Модуль CAN
в микроконтроллерах PIC18CXX8

Статья основывается на технической документации DS39500a
компании Microchip Technology Incorporated, USA.

Москва - 2001

тел. (095) 737-7545
www.microchip.ru

Модуль CAN в микроконтроллерах PIC18CXX8

Статья основывается на технической документации DS39500a
компании Microchip Technology Incorporated, USA.

В статье описывается модуль CAN микроконтроллеров: PIC18C658; PIC18C858.

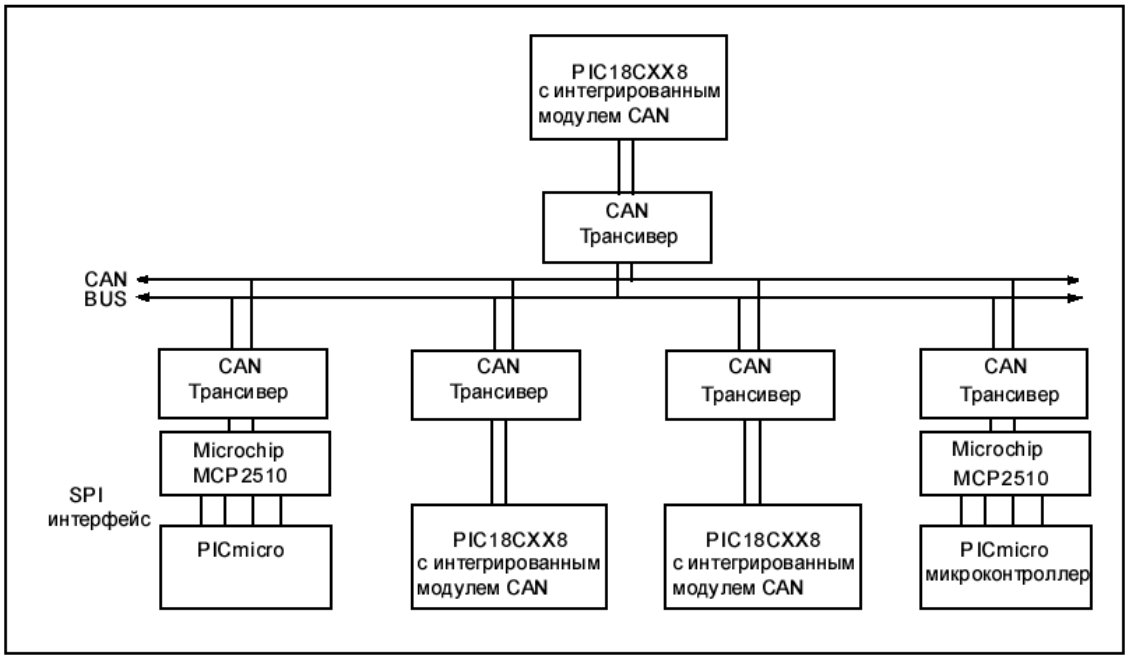
СОДЕРЖАНИЕ

1. Введение

CAN последовательный интерфейс связи, который эффективно поддерживает распределенное управление в реальном масштабе времени с высокой помехозащищенностью. Протокол связи полностью определен Robert Bosch GmbH, в спецификации требований CAN 2.0B от 1991 года.

Область применения CAN протокола: от высокоскоростных сетей связи до замены жгутов электропроводов в автомобиле. Высокая скорость передачи данных (до 1Мбит/с), хорошая помехозащищенность протокола, защита от неисправности узлов – делают шину CAN, подходящей для индустриальных приложений управления типа DeviceNet.

CAN имеет асинхронную последовательную структуру шины с одним логическим сегментом сети. CAN сеть может состоять из двух или более узлов, с возможностью подключения/отключения узлов от шины без перенастройки других устройств.



Логика шины работает по механизму «монтажное И», в котором ‘recessive’ бит соответствует логической единицы, а ‘dominant’ логическому нулю. Пока ни один узел не формирует ‘dominant’ бит, шина находится в ‘recessive’ состоянии, но ‘dominant’ бит от любого узла создает ‘dominant’ состояние шины. Поэтому при выборе среды передачи данных, необходимо определить какое состояние будет ‘dominant’, а какое – ‘recessive’. Одним из наиболее распространенных и дешевых вариантов является пара скрученных проводов. Линии шины тогда называются CANH и CANL и могут быть подключены непосредственно к устройствам. Не существует никакого дополнительного стандарта на среду передачи данных.

При использовании, в качестве линии связи, пары скрученных проводов с нагрузочными резисторами на концах, можно получить максимальную скорость передачи данных 1Мбит/с, при длине линии около 40м. Для линий связи протяженностью более 40 метров необходимо снизить скорость передачи данных (для линии 1000м скорость шины должна быть не более 40кбит/с). Из-за дифференциального характера линии связи, шина CAN мало чувствительна к электромагнитным помехам. Экранирование шины значительно снизит воздействие внешнего электромагнитного поля, что особенно важно для высокоскоростных режимов работы.

Двоичная информация кодируется NRZ кодом (низкий уровень – ‘dominant’, высокий уровень ‘recessive’). Для гарантированной синхронизации данных всеми узлами шины используется наполняющий бит. При последовательной передаче пяти бит одинаковой полярности, передатчик вставляет один дополнительный бит противоположной полярности перед передачей остальных битов. Приемник также проверяет полярность и удаляет дополнительные биты.

В CAN протоколе при передаче данных приемные узлы не адресуются, а указывается идентификатор передатчика. С помощью идентификатора указывается содержание сообщения (например - обороты, температура двигателя и т.д.). Идентификатор дополнительно указывает приоритет сообщения. Меньшее бинарное значение идентификатора – более высокий приоритет.

При коллективном доступе к шине используется неразрушающий арбитраж с опросом состояния шины. Перед началом передачи данных узел проверяет состояние шины (отсутствие активности на шине). При начале передаче сообщения узел становится управляющим шины, все остальные узлы переходят в режим приема. Каждый узел выдает подтверждение приема, проверяет идентификатор сообщения, обрабатывает или удаляет принятые данные. Если два или более узлов начинают передачу данных одновременно, поразрядный арбитраж позволяет избежать конфликта на шине. Каждый узел выдает на шину свой идентификатор (старший бит формируется первым) и контролирует ее состояние. Если узел посылает ‘recessive’ бит, а читает ‘dominant’, значит арбитраж потерян и узел переключается в режим приема. Это происходит тогда, когда идентификатор конкурирующего узла имеет меньшее бинарное значение. Таким образом, узел с высоким приоритетом выигрывает арбитраж, без необходимости повторять сообщение. Все остальные узлы будут пытаться передать сообщение после освобождения шины. Данный механизм

не позволяет передавать одновременно сообщения с одинаковым идентификатором, поскольку ошибки могут возникнуть позже.

Оригинальная спецификация CAN (Версии 1.0, 1.2 и 2.0A) определяет длину идентификатора 11 бит (2048 возможных вариантов). Затем, технические требования были изменены, чтобы уйти от указанного ограничения, и в версии CAN 2.0B длина идентификатора может быть 11 или 29 бит (536 миллионов вариантов). Версия CAN 2.0B, в других документах может называться расширенная версия CAN.

1.1 Стандартная и расширенная версия CAN

Сообщения, содержащие 11-разрядный идентификатор, называются стандартными и соответствуют спецификации технических требований к CAN 2.0A. Возможно использовать всего 2048 различных идентификаторов (0 - 2047), из которых 16 идентификаторов с наименьшим приоритетом зарезервированы (2032 – 2047).

В расширенной версии структура сообщения, согласно техническим требованиям CAN 2.0B, может иметь 29 разрядный идентификатор. 29 разрядный идентификатор состоит из 11 бит стандартного и дополнительных 18 бит расширенного идентификатора.

CAN модули версии 2.0A могут только передавать или принимать данные. Версия 2.0B позволяет активным узлам сделать удаленный запрос и получить данные от нужного устройства, используя расширенное сообщение.

1.2 Модуль CAN с основными функциями и полнофункциональный CAN модуль

Есть еще одна характеристика CAN модуля, которая характеризует интерфейс между CAN и MCU.

В CAN модуле с основными функциями аппаратно реализуются функции приема передачи и поразрядная проверка потока данных. На программном уровне выполняется фильтрация и проверка сообщений, управление передачей данных и многое другое. Нагрузка на MCU в таких устройствах значительна, они могут использоваться при низких скоростях передачи данных и малой информационной нагрузки (присутствуют несколько типов сообщений в сети). Преимуществами устройств с основными функциями CAN модуля является низкая стоимость.

Полнофункциональный модуль CAN аппаратно поддерживает протокол шины, включая фильтрацию принятия и управление передачей сообщений. CAN модули данного типа при работе имеют несколько целей по приему/передаче данных в стандартном или расширенном формате. На этапе инициализации определяются цели работы CAN модуля, таким образом, нагрузка на MCU сокращена. Полнофункциональные модули CAN могут использоваться при высоких скоростях передачи данных и большом информационном потоке.

1.3 ISO модель

ISO/OSI эталонная модель используется для того, чтобы определить уровни протокола системы связи. На самом верхнем уровне модели находятся приложения, которые должны связываться друг с другом. На самом низком уровне модели, расположена физическая среда, обеспечивающая электрическую сигнализацию связи.

Верхние уровни сетевой модели CAN реализуются программно. В технических требованиях к CAN протоколу нет никаких дополнительных требований к содержанию сообщений.

Полнофункциональный CAN модуль поддерживает два нижних уровня эталонной модели сети:

- Управление приемом/передачей данных
  - управление логической связью (LCC подуровень);
  - среднее управление доступом (MAC подуровень).
- Физический уровень
  - физическая сигнализация (PLS подуровень).

LCC подуровень обеспечивает:

- фильтрацию сообщений;
- уведомление о перегрузке;
- управление обнаружением ошибок.

MAC подуровень обеспечивает управление средой передачи:

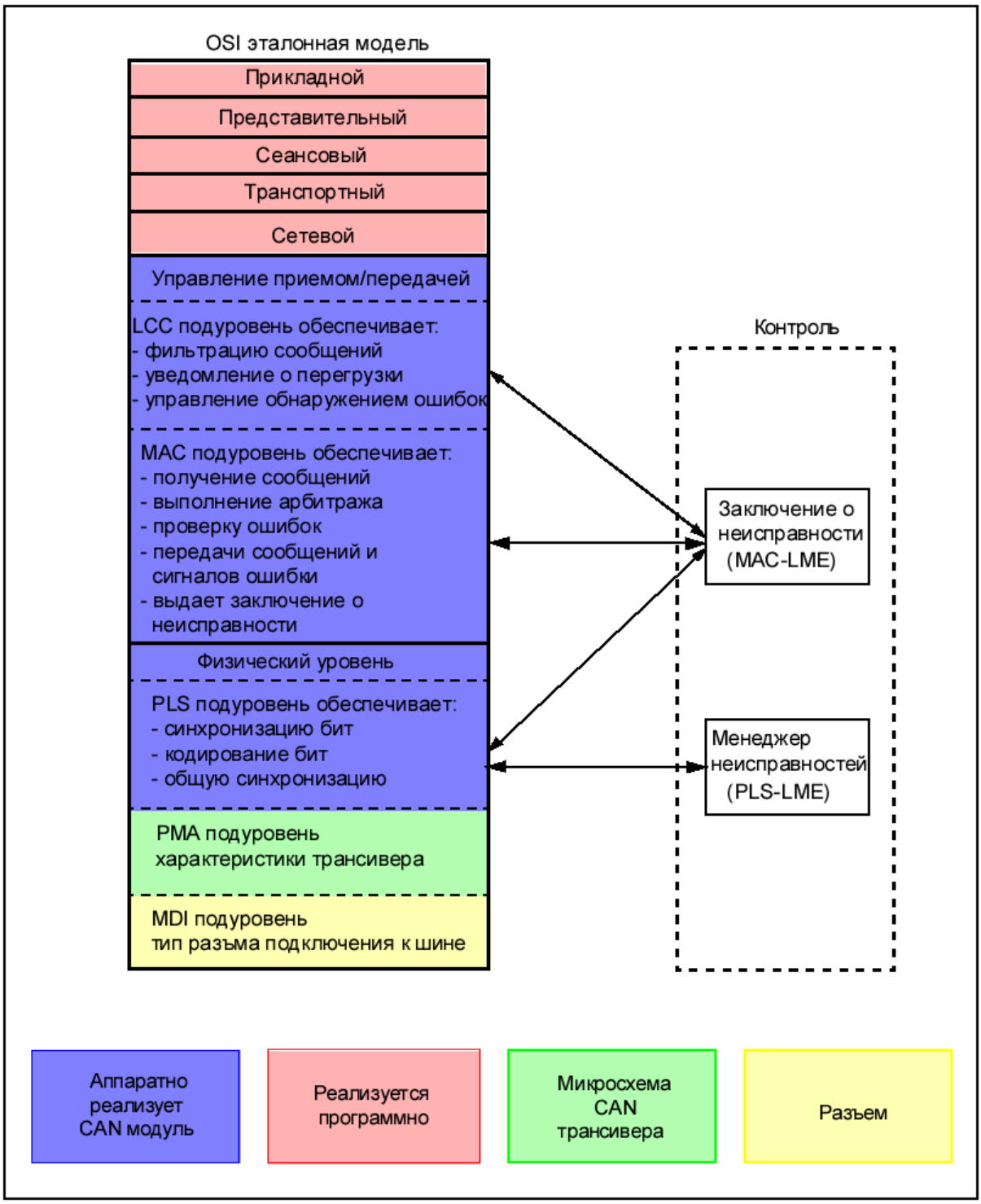
- получение сообщений;
- выполнение арбитража;
- проверка ошибок;
- передача сообщений и сигналов ошибки;
- выдает заключение о неисправности.

MAC подуровень получает сообщения от LCC для передачи по шине и передает сообщения, принятые с шины в LCC подуровень. В пределах подуровня MAC определяется отсутствие активности на шине CAN для начала передачи сообщения. Заключение о неисправности является механизмом самоконтроля при постоянном возникновении кратковременных ошибок.

Физический уровень определяет фактическую передачу бит между различными устройствами с выполнением всех электрических требований. PLS подуровень определяет физическую структуру сигнала и контролирует:

- синхронизацию бит;
- кодирование бит;
- общую синхронизацию.

Низшие уровни протокола определяют фактический тип интерфейса связи: витая пара, волоконный световод и т.д. В пределах одной сети физический уровень должен быть одинаков для всех узлов. Характеристики драйвера физического уровня не определены в спецификации на CAN протокол, чтобы позволить оптимизировать среду передачи для конкретных приложений.



ISO/OSI эталонная модель CAN шины

1.4 Характеристики CAN модуля

CAN модуль – контроллер связи, поддерживающий протокол CAN 2.0A/B, описанный в технической спецификации фирмы BOSCH. Полнофункциональный модуль CAN аппаратно поддерживает протоколы CAN 1.2, CAN 2.0A активную и пассивную версию CAN 2.0B.

Дополнительные характеристики CAN модуля:

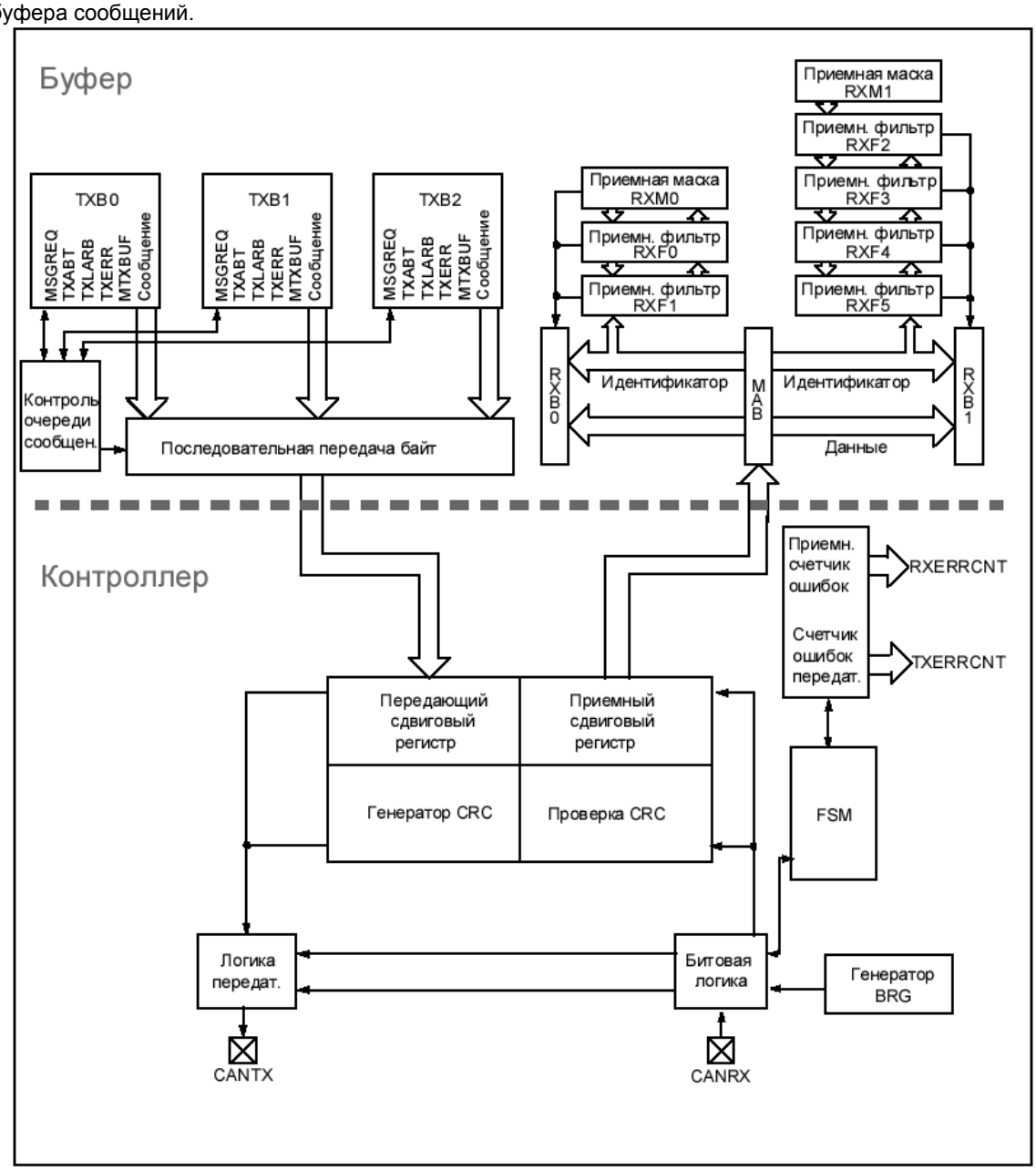
- стандартный и расширенный тип сообщений;
- длина данных в сообщении от 0 до 8 байт;
- программируемая скорость передачи информации до 1Мбит/с;
- поддержка удаленного запроса данных;
- два буфера приемника с двойной буферизацией данных;
- 6 полных приемных фильтров (стандартный/расширенный идентификатор), 2 фильтра связаны с сообщениями высокого приоритета, 4 с сообщениями низкого приоритета;
- 2 полных маски принимаемых сообщений;
- 3 передающих буфера, с возможностью указания приоритета и аварийного прекращения передачи;
- программируемая возможность пробуждения из SLEEP режима со встроенным НЧ фильтром;
- программируемый генератор тактовых импульсов;
- программируемый контроль шлейфа, синхронизирующий функцию самоконтроля;
- гибкая система прерываний от CAN модуля;
- низкое энергопотребление в SLEEP режиме.

2. Работа CAN модуля

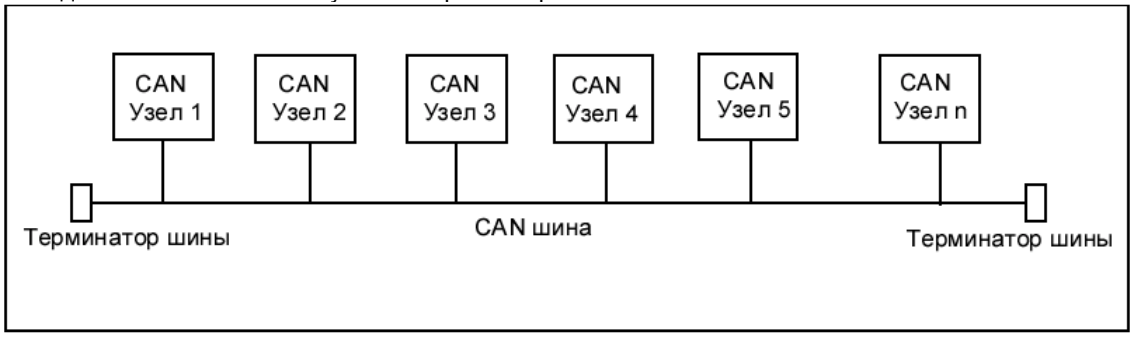
В этом разделе будет рассмотрена работа CAN модуля и поддерживаемые форматы сообщений.

2.1 Краткий обзор CAN модуля

CAN модуль укрупнено состоит из:
- контроллера протокола;
- буфера сообщений.



Типовое подключение к CAN шине узлов и терминаторов



2.2 Контроллер CAN протокола

Контроллер протокола состоит из нескольких функциональных блоков, показанных на рисунке.

Основной частью контроллера является управляющий блок (FSM). Это последовательный конечный автомат, малого разрешения, с изменением управляющих сигналов при различных типах сообщений и состояния передачи/приема данных. FSM имеет следующие функции:

- управление последовательным потоком данных между TX/RX и сдвиговым регистром;
- вычисление и контроль CRC;
- управление состоянием шины;
- обработка ошибок (EML);
- управление параллельным потоком данных между сдвиговым регистром и буфером;
- выполнение арбитража;
- сигнализация об ошибках согласно CAN протоколу;
- автоматическая повторная передача сообщений.

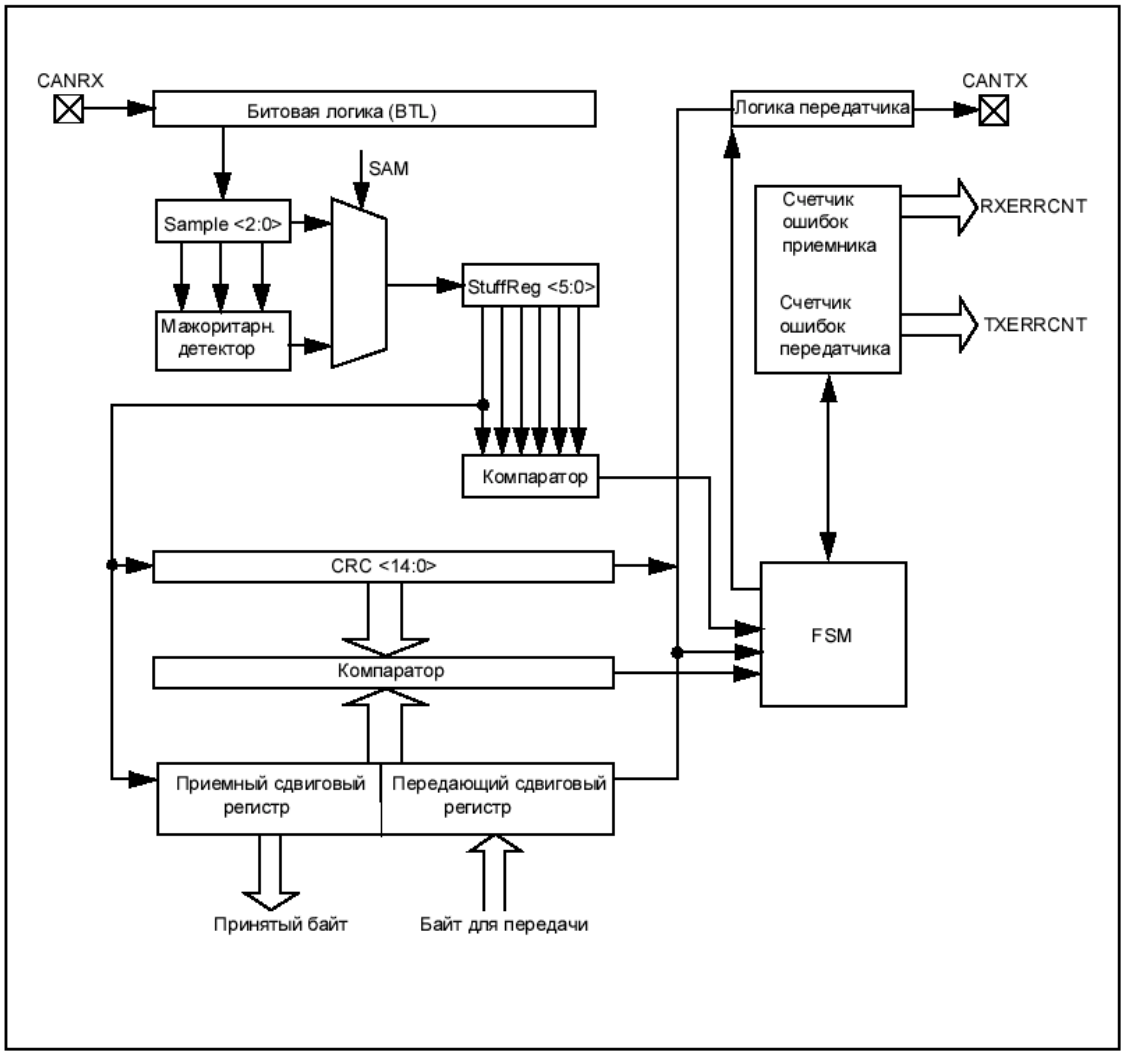
Интерфейс передачи данных от контроллера к буферу – параллельный 8-разрядный. Сообщение разбивается на байты и побайтно загружается/читается в/из сдвиговый регистр для передачи/приема данных. FSM контролирует какая часть сообщения принята/передана.

Регистр циклического контроля избыточности (CRC) формирует CRC код, который побайтно будет передан при передаче данных, и проверяет код CRC при получении сообщения.

Логика управления обработкой ошибок (EML) формирует сигнал о неисправности CAN устройства. Принимающий и передающий счетчики ошибок увеличиваются и уменьшаются от разрядного процессора потока. Согласно значениям счетчиков ошибок CAN модуль может находиться в состоянии:

- активной ошибки
- пассивной ошибки
- отключен от шины

С помощью BTL осуществляется контроль линейного входа шины, обрабатывается состояние шины и выполняется синхронизация бита согласно CAN протоколу. BTL синхронизируется при переходе от ‘recessive’ к ‘dominant’ биту. BTL также обеспечивает программируемые сегменты времени выборки элемента для компенсации задержки времени распространения и смещения фазы. Программирование BTL зависит от скорости передачи данных и внешнего физического времени запаздывания.



2.2.1 Функциональные возможности контроллера CAN модуля

Контроллер в CAN модуле выполняет все функции для приема и передачи сообщений на шине. Данные, предназначенные для передачи, записываются в соответствующие регистры. Состояние CAN модуля и возникшие ошибки проверяются чтением регистров статуса. Любое сообщение, передаваемое по шине, проверяется на наличие ошибок, согласуется по фильтрам и сохраняется в регистрах приемного буфера при выполнении всех условий.

CAN модуль поддерживает следующие типы сообщений:

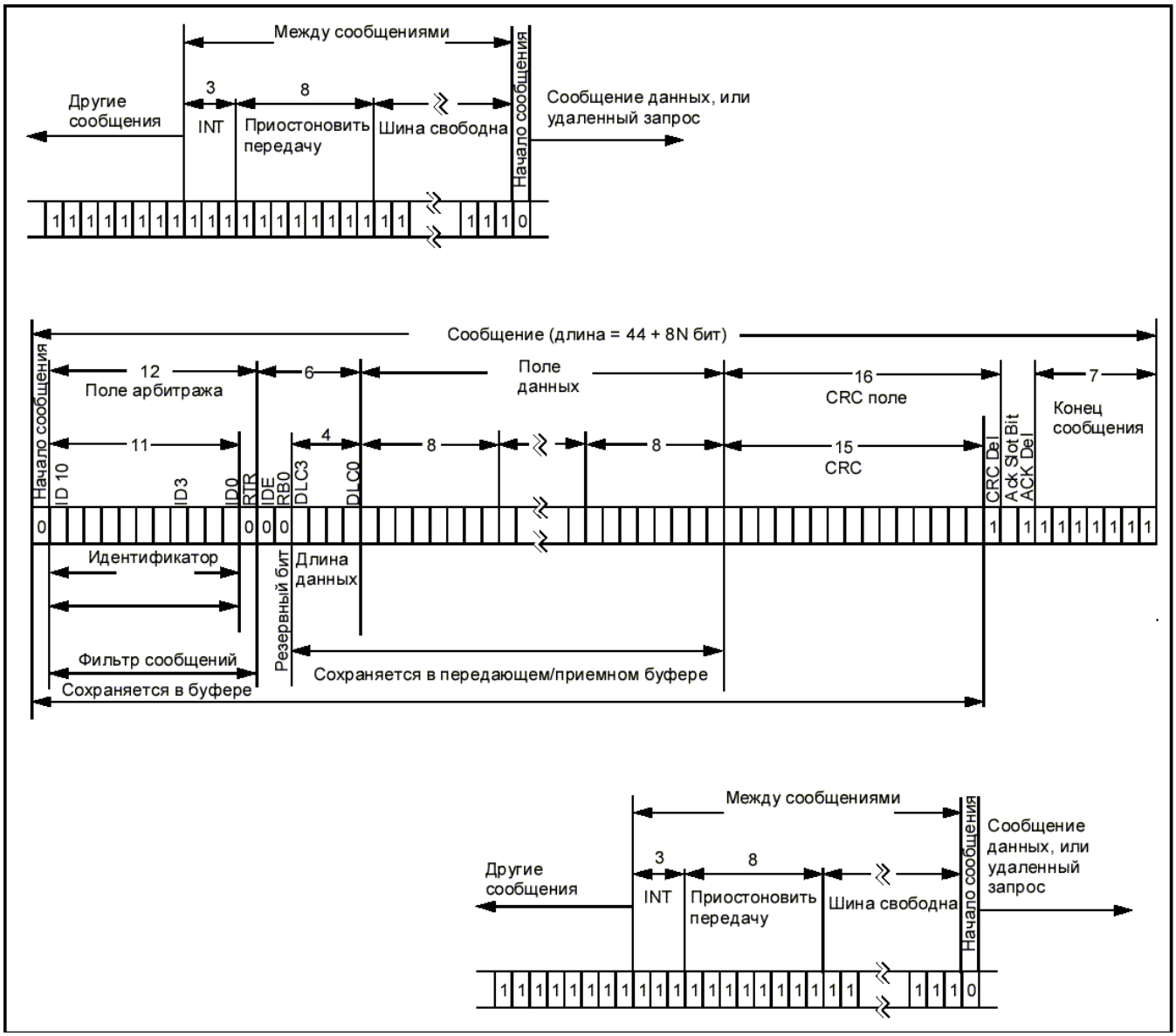
- стандартное сообщение;
- расширенное сообщение;
- удаленный запрос данных;
- ошибка;
- перезагрузка;
- простой шины.

В следующем разделе будет описана структура всех типов сообщений.

3. Структура сообщений

3.1 Стандартное сообщение

Стандартное сообщение формируется, когда узел желает передать данные. На рисунке представлена структура стандартного сообщения.

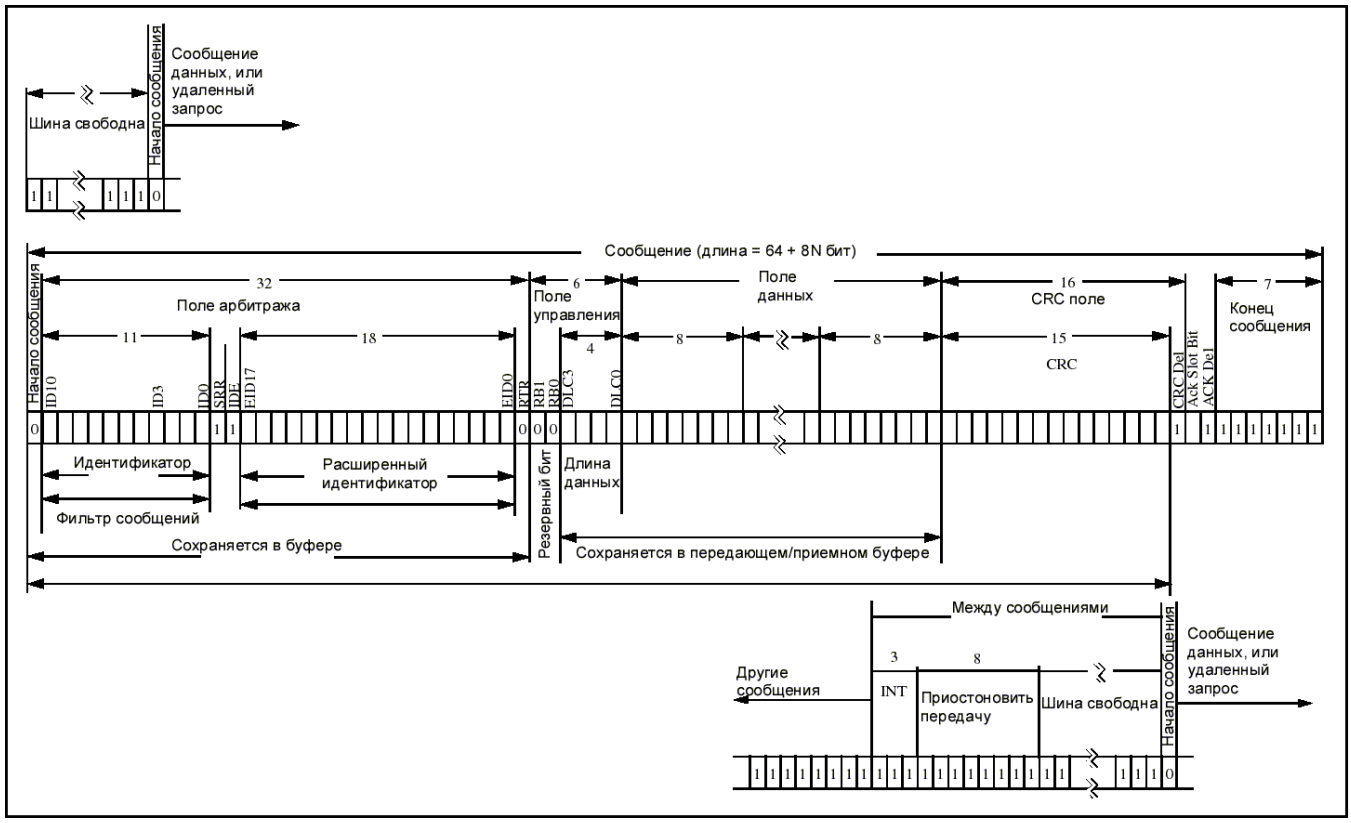


Стандартное сообщение состоит из полей:

- ‘dominant’ бита начала сообщения для жесткой синхронизации всех узлов;
- поле арбитража 12 бит: 11 бит идентификатора и RTR бит передачи по удаленному запросу;
- поле управления 6 бит: 1 бит IDE указатель расширенного идентификатора (состояние ‘dominant’ определяет стандартное сообщение); RB0 резервный ‘dominant’ бит; 4 бита – число байт данных содержащихся в сообщении (от 0 до 8);
- CRC поле 16 бит: 15 бит используется для обнаружения ошибок передачи данных; 1 бит завершения;
- поле подтверждения 2 бита: 1 бит ACK - передающий узел выдает ‘recessive’ бит, а любой узел, который принял сообщение без ошибок, формирует ‘dominant’ бит подтверждая прием; 2-й завершающий ‘recessive’ бит.

3.2 Расширенное сообщение

Расширенное сообщение формируется, когда узел желает передать данные или сделать удаленный запрос. На рисунке представлена структура расширенного сообщения.



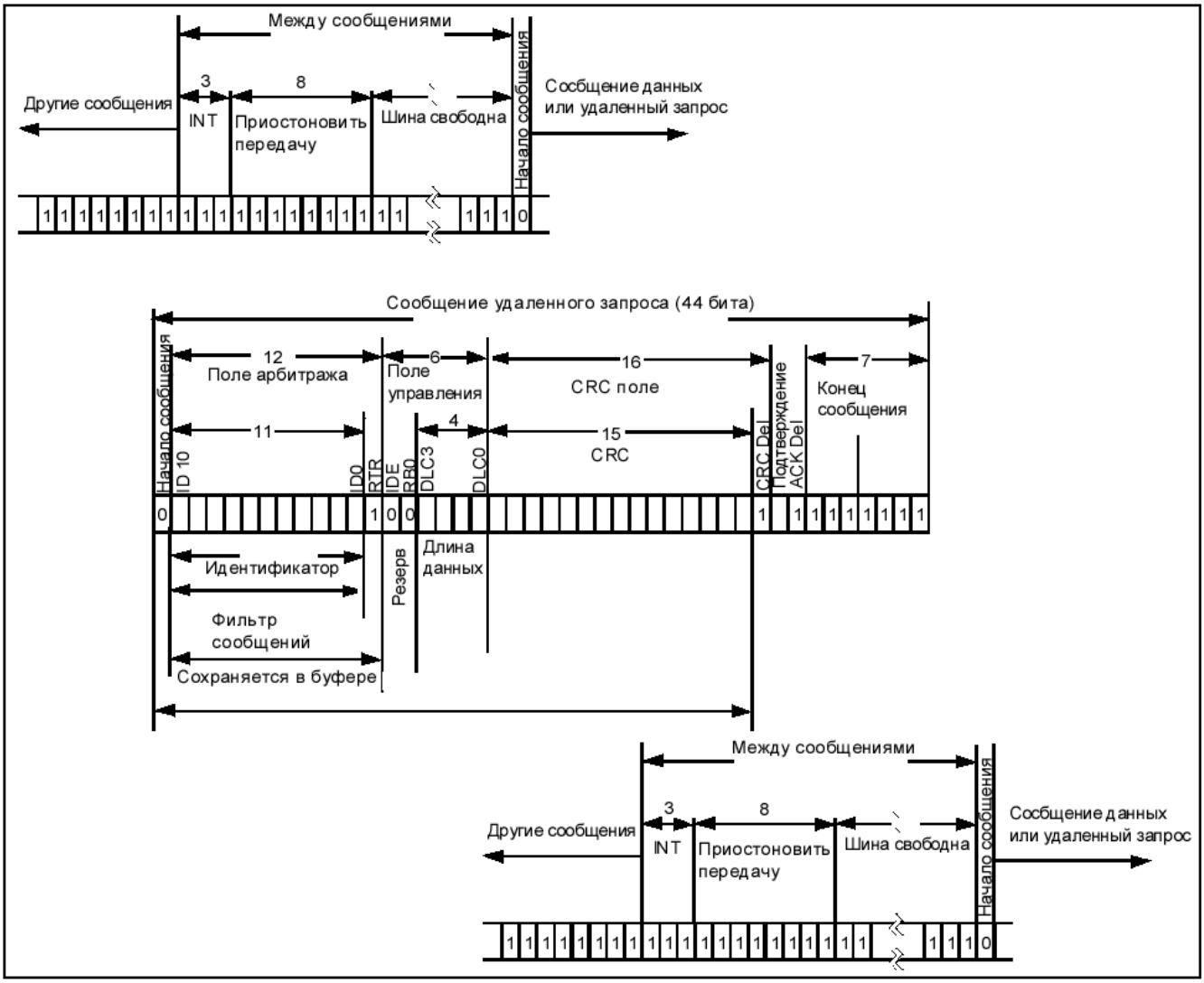
Расширенное сообщение состоит из полей:

- ‘dominant’ бита начала сообщения для жесткой синхронизации всех узлов;
- поле арбитража 38 бит: 11 старших бит идентификатора и SRR бит удаленного запроса; 1 бит IDE указатель расширенного идентификатора (состояние ‘recessive’ определяет расширенное сообщение); 18 младших бит идентификатора и RTR бит передачи по удаленному запросу;
- поле управления 6 бит: RB0:RB1 зарезервированы ‘dominant’ биты; 4 бита – число байт данных содержащихся в сообщении (от 0 до 8);
- CRC поле 16 бит: 15 бит используется для обнаружения ошибок передаче данных; 1 бит завершения;
- поле подтверждения 2 бита: 1 бит ACK - передающий узел выдает ‘recessive’ бит, а любой узел, который принял сообщение без ошибок, формирует ‘dominant’ бит подтверждая прием; 2-й завершающий ‘recessive’ бит.

3.3 Удаленный запрос данных

Обычно данные передаются от источника автономно, но возможно запросить данные из конкретного источника. Для этого узел назначения посылает удаленный запрос с идентификатором источника. Соответствующий узел источника посылает стандартное или расширенное сообщение в ответ на запрос.

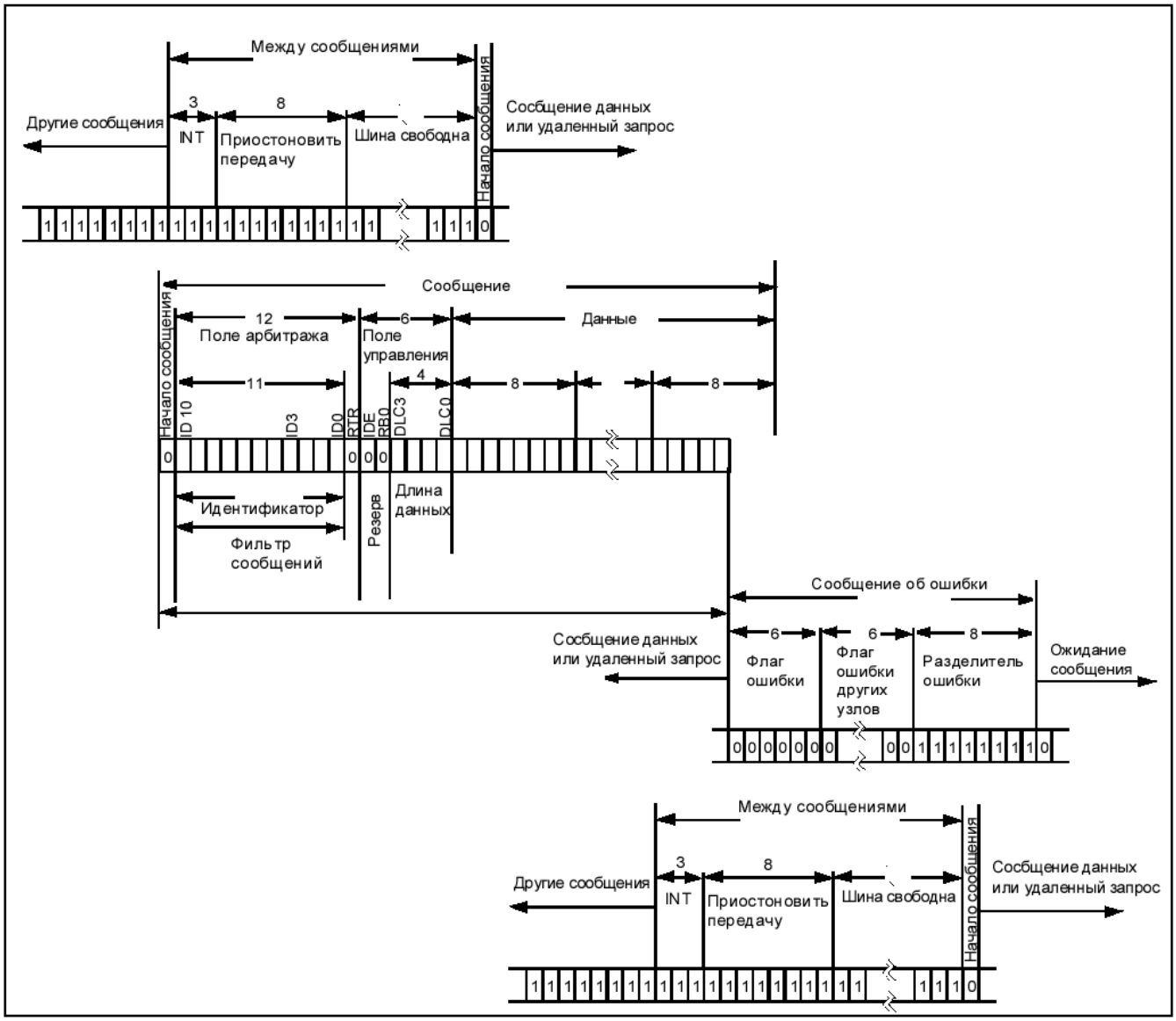
Имеется различие между удаленным запросом и стандартным сообщением.



Бит RTR в удаленном запросе передается в ‘recessive’ состоянии, а в сообщении не передается никаких данных. В самом маловероятном случае, когда одновременно формируется удаленный запрос, и устройство пытается передать данные с одинаковыми идентификаторами, арбитраж будет выигран устройством, передающим данные из-за ‘dominant’ состояния бита RTR. Узел, который посылал запрос, получает данные немедленно.

3.4 Ошибка

Сообщение об ошибке формируется любым узлом, который обнаруживает ошибку на шине. Сообщение об ошибке состоит из двух полей.



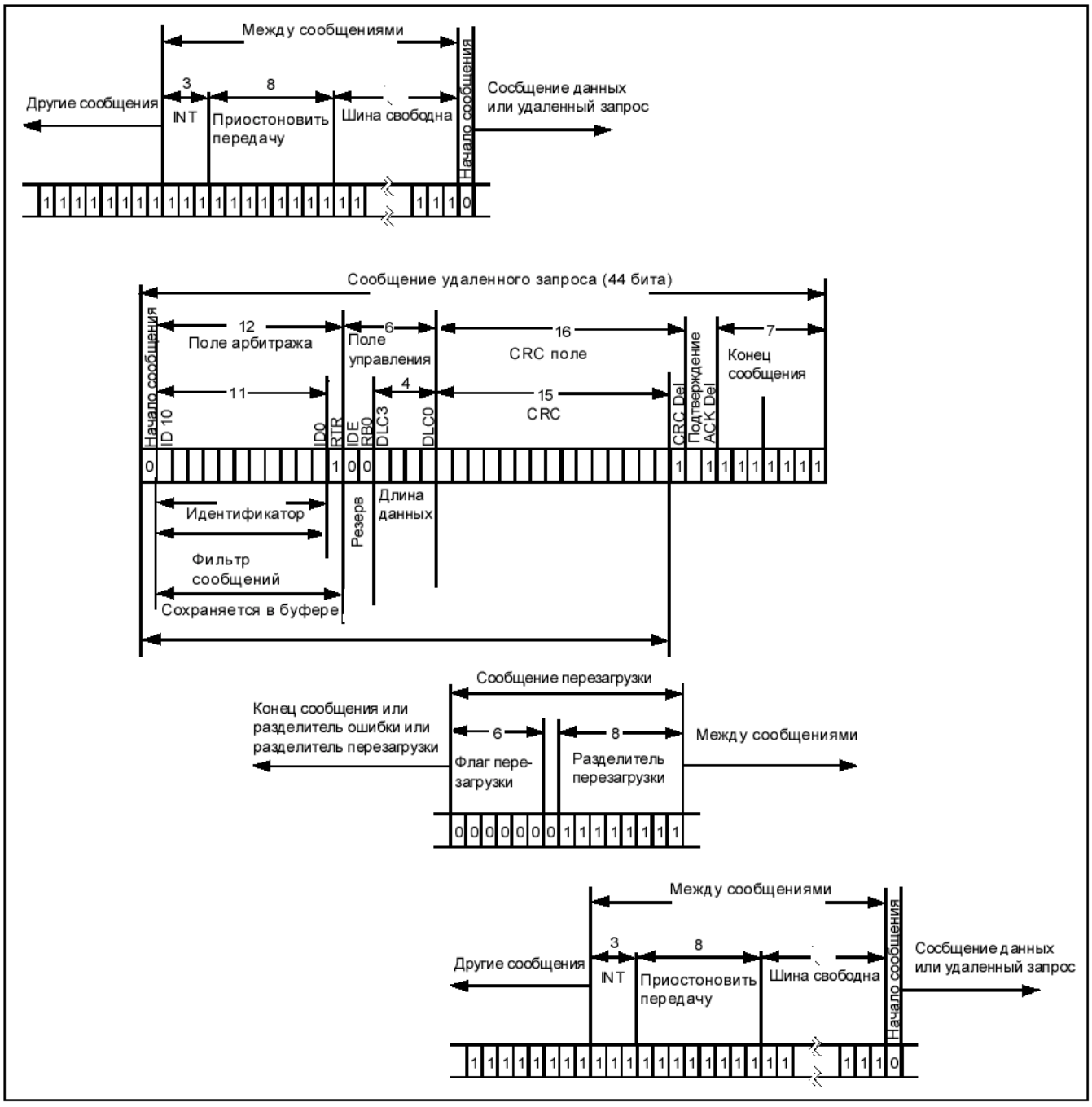
Поле признака ошибки сопровождается разделителем ошибки, состоящим из 8 ‘recessive’ битов и позволяющим перезапускать связь с шиной. Значение поля признака ошибки зависит от текущего состояния узла, который обнаружил ошибку.

Если ошибку обнаружил активный узел, тогда он прерывает передачу текущего сообщения, формируя активный флаг ошибки. Активный флаг ошибки состоит из 6 последовательных ‘dominant’ битов, которые нарушают правило наполнения. Остальные узлы также обнаруживают ошибку и начинают формировать сообщение об ошибке. После перехода шины в нормальное состояние узлы вновь пытаются передать данные.

Если ошибку обнаружил пассивный узел, тогда он формирует пассивный флаг ошибки, состоящий из 6 последовательных ‘recessive’ битов. После передачи пассивной ошибки, узел должен ожидать 6 последовательных ‘recessive’ битов для восстановления связи с шиной.

3.5 Перезагрузка

Формат сообщения перезагрузки аналогичен формату сообщения активной ошибки, но может быть сформирован только, когда шина простаивает.



Сообщение о перезагрузке состоит из двух полей: флаг перезагрузки и разделитель перезагрузки. Флаг перезагрузки состоит из 6 последовательных ‘dominant’ битов. Другие узлы обнаруживают перезагрузку и начинают формировать ее самостоятельно. Поэтому на шине, во время выполнения перезагрузки, может быть до 12 ‘dominant’ битов. Разделитель перезагрузки состоит из 8 последовательных ‘recessive’ бит.

Узел может сформировать сообщение о перезагрузке в двух случаях:

- между сообщениями обнаружен ‘dominant’ бит, что является ненормальным во время простоя шины;
- для задержки передачи нового сообщения.

Узел может последовательно сформировать не более 2 сообщений перезагрузки.

3.6 Простой шины

Между сообщениями шина CAN находится в ‘recessive’ состоянии. Для выполнения условий «простой шины» необходимо, чтобы было получено как минимум 3 ‘recessive’ бита после завершения передачи/приема последнего сообщения.

4. Режимы работы

CAN модуль может находиться в одном из нескольких режимах работы, выбранных пользователем:
- инициализация;
- выключенное состояние;
- нормальный режим;
- только прием данных;
- петлевой режим;
- распознавание ошибки.

Выбор требуемого режима производится установкой соответствующих битов REQOP2:REQOP0, кроме режима распознавания ошибки, который включается битом CANRXM. Текущий режим работы не будет изменен (биты OPMODE2 : OPMODE0) до тех пор, пока на шине не возникнет холостой ход (простой шины), определяемый 11 последовательными ‘recessive’ битами.

4.1 Инициализация

В режиме инициализации CAN модуль не будет принимать или передавать никаких данных, счетчики ошибок сброшены, флаги прерываний остаются неизменными. Пользователь имеет доступ ко всем конфигурационным регистрам модуля, некоторые из которых могут быть недоступны в других режимах. Порядок настройки CAN модуля будет описан в разделе 5.

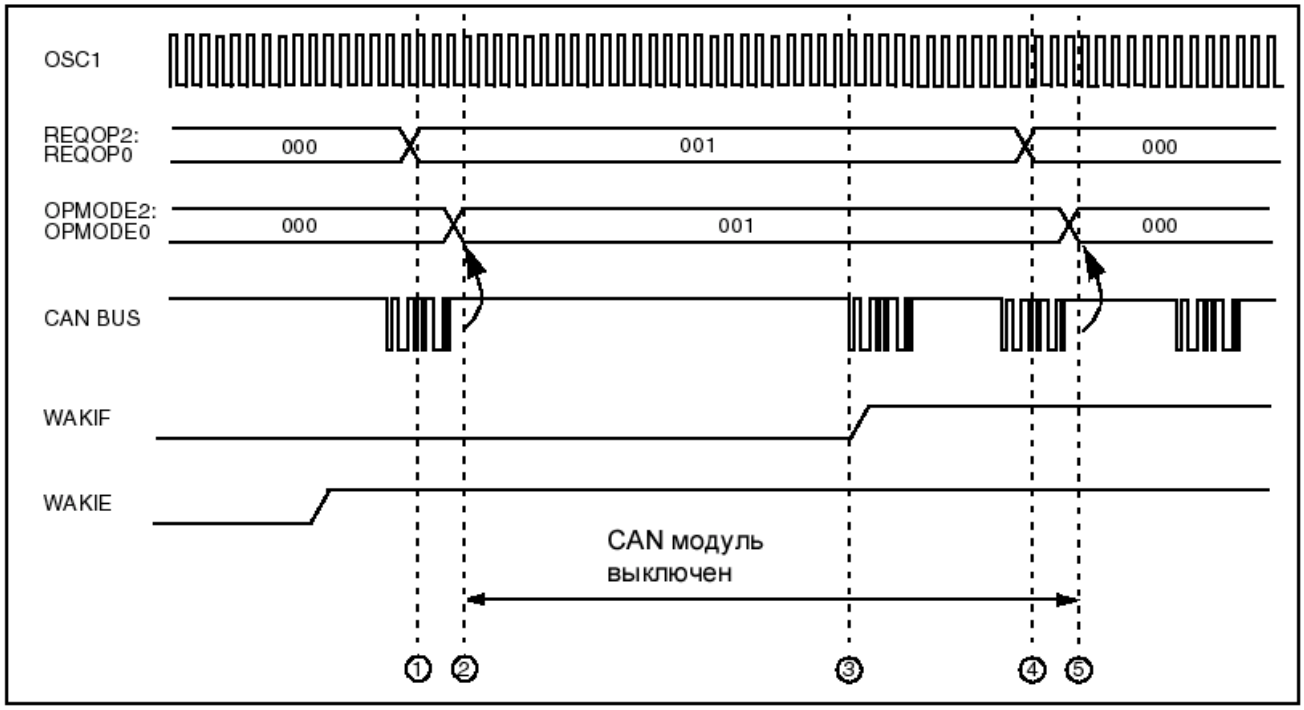
4.2 Выключенное состояние

В выключенном состоянии модуль не может принимать или передавать данные, счетчики ошибок не изменяют своего значения, флаги прерываний сохраняют свое состояние, кроме флага WAKIF обнаружения активности на шине.

Если биты REQOP2:REQOP0 = 001, то модуль переходит в выключенное состояние, останавливая внутреннюю синхронизацию, при неактивном состоянии CAN модуля (т.е. модуль не должен передавать или принимать сообщение).

Если модуль находиться в активном состоянии, он будет ожидать 11 последовательных ‘recessive’ битов (простой шины) перед началом перехода в выключенное состояние. Когда биты OPMODE2:OPMODE0 = 001, то модуль успешно перешел в выключенное состояние.

Порты ввода/вывода, задействованные CAN модулем, переходят в режим цифровых входов/выходов при выключенном состоянии CAN модуля.

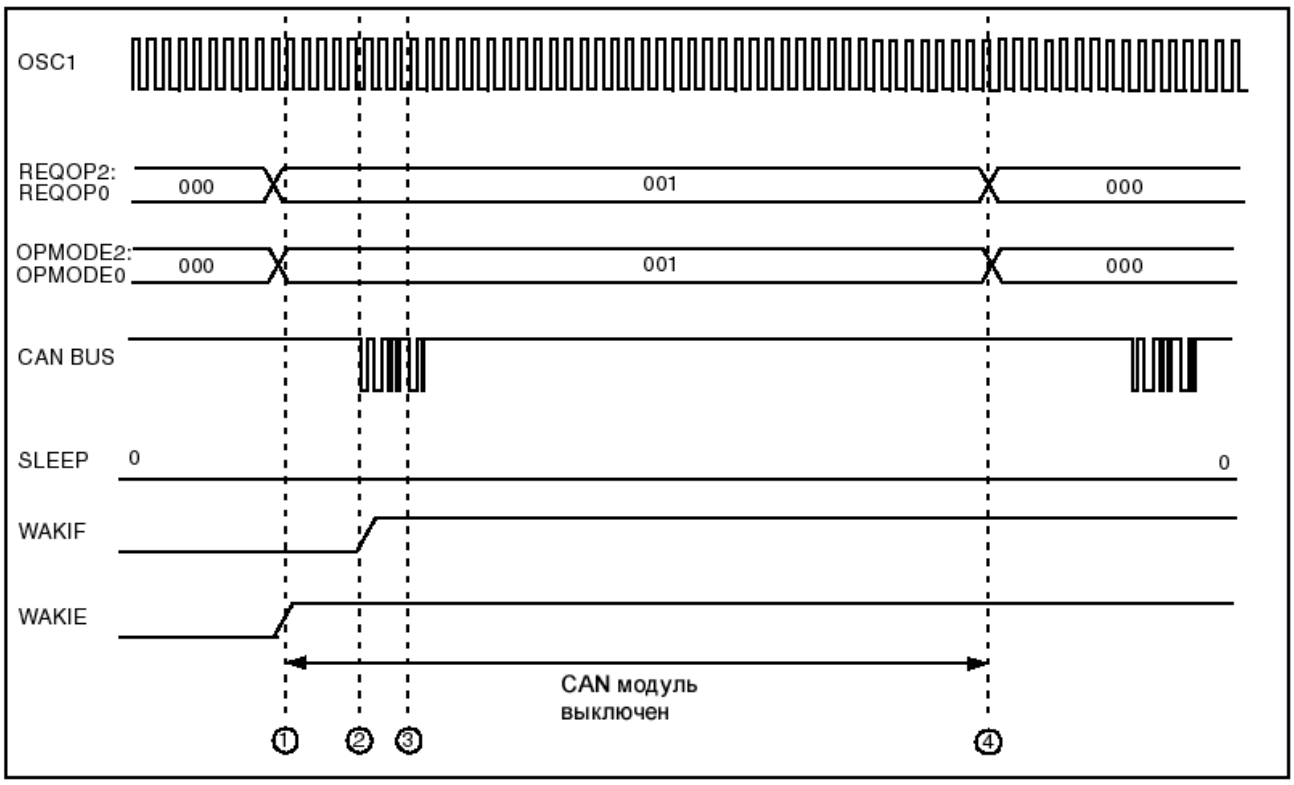


Пояснения к рисунку:

1. Запись в REQOP2:REQOP0 значения 001, в то время, когда модуль CAN выполняет прием/передачу сообщения. Сообщение продолжает обрабатываться модулем.
2. Обнаруживая 11 ‘recessive’ битов, модуль переходит в выключенное состояние, устанавливая OPMODE2:OPMODE0 = 001, подтверждая переход в указанный режим.
3. CAN модуль реагирует на активность шины устанавливая флаг WAKIF (если WAKIE=1), сообщение игнорируется.
4. Запись в REQOP2:REQOP0 = 000 во время выполнения действий на шине. Модуль ожидает 11 ‘recessive’ бит, прежде чем начать переход в другой режим.
5. Модуль обнаруживает 11 ‘recessive’ битов и изменяет режим функционирования, подтверждая записью в OPMODE2:OPMODE0 биты.

4.2.1 SLEEP режим

Команда SLEPP останавливает кварцевый генератор, а процессор переходит в режим энергосбережения. Порты ввода/вывода переходят в режим цифрового входа/выхода в зависимости от состояния битов в регистре TRIS. Рекомендуется сначала перевести модуль CAN в выключенное состояние, затем выполнить команду SLEEP.



Пояснения к рисунку:

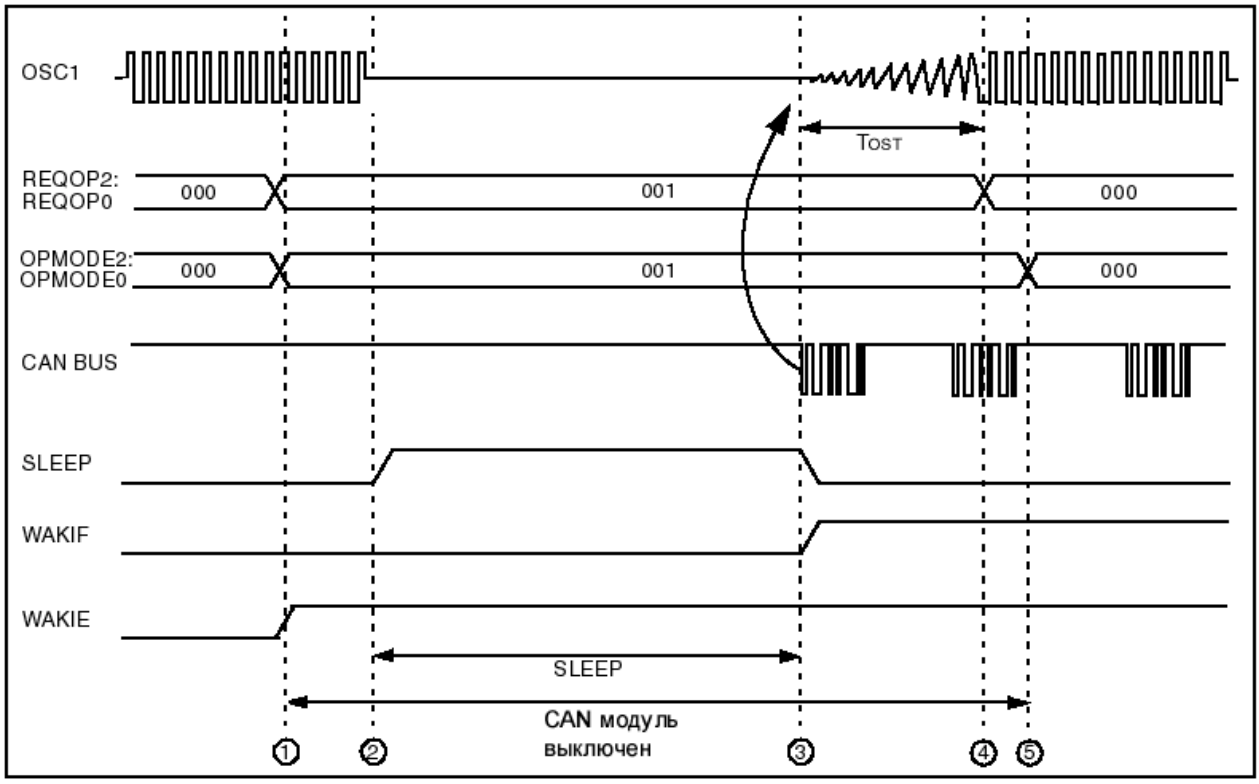
1. В нормальном режиме CAN модуля разрешить прерывания при активности на шине.
2. Активность на шине устанавливает флаг WAKIF. Если GIE=1 процессор перейдет по вектору прерывания.
3. Процессор пытается выполнить команду SLEEP, когда WAKIF=1, WAKIE=1 и GIE=0, вместо команды SLEEP выполняется NOP.
4. Запрос перехода в нормальный режим CAN модуля.

4.2.2 Выход из режима SLEEP

Если микроконтроллер находится в режиме SLEEP, а бит WAKIE установлен, то начало сообщения, передаваемое по шине CAN, выведет микроконтроллер из SLEEP режима сгенерировав прерывание. Само сообщение, вызвавшее «пробуждение», будет потеряно.

Если бит WAKIE сброшен, никакие действия на шине не вызовут выход микроконтроллера из SLEEP режима.

Модуль CAN позволяет включить фильтр нижних частот (установкой бита WAKFIL), в SLEEP режиме микроконтроллера, для защиты от кратковременных помех на шине, которые могут вызвать «пробуждение» микроконтроллера.



Пояснения к рисунку:

1. Перевод модуля CAN в выключенное состояние с разрешением прерываний по обнаружению активности на шине.
2. Выполнение команды SLEEP.
3. Первый бит сообщения устанавливает флаг прерывания WAKIF и запускает тактовый генератор.
4. Процессор выжидает время запуска генератора, продолжает программу или обрабатывает прерывание в зависимости от состояния бита GIE. CAN модуль ожидает 11 последовательных ‘recessive’ битов пред началом действий на шине.
5. Модуль обнаруживает 11 последовательных ‘recessive’ бит, и может передавать и принимать сообщения.

4.3 Нормальный режим

В этом режиме порты ввода/вывода работают под управлением CAN модуля. CAN модуль может передать и принять сообщения, как описано в последующих разделах. Нормальный режим работы CAN модуля выбран, когда OPMODE2:OPMODE0 = 000.

4.4 Только прием данных

Режим «только прием данных» – частный случай нормального режима работы, предназначенный для системной отладки. Модуль CAN находится в пассивном состоянии по отношению к шине, без формирования сигналов подтверждения приема сообщения. Счетчики ошибок выключены. Этот режим может использоваться для определения скорости передачи данных на шине опытным путем. Для определения скорости передачи необходимо как минимум пара узлов, обменивающихся информацией.

4.5 Распознавание ошибки

Режим используется для приема всех сообщений, игнорируя любые ошибки. Включение режима осуществляется установкой соответствующих битов RXM1:RXM0 в регистре RXBNCON. В этом режиме данные, принятые до возникновения ошибки, переписываются в приемный буфер и могут быть обработаны MCU.

4.6 Петлевой режим

В этом режиме передаваемое сообщение принимается по внутренним цепям модуля CAN. Специальные аппаратные средства формируют подтверждение для передатчика.

5. Инициализация

После сброса микроконтроллера, модуль CAN находится в режиме настройки (бит OPMODE2 установлен). Счетчики ошибок очищены, все управляющие регистры содержат значения по умолчанию. Прежде чем сбросить бит REQOP2, необходимо настроить модуль CAN.

Все регистры, управляющие настройкой режима работы модуля CAN, недоступны для изменения в любом другом режиме, что защищает шину от возникновения ошибок.

Защищенные регистры:

- все управляющие регистры;
- регистры идентификаторов приемных фильтров;
- регистры идентификаторов приемных масок.

6. Прием сообщений

6.1 Приемный буфер

В CAN модуле есть 3 приемных буфера, один из которых используется для приема текущего сообщения. Этот буфер называется буфером сборки (MAB). Два других доступных буфера RXB0 и RXB1, по существу, «мгновенно» принимают законченное сообщение от контроллера модуля CAN. MCU может обрабатывать полученные данные, в то время, как принимается новое сообщение.

После приема сообщения в буфер сборки данные транслируются в приемный буфер RXBN, если идентификатор сообщения совпадает с указанным в фильтре. После приема сообщения аппаратно устанавливается бит RXFUL. Бит сбрасывается программно после обработки сообщения - это гарантирует защиту сообщения от обновления приемного буфера. Если бит RXNIE установлен, то формируется прерывание по приему нового сообщения.

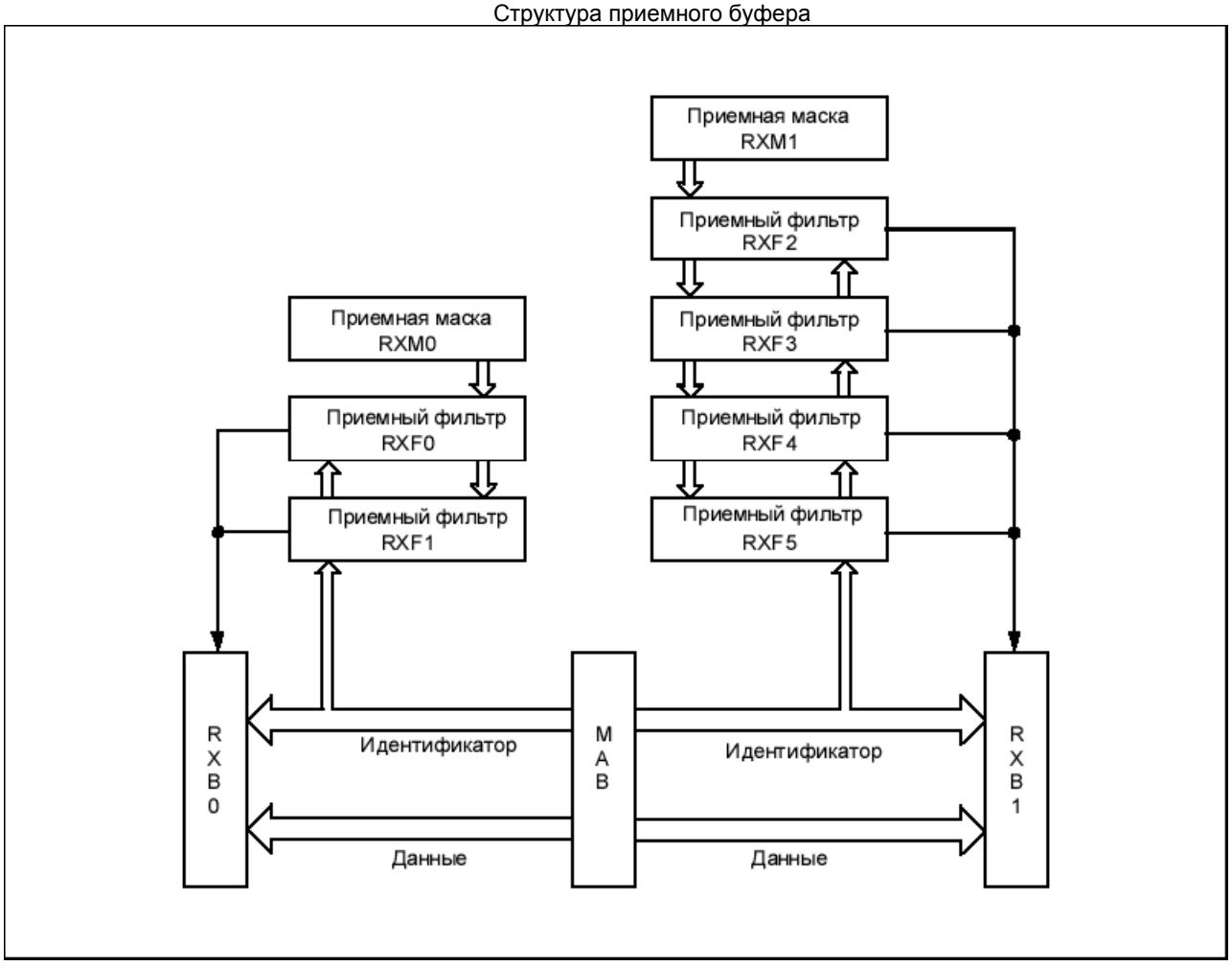
В модуле CAN для приема сообщений используется две маски идентификаторов, по одному для каждого из приемных буферов.

Когда сообщение принято, биты FILHIT2:FILHIT0 регистра RXBnCON указывают приемный критерий для сообщения.

6.1.2 Приоритет приемных буферов

Для каждого приемного буфера в CAN модуле существует несколько приемных фильтров. Буфер RXB0 имеет высокий приоритет и 2 фильтра сообщений. Буфер RXB1 имеет более низкий приоритет и 4 фильтра сообщений. Меньшее число приемных фильтров для буфера RXB0 делает его более ограниченным и подразумевает критичность принимаемых данных. Если буфер RXB0 содержит правильно принятое сообщение, а другое правильно принятое сообщение получено, RXB0 может быть настроен таким образом, что новое сообщение для RXB0 будет записано в RXB1.

Структура приемного буфера



6.2 Фильтры приема сообщений

Фильтры и маски приема сообщений используются для определения в какой из приемных буферов поместить сообщение из буфера сборки. Как только правильно получено сообщение в MAB, поле идентификатора сообщения сравнивается со значением фильтра. Если есть совпадение, то сообщение будут загружено в один из приемных буферов. Маска применяется для указания того, какие биты фильтра будут использоваться для проверки идентификатора сообщения. Если бит маски равен нулю, то соответствующий бит идентификатора сообщения будет принят, не зависимо от состояния бита фильтра. В таблице показан механизм обработки битов маски, фильтра и идентификатора сообщения.

Бит маски n	Бит фильтра n	Бит идентификатора сообщения n	Принять или запретить бит n
0	X	X	Принять
1	0	0	Принять
1	0	1	Запретить
1	1	0	Запретить
1	1	1	Принять

Бит EXIDEN указывает, какой тип идентификаторов проверять. Если бит EXIDEN сброшен, то проверяется 11-разрядный идентификатор стандартного сообщения. Если битам RXM1:RXM0 присвоить значение 01 или 10, назначение бита EXIDEN может быть изменено.

Фильтры RXF0, RXF1 и маска RXM0 связаны с приемным буфером RXB0. Фильтры RXF2, RXF3, RXF4, RXF5 и маска RXM1 связаны с приемным буфером RXB1. Назначение фильтров для буфера RXB1 (регистр RXB1CON) представлено в таблице.

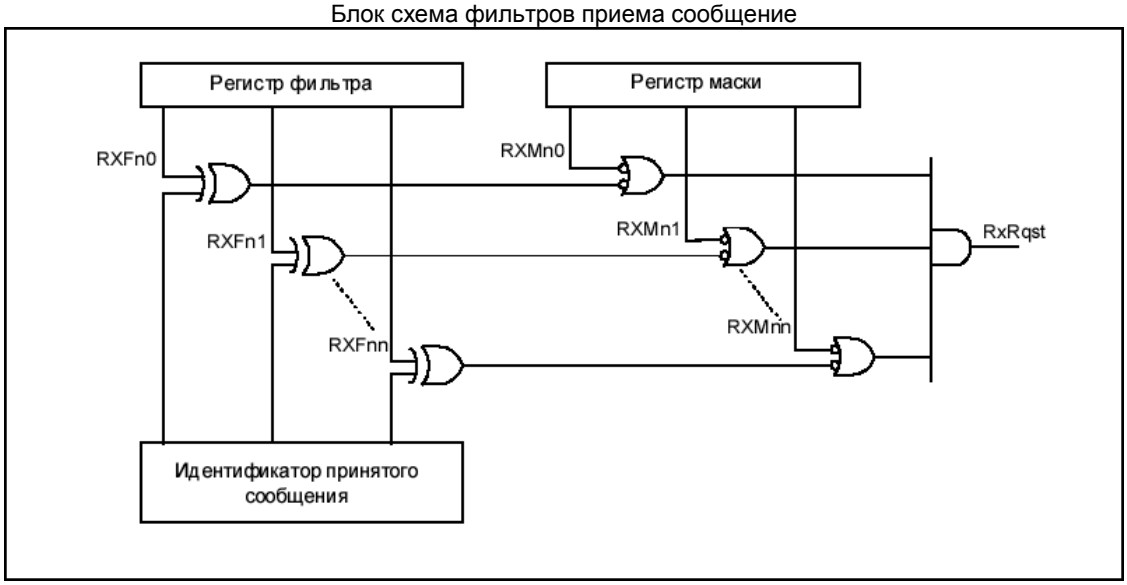
FILHIT2:FILHIT0	Приемный фильтр	Примечание
000	RXF0	Только если RX0DBEN=1
001	RXF1	Только если RX0DBEN=1
010	RXF2	
011	RXF3	
100	RXF4	
101	RXF5	

Битом RX0DBEN можно управлять использованием фильтров, аналогично FILHIT битам, отличать фильтры RXF0 и RXF1 при записи сообщения в RXB0 или в RXB1 (при занятом буфере RXB0).

111= приемный фильтр 1 (RXF1)
110= приемный фильтр 0 (RXF0)
001= приемный фильтр 1 (RXF1)
001= приемный фильтр 0 (RXF0)

Если бит RX0DBEN сброшен, то приемному буферу RXB1 соответствует 4 фильтра. Если бит RX0DBEN установлен, то приемному буферу RXB1 соответствует 4 фильтра, плюс 2 фильтра RXF0, RXF1 при занятом буфере RXB0.

Блок схема фильтров приема сообщение



6.3 Переполнение

Переполнение происходит, когда буфер сборки MAB правильно принял сообщение, выполнено условие фильтрации, но приемный буфер не свободен. Если возникло условие переполнения, будет установлен флаг ошибки RXnOVR и EERIF, сообщение из MAB будет потеряно. Модуль CAN продолжает нормально функционировать, но все новые сообщения для занятого буфера будут потеряны.

При установленном бите RX0DBEN приемные буферы RXB0 и RXB1 работают независимо друг от друга. Если бит RX0DBEN установлен, обработка переполнения для буфера RXB0 выполняется иначе. Новое сообщение, полученное для буфера RXB0, будет записано в буфер RXB1, если RXB0 полон, а RXB1 пуст. При этом ошибка переполнения не возникнет. Если оба буфера полные, то будет сформирована ошибка переполнения для буфера RXB1.

Проверка по фильтрам 0, 1	Проверка по фильтрам 2, 3, 4, 5	RXFUL0	RXFUL1	RX0DBEN	Действие	Примечание
0	0	X	X	X	Нет	Нет сообщений
0	1	X	0	X	MAB -> RXB1	Сообщение для RXB1, RXB1 изменен
0	1	X	1	X	MAB потерян RX1OVFL=1	Сообщение для RXB1, RXB1 полон
1	0	0	X	X	MAB -> RXB0	Сообщение для RXB0, RXB0 изменен
1	0	1	X	0	MAB потерян RX0OVFL=1	Сообщение для RXB0, RXB0 полон
1	0	1	0	1	MAB -> RXB1	Сообщение для RXB0, RXB0 полон, RXB1 изменен
1	0	1	1	1	MAB потерян RX1OVFL=1	Сообщение для RXB0, RXB0 полон, RXB1 полон
1	1	0	X	X	MAB -> RXB0	Сообщение для RXB0 и RXB1, RXB0 изменен
1	1	1	X	0	MAB потерян RX0OVFL=1	Сообщение для RXB0 и RXB1, RXB0 полон
1	1	1	0	1	MAB -> RXB1	Сообщение для RXB0 и RXB1, RXB0 полон, RXB1 изменен
1	1	1	1	1	MAB потерян RX1OVFL=1	Сообщение для RXB0 и RXB1, RXB0 полон, RXB1 полон

6.4 Эффект сброса

После сброса все регистры имеют значения повторной установки, модуль CAN необходимо инициировать заново. Полученные сообщения будут потеряны.

6.5 Ошибки при приеме сообщений

CAN модуль обнаруживает следующие типы ошибок при приеме сообщений:

- ошибка CRC;
- бит наполняющая ошибка;
- недопустимое сообщение.

При возникновении ошибок во время приема сообщений, прерывания не формируются. Однако если приемный счетчик ошибок превысит значения 96, то устанавливается флаг RXWARN, и может быть сформировано прерывание.

6.5.1 Ошибка CRC

Ошибка CRC возникает при не совпадении вычисленного значения циклического кода с принятым. Обнаружив ошибку, модуль CAN формирует сообщение об ошибке, приемный счетчик ошибок увеличивается на единицу.

6.5.2 Бит наполняющая ошибка

Если между началом сообщения и разделителем CRC обнаружено подряд 6 бит одинаковой полярности, возникает бит наполняющая ошибка. Модуль CAN, обнаружив ошибку, формирует сообщение об ошибке, приемный счетчик ошибок увеличивается на единицу.

6.5.3 Недопустимое сообщение

Возникновение ошибки «недопустимое сообщение» отмечается установкой бита IXRIF. Этот бит может использоваться для автоподбора скорости передачи данных по шине CAN.

6.5.4 Счетчик ошибок приемника

Приемный счетчик ошибок изменяется согласно следующим правилам:

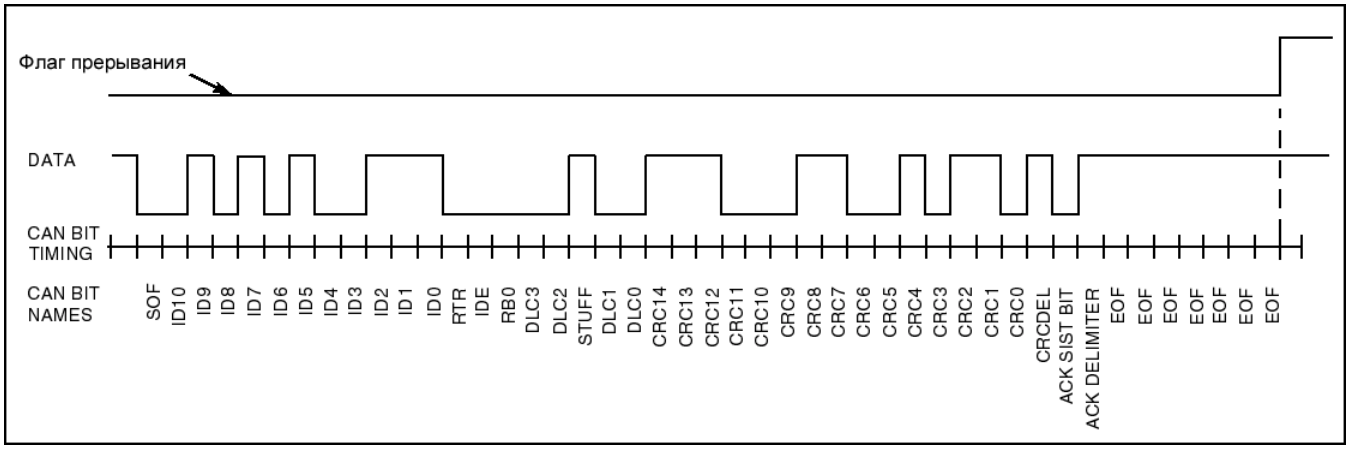
- Когда приемник обнаруживает ошибку, приемный счетчик ошибок увеличивается на единицу, кроме возникновения ошибок при сообщении активной ошибки и перезагрузки.
- Когда приемник обнаруживает ‘dominant’ бит после сообщение активной ошибки, приемный счетчик ошибок увеличивается на 8.
- Если возникает ошибка при поиске получателя, ожидании активного флага ошибки или перезагрузки, приемный счетчик ошибок увеличивается на 8.
- После успешного приема сообщения и формирования подтверждения ACK, приемный счетчик ошибок уменьшается на единицу, если его значение было от 1 до 127. Если приемный счетчик ошибок имеет значение 0, то он не изменяется. Если приемный счетчик ошибок имел значение большее чем 127, то он принимает значение от 119 до 127.

6.6 Прерывания

Прерывания при приеме сообщений можно разделить на две группы:

- прерывания при приеме сообщения
- прерывания при возникновении ошибок

После успешного приема сообщения и загрузки его в один из приемных буферов (окончание приема поля EOF) генерируется прерывание. Проверкой флага RXnIF можно определить, в какой из буферов было принято сообщение. На рисунке показано, когда устанавливается флаг RXnIF.



Бит ERRIF указывает, что произошла ошибка. Источник ошибки определяется проверкой состояния битов регистра COMSTAT. В этом регистре находятся флаги ошибок при приеме и передаче сообщений. Следующие биты регистра используются для определения ошибок при приеме сообщений.

IXRIF – возникла ошибка во время приема последнего сообщения. Этот бит не указывает конкретный тип ошибки. Может быть использован для определения скорости передачи данных на шине CAN в режиме «только прием данных».

RXnOVR – переполнение возникает при правильном приеме нового сообщения, выполнения условий фильтрации и занятом буфере получателе. Новое принятое сообщение будет потеряно. CAN модуль продолжает нормально работать. (см. отличия для буфера RXB0)

RXWARN – предупреждение об достижении приемного счетчика ошибок значения 96. Сбрасывается программно. Аппаратно сбрасывается при уменьшении счетчика ошибок до 95.

RXBP – состояние пассивной ошибки, значение приемного счетчика ошибок превысило 127. Сбрасывается программно.

6.7 Режимы приема

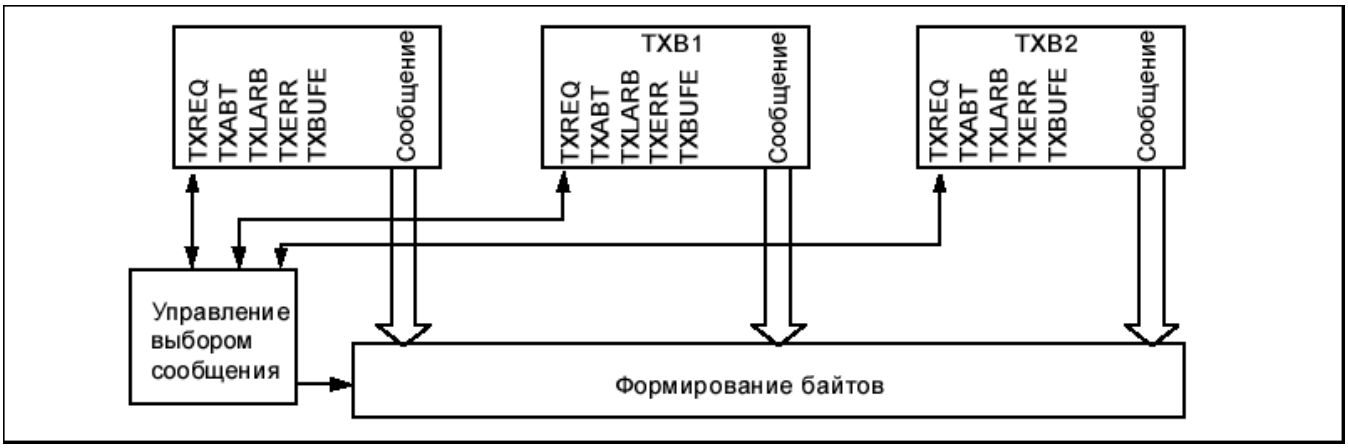
С помощью битов RXM1:RMX0 устанавливают специальные режимы приема сообщений. Обычный режим работы – прием всех правильных сообщений, с учетом приемных фильтров (RXM1 : RXM0=00). Если биты RXM1:RXM0 установлены в 01 или 10, получатель примет только стандартные или расширенные сообщения соответственно. Если значение бита EXIDEN не соответствует RXM1:RXN0, то приемный фильтр не используется. Эти два способа приема сообщений могут использоваться в системах, в которых известно, что будут передаваться только стандартные или расширенные сообщения.

Если значение RXM1:RXM0 = 11, буфер примет все сообщения, не зависимо от значения приемных фильтров и наличия ошибок. Этот способ приема может быть полезен во время отладки системы.

7. Передача сообщений

Для передачи сообщений в реальном масштабе времени, CAN модуль должен удерживать арбитраж шины, а передающие сообщения иметь высокий приоритет. Если узел имеет только один буфер передачи, то, передав сообщение, необходимо «отпустить» шину для подготовки новых данных. С двумя передающими буферами схема работы следующая: один буфер передает данные, другой - перезагружается. Однако нагрузка на MCU, и при такой схеме, значительна. Необходимо гарантировать обновление второго буфера, до того как закончиться передача из первого.

Типовые приложения требуют три передающих буфера. Один буфер передает данные, второй подготовлен к передаче, а третей может быть перезагружен MCU. Данная схема несколько снижает нагрузку на MCU. Имея три передающих буфера, есть возможность распределить приоритет между ними.



7.1 Передающий буфер

CAN модуль имеет 3 передающих буфера, каждый из которых занимает 14 байт памяти:

- 8 байт, данные сообщения;
- 5 байт, стандартный и расширенный идентификаторы, информация арбитража сообщения;
- 1 байт, служебный байт, связанный с каждым сообщением. Указываются условия и статус передачи.

Данные, подготовленные для передачи, сохраняются в регистрах TXBnDm. Как минимум, должен быть указан стандартный идентификатор в TXBnSIDH:TXBnSIDL регистрах. Если в сообщении будет использоваться расширенный идентификатор, то его необходимо записать в регистры TXBnEIDm и установить бит EIXDEN. Перед началом передачи пользователь должен указать приоритет и включить прерывания TXIE, если это необходимо. После окончания передачи устанавливается флаг TXBnIF и сбрасывается TXREQ.

7.2 Приоритет сообщения

До посылки бита SOF на шину CAN, приоритеты всех буферов одинаковы. Сначала передаются данные из буфера с самым высоким приоритетом. Если два буфера имеют одну и туже приоритетную уставку, то первым будет передан буфер с более высоким адресным приоритетом. Например, буфер 0 и 1 имеют одинаковый приоритет, первым будут переданы данные из буфера 1. Биты TXPRI1:TXPRI0 используются для указания приоритета буфера (11 наивысший приоритет).

7.3 Передача сообщения

Чтобы инициировать передачу сообщения, необходимо установить бит TXREQ, CAN модуль сам решает конфликты синхронизации между установкой бита TXREQ и формированием бита SOF. При установке бита TXREQ, автоматически сбрасываются биты TXABT, TXLARB и TXERR.

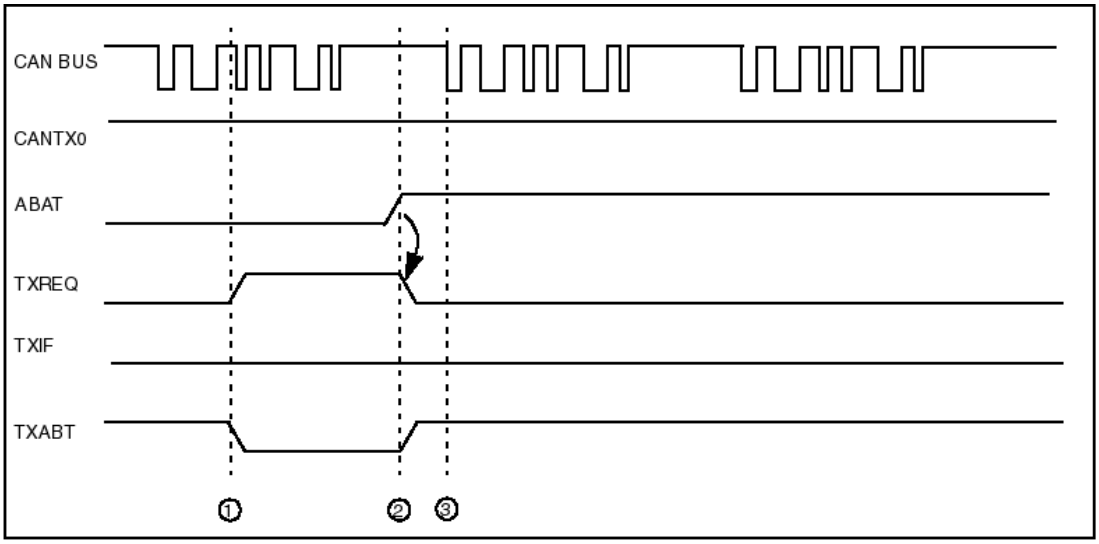
Установка бита TXREQ фактически не запускает передачу, а помещает буфер сообщения в очередь для передачи. Передача начнется, когда модуль CAN обнаружит простой шины, с сообщения с самым высоким приоритетом.

Если сообщение успешно передано с первой попытки, TXREQ сброситься, будет сформировано прерывание, если оно разрешено битом TXIE.

Если сообщение не удается отправить с первого раза, один из флагов условия будет установлен. Бит TXREQ остается установленным, показывая, что сообщение все еще ожидает передачи. Если во время передачи возникла ошибка, будет установлен бит TXERR, что может вызвать прерывание. При потере арбитража, во время попытки передать сообщение, устанавливается бит TXLARB, прерывание не формируется.

7.4 Отмена передачи сообщения

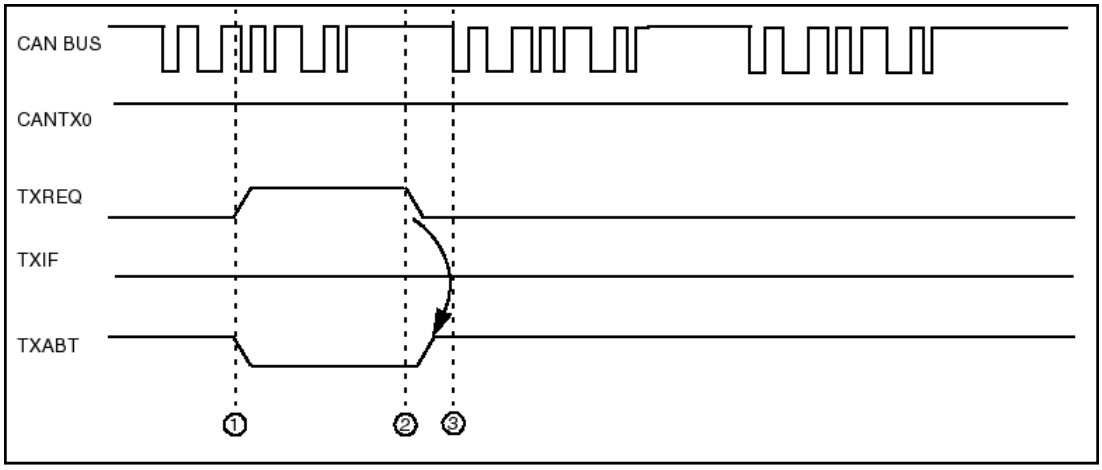
Модуль CAN позволяет отменить передачу сообщения, сбросом бита TXREQ для каждого из буферов в отдельности. Установкой бита ABAT будет выполнен запрос на прекращение передачи всех буферов.



Пояснения к рисунку:

1. Установка бита TXREQ для начала передачи сообщения.
2. Установка бита ABAT для отмены всех поставленных в очередь сообщений, CAN модуль ожидает 11 ‘recessive’ бит, через 2 такта устанавливает TXABT.
3. Передача сообщений отменена.

Поставленное в очередь сообщение отменено, сбросом бита TXREQ.

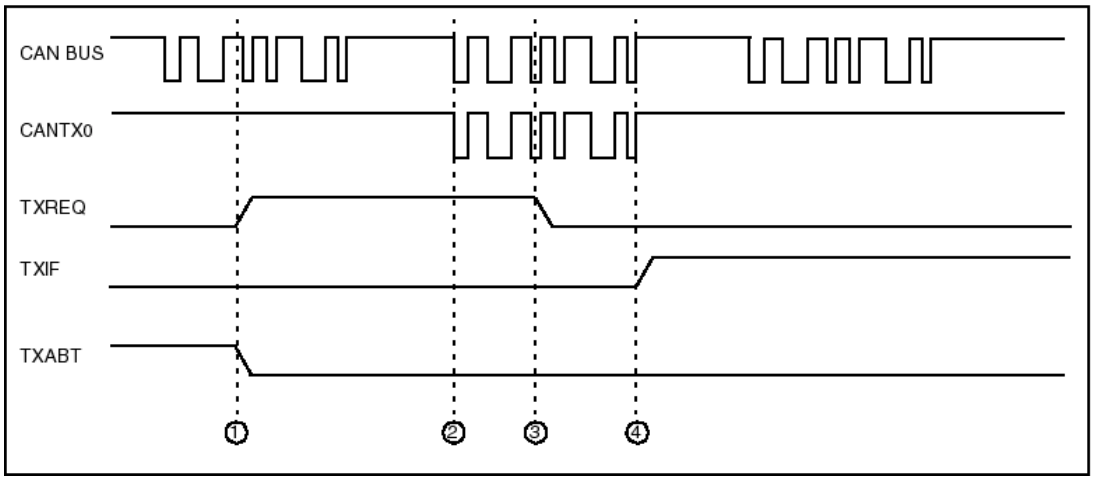


Пояснения к рисунку:

1. Установка бита TXREQ для начала передачи сообщения.
2. Сброс бита TXREQ для отмены передачи сообщения, CAN модуль ожидает 11 ‘recessive’ бит, через 2 такта устанавливает TXABT.
3. Передача сообщения отменена.

Если сообщение еще не запущено на передачу, или если передача была прервана в результате ошибки или потере арбитража, отмена передачи будет обработана. После прекращения, выполнения операции отмены передачи сообщения, устанавливается бит TXABT, флаг TXnIF не устанавливается.

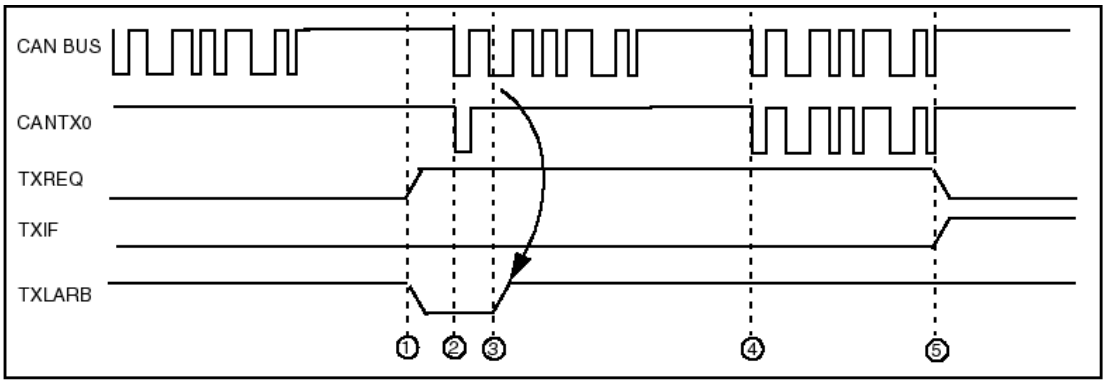
Если начата передача сообщения, и поступила команда отмены, модуль CAN будет пытаться передать сообщение полностью. При удачной передаче сообщения бит TXABT не устанавливается.



Пояснения к рисунку:

1. Установка бита TXREQ для начала передачи сообщения.
2. CAN модуль начинает передачу сообщения после обнаружения 11 ‘recessive’ бит.
3. Сброс бита TXREQ для отмены передачи сообщения, подтверждение отмены сообщения нет.
4. Передача сообщения завершена, бит TXABT сброшен, бит TXnIF установлен.

Если попытку передать сообщение, при поступившей команде отмены, выполнить не удалось, повторно сообщение не передается. Будет выполнена команда отмены и установлен бит TXABT.



Пояснения к рисунку:

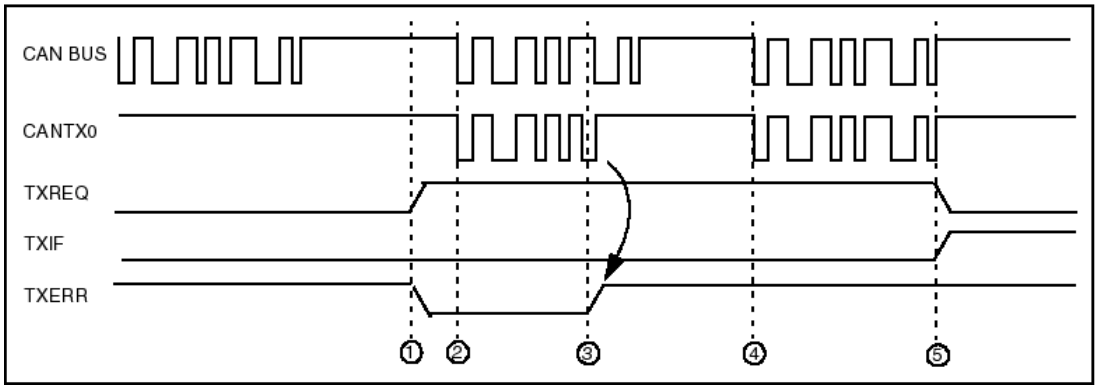
1. Установка бита TXREQ для начала передачи сообщения. TXLARB автоматически сбрасывается.
2. CAN модуль начинает передачу сообщения .
3. Потеря арбитража, устанавливается бит TXARB.
4. После обнаружения 11 ‘recessive’ бит, вторая попытка передать сообщение.
5. Передача сообщения завершена, бит TXREQ сброшен, бит TXIF установлен.

7.5 Ошибки при передаче сообщений

CAN модуль может обнаруживать следующие типы ошибок при передаче:
- ошибка подтверждения
- ошибка формы
- ошибка бита

Эти ошибки не обязательно вызывают прерывания, но обязательно увеличивают счетчик ошибок передатчика. Как только значение счетчика ошибок превысит 96. Устанавливается флаг ERRIF и предупреждение TXWARN.

Пример ошибки передатчика показан на рисунке.



Пояснения к рисунку:
1. Установка бита TXREQ для начала передачи, сбрасывается бит TXERR.
2. Начало передачи сообщения.
3. Обнаружение ошибки. Устанавливается бит TXERR.
4. Ожидание 11 ‘recessive’ бит для повторной передачи.
5. Передача сообщения завершена, бит TXREQ сброшен, бит TXIF установлен.

7.5.1 Ошибка подтверждения

Ошибка подтверждения возникает, если отсутствует ‘dominant’ бит в поле подтверждения, т.е. ни один из узлов не получил сообщение правильно. Сообщение должно быть передано повторно.

7.5.2 Ошибка формы

Ошибка формы возникает, если передатчик обнаруживает ‘dominant’ бит в одном из четырех сегментов сообщения: конец сообщения; в паузе между сообщениями, разделитель подтверждения и разделитель CRC. Формируется ошибка, сообщение должно быть передано повторно.

7.5.3 Ошибка бита

Ошибка бита возникает, если передатчик посылает ‘recessive’ бит, а обнаруживает ‘dominant’. В этом случае формируется ошибка, поэтому сообщение должно быть передано повторно. Исключением является поле подтверждения ACK.

7.5.4 Счетчик ошибок передатчика

Счетчик ошибок передатчика изменяется по следующим правилам:
- Счетчик увеличивается на 8, если обнаруживает ошибку подтверждения, и если не обнаруживает ‘dominant’ битов на шине при формировании пассивной ошибки.
- Любой узел допускает 7 последовательных ‘dominant’ битов после перезагрузки или сообщения активной ошибки. Если обнаружено более 7 ‘dominant’, счетчик ошибок увеличивается на 8.
- После успешной передачи сообщения счетчик ошибок уменьшается на единицу, если значение счетчика не нуль.

7.6 Прерывания

Есть два типа прерываний, связанных с передачей сообщений:
- передача сообщения завершена;
- ошибка при передаче сообщения.

После завершения передачи сообщения устанавливается флаг TXIF, указывая, что, по крайне мере, один из трех буферов пуст.

При возникновении ошибки во время передачи сообщения устанавливается флаг ERRIF. Для определения источника ошибки необходимо проверить биты регистра COMSTAT. Следующие биты регистра связаны с передатчиком:

TXWARN – предупреждение о достижении счетчиком ошибок передатчика значения 96. Прерывание возникает при переходе из 0 в 1. Бит не может быть сброшен программно, он сбрасывается автоматически при значении счетчика меньше 96. Флаг ERRIF сбрасывается программно.

TXEP – указывает, что модуль находится в пассивном состоянии. Значение счетчика ошибок передатчика превысило 127. Прерывание возникает при переходе значения бита из 0 в 1. Бит не может быть сброшен программно, он сбрасывается автоматически при значении счетчика меньше 127. Флаг ERRIF сбрасывается программно.

TXBO – указывает, что модуль CAN находится в состоянии отключенном от шины. Значение счетчика ошибок передатчика превысило 255. Прерывание возникает при переходе значения бита из 0 в 1. Бит не может быть сброшен программно. Флаг ERRIF сбрасывается программно.

7.7 Эффект сброса

После сброса все регистры имеют значения повторной установки, модуль CAN необходимо инициировать заново. Полученные сообщения будут потеряны.

7.8. Скорость передачи данных

Все узлы шины CAN должны работать на одной скорости передачи данных. Установка скорости передачи может быть выполнена только в режиме инициализации и не может быть изменена. Описание настройки скорости передачи данных смотрите в разделе 9.

8. Обнаружение ошибок

CAN протокол обеспечивает сложные механизмы обнаружения ошибок. Ошибки делятся на две основных группы:

Ошибки приемника:
- ошибка CRC;
- бит наполняющая ошибка;
- недопустимое сообщение.

Ошибки передатчика:
- ошибка подтверждения;
- ошибка формы;
- ошибка бита.

8.1 Статус ошибки

При возникновении ошибки во время передачи сообщения, ошибочное сообщение прерывается и будет повторно передано как можно раньше. Каждый узел шины CAN может находиться в одном из трех состояний:

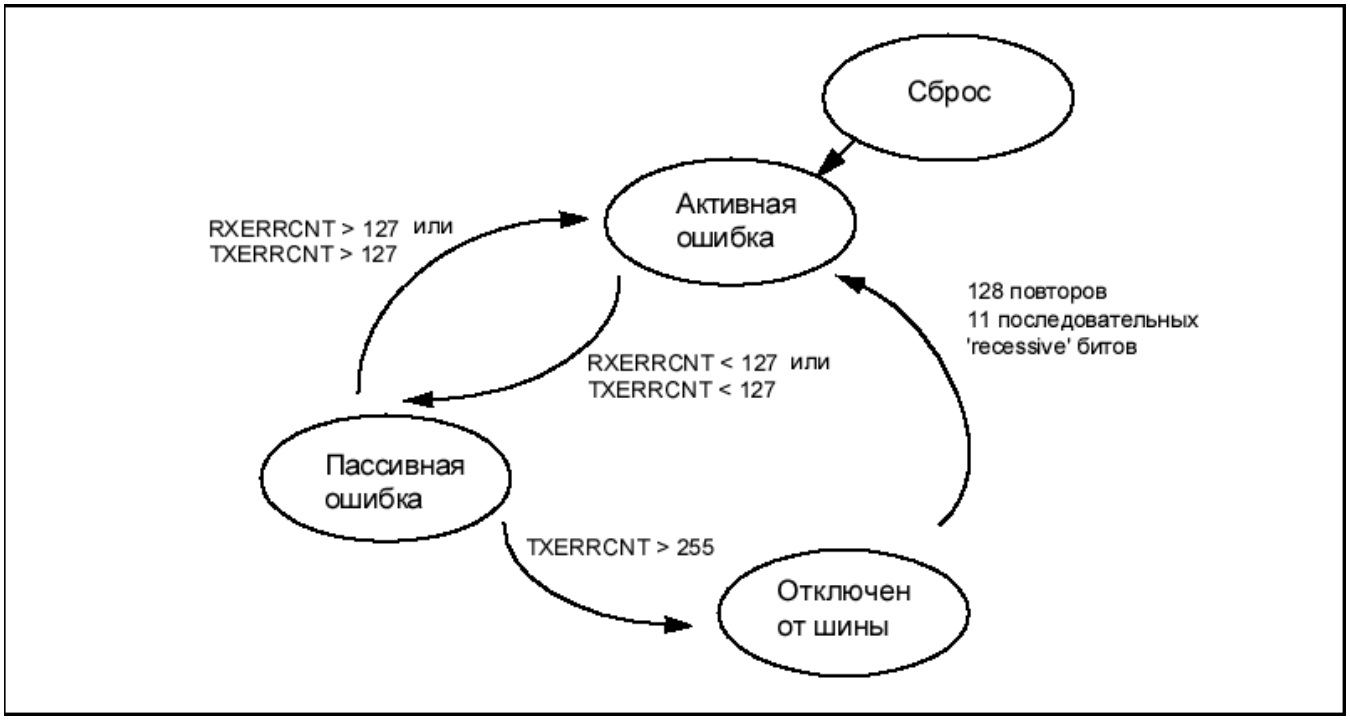
Состояние активной ошибки – обычное состояние: узел может передавать, принимать данные и формировать ‘dominant’ сообщение об ошибке.

Состояние пассивной ошибки: узел может передавать, принимать данные и формировать ‘recessive’ сообщение об ошибке.

Состояние отключено от шины – состояние «выключено»: узел не может передавать и принимать сообщения.

8.2 Счетчики ошибок

CAN модуль содержит два счетчика ошибок: приемный счетчик ошибок (RXERRCNT), счетчик ошибок передатчика (TXERRCNT). Значение обоих счетчиков доступно для MCU. Значение счетчиков изменяется согласно техническим требованиям к CAN шине. CAN модуль находится в состоянии активной ошибки, если значение обоих счетчиков меньше 128. Состояние пассивной ошибки возникает, когда один из счетчиков превысит значение 127. Если счетчик переполнится (значение больше 255), узел переходит в состояние - отключен от шины. Дополнительно имеется промежуточное (предупреждающее) состояние, когда значение одного из счетчиков превышает 96.



9. Установка скорости передачи данных

Все узлы шины CAN должны работать на одной скорости передачи сообщений. CAN шина позволяет использовать кодирование NZR, в котором не присутствует синхронизация. Поэтому приемники, с независимой синхронизацией, должны восстанавливать синхронизацию данных.

Для настройки скорости передачи должны быть выполнены следующие настройки:

- ширина перехода синхронизации;
- предварительный делитель;
- фазовые сегменты;
- длина фазового сегмента 2;
- элементы выборки;
- сегменты распределения.

9.1 Разрядная синхронизация

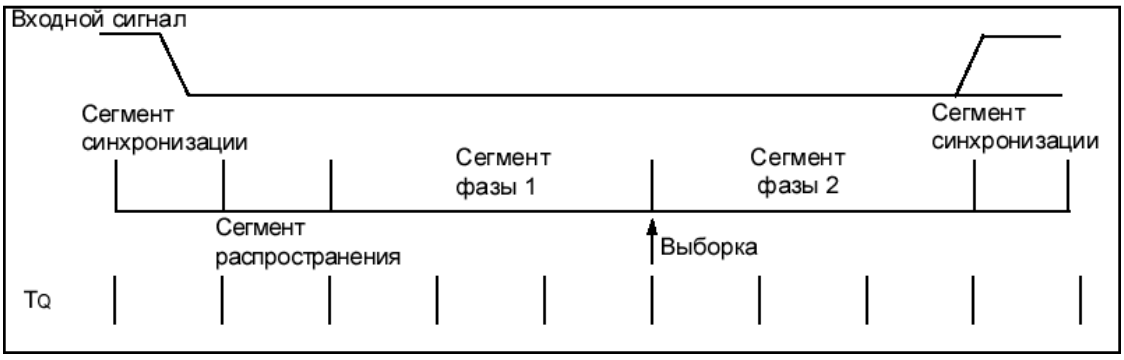
Поскольку тактовые генераторы и время распространения сигнала от узла к узлу могут быть различные, приемник должен иметь некоторое подобие фазовой автоподстройки, синхронизированной к переходу уровней во время передачи данных. Для кодирования NZR необходимо использовать наполняющий бит, чтобы гарантировать изменение полярности передаваемых данных, по крайне мере, каждые 6 бит передачи. Эту функцию выполняет устройство цифровой фазовой автоподстройки синхронизации (DPLL).

Логика работы DPLL программно настраивается с учетом следующих факторов:

- время передачи бита;
- тип синхронизации к локальному генератору;
- компенсация задержки в цепи передачи;
- позиционирование элемента выборки.

Время передачи одного бита можно разделить на сегменты:

- сегмент синхронизации (синхронизация SEG);
- сегмент распространения (опора SEG);
- буфер фазы сегмента 1 (Фаза 1 SEG);
- буфер фазы сегмента 2 (фаза 2 SEG).



Сегменты времени, номинальное время передачи бита - составлены из целочисленных отрезков времени, называемых квантами времени TQ. Номинальное время передачи бита колеблется от 8TQ до 25TQ. Минимальное номинальное время передачи бита 1мкс, что соответствует скорости передачи 1Мбит/с.

9.2 Предварительный делитель частоты

Программируемый предварительный делитель частоты предназначен для определения длительности кванта времени TQ. Длительность кванта времени рассчитывается по формуле:

TQ = 2 x (BRP + 1) x Tosc
BRP <5:0> имеет значения от 0 до 63

Например:

1. Fosc = 16МГц, BRP5:BRP0 = 00h, номинальное время передачи бита = 8TQ
тогда, TQ = 125нс, скорость передачи данных = 1Мбит/с

2. Fosc = 32МГц, BRP5:BRP0 = 01h, номинальное время передачи бита = 8TQ
тогда, TQ = 125нс, скорость передачи данных = 1Мбит/с

3. Fosc = 32МГц, BRP5:BRP0 = 3Fh, номинальное время передачи бита = 25TQ
тогда, TQ = 4мкс, скорость передачи данных = 10Кбит/с

9.3 Сегмент распространения

Сегмент распространения используется для компенсации физической времени запаздывания в пределах сети. Задержка рассчитывается с учетом: прохождения сигнала от передатчика к приемнику и обратно, входной задержки компаратора и задержки выхода драйвера. Значения сегмента распространения колеблются от 1TQ до 8TQ, и устанавливаются битами PRESE2:PRESE0.

9.4 Фазовые сегменты

Фазовые сегменты предназначены для оптимального расположения выборки полученного бита, в пределах времени переданного бита. Точка выборки располагается между фазовым сегментом 1 и фазовым сегментом 2.

Фазовый сегмент 1 определяет точку выборки в пределах передаваемого бита и имеет значение от 1TQ до 8TQ (SEG1PH2 : SEG1PH0). Фазовый сегмент 2 обеспечивает задержку до начала следующего бита, устанавливается в пределах от 1TQ до 8TQ (SEG2PH2 : SEG2PH0).

9.5 Элементы выборки

Элемент выборки – время, когда читается состояние шины для определения принятого бита. Выборка размешается в конце фазового сегмента 1. Если синхронизация бита медленная (содержит большое количество TQ), возможно задать многократный выборочный контроль состояния шины. CAN модуль производит выборку три раза для каждого принимаемого бита, с периодичностью TQ/2. Значение, полученное два (три) раза, принимается как истинное. Включение многократной выборки производится установкой бита SAM в регистре BRG2CON.

9.6 Синхронизация

Чтобы компенсировать смещение фазы между частотами генераторов различных узлов шины, каждый CAN модуль должен синхронизироваться к переходу уровня входного сигнала. Как только переход обнаружен, то логика синхронизации сравнит размещение перехода с ожидаемым и выполнит настройку значений фазовых сегментов 1 и 2. Есть два механизма синхронизации:

- аппаратная синхронизация;
- синхронизация с восстановлением тактовых интервалов.

9.6.1 Аппаратная синхронизация

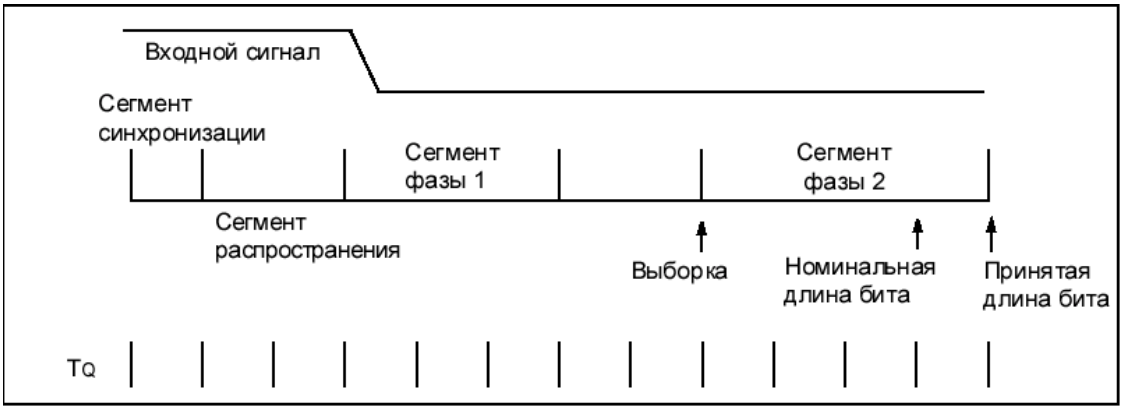
Аппаратная синхронизация выполняется при переходе от ‘recessive’ к ‘dominant’ биту в течение холостого хода шины, указывая начало сообщения. При жесткой синхронизации тактовые интервалы не изменяются в течение всего сообщения.

9.6.2 Синхронизация с восстановлением тактовых интервалов

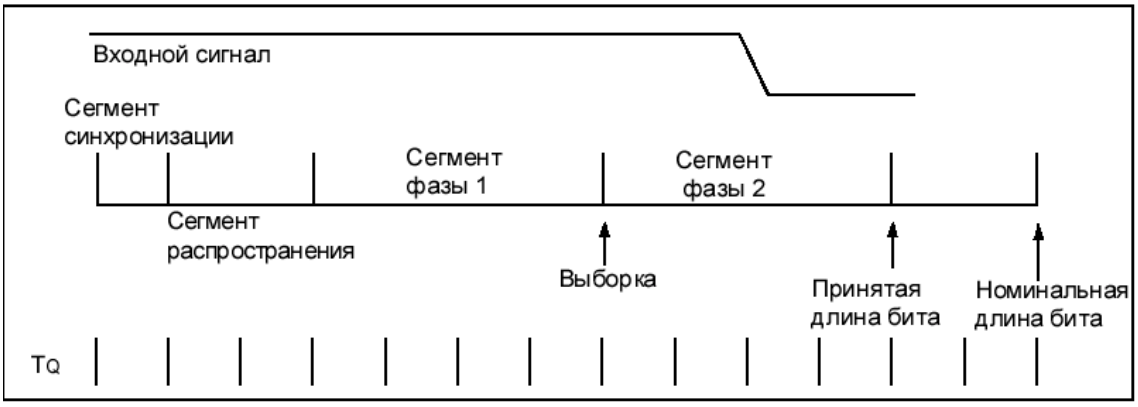
Синхронизация с восстановлением тактовых интервалов выполняется автоматическим удлинением фазы сегмента 1 или укорачивая фазовый сегмент 2. Максимальное значение изменения фазовых сегментов колеблется в пределах от 1TQ до 4TQ (SJMP1:SJMP0). Синхронизация выполняется только при переходах от ‘recessive’ к ‘dominant’ биту. Фиксированное значение максимального числа последовательных бит одинаковой полярности гарантирует своевременное восстановление синхронизации. Фазовое искажение перехода выполняется относительно синхронного сегмента, измеренного в квантах TQ.

Если величина фазового искажения меньше или равна запрограммированному значению ширины перехода восстановления тактовых интервалов, синхронизация выполняется жестким методом.

Если величина фазового искажения большая, чем ширина перехода восстановления тактовых интервалов, и если фазовое искажение положительно, то фазовый сегмент 1 удлиняется.



Если величина фазового искажения большая, чем ширина перехода восстановления тактовых интервалов, и если фазовое искажение отрицательно, то фазовый сегмент 2 укорачивается.



9.7 Программирование времени сегментов

Некоторые требования для программирования времени сегментов:

Сегмент распространения + сегмент фазы 1 >= Сегмент фазы 2 > синхронная ширина перехода

Скорость передачи данных = 125Кбит/с
Fosc = 20МГц

Tosc = 50нс
BR5:BRP0 = 04h, TQ = 500 нс

Для скорости передачи данных 125Кбит/с, номинальное время бита 16TQ

Как правило, осуществление выборки должно иметь место приблизительно 60 – 70% от времени передачи бита, в зависимости от системных параметров.

Синхронный сегмент – 1TQ
Сегмент распространения – 2TQ
Сегмент фазы 1 – 7TQ
Сегмент фазы 2 – 6TQ

Ширина перехода (SJW1:SJW0) может быть установлена максимальной 4TQ, однако в типовых приложениях достаточно 1TQ.

10. Прерывания

CAN модуль имеет несколько источников прерываний, каждое из которых можно разрешать или запрещать. Регистр PIR3 содержит флаги прерываний. В регистре PIE3 находятся маски 8 основных прерываний CAN модуля. Специальный набор битов ICODE2:ICODE0 в регистре CANSTAT используется для эффективной обработки прерываний.

Прерывание по возникновению любой из ошибок устанавливает флаг ERRIF. Идентифицировать тип ошибки можно с помощью регистра COMSTAT.

Прерывания можно разделить на две категории: прерывания приемника, прерывания передатчика.

Прерывания приемника:
- прием сообщения;
- переполнение;
- «пробуждение»;
- предупреждение;
- переход в пассивное состояние.

Прерывания передатчика:
- передача сообщения;
- предупреждение;
- переход в пассивное состояние;
- переход в состояние «отключен от шины».

10.1 обработка прерываний

Возникновение прерываний связано с установкой одного или более битов в регистрах COMSTAT или PIR. Прерывания ожидаются, пока один из соответствующих флагов не будет установлен. Флаги прерываний в указанных регистрах должны быть сброшены в подпрограмме обработки прерываний, чтобы была возможность обработать следующее прерывание. Флаг прерывания не может быть сброшен, пока существует условие возникновения прерывания, за исключением условий, которые вызваны достижением счетчиком ошибок определенного значения.

10.2 Биты ICODE

ICODE2:ICDE0 биты – набор битов доступных только для чтения, предназначены для эффективной обработки прерываний через таблицу переходов. Единовременно эти биты могут указывать только на одно прерывание с наивысшим приоритетом. В таблице представлена операция формирования ICODE2:ICODE0 битов.

ICODE2 : ICODE0	Операция
000	-ERR • - WAK • -TX0 • -TX1 • -TX2 • -RX0 • -RX1
001	ERR
010	-ERR • TX0
011	-ERR • -TX0 • TX1
100	-ERR • -TX0 • -TX1 • TX2
101	-ERR • -TX0 • -TX1 • -TX2 • RX0
110	-ERR • -TX0 • -TX1 • -TX2 • -RX0 • RX1
111	-ERR • -TX0 • -TX1 • -TX2 • -RX0 • -RX1 • WAK

Примечание к таблице:

ERR = ERRIF & ERRIE
TX0 = TX0IF & TX0IE
TX1 = TX1IF & TX1IE
TX2 = TX2IF & TX2IE
RX0 = RX0IF & RX0IE
RX1 = RX0IF & RX1IE
WAK = WAKIF & WAKIE

11. Порты ввода/вывода

CAN модуль использует до 3 портов ввода/вывода: 1 или 2 передающие выводы, и 1 приемный. Управление портами ввода /вывода осуществляется установкой/сбросом соответствующих бит в регистр CIOCON.

Когда модуль CAN находится в режиме настройки, петлевом режиме или в состоянии «отключен от шины», порты ввода/вывода работают как цифровые входы/выходы.

Если CAN модуль работает в режиме не симметричного драйвера, то используется только вывод TX0. При использовании дифференциального выхода, необходимо подключить вывод TX1 установкой бита TX1EN.

В активном состоянии CAN модуля биты регистра TRIS, для портов ввода вывода используемых модулем, игнорируются.

CAN модуль может принимать и передавать данные по одному порту ввода/вывода.

Дополнительно в CAN модуле предусмотрен выход, на котором появляется единичный импульс при правильном приеме нового сообщения.

11. Управляющие регистры CAN модуля

Все регистры, связанные с CAN модулем, подразделяются на пять групп:
- регистры управления и статуса;
- регистры передающего буфера;
- регистры приемного буфера;
- регистры, управляющие скоростью передачи;
- регистры управления и прерываний.

11.1 Регистры управления и статуса

CANCON: регистр управления CAN модуля

R/W - 1	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0
REQOP2	**REQOP1**	**REQOP0**	**ABAT**	**WIN2**	**WIN1**	**WIN0**	**-**
бит7							бит0

бит 7-5: **REQOP2:REQOP0:** запрос режим работы CAN модуля
1xx = настройка CAN модуля
011 = только прием данных
010 = режим циклического возврата
001 = CAN модуль отключен от шины
000 = нормальный режим работы

бит 4: **ABAT:** удаление всех битов, ожидающих передачу
1 = удалить все биты, ожидающие передачу (во всех передающих буферах)
0 = передача данных выполняется или была остановлена
Примечание: бит автоматически сбрасывается после выполнения действия

бит 3-1: **WIN2:WIN0:** указатель буфера
Биты указывают, с какого буфера выполнялся доступ к памяти. При возникновении прерывания значения из ICODE2:ICODE0 переписываются в WIN2:WIN0.
111 = приемный буфер 0
110 = приемный буфер 0
101 = приемный буфер 1
100 = передающий буфер 0
011 = передающий буфер 1
010 = передающий буфер 2
001 = приемный буфер 0
000 = приемный буфер 0

бит 0: **Не используется:** читается как ‘0’

CANSTAT: регистр статуса CAN модуля

R - 1	R - 0	R - 0	U – 0	R - 0	R - 0	R - 0	U - 0
OPMODE2	**OPMODE1**	**OPMODE0**	**-**	**ICODE2**	**ICODE1**	**ICODE0**	**-**
бит7							бит0

бит 7-5: **OPMODE2:OPMODE0:** режим работы CAN модуля
111 = резерв
110 = резерв
101 = резерв
100 = настройка CAN модуля
011 = только прием данных
010 = режим циклического возврата
001 = CAN модуль отключен от шины
000 = нормальный режим работы
Примечание: при переводе микроконтроллера в SLEEP режим, необходимо выбрать режим «отключен от шины».

бит 4: **Не используется:** читается как ‘0’

бит 3-1: **ICODE2:ICODE0:** код прерывания
111 = «пробуждение» по прерыванию
110 = прерывание RXB0
101 = прерывание RXB1
100 = прерывание TXB0
011 = прерывание TXB1
010 = прерывание TXB2
001 = ошибка прерывания
000 = нет прерывания

бит 0: **Не используется:** читается как ‘0’

COMSTAT: регистр коммутационного статуса CAN модуля

R/C - 0	R/C - 0	R - 0	R - 0	R - 0	R - 0	R - 0	R - 0
RX0OVFL	**RX1OVFL**	**TXBO**	**TXBP**	**RXBP**	**TXWARN**	**RXWARN**	**EWARN**
бит7							бит0

бит 7: **RX0OVFL:** переполнение приемного буфера 0
1 = приемный буфер переполнен
0 = нет переполнения

бит 6: **RX1OVFL:** переполнение приемного буфера 1
1 = приемный буфер переполнен
0 = нет переполнения

бит 5: **TXBO:** флаг, передатчик отключен от шины
1 = значение счетчика ошибок передатчика больше 255
0 = значение счетчика ошибок передатчика меньше или равно 255

бит 4: **TXBP:** флаг, передатчик в пассивном состоянии
1 = значение счетчика ошибок передатчика больше 127
0 = значение счетчика ошибок передатчика меньше или равно 127

бит 3: **RXBP:** флаг, приемник в пассивном состоянии
1 = значение счетчика ошибок приемника больше 127
0 = значение счетчика ошибок приемника меньше или равно 127

бит 2: **TXWARN:** флаг предупреждения о наличии большого числа ошибок передатчика
1 = значение счетчика ошибок передатчика больше 95
0 = значение счетчика ошибок передатчика меньше или равно 95

бит 1: **RXWARN:** флаг предупреждения о наличии большого числа ошибок приемника
1 = значение счетчика ошибок приемника больше 95
0 = значение счетчика ошибок приемника меньше или равно 95

бит 0: **EWARN:** флаг предупреждения
1 = один из битов TXWARN или RXWARN установлен
0 = оба бита TXWARN и RXWARN сброшены

11.2 Регистры передающего буфера

TXB0CON: управляющий регистр передающего буфера 0
TXB1CON: управляющий регистр передающего буфера 1
TXB2CON: управляющий регистр передающего буфера 2

U - 0	R - 0	R - 0	R - 0	R/W - 0	U - 0	R/W - 0	R/W - 0
-	**TXABT**	**TXLARB**	**TXERR**	**TXREQ**	**-**	**TXPRI1**	**TXPRI0**
бит7							бит0

бит 7: **Не используется:** читается как ‘0’

бит 6: **TXABT:** флаг прерванного сообщения
1 = передача сообщения была прервана
0 = сообщение передано полностью

бит 5: **TXLARB:** флаг потерянного арбитража
1 = арбитраж был потерян
0 = арбитраж сохранен

бит 4: **TXERR:** флаг обнаружения ошибки во время передачи
1 = обнаружена ошибка
0 = ошибки не обнаружено

бит 3: **TXREQ:** запрос на передачу сообщения
1 = выполнить запрос на передачу сообщения. Сбрасываются биты TXABT, TXLARB и TXERR.
0 = автоматически сбрасывается возможной передачи

бит 2: **Не используется:** читается как ‘0’

бит 1-0: **TXPRI1:TXPRI0:** приоритет передачи
11 = уровень приоритета 3 (наивысший)
10 = уровень приоритета 2
01 = уровень приоритета 1
00 = уровень приоритета 0

TXB0SIDH: идентификатор стандартного сообщения передающего буфера 0, старший байт
TXB1SIDH: идентификатор стандартного сообщения передающего буфера 1, старший байт
TXB2SIDH: идентификатор стандартного сообщения передающего буфера 2, старший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
SID10	**SID9**	**SID8**	**SID7**	**SID6**	**SID5**	**SID4**	**SID3**
бит7							бит0

бит 7-0: **SID10:SID3:** биты идентификатора стандартного сообщения

TXB0SIDL: идентификатор стандартного сообщения передающего буфера 0, младший байт
TXB1SIDL: идентификатор стандартного сообщения передающего буфера 1, младший байт
TXB2SIDL: идентификатор стандартного сообщения передающего буфера 2, младший байт

R/W - x	R/W - x	R/W - x	U - 0	R/W - x	U - 0	R/W - x	R/W - x
SID2	**SID1**	**SID0**	**-**	**EXIDEN**	**-**	**EID17**	**EID16**
бит7							бит0

бит 7-5: **SID2:SID0:** биты идентификатора стандартного сообщения

бит 4: **Не используется:** читается как ‘0’

бит 3: **EXIDEN:** тип сообщения
1 = расширенное сообщение (идентификатор 29 бит)
0 = стандартное сообщение (идентификатор 11 бит)

бит 2: **Не используется:** читается как ‘0’

бит 1-0: **EID17:EID16:** биты идентификатора расширенного сообщения

TXB0EIDH: идентификатор расширенного сообщения передающего буфера 0, старший байт
TXB1EIDH: идентификатор расширенного сообщения передающего буфера 1, старший байт
TXB2EIDH: идентификатор расширенного сообщения передающего буфера 2, старший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
EID15	**EID14**	**EID13**	**EID12**	**EID11**	**EID10**	**EID9**	**EID8**
бит7							бит0

бит 7-0: **EID15:EID8:** биты идентификатора расширенного сообщения

TXB0EIDL: идентификатор расширенного сообщения передающего буфера 0, младший байт
TXB1EIDL: идентификатор расширенного сообщения передающего буфера 1, младший байт
TXB2EIDL: идентификатор расширенного сообщения передающего буфера 2, младший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
EID7	**EID6**	**EID5**	**EID4**	**EID3**	**EID2**	**EID1**	**EID0**
бит7							бит0

бит 7-0: **EID7:EID0:** биты идентификатора расширенного сообщения

TXB0Dm: байт данных m передающего буфера 0
TXB1Dm: байт данных m передающего буфера 1
TXB2Dm: байт данных m передающего буфера 2

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
TXBnDm7	**TXBnDm6**	**TXBnDm5**	**TXBnDm4**	**TXBnDm3**	**TXBnDm2**	**TXBnDm1**	**TXBnDm0**
бит7							бит0

бит 7-0: **TXBnDm7:TXBnDm0:** биты данных байта m буфера передатчика n (0<=n<=2, 0<=m<=7)

TXB0DLC: управляющий регистр, количество байт данных в сообщении передающего буфера 0
TXB1DLC: управляющий регистр, количество байт данных в сообщении передающего буфера 1
TXB2DLC: управляющий регистр, количество байт данных в сообщении передающего буфера 2

U - 0	R/W - x	U - 0	U - 0	R/W - x	R/W - x	R/W - x	R/W - x
-	**TXRTR**	**-**	**-**	**DLC3**	**DLC2**	**DLC1**	**DLC0**
бит7							бит0

бит 7: **Не используется:** читается как ‘0’

бит 6: **TXRTR:** бит удаленного запроса данных RTR
1 = RTR установлен
0 = RTR сброшен

бит 5-4: **Не используется:** читается как ‘0’

бит 3-0: **DLC3:DLC0:** количество байт данных в сообщении
1111 - 1001 = резерв
1000 = 8 байт данных
0111 = 7 байт данных
0110 = 6 байт данных
0101 = 5 байт данных
0100 = 4 байта данных
0011 = 3 байта данных
0010 = 2 байта данных
0001 = 1 байт данных
0000 = 0 байт данных

TXERRCNT: счетчик ошибок передатчика

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
TEC7	**TEC6**	**TEC5**	**TEC4**	**TEC3**	**TEC2**	**TEC1**	**TEC0**
бит7							бит0

бит 7-0: **TEC7:TEC0:** значение счетчика ошибок передатчика
Когда счетчик переполняется, узел отключается от шины. Если на шине будет последовательно передано 11 ‘recessive’ бит, значение счетчика сбросится.

11.3 Регистры приемного буфера

RXB0CON: управляющий регистр приемного буфера 0

R/C - 0	R/W - 0	R/W - 0	U - 0	R - 0	R/W - 0	R - 0	R/W - 0
RXFUL	**RXM1**	**RXM0**	**-**	**RXRTRRO**	**RX0DBEN**	**JTOFF**	**FILHIT0**
бит7							бит0

бит 7: **RXFULL:** статус буфера
1 = буфер полон
0 = буфер пуст, можно принять новое сообщение
Примечание: Сбрасывается программно после чтения данных из буфера

бит 6-5: **RXM1:RXM0:** режим работы приемного буфера
11 = принимать все сообщения
10 = принимать только расширенные сообщения без ошибок
01 = принимать только стандартные сообщения без ошибок
00 = принимать сообщения без ошибок

бит 4: **Не используется:** читается как ‘0’

бит 3: **RXRTRRO:** запрос удаленной передачи данных
1 = принят запрос удаленной передачи
0 = запроса удаленной передачи не было

бит 2: **RX0DBEN:** разрешение дублирования буфера
1 = если буфер 0 переполнен данные будут записаны в буфер 1
0 = дублирование запрещено

бит 1: **JTOFF:** бит смещения таблицы переходов
1 = смещение таблицы между 6 и 7
0 = смещение таблицы между 1 и 0

бит 0: **FILHIT0:** активный приемный фильтр
1 = фильтр 1 (RXIF1)
0 = фильтр 0 (RXIF0)

RXB1CON: управляющий регистр приемного буфера 1

R/C - 0	R/W - 0	R/W - 0	U - 0	R - 0	R/W - 0	R/W - 0	R/W - 0
RXFUL	**RXM1**	**RXM0**	**-**	**RXRTRRO**	**FILHIT2**	**FILHIT1**	**FILHIT0**
бит7							бит0

бит 7: **RXFULL:** статус буфера
1 = буфер полон
0 = буфер пуст, можно принять новое сообщение
Примечание: сбрасывается программно после чтения данных из буфера

бит 6-5: **RXM1:RXM0:** режим работы приемного буфера
11 = принимать все сообщения
10 = принимать только расширенные сообщения без ошибок
01 = принимать только стандартные сообщения без ошибок
00 = принимать сообщения без ошибок

бит 4: **Не используется:** читается как ‘0’

бит 3: **RXRTRRO:** запрос удаленной передачи данных
1 = принят запрос удаленной передачи
0 = запроса удаленной передачи не было

бит 2-0: **FILHIT2:FILHIT0:** активный приемный фильтр
111 = резерв
110 = резерв
101 = фильтр 5 (RXIF5)
100 = фильтр 4 (RXIF4)
011 = фильтр 3 (RXIF3)
010 = фильтр 2 (RXIF2)
001 = фильтр 1 (RXIF1), только при установленном бите RX0DBEN
000 = фильтр 0 (RXIF0), только при установленном бите RX0DBEN

RXB0SIDH: идентификатор стандартного сообщения приемного буфера 0, старший байт
RXB1SIDH: идентификатор стандартного сообщения приемного буфера 1, старший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
SID10	**SID9**	**SID8**	**SID7**	**SID6**	**SID5**	**SID4**	**SID3**
бит7							бит0

бит 7-0: **SID10:SID3:** биты идентификатора стандартного сообщения

RXB0SIDL: идентификатор стандартного сообщения приемного буфера 0, младший байт
RXB1SIDL: идентификатор стандартного сообщения приемного буфера 1, младший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	U - 0	R/W - x	R/W - x
SID2	**SID1**	**SID0**	**SRR**	**EXID**	**-**	**EID17**	**EID16**
бит7							бит0

бит 7-5: **SID2:SID0:** биты идентификатора стандартного сообщения

бит 3: **SRR:** флаг удаленного запроса данных (только когда EXID = 1)
1 = получен удаленный запрос данных
0 = нет удаленного запроса данных

бит 3: **EXID:** тип сообщения
1 = принято расширенное сообщение (идентификатор 29 бит)
0 = принято стандартное сообщение (идентификатор 11 бит)

бит 2: **Не используется:** читается как ‘0’

бит 1-0: **EID17:EID16:** биты идентификатора расширенного сообщения

RXB0EIDH: идентификатор расширенного сообщения приемного буфера 0, старший байт
RXB1EIDH: идентификатор расширенного сообщения приемного буфера 1, старший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
EID15	**EID14**	**EID13**	**EID12**	**EID11**	**EID10**	**EID9**	**EID8**
бит7							бит0

бит 7-0: **EID15:EID8:** биты идентификатора расширенного сообщения

RXB0EIDL: идентификатор расширенного сообщения приемного буфера 0, младший байт
RXB1EIDL: идентификатор расширенного сообщения приемного буфера 1, младший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
EID7	**EID6**	**EID5**	**EID4**	**EID3**	**EID2**	**EID1**	**EID0**
бит7							бит0

бит 7-0: **EID7:EID0:** биты идентификатора расширенного сообщения

RXB0Dm: байт данных m приемного буфера 0
RXB1Dm: байт данных m приемного буфера 1

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
RXBnDm7	**RXBnDm6**	**RXBnDm5**	**RXBnDm4**	**RXBnDm3**	**RXBnDm2**	**RXBnDm1**	**RXBnDm0**
бит7							бит0

бит 7-0: **RXBnDm7:RXBnDm0:** биты данных байта m буфера приемника n (0<=n<=1, 0<=m<=7)

RXB0DLC: управляющий регистр, количество байт данных в сообщении приемного буфера 0
RXB1DLC: управляющий регистр, количество байт данных в сообщении приемного буфера 1

U - 0	R/W - x	R - x	R - x	R - x	R - x	R - x	R - x
-	**RXRTR**	**RB1**	**RB0**	**DLC3**	**DLC2**	**DLC1**	**DLC0**
бит7							бит0

бит 7: **Не используется:** читается как ‘0’

бит 6: **RXRTR:** бит удаленного запроса данных RTR
1 = RTR установлен
0 = RTR сброшен

бит 5-4: **RB1:RB0:** читается как ‘0’, резерв протокола CAN

бит 3-0: **DLC3:DLC0:** количество байт данных в сообщении
1111 - 1001 = ошибка
1000 = 8 байт данных
0111 = 7 байт данных
0110 = 6 байт данных
0101 = 5 байт данных
0100 = 4 байта данных
0011 = 3 байта данных
0010 = 2 байта данных
0001 = 1 байт данных
0000 = 0 байт данных

RXERRCNT: счетчик ошибок приемника

R - 0	R - 0	R - 0	R - 0	R - 0	R - 0	R - 0	R - 0
REC7	**REC6**	**REC5**	**REC4**	**REC3**	**REC2**	**REC1**	**REC0**
бит7							бит0

бит 7-0: **REC7:REC0:** значение счетчика ошибок приемника
Когда счетчик переполняется, узел отключается от шины. Если на шине будет последовательно передано 11 ‘recessive’ бит, значение счетчика сбросится.

11.4 Фильтры сообщений

RXF0SIDH: идентификатор стандартного сообщения фильтра 0, старший байт
RXF1SIDH: идентификатор стандартного сообщения фильтра 1, старший байт
RXF2SIDH: идентификатор стандартного сообщения фильтра 2, старший байт
RXF3SIDH: идентификатор стандартного сообщения фильтра 3, старший байт
RXF4SIDH: идентификатор стандартного сообщения фильтра 4, старший байт
RXF5SIDH: идентификатор стандартного сообщения фильтра 5, старший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
SID10	**SID9**	**SID8**	**SID7**	**SID6**	**SID5**	**SID4**	**SID3**
бит7							бит0

бит 7-0: **SID10:SID3:** биты идентификатора фильтра стандартного сообщения

RXF0SIDL: идентификатор стандартного сообщения фильтра 0, младший байт
RXF1SIDL: идентификатор стандартного сообщения фильтра 1, младший байт
RXF2SIDL: идентификатор стандартного сообщения фильтра 2, младший байт
RXF3SIDL: идентификатор стандартного сообщения фильтра 3, младший байт
RXF4SIDL: идентификатор стандартного сообщения фильтра 4, младший байт
RXF5SIDL: идентификатор стандартного сообщения фильтра 5, младший байт

R/W - x	R/W - x	R/W - x	U - 0	R/W - x	U - 0	R/W - x	R/W - x
SID2	**SID1**	**SID0**	**-**	**EXIDEN**	**-**	**EID17**	**EID16**
бит7							бит0

бит 7-5: **SID2:SID0:** биты идентификатора фильтра стандартного сообщения

бит 4: **Не используется:** читается как ‘0’

бит 3: **EXIDEN:** фильтр расширенных сообщений
1 = фильтр расширенных сообщений (идентификатор 29 бит)
0 = фильтр стандартных сообщений (идентификатор 11 бит)

бит 2: **Не используется:** читается как ‘0’

бит 1-0: **EID17:EID16:** биты идентификатора фильтра расширенного сообщения

RXF0EIDH: идентификатор расширенного сообщения фильтра 0, старший байт
RXF1EIDH: идентификатор расширенного сообщения фильтра 1, старший байт
RXF2EIDH: идентификатор расширенного сообщения фильтра 2, старший байт
RXF3EIDH: идентификатор расширенного сообщения фильтра 3, старший байт
RXF4EIDH: идентификатор расширенного сообщения фильтра 4, старший байт
RXF5EIDH: идентификатор расширенного сообщения фильтра 5, старший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
EID15	**EID14**	**EID13**	**EID12**	**EID11**	**EID10**	**EID9**	**EID8**
бит7							бит0

бит 7-0: **EID15:EID8:** биты идентификатора фильтра расширенного сообщения

RXF0EIDL: идентификатор расширенного сообщения фильтра 0, младший байт
RXF1EIDL: идентификатор расширенного сообщения фильтра 1, младший байт
RXF2EIDL: идентификатор расширенного сообщения фильтра 2, младший байт
RXF3EIDL: идентификатор расширенного сообщения фильтра 3, младший байт
RXF4EIDL: идентификатор расширенного сообщения фильтра 4, младший байт
RXF5EIDL: идентификатор расширенного сообщения фильтра 5, младший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
EID7	**EID6**	**EID5**	**EID4**	**EID3**	**EID2**	**EID1**	**EID0**
бит7							бит0

бит 7-0: **EID7:EID0:** биты идентификатора фильтра расширенного сообщения

RXM0SIDH: идентификатор стандартного сообщения маски 0, старший байт
RXM1SIDH: идентификатор стандартного сообщения маски 1, старший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
SID10	**SID9**	**SID8**	**SID7**	**SID6**	**SID5**	**SID4**	**SID3**
бит7							бит0

бит 7-0: **SID10:SID3:** биты идентификатора маски стандартного сообщения

RXM0SIDL: идентификатор стандартного сообщения маски 0, младший байт
RXM1SIDL: идентификатор стандартного сообщения маски 1, младший байт

R/W - x	R/W - x	R/W - x	U - 0	U - 0	U - 0	R/W - x	R/W - x
SID2	**SID1**	**SID0**	**-**	**-**	**-**	**EID17**	**EID16**
бит7							бит0

бит 7-5: **SID2:SID0:** биты идентификатора маски стандартного сообщения

бит 4-2: **Не используется:** читается как ‘0’

бит 1-0: **EID17:EID16:** биты идентификатора маски расширенного сообщения

RXM0EIDH: идентификатор расширенного сообщения маски 0, старший байт
RXM1EIDH: идентификатор расширенного сообщения маски 1, старший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
EID15	**EID14**	**EID13**	**EID12**	**EID11**	**EID10**	**EID9**	**EID8**
бит7							бит0

бит 7-0: **EID15:EID8:** биты идентификатора маски расширенного сообщения

RXM0EIDL: идентификатор расширенного сообщения маски 0, младший байт
RXM1EIDL: идентификатор расширенного сообщения маски 1, младший байт

R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x	R/W - x
EID7	**EID6**	**EID5**	**EID4**	**EID3**	**EID2**	**EID1**	**EID0**
бит7							бит0

бит 7-0: **EID7:EID0:** биты идентификатора маски расширенного сообщения

11.5 Регистры, управляющие скоростью передачи

BRGCON1: первый управляющий регистр

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
SJW1	**SJW0**	**BRP5**	**BRP4**	**BRP3**	**BRP2**	**BRP1**	**BRP0**
бит7							бит0

бит 7-6: **SJW1:SJW0:** переход синхронизации
11 = 4 x Tq
10 = 3 x Tq
01 = 2 x Tq
00 = 1 x Tq

бит 5-0: **BRP5:BRP0:** предварительный делитель генератора скорости
11111 = Tq = (2 x 64)/Fosc
11110 = Tq = (2 x 63)/Fosc
:
:
00001 = Tq = (2 x 2)/Fosc
00000 = Tq = (2 x 1)/Fosc

BRGCON2: второй управляющий регистр

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
SEG2PHTS	**SAM**	**SEG1PH2**	**SEG1PH1**	**SEG1PH0**	**PRSEG2**	**PRSEG1**	**PRSEG0**
бит7							бит0

бит 7: **SEG2PHTS:** фаза сегмента 2
1 = свободное программирование
0 = максимальное PHEG1 или IPT

бит 6: **SAM:** выборка данных с CAN линии
1 = 3 точки выборки
0 = 1 точка выборки

бит 5-3: **SEG1PH2:SEG1PH0:** фаза сегмента 1
111 = 8 x Tq
110 = 7 x Tq
101 = 6 x Tq
100 = 5 x Tq
011 = 4 x Tq
010 = 3 x Tq
001 = 2 x Tq
000 = 1 x Tq

бит 2-0: **PRSEG2:PRSEG0:** выбор времени распространения
111 = 8 x Tq
110 = 7 x Tq
101 = 6 x Tq
100 = 5 x Tq
011 = 4 x Tq
010 = 3 x Tq
001 = 2 x Tq
000 = 1 x Tq

BRGCON3: третий управляющий регистр

U - 0	R/W - 0	U - 0	U - 0	U - 0	R/W - 0	R/W - 0	R/W - 0
-	**WAKFIL**	**-**	**-**	**-**	**SEG2PH2**	**SEG2PH1**	**SEG2PH0**
бит7							бит0

бит 7: **Не используется:** читается как ‘0’

бит 6: **WAKFIL:** пробуждение микроконтроллера при активности на шине CAN
1 = использовать пробуждение
0 = не использовать пробуждение

бит 5-3: **Не используется:** читается как ‘0’

бит 2-0: **SEG2PH2:SEG2PH0:** фаза сегмента 2
111 = 8 x Tq
110 = 7 x Tq
101 = 6 x Tq
100 = 5 x Tq
011 = 4 x Tq
010 = 3 x Tq
001 = 2 x Tq
000 = 1 x Tq

11.6 Регистры управления и прерываний

CIOCON: управление портами ввода/вывода

R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0	U - 0	U - 0	U - 0
TX1SRC	**TX1EN**	**ENDRHI**	**CANCAP**	**-**	**-**	**-**	**-**
бит7							бит0

бит 7: **TX1SRC:** назначение вывода TX1
1 = вывод тактового сигнала CAN
0 = вывод данных TXD

бит 6: **TX1EN:** включение вывода TX1
1 = вывод тактового сигнала CAN или данных TXD
0 = функционирует как цифровой порт ввода/вывода

бит 5: **ENDRHI:** включение подтягивающих резисторов
1 = TX0, TX1 подтягивающие резисторы к Vdd при ‘recessive’ бите
0 = TX0, TX1 третье состояние вывода при ‘recessive’ бите

бит 4: **CANCAP:** прием сообщений
1 = разрешить
0 = запретить

бит 3-0: **Не используется:** читается как ‘0’

PIR3: флаги прерываний от CAN модуля

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
IRXIF	**WAKIF**	**ERRIF**	**TXB2IF**	**TXB1IF**	**TXB0IF**	**RXB1IF**	**RXB0IF**
бит7							бит0

бит 7: **IRXIF:** флаг неправильно принятого сообщения
1 = принято сообщение с ошибкой
0 = принято сообщение без ошибок

бит 6: **WAKIF:** флаг активности на шине
1 = обнаружена активность на шине
0 = активности на шине нет

бит 5: **ERRIF:** флаг ошибки на CAN шине
1 = обнаружена ошибка
0 = нет ошибок

бит 4: **TXB2IF:** флаг завершения передачи данных из буфера 2
1 = данные переданы полностью, могут быть загружены новые
0 = передача данных не завершена

бит 3: **TXB1IF:** флаг завершения передачи данных из буфера 1
1 = данные переданы полностью, могут быть загружены новые
0 = передача данных не завершена

бит 2: **TXB0IF:** флаг завершения передачи данных из буфера 0
1 = данные переданы полностью, могут быть загружены новые
0 = передача данных не завершена

бит 1: **RXB1IF:** флаг принятых данных в буфер 1
1 = принято новое сообщение
0 = новых данных нет

бит 0: **RXB0IF:** флаг принятых данных в буфер 0
1 = принято новое сообщение
0 = новых данных нет

PIE3: разрешение прерываний от CAN модуля

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
IRXIE	**WAKIE**	**ERRIE**	**TXB2IE**	**TXB1IE**	**TXB0IE**	**RXB1IE**	**RXB0IE**
бит7							бит0

бит 7: **IRXIE:** разрешение прерывания при неправильно принятом сообщении
1 = прерывание разрешено
0 = прерывание запрещено

бит 6: **WAKIE:** разрешение прерывания при активности на шине
1 = прерывание разрешено
0 = прерывание запрещено

бит 5: **ERRIE:** разрешение прерывания при ошибки на CAN шине
1 = прерывание разрешено
0 = прерывание запрещено

бит 4: **TXB2IE:** разрешение прерывания при завершении передачи данных из буфера 2
1 = прерывание разрешено
0 = прерывание запрещено

бит 3: **TXB1IE:** разрешение прерывания при завершении передачи данных из буфера 1
1 = прерывание разрешено
0 = прерывание запрещено

бит 2: **TXB0IE:** разрешение прерывания при завершении передачи данных из буфера 0
1 = прерывание разрешено
0 = прерывание запрещено

бит 1: **RXB1IE:** разрешение прерывания при новых принятых данных в буфер 1
1 = прерывание разрешено
0 = прерывание запрещено

бит 0: **RXB0IE:** разрешение прерывания при новых принятых данных в буфер 0
1 = прерывание разрешено
0 = прерывание запрещено

IPR3: приоритет прерываний от CAN модуля

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
IRXIP	**WAKIP**	**ERRIP**	**TXB2IP**	**TXB1IP**	**TXB0IP**	**RXB1IP**	**RXB0IP**
бит7							бит0

бит 7: **IRXIP:** приоритет прерывания при неправильно принятом сообщении
1 = высокий приоритет
0 = низкий приоритет

бит 6: **WAKIP:** приоритет прерывания при активности на шине
1 = высокий приоритет
0 = низкий приоритет

бит 5: **ERRIP:** приоритет прерывания при ошибки на CAN шине
1 = высокий приоритет
0 = низкий приоритет

бит 4: **TXB2IP:** приоритет прерывания при завершении передачи данных из буфера 2
1 = высокий приоритет
0 = низкий приоритет

бит 3: **TXB1IP:** приоритет прерывания при завершении передачи данных из буфера 1
1 = высокий приоритет
0 = низкий приоритет

бит 2: **TXB0IP:** приоритет прерывания при завершении передачи данных из буфера 0
1 = высокий приоритет
0 = низкий приоритет

бит 1: **RXB1IP:** приоритет прерывания при новых принятых данных в буфер 1
1 = высокий приоритет
0 = низкий приоритет

бит 0: **RXB0IP:** приоритет прерывания при новых принятых данных в буфер 0
1 = высокий приоритет
0 = низкий приоритет

11.7 Карта памяти регистров управления CAN модуля

Адрес	Имя	Адрес	Имя	Адрес	Имя	Адрес	Имя
F7Fh	-	F5Fh	-	F3Fh	-	F1Fh	RXM1EIDL
F7Eh	-	F5Eh	CANSTATRO1	F3Eh	CANSTATRO3	F1Eh	RXM1EIDH
F7Dh	-	F5Dh	RXB1D7	F3Dh	TXB1D7	F1Dh	RXM1SIDL
F7Ch	-	F5Ch	RXB1D6	F3Ch	TXB1D6	F1Ch	RXM1SIDH
F7Bh	-	F5Bh	RXB1D5	F3Bh	TXB1D5	F1Bh	RXM0EIDL
F7Ah	-	F5Ah	RXB1D4	F3Ah	TXB1D4	F1Ah	RXM0EIDH
F79h	-	F59h	RXB1D3	F39h	TXB1D3	F19h	RXM0SIDL
F78h	-	F58h	RXB1D2	F38h	TXB1D2	F18h	RXM0SIDH
F77h	-	F57h	RXB1D1	F37h	TXB1D1	F17h	RXF5EIDL
F76h	TXERRCNT	F56h	RXB1D0	F36h	TXB1D0	F16h	RXF5EIDH
F75h	RXERRCNT	F55h	RXB1DLC	F35h	TXB1DLC	F15h	RXF5SIDL
F74h	COMSTAT	F54h	RXB1EIDH	F34h	TXB1EIDH	F14h	RXF5SIDH
F73h	CIOCON	F53h	RXB1EIDL	F33h	TXB1EIDL	F13h	RXF4EIDL
F72h	BRGCON3	F52h	RXB1SIDL	F32h	TXB1SIDL	F12h	RXF4EIDH
F71h	BRGCON2	F51h	RXB1SIDH	F31h	TXB1SIDH	F11h	RXF4SIDL
F70h	BRGCON1	F50h	RXB1CON	F30h	TXB1CON	F10h	RXF4SIDH
F6Fh	CANCON	F4Fh	-	F2Fh	-	F0Fh	RXF3EIDL
F6Eh	CANSTAT	F4Eh	CANSTATRO2	F2Eh	CANSTATRO4	F0Eh	RXF3EIDH
F6Dh	RXB0D7	F4Dh	TXB0D7	F2Dh	TXB2D7	F0Dh	RXF3SIDL
F6Ch	RXB0D6	F4Ch	TXB0D6	F2Ch	TXB2D6	F0Ch	RXF3SIDH
F6Bh	RXB0D5	F4Bh	TXB0D5	F2Bh	TXB2D5	F0Bh	RXF2EIDL
F6Ah	RXB0D4	F4Ah	TXB0D4	F2Ah	TXB2D4	F0Ah	RXF2EIDH
F69h	RXB0D3	F49h	TXB0D3	F29h	TXB2D3	F09h	RXF2SIDL
F68h	RXB0D2	F48h	TXB0D2	F28h	TXB2D2	F08h	RXF2SIDH
F67h	RXB0D1	F47h	TXB0D1	F27h	TXB2D1	F07h	RXF1EIDL
F66h	RXB0D0	F46h	TXB0D0	F26h	TXB2D0	F06h	RXF1EIDH
F65h	RXB0DLC	F45h	TXB0DLC	F25h	TXB2DLC	F05h	RXF1SIDL
F64h	RXB1EIDL	F44h	TXB0EIDL	F24h	TXB2EIDH	F04h	RXF1SIDH
F63h	RXB1EIDH	F43h	TXB0EIDH	F23h	TXB2EIDL	F03h	RXF0EIDL
F62h	RXB1SIDL	F42h	TXB0SIDL	F22h	TXB2SIDL	F02h	RXF0EIDH
F61h	RXB1SIDH	F41h	TXB0SIDH	F21h	TXB2SIDH	F01h	RXF0SIDL
F60h	RXB1CON	F40h	TXB0CON	F20h	TXB2CON	F00h	RXF0SIDH

12. Заключение

Протокол CAN оптимизирован для систем, в которых должны передаваться относительно небольшое количество информации (по сравнению с Ethernet или USB, разработанные специально для больших объемов данных) к любому или всем узлам сети. Множественный доступ, с опросом состояния шины, позволяет каждому узлу получить доступ к шине с учетом приоритетов.

Не адресатная структура сообщений, позволяет организовать многоабонентскую доставку данных с сокращением трафика шины.

Быстрая устойчивая передача информации, с системой контроля ошибок, позволяет отключать неисправные узлы от шины, что гарантирует доставку критических по времени сообщений.

Преимущества CAN протокола позволили использовать его в автомобилестроении, автоматизации технологических процессов, медицине и других приложениях.

Статья основывается на технической документации DS39500a
компании Microchip Technology Incorporated, USA.

Уважаемые господа!

ООО «Микро-Чип» поставляет полную номенклатуру комплектующих фирмы **Microchip Technology Inc** и осуществляет качественную техническую поддержку на русском языке.

С техническими вопросами Вы можете обращаться по адресу support@microchip.ru

По вопросам поставок комплектующих Вы можете обращаться к нам по телефонам:
(095) 963-9601
(095) 737-7545
и адресу sales@microchip.ru

На сайте

www.microchip.ru

Вы можете узнать последние новости нашей фирмы, найти техническую документацию и информацию по наличию комплектующих на складе.