN FD с более высокой скоростью. Так что же правильно?

Это зависит. Если посмотреть на ISO 11898-2 (стандарт микросхем приемопередатчика), то в нем указаны два набора параметров симметрии. Это рекомендуется использовать приемопередатчики с улучшенными параметрами симметрии, которые часто рекламируются как приемопередатчики со скоростью 5 Мбит / с. В достижимая фазовая скорость передачи данных зависит от многих факторов. Одним из наиболее важных является желаемый температурный диапазон. Для перепрошивки блока управления не требуется поддержка низких температур. Это означает, что для перепрошивки блока управления можно увеличить скорость до 12 Мбит / с. Другое важное ограничение скорости передачи данных вызвано выбранной топологией. Шинная топология с очень короткими заглушками обеспечивает значительно более высокую скорость передачи данных по сравнению с гибридными топологиями с длинными заглушками или даже звездочками. Большинство многоадресных скорости шинных сетей ограничена 2 Мбит / с в диапазоне температур от -40 ° C до + 125 ° C. CiA предоставляет соответствующие практические правила в рекомендации CiA 601-3 по проектированию сети. Сюда входят рекомендации по установке точек выборки на этапе передачи данных.

Калькулятор CAN FD: эффективность и скорость передачи в бодах

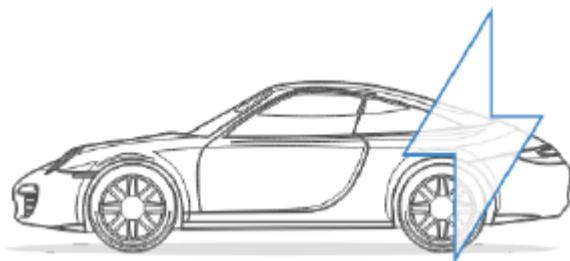
Для получения подробной информации об эффективности CAN FD и разрядности [смотрите наш калькулятор CAN FD](#).

Калькулятор сравнивает классический CAN и CAN FD в относится к фреймам и визуализирует структуры фреймов в зависимости от предоставленных вами настроек.



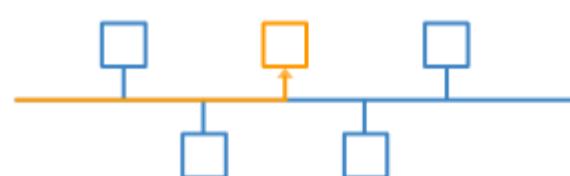
Примеры: Приложения CAN FD

Короче говоря, CAN FD позволяет получать больше данных с более высокой скоростью. Это имеет ключевое значение для множества все более актуальных вариантов использования.:



Электромобили

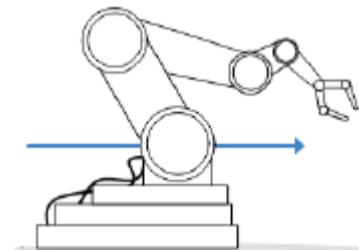
Электромобили и гибриды используют новые трансмиссии концепции, требующие гораздо более высоких разрядов скорости. Дополнительная сложность связана с новыми блоками управления, связанными с Инвертором постоянного тока, аккумулятором, зарядным устройством, расширитель диапазона и т.д. К 2025 году ожидается, что требуемая скорость передачи данных превысит CAN - и со взрывной рост числа электромобилей, это может стать началом внедрения CAN FD.



Перепрошивка ECU

Программное обеспечение автомобиля становится все более сложным. Таким образом, выполнение обновлений ECU, например, через Порт OBD2 сегодня может занять часы. С Можно, например, ускорить такие процессы > в 4 раза. Этот вариант использования был одним из оригинальных драйверов, стоящих за спрос на CAN FD со стороны автомобильной промышленности

OEM-производители.



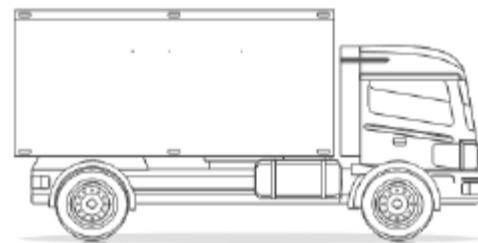
Робототехника

Некоторые приложения зависят от синхронизированного по времени поведения - например, робота манипуляторов с несколькими осями. Таких устройства часто используют CANopen и требуют отправки нескольких кадров CAN (PDO) каждым контроллером с синхронизацией по времени (без прерываний из кадров с более высоким приоритетом). Автор: при переходе на CAN FD данные могут быть для большей эффективности отправлены в одном кадре.



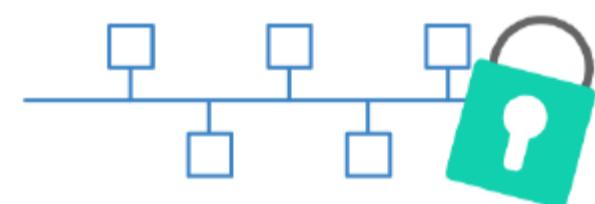
ADAS и безопасное вождение

Все больше продвинутых водителей системы помощи (ADAS) внедряются в легковых автомобилях и коммерческих транспортных средствах. Это оказывает давление на загрузку автобуса Classic CAN, однако ADAS - ключ к повышению безопасности. Здесь, CAN FD будет ключом к повышению безопасности вождение в ближайшем будущем.



Грузовики и автобусы

В грузовых автомобилях и автобусах используются длинные шины CAN (10-20 метров). В результате они полагаются на низкие скорости передачи данных (250 Кбит / с или 500 кбит / с согласно J1939-14). Здесь ожидается, что предстоящий протокол J1939 FD позволит значительно усовершенствовать коммерческие автомобили функции, включая, например, ADAS.

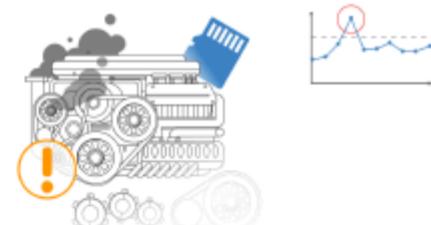


Безопасная шина CAN

[Как показано в недавних атаках CAN-хакера](#) Классическая CAN уязвима. Если хакеры получат доступ к шине CAN (например, по воздуху), они могли бы, например, отключить важные функции. МОЖНО ЛИ выполнить аутентификацию через защищенный Встроенная связь (SecOC) модуль может быть ключевым драйвером развертывания.

Протоколирование ПОЗВОЛЯЕТ получать данные - примеры использования

С появлением CAN FD появится несколько вариантов использования для протоколирования данных CAN FD:



Logging data from cars

As CAN FD gets rolled out in new cars, [CAN FD data loggers](#) will be key for OEM R&D and diagnostics

[learn more](#)

Heavy duty fleet telematics

[IoT CAN FD loggers](#) compatible with J1939 FD (flexible data-rate) will be key to future heavy duty telematics

[learn more](#)

Predictive maintenance

As CANopen FD rolls out, new industrial machinery will need CAN FD IoT loggers to help predict and avoid breakdowns

[learn more](#)

Vehicle/machine blackbox

A [CAN FD logger](#) can serve as a 'blackbox' for e.g. new prototype vehicles, providing data for diagnostics and R&D

[learn more](#)

Протоколирование CAN FD - практические соображения

При протоколировании данных CAN FD важны следующие соображения:

1 ISO CAN FD против не-ISO CAN FD

До обновления ISO 11898-1: 2015 стандарт CAN FD имел недостаток, связанный с проверкой ошибок. Контроллеры соблюдение обновленного стандарта иногда называют "ISO CAN FD". При записи данных с раннего прототипа системы CAN FD, поэтому вам может потребоваться включить режим "CAN FD без ISO", если ваше устройство поддерживает это.

2 Включение / выключение скорости передачи данных?

По умолчанию ваш регистратор данных CAN FD сможет обрабатывать как классические сообщения CAN, так и сообщения данных CAN FD - без предварительной настройки между ними. Аналогично, вам не нужно будет предварительно указывать, включено ли переключение скорости передачи данных исключительно для ведения журнала. Однако при передаче данных по шине CAN вам необходимо указать, использовать ли бит переключение скорости или нет. Если этот параметр включен, полезные данные передаются со второй скоростью передачи данных в системе, которая обычно составляет 2 или 4 Мбит / с.

Преобразование # 3 & Файлы DBC CAN FD

CAN FD сводит к минимуму необходимость обработки многопакетных сообщений. Это может значительно упростить разработку программного обеспечения для преобразования необработанных данных CAN FD в удобочитаемую форму в интересах конечных пользователей анализаторов CAN FD. Кроме того, стандартный формат базы данных CAN, DBC, также поддерживает правила преобразования CAN в FD. Поэтому всегда рекомендуется, чтобы вы собираете свои правила масштабирования в файл DBC, чтобы обеспечить легкий переход между различными программами CAN, такими как, например, asammfd и т.д.

CAN FD - outlook

Ожидается, что CAN FD заменит классический CAN в ближайшие годы:

Ожидается, что будут проданы первые автомобили с поддержкой CAN FD [2019/20 году](#)

На начальном этапе, вероятно, будет использоваться скорость 2 Мбит / с, постепенно переходя до 5 Мбит / с

CANopen FD был адаптирован с помощью CiA 1301 1.0

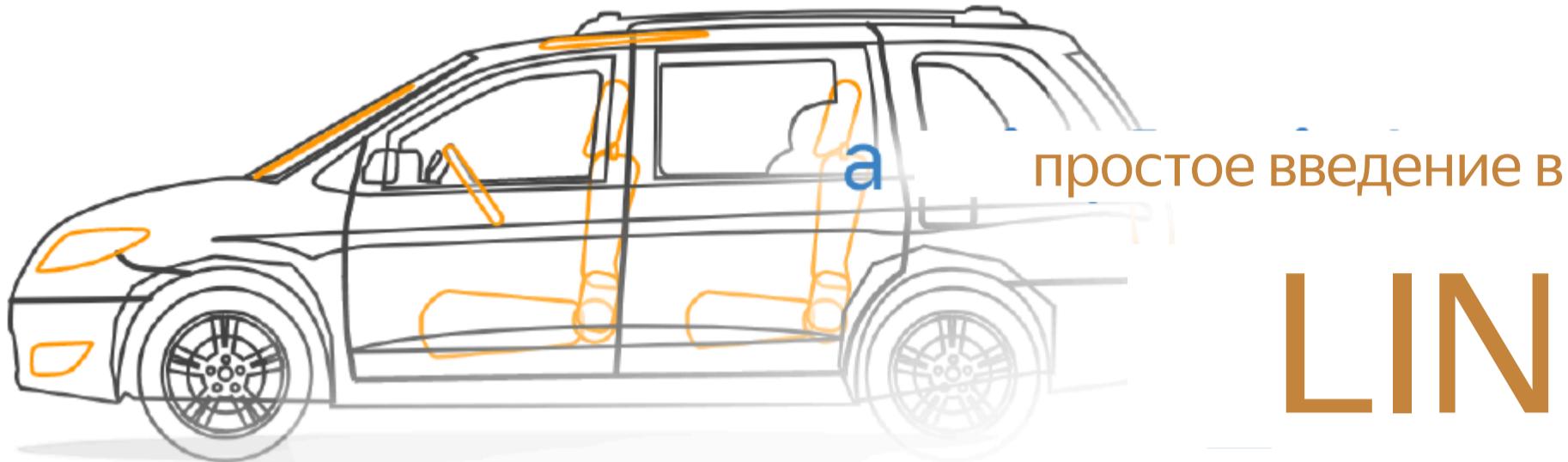
J1939-22 использует фреймы данных CAN FD

Технология CAN по-прежнему развивается, но в последнее время в основном благодаря CAN FD

Ожидается, что в будущем CAN FD будет использоваться в большинстве новых разработок

CAN FD по сравнению с другими протоколами

Конечно, устаревшие приложения без требований к пропускной способности и полезной нагрузке по-прежнему будут использовать классический CAN. Кроме того, CAN community уже разрабатывает следующее поколение канального уровня CAN, поддерживающего полезную нагрузку до 2048 байт. Этот подход можно рассматривать как альтернативу Ethernet со скоростью 10 Мбит / с. Таким образом, еще предстоит точно определить какую роль CAN FD будет играть в будущем, но она определенно будет на подъеме.



Объяснение шины LIN - простое введение

В этом руководстве мы знакомим с основами протокола локальной сети межсоединений (LIN), включая LIN против CAN, варианты использования, как работает LIN и шесть типов фреймов LIN.

Что такое LIN bus?

LIN bus является дополнением к CAN bus. Он обеспечивает более низкую производительность и надежность, но также значительно снижает затраты. Ниже мы приводим краткий обзор шины LIN и сравнение шины LIN с шиной CAN.

Недорогой вариант (если скорость / отказоустойчивость не критичны)

Часто используется в автомобилях для стеклоподъемников, дворников, кондиционеров и т.д....

Кластеры LIN состоят из 1 главного и до 16 подчиненных узлов

Однопроводный (+ заземление) с частотой 1-20 Кбит / с при максимальной длине шины 40 м.

Планирование по времени с гарантированным временем задержки

Переменный объем данных (2, 4, 8 байт)

LIN поддерживает обнаружение ошибок, контрольные суммы и настройку

Рабочее напряжение 12 В

Физический уровень на основе ISO 9141 (K-line)

Поддержка спящего режима и пробуждения

Большинство новых автомобилей имеют более 10 узлов LIN

Шина LIN по сравнению с шиной CAN

LIN дешевле (меньше проводов, нет лицензионной платы, дешевые узлы)

CAN использует скрученные экранированные двойные провода 5 В по сравнению с LIN single wire 12 В

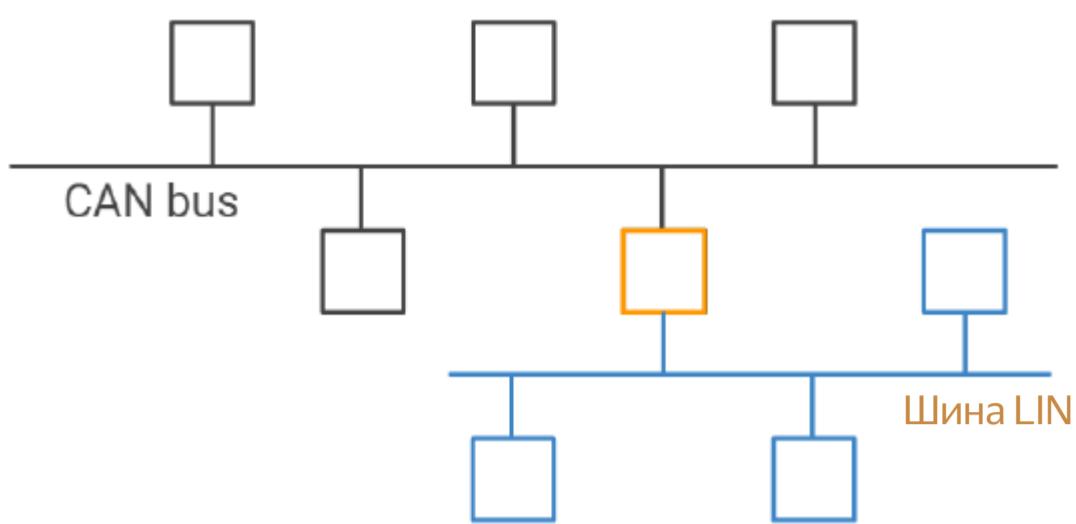
Мастер LIN обычно служит шлюзом к шине CAN

LIN является детерминированным, а не управляемым событиями (т. Е. Без арбитража шины)

Кластеры LIN имеют одного ведущего - CAN может иметь несколько

CAN использует 11 или 29-битные идентификаторы по сравнению с 6-битными идентификаторами в LIN

CAN обеспечивает скорость до 1 Мбит / с по сравнению с LIN при максимальной скорости 20 Кбит / с



История шины LIN

Ниже мы кратко расскажем об истории протокола LIN:

- 1999: LIN 1.0 выпущен Консорциумом LIN (BMW, VW, Audi, Volvo, Mercedes-Benz, Volcano и Motorola)
- 2000: Обновлен протокол LIN (LIN 1.1, LIN 1.2)
- 2002: выпущен LIN 1.3, в основном изменяющий физический уровень
- 2003: выпущен LIN 2.0, добавляющий основные изменения (широко используемые)
- 2006: выпущена спецификация LIN 2.1
- 2010: выпущен LIN 2.2A, в настоящее время широко внедряемые версии**
- 2010-12: SAE стандартизировал LIN как SAE J2602, основанный на LIN 2.0
- 2016: CAN в автоматизации стандартизировал LIN (ISO 17987: 2016)

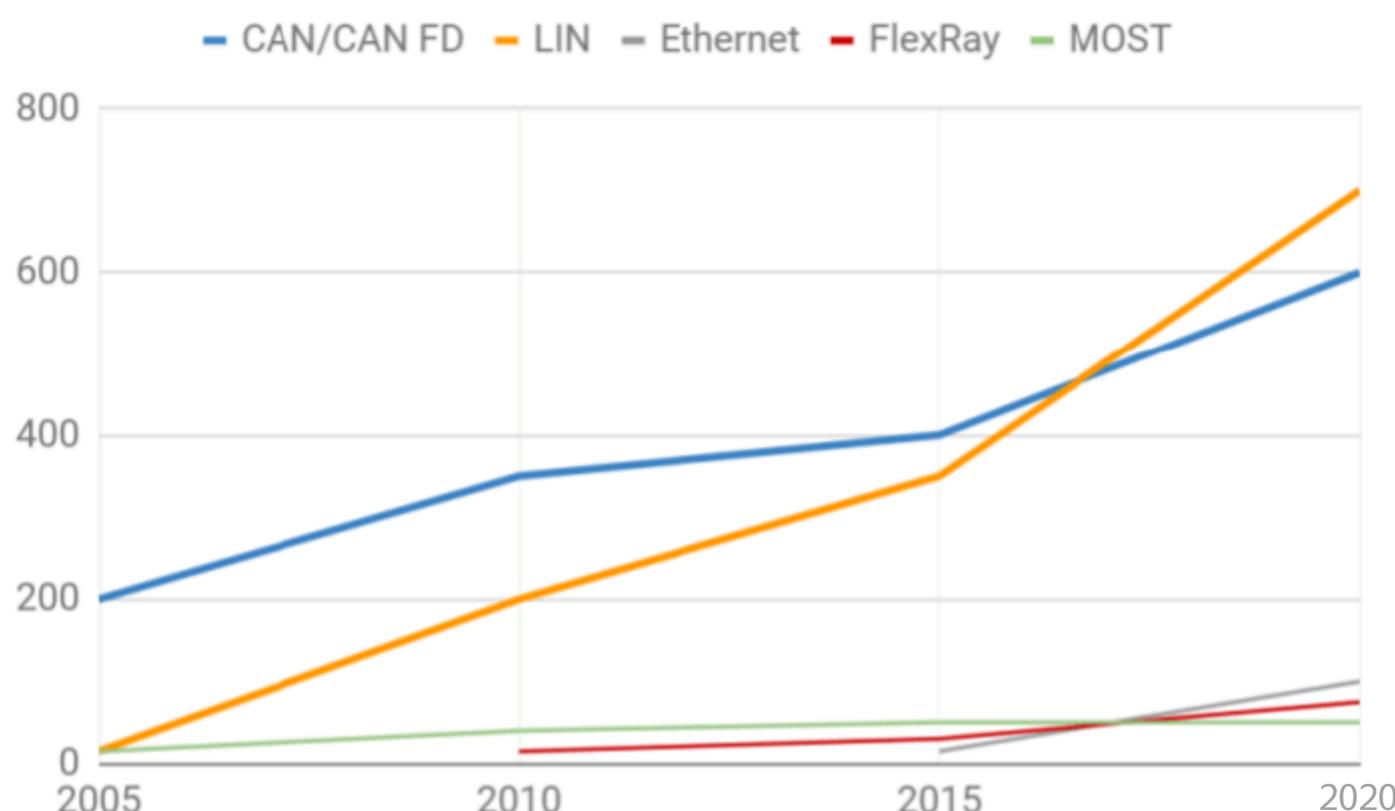


Будущее шины LIN

Протокол LIN играет все более важную роль в обеспечении недорогого расширения функциональных возможностей современных автомобилей. Поскольку таким образом, популярность LIN bus резко возросла за последнее десятилетие: ожидается, что к 2020 году в автомобилях будет > 700 миллионов узлов по сравнению с ~ 200 миллионами в 2010 году.

#Узлы в автомобилях с разбивкой по технологиям (2005-2020)

Источник: Strategic Analytics, 2013



Кибербезопасность и новые протоколы

Однако с появлением LIN также усиливается контроль за кибербезопасностью. LIN подвергается аналогичному риску а поскольку LIN играет определенную роль, например, в сиденьях и рулевом колесе, может потребоваться устранение этих рисков . В будущих автомобильных сетях наблюдается рост популярности CAN FD, FlexRay и automotive Ethernet. В то время как существует неопределенность относительно роли, которую каждая из этих систем будет играть в будущих автомобилях, ожидается, что LIN bus кластеры останутся жизненно важными в качестве недорогого решения для удовлетворения постоянно растущего спроса на функции в современных автомобилях.

Ведущий / ведомый в сравнении с командиром / ответчиком

В рамках разработки более всеобъемлющей формулировки для шины LIN, согласованная формулировка CIA / ISO / SAE перейдет на командир / ответчик. Таким образом, это будет стандартная формулировка де-факто, используемая в большинстве руководств и инструкций будущие спецификации.

Применение шин LIN

Сегодня шины LIN являются стандартом де-факто практически для всех современных автомобилей - примеры использования в автомобилестроении приведены ниже:

Рулевое колесо: круиз-контроль, стеклоочиститель, климат-контроль, радио

Комфорт: датчики температуры, люка, освещенности, влажности

Трансмиссия: датчики положения, скорости, давления

Двигатель: небольшие двигатели, двигатели охлаждающих вентиляторов

Кондиционер: двигатели, панель управления (кондиционер часто является сложным)

Двери: Боковые зеркала, стекла, система управления сиденьем, замки

Сиденья: двигатели положения, датчики давления

Прочее: стеклоочистители, датчики дождя, фары, воздушный поток

Кроме того, шина LIN также используется в других отраслях промышленности:

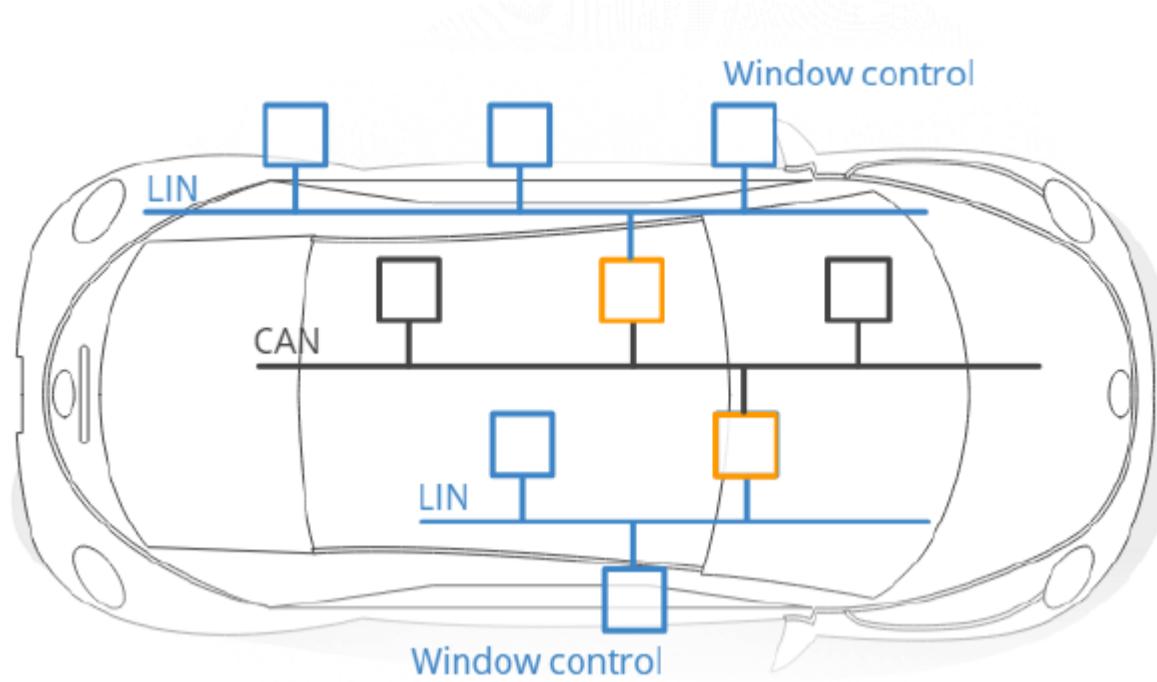
Бытовая техника: Стиральные машины, холодильники, плиты

Автоматизация: Производственное оборудование, металлообработка

Пример: LIN vs CAN window control

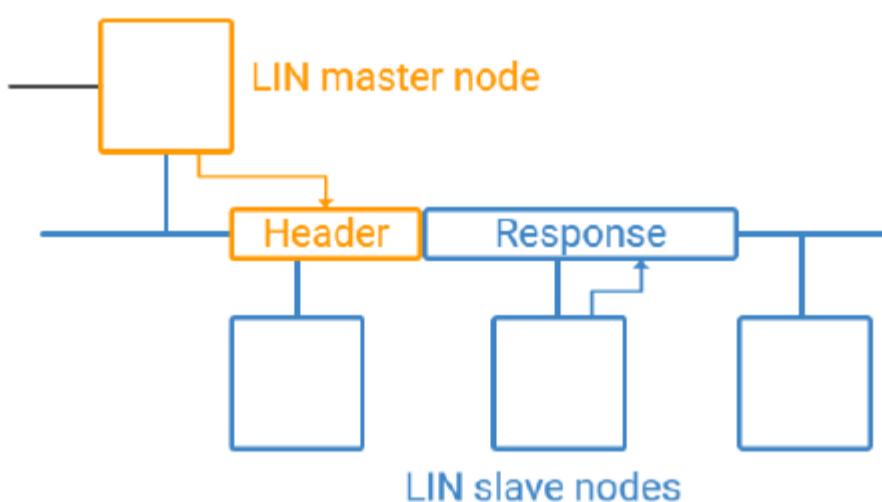
Узлы LIN обычно объединены в кластеры, каждый из которых имеет ведущий, который взаимодействует с магистральной шиной CAN.

Пример: На правом сиденье автомобиля вы можете опустить стекло левого сиденья. Для этого вы нажимаете кнопку для отправки сообщения через один кластер LIN в другой кластер LIN по шине CAN. Это запускает второй блок LIN, который опускает левое сиденье окно.



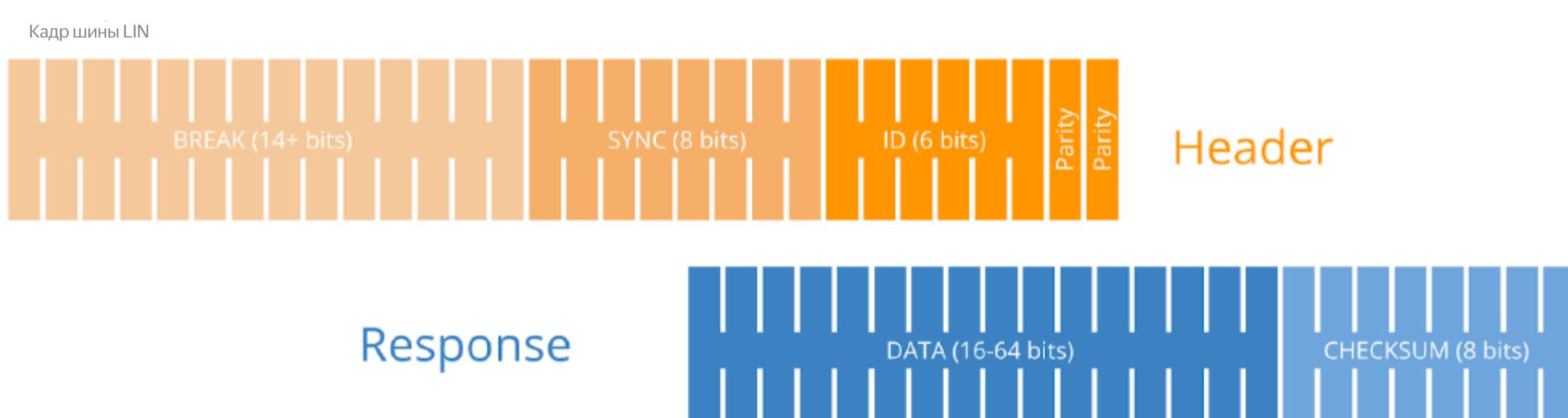
Как работает шина LIN?

Коммуникация LIN по своей сути относительно проста: главный узел проходит через каждый из подчиненных узлов, отправляя запрос информации - и каждый подчиненный узел отвечает данными при опросе. Байты данных содержат сигналы шины LIN (в необработанном виде). Однако с каждым обновлением спецификации в спецификацию LIN добавлялись новые функции, что делало ее более сложной. Ниже мы рассмотрим основы: фрейм LIN и шесть типов фреймов.



Формат фрейма LIN

Проще говоря, фрейм сообщения шины LIN состоит из заголовка и ответа. Обычно ведущее устройство LIN передает заголовок на шину LIN. Это запускает ведомое устройство, которое в ответ отправляет до 8 байт данных. Этот общий формат кадра LIN может быть проиллюстрировано, как показано ниже:



Поля кадра LIN

Разрыв: Поле разрыва синхронизации (SBF), также известное как Break, имеет длину не менее 13 + 1 бита (а на практике чаще всего 18 + 2 бита). Поле Break действует как уведомление о начале кадра для всех узлов LIN на шине.

Синхронизация: 8-разрядное поле синхронизации имеет предопределенное значение 0x55 (в двоичном формате 01010101). Эта структура позволяет узлам LIN определить время между восходящими / нисходящими фронтами и, следовательно, скорость передачи данных в бодах, используемую главным узлом. Это позволяет каждому из них оставаться синхронизированными.

Идентификатор: Идентификатор состоит из 6 бит, за которыми следуют 2 бита четности. Идентификатор действует как идентификатор для каждого отправляемого сообщения LIN и какие узлы реагируют на заголовок. Подчиненные устройства определяют достоверность поля ID (на основе битов четности) и действуют с помощью ниже:

1. Игнорируют последующую передачу данных
2. Прослушивают данные, передаваемые с другого узла.
3. Публикуют данные в ответ на заголовок

Как правило, информация запрашивается у одного подчиненного устройства одновременно, что означает нулевой риск столкновения (и, следовательно, отсутствие необходимости в арбитраже). Обратите внимание, что 6 бит допускают 64 идентификатора, из которых идентификаторы 60-61 используются для диагностики (подробнее ниже), а 62-63 зарезервированы.

Данные: Когда ведущее устройство опрашивает подчиненное устройство LIN, оно может ответить передачей 2, 4 или 8 байт данных. Длина данных может быть настроен, но обычно привязан к диапазону идентификаторов (идентификатор 0-31: 2 байта, 32-47: 4 байта, 48-63: 8 байт). Данные байты содержат фактическую информацию, передаваемую в форме сигналов LIN. Сигналы LIN упакованы в байты данных и могут иметь длину, например, всего 1 бит или несколько байт.

Контрольная сумма: Как и в CAN, поле контрольной суммы гарантирует достоверность кадра LIN. Классическая 8-битная контрольная сумма основана на суммировании только байтов данных (LIN 1.3), в то время как улучшенный алгоритм контрольной суммы также включает поле идентификатора (LIN 2.0).

Протоколирование данных шины LIN

CANedge позволяет легко записывать данные шины LIN на SD-карту объемом 8-32 ГБ. Просто подключите ее к своему приложению LIN, чтобы начните ведение журнала и обрабатывайте данные с помощью бесплатного программного обеспечения / API.

Например, бесплатный графический интерфейс / API asammdf позволяет вам выполнять DBC расшифруйте ваши данные LIN до физических значений и, например, нанесите на график ваши [Сигналы LIN](#). Узнать больше



Шесть типов рамок LIN

Существует множество типов рамок LIN, хотя на практике подавляющее большинство большая часть обмена данными осуществляется с помощью "безусловных фреймов".

Обратите также внимание, что все нижеприведенные фреймы соответствуют одной и той же базовой линии структура фрейма - и отличаются только временем или содержанием данных байт. Ниже мы кратко описываем каждый тип кадра LIN:

ИДЕНТИФИКАТОР (dec)	ИДЕНТИФИКАТОР (шестнадцатеричный)	Тип кадра LIN
0-59	00-3B	Безусловный
0-59	00-3B	Запущено событие
0-59	00-3B	Сporadicкий
60-61	3C-3D	Диагностика
62...	3E...	Определяется пользователем
63...	3F...	Зарезервировано



Unconditional Frames	Event Trigger Frames	Sporadic Frames	Diagnostic Frames	User Defined Frames	Reserved Frames
The default form of communication where the master sends a header, requesting information from a specific slave. The relevant slave reacts accordingly	The master polls multiple slaves. A slave responds if its data has been updated, with its protected ID in the 1st data byte. If multiple respond, a collision occurs and the master defaults to unconditional frames	Only sent by the master if it knows a specific slave has updated data. The master "acts as a slave" and provides the response to its own header - letting it provide slave nodes with "dynamic" info	Since LIN 2.0, IDs 60-61 are used for reading diagnostics from master or slaves. Frames always contain 8 data bytes. ID 60 is used for the master request, 61 for the slave response	ID 62 is a user-defined frame which may contain any type of information	Reserved frames have ID 63 and must not be used in LIN 2.0 conforming LIN networks

Расширенные разделы LIN

Ниже мы включаем два расширенных раздела - нажмите, чтобы развернуть.

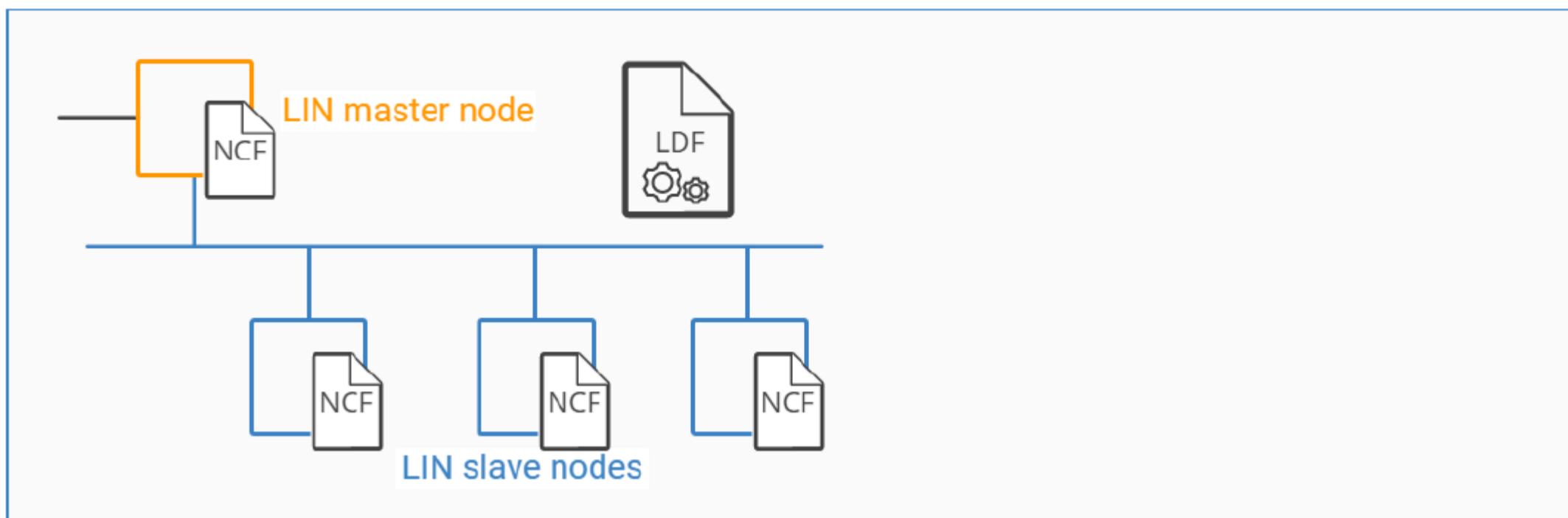
Файл конфигурации узла LIN (NCF) и файл описания LIN (LDF)

Для быстрой настройки сетей шин LIN готовые узлы LIN поставляются с файлами конфигурации узлов (NCF). Сведения о NCF возможности узла LIN и является ключевой частью топологии LIN.

Затем OEM-производитель объединит эти NCF узлов в файл кластера, называемый файлом описания LIN (LDF). Мастер затем настраивает и управляет кластером LIN на основе этого LDF - например, расписания для заголовков.

Обратите внимание, что узлы шины LIN могут быть повторно сконфигурированы с помощью диагностических фреймов, описанных ранее. Этот тип настройка может выполняться во время производства - или, например, при каждом запуске сети. Например, это может быть использовано для изменения идентификаторов сообщений узла.

Если вы знакомы с CANopen, вы можете увидеть параллели с файлом конфигурации устройства, используемым для предварительной настройки CANopen узлы - и роль объектов служебных данных в обновлении этих конфигураций.

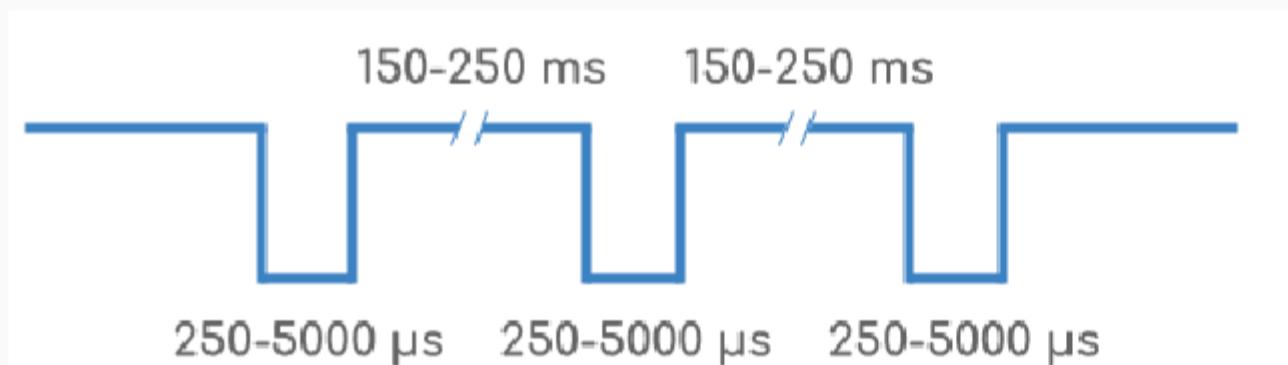


Режим сна и пробуждения LIN

Ключевым аспектом LIN является не только экономия затрат, но и электроэнергии.

Для достижения этого каждое подчиненное устройство LIN может быть переведено в спящий режим ведущим устройством, отправляющим диагностический запрос (ID 60) с первым байтом, равным 0. Каждое подчиненное устройство также автоматически переходит в режим сна после 4 секунд простого шины.

Подчиненные устройства могут быть разбужены либо ведущим, либо подчиненным узлами, отправляющими запрос на пробуждение. Это делается путем принудительного включения шины в режиме доминирования в течение 250-5000 микросекунд с последующей паузой в 150-250 мс. Это повторяется до 3 раз, если главный сервер не отправляет заголовок. После этого требуется пауза в 1,5 секунды перед отправкой четвертого запроса на пробуждение. Обычно узлы просыпаются после 1-2 импульсов.



Файл описания LIN (LDF) по сравнению с Файлами DBC

В рамках рабочего процесса вашего регистратора данных LIN может потребоваться декодирование необработанных данных шины LIN в физические значения. В частности, это включает в себя извлечение сигналов LIN из полезной нагрузки кадра LIN и декодирование их в удобочитаемую форму.

Этот процесс декодирования шины LIN аналогичен декодированию шины CAN и требует той же информации.:

ID: Какой идентификатор кадра LIN содержит сигнал шины LIN

Name: Имя сигнала LIN должно быть известно

Начальный бит: начальная позиция сигнала LIN в полезной нагрузке

Length: длина сигнала шины LIN

Порядковый номер: сигналы LIN имеют строчный порядковый номер (порядок байтов Intel)

Масштаб: как умножить десятичное значение битов сигнала LIN

Смещение: на какую константу должно быть смещено значение сигнала LIN

Единица измерения / Мин / Макс: Дополнительная вспомогательная информация (необязательно)

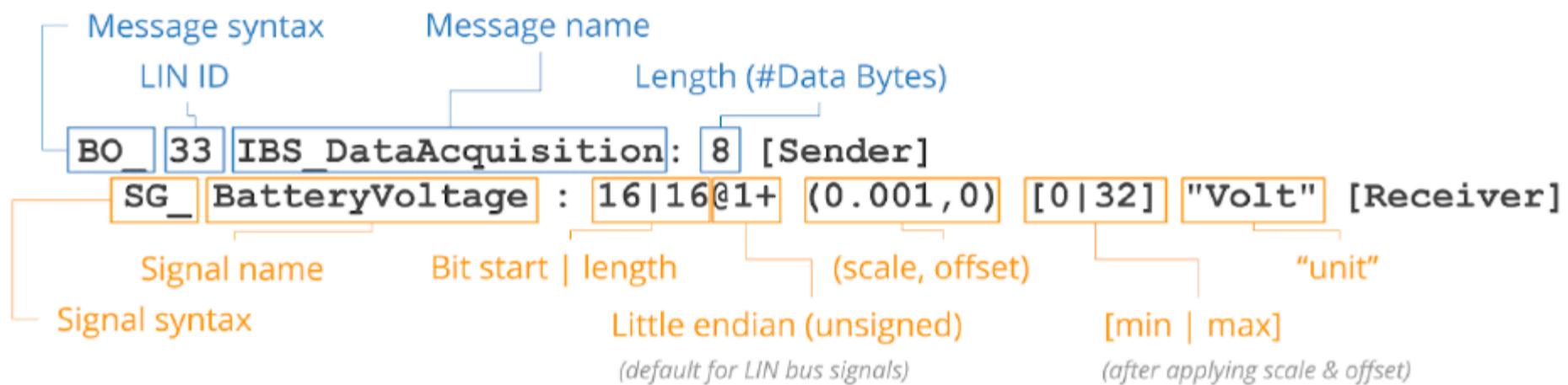
LIN Description File (LDF)

```
Signals {
    BatteryVoltage: 16 65535, IBS, SG ;
}
Signal name Bit length
```

```
Message name LIN ID Length
Frames {
    IBS_DataAcquisition: 33, IBS, 8 {
        BatteryCurrent, 0 ;
        BatteryVoltage, 16 ; Bit start
        CentreTapVoltage, 32 ;
        InChipTemperature, 48 ;
        CurrentRangeMeas, 57 ;
    }
}
```

```
Signal_encoding_types { min, max, scale, offset, "unit"
Enc_BatteryVoltage {
    physical_value, 0, 32000, 0.001, 0, "Volt" ;
    physical_value, 32001, 65533, 1, 0, "invalid" ;
    logical_value, 65534, "BatteryVoltage_Error" ;
    logical_value, 65535, "LIN_INIT" ;
}
```

CAN Database (DBC)



Эта информация обычно доступна как часть файла описания LIN (LDF) для локальной сети межсоединений. Однако, поскольку многие программные средства изначально не поддерживают формат LDF, ниже мы объясним, как использовать файлы DBC в качестве альтернативы.

Файл описания LIN (LDF) по сравнению с Файлами DBC

Как видно из нашего [CAN](#) ввода к шине и DBC файлу, приведенные выше записи эквивалентны информации, хранящейся в CAN DBC файле. Это означает, что простым методом хранения правил декодирования шины LIN является использование формата файла DBC, который поддерживается многими программными средствами и API (вкл. программные инструменты CANedge, такие как asammfdf). Например, вы можете загрузить файл LIN DBC и ваши необработанные данные шины LIN из CANedge в asammfdf для извлечения сигналов шины LIN из данных, которые затем вы можете построить, проанализировать или экспорттировать.

Во многих случаях у вас может не быть файла LIN DBC, доступного напрямую, но вместо этого у вас может быть файл описания LIN (LDF). Поэтому ниже мы сосредоточимся на том, как вы можете преобразовать соответствующую информацию о сигнале LIN в формат DBC.

Примечание: LDF обычно содержит различную другую информацию, относящуюся к работе шины LIN, которой у нас нет сосредоточьтесь на этом. Для полного ознакомления с протоколом LIN и подробного описания спецификации LDFсмотрите Стандарт LIN protocol PDF.

Как преобразовать файл описания LIN (LDF) в DBC [вкл. Примеры]

Ниже мы приводим пример, демонстрирующий, как вы можете извлечь информацию о сигнале LIN из LDF и ввести ее в Файл DBC. Мы используем очень упрощенный файл описания LIN (содержащий только один сигнал и исключающий некоторые разделы). Вы можете развернуть приведенные ниже примеры, чтобы увидеть сигнал LIN, напряжение батареи, в формате LDF и в формате DBC. Вы также можете загрузить необработанный файл журнала шины LIN (MF4) с CANedge2 с данными для этого сигнала, который вы можете открыть и декодировать DBC в asammfdf:

Пример: сигнал напряжения батареи (LDF)

[скачать.ldf | downloaddbc](#)

Руководство по преобразованию LDF в DBC

Короче говоря, чтобы преобразовать LDF-файл в DBC, вы выполните следующие шаги для каждого сигнала LIN:

Получите название и длину сигнала LIN в разделе Сигналы

Получите имя, идентификатор и длину сообщения сигнала LIN из раздела

Кадры

Получите бит сигнала LIN, начиная с раздела Кадры

Перейдите в раздел LDF Signal_encoding_types и найдите "Enc_[имя_сигнала]".

Получите оставшуюся информацию с помощью синтаксиса: 'physical_value, [min], [max], [scale], [offset], "[единица измерения]" ;'

Если вы хотите создать свой собственный файл LIN DBC, мы рекомендуем вам ознакомиться с нашим введением к файлу DBC для получения подробной информации о синтаксисе, а также инструментах редактора DBC.

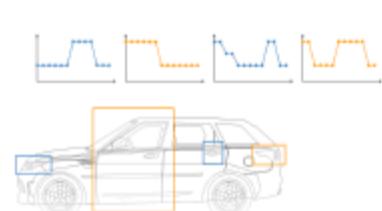
Незначительные ошибки.

Преобразование из LDF в DBC выполняется не полностью 1 к 1. В частности, обратите внимание, что напряжение батареи сигнала LIN составляет 2 записи для физического значения, одна для десятичного диапазона от 0 до 32000 и одна для от 32001 до 65533. В этом конкретном случае, действительны только данные из первого диапазона (единица измерения "недействительна" для второго диапазона). Однако в некоторых случаях могут быть несколько диапазонов, для которых требуются отдельные коэффициенты масштабирования, что невозможно обработать в формате файла DBC. В этом случае вам нужно будет выбрать один из диапазонов и, например, считать результаты за пределами этого диапазона недействительными.

Это также самый простой способ обработки записей LIN signal 'logical_value' в разделе Signal_encoding_types . Эти обычно отражают, как следует обрабатывать конкретные значения сигнала LIN (например, как ошибки). Один из способов обработки этих записей было бы игнорировать их и, возможно, исключить как часть постобработки ваших данных - аналогично тому, как значения в FF байтах в шине CAN часто исключаются, поскольку они представляют недопустимые или N / A данные.

Ведение журнала данных по шине LIN - примеры вариантов использования

Ведение журнала данных по шине LIN актуально в различных вариантах использования:



Vehicle CAN/LIN development

Logging CAN/LIN data via a hybrid logger is key to OEM vehicle development and can be used in optimization or diagnostics

[learn more](#)



Field prototype telematics

CAN/LIN data from automotive prototype equipment can be collected at scale using IoT CAN/LIN hybrid loggers to speed up R&D

[learn more](#)



Predictive maintenance

Industrial machinery can be monitored via IoT CAN/LIN loggers in the cloud to predict and avoid breakdowns via prediction models

[learn more](#)



Rare LIN issue diagnostics

A [LIN bus logger](#) can serve as a 'blackbox' for industrial machinery, providing data for e.g. disputes or rare issue diagnostics

[learn more](#)

Практические рекомендации по ведению журнала данных по шине LIN

Ниже мы перечислили ключевые рекомендации по ведению журнала данных по шине LIN:

Сравнение LIN logger с Интерфейсом LIN

Для записи данных шины LIN вам понадобится регистратор данных шины LIN и / или интерфейс. Регистратор данных шины LIN с SD-картой имеет преимущество возможности записывать данные в автономном режиме, то есть во время фактического использования автомобиля. Интерфейс, с другой стороны, полезен, например, при динамическом тестировании функциональности автомобиля.

Для автономных регистраторов LIN важно, чтобы устройство было подключаемым и воспроизводимым, компактным и недорогим, что позволяет использовать его в масштабируемых приложениях, например, в автопарках.

Поддержка CAN и LIN

Часто требуется объединить данные LIN с данными CAN, чтобы получить целостное представление об используемом транспортном средстве - например,:

Как поведение водителя соотносится с использованием различных функций шины LIN?

Возникают ли проблемы при взаимодействии между LIN-мастерами и CAN шиной?

Связаны ли проблемы, связанные с LIN, с определенными событиями на основе CAN?

Для объединения этих данных вам понадобится гибридный регистратор CAN / LIN с несколькими каналами. Кроме того, поддержка CAN FD также является ключевой поскольку ожидается, что она будет все чаще применяться в новых автомобилях.

Wi-Fi

Сбор данных журнала шины может быть проблемой, если вам нужно физически извлечь данные из, например, большие испытания корабля флотов. Здесь, регистратор CAN / LIN с поддержкой Wi-Fi может быть мощным решением. Вы просто указываете точку доступа Wi-Fi, которую автомобиль будет попадать в диапазон от времени до времени, а затем данные будут автоматически загружены с SD-карты, когда в ассортименте. Также можно добавить точку доступа сотовой связи в автомобиле для передачи данных практически в режиме реального времени.



МОЖНО ли объяснить файл DBC - простое введение

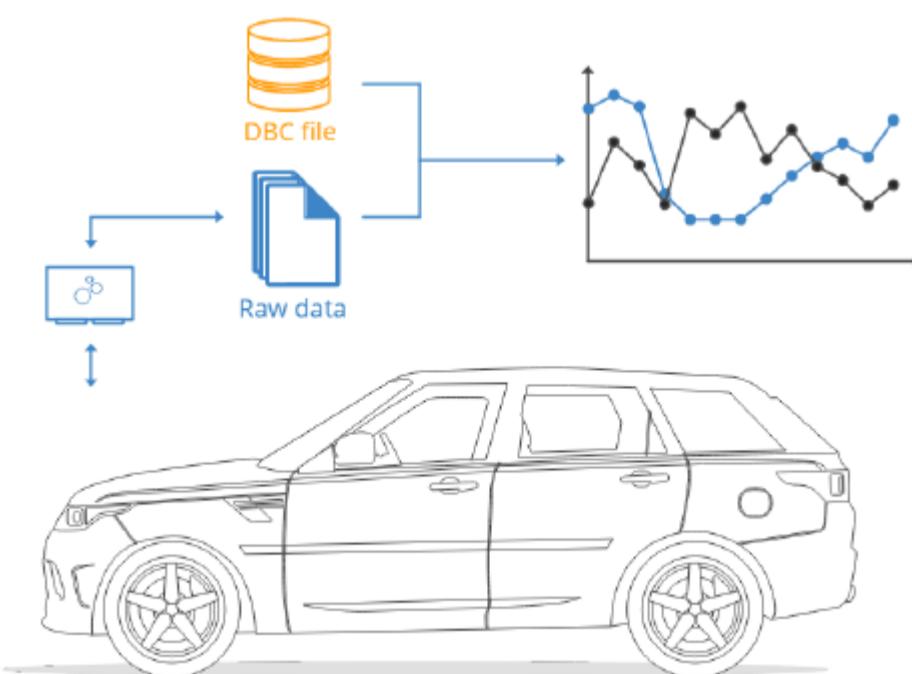
В этом руководстве мы объясняем файлы DBC (базы данных шины CAN), включая структуру, синтаксис, примеры - и классный редактор DBC игровая площадка. Для удобства мы также включаем реальные данные J1939 / OBD2 и файлы DBC, которые вы можете загрузить в бесплатном открытом исходном коде CAN tools.

Что такое файл CAN DBC?

Файл CAN DBC (база данных CAN) - это текстовый файл, содержащий информацию для декодирования необработанных данных шины CAN в "физические значения".

Чтобы понять, как выглядят "необработанные данные CAN", посмотрите приведенный ниже пример кадра CAN из грузовика:

```
>CAN ID Data bytes 0CF00400  
FF FF FF 68 13 FF FF FF
```



<Если у вас есть CAN DBC, который содержит правила декодирования для CAN ID, вы можете "извлекать" параметры (сигналы) из данных байты. Одним из таких сигналов может быть EngineSpeed:

Сообщение	Сигнал	Значение	Единица измерения
EEC1	Скорость двигателя	621	rpm

Чтобы понять, как работает декодирование DBC, мы объясним синтаксис DBC и предоставим пошаговые примеры декодирования.

Синтаксис сообщений и сигналов DBC

Давайте начнем с реального примера файла CAN DBC. Ниже приведен демонстрационный файл J1939 DBC, который содержит правила декодирования для повышения скорости (км / ч) и частота вращения двигателя (об / мин). Вы можете скопировать это в текстовый файл, переименовать его, например, в j1939dbc и использовать для извлечения частота вращения / об/мин грузовых автомобилей, тракторов или других транспортных средств большой грузоподъемности.

>J1939 DBC demo example

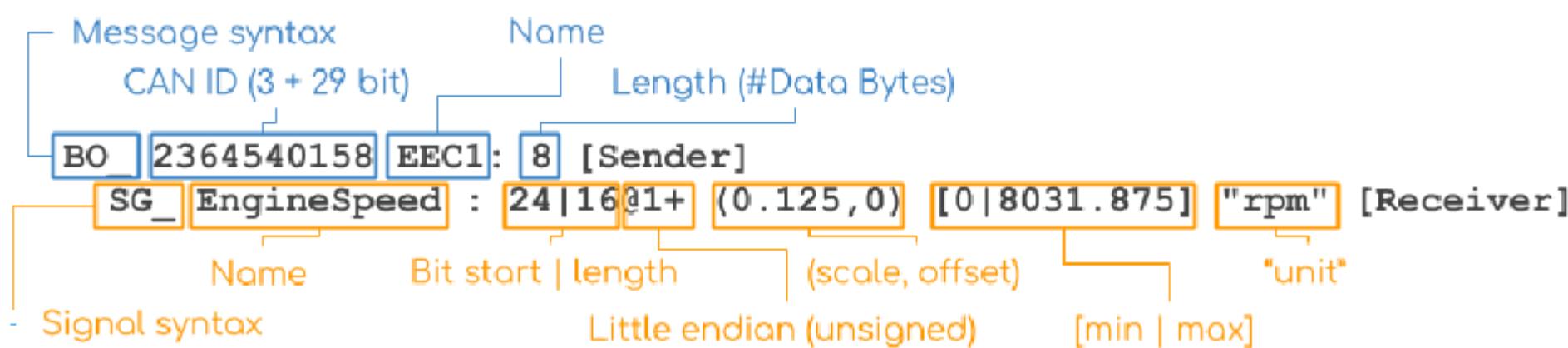
```
VERSION ""

NS_ :
  CM_
  BA_DEF_
  BA_
  BA_DEF_DEF_

BS_ :
BU_ :

B0_ 2364540158 EEC1: 8 Vector_XXX
SG_ EngineSpeed : 24|16@1+ (0.125,0) [0|8031.875] "rpm" Vector_XXX
B0_ 2566844926 CCVS1: 8 Vector_XXX
SG_ WheelBasedVehicleSpeed : 8|16@1+ (0.00390625,0) [0|250.996] "km/h" Vector_XXX
CM_ B0_ 2364540158 "Electronic Engine Controller 1";
CM_ SG_ 2364540158 EngineSpeed "Actual engine speed which is calculated over a minimum crankshaft angle
of 720 degrees divided by the number of cylinders...
CM_ B0_ 2566844926 "Cruise Control/Vehicle Speed 1"
CM_ SG_ 2566844926 WheelBasedVehicleSpeed "Wheel-Based Vehicle Speed: Speed of the vehicle as calculated
from wheel or tailshaft speed.";
BA_DEF_ SG_ "SPN" INT 0 524287;
BA_DEF_ B0_ "VFrameFormat" ENUM "StandardCAN", "ExtendedCAN", "reserved", "J1939PG";
BA_DEF_ "BusType" STRING ;
BA_DEF_ "ProtocolType" STRING ;
BA_DEF_DEF_ "SPN" 0;
BA_DEF_DEF_ "VFrameFormat" "J1939PG";
BA_DEF_DEF_ "BusType" "
BA_DEF_DEF_ "ProtocolType" "
BA_ "ProtocolType" "J1939";
BA_ "BusType" "CAN";
BA_ "VFrameFormat" B0_ 2364540158 3;
BA_ "VFrameFormat" B0_ 2566844926 3;
BA_ "SPN" SG_ 2364540158 EngineSpeed 190;
BA_ "SPN" SG_ 2566844926 WheelBasedVehicleSpeed 84;
```

<В основе файла DBC лежат правила, описывающие, как декодировать сообщения CAN и сигналы:



Объяснен синтаксис сообщения DBC

Сообщение начинается с **BO**, идентификатор должен быть уникальным и в десятичной системе счисления (не шестнадцатеричной)

Идентификатор DBC добавляет 3 дополнительных бита к 29-битным идентификаторам CAN, чтобы служить флагом "extended" ID"

Имя должно быть уникальным, состоять из 1-32 символов и может содержать [A-Z], цифры и подчеркивания

Длина (DLC) должна быть целым числом от 0 до 1785

Отправителем является имя передающего узла или Vector_XXX если имя недоступно

Объясняется синтаксис сигнала DBC

Каждое сообщение содержит 1 + сигналов, которые начинаются с **SG**.

Имя должно быть уникальным, состоять из 1-32 символов и может содержать [A-Z], цифры и подчеркивания

Начало бита отсчитывается от 0 и отмечает начало сигнала в полезной нагрузке данных

Длина бита - это длина сигнала

The @1 указывает, что порядок байтов является младшим по порядку / Intel (против @ 0 для старшего по порядку / Motorola)

The + сообщает, что тип значения беззнаковый (против - для сигналов со знаком)

Значения (масштаб, смещение) используются в линейном уравнении физического значения (подробнее ниже)

[min | max] и единица измерения являются необязательной метаинформацией (например, они могут быть установлены в [0 | 0] и "")

Приемник - это имя принимающего узла (опять же, по умолчанию используется Vector_XXX)

Пример: Извлечение физического значения сигнала EngineSpeed

Чтобы понять, как работает декодирование DBC, мы покажем, как извлечь скорость передачи сигнала из кадра CAN в вступление:

>CAN ID DataBytes 00400
FFCF00FF068 1B FF FF F68 13 FF FF FF



<1 # Сопоставьте идентификатор CAN с идентификатором DBC

На практике большинство файлов журнала необработанных данных CAN содержат 20-80 уникальных идентификаторов CAN. Таким образом, первым шагом является сопоставление каждого идентификатора CAN с соответствующими правилами преобразования в DBC. Для обычных 11-разрядных идентификаторов CAN это можно просто сделать, сопоставив десятичное значение идентификатора CAN с идентификаторами CAN DBC. Для расширенных 29-битных идентификаторов CAN необходимо применить маску (0x1FFFFFFF) к 32-разрядный идентификатор DBC для получения 29-разрядного идентификатора CAN, который затем может быть сопоставлен с файлом журнала.

Чтобы легко конвертировать между идентификаторами, смотрите таблицу "DBC ID vs CAN ID" на нашей игровой площадке DBC editor.

Что касается преобразования PGN в J1939

В этом примере мы напрямую сопоставляем 29-битный идентификатор CAN с "замаскированным" идентификатором DBC. На практике преобразование J1939 часто выполняется с помощью извлечения 18-разрядного PGN J1939 из идентификатора CAN и идентификатора DBC и последующее сравнение PGN.

Способ зависит от вашего программного обеспечения. В инструментах CANedge, таких как asammfd и наших инструментах Python API, загрузка J1939 DBC автоматически заставит инструменты использовать соответствие PGN J1939.

Сообщение J1939 (PGN & SPNs)



2# Извлеките биты сигнала

Затем используйте начало, длину и порядковый номер бита DBC, чтобы извлечь соответствующие биты из полезной нагрузки данных кадра CAN. В этом примере, начальный бит равен 24 (что означает 3 байта при отсчете от 0), а длина бита равна 16 (2 байта):

>FF FF FF 68 13 FF FF FF

<Большой порядковый номер или младший порядковый номер (расширенный)

В примере мы используем младший порядковый номер, что означает, что "начало бита" в DBC файл отражает позицию младшего значащего бита (LSB). Если вы добавляете сигнал DBC в средство просмотра файлов DBC (например, Vector CANDB ++), LSB также используется в качестве начального бита в редакторе DBC.

Если вы вместо этого добавите сигнал большого порядка в графический интерфейс редактора DBC, вы все равно увидите LSB в качестве начального бита - но при сохранении DBC начальный бит устанавливается равным старшему значащему биту (MSB) в сигнале. Такой подход упрощает редактирование графического интерфейса более интуитивно понятный, но может сбивать с толку при переключении между графическим интерфейсом и текстом редактора. Чтобы полностью разобраться в этом, ознакомьтесь с нашей игровой площадкой CAN DBC editor.

Bit indices							
7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8
23	22	21	20	19	18	17	16
31	30	29	28	27	26	25	24
39	38	37	36	35	34	33	32
47	46	45	44	43	42	41	40
55	54	53	52	51	50	49	48
63	62	61	60	59	58	57	56

3 # Масштабируйте извлеченные фрагменты.

Сигнал EngineSpeed имеет младший порядковый номер (@1), и поэтому нам необходимо изменить порядок следования байтов с 6813 на 1368. Далее мы преобразуем шестнадцатеричную строку в десятичную и применяем линейное преобразование:

>physical_value = offset + scale * raw_value_decimal

621 rpm = 0+0.125 * 4968

<Короче говоря, физическое значение скорости двигателя (оно же масштабированное инженерное значение) равно 621 обороту в минуту.

4 # DBC расшифруйте ваш полный набор данных.

Шаги 1-3 начинаются с одного кадра CAN. На практике вы будете декодировать необработанные данные, например, с транспортных средств, механизмов или лодок с миллионами кадров CAN, временными метками, несколькими уникальными идентификаторами CAN и иногда сотнями сигналов.

Чтобы справиться с этой сложностью, используется специализированное программное обеспечение для декодирования необработанных данных CAN в удобочитаемую форму путем загрузки файлы журнала данных и связанные файлы DBC. Чтобы проиллюстрировать, как может выглядеть результат, мы добавили фрагмент DBC расшифрованные данные J1939, зарегистрированные с помощью CANedge. Данные были преобразованы с помощью asammdf, отфильтрованы для включения набора соответствующих сигналов и экспортаны в формате CSV:

```
>timestamps, ActualEnginePercentTorque, EngineSpeed, EngineCoolantTemperature, EngineOilTemperature1, EngineFuelRate, EngineTotalIdleHours, FuelLevel1, Latitude, Longitude, WheelBasedVehicleSpeed
2020-01-13 16:00:13.259449959+01:00, 0, 1520.13, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.23
2020-01-13 16:00:13.268850088+01:00, 0, 1522.88, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.23
2020-01-13 16:00:13.270649910+01:00, 0, 1523.34, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.23
2020-01-13 16:00:13.271549940+01:00, 0, 1523.58, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.23
2020-01-13 16:00:13.278949976+01:00, 0, 1525.5, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.23
2020-01-13 16:00:13.289050102+01:00, 0, 1527.88, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.23
2020-01-13 16:00:13.299000025+01:00, 0, 1528.13, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.23
2020-01-13 16:00:13.308300018+01:00, 0, 1526.86, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.23
2020-01-13 16:00:13.309099913+01:00, 0, 1526.75, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.23
2020-01-13 16:00:13.317199945+01:00, 0, 1526.45, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.23
2020-01-13 16:00:13.319149971+01:00, 0, 1526.38, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.23
2020-01-13 16:00:13.320349932+01:00, 0, 1526.15, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.23
2020-01-13 16:00:13.326800108+01:00, 0, 1524.93, 92, 106, 3.8, 1868.3, 52, 40.6440124, -76.1223603, 86.25
```

<МОЖНО ли ИСПОЛЬЗОВАТЬ DBC Editor playground

Добавление или редактирование сигналов в ваш DBC-файл может сбить с толку. Чтобы помочь вам лучше понять, как это работает, мы создали очень простую онлайн-“игровую площадку для редактирования CAN DBC”. Вы можете открыть его и отредактировать желтые ячейки - или сделать копию для личного использования. Не стесняйтесь добавлять редактор в закладки и делиться им!

J1939 / Данные OBD2 и образцы DBC

Лучший способ узнать больше о преобразовании DBC необработанных данных шины CAN - попробовать это. Ниже вы можете загрузить raw Данные J1939 / OBD2 из CANedge2. Zip также включает в себя OBD2 DBC и демонстрационный J1939 DBC.

По ссылке вы также можете бесплатно загрузить графический интерфейс asammdf, чтобы открыть файлы data и DBC. Попробуйте построить график EngineSpeed в качестве [примера](#). Загрузите [примеры данных и dbc-файлов](#).

Дополнительно: метаинформация, атрибуты и мультиплексирование.

В этом разделе мы кратко опишем некоторые из более сложных разделов, касающихся файлов CAN DBC, включая метаинформацию, атрибуты, значения таблицы и мультиплексирование. Если вы новичок в файлах DBC, вы можете при желании пропустить этот раздел.

Комментарии (названия сообщений / сигналов, ...)

DBC может содержать различную дополнительную информацию, и ниже мы перечислим наиболее распространенные типы. Эта информация сохраняется в файле DBC после всех сообщений. Атрибут comment позволяет сопоставить комментарий из 1-255 символов с идентификатором сообщения или названием сигнала, например, для предоставления дополнительной информации. Пример для сообщения EEC1:

```
>CM_ B0_ 2364540158 "Electronic Engine Controller 1"
```

<Пример для сигнала EngineSpeed:

```
>CM_ SG_ 2364540158 EngineSpeed "Actual engine speed which is calculated over  
a minimum crankshaft angle of 720 degrees divided by the number of cylinders..."
```

<Атрибуты: Формат кадра, идентификаторы SPN,...

Пользовательские атрибуты могут быть добавлены к сообщениям и сигналам аналогично синтаксису комментария. Типичным атрибутом является VFrameFormat, который может использоваться для описания типа фрейма сообщения. Он инициализируется следующим образом: BA_DEF_BO_ "VFrameFormat" ПЕРЕЧИСЛЯЕТ "StandardCAN", "ExtendedCAN", "reserved", "J1939PG";

После инициализации сообщение может быть отображено следующим образом (индексация с 0):

```
>BA_ "VFrameFormat" BO_ 2364540158 3;
```

<В этом случае мы сообщаем, что сообщение EEC1 имеет тип J1939 PGN, что может привести к специальному отображению или обработке в различных инструментах графического интерфейса DBC editor, а также в инструментах обработки данных.

Аналогично, вы можете добавить идентификаторы SPN J1939 в качестве атрибута, как показано ниже.:

```
>BA_DEF_ SG_ "SPN" INT 0 524287; BA_ "SPN" SG_ 2364540158  
EngineSpeed 190;
```

<Здесь EngineSpeed присваивается SPN 190. Возможно, вам покажется более естественным сделать это противоположным способом - т. Е. Использовать Идентификаторы SPN в разделе сообщение / сигнал, затем сопоставить имена SPN с помощью атрибутов. Хотя вы определенно можете это сделать, это не наиболее распространенное соглашение. Кроме того, первый символ в имени сигнала CAN DBC не может быть числом, поэтому вам нужно необходимо написать, например, _190 или SPN_190.

Таблицы значений

Некоторые сигналы CAN DBC являются "переменными состояния", такими как переключение передач, диагностические коды неисправностей, коды состояния и т.д. Они принимают дискретные значения, и для их интерпретации потребуется таблица сопоставления.

Формат CAN DBC позволяет это с помощью таблиц значений, которые позволяют назначить описание состояния физическому десятичному значению каждого состояния сигнала. Состояния должны располагаться в порядке убывания.

Пример:

```
VAL_ 2297441534 MaterialDropActiveStatus 3 "Недоступно" 2 "Ошибка" 1 "Включено" 0 "Выключено" ;
```

Мультиплексированные сигналы (пример OBD2 DBC)

Мультиплексирование иногда используется при передаче данных по шине CAN, и, возможно, наиболее известный пример находится в OBD2 Информационные материалы. Рассмотрим, например, приведенные ниже кадры ответа OBD2:

```
>7E8 03 41 11 30 FF FF FF FF  
7E8 03 41 0D 37 FF FF FF FF
```

<Здесь оба фрейма ответа содержат 1 байт данных в байте 3, но, несмотря на идентичный идентификатор CAN, интерпретация данные различаются между двумя фреймами. Это связано с тем, что байт 2 служит мультиплексором, указывающим, с какого OBD2 PID поступают данные данные поступают из. Чтобы обработать это в контексте файла DBC, может быть применен приведенный ниже синтаксис:

```

>B0_ 2024 OBD2: 8 Vector_XXX
<>SG_ S1_PID_0D_VehicleSpeed m13 : 31|8@0+ (1,0) [0|255] "km/h" Vector_XXX SG_ S1_PID_11_ThrottlePosition
m17 : 31|8@0+ (0.39216,0) [0|100] "%" Vector_XXX
SG_ S1 m1M : 23|8@0+ (1,0) [0|255] "" Vector_XXX "
SG_ Service M : 11|4@0+ (1,0) [0|15] "" Vector_XXX

<>SG_MUL_VAL_ 2024 S1_PID_0D_VehicleSpeed S1
13-13; SG_MUL_VAL_ 2024 S1_PID_11_ThrottlePosition
S1 17-17; SG_MUL_VAL_ 2024 S1 Service 1-1;

```

<Здесь M в служебном сигнале служит "сигналом мультиплексора". В этом случае он переключает, какой сервисный режим OBD2 используется (режим 01, 02, ...). Сигнал S1 мультиплексируется службой, что видно из поля SG_MUL_VAL_, в котором сгруппированы два значения. Как видно, сигнал S1 имеет значение m1 после названия сигнала, что означает, что если Сервис сигнал принимает значение 1, данные отражают сервисный режим OBD2 01.

Вышеупомянутое называется простым мультиплексированием. Но могут ли файлы DBC также поддерживать расширенное мультиплексирование, где мультиплексированный сигнал (в данном случае S1) также может быть мультиплексором и, таким образом, управлять интерпретацией других частей полезной нагрузки данных. Чтобы убедиться в этом, обратите внимание, что S1 играет ту же роль, что и Service в синтаксисе, добавляя M после m1 и будучи сгруппированным с двумя PID OBD2, скоростью и положением дроссельной заслонки.

Расширенное мультиплексирование работает как и раньше: если S1 принимает значение 13 (шестнадцатеричное значение 0D), это означает, что только сигналы, которые являются A) частью следует учитывать, что группы S1 и B) имеют значение переключателя мультиплексора 13. В этом примере это означает, что байт 4 отражает данные о скорости транспортного средства. Если байт 3 равен 17 (ШЕСТНАДЦАТИЧНЫЙ 11), то байт 4 отражает данные о положении дроссельной заслонки.

Мультиплексирование DBC и расширенное мультиплексирование - это сложная тема, которая поддерживается не всеми инструментами обработки данных. Однако вы можете использовать, например, редактор DBC CANDB ++ для упрощения просмотра и понимания DBC-файлов с мультиплексированием, например, OBD2 DBC: Файл obd2 dbc

Файл OBD2 DBC можно использовать вместе с нашими CANedge CAN loggers для декодирования кадров OBD2 в например, asammfd или наши модули Python API. Подробнее об этой теме читайте в руководстве Vector по расширенному мультиплексированию в DBC файлы здесь.

Программные инструменты DBC (редактирование и обработка)

Программное обеспечение, использующее файлы CAN DBC, можно разделить на две группы: Редактирование и обработка данных.

Инструменты редактора DBC.

Vector CANDB ++: Бесплатная демо-версия Vector's CANalyzer включает полезную версию CANDB ++, векторного DBC-редактора. Он предлагает самую обширную доступную функциональность, включая быструю "проверку согласованности" файлов DBC

- Редактор базы данных Kvaser: Kvaser предоставляет бесплатный и простой в использовании редактор CAN DBC, который включает в себя возможность добавлять файлы DBC и визуализировать конструкцию сигнала
- canmatrix: Этот модуль Python DBC с открытым исходным кодом позволяет загружать файлы DBC, редактировать их и экспортить в другие форматы. Он используется, например, в asammfd и нашем Python API

Инструменты обработки данных CAN

Большинство инструментов обработки данных CAN поддерживают файлы DBC, включая приведенные ниже:

графический интерфейс asammfd: Графический интерфейс asammfd позволяет загружать необработанные данные MDF4 и преобразовывать их DBC в удобочитаемую форму - а также создавать быстрые графики, анализировать и экспортить

Python API: Наш Python API позволяет автоматизировать преобразование ваших данных в DBC, например, для крупномасштабного анализа данных или как часть интеграции с телематическими информационными панелями

Другие инструменты: Наши конвертеры MDF4 позволяют быстро конвертировать ваши данные CANedge MF4, например, в Vector ASC, PEAK TRC и другие

Как создать файл DBC?

Начало работы: Если вам нужно создать новый файл DBC, мы рекомендуем использовать один из редакторов DBC, описанных выше. Для новичков мы рекомендуем редактор DBC Kvaser. При создании нового файла DBC обычно можно выбрать DBC шаблон (например, CAN FD DBC, J1939 DBC, NMEA 2000 DBC и т.д.). Затем начните с добавления одного сообщения и одного сигнала в DBC и сохраните его.

Чтобы убедиться, что ваш DBC выглядит нормально, мы рекомендуем открыть DBC в текстовом редакторе. Таким образом, вы можете убедиться, что базовый Синтаксис DBC выглядит так, как вы ожидаете, и вы можете использовать эту версию в качестве эталона для сравнения. Рекомендуется продолжайте пересматривать git при любых изменениях в DBC отсюда.

Протестируйте свой DBC: В качестве второго шага мы рекомендуем протестировать файл DBC с помощью программного средства CAN bus decoder. Для примера, если вы используете регистратор данных CANedge CAN bus для записи необработанных данных CAN из вашего приложения, вы можете использовать бесплатное программное обеспечение asammfdf для декодирования CAN для загрузки ваших необработанных данных и файла DBC. Таким образом, вы сможете быстро извлечь сообщение, что вы добавили в DBC, и убедитесь с помощью визуального графика, что конструкция выглядит нормально.

Разверните свой DBC: Затем вы можете добавить все оставшиеся сообщения и сигналы CAN, а также комментарии / описания, таблицы значений и т.д. Мы рекомендуем выполнять регулярные проверки, как и раньше, чтобы убедиться, что конструкция в порядке.

Проверка согласованности: Наконец, мы рекомендуем выполнить полную проверку согласованности как в редакторе DBC Vector, CANDB ++, так и Редакторе DBC Kvaser. В CANDB ++ выберите "Проверка файла / согласованности" и следите за критическими ошибками (хотя учтите, что ваш DBC может быть достаточно корректным для большинства инструментов, даже если сообщается о некоторых проблемах). В редакторе базы данных Kvaser, вы можете выбрать каждое сообщение, чтобы быстро находить сигналы с недопустимыми полями (они будут выделены желтым). Как только вы сделано, мы всегда рекомендуем делать на визуальный анализ масштабированных данных, чтобы гарантировать, что мы не имеем, например, порядок байтов, масштабный фактор и ошибки смещения.



простое интроверто в

Может ошибки

Объяснение ошибок CAN Bus - простое введение

Нужно практическое введение к ошибкам CAN bus?

В этом руководстве вы узнаете об основах обработки ошибок CAN, 5 типах ошибок шины CAN, фрейме ошибок CAN и Состояниях ошибок узла CAN.

Чтобы быть практическими, мы также сгенерируем и запишем ошибки CAN в 6 экспериментах.

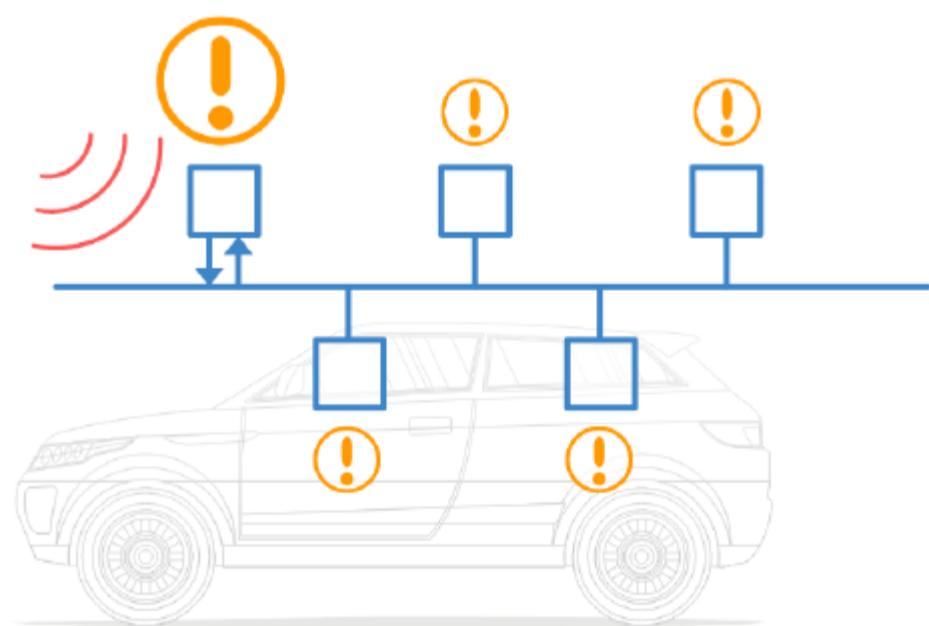
Что такое ошибки шины CAN?

Как объясняется в нашем простом введении к CAN bus, контроллер Локальная сеть сегодня является стандартом де-факто для автомобилей и систем промышленной автоматизации.

Основным преимуществом CAN является надежность, что делает его идеальным для приложений, критически важных для безопасности. Здесь стоит отметить:

Обработка ошибок жизненно важна для надежности CAN.

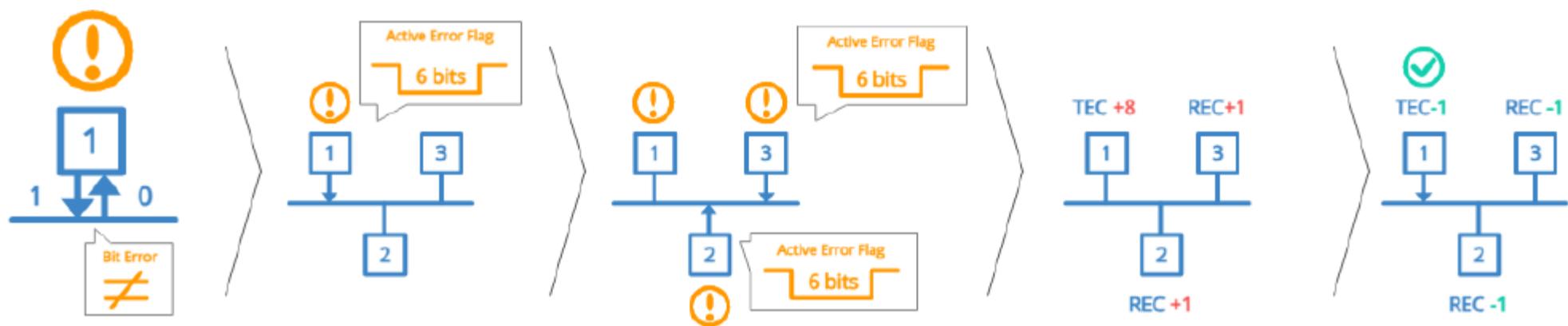
Ошибки шины CAN могут возникать по нескольким причинам - неисправные кабели, шум, неправильное завершение работы, неисправные узлы CAN и т.д. Выявление, классификация и устранение таких ошибок CAN является ключом к обеспечению непрерывной работы всей системы CAN .



В частности, обработка ошибок идентифицирует и отклоняет ошибочные сообщения, позволяя отправителю повторно передать сообщение. Кроме того, процесс помогает идентифицировать и отключать узлы CAN, которые постоянно передают ошибочные сообщения.

Как работает обработка ошибок CAN?

Обработка ошибок является встроенной частью стандарта CAN и каждого контроллера CAN. Другими словами, каждый узел CAN обрабатывает идентификацию и локализацию неисправностей одинаково. Ниже мы привели простой наглядный пример:



Пошаговый пример

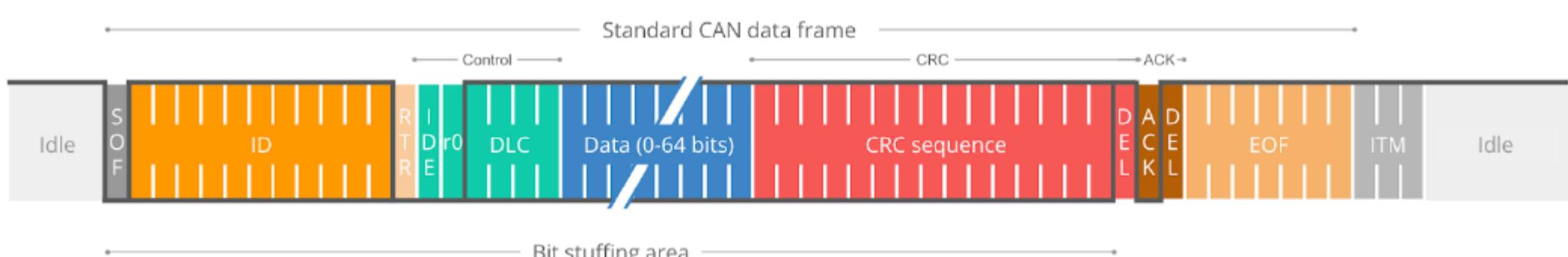
1. CAN-узел 1 передает сообщение на шину CAN - и считывает каждый отправляемый бит
2. При этом он обнаруживает, что один бит, который был отправлен доминирующим, был считан рецессивным
3. Это "Ошибка в битах", и узел 1 поднимает активный флаг ошибки для информирования других узлов
4. На практике это означает, что узел 1 отправляет последовательность из 6 доминирующих битов на шину
5. В свою очередь, 6 доминирующих битов рассматриваются другими узлами как "Ошибка заполнения битов"
6. В ответ узлы 2 и 3 одновременно поднимают активный флаг ошибки
7. Эта последовательность поднятых флагов ошибок составляет часть "CAN error frame"
8. CAN узел 1, передатчик, увеличивает свой "Счетчик ошибок передачи" (TEC) на 8
9. МОГУТ ли узлы 2 и 3 увеличить свой "Счетчик ошибок приема" (REC) на 1
10. CAN-узел 1 автоматически повторно передает сообщение - и теперь успешно
11. В результате узел 1 уменьшает свой TEC на 1, а узлы 2 и 3 уменьшают свой REC на 1.

В примере упоминается ряд концепций, которые мы вскоре подробно рассмотрим: фреймы ошибок, типы ошибок, счетчики и состояния.

Фрейм ошибки шины CAN

В иллюстративном примере узлы CAN "поднимают активные флаги ошибки", таким образом, создается "кадр ошибки" в ответ на обнаружение ошибки CAN.

Чтобы понять, как это работает, давайте сначала посмотрим на "обычный" кадр CAN (без ошибок):



Заполнение битов шины CAN

Обратите внимание, что мы выделили "заполнение битов" поперек рамки CAN.

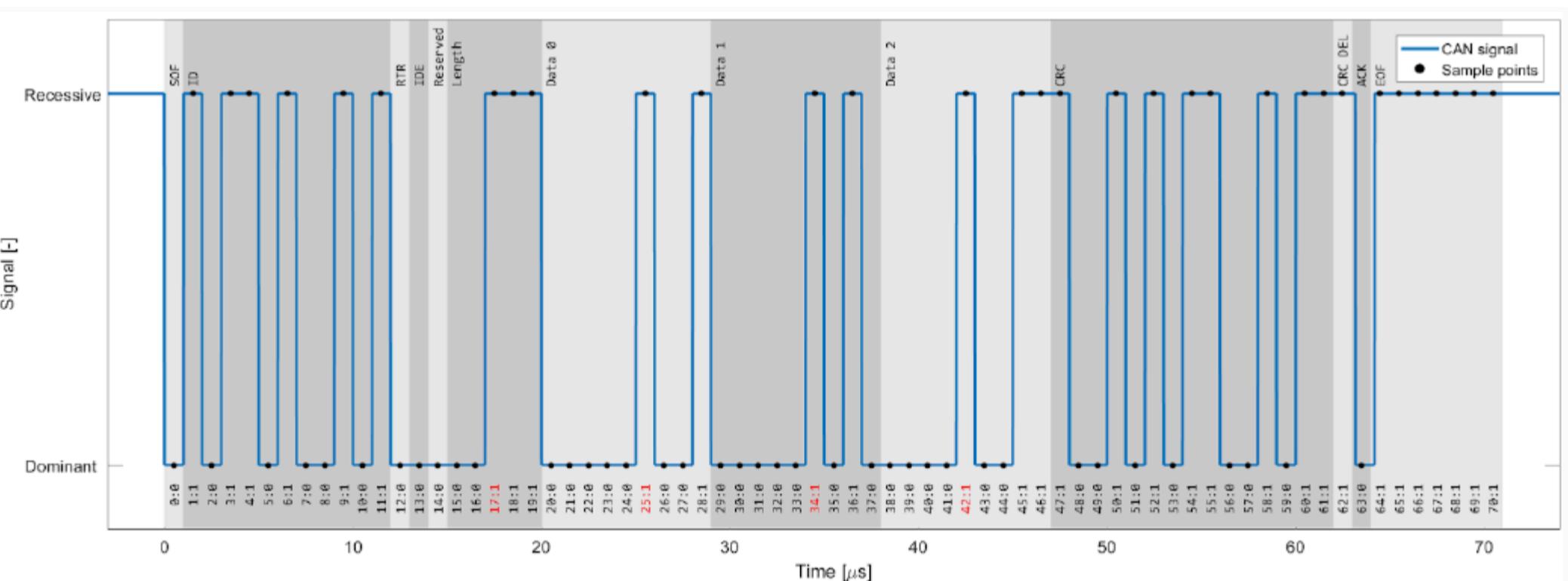
Заполнение битов - это тонкая, но важная часть стандарта CAN. В основном это гласит, что всякий раз, когда узел CAN отправляет пять битов одного и того же логического уровня (доминантного или рецессивного), он должен отправлять один бит противоположного уровня. Этот дополнительный бит автоматически удаляется принимающими узлами CAN. Этот процесс помогает обеспечить непрерывную синхронизацию сети.



Согласно предыдущему примеру, когда узел CAN 1 обнаруживает ошибку во время передачи сообщения CAN, это немедленно передает последовательность из 6 битов того же самого логический уровень - также называемый повышением активного флага ошибки.

Пример осциллографа.

Если мы измерим передачу кадра CAN с помощью осциллографа и оцифруем результат, мы также сможем увидеть исходные биты на практике (см. Красные временные метки):



Активные флаги ошибок

Как мы только что узнали, такая последовательность является нарушением правила заполнения битов - она же "Ошибка заполнения битов". Кроме того, эта ошибка видимый для всех узлов CAN в сети (в отличие от "Битовой ошибки", которая привела к поднятию этого флага ошибки). Таким образом, повышение флагов ошибок можно рассматривать как способ "глобализации" обнаружения ошибки, гарантирующий, что каждый узел CAN будет проинформирован. Обратите внимание, что другие узлы CAN будут видеть флаг активной ошибки как ошибку заполнения битов. В ответ они также поднимают флаг активной ошибки.

Как мы вскоре объясним, важно различать флаги ошибок. В частности, первый флаг ошибки (из "обнаруживающий" узел) часто упоминается как "первичный" активный флаг ошибки, в то время как флаги ошибок последующих "реагирующих" узлов называются "вторичными" активными флагами ошибок.

3 Примера фреймов ошибок CAN

Давайте рассмотрим три примера сценариев:

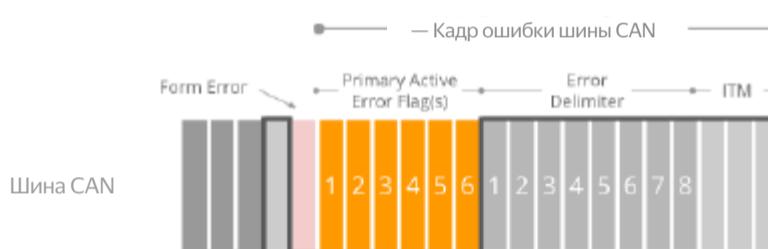
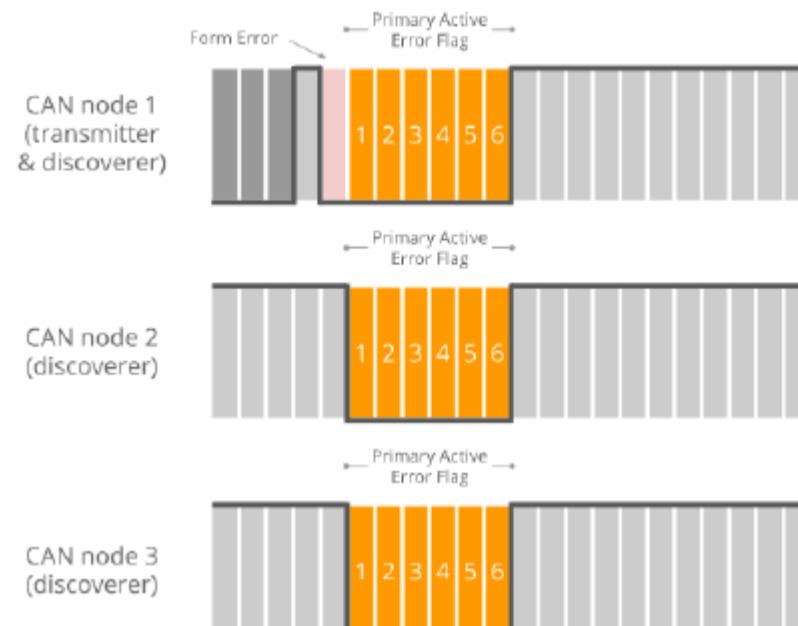
Пример 1: 6 битов флагов ошибок

Здесь все узлы CAN одновременно обнаруживают, что произошла ошибка существует в сообщении CAN и одновременно поднимают свои флаги ошибок.

В результате все флаги ошибок перекрываются, и общее значение последовательность доминирующих битов длится в общей сложности 6 битов. Все узлы CAN в этом случае узлы будут считать себя "открывающими" узлы CAN.

Этот тип одновременного обнаружения менее распространен в практике. Однако это может произойти, например, в результате формы Ошибки (например, если разделитель CRC является доминирующим вместо рецессивного), или если CAN-передатчик выдает битовую ошибку во время записи CRC- поля.

Пример 1: 'Активный' кадр ошибки шины CAN (6 бит активных флагов ошибок)

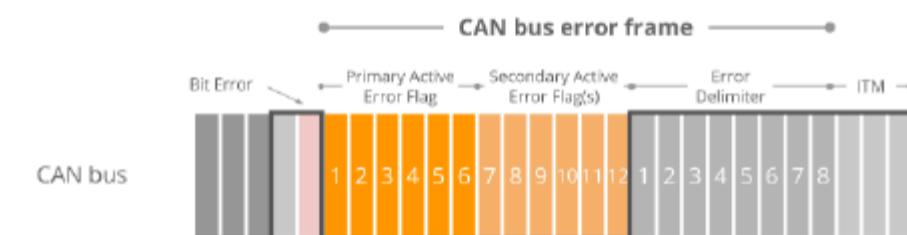
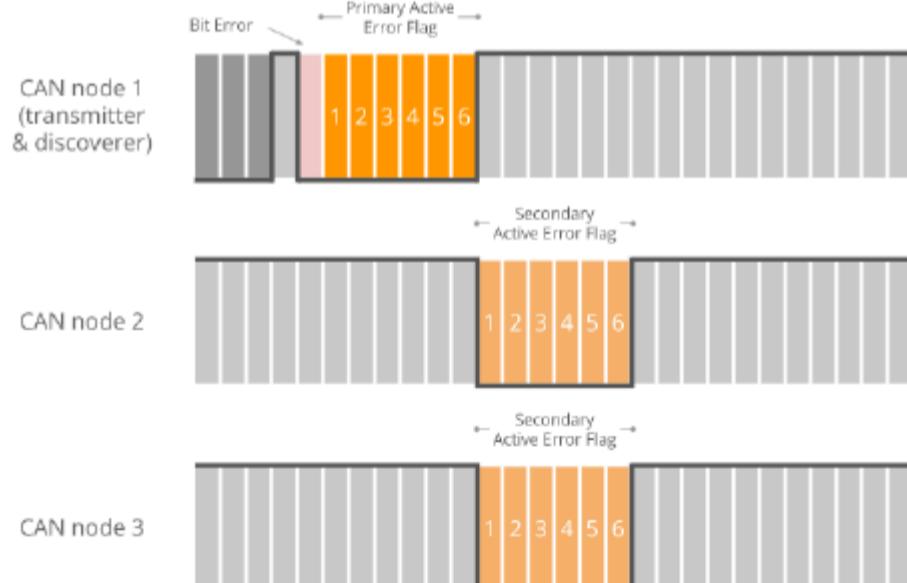


Пример 2: 12 битов флагов ошибок

Здесь, CAN-узел 1 передает доминирующий бит, но считывает его как рецессивный - это означает, что он обнаруживает битовую ошибку. ИТ немедленно передает последовательность из 6 доминирующих битов.

Другие узлы обнаруживают ошибку заполнения битов только после того, как были прочитаны полные 6 битов, после чего они одновременно повышают свои флаги ошибок, что приводит к последующая последовательность из 6 доминирующих битов, то есть всего 12.

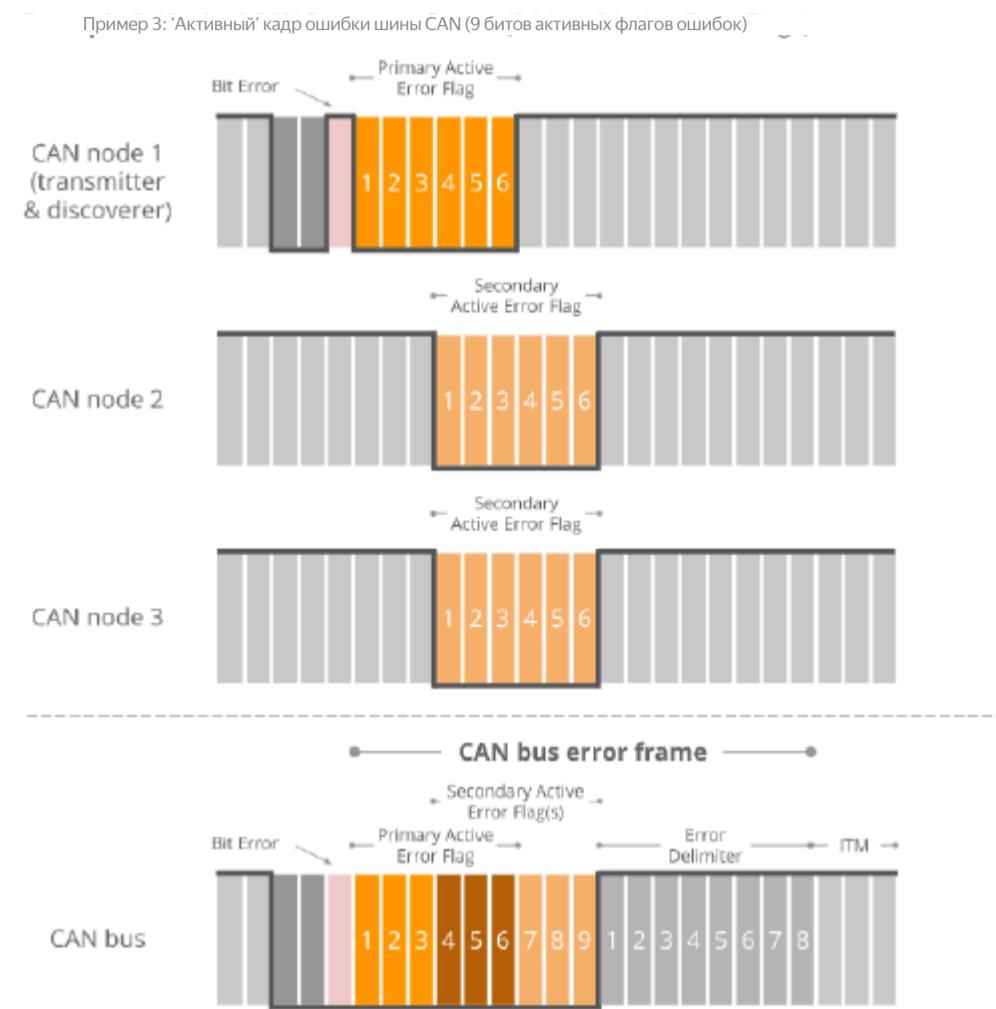
Пример 2: 'Активный' кадр ошибки шины CAN (12 битов активных флагов ошибок)



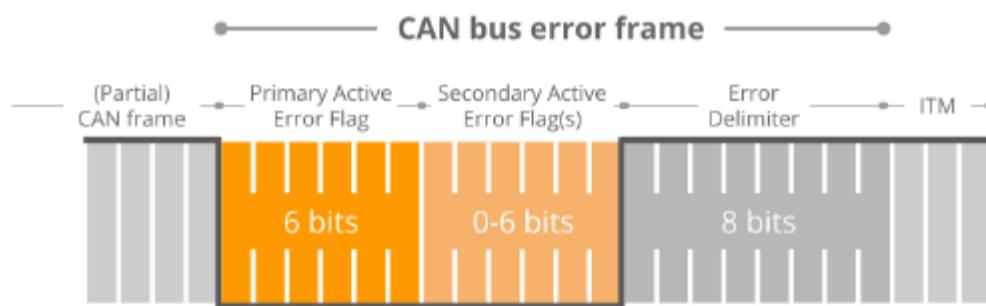
Пример 3: 9 битов флагов ошибок

Здесь CAN-узел 1 уже передал последовательность из 3 доминирующих бита, когда он обнаруживает битовую ошибку и начинает отправку 6 доминирующих битов.

После того, как на полпути установлен основной активный флаг ошибки, узлы 2 и 3 распознают ошибку заполнения битов (из-за того, что за 3 начальными доминирующими битами следуют еще 3 доминирующие биты), и они начинают повышать свои флаги ошибок. В результате последовательность доминирующих битов из error flags становится длиной 9 бит.



Приведенная выше логика повышения флагов ошибок отражена в том, что мы называем "активным" фреймом ошибки CAN.



Обратите особое внимание на то, как вторичные флаги ошибок поднимаются с помощью различных узлов перекрывают друг друга - и как первичные и вторичные флаги также могут перекрываться. Результатом является то, что доминирующая последовательность битов из поднятых флагов ошибок может быть Длина от 6 до 12 бит.

Эта последовательность всегда завершается последовательностью из 8 рецессивные биты, обозначающие конец кадра ошибки.

На практике активный кадр ошибки может "начинаться" в разных местах в ошибочном кадре CAN, в зависимости от того, когда ошибка обнаружена. Результат, однако, будет таким же: все узлы отбрасывают ошибочный кадр CAN, и передающий узел может попытаться повторно передать неудачный Сообщение.

Пассивные флаги ошибок

Если узел CAN перешел из "активного" состояния по умолчанию в "пассивное" состояние (подробнее об этом в ближайшее время), он сможет вызывать только так называемые "пассивные флаги ошибок". Пассивный флаг ошибки представляет собой последовательность из 6 рецессивных битов, как показано ниже.

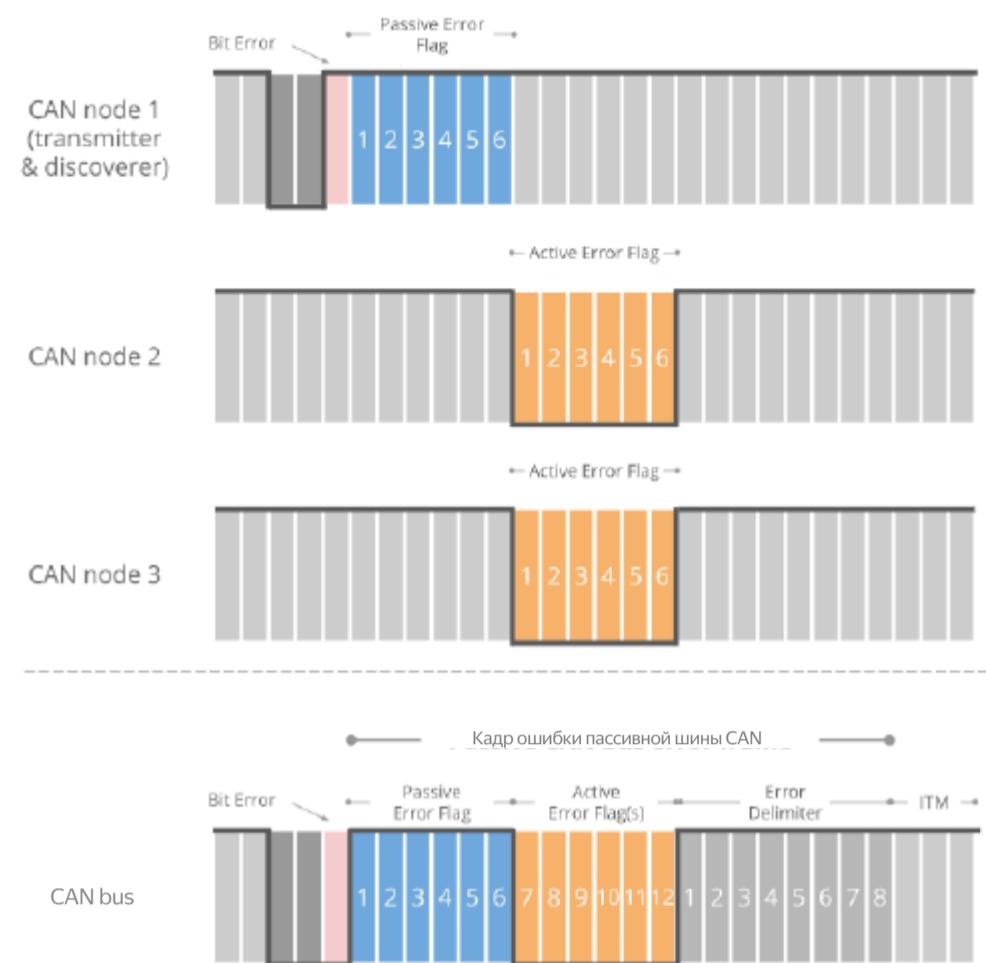
В этом случае важно различать флаг пассивной ошибки, поднятый передающим узлом и принимающим узлом.

Пример 4: Передатчик является пассивным по ошибке

Как показано на рисунке (пример 4), если передатчик (например, CAN node 1 в нашем примере) вызывает пассивный флаг ошибки (например, в ответ на битовую ошибку), это будет соответствовать последовательной последовательности из 6 рецессивных битов.

Это, в свою очередь, определяется как ошибка заполнения битов всеми CAN узлы. Предполагая, что другие узлы CAN все еще находятся в своем Активном состоянии ошибки, они поднимут активные флаги ошибок из 6 доминирующих битов. Другими словами, пассивный передатчик может по-прежнему "сообщать", что кадр CAN является ошибочным.

Пример 4: "Пассивный" кадр ошибки шины CAN (передатчик в пассивном режиме ошибки).

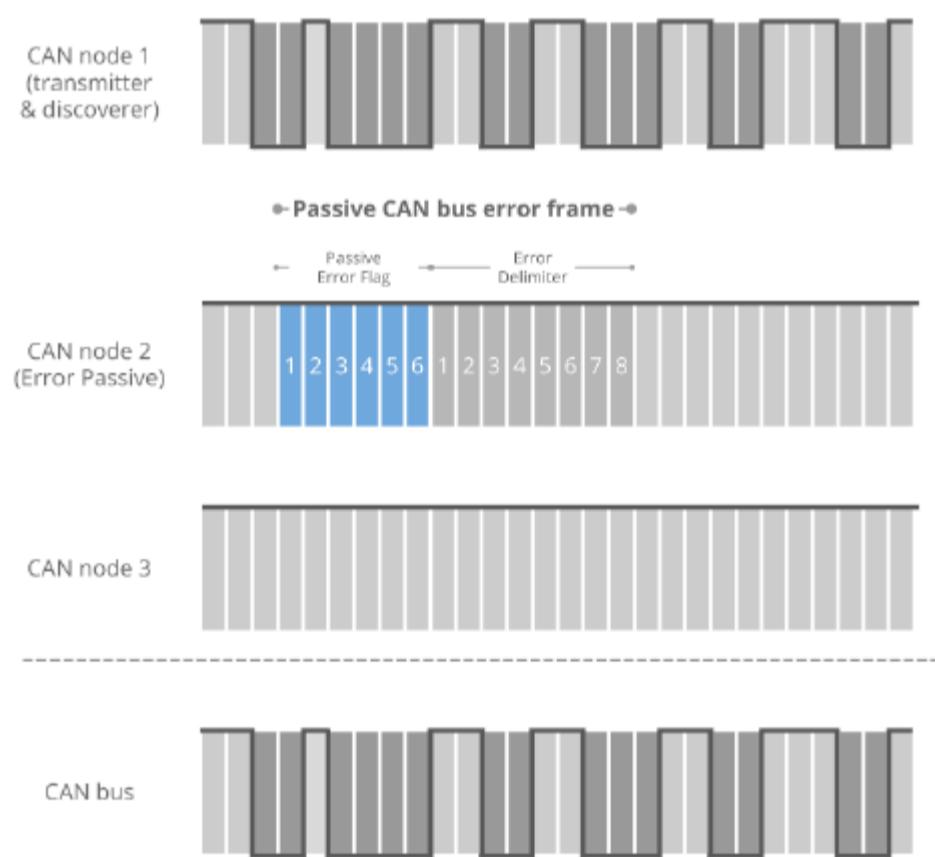


Пример 5: Приемник является пассивным по ошибке

В отличие, если приемник выдает пассивный флаг ошибки, это на практике "невидимо" для всех других узлов CAN нашине (поскольку любые доминирующие биты выигрывают последовательность рецессивных битов) - смотрите также пример 5.

Фактически это означает, что ошибка пассивного приемника отсутствует longer имеет возможность уничтожать кадры, передаваемые другими узлами CAN.

Пример 5: "Пассивный" кадр ошибки шины CAN (приемник находится в пассивном режиме ошибки)



Типы ошибок CAN

Далее давайте посмотрим, какие ошибки могут привести к тому, что узлы CAN поднимут флаги ошибок.

Протокол шины CAN определяет 5 типов ошибок CAN:

1. Ошибка в битах [Передатчик]
2. Ошибка заполнения битов [приемник]
3. Ошибка формы [приемник]
4. Ошибка подтверждения (ACK) [Передатчик]
5. Ошибка CRC (циклическая проверка избыточности) [Приемник]

Мы уже рассматривали битовые ошибки и ошибки заполнения битов вкратце, оба из них оцениваются на битовом уровне. Остальные три типа ошибок CAN оцениваются на уровне сообщений.

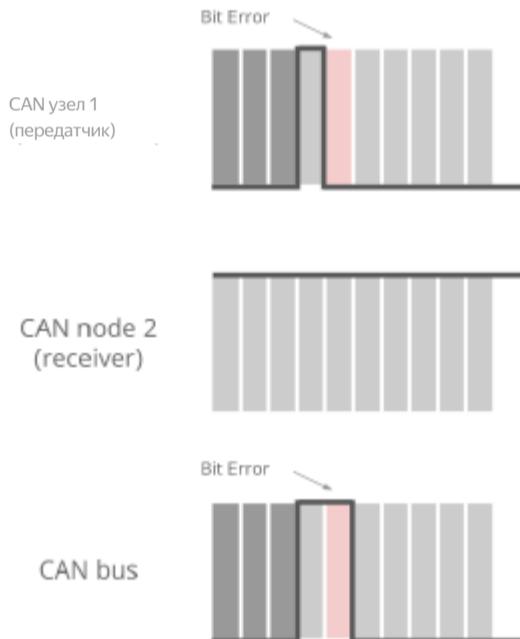
Ниже мы подробно описываем каждый тип ошибок:

Типы ошибок шины CAN

1	Bit Error	Узел передает доминирующий / рецессивный бит, но считывает обратно противоположный логический уровень
2	Bit Stuffing Error	Узел обнаруживает последовательность из 6 битов тот же логический уровень между SOF и CRC
3	Form Error	Узел обнаруживает бит недопустимого логического уровня в полях SOF / EOF или разделителях ACK / CRC
4	ACK Error	Узел передает сообщение CAN, но ACK слот не является доминирующим для получателя (ов)
5	CRC Error	Узел вычисляет CRC сообщения CAN, который отличается от значения переданного поля CRC

Пример: Ошибка в битах шины CAN

Ошибка в битах # 1

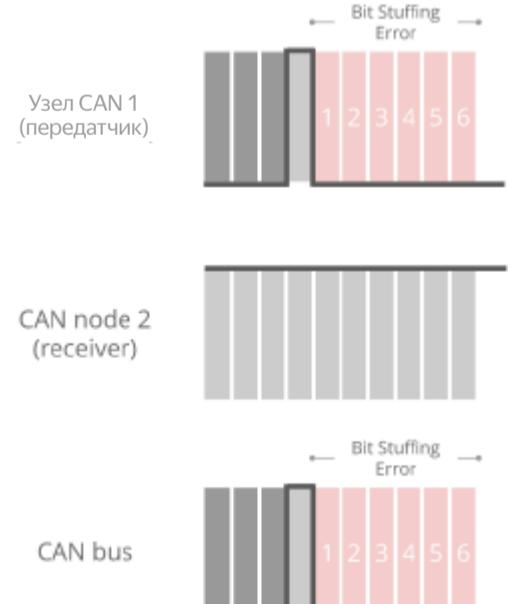


Каждый узел CAN на шине CAN будет отслеживать уровень сигнала в любой момент времени - это означает, что передающий CAN-узел также "считывает обратно" каждый передаваемый им бит. Если передатчик считывает разряд данных другого уровня по сравнению с тем, что он передал, то передатчик определяет это как битовую ошибку.

Если во время арбитражного процесса происходит несоответствие битов (т.е. при отправке CAN ID), оно не интерпретируется как битовая ошибка. Аналогично, несоответствие в слот подтверждения (поле ACK) не вызывает битовой ошибки, поскольку поле ACK в частности, требуется, чтобы рецессивный бит от передатчика был перезаписан доминирующим битом от приемника.

Пример: Ошибка заполнения битов шины CAN

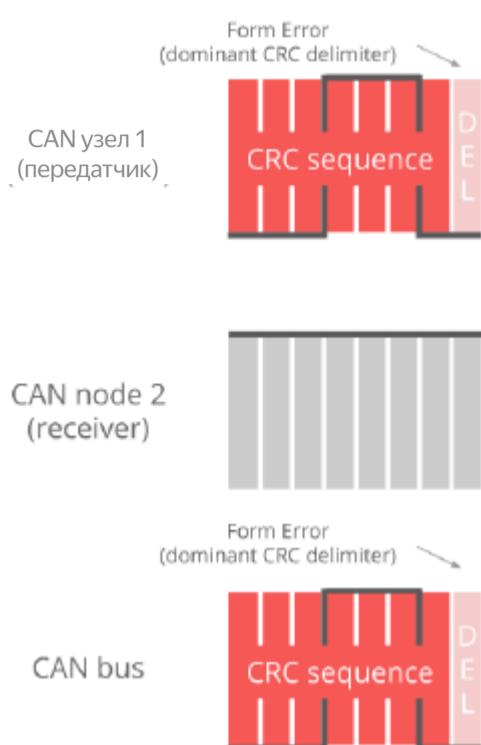
Ошибка заполнения битов # 2



Как объяснялось, заполнение битов является частью стандарта CAN. Это диктует, что после каждого 5 последовательных битов одного и того же логического уровня 6-й бит должен быть дополнением. Это требуется для обеспечения непрерывной синхронизации сети путем предоставления восходящих ребер. Кроме того, это гарантирует, что поток битов не будет ошибочно интерпретирован как ошибка кадра или как межкадровое пространство (7-битная рецессивная последовательность), которое отмечает конец Сообщение. Все узлы CAN автоматически удаляют лишние биты.

Если на шине внутри CAN наблюдается последовательность из 6 битов одного и того же логического уровня сообщение (между полем SOF и CRC), получатель определяет это как заполнение битов Ошибка, также известная как ошибка заполнения.

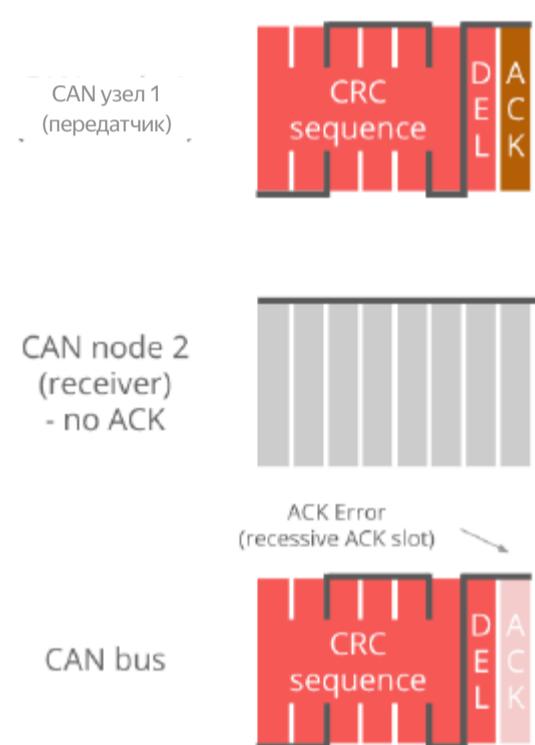
Пример: Ошибка формы шины CAN



Ошибка формы # 3

Эта проверка на уровне сообщения использует тот факт, что определенные поля / биты в сообщении CAN всегда должны иметь определенный логический уровень. В частности, 1-битный SOF должен быть доминирующим, в то время как все 8-битное поле EOF должно быть рецессивным. Далее, ACK и Разделители CRC должны быть рецессивными. Если получатель обнаруживает, что какой-либо из этих битов имеет недопустимый логический уровень, получатель определяет это как ошибку формы.

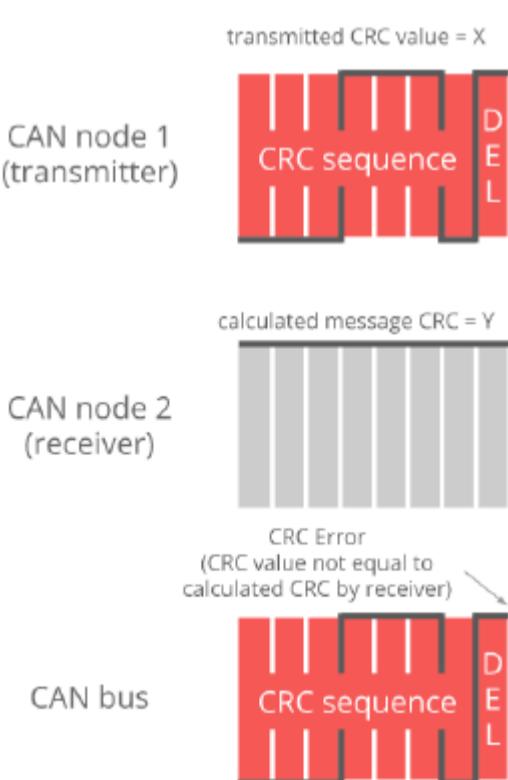
Пример: Ошибка подтверждения на шине CAN



Ошибка #4 ACK (подтверждение)

Когда передатчик отправляет сообщение CAN, оно будет содержать поле ACK (Подтверждение), в котором передатчик будет передавать рецессивный бит. Все ожидается, что прослушивающие CAN-узлы отправят доминирующий бит в этом поле для проверки приема сообщения (независимо от того, заинтересованы ли узлы в сообщение или нет). Если передатчик не считывает доминирующий бит в слоте подтверждения, передатчик определяет это как ошибку подтверждения.

Пример: Ошибка CRC шины CAN



Ошибка CRC # 5 (проверка циклической избыточности)

Каждое сообщение CAN содержит поле контрольной суммы циклической избыточности, равное 15 битам. Здесь, передатчик рассчитал значение CRC и добавил его к сообщению. Каждый принимающий узел также вычислит CRC самостоятельно. Если CRC получателя вычисление не соответствует CRC передатчика, приемник определяет это как CRC Ошибку.

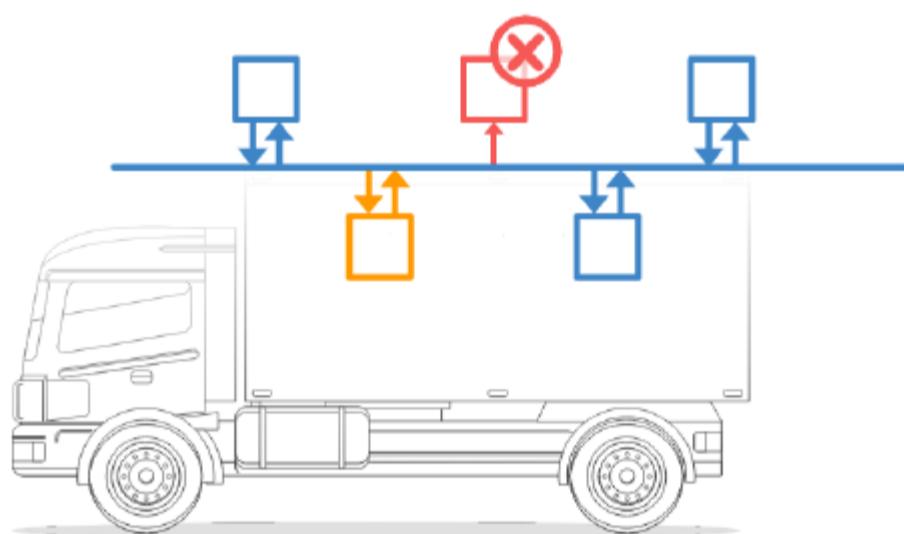
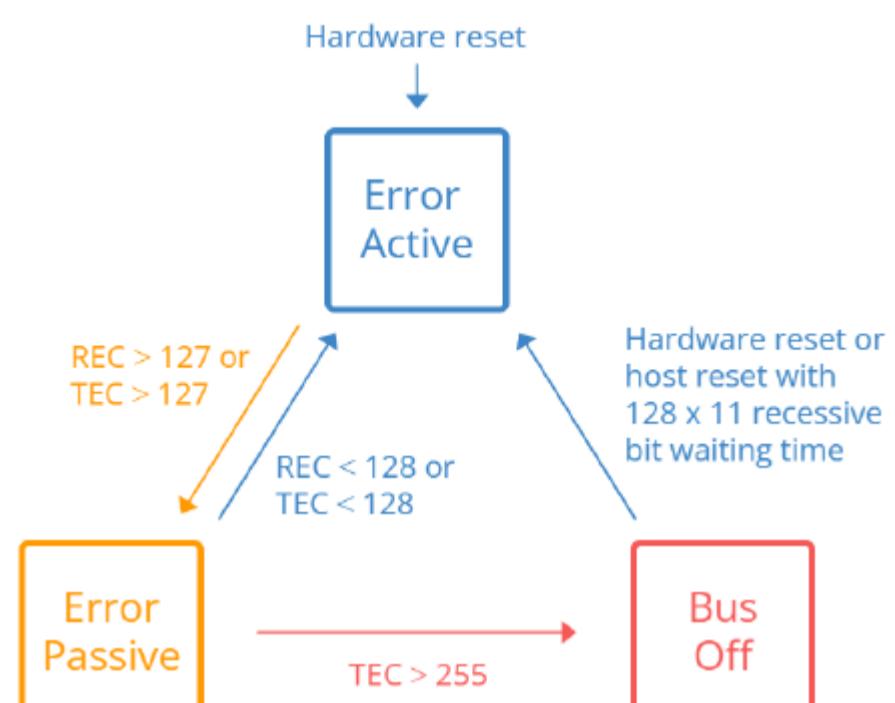
Состояния и ошибки узла CAN счетчики

Как видно, обработка ошибок CAN помогает уничтожать ошибочные сообщения - и позволяет узлам CAN повторить передачу ошибочных сообщений.

Это гарантирует, что кратковременные локальные помехи (например, от шума) не приведут к недействительным/потерянным данным. Вместо этого передатчик пытается повторно отправить сообщение. Если он выигрывает арбитраж (и ошибок нет), сообщение успешно отправлено.

Однако, что, если ошибки вызваны систематическим неисправностью в передающем узле? Это может вызвать бесконечный цикл отправки / уничтожения одного и того же сообщения - заклинивание шины CAN.

Здесь могут использоваться состояния узлов и счетчики ошибок .



Короче говоря, целью отслеживания ошибок CAN является ограничение ошибок путем изящного уменьшения привилегий проблемных Узлов CAN.

В частности, давайте рассмотрим три возможных состояния:

1. Ошибка активна: это состояние по умолчанию для каждого CAN-узла, в котором он способен передавать данные и поднимать "Активные флаги ошибок" при обнаружении ошибок
2. Пассивная ошибка: В этом состоянии узел CAN по-прежнему способен передавать данные, но теперь он выдает "Пассивные" Флаги ошибок при обнаружении ошибок. Кроме того, CAN-узел теперь должен ждать дополнительных 8 бит (он же Время приостановки передачи) в дополнение к 3 времени битового перерыва, прежде чем он сможет возобновить передачу данных (чтобы позволить другим узлам CAN взять на себя управление шиной)
3. Шина выключена: в этом состоянии, узел CAN отключается сам отключается от шины CAN и больше не может передавать данные или поднимать флаги ошибок

Каждый контроллер CAN отслеживает свое собственное состояние и действует соответствующим образом. Узлы CAN меняют состояние в зависимости от значения их счетчики ошибок. В частности, каждый узел CAN отслеживает счетчик ошибок передачи (TEC) и приема (Receive Error) Счетчик (REC):

Узел CAN переходит в пассивное состояние с ошибкой, если REC или TEC превышают 127

Узел CAN переходит в состояние отключения шины, если TEC превышает 255

Как изменяются счетчики ошибок?

Прежде чем мы перейдем к логике того, как создаются счетчики ошибок увеличение / уменьшение, давайте вернемся к фрейму ошибки CAN as a также к флагам первичной / вторичной ошибки.

Как видно из иллюстрации фрейма ошибки CAN, CAN узел, который наблюдает доминирующий бит после своей собственной последовательности из 6 доминирующих битов, будет знать, что он вызвал основную ошибку Отметить. В этом случае мы можем назвать этот CAN-узел "первооткрывателем" ошибки.

Поначалу может показаться положительным наличие CAN-узла, который неоднократно обнаруживает ошибки и оперативно реагирует, поднимая флаг ошибки перед другими узлами. Однако на практике discoverer, как правило, также является виновником ошибок - и следовательно, он наказывается более строго в соответствии с обзором.

Логика счетчика ошибок шины CAN

TEC +8	Transmitter raises primary error flag
REC +8	Receiver raises primary error flag
REC +1	Receiver raises secondary error flag
REC -1	Receiver successfully receives message
TEC -1	Transmitter successfully sends message

TEC: счетчик ошибок передачи, REC: счетчик ошибок приема.

Подробнее о счетчиках TEC / REC

Существуют некоторые дополнения / исключения из вышеуказанных правил, см., Например, Этот обзор.

Большинство из них довольно просты, основаны на нашем предыдущем наглядном примере. Например, кажется очевидным, что CAN-узел 1 увеличил бы TEC на 8, поскольку он обнаруживает битовую ошибку и поднимает флаг ошибки. Другие узлы в этом случае увеличиваются их значение равно 1.

Это интуитивно понятно, что передающий узел быстро достигнет пассивного источника ошибок и, в конечном итоге, шины Выключен, если он постоянно выдает ошибочные сообщения CAN, в то время как принимающие узлы не меняют состояние.

Случай, когда получатель поднимает флаг первичной ошибки, может показаться нелогичным. Однако это может быть, например, случай, когда CAN-узел получателя работает со сбоями таким образом, что это приводит к неправильному обнаружению ошибок в действительном CAN Сообщения. В таком случае получатель повысил бы флаг первичной ошибки, фактически вызвав ошибку. Альтернативно, это может произойти в случаях, когда все узлы CAN одновременно повышают флаги ошибок.



Устройство регистрации данных и ошибок CAN / LIN

CANedge1 позволяет легко записывать данные с 2 x Шины CAN / LIN на SD-карту объемом 8-32 ГБ, вкл. поддержка регистрации ошибок CAN / LIN. Просто подключите его, например, к автомобилю грузовик, чтобы начать регистрацию - и расшифруйте данные с помощью бесплатного [программного обеспечения / API](#). Кроме того, CANedge2 (Wi-Fi) и CANedge3 (3G / 4G) позволяют автоматически передавать данные на ваш собственный сервер и устройства обновления по воздуху.

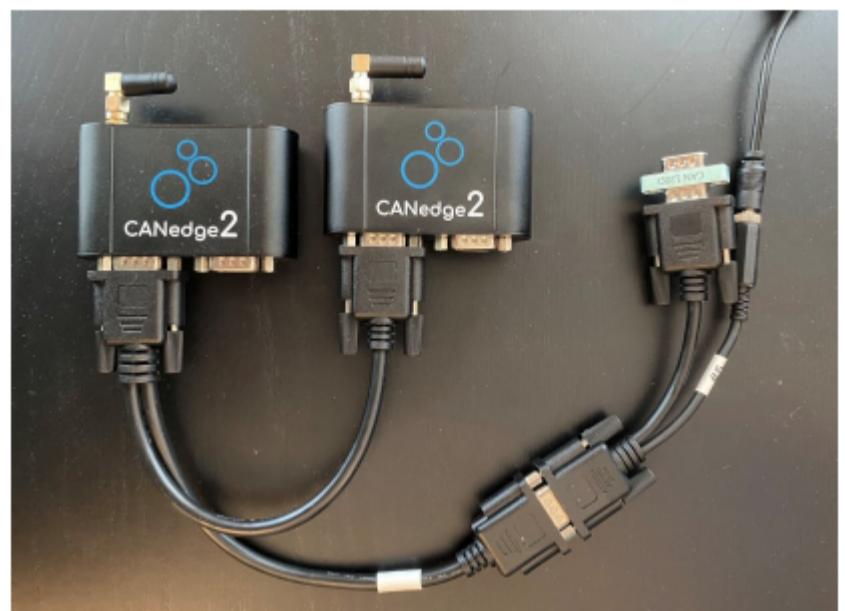
Примеры: Генерация и протоколирование фреймов ошибок

Теперь мы рассмотрели теоретические основы ошибок CAN и обработки ошибок CAN. Далее давайте рассмотрим генерацию и ошибки протоколирования на практике. Для этого мы будем использовать пару устройств CANedge, а для некоторых тестов - устройство PCAN-USB.

Тест # 1: Нет ошибок шины CAN

В качестве эталона мы начнем с теста, в котором нет ошибок шины CAN. Здесь "передатчик" CANedge2 отправляет данные на другой CANedge2 'receiver' - и оба регистрируют ошибки шины CAN.

Загружая файл журнала MF4 в графический интерфейс asammdf, мы проверяем, что никаких ошибок CAN не было во время этого теста произошли ошибки, чего и следовало ожидать.



Тест № 2: Снятие резистора с клеммы шины CAN

В этом teste мы удаляем клемму CAN в середине сеанса ведения журнала. Фактически это соответствует немедленной установке уровня разрядности на доминирующий. В результате передатчик CANedge2 немедленно запускается регистрирует битовые ошибки (которые возникают, когда он пытается передать рецессивный бит, но считывает доминирующий бит). Приемник CANedge2 регистрирует ввод битов Ошибки при обнаружении 6 последовательных доминирующих битов. Эти ошибки записываются до тех пор, пока завершение не будет добавлено снова.

[смотрите онлайн-статью со скриншотами ошибок в asammdf]

Насколько это актуально на практике?

Отсутствие завершения редко является практической проблемой, если вы записываете данные с транспортного средства, станка и т.д. Однако это распространенная проблема при работе с настройками "тестового стенда". Здесь отсутствие завершения может вызвать путаницу, поскольку это может быть трудно отличить от неактивной шины CAN. Если вы сомневаетесь, включение регистрации кадров ошибок на CANedge может быть полезно при устранении неполадок.

Тест № 3: установка неверной скорости передачи данных в бодах

В этом тесте мы настраиваем узел приемника CANedge на передачу в бодах скорость передачи 493,827 Кбит / с по сравнению со скоростью передачи в бодах передатчика 500 Кбит / с. Это довольно значительная разница, приводящая к ошибкам подтверждения для передатчик и ошибкам заполнения битов для приемника.

В более реалистичных сценариях меньшие различия в скорости передачи в бодах конфигурация различных узлов может приводить к прерывистым кадрам ошибок и, следовательно, к потере сообщений.

[смотрите Онлайн-статью со скриншотами ошибок в asammdf]



Насколько это актуально на практике?

Этот пример довольно экстремален. Однако на практике мы иногда видели шины CAN, использующие стандартные скорости передачи данных (250К, 500К, ...), но с определенными настройками синхронизации битов, которые отличаются от тех, которые обычно рекомендуются (и следовательно, используются CANedge). Это приведет не к полному отключению связи, а скорее к периодическому потери кадров составляет несколько процентов. Чтобы решить эту проблему, вы можете создать "повышенную скорость передачи данных" в конфигурации CANedge, по сути, настраивая синхронизацию битов, чтобы лучше соответствовать шине CAN, с которой вы регистрируетесь.

Тест № 4: удаление узла подтверждения CAN

В этом тесте мы используем три модуля CANedge, настроенных следующим образом:

- CANedge1: настроен для подтверждения данных
- CANedge2 A: настроен в "беззвучном режиме" (без подтверждения)
- CANedge2 B: настроен на передачу кадра CAN каждые 500 мс

При настройке по умолчанию данные передаются CANedge2 B на шину CAN и записываются без ошибок. Однако, если мы удалим CANedge1 на шине больше нет узлов CAN для подтверждения кадров отправленных передатчиком.



В результате передатчик обнаруживает ошибки подтверждения. В ответ, это увеличивает его Счетчик ошибок передачи и поднимает активные флаги ошибок на шине CAN. Они, в свою очередь, записываются CANedge2 A (который автоматически контролирует шину) ошибки формы as.

[смотрите онлайн-статью для получения скриншотов ошибок в asammdf]

CANedge записывает ошибки формы из-за того, что передатчик поднимает их после определения отсутствия доминирующего бита в слоте подтверждения. Как только доминирующий бит будет обнаружен получателем в последующем поле EOF (который должен быть рецессивным), обнаружена ошибка формы.

Как очевидно, передатчик передает 16 активных флагов ошибок по мере увеличения его TEC увеличено с 0 до $16 \times 8 = 128$. Теперь передатчик превысил пороговое значение TEC, равное 127, и переходит в пассивный режим с ошибкой. В результате передатчик по-прежнему выдает ошибки подтверждения, но теперь выдает только пассивную ошибку.

Флажки (не видны получателю). На этом этапе передатчик продолжает попытки передать тот же кадр - и приемник продолжает запись эта последовательность повторной передачи.

Насколько это актуально на практике?

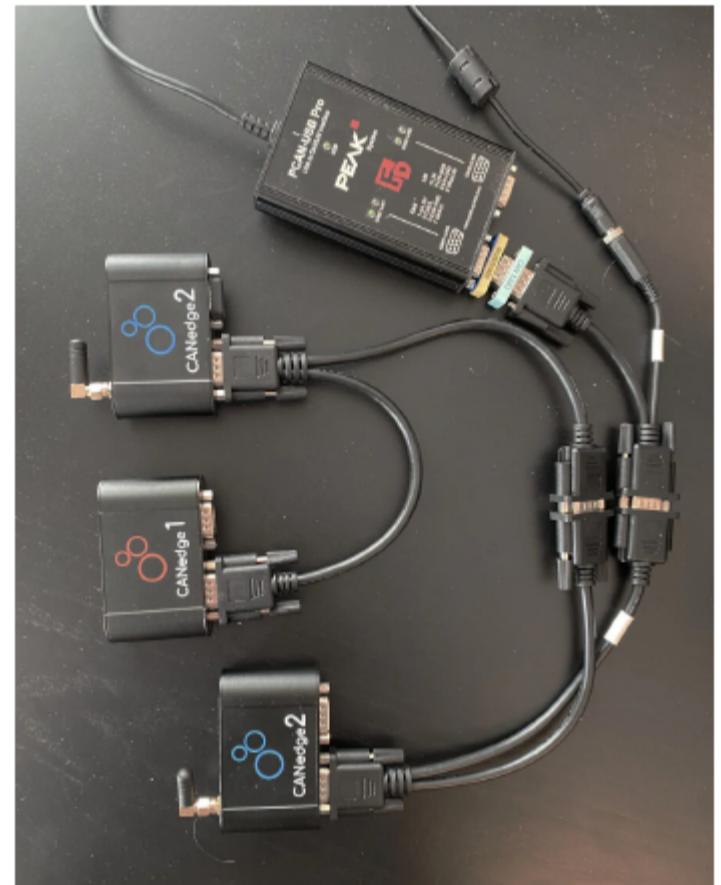
Мы часто сталкиваемся с такого рода ошибками в наших обращениях в службу поддержки. В частности, пользователи могут пытаться использовать наш CAN регистраторы для записи данных с одного узла CAN (например, модуля "датчик-CAN", такого как наш CANmod). Если они решат включите "беззвучный режим" на CANedge при такой установке никакие узлы CAN не будут подтверждать работу единственного узла CAN широковещательные данные - и результатом будут либо пустые файлы журналов, либо файлы журналов, заполненные повторными передачами того же CAN кадр (обычно с очень высокой частотой).

Тест № 5: коллизии кадров CAN (без повторной передачи)

При настройке шины CAN важно избегать перекрытия идентификаторов CAN. В противном случае это может привести к коллизии кадров, поскольку два узла CAN могут оба поверить, что они выиграли арбитраж - и, следовательно, начнут передавать свои кадры одновременно в одно и то же время.

Чтобы смоделировать это, мы используем ту же настройку, что и в тесте №4. Кроме того, мы подключаем устройство PCAN-USB в качестве дополнительного передатчика.

Передатчик CANedge2 теперь настроен на вывод одного кадра CAN каждые 10 мс с идентификатором CAN 1 и полезной нагрузкой в восемь байтов 0xFF. Далее мы настройте CANedge2 так, чтобы отключить повторную передачу кадров, которые были прерваны из-за ошибок. PCAN-USB выводит идентичный CAN-кадр каждые 2 мс с 1-м байтом полезной нагрузки, измененным на 0xFE. Устройство PCAN включило повторные передачи.



Такая настройка быстро создает коллизию кадров, в результате чего CANedge и Передатчики PCAN обнаруживают битовую ошибку. В ответ на это оба поднимают вопрос Активный флаг ошибки, который определяется CANedge как ошибка заполнения битов приемник. Устройство PCAN немедленно предпринимает попытку повторной передачи и завершается успешно, в то время как CANedge ожидает дальнейшей передачи до тех пор, пока не будет отправлено следующее должно быть отправлено сообщение.

[смотрите Онлайн-статью для получения скриншотов ошибок в asammfd]

Насколько это актуально на практике?

Такого рода ошибки, конечно, никогда не должны возникать, например, в автомобиле, поскольку процессы проектирования и тестирования гарантируют, что все Узлы CAN обмениваются данными с помощью глобально уникальных идентификаторов CAN. Однако эта проблема может легко возникнуть, если вы установите третий дополнительное устройство (например, модуль "датчик-CAN") для ввода данных в существующую шину CAN. Если вы не обеспечите глобальную уникальность идентификаторов CAN внешних узлов CAN может привести к конфликтам кадров и, следовательно, к ошибкам нашине CAN. Это особенно важно, если ваш внешний CAN-узел передает данные с высокоприоритетными CAN-идентификаторами, поскольку в этом случае вы можете повлиять на критически важные для безопасности CAN-узлы.

Тест № 6: коллизии CAN-фреймов (вкл. повторная передача)

В этом тесте мы используем ту же настройку, что и раньше, но теперь мы включаем повторную передачу на передатчике CANedge2.

В этом случае коллизия кадров приводит к последовательности последующих коллизий кадров, поскольку и CANedge2, и Устройство PCAN-USB пытаются повторно передать свои прерванные сообщения.

Из-за возникающих битовых ошибок оба устройства поднимают в общей сложности 16 активных флагов ошибок, которые обнаруживаются как ошибки заполнения битов бесшумный приемник CANedge2. Затем оба передатчика переходят в пассивный режим ошибки и прекращают выдавать активные флаги ошибки, это означает, что ни один из них не может уничтожить CAN-кадры на шине. В результате одному из передатчиков удастся передать полное сообщение, тем самым положив конец безумию повторной передачи и позволив обоим устройствам возобновить передачу. Однако это длится всего несколько секунд, прежде чем происходит другое столкновение.

Насколько это актуально на практике?

Обработка коллизий - хороший пример того, насколько эффективна обработка ошибок CAN при "отключении" потенциально проблемных последовательностей и предоставлении возможности узлам CAN возобновить связь. Если произойдет коллизия кадров, вполне вероятно, что оба узла CAN будут настроены на попытку повторной передачи, что вызвало бы замятие, если бы не обработка ошибок и ограничение.

Ошибки шины LIN

Подобно ошибкам шины CAN, протокол LIN также определяет набор из четырех типов ошибок, которые мы кратко опишем ниже. В CANedge поддерживает протоколирование кадров ошибок CAN / LIN.

Ошибка контрольной суммы # 1 LIN

Что касается ошибки CAN CRC, этот тип ошибки подразумевает, что узел LIN рассчитал другую контрольную сумму по сравнению с той встроена в рамку шины LIN передатчиком. Если вы используете CANedge в качестве абонента LIN, эта ошибка может указывать, что вы настроили "таблицу кадров" устройства с неверными идентификаторами для некоторых кадров LIN на шине.

Это, в свою очередь, может быть использовано для "обратного проектирования" правильной длины и идентификаторов проприетарных фреймов LIN с помощью пошаговой процедуры. Подробностисмотрите в документации CANedge .

2 Ошибка приема LIN

Они возникают, если определенная часть сообщения LIN не соответствует ожидаемому значению, или если есть несоответствие между то, что передается, и то, что считывается по шине LIN.

3 Ошибка синхронизации LIN

Эта ошибка указывает на недопустимое поле синхронизации в начале кадра LIN. Это также может указывать на большое отклонение между настроенной скоростью передачи данных для узла LIN и скорость передачи данных, определенная из поля синхронизации.

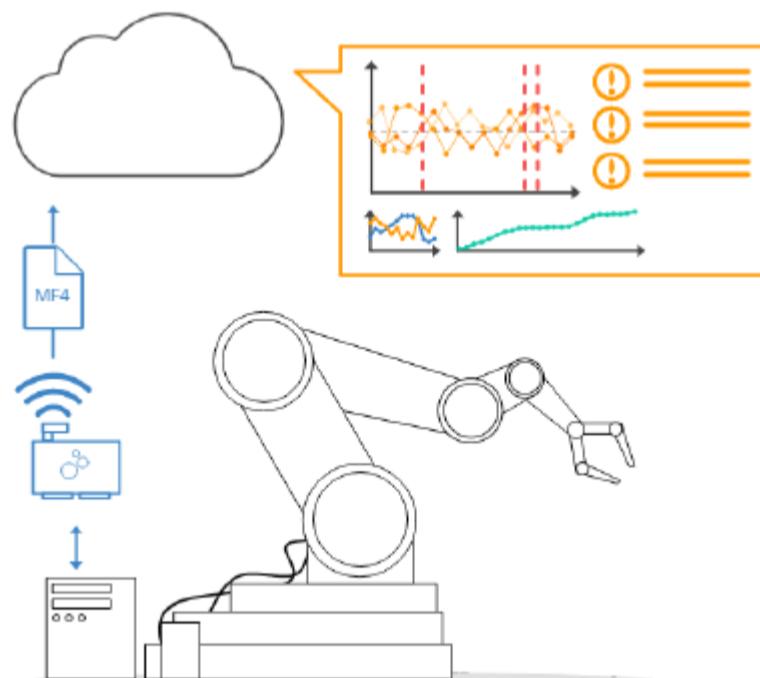
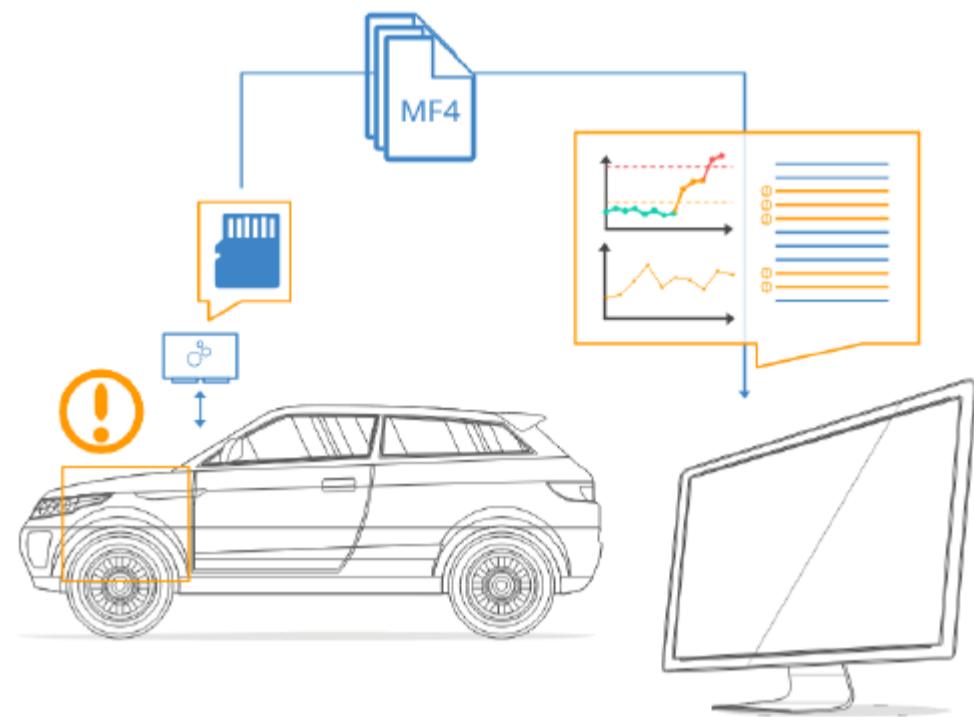
4 Ошибка передачи LIN

Ошибки передачи могут возникать для идентификаторов LIN, зарегистрированных в качестве сообщений ПОДПИСЧИКА. Если узлы не отвечают на сообщение ПОДПИСЧИКА регистрируется ошибка передачи.

Примеры использования для регистрации кадров ошибок CAN

Диагностика шины CAN в автомобилях OEM-производителей

У автомобильного производителя может возникнуть необходимость в записи кадров ошибок CAN в полевых условиях на поздней стадии прототипирования тестирование. Установив CANedge, инженеры-изготовители команды OEM-производителей смогут устранять неполадки на основе фактических сигналов CAN (частота вращения, об / мин, температура) - как также проблемы, связанные с нижним уровнем, МОГУТ связь в их системах-прототипах. Это особенно важно, если интересующие вопросы возникают периодически и, например, происходит только один или два раза в месяц. В таких сценариях интерфейсы шины CAN не очень подходят - и это становится все более актуальным иметь экономичный устройство для масштабируемого развертывания для ускорения устранение неполадок.



Удаленное устранение неполадок МОЖЕТ приводить к ошибкам в оборудовании

Производителю или пользователю вторичного рынка может потребоваться фиксировать редкие События CAN с ошибками на своих компьютерах. Для этого они развертывают CANedge2 для записи данных CAN и связанных с ними ошибок кадры - и автоматически загружают данные через Wi-Fi на их собственный облачный сервер. Здесь ошибки выявляются автоматически и команде инженеров отправляется оповещение, чтобы немедленно проведите диагностику и устранили проблему.

Вопросы и ответы

Поддерживает ли CANedge все типы ошибок CAN / LIN?

Да, CANedge может записывать все типы ошибок CAN / LIN. Однако в настоящее время он не регистрирует свой собственный счетчик ошибок статус, поскольку это считается менее значимым с точки зрения ведения журнала.

Будет ли CANedge поднимать флаги ошибок?

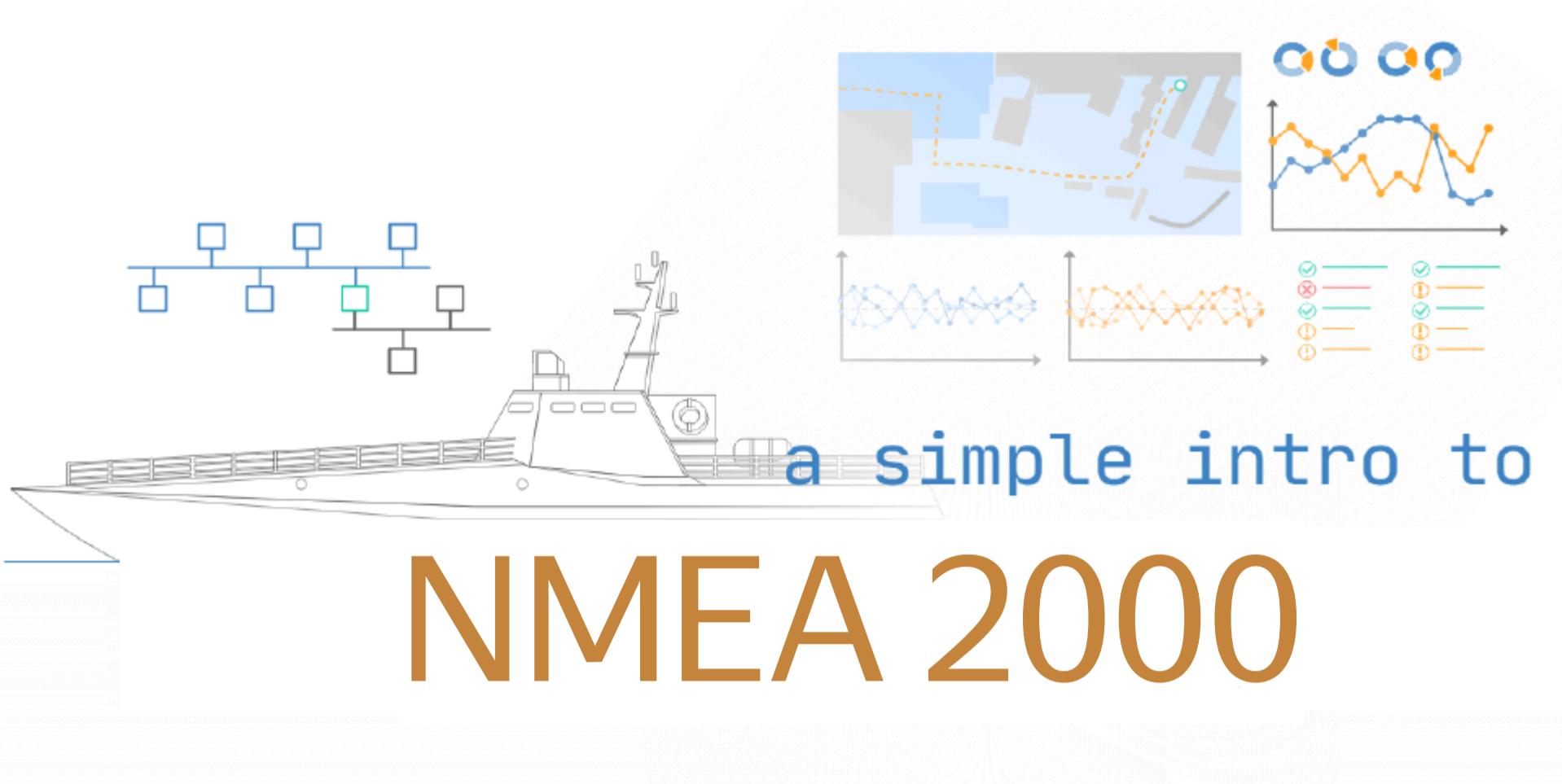
CANedge может передавать флаги ошибок на шину CAN только в том случае, если она настроена в "обычном" режиме, в котором она также способна передавать сообщения. В режиме "ограниченного доступа" он может прослушивать CAN-кадры и подтверждать CAN-кадры, но не поднимает активные флаги ошибок нашине. В режиме "мониторинга" (он же "беззвучный режим") он может прослушивать трафик шины CAN, но не подтверждать сообщения и не поднимать активные флаги ошибок. CANedge всегда будет записывать внутренние кадры ошибок CAN / LIN.

Какая информация может быть записана относительно ошибки CAN / LIN?

Если кадр CAN является ошибочным, что приводит к появлению кадра ошибки, CANedge обычно записывает только тип ошибки - без каких-либо данных, относящихся к ошибочному кадру (помимо временной метки). Одно исключение из этого правила касается подтверждений ошибки, при которых CANedge по-прежнему будет записывать неподтвержденные кадры CAN (включая, из попыток повторной передачи).

Является ли обработка ошибок риском кибербезопасности?

Некоторые исследователи указали на риск того, что "злоумышленники" могут использовать функциональность обработки ошибок шины CAN для принудительное выполнение событий удаленного отключения шины для критически важных для безопасности ECU. Это хороший пример того, почему регистраторы данных CAN bus & интерфейсы, подобные CANedge2, с удаленной передачей данных по воздуху и обновлениями, должны быть высокозащищенными (см. Также наше введение в CAN cybersecurity). Для получения хорошего обзора удаленной атаки с отключением от шины, посмотрите это вступление Адриана Колера.



Объяснение NMEA 2000 - простое введение

Нужно простое, практическое введение в NMEA 2000?

В этом руководстве мы представляем протокол N2K (IEC 61162-3), используемый на морских судах, таких как катера, яхты, буксиры и другие суда. В частности, мы рассмотрим стандарты NMEA 2000, ключевые концепции, ссылки на другие протоколы (NMEA 0183, CAN / J1939), топология и соединители. Мы также рассматриваем PGN NMEA 2000 и быстрые пакеты.

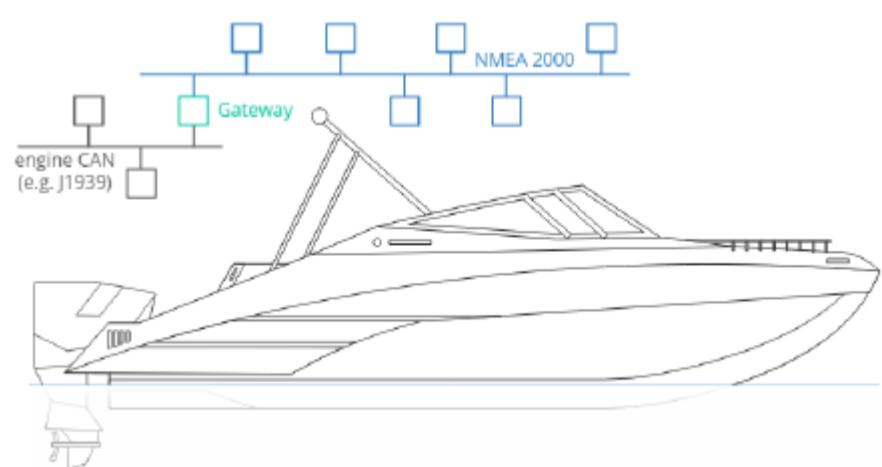
Чтобы это было практично, мы также объясняем, как записывать и декодировать данные NMEA 2000 - с практическими примерами использования и . Узнайте больше ниже!

Что такое NMEA 2000?

NMEA 2000 (IEC 61162-3) - это стандарт связи, используемый в морской отрасли для подключения, например, двигателей, приборов и датчики на лодках. Он основан на локальной сети контроллера (CAN) и позволяет отправлять / получать данные между устройствами по одному сетевому "магистральному" кабелю.

Функциональность более высокого уровня NMEA 2000 (он же N2K) основана на SAE J1939 и ISOBUS (ISO 11783), обычно используемых в тяжелых условиях эксплуатации транспортные средства и сельскохозяйственная / лесная техника соответственно.

NMEA 2000 является преемником более старого NMEA 0183 и сегодня используется на большинстве современных морских судов, таких как катера, лайнеры, яхты и связанные с ними оборудование / датчики.



Несколько лодочных сетей.

Важно отметить, что морское судно может использовать несколько сетей.

Например, двухмоторная скоростная лодка может включать сеть NMEA 2000, соединяющую оба подвесных двигателя с аксессуарами для руля (головка управления, ключи, дисплеи). Эта сеть обычно создается производителем лодки. Двигатели они сами могут осуществлять внутреннюю связь, например, по протоколу J1939, с модулем шлюза / фильтра, передающим связанные с механизмом данные в сеть NMEA 2000.

Кроме того, к той же сети NMEA 2000 могут быть добавлены различные датчики, измеряющие, например, местоположение GPS, ветер, глубину воды и т.д.. Однако существуют также случаи, когда такие датчики подключаются оператором судна к отдельной сети NMEA 2000.

NMEA 2000 против NMEA 0183

NMEA 0183 (IEC 61162-1) является предшественником NMEA 2000 (IEC 61162-3). Организация NMEA опубликовала Протокол NMEA 0183 с целью стандартизации связи между различными электронными устройствами оборудования на морских судах.



Протокол заменил NMEA 0180 и NMEA 0182.

Протокол NMEA 0183 активно используется и сегодня, хотя NMEA 2000 стал отраслевым стандартом.

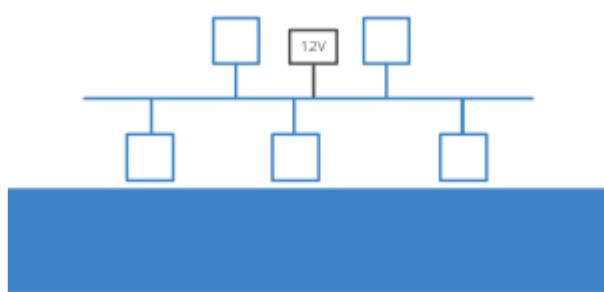
Подробная информация о NMEA 0183

Протокол NMEA 0183 использует электрический стандарт RS232 / RS422 для связи вместо шины CAN. В этом при настройке один "говорящий" (например, модуль GPS) может взаимодействовать с несколькими "слушающими" узлами (например, автопилотом и картплоттером). Однако невозможно иметь несколько "говорящих" в одной сети (в отличие от CAN). В результате для каждого узла "говорящего" требуется отдельная сеть NMEA 0183, что быстро усложняется, когда сети взаимосвязаны.

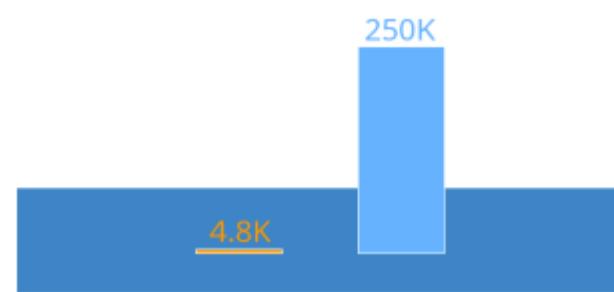
Данные передаются в формате ASCII в виде стандартизованных "предложений", определенных на прикладном уровне. Данные передаются в виде доступных для печати символов ASCII в диапазоне от 0x20 до 0x7E и отражают "физические значения" (например, градусы), а не чем "исходные данные", как в NMEA 2000. Подробный обзор структуры сообщения / предложения смотрите в этой таблице.

Основные преимущества NMEA 2000 по сравнению с NMEA 0183

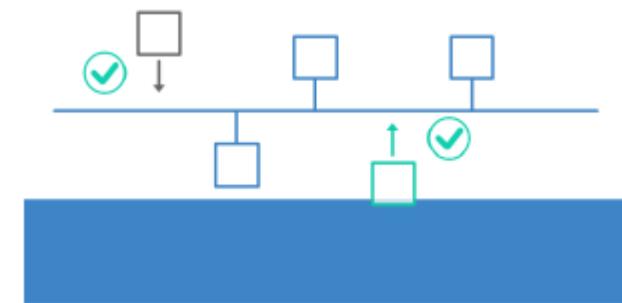
В целом, NMEA 2000 является гораздо более современным и гибким протоколом связи по сравнению с NMEA 0183 и поэтому сегодня широко используется в морских электронных системах. Ниже мы перечислим ключевые преимущества:



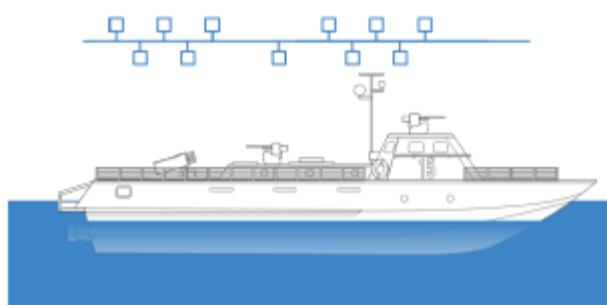
Более простая и недорогая сеть
NMEA 2000 позволяет всем узлам взаимодействовать друг с другом (несколько переговорных устройств), а питание источника питания напрямую интегрирован в магистральную сеть, что снижает затраты на подключение и упрощает сеть по сравнению с NMEA 0183. Стандартизованные Т-образные соединители также упростят интеграцию



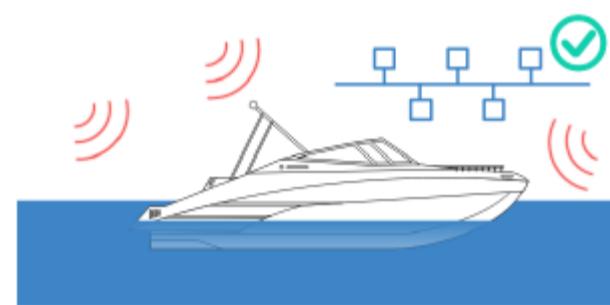
Более быстрая передача данных
NMEA 2000 обеспечивает более высокую скорость передачи данных по сравнению с NMEA 0183 (250 КБ против 4,8 КБ), что означает, что он может передавать больше данных за заданный промежуток времени. Это полезно, когда в режиме реального времени требуются данные, например, для навигации или приложений мониторинга двигателя



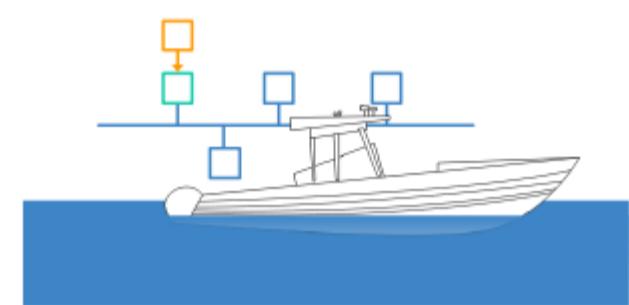
Улучшенная совместимость с устройствами
NMEA 2000 разработана таким образом, чтобы разные устройства из разных производителей должны быть подключены к одной сети и обмениваться данными стандартизированным способом. Это упрощает смешивание и сопоставление устройств на той же сети и снижает риск проблем с совместимостью



Больший размер сети
Сети NMEA 2000 могут быть больше чем сети NMEA 0183, поскольку они предназначены для поддержки большего количества устройств (до 50 узлов) и передавать данные на большие расстояния. Это может быть полезно на более крупных судах или системах где используется большее количество устройств
необходимо подключиться



Более надежная связь
NMEA 2000 использует область контроллера Сетевая шина (CAN), которая является цифровой протокол связи, который менее восприимчив к шуму и помехам чем последовательная связь протокол, используемый NMEA 0183. Это делает сети NMEA 2000 более надежными и отказоустойчивыми



Обратная совместимость
NMEA 2000 позволяет интегрировать Устройства NMEA 0183 по низкой цене шлюзы, позволяющие интегрировать существующую электронику в новые сети. Однако обратное, неверно - т.е. узлы NMEA 2000 не может быть добавлен в NMEA 0183 сети

История NMEA и сертификация

Национальная ассоциация морской электроники (NMEA) - некоммерческая организация основана в 1957 году. Она представляет производителей, дистрибуторов и дилеров в морской отрасли и помогает обеспечить совместимость устройств между брендами / производителями посредством продвижения обмена данными стандартизация, обучение и сертификация.

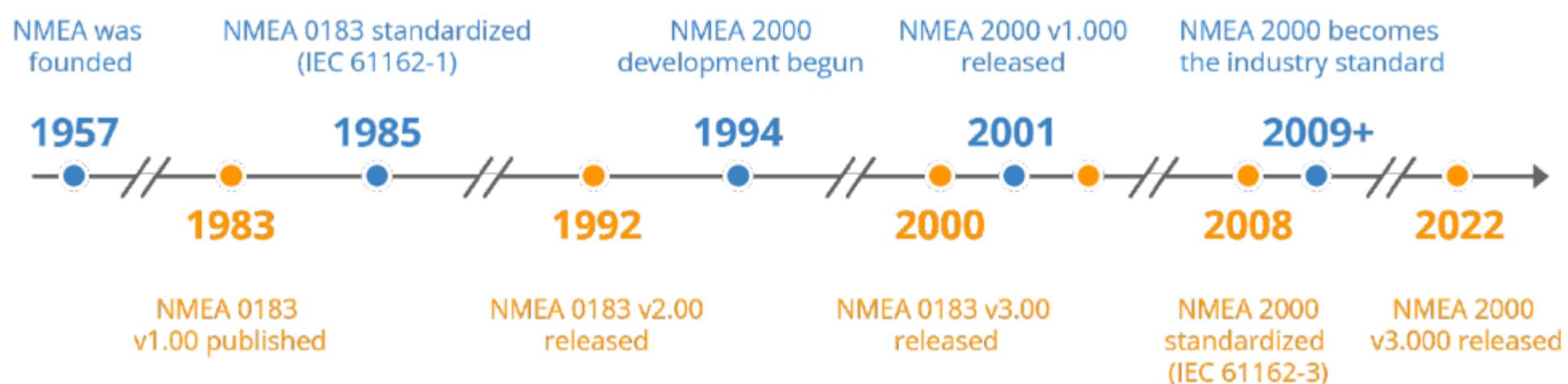
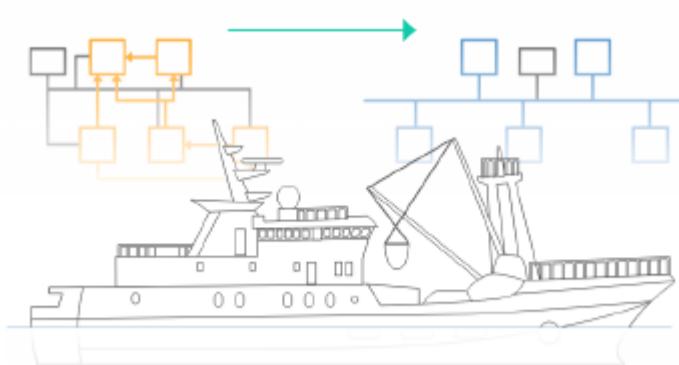


NMEA и AEF

В некоторой степени роль NMEA можно сравнить с ролью Фонда электроники сельскохозяйственной промышленности (AEF) в сельское хозяйство / лесная промышленность. Как NMEA 2000, так и ISOBUS разделяют потребность в строгом соответствии протоколу требования, необходимые для реализации обещания интеграции продуктов разных производителей по принципу plug & play. Здесь организации играют ключевую роль в содействии стандартизированному процессу сертификации продукции, а также в предоставлении базы данных для идентификации соответствующих производителей / продукции.

История NMEA

1957: основана Национальная ассоциация морской электроники.
1983: NMEA 0183 v1.00 опубликовано NMEA
1985: NMEA 0183 стандартизирован как IEC 61162-1
1992: выпущен NMEA 0183 v2.00
1994: Комитетом по стандартам начата разработка N2K
2000: выпущен NMEA 0183 версии v3.00
2001: выпущен NMEA 2000 версии v1.000
2008: NMEA 2000 стандартизирован как IEC 61162-3
2008: выпущен NMEA 0183 v4.00
~ 2009: NMEA 2000 становится отраслевым стандартом
2022: выпущен NMEA 2000 v3.000



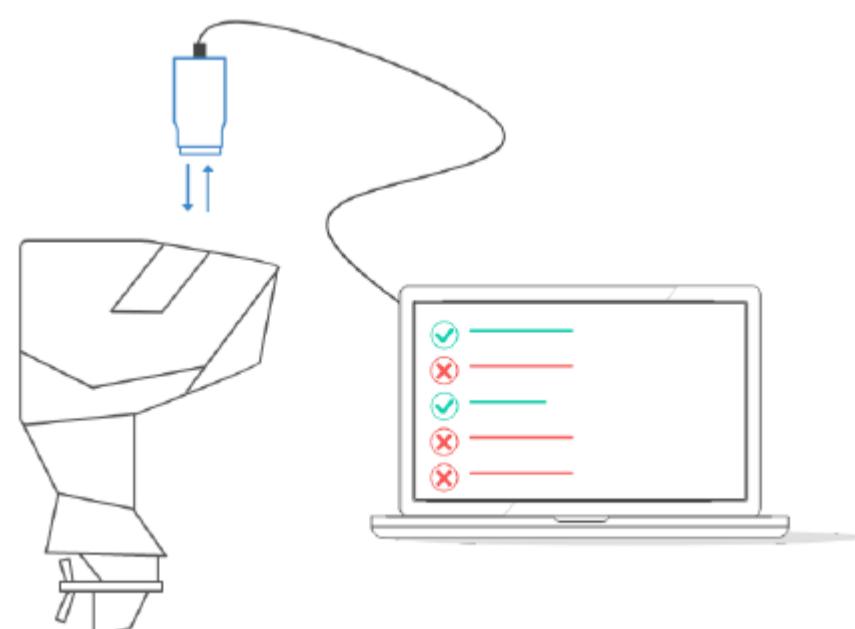
Сертификат соответствия NMEA 2000



Чтобы обеспечить совместимость узлов N2K, NMEA упрощает сертификацию процесса, посредством которого продукт может получить "Сертификат NMEA 2000" и добавить к продукту логотип сертификации. На практике сертификация проводится с помощью процесса самосертификации с использованием обязательного испытательного оборудования (NMEA Инструмент сертификации 2000 года, то есть интерфейс / программное обеспечение CAN) и валидация со стороны NMEA. Сертификация не подразумевает, что конкретный продукт должен выдавать определенные данные. Скорее, это гарантирует совместимость продукта с другими Продукты, сертифицированные NMEA 2000, что позволяет им сосуществовать в одной и той же сети без создания помех.

Классы и уровни сертификации NMEA

Сертификация продукции NMEA 2000 может быть выдана как как класса 1, так и класса 2, с уровнями А и В. Устройства класса 1 используют один интерфейс NMEA 2000 для связи, в то время как устройства класса 2 предоставляют два интерфейса с целью резервирования. Для каждого класса Уровни А и В позволяют различать сложные устройства (с полной поддержкой связи включая, например, реализацию связи ISO TP) и более упрощенные устройства с меньшим объемом требуемых функций коммуникационные функции.



Одобрено NMEA 2000 по сравнению с сертифицированными

Некоторые продукты сторонних производителей могут использовать альтернативные классификации из "Сертифицированных NMEA 2000":

1. Одобрено NMEA 2000: Эти продукты способны реализовать NMEA 2000 и прошли физические аппаратные тесты программы сертификации N2K
2. Сертификат NMEA 2000 для использования с [Утвержденным наименованием продукта]: Эти продукты могут использоваться в сочетании с конкретным продуктом, одобренным NMEA 2000, что в совокупности создает полностью совместимый продукт.
3. Сертификат NMEA 2000 для использования с [Название продукта Certified Gateway]: Эти продукты могут использоваться в сочетании с продуктом gateway, сертифицированным NMEA 2000.

Наконец, еще одна классификация - "Диагностический инструмент, одобренный NMEA 2000 - не для постоянного подключения к NMEA 2000" объединительная плата" - предназначена для инструментов, которые при неправильном использовании могут, например, нарушить критически важную для безопасности функциональность в сети NMEA 2000.

7-уровневая модель OSI | NMEA 2000

Модель NMEA 2000 OSI & стандарты

Ниже мы кратко обрисуем связь между каждым уровнем модели OSI и Стандартом NMEA 2000 (а также ссылочными стандартами).

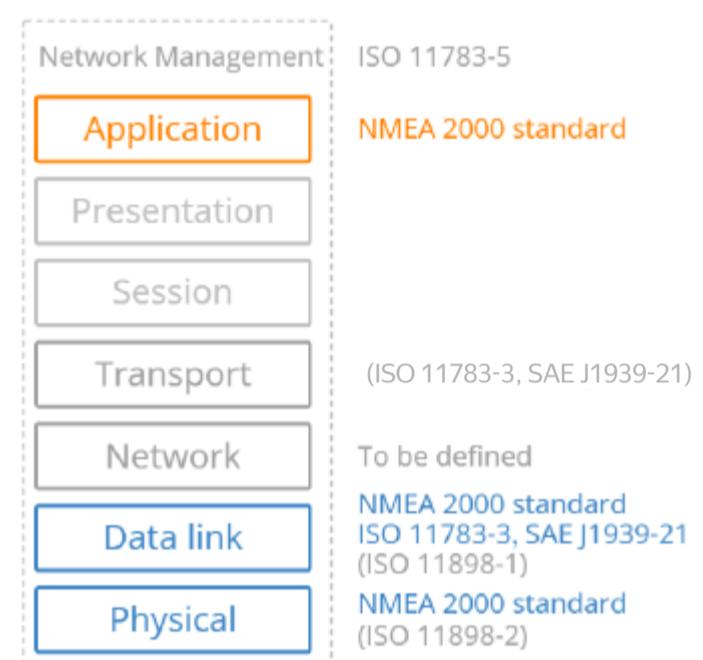
Применение: Этот уровень полностью определен стандартом NMEA 2000 и определяет одобренные PGN / сигналы, включая. положение для фирменных сообщений конкретного производителя. Приложение А NMEA определяет PGN, в то время как Приложение В предоставляет полный список стандартизованные PGN NMEA 2000 и сигналы

Транспорт: В NMEA 2000 транспортный уровень отвечает за установление и поддержание соединений между устройствами, as а также фрагментацию и повторную сборку сообщений по мере необходимости. Это может быть выполнено с помощью транспортного протокола, используемого в ISOBUS / J1939, или с помощью быстрых пакетов NMEA (см. Также уровень канала передачи данных).

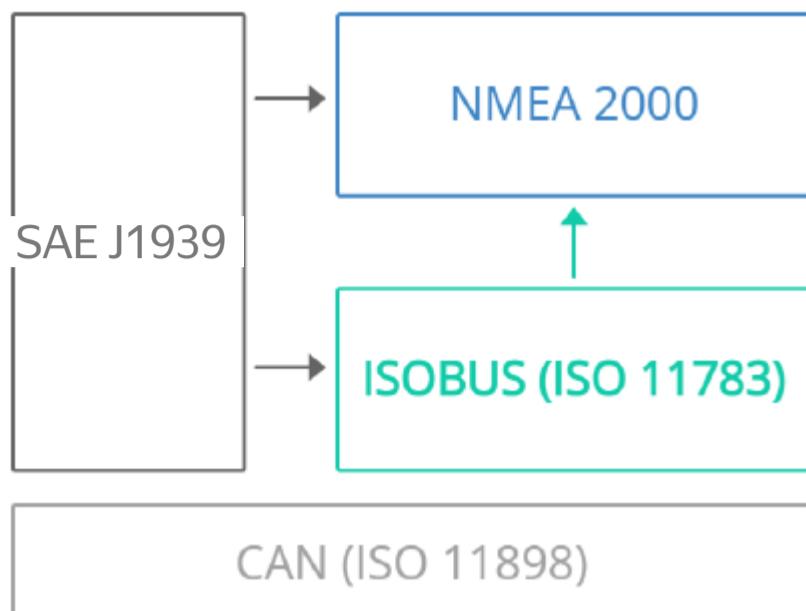
Сеть: Этот уровень будет дополнительно определен в будущем в Стандарт NMEA 2000

Канал передачи данных: Этот уровень отвечает за передачу данных по каналу связи и обеспечение правильной передачи данных . В стандарте NMEA 2000 используется CAN протокол на канальном уровне передачи данных, который включает обнаружение ошибок и механизмы исправления для обеспечения надежной передачи. Обратите внимание, что этот уровень ссылается на ISO 11783-3 (ISOBUS), SAE J1939-21 и 11898-1 (CAN) Стандарт NMEA 2000 определяет дополнительные требования/функциональность, включая быстрый пакет NMEA для многокадровая связь

Физический уровень: этот уровень полностью определяется стандартом NMEA 2000, включая напряжения сигнала, кабели и разъемы. Обратите также внимание на включение стандарта 11898-2 (CAN) для отражения роли шины CAN в качестве основы NMEA 2000. Здесь также указана стандартная скорость передачи данных в бодах 250K для NMEA 2000 и подробная информация о кабелях и разъемах



Кроме того, "Управление сетью" определено ISO 11783-5 (ISOBUS). Обратите особое внимание, что все устройства, совместимые с NMEA 2000, должны поддерживать динамический запрос адреса



NMEA 2000 по сравнению с J1939 по сравнению с ISOBUS

Очевидно, что NMEA 2000 тесно связан с J1939 и ISOBUS. Здесь мы кратко разъясним связь между этими протоколами. Во-первых, давайте сделаем очень краткий обзор истории.:

В 1994 году Общество инженеров автомобильной промышленности (SAE) опубликовало протокол J1939 для использования в тяжелых условиях эксплуатации Автомобили. Затем ISOBUS был заимствован из J1939 для сельскохозяйственной / лесной техники, официально стандартизированной в 2007 (ISO 11783).

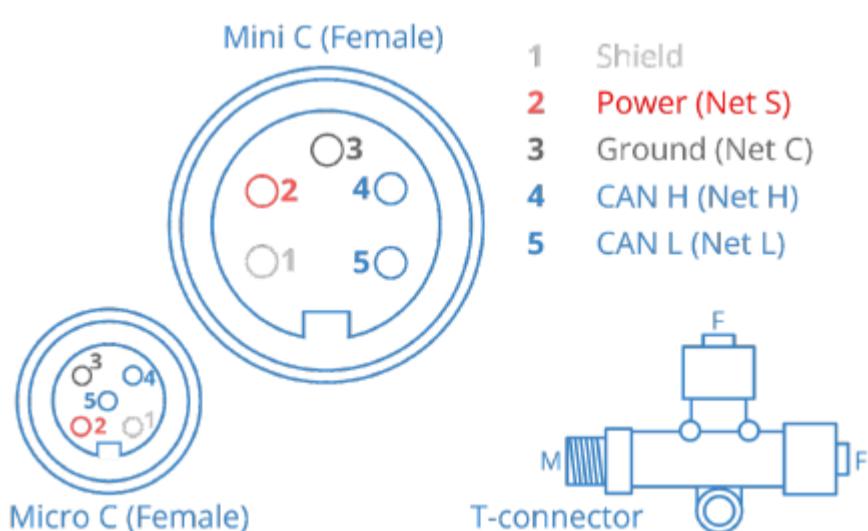
NMEA 2000 был выпущен в 2001 году и ссылается на части SAE J1939 и ISO 11783 практически взаимозаменяемо. ISO Ссылка на 11783-3 используется для уровня канала передачи данных, но, в свою очередь, ссылается на J1939-21. Аналогично, NMEA 2000 ссылается на ISO 11783-5 для управления сетью, который практически идентичен J1939-81. Другими словами, NMEA 2000 основан на стандартах ISO 11783, которые, в свою очередь, основаны на стандартах J1939.

Практическое значение

В практических приложениях NMEA 2000 часто используется в сочетании с J1939 или ISOBUS. Например, некоторые лодки двигатели используют J1939 внутри компании и развертывают шлюз / фильтр для анализа определенной информации в стандартизированном NMEA 2000 PGN. В некоторых случаях, Сообщения NMEA 2000 и J1939 даже существуют на однойшине CAN. Это возможно, поскольку ни одно из PGN NMEA 2000 не пересекается с PGN J1939. Аналогичным образом, сообщения NMEA 2000 часто транслируются на ISOBUS сеть, поскольку многие модули GPS используются как в морских, так и в сельскохозяйственных целях. Смотрите наше введение к ISOBUS для Подробные сведения.

Разъемы NMEA 2000 и топология сети

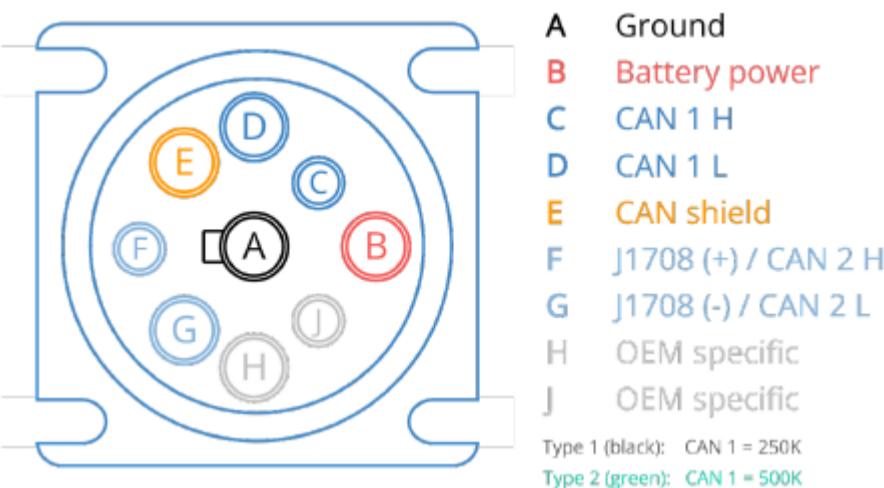
Ниже мы описываем наиболее распространенные разъемы, встречающиеся на морских судах:



Разъем № 1 Т (5-контактный M12)

T-образный разъем (он же тройник) используется при построении "магистральной" шины CAN NMEA 2000 линейным способом и для подключения оборудования. Это трехсторонний разъем состоящий из двух гнездовых разъемов и одного штекерного разъема. Разъемы стандартизированы DeviceNet 5-контактные разъемы M12 с кодировкой А. Физический разъем и распиновка различаются между типами mini (для магистралей) и тип micro (для небольших магистралей и подвесных кабелей). Разъем micro M12 также используется в промышленных таких приложениях, как CANopen, позволяют легко подключать Оборудование на базе CAN - например, регистраторы данных шины CAN.

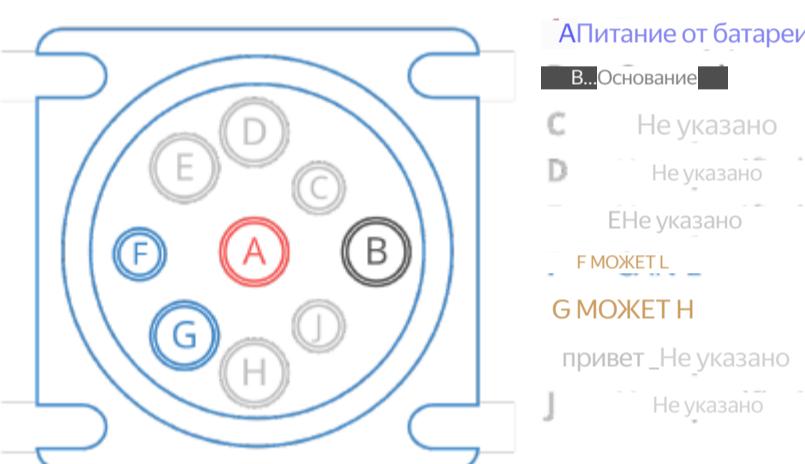
кабель-адаптер m12



Разъем двигателя № 2 (9-контактный J1939)

Многие судовые двигатели (например, от Cummins) используют Протокол J1939 для своей внутренней связи. Часть эта информация может транслироваться на NMEA 2000 подключайтесь к сети напрямую или через шлюз, который преобразует Информацию PGN в соответствующую кодировку NMEA 2000. Многие движки поставляются с отдельной диагностикой разъемы, обеспечивающие прямой доступ к необработанным данным J1939 - и наиболее распространенным разъемом для них был бы 9-контактный разъем J1939 deutsch.

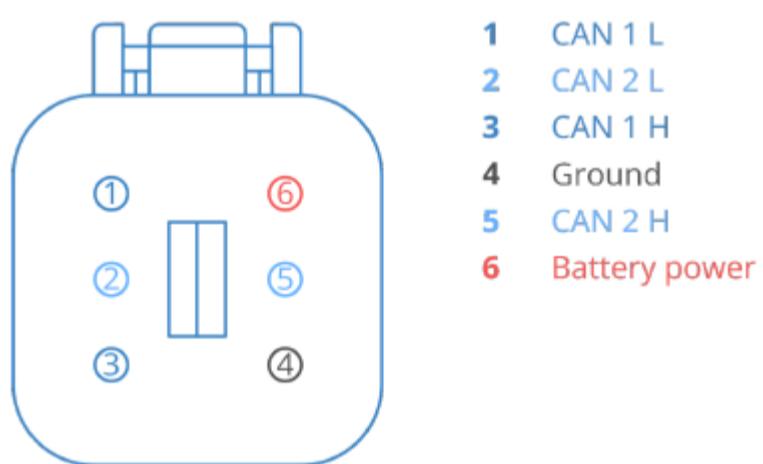
[кабель-адаптер j1939](#)



Разъем двигателя № 3 (9-контактный CAT)

В некоторых морских двигателях (например, от Caterpillar) используется 9-контактный адаптер CAT от deutsch. Он выглядит почти идентично 9-контактному разъему J1939, но использует другую распиновку.

[кабель-адаптер cat](#)

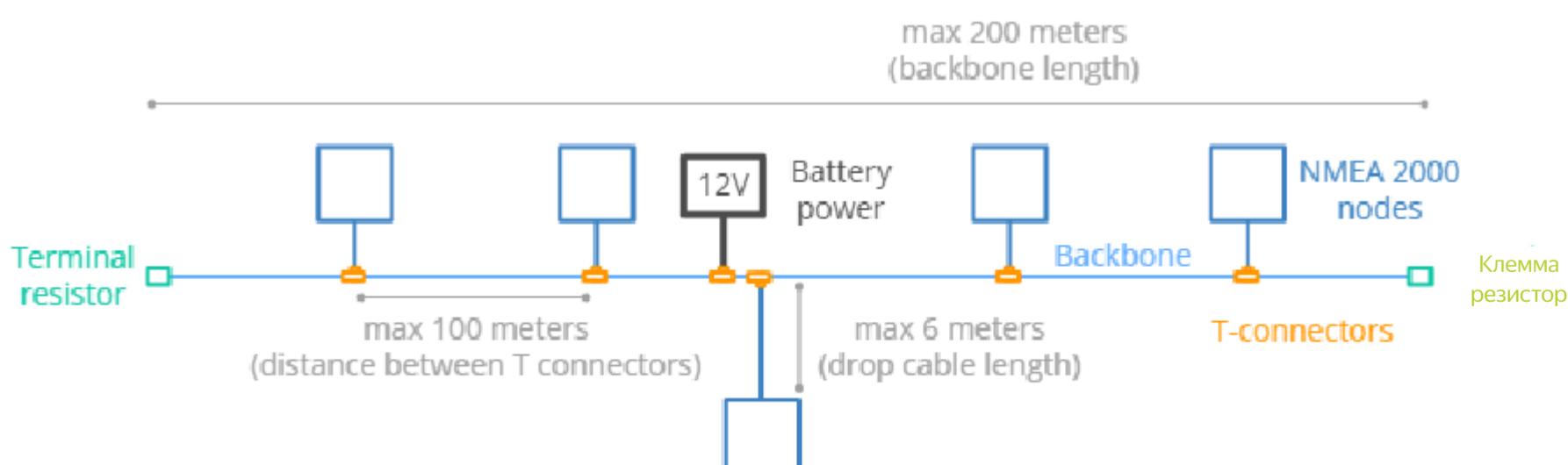


Разъем двигателя № 4 (Deutsch DT06-6P)

В некоторых морских двигателях (например, Volvo Penta), используются альтернативные диагностические разъемы, такие как DT06-6P (мужской 6-контактный разъем) и DT06-6S (женский 6-контактный разъем). Вы часто сможете переключаться с этих разъемов на другие популярные разъемы (например, OBD2) через готовые адаптеры.

Требования к физическому уровню NMEA 2000

Стандарт NMEA 2000 также определяет требования с точки зрения физического уровня, как показано ниже.



Подробная информация о топологии NMEA 2000

Стандартная скорость передачи данных составляет 250 Кб / с

Шина CAN должна быть построена как линейная "магистраль"

Расстояние между двумя Т-образными разъемами не должно превышать 100 м.

Максимальная длина сети составляет 200 м

Сеть поддерживает до 50 подключенных устройств

Максимальная длина "отводного кабеля" (между Т-образным разъемом и устройством) составляет 6 м.

Максимальная суммарная длина кабеля для отвода составляет 78 м

Кабели должны соответствовать стандартам DeviceNet (ODVA).

Магистраль должна включать заземленный источник питания в диапазоне 9-16 В постоянного тока (например от резервуара 12 В постоянного тока источник питания или от изолированного источника питания 15 В постоянного тока)

Три типа кабелей: Мини /толстый (для магистралей, требующих 4-8 ампер на ветку), средний (для магистралей) и микро / тонкий (для подвесных кабелей и небольших магистралей)

Быстрый пакет NMEA 2000

Ключевой частью стандарта NMEA 2000 является "Быстрый пакет".

Согласно SAE J1939-21 и ISO 11783-3, стандарт NMEA 2000 поддерживает передачу нескольких пакетов объемом до 1785 байт в соответствии со стандартом ISO 15765-2 (ISO TP).

Однако для NMEA 2000 часто требуется полезная нагрузка фрейма, превышающая 8 байт, но намного меньшая, чем 1785 байт. Таким образом, быстрый Packet был представлен как более эффективный транспортный протокол для полезной нагрузки среднего размера до 223 байт без каких-либо задержек передачи. Смотрите Обзор таблицы для сравнения.

NMEA 2000 - одиночный кадр против ISO TP по сравнению с быстрыми пакетами

Одиночный кадр	ISO TP (ISO 11783 Multi Packet)	Быстрый пакет NMEA 2000
До 8 байт	До 1785 байт	До 223 байт
Пункт назначения определяется PGN	Отправка любого PGN на определенное устройство	Пункт назначения, определенный PGN
Нет подтверждения связи	Необязательное подтверждение связи	Нет подтверждения связи
Нет задержки протокола передачи	Задержки протокола передачи	Нет задержки протокола передачи
PGN является частью CAN ID	PGN является частью полезной нагрузки данных	PGN является частью CAN ID

Идентификация фрейма

В методологии быстрых пакетов NMEA 2000 каждый фрейм сохраняет исходный идентификатор PGN. Это означает, что многокадровое сообщение может быть однозначно идентифицировано на уровне идентификатора, в отличие от реализации ISO TP в J1939, ISOBUS и UDS, в которых для идентификации сообщения требуется извлечение информации из полезной нагрузки первого кадра CAN данные.

Пример быстрой трассировки пакетов

Ниже мы показываем пример быстрой трассировки пакетов NMEA 2000. Как очевидно, все передаваемые кадры содержат "счетчик последовательности" в 1-м фрагменте и "счетчик кадров" во 2-м фрагменте 1-го байта. Кроме того, первый переданный кадр содержит общую длину полезной нагрузки во 2-м байте. Чтобы включить декодирование, кадр необходимо собрать заново, как показано.

Временная метка (эпоха)	ИДЕНТИФИКАТОР CANPGN (ШЕСТНАДЦАТИБИТНЫЙ)	PGN (десятичный)	Байты данных	условные обозначения
1620692704.079450	11F80F00	1F80F	129039	001B127CEAD5123DNMEA 2000 фунтов стерлингов
1620692704.080000	11F80F00	1F80F	129039	0131F3D0ACF2231A (0x1F80F = Отчет о местоположении класса В АИС)
1620692704.080600	11F80F00	1F80F	129039	0203FFFF00000000 #байты данных (0x1B=27)
1620692704.081200	11F80F00	1F80F	129039	0320FFFF0070FEFF счетчик последовательности
				счетчик кадров
1620692705.079500	11F80F00	1F80F	129039	201B127CEAD5123D данные полезной нагрузки (27 байт)
1620692705.080050	11F80F00	1F80F	129039	2131F3D0ACF2231A
1620692705.080650	11F80F00	1F80F	129039	2207FFFF00000000
1620692705.081200	11F80F00	1F80F	129039	2320FFFF0070FEFF
1620692704.079450	11F80F00	1F80F	129039	127CEAD5123D... 20FFFF0070FEFF
1620692705.079500	11F80F00	1F80F	129039	127CEAD5123D... 20FFFF0070FEFF

CANedge Python API позволяет выполнять автоматическую повторную сборку из быстрых пакетов в единый с полной полезной нагрузкой данных (без счетчиков последовательности и длины полезной нагрузки). Это, в свою очередь, позволяет легко декодировать данные в DBC с помощью [файла DBC NMEA 2000](#). Смотрите наш [репозиторий примеров api](#) примеры сценариев.

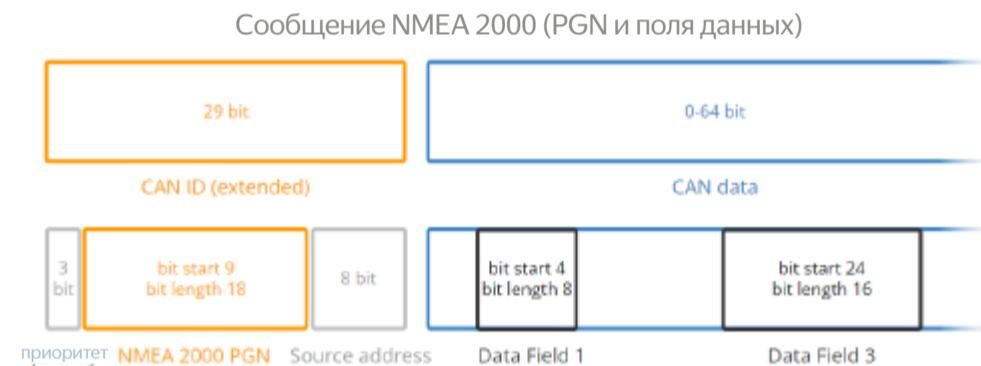


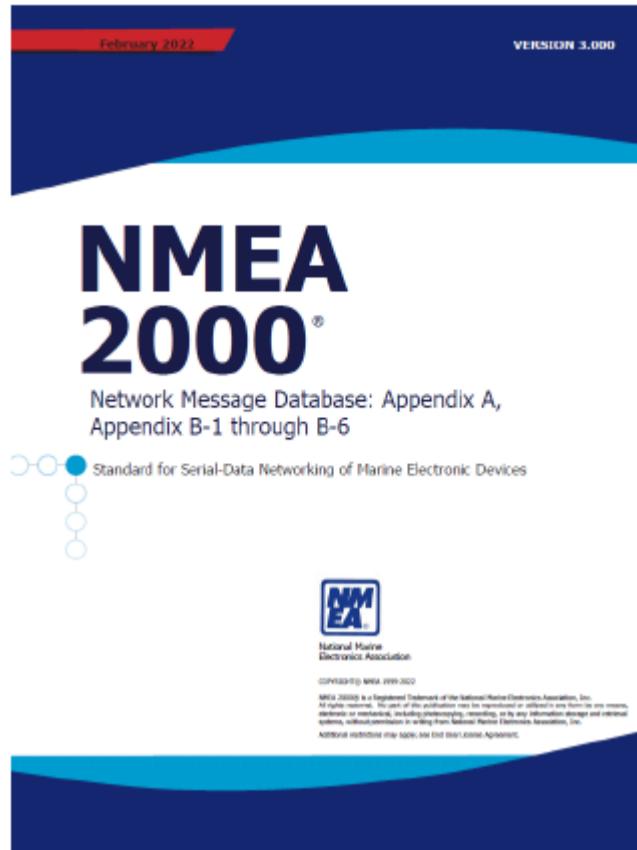
N2K PGN и поля данных

В NMEA 2000 используется концепция группы параметров Числа (PGN) из протокола J1939. Это означает, что сообщения идентифицируются на основе 18-битных PGN, которые могут быть извлечены из полных 29-битных идентификаторов CAN.

Сигналы, также известные как параметры, будут "упакованы" в данные полезная нагрузка каждого сообщения. Обратите внимание, что в отличие от J1939, сигналы обозначаются не как подозрительные номера параметров (SPN), а как поля данных (DF) или просто параметры.

Мы не рассматриваем здесь структуру PGN, поскольку она уже рассмотрена во введении к SAE | 1939. Однако мы предоставим несколько практических замечаний по NMEA 2000 PGN и параметрам.





База данных NMEA 2000 PGN

Хотя NMEA 2000 изначально была основана на J1939, стандарт J1939 PGN (из J1939-71) недостаточно охватывают морские требования Стандарт NMEA.

Чтобы решить эту проблему, уровень приложений NMEA 2000 определяет ряд PGN уникальный для NMEA 2000, который описан в стандарте NMEA 2000, Приложение B. В приложении перечислен ряд PGN и полей данных, содержащихся в каждом из них. Каждое поле данных описывается с помощью "Словаря данных". Это можно сравнить с тем, как J1939-71 описывает каждое PGN и лежащие в их основе SPN.

Как видно из выдержки из PGN, приложение B в формате PDF содержит информацию для расшифровки таких параметров, как Wind Скорость. Однако извлечение этого вручную было бы это отнимает много времени, поскольку PDF содержит более 170 PGN и более 1500 параметров. Вместо этого мы рекомендуем использовать Файл DBC NMEA 2000, который включает PGN и структурированная информация о параметрах, готовая к использованию при декодировании необработанных данных NMEA 2000.

Данные о ветре.		PGN: 130306	
		hex: 1FD02	
Направление и скорость ветра. Истинный ветер может быть привязан к судну или к земле. Каждующийся ветер - это то, что ощущается			
Single Frame	Yes	Priority Default	2
Destination	Global	Query Support	Optional
		Default Update Rate	100 milliseconds
		Command Support	Optional
		ACK Requests	None
Field #	Field Name	Byte Field Size	Request Parameter
1	Sequence ID	1	Command Parameter
		Bit Field Size	Optional
DD056	Sequence ID		
An upward counting number that binds information transmitted in two or more PGNs from a single source address. Identical SID values within two or more different PGN transmissions identify these PGN transmissions as a single			
DF53	Integer, 8 bit unsigned	uint8	Range: 0 to 252
		Resolution	1 bit
		Unit-less number	
2	Wind Speed	2	Request Parameter
		Bit Field Size	Optional
DD044	Generic Speed		Command Parameter
DE14	Speed	uint16	Optional
		Range:	
		Resolution	

Относительно запатентованных PGN NMEA 2000

Стандарт NMEA 2000 позволяет производителям создавать свои собственные проприетарные PGN, однако при соблюдении трех условий:

1. Фирменная ПГН содержит конкретные сведения об устройстве (напр., уникальное калибровки)
 2. Данные используются в целях тестирования
 3. Подходящий PGN недоступен в текущей базе данных NMEA 2000 PGN

Как правило, производителям рекомендуется использовать стандартизированные PGN (или переходить на них, когда они станут доступны). Кроме того, если в устройстве используются проприетарные PGN, NMEA рекомендует производителям публиковать расшифровку Информация.

Примеры сигналов NMEA 2000 приведены в Приложении В

Приложение В NMEA 2000 содержит более 1500 параметров, то есть полей данных, закодированных в специфических для NMEA 2000 PGN. Ниже приведены некоторые примеры включенных сигналов, ориентированных на море.:

Предупреждения
Местоположение GNSS
Скорость судна
Курс
Угол поворота руля

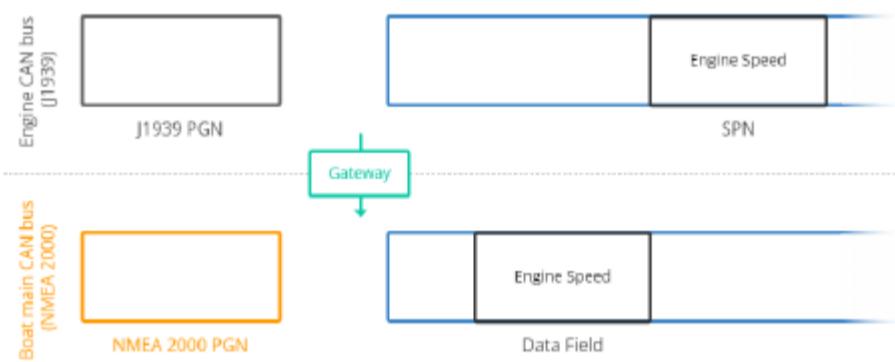
Скорость поворота
Рыскание, тангаж, крен
Частота вращения двигателя
Температура
Давление топлива
Расход топлива
Нагрузка на двигатель
Трансмиссия
Состояние аккумулятора
Глубина воды
Скорость ветра
Температура воздуха
... и многое другое

NMEA против J1939 PGNs [+ файлы DBC]

На морских судах J1939 часто используется "внутри двигателей" (например, подвесных двигателей) для передачи различных данных датчиков, связанных с двигателем, таких как частота вращения двигателя, температуры и т.д.

В таких случаях "шлюз" обычно преобразует подмножество информации J1939 PGN в NMEA 2000 PGN, таким образом позволяя, например, отображать на дисплеях информацию о двигателе.

Например, J1939 PGN 0xF004 (61444) содержит
Частота вращения двигателя - пока она
кодируется в N2K PGN 0x1F200 (127488).



Оригинальная коробка передач J1939 PGN

Таким образом, если вы записываете данные только из сети NMEA 2000, вы обычно получаете "чистый" NMEA 2000 PGN информация плюс "отфильтрованная" информация J1939 PGN (в кодировке NMEA 2000 PGN). Однако в некоторых случаях "оригинальные" PGN J1939 будут выводиться непосредственно в сеть NMEA 2000. Это возможно, поскольку протоколы "гармонизированный", то есть они используют одинаковую скорость передачи данных 250К, и ни один из идентификаторов CAN не перекрывается.

Короче говоря, чтобы максимально увеличить количество данных о судне, которые вы можете декодировать, вам следует рассмотреть следующие файлы DBC:

Данные NMEA 2000 (судно, датчики)



При подключении к сети NMEA 2000 (например, через разъем M12), как правило, выполняется запись широковещательной передачи NMEA 2000 данных. Чтобы расшифровать это, вы можете использовать протокол NMEA 2000 ДБН, которые позволит вам извлечь информацию о стандартизованных PGNs из сети. Сюда также может входить "отфильтрованная" информация о двигателе.

[Узнать больше](#)



Данные J1939 (механизмы)

В некоторых случаях сеть NMEA 2000 может также содержать оригинальные PGN J1939 из движка. В качестве альтернативы вы можете получить доступ к исходной информации J1939, подключив непосредственно к двигателю через отдельный диагностический разъем (обычно это 9-контактный разъем deutsch J1939). В любом случае вы можете использовать файл J1939 DBC для декодирования стандартизированного J1939 PGN.

[Узнать больше](#)



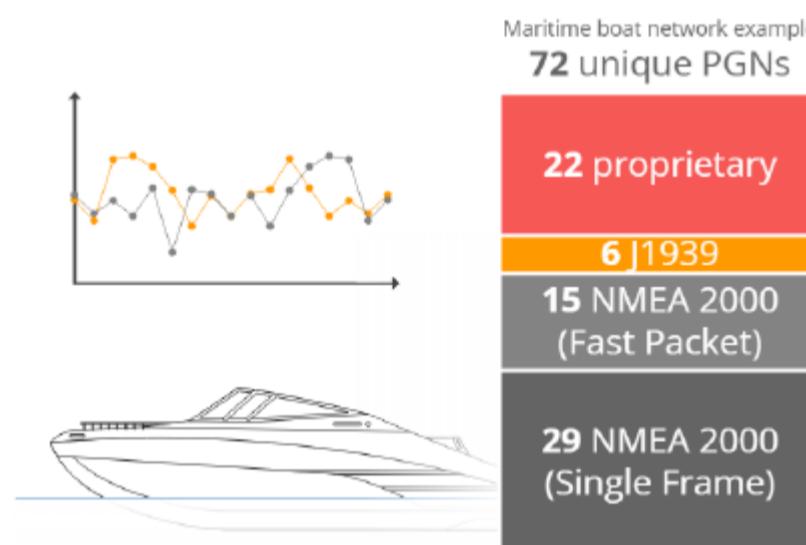
Данные OEM (собственные)

Почти всегда будут какие-то NMEA 2000 PGN, которые являются проприетарными. В принципе, они известны только производителю, но поскольку организация NMEA поощряет производителей ОБОРУДОВАНИЯ публиковать свои правила кодирования, вполне вероятно, что вы найдете соответствующую информацию в технических руководствах или по запросу. Если это так, вы можете добавить их в существующий файл DBC, изменив его с помощью DBC редактор (подробнее смотрите наше введение к DBC).

Пример: Расшифровка данных с судна

Чтобы продемонстрировать разбивку PGN, которую вы часто видите в данные морских CAN, рассмотрим файл журнала с быстроходного судна. В этом случае для записи данных использовался CANedge1 из магистрали NMEA 2000, которая также включает в себя (на по крайней мере, некоторые из) исходных данных J1939 с подвесного двигателя двигателя. Файл журнала содержит 161 уникальный идентификатор CAN, соответствующий 72 уникальным PGN.

Очевидно, что большинство PGN декодируются NMEA 2000 DBC (в виде отдельных кадров или быстрых пакетов). Далее, DBC J1939 способен декодировать некоторые из PGN J1939, в то время как остальные зависят от производителя. проприетарные PGN могут исходить от движка J1939 сеть или часть сетевого оборудования NMEA 2000.



Примечание относительно PGN J1939

Обратите внимание, что подключение CANedge1 было произведено по сети NMEA 2000, а не напрямую к J1939 сеть (т.е. диагностический разъем двигателя). В результате в данном случае неизвестно, будут ли доступны дополнительные PGN J1939 через прямое подключение к двигателю. Кроме того, в большинстве сетей NMEA 2000 PGN engine J1939 не передаются напрямую в сеть NMEA 2000, следовательно, вам обычно нужно записывать их отдельно. Это часто делается с помощью CANedge, подключающего 1-й порт к сети NMEA 2000, а 2-й порт к двигателю J1939 диагностический разъем.

Регистрация данных NMEA 2000 maritime

Для регистрации данных по шине CAN maritime из сети NMEA 2000 или движка J1939 мы рекомендуем ознакомиться с нашей программой maritime введение по телематике. Ниже мы кратко перечислим наиболее важные шаги для рассмотрения:

1 Как выбрать регистратор CAN

Если вам нужно записать данные N2K / J1939 на SD-карту, мы рекомендуем CANedge1. Устройство можно оставить в емкости для записи данных на недели или месяцы. Чтобы собрать данные, просто извлеките SD-карту. В качестве альтернативы, CANedge2 (Wi-Fi) и CANedge3 (3G / 4G) позволяют автоматически загружать данные на ваш собственный сервер. Устройство записывает данные на SD-карту и загружает их, когда судно находится вблизи берега с возможностью подключения.



2 Как подключиться к сети N2K

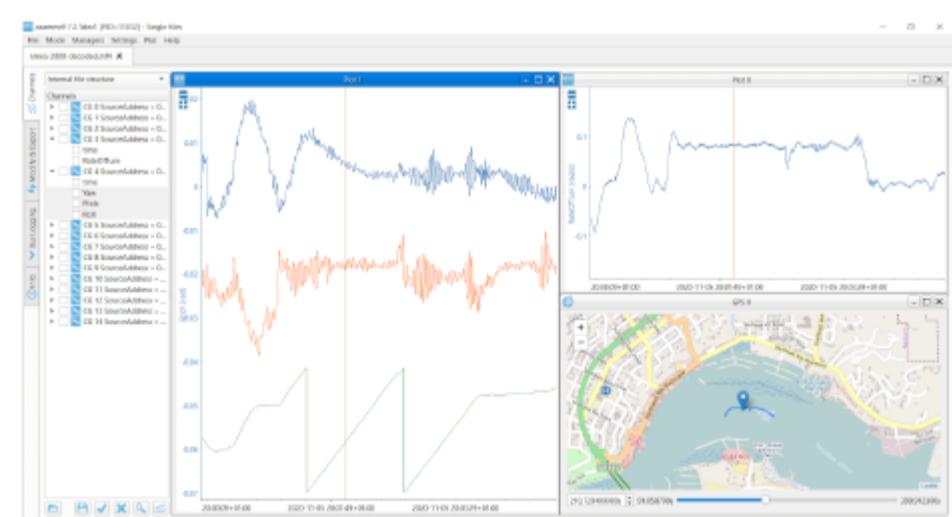
Далее вам нужно будет определить, как подключиться к CAN Автобус. Самый простой способ подключения к NMEA 2000 сеть заключается в использовании адаптера DB9-M12 для подключения через T-образный разъем. Это дает вам доступ ко всем необработанным данным NMEA 2000 данных из этой сети. При желании вы можете использовать 2-й канал CAN CANedge для подключения к другой Сети NMEA 2000 (если таковая существует) или напрямую к J1939 сеть двигателя. В последнем случае вы можете использовать один из диагностических разъемов двигателя, перечисленные ранее.



3 Как обрабатывать данные CAN.

Для обработки исходных данных NMEA 2000 и / или J1939 vessel вам понадобится программное обеспечение CAN. В нем вы загрузите файлы журнала и связанные файлы DBC для преобразования необработанные данные в физические значения.

Программные инструменты CANedge позволяют использовать различные формы анализа . Например, вы можете использовать MF4 конвертеры для преобразования необработанных файлов журналов в популярные форматы файлов (например, ASC, TRC, CSV) для использования в таких инструментах, как Vector CANalyzer, Warwick X-Analyser 3 или PEAK PCAN-Explorer. Или вы можете использовать графический интерфейс asammmdf для DBC декодирует данные и создает визуальные графики, включая GPS-карты маршрутов вашего судна. Кроме того, вы можете развернуть автоматическую обработку данных с помощью Python API, если вам нужно выполнить статистический анализ, автоматизированную отчетность или интеграцию с базой данных. API также может использоваться для интеграции данных с [Панели мониторинга Grafana](#).





Объяснение ISOBUS (ISO 11783) - простое введение

Нужен простой, практическое введение в ISOBUS (ISO 11783)?

В этом руководстве мы представляем протокол ISOBUS, используемый в сельскохозяйственных транспортных средствах (таких как тракторы) и орудиях труда (таких как опрыскиватели, сеялки). В частности, мы рассмотрим стандарты ISO 11783, историю, ссылку на J1939 и соединители.

Чтобы это было практично, мы также объясняем, как записывать и декодировать данные ISOBUS - с практическими примерами использования и [примерами игровой площадки dashboard](#).

Узнайте больше ниже!

Что такое ISOBUS?

ISOBUS (ISO 11783) - стандартизированный протокол связи

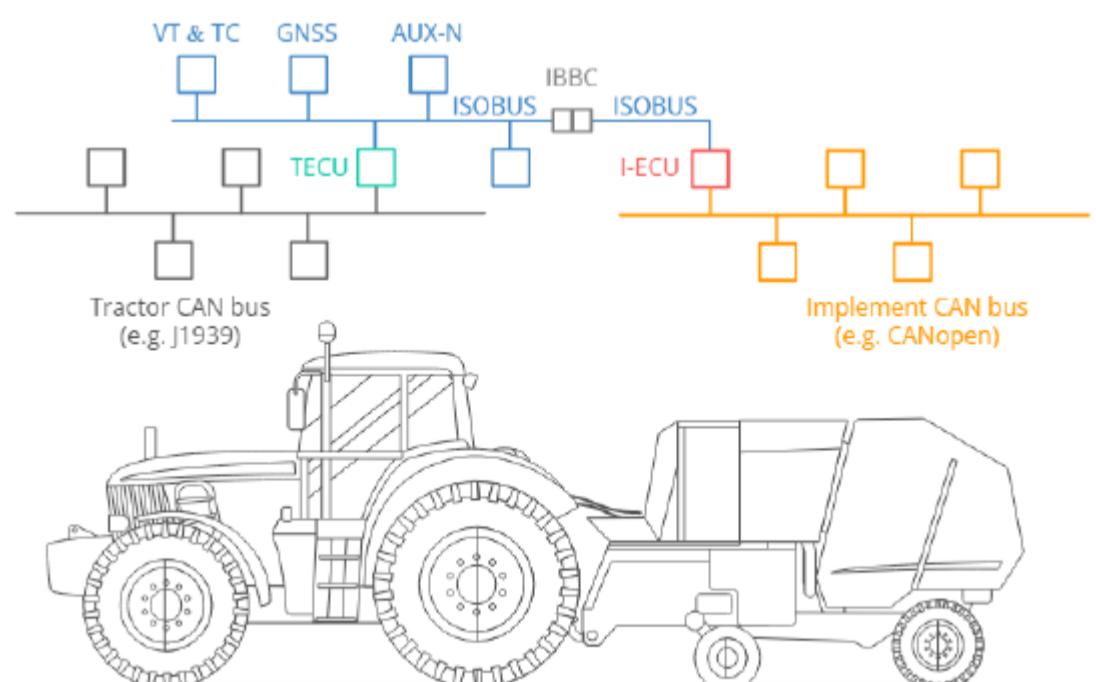
Используемый в сельскохозяйственной и лесохозяйственной технике.

Основная цель - включить функцию plug & play

интеграция между транспортными средствами (например, тракторами) и орудиями труда (например, опрыскивателями) разных производителей.

Протокол основан на локальной сети контроллера (Шина CAN). В частности, он был получен из Протокола SAE | 1939, используемого в большинстве транспортных средств большой грузоподъемности (включая большинство тракторов).

Сегодня ISOBUS применяется в большинстве современных тракторах и орудиях труда (таких как пресс-подборщики, опрыскиватели, удобрения, сеялки) и позволяет стандартизировать связь между ними.



Например, трактор может использовать J1939 внутри компании. Блок управления трактором (TECU) затем служит шлюзом к ISOBUS трактора сеть. Аналогичным образом, опрыскиватель может использовать, например, J1939 / CANopen внутри, в то время как шлюз ISOBUS облегчает связь с трактором. Предполагая, что ISOBUS должным образом реализован как в тракторе, так и в орудии, интеграция проходит гладко.

Благодаря сертифицированным решениям ISOBUS конечные пользователи избегают загромождения кабин терминалами, предназначенными для конкретного оборудования, и вместо этого управляют любым оборудованием с единого универсального терминала.

История ISOBUS и AEF

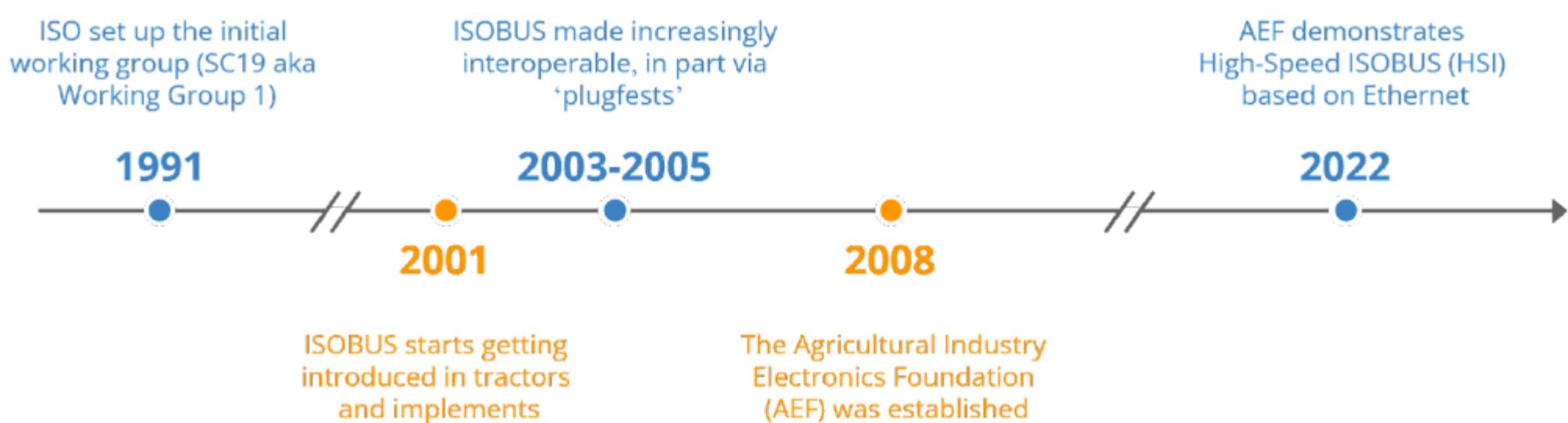
1991: ISO учредила первоначальную рабочую группу (SC19, она же Рабочая группа 1)

2001: ISOBUS начали внедрять в тракторы и орудия труда

2003-2005: ISOBUS становится все более совместимым, частично благодаря "plugfests"

2008: Основан Фонд электроники для сельскохозяйственной промышленности (AEF)

2022: AEF демонстрирует высокоскоростную ISOBUS (HSI) на базе Ethernet



Сертификация AEF и ISOBUS

Когда решения ISOBUS впервые начали внедряться в начале девяностых годов большинство из них были решениями на ранней стадии. С 2001 года ISOBUS пережила "прорыв" в области внедрения с десятками тысяч пользователей новых тракторов / навесного оборудования ISOBUS, продаваемых по всему миру.

Однако на практике стандарт был сложным и интерпретировался производителями по-разному, что вызывало проблемы несовместимости в поле (и общая потеря доверия к ISOBUS обещание). В ответ на это в 2008 году была основана AEF.



СЕЛЬСКОХОЗЯЙСТВЕННАЯ ПРОМЫШЛЕННОСТЬ
ФОНД ЭЛЕКТРОНИКИ

Члены AEF

Первоначально в AEF входили семь производителей сельскохозяйственной техники (John Deere, CNH, Claas, AGCO, Kverneland Group, Grimme, Pottinger) и две ассоциации (Ассоциация производителей оборудования, AEM, и Немецкая Инженерная федерация, VDMA). Сегодня FederUnacoma является третьим основным членом ассоциации, в то время как в настоящее время в нее входят 8 основных членов-производителей, включая KUHN. В общей сложности более 200 компаний / учреждений в настоящее время являются членами AEF.

АЭФ имеет набор основных целей



Определение руководящих принципов для реализации совместимого с ISOBUS
Координация улучшений с ISOBUS ВКЛ. сертификационные испытания
Координировать сотрудничество в сфере сельского хозяйства
Организовать поддержку сертификации, обучения и консалтинга

Кроме того, АЭФ разработала ряд инструментов и продуктов:

Тест на соответствие требованиям АЭФ: используется аккредитованными АЭФ испытательными лабораториями для сертификации компонентов ISOBUS
База данных АЭФ: используется дилерами / пользователями по всему миру для оценки совместимость и соответствие компонентов ISOBUS

Практическая значимость

Тестирование соответствия и связанная с ним база данных жизненно важны для обеспечения последовательной стандартизированной реализации ISOBUS - позволяет конечным пользователям осуществлять поиск компонентов различных брендов и производителей для обеспечения совместимости перед Покупка. Это гарантирует, что исходный посыл 'штепсельная вилка & игра' кросс-бренд опыт работы с ISOBUS средствами и реализует могут быть реализованы на практике.

Геометрия модели и стандарты ИОО

Далее мы кратко рассмотрим 7-слойную модель OSI для ISOBUS включая ссылку на 14 частей стандарта ISO 11783.

Обратите внимание, что мы также ссылаемся на шину CAN (ISO 11898) для физического уровня и уровней канала передачи данных, чтобы отразить роль, которую CAN играет в качестве основы для ISOBUS.

Несмотря на то, что SAE J1939 не показан на рисунке, он также играет роль эталона для некоторых слоев / стандартных деталей - например, J1939-71 предоставляет определение для большинства сообщений о трансмиссии, как описано в стандарте ISO 11783-8.

7-слойная модель OSI | ISOBUS (ISO 11783)

Application	ISO 11783-7/8/9/10/12/13/14
Presentation	ISO 11783-6/7/11
Session	ISO 11783-6/7/10/14
Transport	ISO 11783-3/5
Network	ISO 11783-4/5
Data link	ISO 11783-3 (ISO 11898-1)
Physical	ISO 11783-2 (ISO 11898-2)

Ниже мы кратко перечислим 14 частей стандарта ISO 11783.

ISO 11783-1: Общий стандарт мобильной передачи данных.

Это служит обзором всего набора стандартов ISO 11783, обеспечивающего основу для стандартизированного последовательного соединения сеть связи между лесохозяйственными / сельскохозяйственными тракторами и орудиями труда. Он описывает уровни модели OSI и ссылается на другие стандарты.

ISO 11783-2: Физический уровень

Этот стандарт определяет физический уровень сети. Поскольку ISOBUS основан на CAN, этот стандарт тесно связан с ISO 11898-2, описывающим физический уровень шины CAN. Кроме того, ISO 11783-2 описывает разъемы, специфичные для ISOBUS. например, разъем ISOBUS Breakaway Connector (IBBC).

ISO 11783-3: уровень канала передачи данных.

Этот стандарт описывает форматы сообщений ISOBUS. В частности, он предписывает использовать расширенные идентификаторы CAN и номера групп параметров в стиле J1939 (PGN). Стандарт также определяет транспортный протокол для передачи многокадровых сообщений ISOBUS, то есть данных, которые превышают 8 байт классических кадров CAN.

ISO 11783-4: Сетевой уровень.

Здесь описано использование "сетевых соединительных устройств" для соединения двух отдельных сетей с разными архитектурами. Этот шлюз обеспечивает как электрическую изоляцию, так и изоляцию сообщений в каждой сети.

ISO 11783-5: Сетевое управление

Это определяет процесс определения адресов источников (SA) и разрешения конфликтов адресов. Адреса источников могут быть предварительно установлены или динамически запрошены каждым контроллером во время включения питания.

ISO 11783-6: Виртуальный терминал

В настоящем стандарте виртуальный терминал описывается как блок управления, который предоставляет трактористу интерфейс для управления трактором и/или орудиями труда. Контроль должен осуществляться с помощью стандартизированного стандарта ISO Сообщения 11783. VT является основной функцией ISOBUS.

ISO 11783-7: Реализация сообщений на прикладном уровне.

Этот стандарт описывает стандартизованные сообщения приложения для реализации ISOBUS. Это также служит основой для большинства сообщений в нашем файле ISOBUS DBC. В некоторой степени это можно сравнить с J1939-71, который служит основой для большей части информации о декодировании PGN и SPN J1939. В случае ISOBUS описанные сообщения включают, например, скорость и направление движения колеса / грунта, состояние переключателя клавиш, вспомогательные клапаны, огни и многое другое.

ISO 11783-8: Сообщения о трансмиссии.

Здесь описываются сообщения о трансмиссии, используемые при реализации сети тракторов. По сути, это ссылка на SAE Стандарт J1939-71, в котором указано, что любые перекрывающиеся сообщения должны быть реализованы в соответствии с J1939-71.

ISO 11783-9: Блок управления трактором.

Когда орудие подсоединенено к трактору, крайне важно обеспечить постоянную взаимную связь. В данном случае трактор Блок управления должен позволять агрегату запрашивать информацию от блоков управления в сети трактора в соответствии с ISO 11783-7, а также отвечать на команды агрегата. Здесь также описаны подробности работы в безопасном режиме.

ISO 11783-10: Контроллер задач и обмен данными информационной системы управления

В сети оборудования может быть доступен "компьютер управления фермой". Это позволяет оператору (обычно фермер) для определения задач/действий для выполнения орудия. "Диспетчер задач" получает эти задачи через интерфейс с компьютера. Задачи планируются контроллером, который обеспечивает выполнение в агрегате с помощью

функций управления. Контроллер также получает информацию о состоянии от блоков управления агрегатом, которая передается обратно на компьютер управления фермой. Стандарт определяет работу контроллера задач, включая стандартизированные сообщения, используемые для функций управления.

ISO 11783-11: Словарь мобильных элементов данных.

Это тесно связано с 11783-10, предоставляя словарь данных для определения элементов данных, используемых в контроллере задач Сообщения.

ISO 11783-12: Диагностические услуги

Это определяет сообщения, используемые для диагностики для выявления неисправностей и проблем в сети. Это можно сравнить с J1939-73

ISO 11783-13: Файловый сервер.

Это определяет блок управления, который обеспечивает хранение данных в сети (например, для мультимедиа), а также набор команд для использования при чтении / записи данных на файловый сервер.

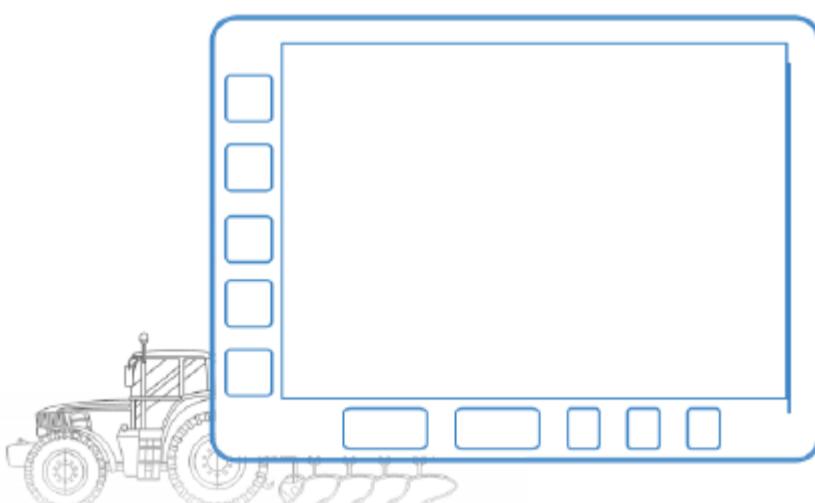
ISO 11783-14: Контроль последовательности действий

Этот более свежий стандарт (впервые выпущен в 2013 году) подробно описывает, как выполнять контроль последовательности действий в сети ISO 11783. Здесь оператор может записывать последовательности функций управления для последующего воспроизведения по требованию.

Функциональные возможности ISOBUS

AEF называет ряд "продуктов" на основе стандарта ISO 11783 функциональными возможностями ISOBUS. Фактически, они могут продаваться конечным пользователям пользователям ISOBUS в виде отдельных модулей. В этом разделе мы кратко описываем каждую из функциональных возможностей AEF ISOBUS:

#1 Виртуальный терминал (VT) / Универсальный Терминал (UT)



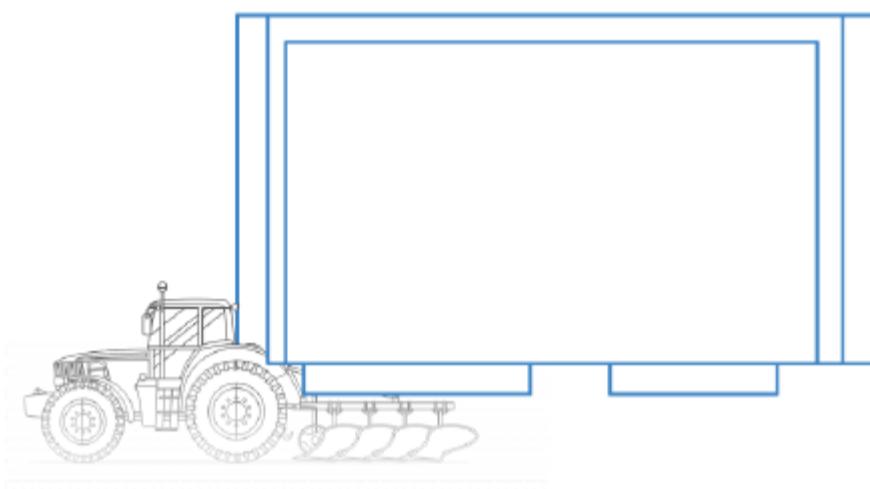
Это относится к виртуальному терминалу, указанному в ISO 11783-6. VT иногда называют Универсальным терминалом (UT) из-за возможностей, которые он предоставляет: Позволяет конечному пользователю управлять различными реализует (даже под разными брендами / производителями) через единый терминал. Мы будем называть его VT повсюду.

VT является основной частью того, что делает ISOBUS визуально привлекательным: вместо загроможденной кабины трактора содержит множество приспособлений для конкретного оборудования дисплеи, теперь все можно обрабатывать с помощью единого дисплея для всех совместимых с ISOBUS устройств.

На практике VT инициализируется с помощью последовательности Сообщения CAN, отправляемые между VT и реализация. Здесь VT информирует реализацию о

своих спецификациях, позволяя реализации отправлять свои пользовательский интерфейс к VT для графической визуализации. После инициализации оператор может выполнять различные выполнять задачи управления с помощью VT.

Блок управления трактором № 2 (TECU)

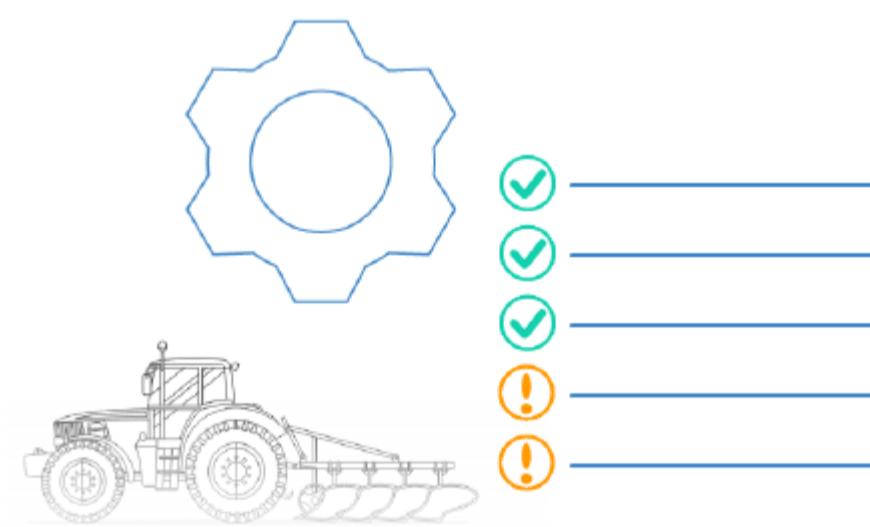


Блок управления трактором (TECU) служит шлюзом между шиной CAN трактора (обычно J1939) и ISOBUS. TECU предоставляет выборочную информацию для орудия например, скорость, обороты в минуту и т.д. Из сети трактора через ISOBUS.

В более поздних версиях тракторных ЭБУ, проведения реализации может взять на себя полный контроль над Трактор - ВКЛ. рулевое управление, скорость, торможение, передача и т. д. Это позволяет создавать все более автономные решения, в которых трактор и орудие работают без сбоев как единая система.

Обратите также внимание, что орудие часто имеет схожую Блок управления, иногда называемый блоком управления реализацией (I-ECU). Он служит шлюзом между внутренней шиной CAN устройства и ISOBUS.

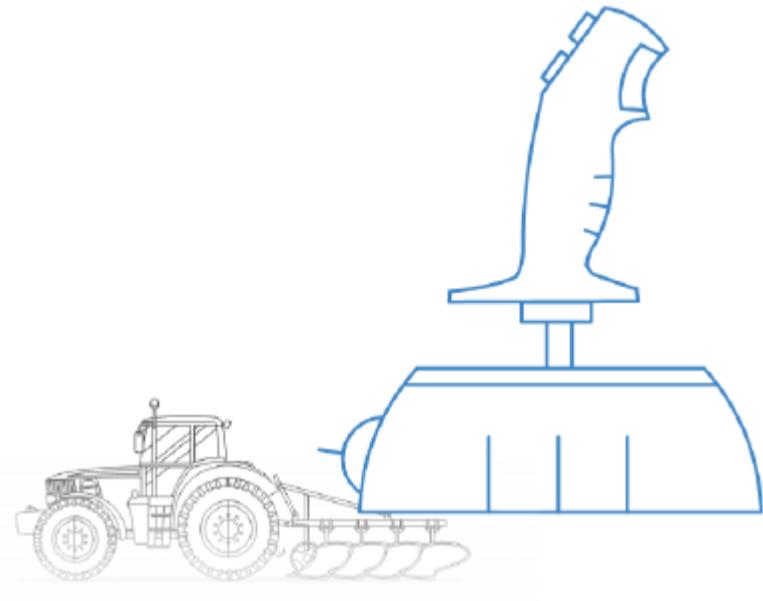
3 Контроллер задач (TC)



Контроллер задач - это часть программного обеспечения, которая записывает и предоставляет информацию об операциях и помогает в планировании. Он принимает входные данные от оператора через интерфейс и служит для планирования задач для устройства, выполняя их с помощью функций управления . В частности, это обеспечивает автоматизацию и таким образом, более детальный контроль над орудием - ключ, например, к точному земледелию.

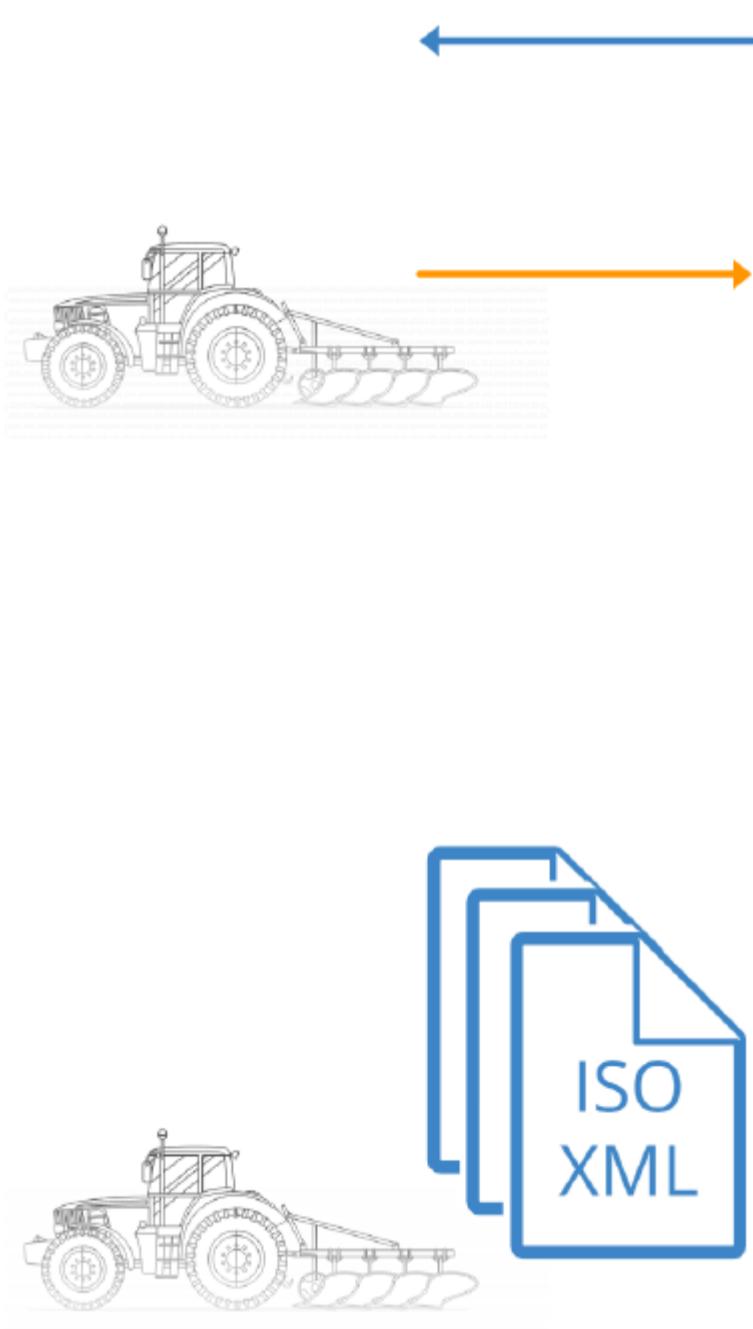
Контроллер задач состоит из трех вспомогательных протоколов.:

1. Базовый (TC-BAS): используется для отслеживания данных итоговые данные (например, сколько продукта было применено) с информацией, предоставленной реализовать. Это упрощает, например, извлечение документирование выполненной работы через интерфейс оператора
2. Секция (TC-SEC): Это позволяет автоматизировать переключение секций, например, в распылителе на основе например, положения GPS, что позволяет более точное использование функциональных возможностей реализации
3. Гео (TC-GEO): используется для автоматизации задач на основе GPS, таких как выполнение управление посевом на основе гео. Например, это полезно, если определенные области в поле требуют индивидуального управления



4 Вспомогательное управление (AUX)

Эта функциональность ISOBUS часто использует джойстик или аналогичный элемент управления, что обеспечивает больший детальный контроль над функциональностью реализации. Обратите внимание, что существуют как новая, так и старая версии, которые не совместимы между собой (AUX-N / AUX-O).



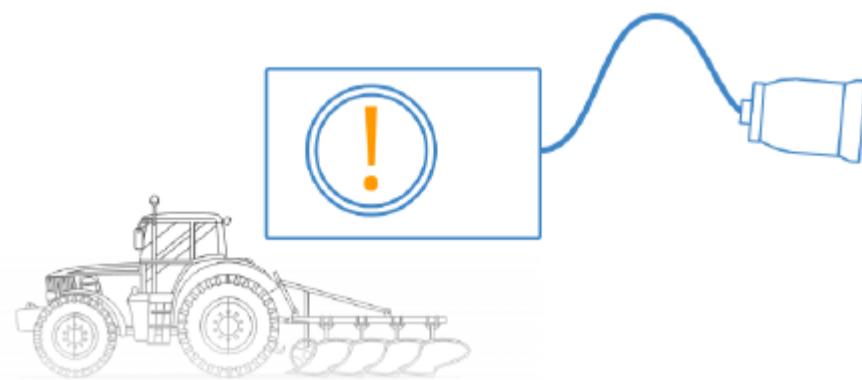
5 Управление навесным оборудованием для трактора (TIM)

Как упоминалось в разделе TECU, существует растущая потребность в двунаправленном управлении между трактором и орудием, что, в частности, означает, что в конечном итоге орудие должно быть способно управлять трактором в некоторых случаях использования. Это повысит производительность и снизит утомляемость оператора.

6 Регистрация значений устройства (LOG)

Это предназначено для регистрации данных из трактора / орудия, независимо от выполняемой задачи. В какой-то степени это можно сравнить с модернизацией регистратора данных CAN для записи всех сообщений передаются по шинам CAN. Однако в случае с журналом цель состоит в том, чтобы обеспечить передачу данных экспортируется в формате ISO XML (аналогично тому, что есть поддерживается для данных контроллера задач).

7 Кнопка быстрого доступа ISOBUS (ISB)



Это позволяет деактивировать выборочную реализацию функции. Это может быть актуально в случаях использования, когда управление несколькими орудиями осуществляется с одного виртуального терминала трактора. Здесь используется одно орудие может быть на переднем плане, в то время как пользователю необходимо быстро отключить функциональность в другом реализовать.

ISOBUS против SAE J1939

Распространенный вопрос заключается в том, как ISO 11783 (ISOBUS) соотносится с протоколом SAE J1939.

Короче говоря, ISOBUS изначально был получен из J1939, и основная цель стандарта ISO 11783 - оставаться совместимым с J1939 за счет тесного согласования с SAE. В некотором смысле, можно сказать, что ISOBUS сегодня гармонизирован с J1939. Это полезно, поскольку многие тракторные сети будут основаны на SAE J1939, что означает, что блоки управления тракторами (TECU) часто должны служить шлюзами между ISOBUS и J1939.

J1939 против ISOBUS (ISO 11783)

Тема	J1939	ISOBUS
Прикладной уровень	PGN и SPN	Виртуальный терминал, контроллер задач, файловый сервер...
Запрос адреса	Необязательно	Обязательно
Быстрый пакет TP	N/A	Используется для GNSS (NMEA 2000)
Расширенный TP	N/A	Используется, например, для VT, файлового сервера
Физический уровень / соединитель	См. J1939-1X	Смотрите ISO 11783-2 (другой)
Проверка на соответствие	Необязательно	На практике является обязательной с помощью AEF

Кроме того, различные стандарты J1939 служат прямыми ссылками на стандарты ISOBUS. Например, J1939-71 определяет сообщения о трансмиссии и служит эталоном для стандарта ISO 11783-8. И ISOBUS, и J1939 также полагаются на шину CAN (CAN 2.0B) в качестве нижнего уровня связи, и обе используют концепцию 18-разрядных PGN для идентификации сообщений.

Однако J1939 в основном используется в "закрытых системах", и поэтому ISOBUS был расширен за пределы J1939, чтобы удовлетворить потребность в устройствах plug & play. Поэтому важно отметить, что ISOBUS не равен J1939. Мы выделяем ряд ключевых различий в таблице. Подробностисмотрите ниже.:

ISO 11783 против J1939 - подробнее

В частности, обратите внимание, что уровень приложений ISOBUS включает в себя ряд дополнительных компонентов, таких как виртуальный Терминал и контроллер задач.

ISOBUS также позволяет использовать альтернативные методы транспортного протокола, такие как быстрые пакеты, например, для передачи данных GNSS (также известные из NMEA 2000) и Расширенный транспортный протокол (например, для отправки данных на VT или файловый сервер). ISOBUS также вводит ряд идентификаторов словаря данных, которые аналогичны SPN, хотя обычно они передаются с использованием 1 SPN на сообщение, а не нескольких. Из-за необходимости взаимодействия реализации, запрос динамического адреса является обязательным в ISOBUS, тогда как в J1939 это часто пропускается (из-за закрытой системной логики). Кроме того, ISOBUS представляет ряд альтернативные физические разъемы по сравнению с J1939, о чем мы вскоре расскажем подробнее.

Разъемы ISOBUS

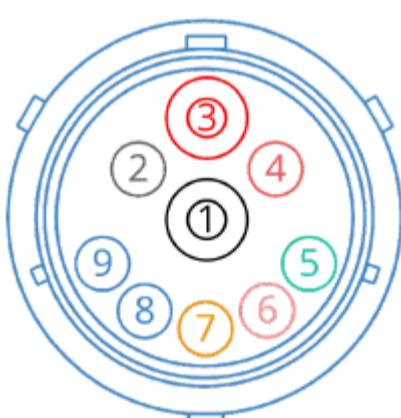
Стандарт ISO 11783-2 описывает ряд физических соединителей, используемых в тракторах и орудиях труда.



- 1 TBC power
- 2 CAN H
- 3 TBC return path
- 4 CAN L

Удлинительный разъем шины № 1

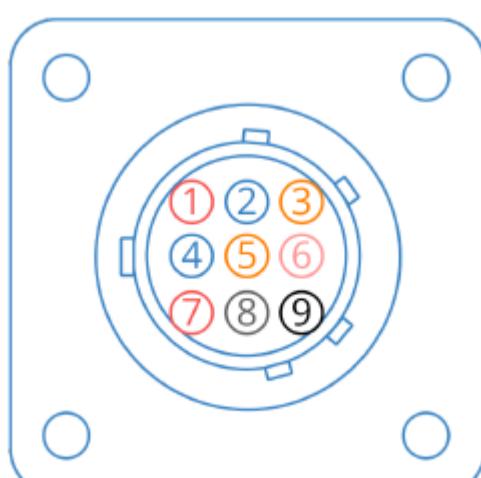
Этот разъем позволяет расширить ISOBUS сигнальные линии от орудия внутри трактора. Это, например, используется для добавления дополнительных устройств, таких как Виртуальные терминалы. Соединитель будет расположен в кабине трактора с правой стороны от сиденья оператора .



- 1...Заземление (шасси)
- Заземление 2 ЭБУ
- Питание 3
- Питание 4 ЭБУ
- Подключение/снятие 5 ТВС
- Мощность 6 ТБИТ/с
- Обратный путь 7 ТБИТ/с
- 8... МОЖЕТ ли ч
- 6...МОГУЛИ я

#2 Устройство для отключения шины Соединитель

IBBC позволяет подключать орудия труда к трактору. Соединитель расположен в задней части трактора. В некоторых случаях также может быть установлен дополнительный IBBC спереди трактора для использования с орудиями, установленными спереди. IBBC должен иметься заглушка для защиты от пыли и непогоды когда агрегат не подключен.



- 1...Мощность блока управления
- 2 БАНКИ по 1 л (вход)
- 3 БАНКИ по 2 л (выход)
- 4 МОЖЕТ 1 час (вход)
- 5 МОЖЕТ 2 часа (выход)
- 6...Питание ТВС
- 7 Питание ЭБУ
- 8 заземление ТВС
- 9...Заземление ЭБУ

Разъем № 3 для подключения в кабине (LBS)

В кабине трактора может быть установлен дополнительный разъем для подключения в кабине разъем для установления соединения с сетью ISO 11783. Иногда его называют LBS.



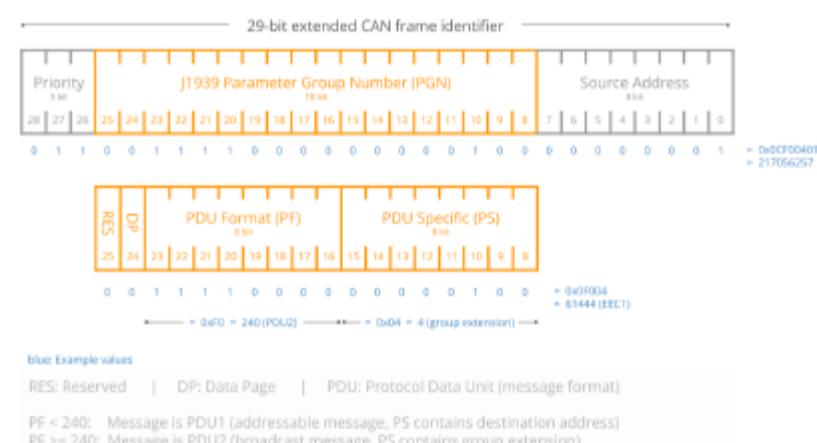
Диагностический разъем № 4

Диагностический разъем расположен в кабине трактора в легкодоступном месте. О чем мы расскажем позже, диагностический разъем может быть хорошим вариантом для подключения внешних устройств регистрации данных устройств для диагностики - в частности, если вам нужно доступ к сетям J1939 и ISOBUS в параллельный.

ISOBUS PGN и SPN [+ DBC]

В стандарте ISO 11783 используется концепция номеров групп параметров (PGN) и номера подозрительных параметров (SPN) из SAE J1939 протокол. Это означает, что сообщения идентифицируются на основе 18-битного PGN, которые могут быть извлечены из полных 29-битных идентификаторов CAN. В свою очередь, сигналы или параметры называются SPN и будут "упакован" в полезную нагрузку данных каждого сообщения.

Мы не будем подробно описывать структуру PGN, поскольку она уже есть рассмотрена во вступлении к SAE J1939. Однако мы предоставим несколько практические комментарии по ISOBUS PGN и SPN.



ISOBUS Data Dictionary - Home

ISOBUS 11783 Online Data Base

VDMA

Manufacturer Code Device Classification Addresses PGN SPN Process Data DBI Functionality Options

Home > SPN > 0x0099 General based machine speed

SPN 0x0099
SP Name General based machine speed
SP Description Actual ground speed of a machine, measured by a sensor such as that is not susceptible to wheel slip (e.g. radar, GPS, UDMA, or stationary object tracking).
SP Notes None
SP Attachment None
SP Historical Remarks None
SP Historical Attachment None
SPn 2015/2016 2016
SP Length 6 bits
Scaling 0.001 m/s per bit
Offset 0
Data Scale 0.001 m/s

База данных ISOBUS PGN и SPN

Стандарт ISO 11783-7 определяет PGN и SPN сведения, уникальные для ISOBUS. Сведения о декодировании общедоступны в справочнике данных ISOBUS VDMA . Сигналы включают:

Угол наклона / крена / рыскания переднего / заднего сцепного устройства

Выбранная машиной скорость

Информация о ВОМ - частота вращения заднего вала

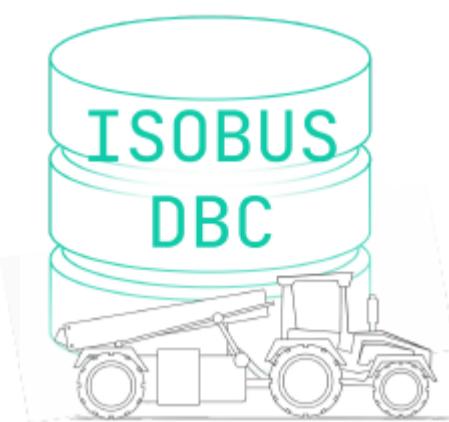
Частота вращения двигателя, в зависимости от колес

Фары трактора

Общедоступная база данных включает правила декодирования для ~ 800 сигналов, из которых ~ 650 могут быть преобразованы напрямую в формат, полезный, например, в телематике и регистрации данных. Как как объясняется ниже, мы предоставляем версию этих сообщений DBC ~ 650 правил декодирования сообщений ISO 11783. Обратите внимание, что сеть ISOBUS также может содержать OEM конфиденциальные данные, которые невозможно расшифровать с помощью общедоступной информации.

Сравнение ISOBUS с J1939 и NMEA 2000 [+ файлы DBC]

При записи данных с трактора/орудия могут иметь значение несколько протоколов (и, следовательно, файлы DBC).



Данные ISOBUS (орудие/трактор)

Естественно, сеть ISOBUS будет содержать данные, относящиеся к связь между трактором и орудием. Эта информация часто будет декодироваться с использованием правил декодирования ISO 11783-7 упомянутых выше. Для обеспечения быстрого декодирования стандартной ISOBUS данные, вы можете использовать наш файл ISOBUS DBC.

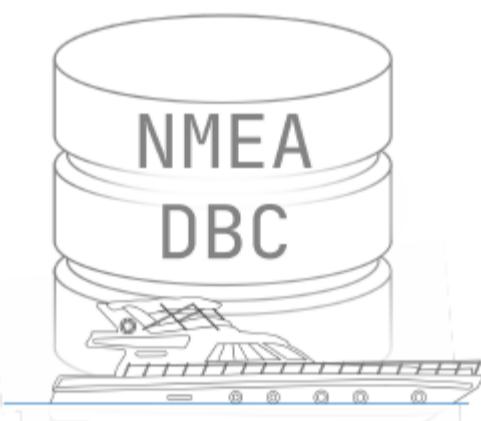
[Узнать больше](#)



Данные J1939 (трактор)

Если вы записываете данные непосредственно с шины CAN трактора, это, скорее всего, скорее всего, это данные J1939. Аналогично, если вы регистрируете данные из сети ISOBUS , некоторые данные из блока управления трактором обычно будут J1939. В любом случае, данные J1939 требуют использования декодирования правила из Цифрового приложения J1939. Для этой цели вы можете воспользоваться нашим [Файл J1939 DBC](#).

[Узнать больше](#)



Данные NMEA 2000 (GNSS)

Часто сеть ISOBUS содержит GNSS в кодировке NMEA 2000 Информация. Данные NMEA 2000 могут передаваться в виде комбинации одиночных кадров и кадров TP с быстрым пакетом NMEA 2000. Для декодирования данные, вы можете использовать наш файл NMEA 2000 DBC.

[Узнать больше](#)



Собственные данные производителя (трактор / орудие)

Конечно, почти всегда будут какие-то сообщения, которые проприетарные независимо от протокола. Они известны только OEM (производителю оригинального оборудования) и не могут быть декодированы с помощью стандартных файлов DBC. Иногда эти сообщения соответствуют одному из вышеуказанных стандартов, хотя в других случаях они могут использовать полностью другой стандарт (например, CANopen).

Пример: Расшифровка данных из сети ISOBUS

Для иллюстрации разделения, которое вы часто видите в сельском хозяйстве, можно данных шины, рассмотрим файл журнала трактора Fendt с орудием другого производителя. В данном случае CANedge2 использовался для записи данных из сети ISOBUS. Благодаря этому было записано 337 уникальных идентификаторов CAN, что соответствует 38 уникальным PGN.

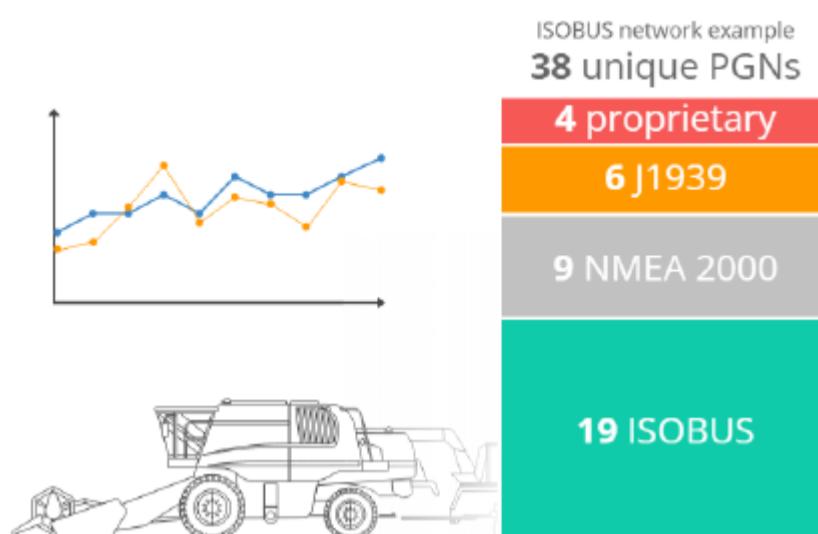
PGN могут быть разделены по протоколу следующим образом:

ISOBUS DBC: 19 PGN (~ 50%)

NMEA 2000 DBC: 9 PGN (~24%)

J1939 DBC: 6 PGN (~ 16%)

Запатентованный: 4 PGN (~ 10%)



Чтобы получить образец данных ISOBUS, воспользуйтесь нашим пакетом данных J1939.

Обратите внимание на PGN J1939

Обратите внимание, что подключение CANedge2 было произведено по сети ISOBUS, а не напрямую к сети J1939. В результате существует очень мало декодируемых PGN J1939, поскольку они отражают только PGN J1939, повторяемые Блок управления трактором (TECU) на ISOBUS. Если бы CANedge2 был подключен к сети J1939 напрямую, было бы общее количество идентификаторов CAN было бы значительно больше, включая гораздо больше PGN J1939.

Данные о лесозаготовительном тракторе / орудии.

В этом разделе мы расскажем, как можно записывать и обрабатывать данные с тракторов / орудий.

1 Как выбрать регистратор данных шины CAN

Если вам необходимо записать свои данные в автономном режиме, вы можете использовать автономный [CANedge1 для входа в систему 2 × CAN](#) на SD-карту. Устройство можно оставить в трактор для записи данных за недели или месяцы. Чтобы собрать данные, просто извлеките SD-карту.



В качестве альтернативы вы можете использовать CANedge2 (Wi-Fi) или CANedge3 (3G / 4G) для автоматической загрузки данных на ваш собственный сервер (автономный или облачный). Это позволяет, например, осуществлять крупномасштабный сбор данных, автоматизированную отчетность или профилактическое обслуживание.

2 Как подключиться к шине CAN

После выбора регистратора данных шины CAN вам необходимо определить, как взаимодействовать с сетями CAN. Обычно мы рекомендуем по возможности использовать диагностический разъем. Как указывалось ранее, это обеспечивает доступ как к J1939 сети трактора, так и к сети ISOBUS.

Вы можете использовать наш кабель-адаптер J1939-DB9/DB9 (H + J), который идеально соответствует распиновке диагностического разъема и позволяет вам для регистрации сетей J1939 и ISOBUS отдельно в один и тот же файл журнала на CANedge.



Дополнительные сведения

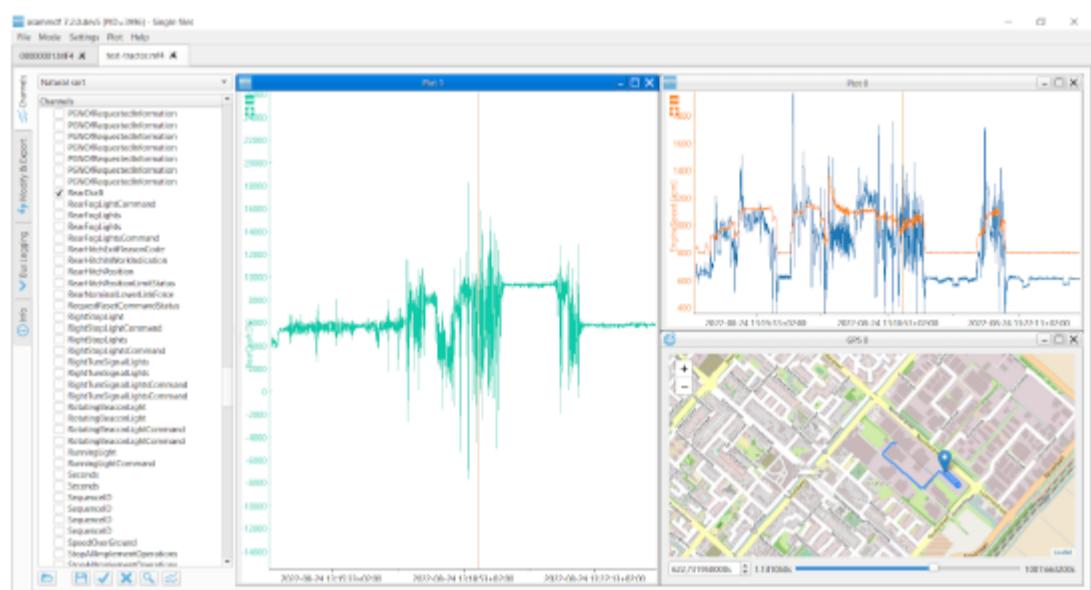
В качестве альтернативы вы также можете использовать наш обычный адаптер J1939-DB9, если вам нужен только доступ к сети J1939. Часто многие часть сообщений будет повторяться по сетям, поэтому для многих вариантов использования этого может быть достаточно.

Обратите внимание, что если вы хотите подключиться напрямую к диагностическому разъему агрегата (а не через кабину трактора), контакты CAN 1 C+D D) обычно предоставляют доступ к внутреннейшине CAN устройства, в то время как H + J по-прежнему предоставляют доступ к сети ISOBUS.

3 Как обрабатывать данные CAN

Для обработки необработанных данных о тракторе / орудии вам понадобится подходящее программное обеспечение CAN. В частности, вы загружаете журнал файлы и связанные файлы DBC для извлечения физических значений.

Программные средства CANedge позволяют выполнять различные формы анализа. Например, вы можете использовать преобразователи MF4 для быстрое преобразование файлов журнала в популярные форматы для использования в таких программах, как Vector CANalyzer или PEAK PCAN-Explorer. Или вы можете использовать графический интерфейс asammfdf для DBC декодировать данные и создавать визуальные графики.



Вы также можете выполнять расширенную обработку данных с помощью Python API - идеальный вариант, если вам нужно выполнить статистический анализ, автоматическую отчетность или интеграцию с базой данных.

API Python также можно использовать для интеграции данных с информационными панелями - подробнее ниже.

Пример: Панель управления трактором с данными ISOBUS, J1939 и NMEA 2000

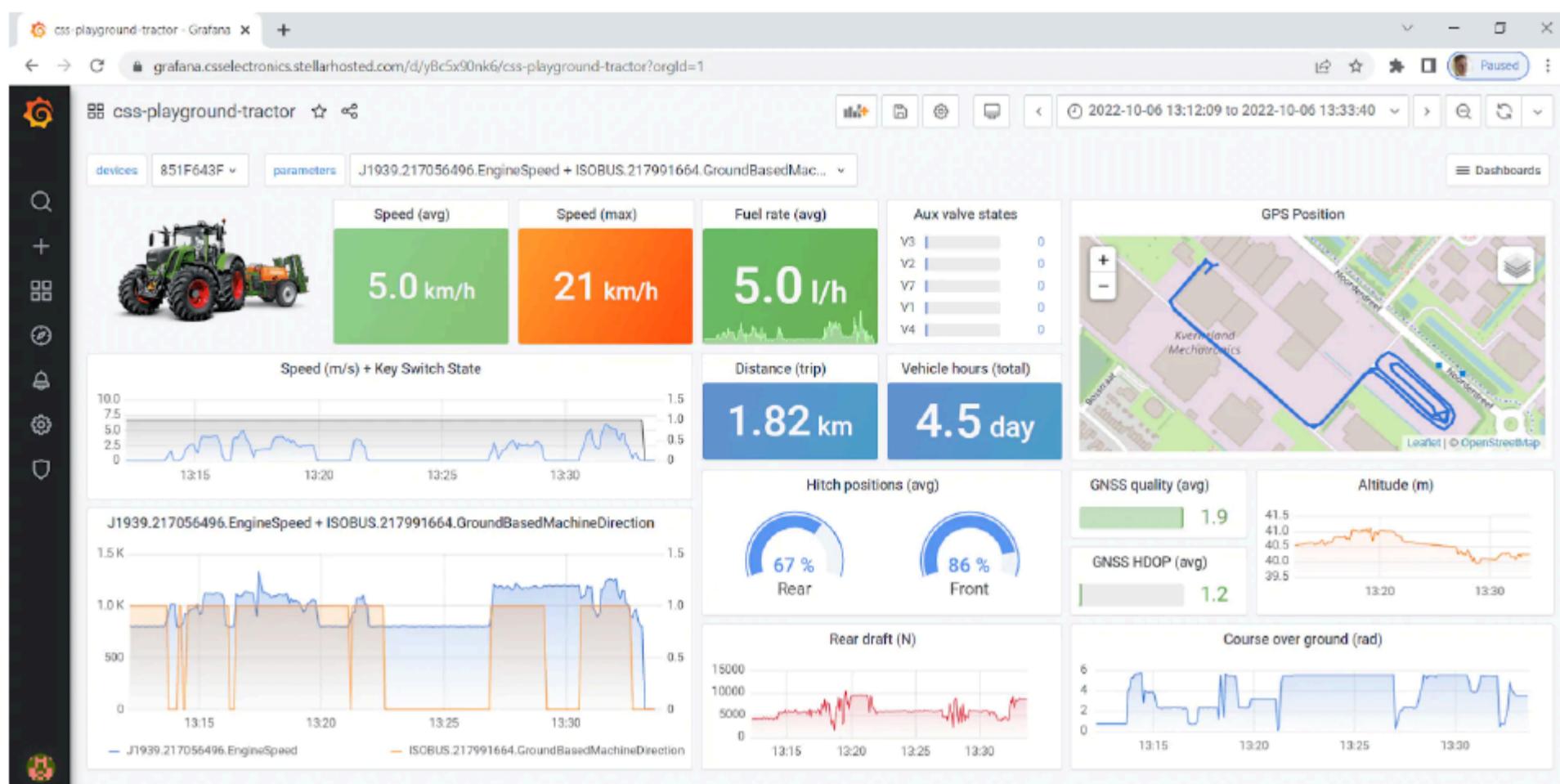
Чтобы проиллюстрировать роль ISOBUS, J1939 и NMEA 2000 в регистрации данных трактора / навесного оборудования, ознакомьтесь с нашей Grafana игровая площадка для приборной панели. Здесь перед каждым сигналом указан протокол, из которого он исходит.

Ознакомьтесь также с нашей вводной частью панели мониторинга телематики, чтобы узнать больше о панелях мониторинга Grafana.

Подробная информация о записанных данных

Файл журнала был записан с помощью CANedge2, подключенного к диагностическому разъему implementation. Здесь CANedge2 специально записал данные из сети ISOBUS. Другими словами, файл журнала не содержит внутренних данных трактора Данные J1939, за исключением сигналов, которые повторяются в сети ISOBUS.

Для декодирования данных мы использовали J1939 DBC, ISOBUS DBC и NMEA 2000 DBC. Мы также использовали ISO TP функциональность CANedge Python API для извлечения некоторых данных NMEA 2000 Fast Packet. Обратите внимание, что проприетарный Данные, относящиеся к конкретному производителю, не включены в playground, поскольку они не входят в стандартные файлы DBC.

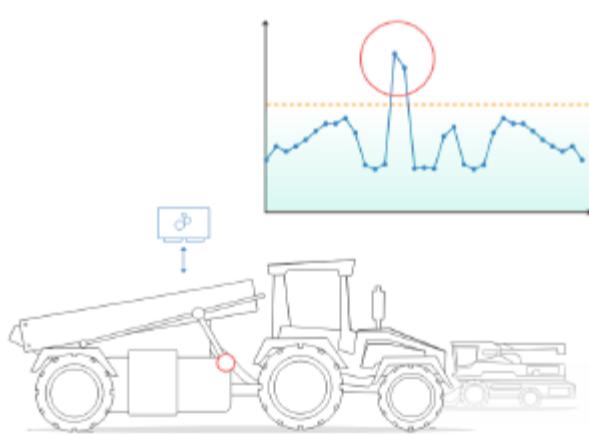


[откройте playground](#)

[получите образцы данных](#)

Примеры использования регистрации данных

Здесь мы описываем ряд классических вариантов использования регистрации данных в сельскохозяйственной технике.



Регистратор данных черного ящика трактора

Необходимо фиксировать периодические неполадки или разрешать споры по гарантии?

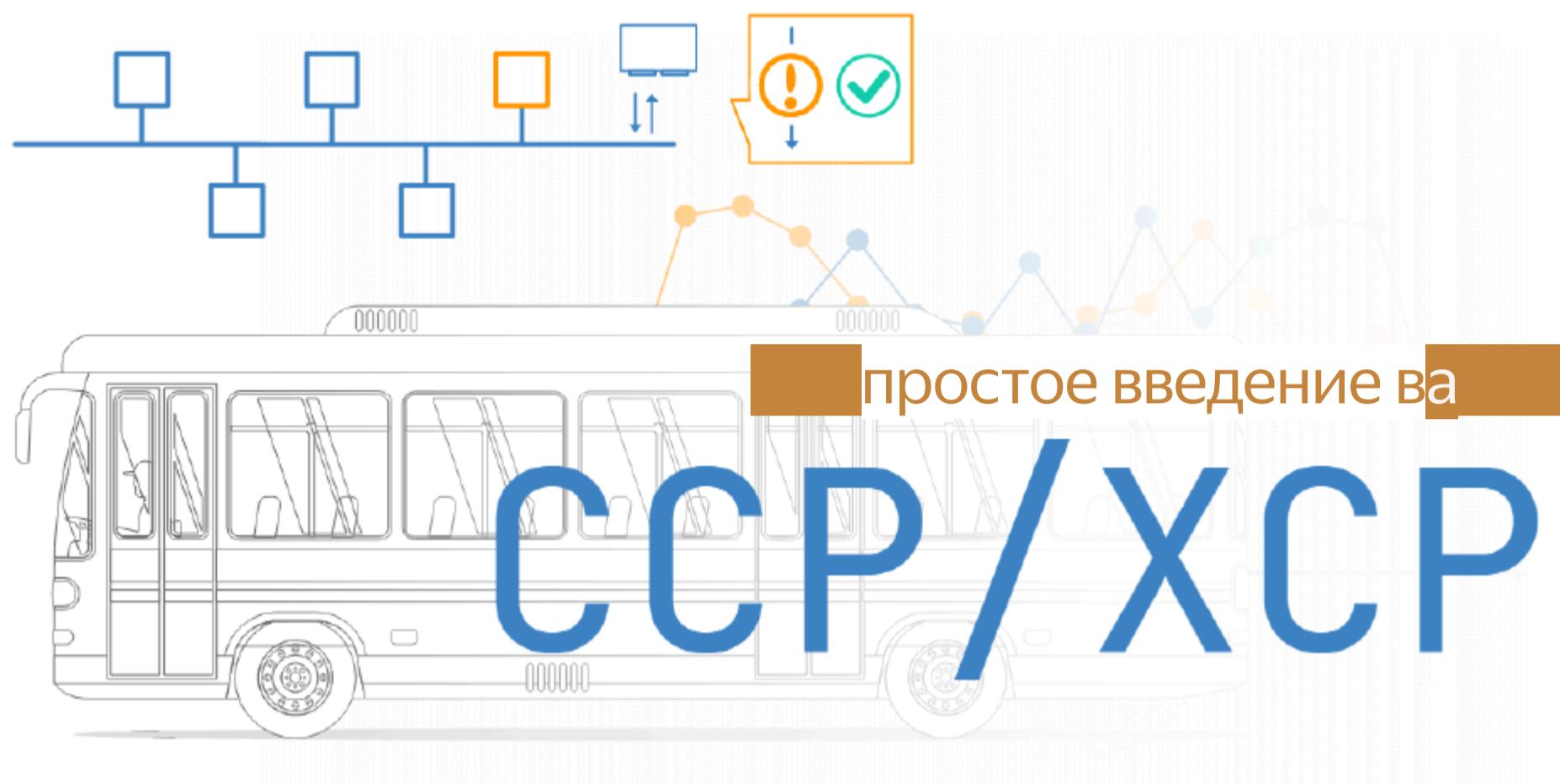
Установив регистратор данных шины CAN на своих тракторах на постоянной основе, вы получите доступ к расширенному обзору всех возможностей ваших Сетей шины CAN. Это делает его идеальным решением для выявления редких неполадок для ускорения диагностики или для разрешения гарантийных случаев споров между производителем и конечным пользователем.



Удаленный мониторинг эксплуатационных данных в сельском хозяйстве.

Требуется удаленный сбор оперативных данных для последующего анализа?

Вам, как оператору или системному интегратору, может потребоваться сбор данных по всему парку сельскохозяйственной техники в полевых условиях. Здесь CANedge2 может быть развернут для включения регистрации необработанных данных CAN. Данные могут быть загружены, когда автомобили вернутся к стационарной сети Wi-Fi - или вы можете установить на каждом CANedge2 точку доступа 3G / 4G для непрерывного сбора данных на местах. Используя CANedge Python API, легко выполнять обработку больших объемов данных, расширенный статистический анализ и отчеты.



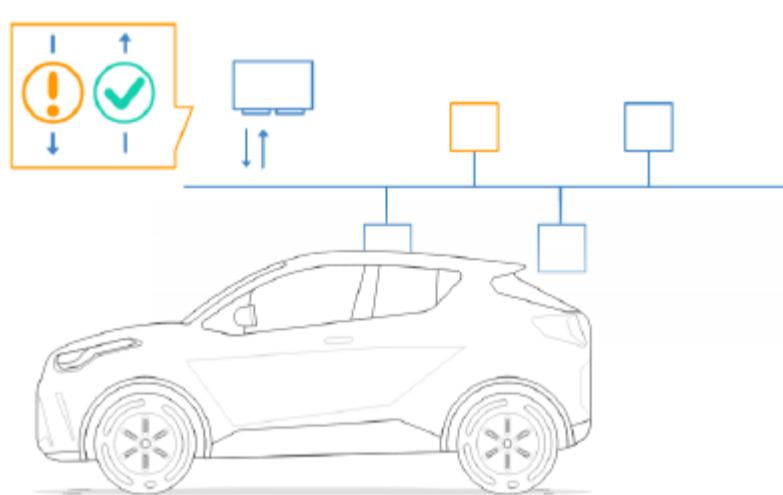
CCP / XCP на CAN объяснено - простое введение

Нужно простое введение в CCP / XCP на CAN bus?

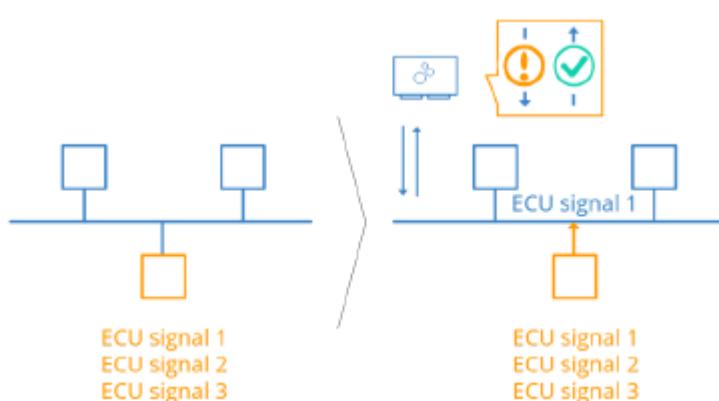
В этом практическом руководстве мы знакомимся с основами протокола калибровки CAN (CCP) и универсального измерения и протокола калибровки (XCP) для CAN. В частности, мы сосредоточимся на фреймовых структурах CCP / XCP, примерах трассировки и A2L Файлы. Мы также расскажем о практической регистрации данных ECU с помощью опроса / DAQ - и о том, как декодировать данные.

Что такое CCP / XCP?

Протокол калибровки CAN (CCP) - это интерфейс, который обеспечивает доступ для чтения/записи к электронному блоку управления (ECU). Он позволяет калибровка, измерение данных, перепрошивка и многое другое. Универсальный Протокол измерений и калибровки (XCP) является преемником CCP с различными улучшениями, включая поддержку большего объема транспортировки уровня, такие как Ethernet, FlexRay и SxL.



Протоколы CCP / XCP имеют большое совпадение, но также и важные различия. Чтобы избежать путаницы, сначала мы сосредоточимся на описании Протокола CCP, а затем рассмотрим XCP на CAN с четкими разъяснениями важных различий.



Чтобы понять мотивацию CCP / XCP, давайте вернемся к нашему простому введению к CAN bus. Как объясняется здесь, CAN обеспечивает обмен данными между разные ЭБУ в транспортном средстве / машине. Входы и выходы каждого ЭБУ будут транслироваться по шине CAN. Однако внутренняя работа ЭБУ скрыта box. Здесь CCP / XCP обеспечивает прямой доступ к внутренней работе блока управления. Это позволяет запрашивать данные высокочастотных параметров, которые в противном случае могут быть только известным ЭБУ. Кроме того, это также позволяет изменять алгоритмы ЭБУ и переменные, упрощая тестирование и калибровку ЭБУ. Важно отметить, что CCP / XCP обеспечивает стандартизированную работу этих интерфейсов у всех производителей ЭБУ.

История CCP / XCP

Протокол калибровки CAN (CCP) был первоначально разработан компанией calibration systems компанией Helmut Kleinknecht. В течение нескольких лет он был усовершенствован работающим group, ASAP (Arbeitskreis zur Standardisierung von Applikationssystemen) that включая Audi, BMW, VW и другие. Позже ASAP была переименована в ASAM (Ассоциация по стандартизации систем автоматизации и измерений).



Протокол калибровки CAN

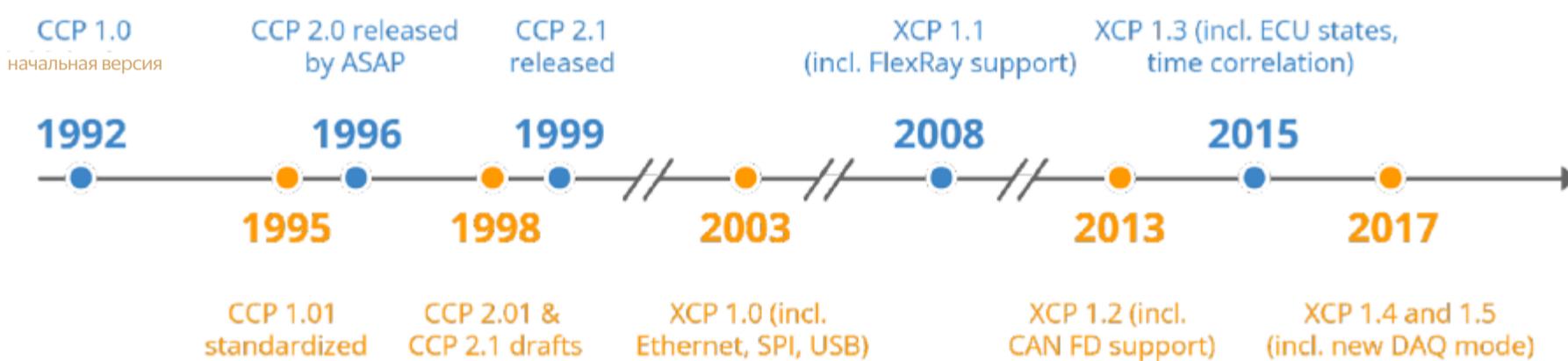


Универсальное измерение
и протокол калибровки

Ниже приведены ключевые этапы:

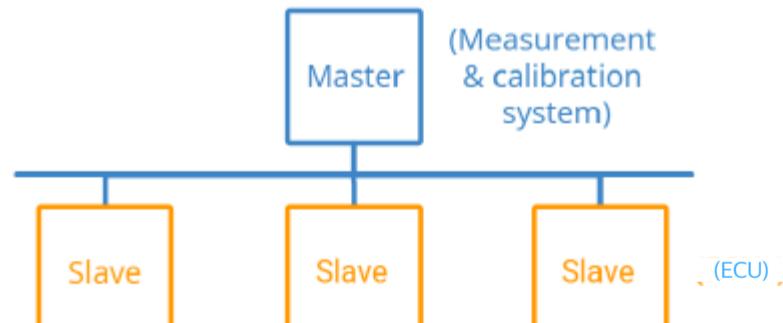
- 1992: первоначальный выпуск CCP 1.0 Гельмутом Кляйнкнхетом.
- 1995: CCP 1.01 стандартизирован ASAP
- 1996: CCP 2.0 выпущен ASAP
- 1998: Подготовлены проекты CCP 2.01 и CCP 2.1
- 1999: CCP 2.1 был выпущен в феврале
- 2003: XCP 1.0 вкл. поддержка CAN, Ethernet, SPI, USB
- 2008: XCP 1.1 вкл. поддержка FlexRay
- 2013: XCP 1.2 вкл. Обновления файла описания ECU + CAN FD
- 2015: XCP 1.3 вкл. Состояния ECU, обработка обхода, временная корреляция
- 2017: XCP 1.4 вкл. Улучшения и новый режим DAQ
- 2017: XCP 1.5 вкл. отладка программного обеспечения без адаптера отладки

Сегодня XCP (он же ASAM MCD-1 XCP) является преемником CCP (он же ASAM MCD-1 CCP). Однако на практике многие ECU по-прежнему пользуются CCP, что позволяет понимать как протоколы, так и ключевые различия.



Архитектура Master-slave

Протокол CCP/XCP основан на концепции единого ведущего устройства / нескольких подчиненных устройств. Внешний инструмент измерения и калибровки (например, ПК / устройство/данные logger) служит ведущим устройством и может считывать / записывать данные с одного или нескольких ECU, также известных как ведомые устройства.



Интерфейс между ведущим и ведомым устройствами называется ASAP1 или ASAM MCD-1. Стандарт CCP/XCP также описывает ASAP2 или ASAM

Интерфейс MCD-2 MC между master и файлом описания ECU.

На практике эта база данных описывает всю необходимую информацию об устройстве Блок управления в стандартном формате файла под названием A2L (файлы описания блока управления) - о котором мы вскоре расскажем.

Варианты использования CCP / XCP

Протокол CCP / XCP позволяет использовать несколько вариантов:

Измерение и калибровка ECU Plug & play

Запись данных ECU с микросекундным разрешением

Доступ к внутренним данным ECU (не транслируются по CAN)

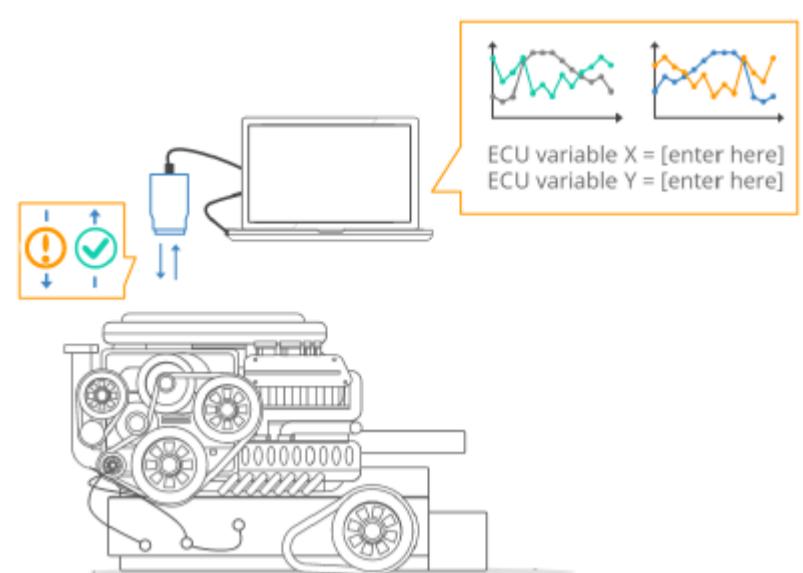
Измерение с помощью опроса или на основе событий (время, триггеры)

Калибровка/настройка переменных алгоритма ECU в реальном времени

Оперативное программирование ECU

Дополнительная аутентификация для безопасного доступа

В основном для OEM-разработки - редко используется после производства



Основные изменения в XCP по сравнению с CCP

XCP добавляет поддержку CAN FD, Ethernet, FlexRay, SxL и других

Меньше интерпретаций - более согласованные реализации

Новый режим "стимулирования" (STIM) для обхода вычислений ECU

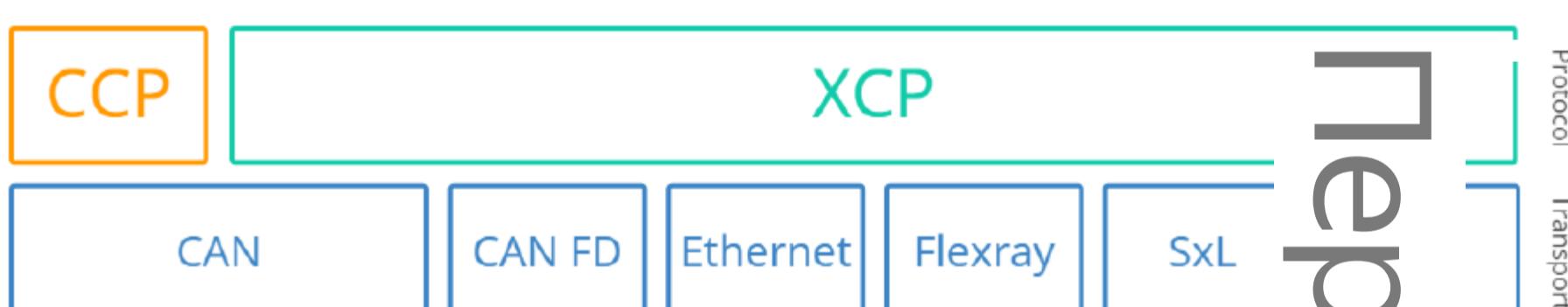
Предопределенные / динамические списки DAQ для эффективной связи

Поддержка синхронного использования различных режимов DAQ

Автоматическое определение ЭБУ (ведущий может опросить подчиненные устройства для получения информации)

Более точный сбор данных за счет измерения временных меток ЭБУ

Большая пропускная способность данных за счет новых дополнительных команд

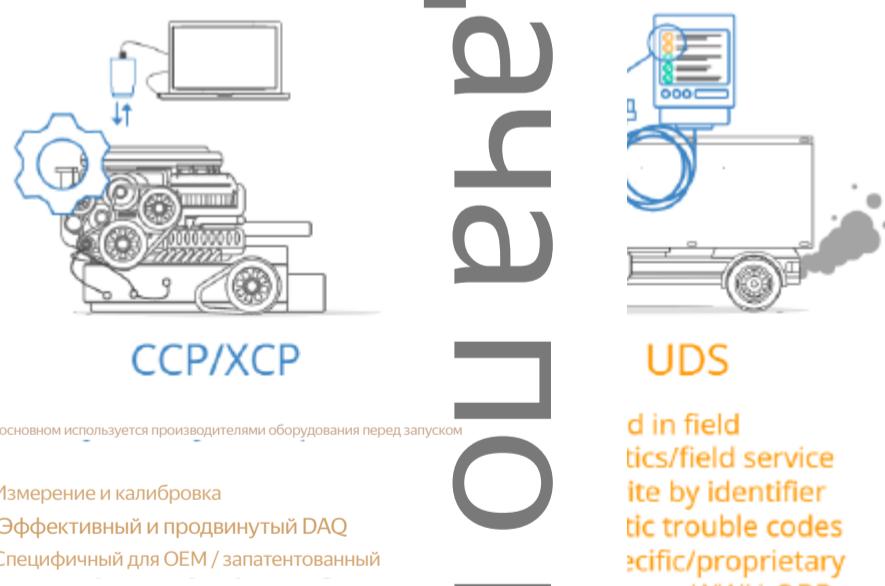


CCP / XCP по сравнению с UDS

Прежде чем мы углубимся в изучение CCP / XCP, может быть полезно понять роль этих протоколов связи по сравнению с [немного похожий протокол, Унифицированные диагностические службы \(UDS\)](#).

Как видно из иллюстрации, CCP / XCP разработан специально для предстартовых измерений и калибровки с помощью OEM-производителям. Как правило, доступ CCP / XCP к ECU отключается как только транспортные средства готовы к запуску. В отличие от UDS, обычно недоступно на ранней стадии разработки прототипа и добавлено только позже - также доступно для сообщение после запуска.

UDS фокусируется на диагностике, в то время как диагностика возможности CCP / XCP невелики. UDS также может быть используется, например, техническими специалистами на местах и в специальных инструментах сканирования OEM.



Однако эти два протокола имеют общие черты - например, в поддержке опроса / циклического сбора данных, перепрошивки ЭБУ и доступ для чтения / записи по адресу (CCP / XCP) или идентификатору (UDS).

Передача по протоколу

Типы сообщений CCP.

Чтобы понять, как работает связь CCP, мы сначала рассмотрим типы сообщений CCP. В целом, связь CCP осуществляется с помощью логики запроса / ответа - аналогичной той, что используется в OBD2, UDS и т.д. Эта связь состоит из двух типов сообщений: Объект приема команд (CRO) и объект передачи данных (DTO).

CRO - объект приема команд.

Объект приема команд (CRO) представляет собой кадр CAN, отправляемый ведущим устройством в блок управления. 1-й байт полезной нагрузки данных является командный байт (CMD). Это определяет, какая команда выдается ведущим устройством блоку управления в соответствии с обзором таблицы.



Для отслеживания выданных команд 2-й байт является счетчиком (CTR). Он имеет отступ +1 для каждой команды, отправленной ведущим устройством и отражается в ответе подчиненного блока управления. Оставшиеся 6 байт зависят от команды.

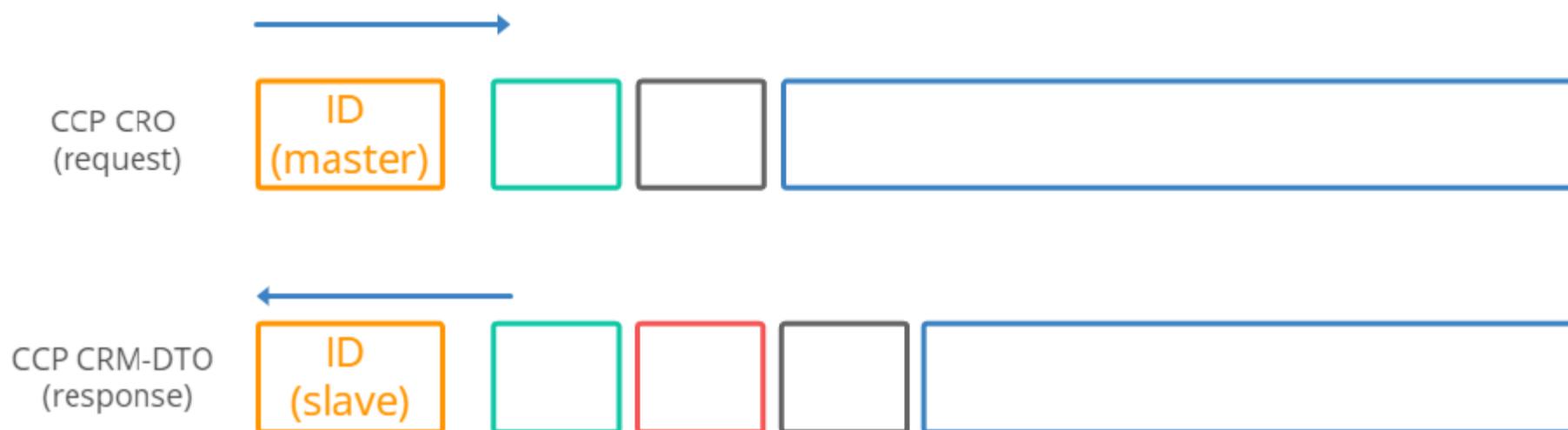
Обзор значений CRO CMD

Список команд CRO, используемых в CCP, смотрите в таблице ниже.

CCP	CRO (Получение команды)	Объект) - CMD	(Команда)	таблица
Код CMD	Команда	Тип	Необходимо?	
0x01	Контакты		Инициализация	
0x1B	GET_CCP_VERSION		Инициализация	
0x17	EXCHANGE_ID		Инициализация	
0x12	GET_SEED	Авторизация	Необходимо	
0x13	РАЗБЛОКИРОВАТЬ	Авторизация	Необходимо	
0x02	SET_MTA			
0x03	DNLOAD			
0x23	DNLOAD_6		Необходимо	
0x04	загрузка			
0x0F	SHORT_UP	Измерение	опроса	Необходимо
0x11	SELECT_CAL_PAGE			Необходимо
0x14	GET_DAQ_SIZE	Измерение	DAQ	
0x15	SET_DAQ_PTR	Измерение	DAQ	
0x16	ЗАПИСЬ DAQ	Измерение	DAQ	
0x06	START_STOP	Измерение	DAQ	
0x07	ОТКЛЮЧЕНИЕ	Измерение	DAQ	
0x08	START_STOP_ALL	Измерение	DAQ	Необходимо
0x0C	SET_S_STATUS			Необходимо
0x0D	GET_S_STATUS			Необходимо
0x0E	BUILD_CHECKSUM			Необходимо
0x10	CLEAR_MEMORY			Необходимо
0x18	ПРОГРАММА			Необходимо
0x22	PROGRAM_6			Необходимо
0x19	ПЕРЕМЕСТИТЬ			Необходимо
0x05	ПРОВЕРИТЬ			Необходимо
0x09	GET_ACTIVE_CAL_PAGE			Необходимо
0x20	DIAG_SERVICE			Необходимо
0x21	ACTION_SERVICE			Необходимо

CCP МОЖЕТ создавать идентификаторы кадра

В CCP используются два идентификатора CAN: один для сообщений, отправляемых ведущим (например, 0x701), и один для сообщений, отправляемых ведомое устройство (например, 0x702). Эти идентификаторы будут указаны как часть A2L (он же файл описания ECU), что означает, что ведущее устройство получит эту информацию до начала подключения.



Обычно для CCP используются идентификаторы с низким приоритетом, чтобы избежать искажения важной для безопасности информации на шине.

Важно отметить, что это означает, что master нацеливается не на конкретные ECU с помощью различных идентификаторов CAN, а скорее через последовательность подключения, как показано ниже.

Пример: сообщение CCP CRO CONNECT

Ниже приведен пример CRO, используемого для инициализации связи с конкретным ECU:

>Trace example: CCP initialization (CONNECT CMD)

<Время	ИДЕНТИФИКАТОР CAN (шестнадцатеричный)	Байты данных (шестнадцатеричный)	Отправитель
1.01	701	01 01 39 01 AA AA AA AA	мастер

Сведения о трассировке

Здесь идентификатор CAN отражает идентификатор, используемый мастером CCP для связи. 1-й байт равен 0x01, что соответствует Команде CONNECT, как видно из предыдущей таблицы. 2-й байт - это значение CTR. 3-й и 4-й байты - это специфичны для команды CONNECT и соответствуют адресу станции целевого блока управления в порядке байтов INTEL. В другом другими словами, указанный выше CRO используется ведущим устройством для установления соединения с ECU 0x0139. Все последующие сообщения в этом случае будет использоваться с этим конкретным блоком управления, пока ведущий не разорвет соединение или не подключится к другому блоку управления.

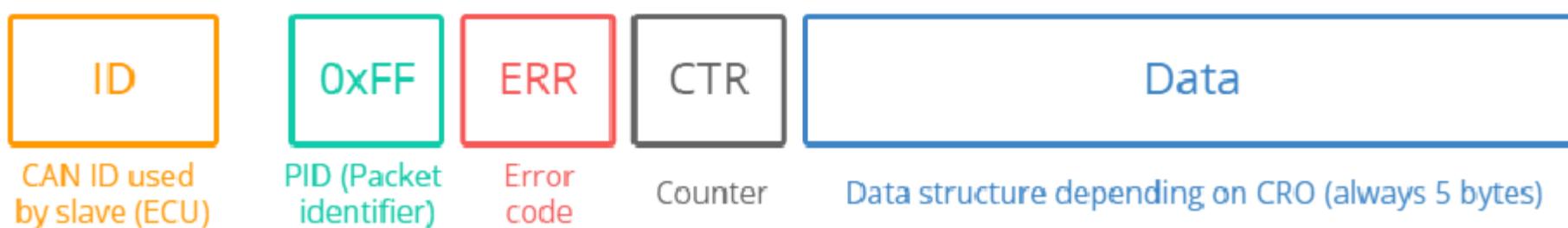
DTO - объект передачи данных.

Объект передачи данных (DTO) представляет собой кадр CAN, отправляемый ECU ведущему устройству. Существуют три типа DTO, как описано ниже:

#1 CRM-DTO: сообщение в ответ на команду.

CRM-DTO отправляется блоком управления в ответ на запрос CRO от ведущего устройства. Здесь идентификатор CAN отражает идентификатор CAN, используемый Блок управления (например, 0x702).

CCP CRM-DTO (объект ответа на команду)



Полезная нагрузка данных для CRM может быть разбита следующим образом:

1-й байт является идентификатором пакета (PID). Для CRM-DTO, PID всегда равен 0xFF

2-й байт - это код ошибки (ERR), который может использоваться, например, для информирования главного устройства о недопустимом запросе.

3-й байт - это счетчик (CTR), который будет соответствовать значению CTR из CRO мастера

Структура оставшихся 5 байтов зависит от исходного запроса, сделанного в CRO

Ниже приведен предыдущий пример отслеживания подключения, включая CRM с положительным ответом.

>Trace example: CCP initialization (CONNECT CMD + ECU response)

<Время	ИДЕНТИФИКАТОР CAN (шестнадцатеричный)	Байты данных (шестнадцатеричный)	Отправитель	Тип фрейма
1.01	701	01 01 39 01 AA AA AA AA	мастер	CRO (ПОДКЛЮЧИТЬ CMD)
1.04	702	FF 00 01 AA AA AA AA AA AA	Подчинение	CRM (OK)

Трассировка детали

В трассировке показывает подключение к блоку управления с адресом станции 0x0139, который отвечает на команду инициализации, отправленную ведущим устройством. Как видно, 1-й байт равен 0xFF (поскольку сообщение является CRM). 2-й байт показывает, что ошибок не было произошло, в то время как 3-й байт соответствует CTR CRO. С помощью этой последовательности устанавливается соединение между ведущим устройством и ECU 0x0139.

Обзор значений ошибок CRM-DTO / EV-DTO

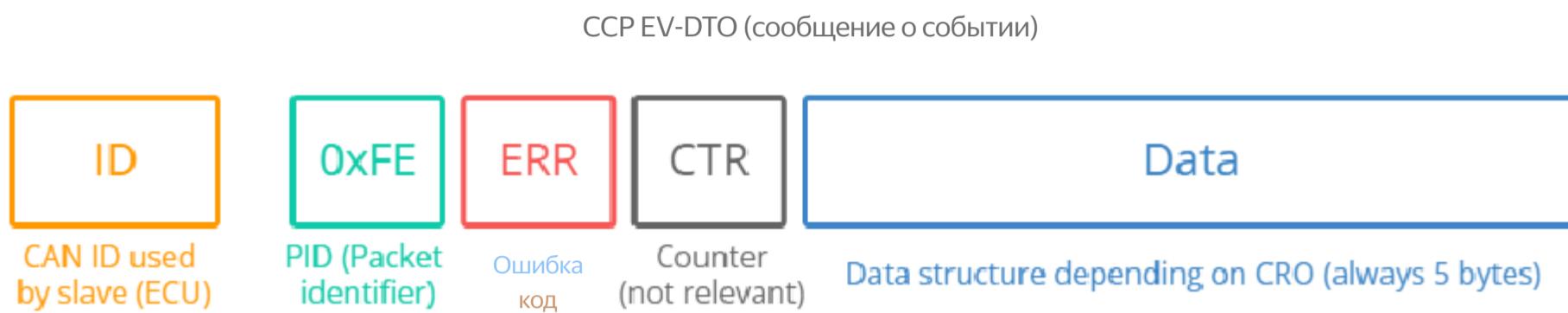
В таблице ниже мы показываем набор значений кодов ошибок CRM-DTO / EV-DTO. Обратите внимание, что коды ошибок могут быть отправлены в обоих CRM-DTO (если они выполняются непосредственно в ответ на запрос CRO от ведущего устройства) и в EV-DTO (если они выполняются асинхронно с командами CRO ведущего устройства).

>CCP CRM-DTO/EV-DTO ERR code table

<ОШИБКА кода	Описание	Ошибка категория	Переход состояния
0x00	Подтверждение (ошибки нет)		
0x01	Перегрузка процессора DAQ	C0...	
0x10	Занят командный процессор	C1...	Нет (ожидание)
0x11	Процессор DAQ занят	C1...	Нет (ожидание)
0x12	Внутренний тайм-аут	C1...	Нет (ожидание)
0x18	Запрос ключа	C1...	Нет
0x19	Запрос состояния сеанса	C1...	Нет
0x20	Запрос холодного запуска	C2...	ХОЛОДНЫЙ ЗАПУСК
0x21	Cal. инициализация данных. запрос	C2...	Cal. инициализация данных
0x22	Инициализация списка DAQ. запрос	C2...	Инициализация списка DAQ
0x23	Запрос на обновление кода	C2...	(ХОЛОДНЫЙ ЗАПУСК)
0x30	Неизвестная команда	C3...	(ОШИБКА)
0x31	Синтаксис команды	C3...	ОШИБКА
0x32	Параметр (ы) вне зоны действия сети	C3...	ОШИБКА
0x33	Доступ запрещен	C3...	ОШИБКА
0x34	Перегрузка	C3...	ОШИБКА
0x35	Доступ заблокирован	C3...	ОШИБКА
0x36	Ресурс / функция. недоступен	C3	ОШИБКА

2 EV-DTO: сообщение о событии

EV-DTO отправляется блоком управления в ответ на внутреннее событие, вызывающее изменение состояния блока управления. Это можно использовать для сообщите мастеру об ошибках, произошедших с момента последнего CRO.



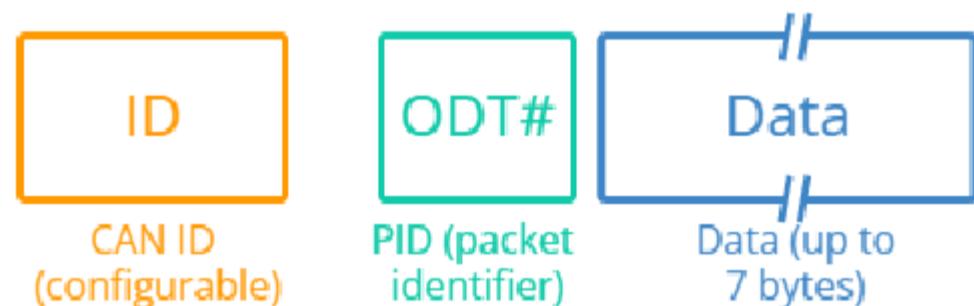
Структура EV-DTO идентична CRM-DTO из предыдущих, с PID EV, всегда равным 0xFE. Обратите внимание, что CTR не имеет значения для EV-DTO.

3 DAQ-DTO: сообщение о получении данных

DAQ-DTO используется ECU для автоматической отправки данных к ведущему устройству в ответ на определенное событие (например, циклический счетчик, кнопка или подобное).

При передаче данных DAQ мастер начинает с "конфигурирования" блока управления для конкретной последовательности измерений. После запуска, ECU будет выводить DAQ-DTO без дальнейших CRO от ведущего устройства.

CCP DAQ-DTO (сообщение о сборе данных)



В DAQ-DTO PID является ссылкой на 'Объект Таблица дескрипторов' (ODT), в то время как до 7 байт содержат данные, относящиеся к ODT. Подробнее об этом чуть позже.

Ниже приведен пример сообщения DAQ-DTO, отправленного ECU ведущему устройству.

>Trace example: CCP DAQ-DTO frame with 5 byte signal payload

<Время ИДЕНТИФИКАТОР CAN (ШЕСТНАДЦАТИБИТНЫЙ) Байты данных (шестнадцатибитный) Отправитель			Тип кадра
12.55 712	25 A1 42 34 AF 22	Подчинение	DAQ-DTO

Объясненная трассировка

Трассировка показывает передачу данных DAQ-DTO от подчиненного устройства в соответствии с последовательностью начальной настройки. Данные относятся к списку ODT # 37 (0x25) с полезной нагрузкой в 5 байт. Обратите внимание, что мы не заполняем неиспользуемые байты (поскольку это не требуется для DAQ-DTO). Обратите также внимание на то, как вводится новый идентификатор CAN для этого ответа DAQ (подробнее об этом позже).

Общие замечания по сообщениям CCP

Длина полезной нагрузки сообщений CRO, CRM-DTO и EV-DTO должна быть равна 8. Все неиспользуемые байты дополняются произвольными значениями (в этом введении мы используем 0xAA). Сообщения DAQ-DTO могут использовать фактический размер полезной нагрузки.

Напомним: главный инструмент использует сообщения CRO для отправки различных команд подчиненному блоку управления. Подчиненный блок управления, в свою очередь, может использовать DTO сообщения для ответа ведущему устройству. Далее мы покажем, как с помощью этих сообщений можно выполнять измерение данных.

Как записать данные блока управления с помощью CCP

Давайте представим, что вы инженер, работающий на автомобильном заводе-изготовителе. Вам было поручено извлекать значение определенного параметра из блока управления в течение длительного периода времени.

Как вы это делаете? В следующих разделах мы описываем два метода: опрос и DAQ.

Сбор данных с помощью опроса CCP.

Самым простым решением было бы использовать концепцию измерения CCP, называемую опросом. Ниже мы проиллюстрируем, как может выглядеть такой коммуникационный поток:

>Trace example: CCP polling via SHORT_UP command

<Время	ИДЕНТИФИКАТОР CAN (ШЕСТНАДЦАТИБИТОВЫЙ)	Байты данных (ШЕСТНАДЦАТИБИТОВЫЙ)	Отправитель	Тип фрейма
1.51	701	0F 02 04 00 12 34 56 78	ведущий	CRO (SHORT_UP CMD)
1.53	702	FF 00 02 F1 2A 71 2F AA	Подчинение	CRM (OK + ДАННЫЕ)

Здесь ведущий отправляет подчиненному запрос на 0x04 байта данных, хранящихся по "адресу источника" 0x12345678. 4-й байт равен 0x00, что отражает "расширение адреса" для этого Адреса источника ECU, который может, например, отражать конкретную память сегмент.

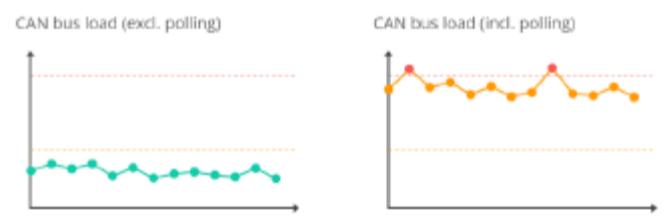
Ведомое устройство отвечает 4 байтами данных, 0xF12A712F, для этого конкретный адрес источника. Этот приведенный выше поток может быть использован для например, извлекать данные параметров в реальном времени, такие как сигнал частоты вращения. Это несколько похоже на то, как вы можете запросить частоту вращения через Поток запросов / ответов OBD2 в большинстве автомобилей.



Фактически, "опрос" - это просто последовательность CRO / CRM-DTO при этом ведущий использует команду SHORT_UP (сокращенно) загрузить).

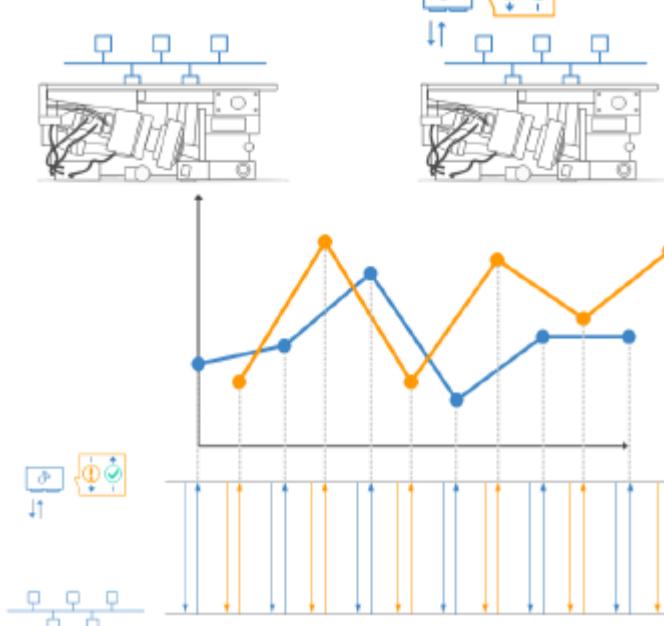
Плюсы и минусы опроса CCP

Опрос прост в настройке - просто установите соединение с ECU и отправляйте правильное сообщение запроса каждые x мс. Во многих практических случаях это может быть вполне жизнеспособным решением. Однако у опроса есть два существенных недостатка.



Опрос № 1 неэффективен

Для опроса требуется сообщение с запросом для каждого ответа. Если вы, например, необходимо измерить 2-байтовый сигнал с частотой 100 Гц, что увеличивает загрузку шины на 200 кадров в секунду. Неэффективность связана как с необходимостью запрашивать каждый отдельный ответ, так и с тем фактом, что ответные сообщения вынуждены быть длиной 8 байт, при этом 3 байта тратятся на служебные данные.



2 Опрос выполняется асинхронно

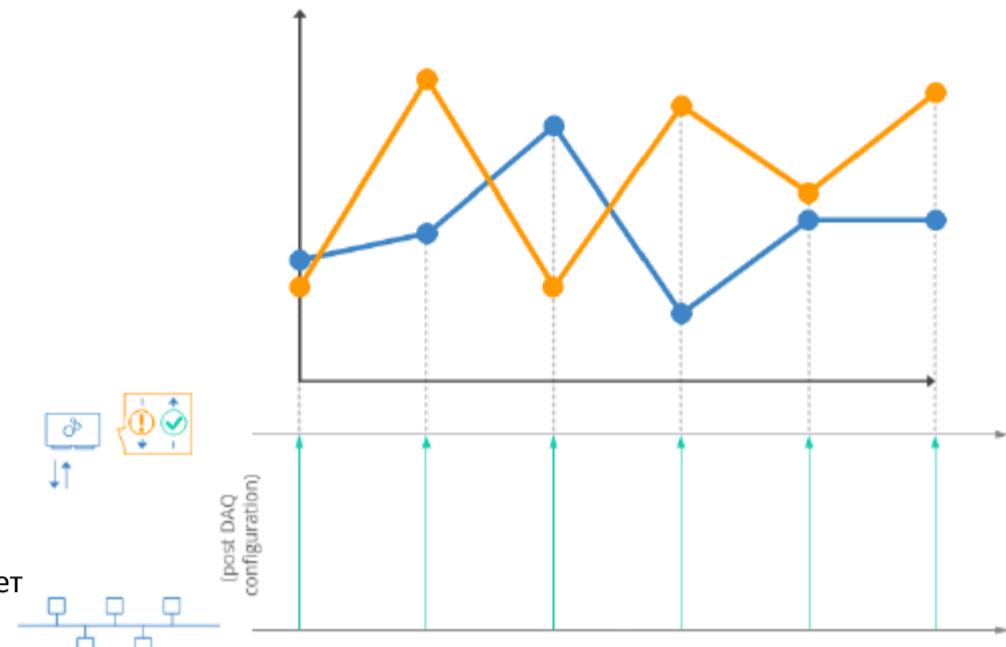
При опросе нескольких сигналов запрос / ответы необходимо отправлять последовательно с небольшой задержкой между каждым запросом. В результате сигнал наблюдения не синхронизируются по времени. Это затрудняет анализ данных и последующую обработку более сложной и менее точной.

Сбор данных с помощью CCP DAQ

CCP предлагает альтернативный метод измерения данных называется DAQ (синхронный сбор данных). DAQ немного сложнее инициировать, но решает оба недостатка опроса.

Проще говоря, DAQ работает следующим образом:

Ведущий определяет, какие измерения записывать от ведомого устройства, а также какое событие должно инициировать передачу данных. Например, ведущий может запросить, чтобы сигналы A и B транслировались каждые 10 мс, в то время как сигнал C может транслироваться каждые 100 мс. Ведущий может даже настроить ведомые устройства для трансляции сигналов на основе таких событий, как нажатие кнопки или изменение угла наклона. После завершения настройки DAQ ведущий "запускает" последовательность, и ведомое устройство теперь работает автономно транслирует запрошенные сигналы, используя сообщения DAQ-DTO.



Это устраняет сообщения-запросы от ведущего устройства. Кроме того, сигнальные данные более эффективно упаковываются в DAQ-DTO по сравнению с CRM-DTO, что снижает нагрузку на шину. Кроме того, с помощью DAQ можно упаковывать связанные сигналы в одни и те же кадры DAQ-DTO, гарантируя, что запрошенные сигналы измеряются синхронно по времени, улучшая качество данных для анализа.

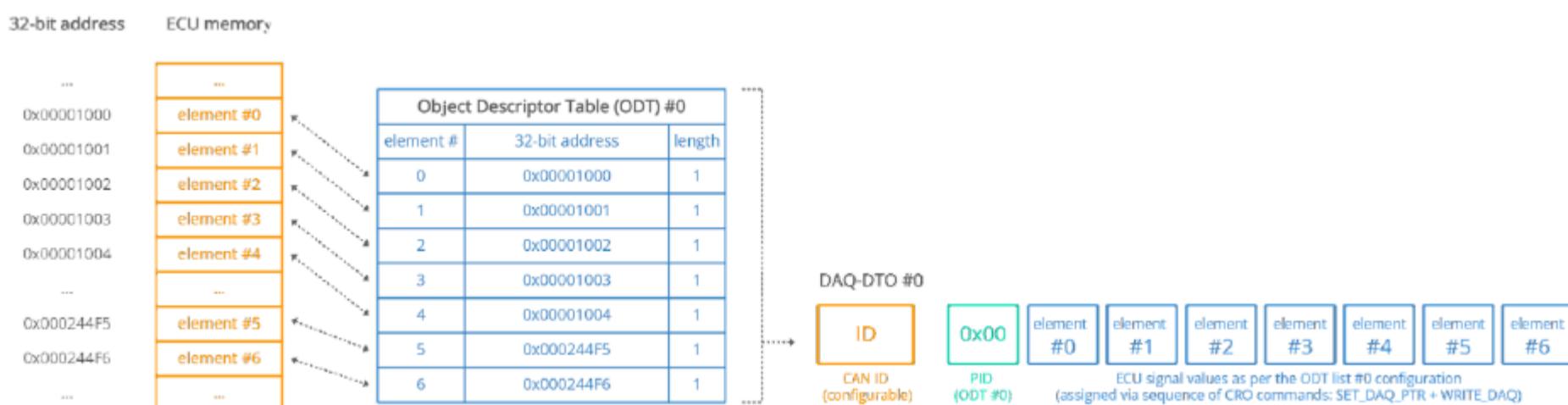
Как настроить последовательность DAQ

Чтобы использовать режим DAQ для измерения данных, мастер должен сначала соответствующим образом настроить ECU. На практике, этот процесс обычно управляет сложными инструментами графического интерфейса (подробнее об этом позже). Однако, чтобы полностью понять, что происходит далее мы рассмотрим, что происходит "под капотом".

Две концепции являются ключевыми в **DAQ: Таблицы дескрипторов объектов (ODT) и списки DAQ**.

Таблица дескрипторов объектов

Таблица дескрипторов объектов (ODT) представляет собой список ссылок на элементы данных из памяти ECU. Если мы возьмем пример 'short upload', используемый в опросе CCP, мастер отправляет запрос с указанием трех элементов: Длина данных (в байтах), Расширение адреса и адрес источника. С помощью этой информации ECU находит данные и отправляет их ведущему устройству. Концепция ODT аналогична: ODT - это просто список ссылок на элементы. Каждая запись в ODT отражает адрес источника ECU - и, при необходимости, также расширение адреса и длину данных. Другими словами, запись ODT содержит ту же информацию, что и мы используется для опроса сигнала.



Пример ODT и подробности

Давайте рассмотрим пример:

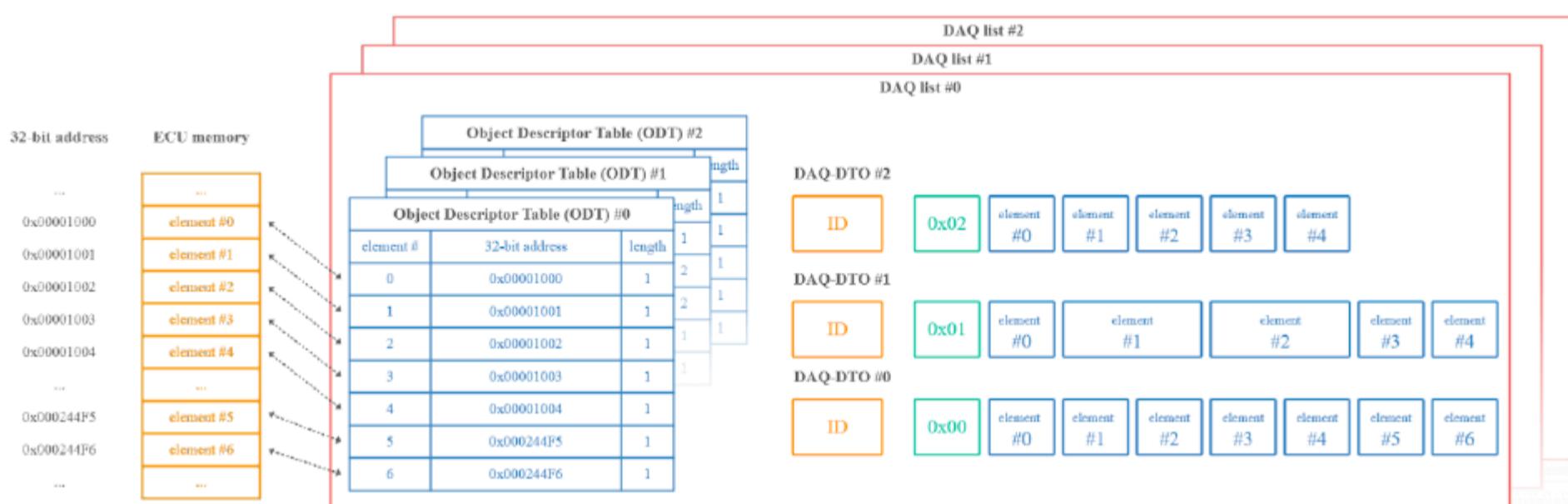
Мы хотим записать 7 сигналов, каждый из которых длиной в 1 байт. Для этого мы определяем новый ODT #0 (PID 0x00) с 7 элементами записи. Элемент 1 относится к сигналу 1 с определенным адресом источника ECU. Элемент 2 относится к сигналу 2 с другим адресом источника и т.д.

Когда мы используем режим DAQ для измерения данных, теперь мы можем обратиться к ODT # 0, чтобы получить информацию от ECU синхронизированные по времени данные по всем 7 сигналам - в пределах одного кадра DAQ-DTO CAN. Другими словами: ODT, который мы определили описывает структуру сообщения DAQ-DTO, означающую, что ECU теперь будет знать, как упаковать 7 сигнальных байтов в одном кадре CAN - и мастер знает, как извлечь 7 сигнальных байт из этого кадра CAN. Обратите внимание на влияние на загрузку шины: если бы мы опрашивали все 7 сигналов, для этого потребовалось бы 14 кадров CRO / CRM-DTO CAN на цикл - теперь требуется только 1 кадр CAN DAQ-DTO.

Кроме того, мы можем легко построить эти 7 сигналов вместе, поскольку они имеют одну и ту же временную метку кадра CAN, что значительно упрощает работу более простой в выполнении анализ. По той же причине списки ODT обычно определяются таким образом, чтобы группировать связанные сигналы вместе. На практике мастер часто определяет несколько списков ODT и назначает PID для каждого из них в диапазоне от 0x00 до 0xFD. Затем каждый ODT определяет структуру отдельного DAQ-DTO.

Списки DAQ

При работе с несколькими списками ODT полезно сгруппировать их вместе. Эти группы списков ODT называются Списки DAQ. Например, список DAQ #0 может содержать ODT #0, ODT #1 и ODT #2.



Список DAQ характеризуется способом выборки данных:

Список DAQ # 0 может быть настроен таким образом, чтобы выборка производилась для всех DAQ-DTO в нем каждые 10 мс

Список DAQ # 1 может быть настроен таким образом, что DAQ-DTO в нем выбираются при нажатии кнопки

Другими словами, требуется отдельный список DAQ для каждой уникальной логики выборки, запрошенной для измерения данных. Мы вкратце опишем инициализацию DAQ и ODT, но сначала рассмотрим приведенный ниже пример трассировки для уже инициированного DAQ последовательность:

>Trace example: CCP DAQ messages (multiple DAQ and ODT lists)

<Время	Идентификатор CAN (шестнадцатеричный)	Байты данных(ШЕСТНАДЦАТЕРИЧНЫЙ)	Отправитель	Тип рамки
2.44	712	00AF 32	4A 00	00 00 F1 Подчинение DAQ-DTO (DAQ) #0 ODT #0
2.44	712	0122 01	00 A0	BB 09 F1 Подчинение DAQ-DTO (DAQ) #1 ODT#1
2.44	712	0201 00	FF FF	FF 1A 20 Подчинение DAQ-DTO (DAQ) #1 ODT#2
2.45	712	00AF 32	4B 00	0A 00 F1 Подчинение DAQ-DTO (DAQ) #0 ODT #0
2.46	712	00AF 32	4B 01	0A 00 F1 Подчинение DAQ-DTO (DAQ) #0 ODT #0
2.47	712	00AF 33	4A 01	0B 00 F1 Подчинение DAQ-DTO (DAQ) #0 ODT #0
....				
2.54	712	00AF 33	4C 01	0B 00 F1 Подчинение DAQ-DTO (DAQ) #0 ODT #0
2.54	712	0123 01	11 A1	BC 08 F4 Подчинение DAQ-DTO (DAQ) #1 ODT#1
2.54	712	0202 01	FE FE	FF 1B 20 Подчинение DAQ-DTO (DAQ) #1 ODT#2
2.55	712	00AF 34	4C 02	0C 00 F1 Подчинение DAQ-DTO (DAQ) #0 ODT #0
2.56	712	00AF 34	4D 02	0C 00 F1 Подчинение DAQ-DTO (DAQ) #0 ODT #0
2.57	712	00AF 34	4E 02	0A 00 F1 Подчинение DAQ-DTO (DAQ) #0 ODT #0
2.64	712	00AF 31	4D 01	0A 00 F1 Подчинение DAQ-DTO (DAQ) #0 ODT #0
2.64	712	0124 05	01 A0	BC 09 F6 Подчинение DAQ-DTO (DAQ) #1 ODT#1
2.64	712	0201 00	ФФ ФФ	FF 1C 21 Подчинение DAQ-DTO (DAQ) #1 ODT #2
2.65	712	00AF 31	4D 01	0B 00 F1 Подчинение DAQ-DTO (DAQ) #0 ODT #0

Трейс объяснил

В этом примере целевой ECU передает данные в общей сложности по трем спискам ODT, как видно из первых байтов диапазон от 0x00 до 0x02. Хотя из трассировки это явно не ясно, три ODT разделены на два списка DAQ: Один список DAQ содержит ODT #0 с частотой дискретизации 10 мс, в то время как список DAQ # 1 содержит остальные списки ODT с # 1 по # 2 с частотой дискретизации 100 мс. Вот почему DAQ-DTO с PID 0x00 чаще наблюдается в трассировке по сравнению с другими DAQ-DTO.

Как инициализировать последовательность DAQ

Теперь мы рассмотрели оба списка ODT и DAQ - но как нам определить их с помощью CAN?

Проще говоря, настройка списков DAQ выполняется с помощью (потенциально длинной) последовательности кадров CRO/ CRM-DTO. Здесь мастер, по сути, определяет всю структуру DAQ / ODT /element поэлементно. Чтобы определить новый сигнал (он же элемент), мастер определяет адрес, длину и расширение адреса. Затем мастер информирует ECU как упаковать элемент в DAQ-DTO, связав элемент с ODT # и DAQ #. Чтобы понять это подробно, смотрите Приведенный Ниже Пример трассировки и последующее объяснение:

>Trace example: CCP DAQ initialization and initial DAQ-DTOs

<Время	Идентификатор CAN (ШЕСТНАДЦАТЕРИЧНЫЙ)	Байты данных				(ШЕСТНАДЦАТЕРИЧНЫЙ)	
1.51	701	14	03	05		AA	00
1.55	702	FF	00	03		03	07
1.62	701			15 04 05 07 00 AA			
1.69	702	FF	00	04		AA	AA
1.71	701			16 05 01 00 00 00			
1.74	702	FF	00	05		AA	AA
1.79	701			15 06 05 07 01 AA			
1.82	702	FF	00	06		AA	AA
1.85	701	0F					07 01 00 00 00 10
1.88	702	FF	00	07		AA	AA
....							
2.24	701			15 16 05 07 06 AA			
2.27	702	FF	00	16		AA	AA
2.30	701	0F					17 01 00 00 00 10
2.34	702	FF	00	17		AA	AA
2.37	701			06 18 01 05 07 03			
2.41	702	FF	00	18		AA	AA
2.44	712	07	AF	32		4A	00
2.54	712	07	AB	22		4C	01
2.64	712	07	РЕКЛАМА	24		4B	02
2.74	712	07	AB	25		4C	03
....							

Трассировка объяснена

Итак, у нас здесь довольно много работы, но давайте разберем ее по порядку:

Мы начнем с использования команды CRO GET_DAQ_SIZE . Это имеет две цели: информирует нас о размере указанного списка DAQ в терминах списков ODT - и очищает текущий список. По сути, это работает как команда 'reset' для списка DAQ, в данном случае списка DAQ #5.

Как часть команды GET_DAQ_SIZE, мы также предоставляем 11-битный идентификатор CAN 0x712 в последних 4 байтах. Это информирует Блок управления, который он должен использовать, может использовать идентификатор 0x712 для трансляции последующих DAQ-DTO. Это тонкий, но важный аспект: По сути, ведущее устройство может указать идентификатор CAN для каждого списка DAQ, что, в свою очередь, также позволяет ведущему устройству инициировать Параллельное измерение DAQ в нескольких ECU.

Блок управления подтверждает CRO от главного устройства и сообщает нам, что в списке DAQ # 5 есть 3 ODT с первым PID, равным 0x07. Затем мастер начинает заполнять список DAQ #5 и список ODT # 7 информацией об адресе источника для сигналов размером в 1 байт. 7×1 . Это делается с помощью простого цикла, состоящего из 7 повторений двух команд: SET_DAQ_PTR и WRITE_DAQ.

Сначала мастер использует SET_DAQ_PTR для "выбора" списка DAQ # 5, списка ODT #7 и элемента # 0 из списка ODT. Затем мастер использует WRITE_DAQ для записи содержимого ссылки на этот элемент: длина равна 1, расширение адреса равно 0 и исходный адрес равен 0x00001000.

По сути, теперь мы сказали ECU "упаковать" значение нашего первого сигнала (которое хранится в указанном источнике адрес) во 2-й байт DAQ-DTO, который соответствует списку DAQ #5 и списку ODT #7 (1-й байт DAQ-DTO - это PID 0x07).

После этого мы повторяем процесс для оставшихся 6 сигналов, пока не будет завершен ODT # 7.

В этом упрощенном случае нас интересует только этот конкретный список DAQ и ODT. Следовательно, последним шагом является запуск DAQ измерение с помощью команды START_STOP. Здесь мы указываем, что мы хотим запустить сбора данных список № 5 и список поддерживаемых #7. Как

частично мы указываем параметры синхронизации, а именно, что канал событий 0x03 должен использоваться с предустановкой 10.
Сведения о канале событий для конкретного ECU указаны в файле описания ECU.

Очевидно, что теперь ECU начнет транслировать данные из списка DAQ № 5 и списка ODT № 7 на указанной частоте. Полезная нагрузка кадра CAN включает в себя ODT PID 0x05 в 1-м байте и 7 значений сигнала "элемента" в оставшейся части полезной нагрузки.

Обратите также внимание, что вы можете в качестве альтернативы использовать команду START_STOP для "подготовки" нескольких списков DAQ и ODT для измерение, а затем используйте команду START_STOP_ALL для одновременного запуска или остановки всех из них.

Как отключить от блока управления

После завершения сбора данных мастер может отключиться от блока управления с помощью команды DISCONNECT.

Давайте рассмотрим пример трассировки:

>Trace example: Отключение CCP (DISCONNECT CMD + ответ ECU)

<ВремяИдентификатор CAN (шестнадцатеричный)Байты данных (шестнадцатеричный)		Отправитель	Тип кадра
24.01	701	07 88 01 AA 02 07 AA AA	мастер
24.04	702	FF 00 88 AA AA AA AA AA AA	Подчинение

Трассировка объяснена

Здесь 3-й байт 0x01 означает, что мы полностью завершаем сеанс (в отличие от временного отключения). Это эффективно сбрасывает ECU, включая нашу предыдущую конфигурацию. Чтобы настроить таргетинг на конкретный интересующий нас блок управления, мы указываем адрес станции (формат Intel) блока управления, т.е. 0x702. Как видно, DAQ более сложен в настройке, но обеспечивает эффективную передачу данных ECU с синхронизацией по времени.

Декодирование данных сигналов CCP от ECU.

В CCP polling / DAQ вы на практике записываете необработанные кадры CAN с определенными структурами кодирования сигнала. Чтобы чтобы понять смысл записанных данных, вам нужно декодировать их в удобочитаемую форму, иначе физические значения. Это то же самое концепция, которую мы объясняли в нескольких других вводных, связанных с CAN, включая наше введение к файлам CAN bus, J1939, OBD2 и DBC. Однако декодирование CCP сопряжено с некоторыми дополнительными сложностями, которые мы объясним ниже.

Декодирование данных опроса CCP

Давайте рассмотрим расширенную версию запроса / ответа на опрос CCP из предыдущего времени:

>Trace example: CCP polling via SHORT_UP command (two signals)

<Time	Идентификатор CAN (ШЕСТНАДЦАТЕРИЧНЫЙ)	Байты данных (ШЕСТНАДЦАТЕРИЧНЫЙ)	Отправитель
1.51	701	0F 02 04 00 12 34 56 78	мастер
1.53	702	FF 00 02 F1 2A 71 2F AA	Подчинение
1.55	701	0F 03 04 00 AB CD EF 00	ведущий
1.57	702	FF 00 03 00 37 1A 81 AA	Подчинение

Это отражает проблему декодирования: ECU отправляет разные сигналы с одним и тем же идентификатором CAN - без возможности их идентификации в полезной нагрузке.

Подробное объяснение

В приведенном выше примере мастер запрашивает два разных 4-байтовых значения сигнала от ECU, а именно от источника адреса 0x12345678 и 0xABCD00. В обоих случаях ECU выдает ответ с идентификатором CAN 0x702. В большинстве сценариев декодирования шины CAN вы могли бы просто просмотреть идентификатор CAN ответа и найти "начальный бит" и "разрядность" данного сигнала для извлечения необработанных данных из полезной нагрузки.

Однако здесь это невозможно, поскольку ECU использует один и тот же идентификатор CAN для двух разных сигналов. Другими словами, Идентификатора CAN недостаточно, чтобы отличить сигнал, поступающий от 0x12345678, от сигнала, поступающего от 0xABCD00.

В какой-то степени это похоже на то, как ответы OBD2 от ECU используют один и тот же идентификатор CAN (обычно 0x7E8) для разных сигналов, такие как скорость и обороты в минуту. Однако в случае OBD2 мы можем легко решить эту проблему, потому что OBD2 PID (Идентификатор параметра) включен в полезную нагрузку. Вместе с идентификатором CAN это служит уникальным идентификатором в фрейме ответа, позволяя нам рассматривать ответ как случай мультиплексирования.

Мы не можем сделать это напрямую в опросе CCP, потому что полезная нагрузка ответа не включает адрес источника ECU. В другими словами: В случаях использования с несколькими сигналами опроса CCP нам необходимо объединить информацию из запроса и ответное сообщение.

Проще говоря, мы можем решить эту проблему, "переупаковав" полезные нагрузки следующим образом:

>Trace example: CCP polling - decoding via repackaging of payload

<Время	Идентификатор CAN (ШЕСТНАДЦАТИБИТОВЫЙ)	Байты данных (ШЕСТНАДЦАТИБИТОВЫЙ)	Отправитель
1.51 1.53	701 702	0F 02 04 00 12 34 56 78 FF 00 02 F1 2A 71 2F AA	ведущий ведомый
1.55	701	0F 03 04 00 AB CD EF 00	master
1.57	702	FF 00 03 00 37 1A 81 AA	Подчинение
1.53 1.57	702 702	0F 02 04 00 12 34 56 78 FF ИЗ 03 04 00 AB CD EF 00 FF	00 02 F1 00 03 00

"multiplexor"

Поступая таким образом, мы можем обработать это, используя логику мультиплексирования - аналогично тому, как мы обрабатываем декодирование OBD2 и UDS. Здесь идентификатор CAN для поиска равен 0x702, а мультиплексор находится в 5-8-м байтах, в то время как значение сигнала находится в 12-м до 15-го байта. Теперь можно указать два отдельных правила декодирования, причем соответствующее зависит от значения 4-байтовый мультиплексор.

На практике такая реконструкция фреймов CAN может быть выполнена, например, в скрипте Python, предполагая, что трассировка CAN включает в себя как данные запроса, так и ответа. После восстановления кадров CAN можно использовать файлы DBC для декодирования данных посредством мультиплексирования. В качестве альтернативы, данные могут быть загружены в программное средство, поддерживающее декодирование CCP / XCP напрямую.

Декодирование данных CCP DAQ

Давайте кратко рассмотрим фрагмент трассировки DAQ после инициализации из предыдущего:

>Trace example: CCP DAQ messages

<ВремяИДЕНТИФИКАТОР CAN (ШЕСТНАДЦАТЕРИЧНЫЙ)Байты данных (ШЕСТНАДЦАТЕРИЧНЫЙ)			Отправитель	Тип ф
2.44	712	07 AF 32 4A 00 00 00 F1	Подчинение	DAQ-
2.54	712	07 AB 22 4C 01 00 00 F1	Подчинение	DAQ-
2.64	712	07 AD 24 4B 02 0F 00 F2	Подчинение	DAQ-
2.74	712	07 AB 25 4C 03 0E 00 F1	Подчинение	DAQ-

Как очевидно, сообщения DAQ-DTO имеют предварительно указанный идентификатор CAN (0x712 выше) и полезную нагрузку, в которой 1-й байт равен идентификатору списка ODT, также известному как ODT PID (0x07 выше). В результате нет необходимости "комбинировать" DAQ-DTO кадры ответа для однозначного определения того, какие кадры содержат какие сигналы - это можно определить напрямую с помощью комбинации идентификатора CAN и ODT PID.

Этот факт упрощает декодирование сообщений DAQ-DTO по сравнению с сообщениями опроса. Фактически, вы можете напрямую создать Файл DBC с мультиплексированием при условии, что вам известна кодировка сигнала в оставшихся байтах каждого списка ODT.

Будут ли известны правила декодирования CCP DAQ?

В большинстве практических приложений разумно предположить, что вы будете знать эту кодировку сигнала. Это потому, что сам master управляет (через последовательность инициализации) тем, как упаковать каждый сигнал в списки DAQ и ODT, как объяснялось в предыдущем разделе.

Например, мы знаем, что DAQ-DTO с идентификатором CAN 0x712 и ODT PID 0x07 содержат сигнал с адресом источника 0x00001000 во 2-м байте - потому что именно так мы упаковали его во время инициализации. Из файла описания ECU затем мы можем рассмотреть, как интерпретировать это 1-байтовое значение сигнала. В описании может быть указано, что этот сигнал представляет собой температуру измеряется в градусах Цельсия, и его следует умножить на коэффициент 0,8 и уменьшить на 20. Если вы знакомы с файлами DBC, вы обратите внимание, что такую информацию можно легко ввести в DBC, что позволяет быстро расшифровать данные трассировки в большинстве CAN программные инструменты.

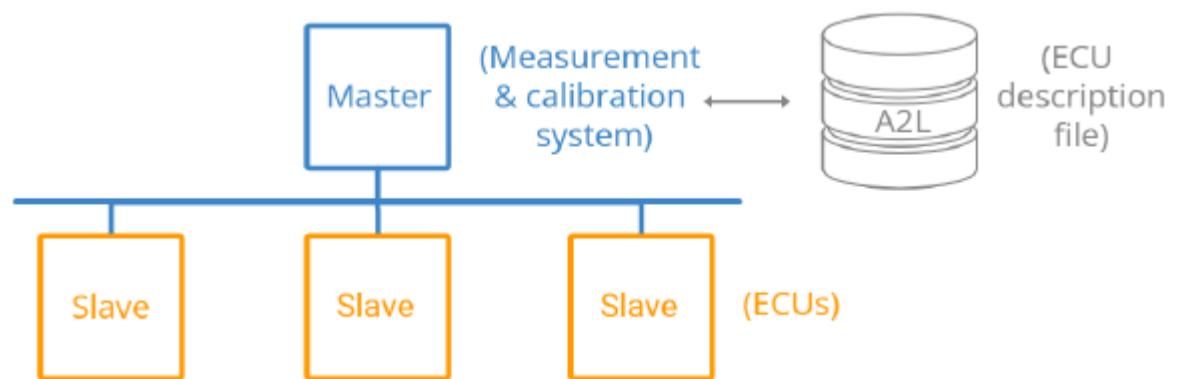
Естественно, если мастер изменит инициализацию, чтобы упаковать данные по-другому при другом тестовом запуске, это будет означать, что файл DBC должен быть соответствующим образом обновлен.

Вышесказанное должно прояснить, что опрос CCP / DAQ может обрабатываться в рамках обычной логики декодирования кадров шины CAN - хотя оба варианта требуют некоторых настроек. Файлы DBC могут использоваться для обработки данных, но этот формат не самый распространенный выбор для работы с данными CCP. Ниже мы рассмотрим другой обычно используемый формат файла, формат A2L или ASAP2.

Файлы описания A2L - ECU

В предыдущих разделах мы иногда ссылались на "Файлы описания ЭБУ". Файл описания ЭБУ содержит все, что необходимо знать мастеру-инструменту для связи с ЭБУ. Из данных с точки зрения измерений, сюда входит информация об идентификаторах CAN, доступных сигналах, декодировании сигналов правила, информация DAQ / ODT и т.д.

На практике используется формат описания ASAP2 (*.A2L) для структурирования этой информации. Данные ASAP2 определение было стандартизировано ASAM в ASAM MCD-2 MC, причем первая версия 1.3.1 была выпущена 15 июня 1999 г., то есть в том же году, когда Выпущен стандарт CCP 2.1. Последняя версия стандарт 1.7.0 был выпущен в 2015 году.



Вы можете просмотреть пример файла A2L здесь.

На практике файл A2L используется для "настройки" главного устройства, обеспечивая связь с блоком управления. Например, в наших предыдущих примерах мы использовали идентификаторы CAN 0x701 и 0x702 для связи CRO / DTO между master и ECU. Эти идентификаторы будут указаны в файле A2L. Файл A2L также описывает, как сигналы хранятся в ECU и как их декодирование. Смотрите, например, приведенный ниже фрагмент из файла примера A2L:



```
/begin MEASUREMENT
/* Name */ Airflow
/* LongIdentifier */ "Final value after Linear Sensor Characterization"
/* Datatype */ FLOAT32_IEEE
/* Conversion */ CompuMethod_6
/* Resolution */ 1
/* Accuracy */ 0
/* LowerLimit */ -1E+17
/* UpperLimit */ 1E+17
ECU_ADDRESS 0x40000144
/end MEASUREMENT
```

Фрагмент A2L объясняется

Как и прежде, мы можем сравнить это с логикой кодирования сигнала в других протоколах CAN и в файлах DBC. Метка "ИЗМЕРЕНИЕ" используется для завершения описания сигнала измерения, в данном случае воздушного потока. Она содержит подробное описание и минимальное / максимальное значение, определяемое нижним / верхним пределом.

Значение сигнала также имеет 4-байтовый адрес источника, 0x40000144. Если мы вспомним схему опроса CCP, этот адрес источника может использоваться в последовательности опроса команды SHORT_UP для запроса необработанного значения для этого конкретного сигнала ECU. Аналогично, в контексте измерения DAQ этот адрес источника будет использоваться в процессе инициализации. Другими словами, инженер может выполнить поиск интересующего сигнала (airflow) в файле A2L или инструменте A2L GUI и настроить свой главный инструмент для запроса этого.

Предполагая, что теперь мы записали трассировку необработанных значений, нам потребуется расшифровать информацию. Как объяснялось ранее, первым шагом было бы извлечь необработанные байты из полезной нагрузки ответа - и способ выполнения этого будет зависеть от используем ли мы опрос или измерение DAQ. Кроме того, порядок байтов не определен протоколом CCP, но будет указан в файле A2L на глобальном уровне или уровне сигнала.

После того, как мы извлекли сигнальные байты и преобразовали их в десятичную форму, нам нужно знать, как их преобразовать.

В контексте файла DBC это всегда будет выполняться в форме "линейного" уравнения, т.е. $y = bx + c$, где b равно масштабу коэффициента и c равно смещению - оба указаны в файле DBC для каждого сигнала.

В отличие от файлов DBC, файлы A2L допускают более сложные методы преобразования. Как видно из описания сигнала, это относится к правилу преобразования, определенному как "CompuMethod_6". Мы можем посмотреть это в другом разделе файла A2L:



```
/begin COMPU_METHOD
/* Name */ CompuMethod_6
/* LongIdentifier */ ""
/* ConversionType */ RAT_FUNC
/* Format */ "%6.2"
/* Unit */ "kg/s"
COEFFS 0 1 0 0 0 1
/end COMPU_METHOD
```

Фрагмент A2L объясняется

Как указано, в этом правиле преобразования используется тип преобразования с именем RAT_FUNC. Это один из нескольких поддерживаемых типов преобразования он определяется следующим образом:

$$y = (axx+bx+c) / (dx+ex+f)$$

Как видно, рациональная функция представляет собой более сложную версию линейной функции, используемой в файлах DBC. В частности, она поддерживает 5 коэффициентов, которые необходимо указать в методе вычисления. Вверху они установлены в 0 1 0 0 0 1, это означает, что функция упрощается до приведенной ниже:

$$y = x = x$$

В этом простом случае рациональная функция упрощается до линейной функции, аналогичной той, которая используется в файлах DBC. Файлы A2L в на самом деле также поддерживают другой тип преобразования, ЛИНЕЙНЫЙ, который просто $y = ax+b$. В случае с сигналом воздушного потока это можно было бы использовать вместо этого. Наконец, единица измерения также указана в деталях измерения, что означает, что теперь мы можем построить график физическое значение расхода воздуха, измеренное в кг / с. Поле формата равно %6.2, которое следует читать как %длины. Макет с Длина, отражающая общую длину декодированного сигнала, и расположение, отражающее количество знаков после запятой.

Практическое использование файлов A2L.

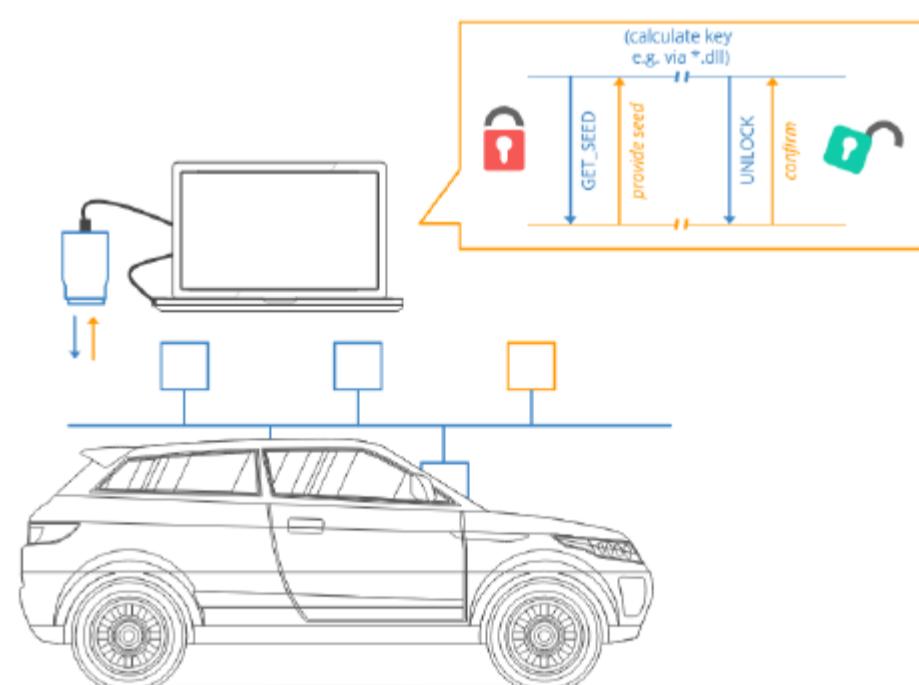
Вероятно, понятно, что обычно файл A2L не просматривается в текстовом редакторе. Скорее, файл A2L просто загружается в совместимый инструмент (например, инструмент с графическим интерфейсом ПК с подключенным интерфейсом CAN или регистратор данных CAN). Затем это позволяет инженер может использовать инструмент для выполнения соответствующих измерений с помощью блока управления без необходимости разбираться в A2L структура файла и синтаксис.

Этот конкретный сигнал достаточно упрощен, чтобы быть описанным в ограничительном формате файла DBC, но файлы A2L явно допускают более сложные правила декодирования. Для получения более подробной информации о формате файла A2L см. Стандарт ASAM MCD-2 MC, а также [практический обзор](#).

Начальная авторизация и авторизация по ключу

Как объяснялось ранее, Связь CCP обычно используется инженерами-изготовителями автомобилей для облегчения связи с ECU в предпроизводственных приложениях.

В некоторых случаях все, что требуется для связи с блоком управления, - это Файл A2L и подходящий инструмент CAN. Как очевидно, можно также просто извлеките подмножество файла A2L для настройки устройства например, для измерения определенных сигналов. Однако без файла A2L практически невозможно выполнить обмен данными и, следовательно, многие OEM-производители не требуют никаких дополнительных мер безопасности, помимо сохранения файла A2L под блокировкой.



Однако некоторые варианты использования требуют дополнительной безопасности. CCP поддерживает это с помощью концепции, называемой авторизацией "начальный код и ключ". Здесь мастер запрашивает случайное начальное значение от ECU как часть инициализации связи ECU. Мастер получает исходное значение от ECU и использует его в качестве входных данных для внутреннего алгоритма безопасности для вычисления ключа. Затем мастер отправляет ключ в ECU через CAN - и если он соответствует ключу, вычисленному внутри ECU, предоставляется авторизация для дальнейшей связи. Ниже приведен пример трассировки для такого сообщения:

>Trace example: CCP seed & key authorization sequence

<Время	ИДЕНТИФИКАТОР CAN (ШЕСТНАДЦАТИБИТНЫЙ)	Байты данных (ШЕСТНАДЦАТИБИТНЫЙ)	Отправитель	Тип фрейма
1.51	701	12 05 02 AA AA AA AA AA AA	мастер	CRO (GET_SEED)
1.53	702	FF 00 05 01 14 15 16 17	Подчинение	CRM (OK + SEED)
1.55	701	13 06 A3 12 FF B0 AA AA	мастер	CRO (РАЗБЛОКИРОВКА + КЛЮЧ)
1.57	702	FF 00 06 02 AA AA AA AA	Подчинение	CRM (OK + МАСКА РЕСУРСОВ)

Трассировка объяснена.

В этом примере мастер запрашивает начальное значение через команду GET_SEED 0x12, указывая "маску ресурса" равной 0x02 в 3-м байте. Мaska ресурса ссылается на запрашиваемый уровень доступа - при этом 0x02 является доступом только DAQ (подробнее см. Стандарт CCP 2.1).

ECU отвечает кадром, в котором 4-й байт равен 0x01, подразумевая, что доступ DAQ защищен (в отличие от 0x00 что подразумевает, что аутентификация не требуется). ECU предоставляет запрошенное начальное значение в оставшихся 4 байтах. На основе начального значения ECU вычисляет ключ и предоставляет его с помощью команды разблокировки 0x13. Ключ находится в данном случае длиной 4 байта. ECU отвечает положительно, включая запрошенную маску ресурса 0x02, что означает, что DAQ доступ теперь разблокирован.

Что касается реализации алгоритма безопасности

Наиболее распространенной реализацией алгоритма безопасности является использование файла *.dll, поскольку основным устройством часто является ПК с Windows с графическим интерфейсом и интерфейсом CAN. Использование файла *.dll также обеспечивает стандартизацию всех инструментов CCP, при этом мастер-инструменту не нужно знать используемый базовый алгоритм. Однако использовать его сложно файлы *.dll в автономных регистраторах данных шины CAN и телематических устройствах (поскольку они не работают под управлением Windows), поэтому некоторые компании используют альтернативные решения - например, проприетарные форматы файлов, которые лучше подходят для встраиваемых устройств.

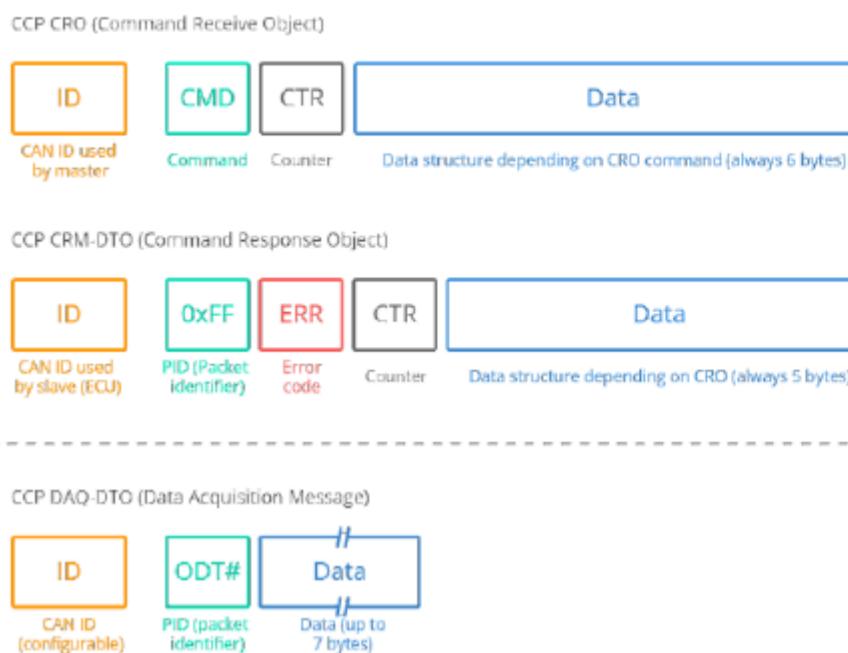
XCP на CAN - основы

Теперь мы рассмотрели основы CCP. Далее, мы рассмотрим XCP на CAN с акцентом на фреймовые структуры.

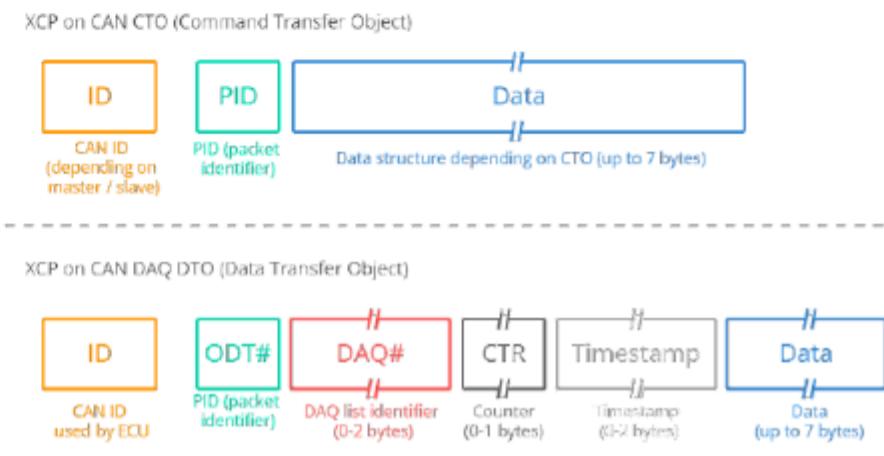
Пакет XCP

Чтобы понять структурные изменения в XCP в сравнении CAN и CCP, рассмотрим приведенное ниже сравнение CCP CRO / DTO и XCP CTO / DTO. Как очевидно, пакеты XCP включают в себя XCP CTO (с аналогичной ролью CCP CRO и CRM-DTO) и XCP DTO (при этом XCP DAQ-DTO играет ту же роль, что и CCP DAQ-DTO).

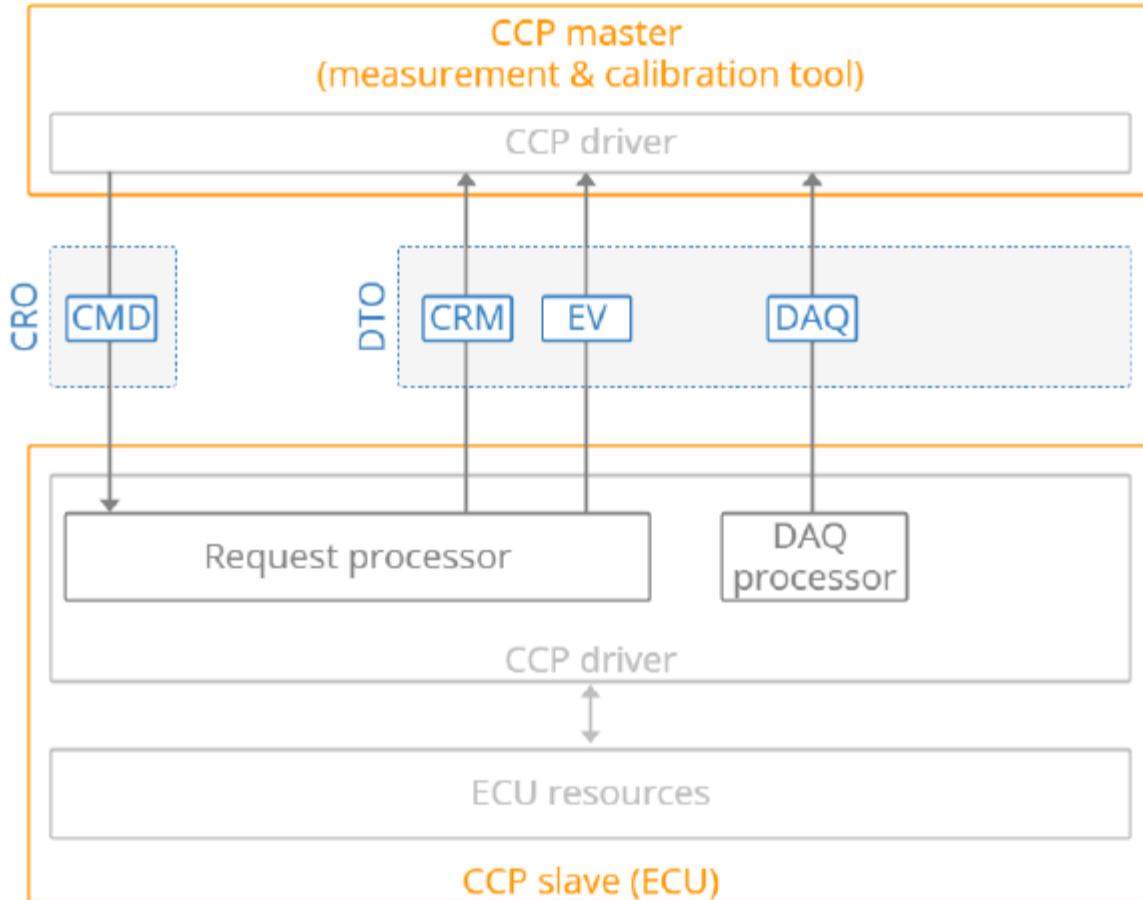
CCP - CRO/DTO frame structures



XCP on CAN - CTO/DTO frame structures



Альтернативный способ проиллюстрировать роль CCP и сообщений XCP - это сравнение архитектуры, приведенное ниже.:



Ниже мы более подробно рассмотрим XCP для типов сообщений CAN.

XCP СТО - объект передачи команд.

В XCP на CAN технический директор - это CAN-фрейм для передачи управления команды, включая команды (CMD), ответы на команды (RES), ошибки (ERR), события (EV) и запросы на обслуживание (SERV). В отличие от CRO в CCP, технический директор используется обоими ведущими и ECU. Обратите также внимание, что на пакет CMD от ведущего устройства должен быть получен ответ пакетом RES или ERR от ECU, в то время как на другие типы пакетов отправляются асинхронно.

XCP на CAN СТО (объекте передачи команд)

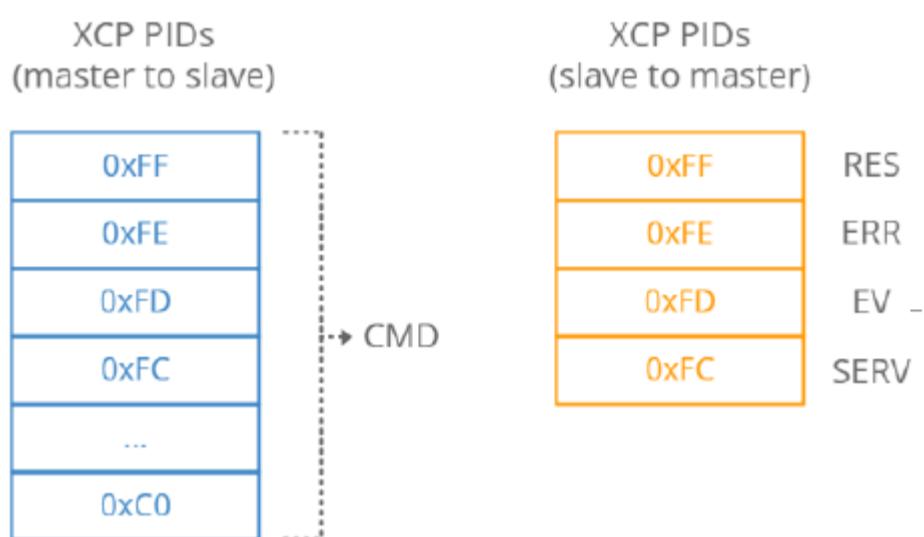


Если мы посмотрим на полезную нагрузку СТО, то 1-й байт отражает идентификатор пакета (PID). Остальные 7 байт в XCP на CAN СТО полезная нагрузка состоит из данных, специфичных для типа пакета СТО. Если мы сравним эту структуру с CRO CCP, она будет аналогичной, за исключением исключения байта счетчика команд в XCP.

Идентификаторы шины CAN в XCP

Для связи по протоколу XCP требуется по крайней мере два идентификатора шины CAN: один для ведущего устройства (например, 0x551) и один для блока управления (например, 0x552). Если ведущему устройству необходимо установить связь через XCP более чем с одним блоком управления, потребуется дополнительный набор идентификаторов.

Фактические значения байта CMD в XCP и CCP также различаются. Например, CCP использует 0x01 для команды CONNECT, в то время как XCP использует 0xFF. Смотрите также обзор таблицы для назначения PID CTO в зависимости от того, сообщения отправляются от ведущего или ведомого устройства.



Смотрите также сравнение кодов CCP и XCP CMD ниже.:

Выберите значения CCP и XCP CMD.

В таблице ниже сравнивается подмножество кодов команд между CCP и XCP:

>CCP vs. XCP command code comparison (examples)

<CCP CMD	XCP CMD	Команда
0x01	0xFF	Контакты
0x12	0xF8	GET_SEED
0x13	0xF7	РАЗБЛОКИРОВАТЬ
0x02	0xF6	SET_MTA
0x04	0xF5	загрузка
0x0F	0xF4	SHORT_UP
0x15	0xE2	SET_DAQ_PTR
0x16	0xE1	WRITE_DAQ
0x07	0xFE	ОТКЛЮЧЕНИЕ
0x0E	0xF3	BUILD_CHECKSUM

Пример: сообщение о подключении технического директора XCP

Как и в CCP, давайте посмотрим, как может выглядеть последовательность подключения в XCP для CAN:

>Trace example: XCP on CAN initialization (CONNECT CMD + ECU response)

<Время	ИДЕНТИФИКАТОР CAN (шестнадцатеричный)	Байты данных (ШЕСТНАДЦАТЕРИЧНЫЙ)	Отправитель	Тип фрейма
1.01	701	FF 00	мастер	ТЕХНИЧЕСКИЙ директор (ПОДКЛЮЧЕНИЕ)
1.04	702	ФФ 04 80 08 08 00 01 01	Подчинение	Технический директор (OK + ИНФОРМАЦИЯ)

Трассировка объяснена

Обратите внимание, что в примере не используется заполнение 0xAA, поскольку это необязательно в XCP на CAN.

При трассировке мастер отправляет команду CONNECT CMD (0xFF) в блок управления с установленным режимом связи normal (0x00). Обратите внимание, что, в отличие от CCP, ведущему устройству не нужно указывать адрес станции в полезной нагрузке СОЕДИНЕНИЯ фрейм. Это связано с тем, что идентификаторы CAN уже однозначно определяют, с каким ECU связывается ведущее устройство.

ECU отвечает кадром CAN, в котором 1-й байт равен PID положительного ответа (RES). 2-й байт - это доступность ресурсов, аналогичная связи CCP с исходным кодом и ключом с 0x04, например, означающая, что DAQ доступен. В 3-й байт относится к "режиму связи" (здесь "необязательно"), а 4-й байт - это максимальный размер СТО (здесь 8 байт). 5-й и 6-й байты равны максимальному размеру DTO (здесь 8 байт), в то время как 7-й и 8-й байты равны XCP версия уровня протокола и версия транспортного уровня соответственно (в данном случае оба по 1).

Пример: Опрос XCP

После установления соединения ведущий может, например, инициировать опрос. Как и в CCP, это можно сделать с помощью SHORT_UPLOAD команда:

>Trace example: XCP on CAN polling via SHORT_UPLOAD command

<ВремяИдентификатор CAN (ШЕСТЬНАДЦАТЕРИЧНЫЙ)	Байты данных (ШЕСТЬНАДЦАТЕРИЧНЫЙ)	Отправитель	Тип фрейма
1.51 551	F4 02 00 00 78 56 34 12	мастер	ТЕХНИЧЕСКИЙ директор (SHORT_UPLOAD)
1.53 552	FF F1 2A	Подчинение	Технический директор (OK + ДАННЫЕ)

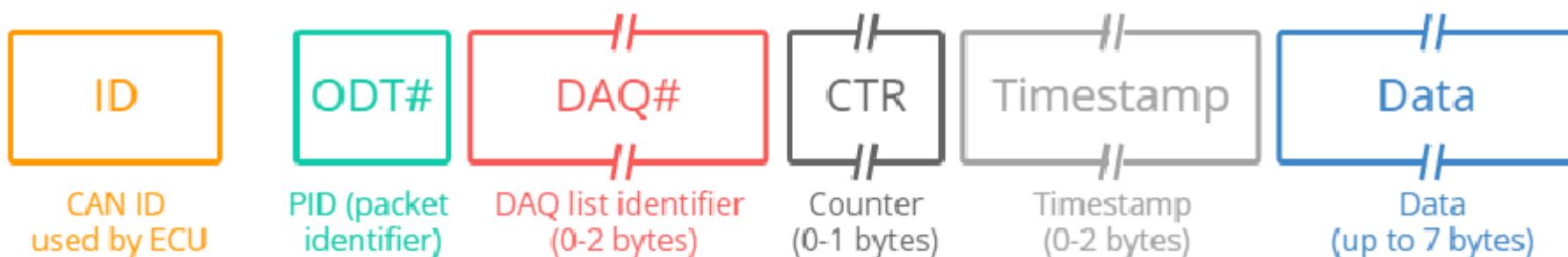
Объяснена трассировка

Здесь мастер отправляет команду SHORT_UPLOAD CMD (PID 0xF4) для 2 байт данных. 3-й байт зарезервирован, а 4-й байт СТО - это расширение адреса (в данном случае 0). Оставшиеся 4 байта равны адресу источника 0x12345678 (Порядок байтов INTEL). В ответ на команду СТО блок ECU отправляет ответный СТО со значением в 2 байта данные.

XCP DTO: объект передачи данных

XCP DTO используется для отправки синхронных данных. В частности, DTO используется при измерении DAQ (аналогично роли DAQ-DTO в CCP). В XCP это также позволяет передавать данные "стимуляции" (STIM), которые могут использоваться в обход обычного алгоритма в ECU. Стимуляция и обход - это темы, которые мы здесь не будем рассматривать.

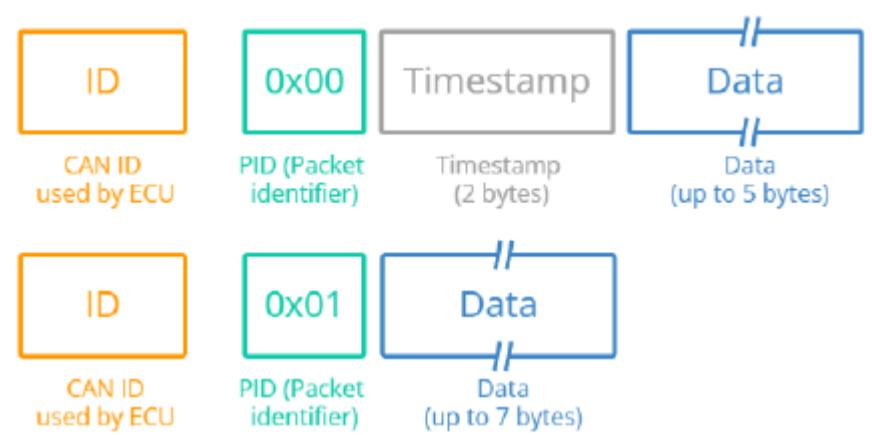
XCP на CAN DAQ DTO (объекте передачи данных)



Мы также не будем вдаваться в подробности измерения XCP DAQ, поскольку по концепции оно аналогично CCP DAQ.

Временная метка XCP DTO

Однако следует отметить одну важную вещь, касающуюся измерения XCP DAQ, это то, что он позволяет ECU записывать время измерения в XCP Пакеты DTO. Подразумевается, что мастер способен корректно синхронизировать данные ECU, разделенные по нескольким кадрам, используя ECU временную метку измерения, а не собственная внутренняя система мастера временная метка. Временная метка XCP DTO является необязательной и реализована как увеличивающийся счетчик с логикой увеличения, указанной в файле A2L. Если временная метка должна быть включена в определенный список DAQ, она будет записана в первый список ODT (но не в последующие, если их больше Списки ODT существуют в одном списке DAQ).



Идентификация DAQ / ODT

Также стоит отметить, что существует несколько методов для упаковки информации DAQ list # и ODT list # в DAQ DTO. Простейшим случаем являются "абсолютные числа ODT", где список ODT # упакован во всех списках DAQ. Здесь PID (т.е. ODT #) достаточно для глобальной идентификации списка ODT. Однако существует другой метод, при котором относительный список ODT номера используются во всех списках DAQ. Таким образом, вы могли бы, например, иметь два списка ODT с PID = 0x00 что означает, что они не могут быть однозначно идентифицирован, поскольку идентификатор CAN идентичен для двух DTO. Здесь может быть указан идентификатор списка DAQ в 1 или 2 байта добавлен во 2-3 байта полезной нагрузки кадра CAN. Благодаря этому снова можно однозначно идентифицировать конкретный список ODT. Файл A2L и ECU предоставляют информацию о том, какой метод используется.

Необязательное поле счетчика XCP DTO

Поле счетчика в 1 байт может необязательно использоваться при передаче XCP DTO. Если поле CTR используется в XCP DTO DAQ пакетов, подчиненное устройство вставит значение CTR только в 1-й кадр списка DAQ, то есть в 1-й список ODT.

Ссылки для дальнейшего чтения на CCP / XCP

CCP / XCP, конечно, содержат множество других тем, на обсуждение которых мы здесь не потратили время. Мы намеренно сосредоточились на основных темах и концепциях, связанных с регистрацией данных. Однако ниже вы найдете полезные ссылки для дальнейшего чтения.

Vector's XCP Book v1.5: подробный обзор протокола XCP 1.5.

XCP (Википедия): Основы протокола XCP

Обзор ASAM MCD-1 XCP: также предоставляет подробный обзор протокола XCP

Обзор ASAM ASAM MCD-2 MC: содержит обзор стандарта файла описания ECU

Начальный уровень Основные сведения (вектор): полезное введение в основы авторизации.

Использование CANedge для Сбора данных CCP/XCP

CANedge представляет собой серию недорогих компактных 2-х шинных регистраторов данных CAN/LIN . CANedge обычно используется производителями автомобильной техники - и по этой причине часто задаваемый вопрос заключается в том, поддерживает ли устройство CCP / XCP на CAN Информационные материалы. Ответ на этот вопрос "да, в зависимости от



вашего варианта использования". Подробности смотрите ниже. Если вы хотите использовать CANedge для CCP / XCP на CAN, свяжитесь с нами.

CANedge + CCP / XCP

Как мы видели в предыдущих разделах, все, что требуется для облегчения опроса CCP / XCP и измерения CCP / XCP DAQ, - это способен передавать определенную последовательность кадров CAN на основе информации, которая может быть идентифицирована из файла A2L. Если вы знаете, как создавать фреймы запроса, вы можете использовать функциональность CANedge transmit для настройки список одиночных или периодических пользовательских CAN-кадров, например, для инициализации CCP / XCP и опроса CCP / XCP / DAQ измерение. Записанные файлы журнала MF4 с данными измерений затем могут быть обработаны, например, в Vector tools или с помощью наших бесплатных программных решений с открытым исходным кодом. В первом случае вы могли бы, например, использовать наши конвертеры MF4 для преобразования данных в Векторные инструменты и использовать файлы A2L для декодирования данных. В последнем случае вы могли бы обрабатывать данные через наш Python API (для опроса) и / или через asammfdf. В настоящее время для этих инструментов потребуется создать файл DBC с правилами декодирования, как объяснялось ранее.

Другими словами, вы можете использовать CANedge для измерения данных сигнала через CCP / XCP на CAN в автономных полевых развертываниях, который предлагает недорогой метод сбора этой информации в масштабе. В отличие, автономные регистраторы данных шины CAN не идеально подходят для калибровки CCP / XCP, за исключением очень специфических случаев использования, таких как удаленная калибровка переменных с помощью обновлений по воздуху.

Начальная аутентификация CANedge и проверка подлинности по ключу

В настоящее время CANedge не поддерживает начальную аутентификацию и проверку подлинности по ключу, поэтому OEM-производители намерены использовать ее для CCP для связи потребуется временно отключить аутентификацию или разблокировать ее с помощью отдельного устройства. Мы, однако, возможно, позже добавим поддержку для этого. Если у вас есть варианты использования, которые включают эту функциональность, не стесняйтесь свяжитесь с нами для дальнейшего обсуждения.

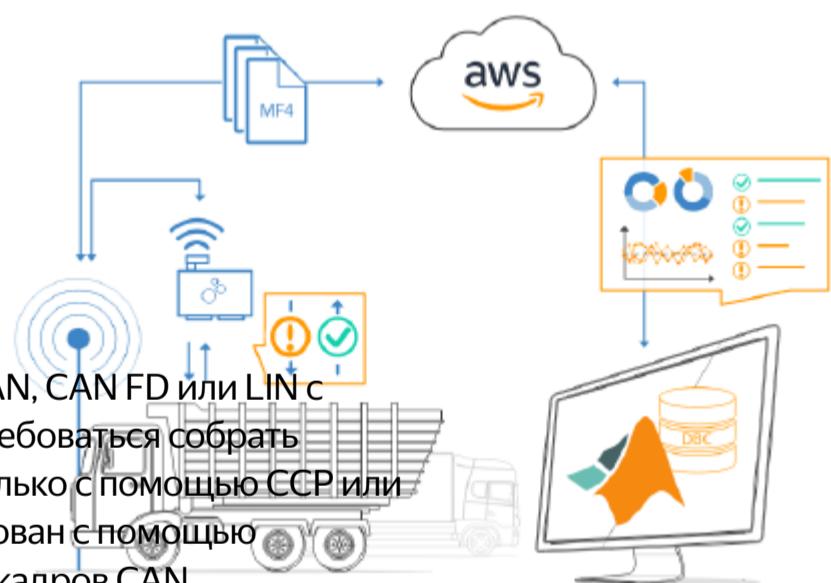
Регистрация данных CCP / XCP - приложения

В этом разделе мы приводим краткие примеры того, как регистрация данных CCP / XCP может быть полезна на практике.

Телематика CCP / XCP для опытного парка транспортных средств.

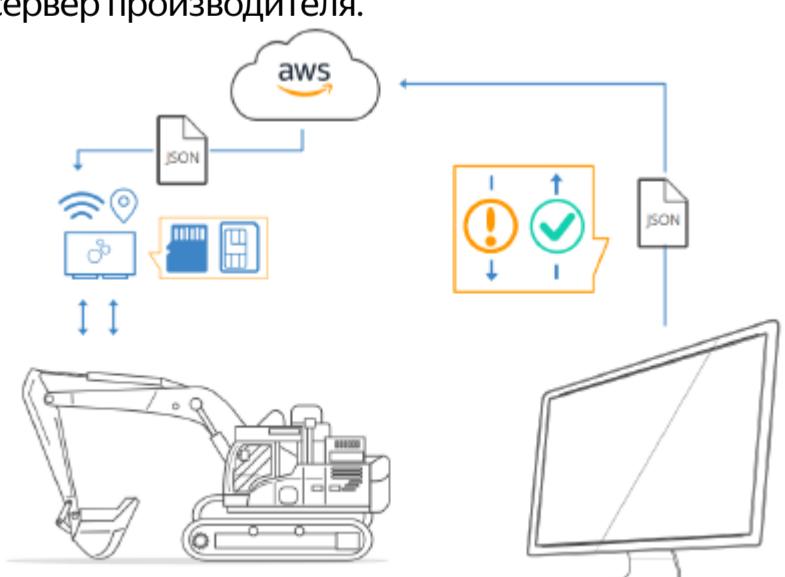
Как производителю, вам может потребоваться собрать, например, данные шины CAN, CAN FD или LIN с прототипов транспортных средств в полевых условиях. Кроме того, вам может потребоваться собрать конкретные данные данные, внутренние для ECU, которые могут быть измерены только с помощью CCP или XCP на CAN. Здесь регистратор CAN, такой как CANedge, может быть сконфигурирован с помощью настраиваемый "список передач", состоящий из одиночных и / или периодических кадров CAN.

При установке в транспортном средстве CANedge автоматически отключается передавайте заранее определенные кадры, позволяя ему, например, выполнять CCP / XCP опрос данных ECU или инициализировать измерение DAQ. Собранные данные CCP / XCP могут быть объединены с общими данными CAN в файлах журнала MF4 и отправлены через Wi-Fi / 3G / 4G на Собственный облачный сервер производителя.



Удаленная калибровка CCP / XCP с помощью OTA-обновлений

В качестве более сложного варианта использования OEM-производитель может использовать CANedge2 для выполнения обновлений ECU по воздуху. Например, CANedge2 может быть установлен в прототипе транспортного средства, в котором инженер-изготовитель хотел бы повторно откалибровать определенные переменные в ECU. Для достижения этой цели инженер создает соответствующую последовательность одноразовых кадров CAN в новом файле конфигурации CANedge. Затем инженер развертывает новый



файл конфигурации на сервере CANedge2 S3, что запускает обновление OTA, позволяя CANedge2 передавать последовательность в автомобиль удаленно.

Благодарим вас за прочтение нашего руководства - мы надеемся, что оно было для вас полезным!

CSS Electronics | DK36711949 | Сорен Фрикс Вей, 38K, 8230 Абихой, Дания

www.csselectronics.com | свяжитесь с @csselectronics.<url> | +45 91 25 25 63 | LinkedIn

[Продукты](#) | [Программное обеспечение](#) | [Руководства](#) | [Тематические исследования](#) | 5 min CANedge вступление

