СОДЕРЖАНИЕ ПЕРЕВОДА

8-bit **AVR** ® Microcontroller with 32K/64K/128K Bytes of ISP Flash and CAN Controller

AT90CAN32
AT90CAN64
AT90CAN128

Ред. 7679G–CAN–03/08

Перевод: Андрей Шлеенков
http://andromega.narod.ru
mailto:andromega@narod.ru

19. Контроллерная сеть CAN

Протокол контроллерных сетей CAN (Controller Area Network) является последовательным, широковещательным протоколом реального времени с очень высоким уровнем безопасности. Контроллер CAN в микроконтроллерах AT90CAN32/64/128 полностью совместим с Частью A и Частью B Спецификации CAN 2.0. Это предоставляет возможности, требуемые для реализации ядра протокола шины CAN в соответствии со Справочной моделью ISO/OSI:

- Уровень управления передачей данных
  - Подуровень управления логическим каналом (LLC – Logical Link Control)
  - Подуровень управления доступом к среде (MAC – Medium Access Control)
- Физический уровень
  - Подуровень физической сигнализации (Physical Signaling – PLS)
  - Не поддерживается – подсоединение к физической среде (Physical Medium Attach – PMA)
  - Не поддерживается – зависимый от среды интерфейс (Medium Dependent Interface – MDI)

Контроллер CAN может обрабатывать все типы кадров (данные, удаленные, ошибки и перегрузка) со скоростью обмена, достигающей 1 Mbit/s.

19.1 Основные характеристики

- Полный CAN контроллер
- Полное соответствие стандарту CAN ред. 2.0 A и 2.0 B
- 15 объектов сообщений (MOb – Message Object) с их собственными:
  - 11-битными тегами идентификаторов (ред. 2.0 A), 29-битными тегами идентификаторов (ред. 2.0 B)
  - 11-битными масками идентификаторов (ред. 2.0 A), 29-битными масками идентификаторов (ред. 2.0 B)
  - 8-байтными буферами данных (статически размещенными)
  - конфигурациями передачи (Tx), приема (Rx), буфера кадра или автоматического ответа
  - временными отметками
- Максимальная скорость передачи 1 Mbit/s на тактовой частоте 8 MHz
- Таймер TTC
- Режим прослушивания (для наблюдения или автоподстройки)

19.2 Протокол CAN

Протокол CAN является международным стандартом, определенным в ISO 11898 для высоких скоростей и в ISO 11519-2 для низких скоростей.

19.2.1 Принципы

Протокол CAN основан на механизме широковещательных коммуникаций. Широковещательные коммуникации достигаются использованием протокола передачи, ориентированного на сообщения. Сообщение идентифицируется при помощи своего идентификатора. Идентификатор должен быть уникален в пределах всей сети, что определяет не только содержание, но также и приоритет сообщения.

Приоритет передаваемого сообщения сравнивается с приоритетом другого менее срочного сообщения, имеющего свой идентификатор. Приоритеты устанавливаются при проектировании системы в форме соответствующих двоичных значений и не могут быть изменены динамически. Идентификатор с наименьшим двоичным числом имеет наивысший приоритет.

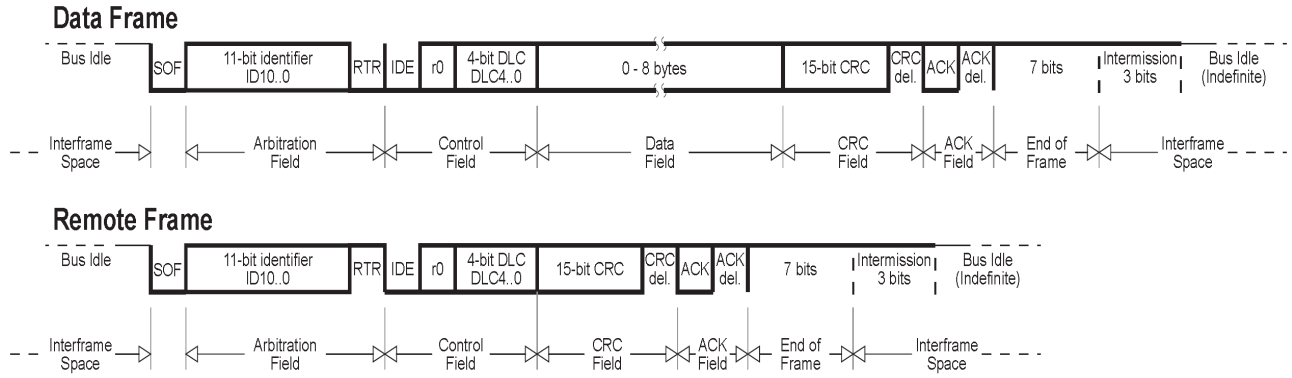
Конфликты доступа к шине разрешаются поразрядным арбитражем идентификаторов, используемых узлами, путем отслеживания уровня бита на шине. Это основано на механизме "монтажное И", при котором доминантное состояние подавляет рецессивное состояние. Соревнование за шину проигрывается всеми узлами с рецессивной передачей и доминирующим наблюдением. Все "проигравшие" автоматически становятся получателями сообщения с самым высоким приоритетом и не повторяют попытку передачи до тех пор, пока шина снова не станет доступна.

19.2.2 Форматы сообщений

Протокол CAN поддерживает два формата кадра сообщения с одним основным отличием, заключающемся в разной длине идентификатора. Стандартный кадр CAN, также известный как CAN 2.0 A, поддерживает длину идентификатора 11 бит. Расширенный кадр CAN, также известный как CAN 2.0 B, поддерживает длину идентификатора 29 бит.

19.2.2.1 Стандартный кадр CAN

Рисунок 19-1. Стандартные кадры CAN



Сообщение в формате стандартного кадра CAN начинается со старта кадра "Start Of Frame (SOF)", сопровождаемого полем арбитража "Arbitration field", состоящим из идентификатора и бита удаленного запроса передачи "Remote Transmission Request (RTR)", используемого для различения кадра данных и кадра запроса данных, вызываемых удаленным кадром.

Следующее поле управления "Control field" содержит бит расширения идентификатора "IDentifier Extension (IDE)" и код длины данных "Data Length Code (DLC)", указывающий число последующих байтов в поле данных "Data field". В удаленном кадре DLC содержит число запрашиваемых байтов данных. Следующее поле данных может содержать до 8 байтов данных.

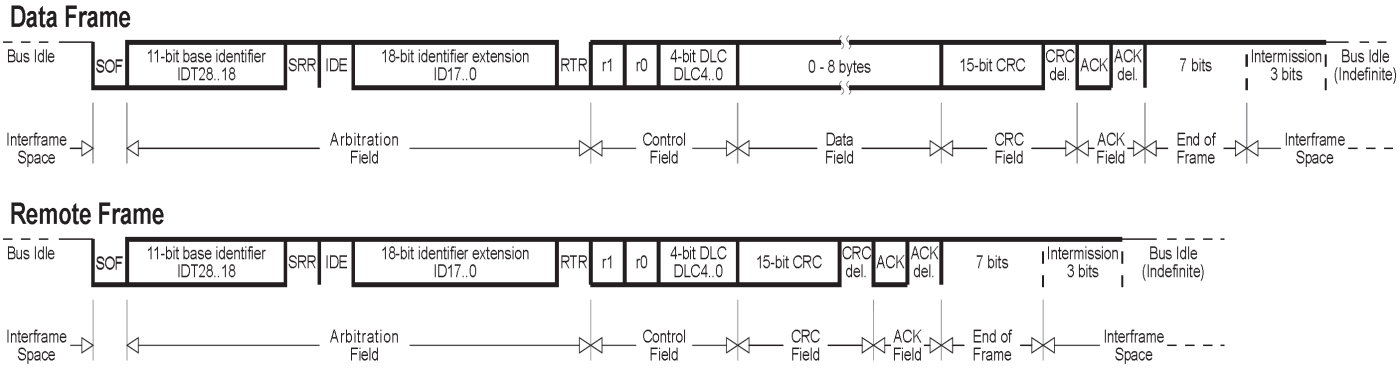
Целостность кадра обеспечивается проверкой следующей контрольной суммы циклического избыточного кода "Cyclic Redundant Check (CRC)".

Поле подтверждения "ACKnowledge (ACK) field" совмещает промежуток ACK и разделитель ACK. Бит в промежутке ACK посылается как рецессивный бит и перекрывается как доминантный бит получателями, получившими в данный момент правильные данные. Корректные сообщения подтверждаются получателями независимо от результата проверки приема.

Конец сообщения обозначается концом кадра "End Of Frame (EOF)". Межкадровый промежуток "Intermission Frame Space (IFS)" является минимальным числом бит, отделяющих последовательные сообщения. Если далее ни один узел не пытается получить доступ к шине, шина остается в холостом состоянии.

19.2.2.2 Расширенный кадр CAN

Рисунок 19-2. Расширенные кадры CAN



Сообщение в формате расширенного кадра CAN подобно сообщению в формате стандартного кадра CAN. Различие состоит в длине используемого идентификатора. Идентификатор расширенного кадра состоит из основного 11-битного идентификатора и 18-битного расширения. Различение форматов стандартного и расширенного кадра CAN осуществляется при помощи бита IDE, который передается как доминантный бит в случае стандартного кадра CAN и как рецессивный бит в противном случае.

19.2.2.3 Сосуществование форматов

Поскольку два формата должны сосуществовать на одной шине, установлено правило, определяющее, какое из сообщений в разных форматах с тем же самым идентификатором / основным идентификатором будет иметь более высокий приоритет в случае конфликта на шине. У сообщения в стандартном формате кадра всегда есть приоритет перед сообщением в расширенном формате.

Существует три различных типа доступных модулей CAN:

- 2.0A – рассматривает идентификатор из 29 бит как ошибку
- 2.0B пассивный – игнорирует сообщения с идентификатором из 29 бит
- 2.0B активный – поддерживает оба вида сообщений с идентификаторами из 11 и 29 бит

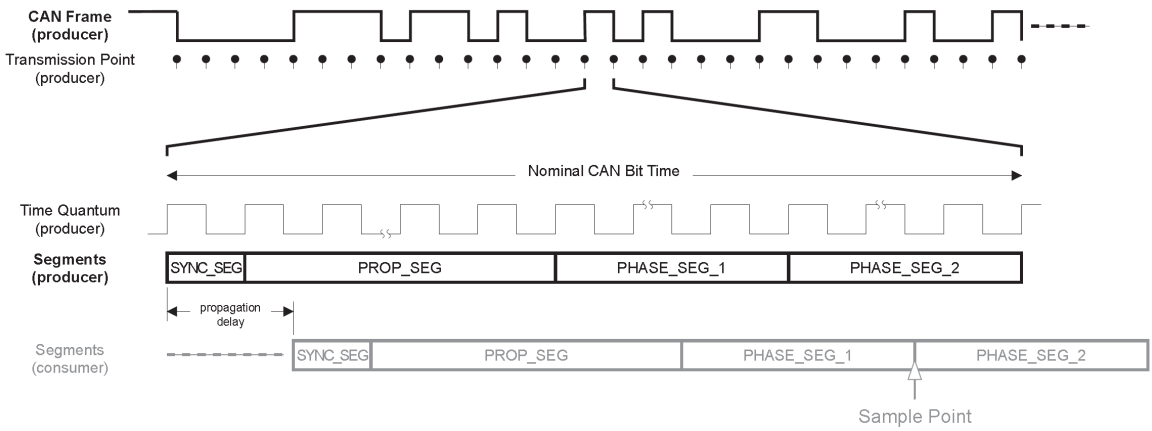
19.2.3 Битовая синхронизация CAN

Чтобы гарантировать правильную выборку вплоть до последнего бита, узел CAN должен ресинхронизироваться в течение всего кадра. Это делается в начале каждого сообщения по спадающему фронту SOF и по каждому фронту перехода от рецессивного уровня к доминантному уровню.

19.2.3.1 Структура бита

Один бит CAN определяется как четыре неперекрывающихся сегмента времени. Каждый сегмент состоит из целого числа квантов времени. Квант времени (TQ) представляет наименьшее дискретное временное разрешение, используемое узлом CAN.

Рисунок 19-3. Структура бита CAN



19.2.3.2 Сегмент синхронизации

Первый сегмент используется для синхронизации разных узлов шины.

При передаче в начале этого сегмента выводится текущий уровень бита. Если предыдущий бит отличается от текущего бита, то получающие узлы ожидают изменение состояния шины в пределах этого сегмента.

19.2.3.3 Сегмент времени распространения

Этот сегмент используется для компенсации задержки сигнала в сети.

Необходимо компенсировать задержки распространения сигнала по линиям шины и через приемопередатчики узлов шины.

19.2.3.4 Сегмент фазы 1

Сегмент фазы 1 используется для компенсации ошибок фазы фронта.

Этот сегмент может быть удлинен во время ресинхронизации.

19.2.3.5 Точка выборки

Точка выборки – это момент времени, в котором уровень шины считывается и интерпретируется как значение соответствующего бита. Она находится в конце сегмента фазы 1 (между двумя сегментами фазы).

19.2.3.6 Сегмент фазы 2

Сегмент фазы 2 также используется для компенсации ошибок фазы фронта.

Этот сегмент может быть сокращен во время ресинхронизации, но его длина должна быть не менее времени обработки информации (Information Processing Time - IPT), но не больше, чем длина сегмента фазы 1.

19.2.3.7 Время обработки информации

Это время требуется логике для определения уровня выбранного бита.

IPT в Atmel CAN начинается в точке выборки, измеряется за TQ и фиксируется за 2TQ. Так как сегмент фазы 2 также начинается в точке выборки и является последним сегментом при передаче бита, минимальный PS2 не должен быть меньше, чем IPT.

19.2.3.8 Удлинение бита

В результате ресинхронизации для компенсации допуска генератора может быть удлинен сегмент фазы 1 или сокращен сегмент фазы 2. Если, например, генератор передатчика медленнее, чем генератор получателя, следующий спадающий фронт, используемый для ресинхронизации, может быть задержан. Сегмент фазы 1 может быть удлинен для корректировки точки выборки и момента окончания бита.

19.2.3.9 Укорачивание бита

Если, с другой стороны, генератор передатчика быстрее, чем у получателя, следующий спадающий фронт, используемый для ресинхронизации, может быть слишком ранним. Сегмент фазы 2 в бите N может быть сокращен, для коррекции точки выборки для бита N+1 и момента окончания бита.

19.2.3.10 Ширина перехода ресинхронизации

Предел удлинения или сокращения сегментов фазы устанавливается шириной перехода ресинхронизации.

Этот сегмент не может быть длиннее сегмента фазы 2.

19.2.3.11 Программирование точки выборки

Программирование точки выборки позволяет настраивать характеристики устройств для удовлетворения требований шины.

Ранняя выборка позволяет кванты большей длительности в сегменте фазы 2, и таким образом ширина перехода синхронизации может быть запрограммирована на ее максимум. Большая способность сокращения или увеличения длительности бита уменьшает чувствительность к допускам генератора узла, что позволяет использовать генераторы низкой стоимости, такие как керамические резонаторы.

Поздняя выборка позволяет кванты большей длительности в сегменте времени распространения, что позволяет менее качественную топологию и максимальную длину шины.

19.2.3.12 Синхронизация

Жесткая синхронизация происходит на переходе от рецессивного к доминантному уровню стартового бита. Длительность бита перезапускается от этого фронта.

Ресинхронизация происходит тогда, когда в сообщении фронт перехода от рецессивного к доминантному уровню не происходит в пределах сегмента синхронизации.

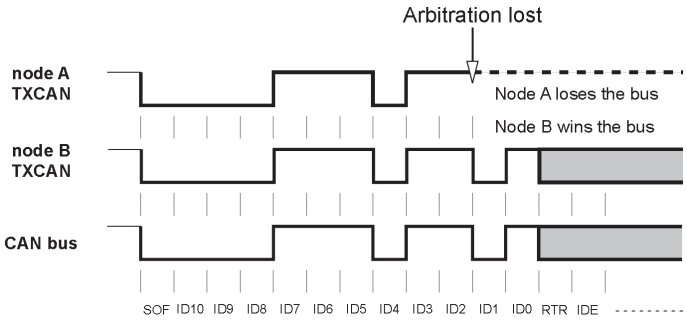
19.2.4 Арбитраж

Протокол CAN обрабатывает доступ к шине согласно концепции, называемой “множественный доступ с опросом несущей и с арбитражем по приоритету сообщения”.

При передаче арбитраж на шине CAN может быть потерян в пользу конкурирующего устройства с более высокоприоритетным идентификатором CAN. Эта концепция разрешает конфликт сообщений, передача которых была начата больше чем одним узлом одновременно и гарантирует, что наиболее важное сообщение будет передано первым без потери времени.

Конфликт доступа к шине разрешается в поле арбитража в основном значением идентификатора. Если кадр данных и удаленный кадр с тем же самым идентификатором инициализируются одновременно, кадр данных будет превалировать над удаленным кадром (бит RTR).

Рисунок 19-4. Арбитраж шины



19.2.5 Ошибки

При возникновении любой ошибки протокол CAN немедленно о ней сигнализирует. Реализованы три механизма обнаружения ошибок на уровне сообщения и два на уровне битов:

19.2.5.1 Ошибки уровня сообщения

- Контроль циклическим избыточным кодом (CRC)

CRC защищает информацию кадра, добавляя избыточные контрольные разряды в конец передачи. Получатель повторно вычисляет эти биты и сверяет с полученными. Если они не совпадают, произошла ошибка CRC.

- Проверка кадра

Этот механизм верифицирует структуру переданного кадра, сверяя битовые поля с фиксированным форматом и размером кадра. Ошибки, обнаруженные проверкой кадра, обозначаются как "ошибки формата".

- Ошибки подтверждения

Как упоминалось, полученные кадры подтверждаются всеми получателями позитивным подтверждением. Если передатчиком сообщения не было получено ни одно подтверждение, индицируется ошибка подтверждения.

19.2.5.2 Ошибки уровня битов

- Мониторинг

Способность передатчика обнаруживать ошибки основана на контроле сигналов шины. Каждый передающий узел также наблюдает уровень шины и таким образом обнаруживает различия между посланным и полученным битом. Это позволяет надежно обнаруживать глобальные ошибки и локальные ошибки передатчика.

- Битовое заполнение

Кодирование отдельных битов проверяется на разрядном уровне. CAN использует битовое представление типа "невозвращение к нулю (NRZ)", которое гарантирует максимальную эффективность битового кодирования. Фронты синхронизации генерируются посредством заполнения битами.

19.2.5.3 Сигнализация ошибок

Если хотя бы одним узлом обнаруживается одна или более ошибок, используя вышеупомянутые механизмы, текущая передача прерывается посылкой "флага ошибки". Это защищает другие узлы, принимающие сообщение, и этим гарантирует непротиворечивость данных в целом по сети. После передачи ошибочного сообщения, которое было прервано, отправитель автоматически делает повторную попытку передачи.

19.3 Контроллер CAN

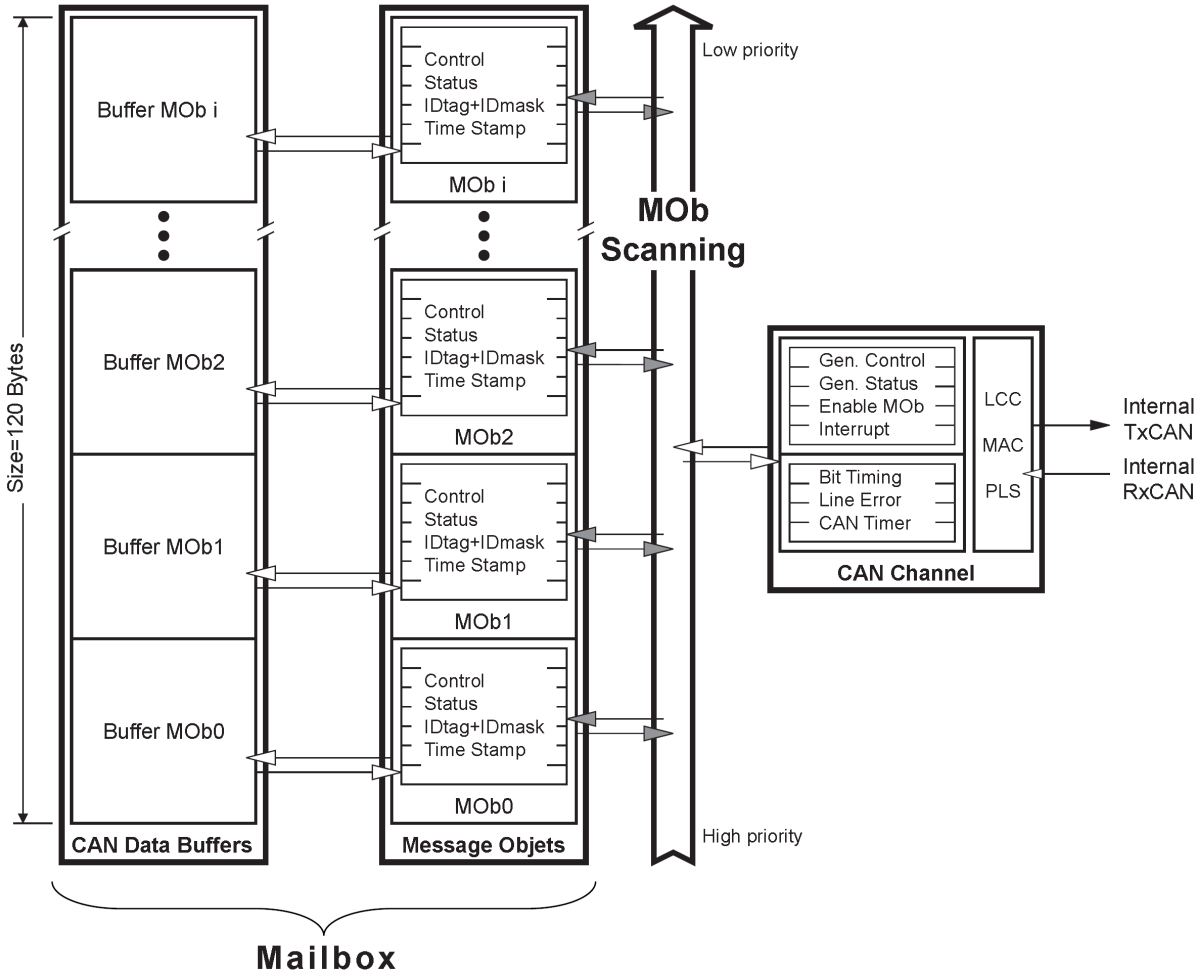
В AT90CAN32/64/128 реализован контроллер CAN V2.0B Active.

Данный контроллер CAN предоставляет полные аппаратные средства для удобной фильтрации приема и управления сообщениями. Для каждого передаваемого или получаемого сообщения этот модуль содержит один так называемый объект сообщения, в котором хранится вся информация, относящаяся к сообщению (например, идентификатор, байты данных и т.д.).

При инициализации модуля приложение определяет, какие сообщения посылать и какие получать. Если контроллер CAN получает сообщение с идентификатором, соответствующим одному из объектов сообщений, запрограммированных для приема, принятое сообщение сохраняется и приложение получает прерывание. Другое преимущество состоит в том, что полный контроллер CAN может автоматически ответить входящим удаленным кадрам, соответствующим кадром данных, что по сравнению с базовым решением резко уменьшает нагрузку на центральный процессор.

Используя полный контроллер CAN, может быть обработано множество сообщений с высокими скоростями и высокой нагрузкой на шину.

Рисунок 19-5. Структура контроллера CAN



Перевод: Андрей Шлеенков, http://andromega.narod.ru, mailto:andromega@narod.ru 7679G–CAN–03/08

19.4 Канал CAN

19.4.1 Конфигурация

Канал CAN может находиться в следующих режимах:

- Режим разрешения

  В этом режиме:

  – канал CAN разрешен (внутренние TxCAN и RxCAN),
  – входная синхронизация разрешена.

- Режим ожидания:

  В этом режиме:

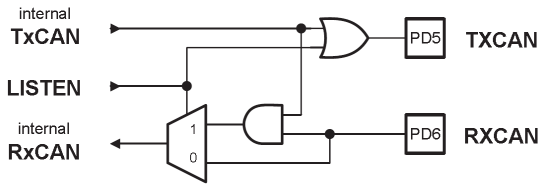
  – передатчик постоянно поддерживает рецессивный уровень (на внутреннем TxCAN), а приемник заблокирован,
  – входная синхронизация разрешена,
  – регистры и страницы остаются доступными.

- Режим прослушивания

  Этот режим прозрачен для канала CAN:

  – разрешена аппаратная обратная связь, внутренний TxCAN на внутреннем RxCAN,
  – на выводе выхода TXCAN поддерживается рецессивный уровень,
  – не запрещается вывод входа RXCAN,
  – замораживаются счетчики ошибок TEC и REC.

Рисунок 19-6. Режим прослушивания



19.4.2 Битовая синхронизация

Конечный автомат канала CAN должен быть синхронен квантам времени. Поэтому входной частотой для битовой синхронизации является частота, используемая в конечном автомате канала CAN.

Сокращения для имен полей и сегментов:

- BRP: Baud Rate Prescaler – предделитель скорости в бодах.
- TQ: Time Quantum – квант времени (выход предделителя).
- SYNS: SYNchronization Segment – сегмент синхронизации (1 TQ).
- PRS: PRopagation time Segment – сегмент времени распространения, (значения 1, 2, ..., 8 TQ).
- PHS1: PHase Segment 1 – сегмент фазы 1 (значения 1, 2, ..., 8 TQ).
- PHS2: PHase Segment 2 – сегмент фазы 2 (значения PHS2 ≤ PHS1 и PHS2 ≥ IPT).
- IPT: Information Processing Time – время обработки информации (2 TQ).
- SJW: (Re) Synchronization Jump Width – ширина перехода (ре) синхронизации (значения между 1 и min(4, PHS1)).

Общее число TQ в длине бита должно быть запрограммировано от 8 до 25.

Рисунок 19-7. Точка выборки и передачи

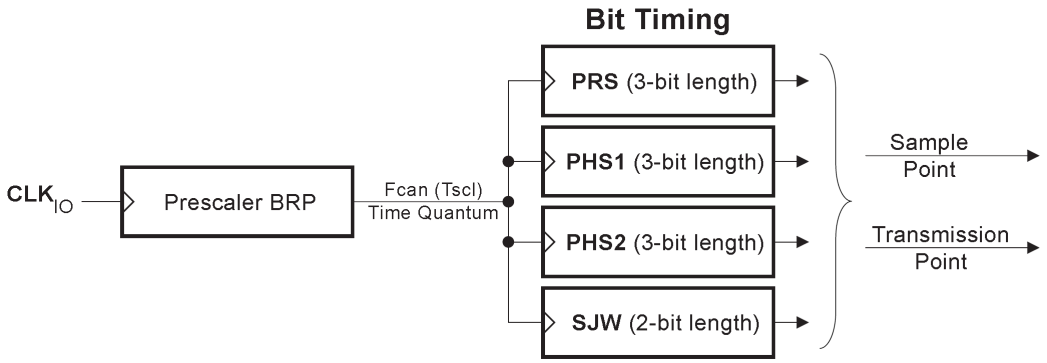
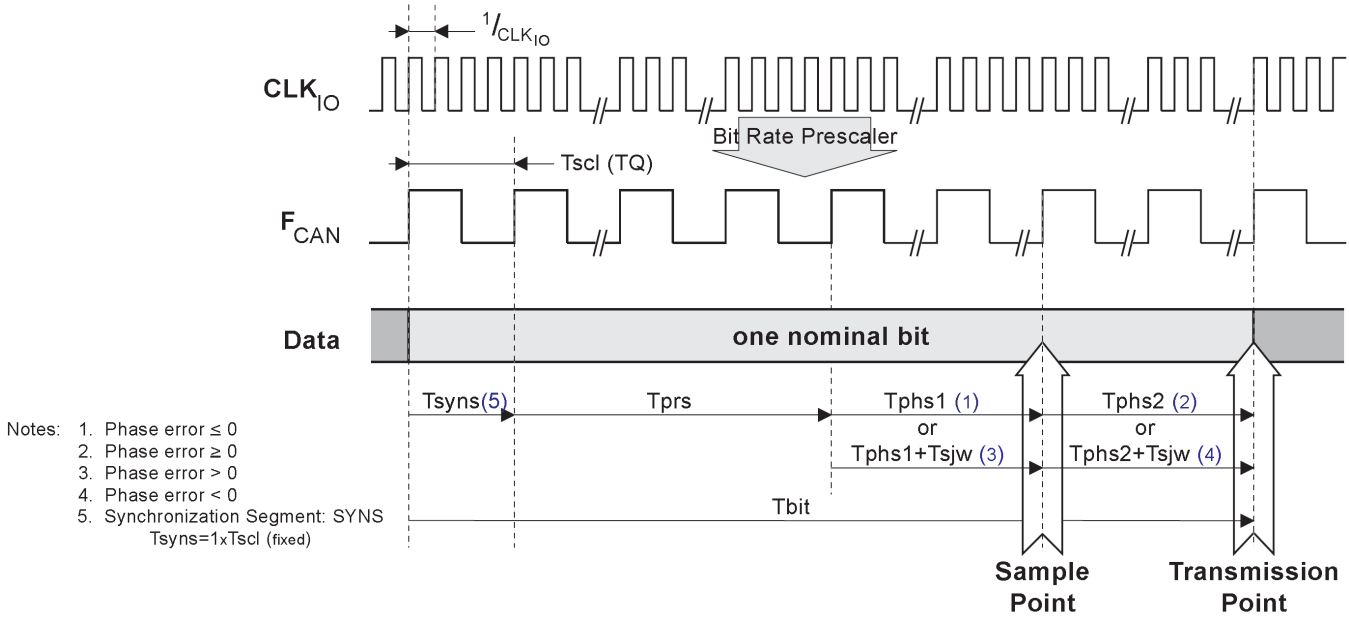


Рисунок 19-8. Общая структура периода бита



19.4.3 Скорость в бодах

Без предделителя скорости (BRP[5..0]=0) точка выборки становится одним квантом времени и появляется слишком рано. Это приводит к сбою согласно плану тестирования ISO16845. Для компенсации этого необходимо удлинить сегмент фазы на 1 и сократить сегмент фазы 2 на один квант времени.

Выбор скорости в бодах выполняется вычислением Tbit:

$$Tbit_{(1)} = Tsyns + Tprs + Tphs1 + Tphs2$$

1. Tsyns = 1 x Tscl = (BRP[5..0]+ 1)/clk_{IO} (= 1TQ)
2. Tprs = (1 to 8) x Tscl = (PRS[2..0]+ 1) x Tscl
3. Tphs1 = (1 to 8) x Tscl = (PHS1[2..0]+ 1) x Tscl
4. Tphs2 = (1 to 8) x Tscl = (PHS2[2..0]$_{(2)}$+ 1) x Tscl
5. Tsjw = (1 to 4) x Tscl = (SJW[1..0]+ 1) x Tscl

Замечания: 1. Общее число Tscl (квантов) в бите должно быть от 8 до 25.
2. PHS2[2..0] 2 программируется значением ≤ PHS1[2..0] и ≥ 1.

Перевод: Андрей Шлеенков, http://andromega.narod.ru, mailto:andromega@narod.ru 7679G–CAN–03/08

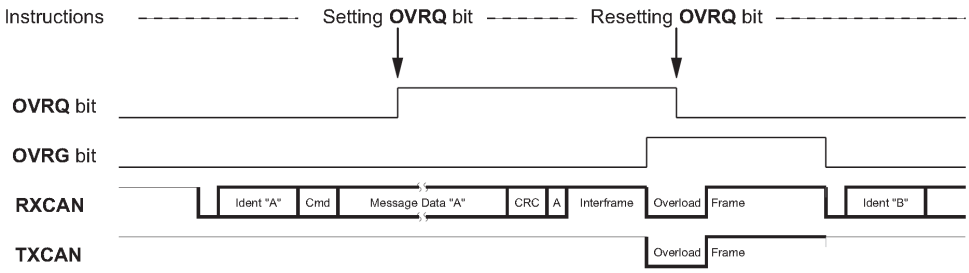
19.4.4 Подавление ошибок

(см. Раздел 19.7 ”Обработка ошибок” на стр. 248).

19.4.5 Кадр перегрузки

Кадр перегрузки посылается установкой запроса перегрузки (OVRQ). После следующего приема канал CAN посылает кадр перегрузки в соответствии со спецификацией CAN. Состояние или флаг будут установлены (OVRG) пока передается кадр перегрузки.

Рисунок 19-9. Кадр перегрузки



19.5 Объекты сообщений

Объект сообщения MOb является дескриптором кадра CAN и содержит всю информацию для обработки кадра. MOb выделен для возможности описания сообщения CAN как объекта. Набор объектов MOb – это часть пользовательского интерфейса "почтового ящика", где сообщения для пересылки и/или получения определены насколько это возможно для уменьшения нагрузки на программу.

Каждый MOb независим, но в случае множественного совпадения приоритет отдается более низкому. Режимы работы:

- Режим запрета
- Режим передачи
- Режим приема
- Режим автоматического ответа
- Режим буфера приема кадров

19.5.1 Число объектов сообщения

Данное устройство имеет 15 объектов MOb, нумеруемых от 0 до 14 (i=14, не MOb 15).

19.5.2 Режимы работы

После сброса (RESET) **НЕ** существует режима по умолчанию.

Каждый MOb имеет свои собственные поля для управления режимом работы. Прежде, чем разрешить модуль CAN, каждый MOb должен быть конфигурирован (исключая режим запрета – CONMOB=00).

Таблица 19-1. Конфигурация MOb

Конфигурация MOb		Ответ допустим	Тэг RTR	Режим работы
0	0	x	x	Запрет
0	1	x	0	Передача (Tx) кадра данных
		x	1	Передача (Tx) удаленного кадра
1	0	x	0	Прием (Rx) кадра данных
		0	1	Прием (Rx) удаленного кадра
		1		Прием (Rx) удаленного кадра и затем передача (Tx) кадра данных (ответ)
1	1	x	x	Буфер приема кадров

19.5.2.1 Запрет

В этом режиме MOb "свободен".

19.5.2.2 Передача данных и удаленного кадра

1. Перед посылкой должны быть инициализированы несколько полей:
   - Тэг идентификатора (IDT)
   - Расширение идентификатора (IDE)
   - Запрос удаленной передачи (RTRTAG)
   - Код длины данных (DLC)
   - Тэг зарезервированных битов (RBnTAG)
   - Байты данных сообщения (MSG)
2. Если конфигурация MOb установлена, MOb готов посылать данные или удаленный кадр (CONMOB).
3. Далее канал CAN просматривает все MOb в конфигурации Tx, находит MOb, имеющий самый высокий приоритет и пытается его послать.
4. Когда передача завершается, устанавливается флаг TXOK (прерывание).
5. Все параметры и данные доступны в MOb до новой инициализации.

19.5.2.3 Прием данных и удаленного кадра

1. Перед приемом должны быть инициализированы несколько полей:
   - Тэг идентификатора (IDT)
   - Маска идентификатора (IDMSK)
   - Расширение идентификатора (IDE)
   - Маска расширения идентификатора (IDEMSK)
   - Запрос удаленной передачи (RTRTAG)
   - Маска запроса удаленной передачи (RTRMSK)
   - Код длины данных (DLC)
   - Тэг зарезервированных битов (RBnTAG)

2. Если конфигурация MOb установлена, MOb готов принимать данные или удаленный кадр (CONMOB).
3. Если по сети получен идентификатор кадра, канал CAN просматривает все Mob в конфигурации Rx и пытается найти соответствующий MOb, имеющий самый высокий приоритет.
4. В случае успеха, IDT, IDE и DLC соответствующего MOb обновляются значениями из принятого кадра.
5. При завершении приема байты данных полученного сообщения сохраняются (но не для удаленного кадра) в буфере данных соответствующего MOb, и устанавливается флаг RXOK (прерывание).
6. Все параметры и данные доступны в MOb до новой инициализации.

19.5.2.4 Автоматический ответ

Ответ на удаленный кадр (кадр данных) можно автоматически послать после получения ожидаемого отдаленного кадра.

1. Перед получением удаленного кадра должны быть инициализированы несколько полей:
   – Ответ (RPLV) в идентичном потоке, описанном в Разделе 19.5.2.3 "Прием данных и удаленного кадра" на стр. 244.
2. Если удаленный кадр соответствует установкам, автоматически сбрасываются RTRTAG и бит правильный ответ (RPLV). В этот момент не устанавливается никаких флагов (или прерываний). Так как буфер данных CAN не использовался входящим удаленным кадром, то MOb в режиме передачи будет готов без дополнительных установок. Для ответа используются IDT, IDE, другие тэги и DLC полученного удаленного кадра.
3. По завершении передачи ответа устанавливается флаг TXOK (прерывание).
4. Все параметры и данные доступны в MOb до новой инициализации.

19.5.2.5 Режим буфера приема кадров

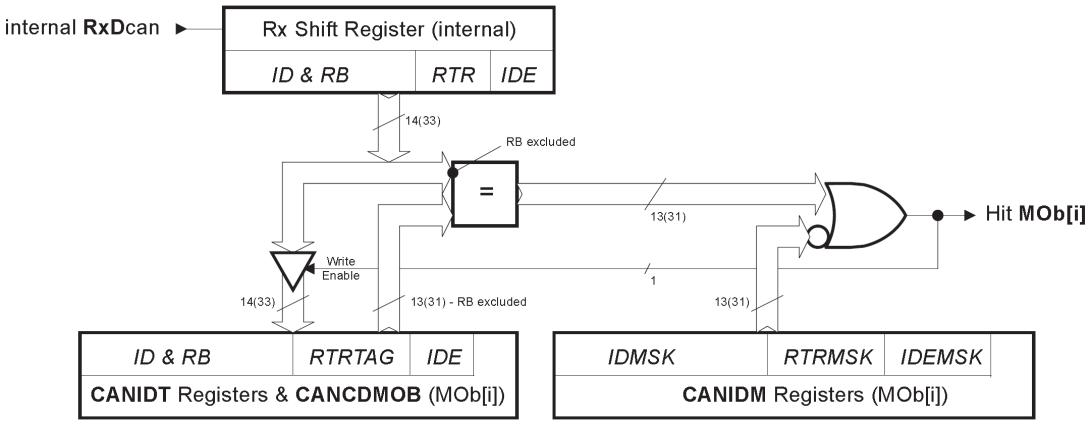
Этот режим полезен для получения множества кадров. Приоритет между MOb позволяет управлять входящими кадрами. Один набор MOb (включая непоследовательные MOb) создается когда объекты MOb устанавливаются в этом режиме. Из-за установки режима возможен только один набор. Флаг заполнения буфера кадра BXOK (или прерывание) установится только тогда, когда все MOb набора получат предназначенные им кадры.

1. MOb в режиме буфера приема кадров должны быть инициализированы как MOb в стандартном режиме приема.
2. MOb готовы принимать данные или удаленные кадры, когда установлены их соответствующие конфигурации (CONMOB).
3. Когда по сети получен идентификатор кадра, канал CAN просматривает все MOb в режиме приема, пытаясь найти соответствующий MOb, имеющий самый высокий приоритет.
4. При удаче, IDT, IDE и DLC соответствующего MOb обновляются значениями входящего кадра.
5. Как только прием завершен, байты данных полученного сообщения сохраняются (но не для удаленного кадра) в буфере данных соответствующего MOb и устанавливается флаг RXOK (прерывание).
6. Когда прием в последнем MOb набора завершен, устанавливается флаг заполнения буфера кадра BXOK (прерывание). Флаг BXOK может быть очищен, только если все поля CONMOB набора ранее перезаписаны.
7. Все параметры и данные доступны в MOb до новой инициализации.

19.5.3 Фильтр приема

При удачном приеме (то есть, положительные результаты сравнения ID + RTR + RBn + полученный IDE и IDT + RTRTAG + RBnTAG + ожидаемый IDE, учитывая маску сравнения) IDT + RTRTAG + RBnTAG + полученный IDE обновляются в MOb (перезапись регистров).

Рисунок 19-10. Блок-схема фильтра приема



Примеры:

Полная фильтрация: для приема только ID = 0x317 в части A.

- ID MSK = 111 1111 1111 $_b$
- ID TAG = 011 0001 0111 $_b$

Частичная фильтрация: для приема ID от 0x310 до 0x317 в части A.

- ID MSK = 111 1111 1000 $_b$
- ID TAG = 011 0001 0xxx $_b$

Без фильтрации: для приема ID от 0x000 до 0x7FF в части A.

- ID MSK = 000 0000 0000 $_b$
- ID TAG = xxx xxxx xxxx $_b$

19.5.4 Страница объекта сообщения

Каждый MOb отображается на страницу для экономии места. Номер страницы – это номер MOb. Номер страницы устанавливается в регистре CANPAGE. Номер 15 зарезервирован для заводских испытаний.

Регистр CANHPMOB задает MOb с самым высоким приоритетом в регистрах CANSIT. Он имеет формат, дающий прямой вход для регистра CANPAGE. Поскольку CANHPMOB кодирует регистры CANSIT, он будет обновлен только, если разрешены соответствующие биты разрешения (ENRX, ENTX, ENERR) (см. Рисунок 19-14).

19.5.5 Буферы данных CAN

Для экономии памяти регистров буфер данных CAN видится как FIFO (с доступным указателем адреса) при выборе MOb. Это также позволяет уменьшить риск бесконтрольного доступа.

Имеется по одному FIFO на MOb. Этот FIFO доступен в странице MOb благодаря регистру сообщений CAN.

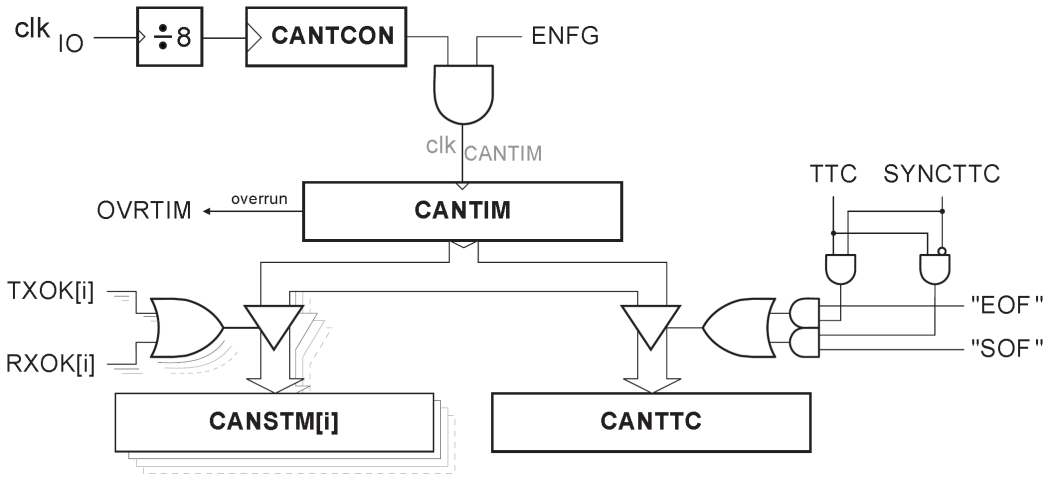
Индекс данных (INDX) – это указатель на требуемый байт данных. Байт данных может быть прочитан или записан. Индекс данных автоматически инкрементируется после каждого доступа, если бит AINC* сброшен. Индекс данных изменяется циклически – после 7 он принимает значение 0.

Первый байт кадра CAN сохраняется в данных по индексу 0, второй в данных по индексу 1 и т.д.

19.6 Таймер CAN

Программируемый 16-разрядный таймер используется для отметки сообщений и коммуникации по расписанию (TTC).

Рисунок 19-11. Блок-схема таймера CAN



19.6.1 Предделитель

8-битный предделитель инициализируется регистром CANTCON. Он получает частоту clk_{IO}, деленную на 8, и вы дает частоту clk_{CANTIM} таймеру CAN, если контроллер CAN разрешен.

$$Tclk_{CANTIM} = \mathbf{T}clk_{IO} \times 8 \times (CANTCON\ [7:0] + 1)$$

19.6.2 16-битный таймер

Таймер начинает счет от 0x0000 при разрешении контроллера CAN (бит ENFG). Когда таймер переполняется от 0xFFFF к 0x0000, генерируется прерывание (OVRTIM).

19.6.3 Синхронизация времени

Для TTC реализованы два режима синхронизации (бит TTC):

- синхронизация по началу кадра (SYNCTTC=0),
- синхронизация по концу кадра (SYNCTTC=1).

В режиме TTC кадр посылается однократно, даже если происходит ошибка.

19.6.4 Отметка сообщений

Отметка значения таймера выполняется в MOb, получающем или посылающем кадр. Все управляемые MOb отмечаются, отметка полученного (посланного) кадра происходит по RxOK (TXOK).

19.7 Обработка ошибок

19.7.1 Уменьшение ошибок

Канал CAN может находиться в одном из трех следующих состояний:

- Ошибка активная (по умолчанию):

  Канал CAN принимает участие в коммуникации по шине и может посылать активный кадр ошибки, когда макро CAN обнаруживает ошибку.

- Ошибка пассивная:

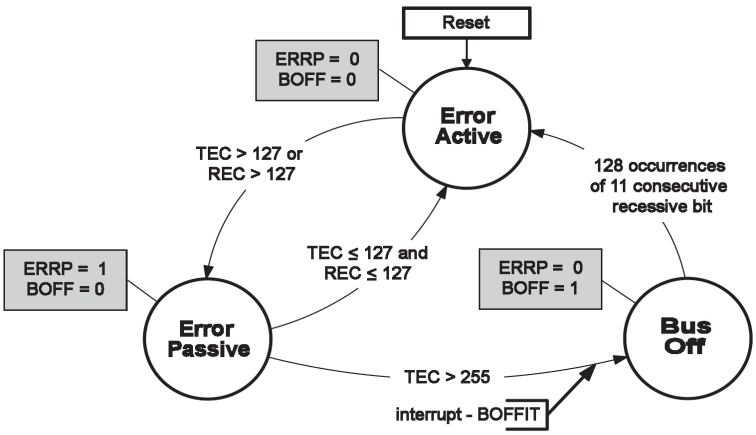
  Канал CAN не может посылать активный кадр ошибки. Он принимает участие в коммуникации по шине, но когда ошибка обнаруживается, он посылает пассивный кадр ошибки. Также после передачи пассивной ошибки модуль будет ждать инициализации дальнейшей передачи.

- Шина отключена:

  Каналу CAN не дозволено любое воздействие на шину.

Для уменьшения ошибок реализованы счетчик ошибок передачи (TEC) и счетчик ошибок приема (REC). Биты BOFF и ERRP дают информацию о состоянии канала CAN. Установка BOFF в 1 может вызывать прерывание.

Рисунок 19-12. Режим ошибки линии



Замечание: Во время передачи данного сообщения может происходить больше чем одно изменение REC/TEC.

19.7.2 Типы ошибок

- **BERR:** Ошибка бита. Значение контролируемого бита отличается от посланного значения.

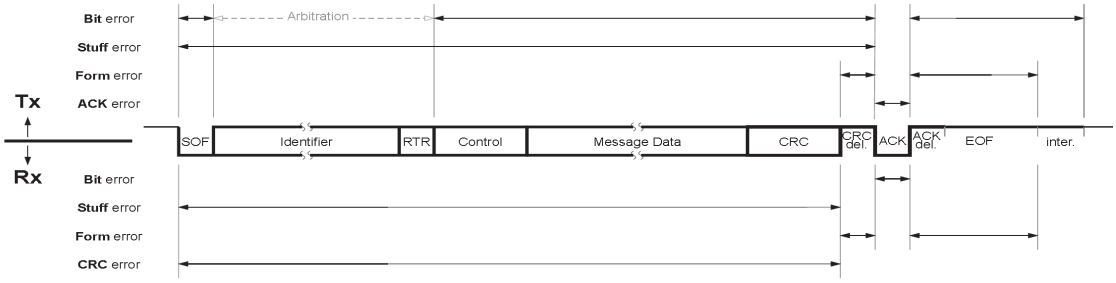
  Исключения:

  – Посланный рецессивный бит наблюдается как доминирующий бит в поле арбитража и промежутка подтверждения.
  – Обнаружение доминантного бита при посылке кадра ошибки.

- **SERR**: Ошибка заполнения. Обнаружение более пяти последовательных бит одной полярности.
- **CERR**: Ошибка CRC (только прием). Приемник выполняет проверку каждого принимаемого незаполненного сообщения от начала кадра до поля данных на соответствие CRC. Если проверка дает несоответствие незаполненного CRC, устанавливается ошибка CRC.

- **FERR**: Ошибка формы. Ошибка формы происходит из-за нарушения фиксированной формы одного или более следующих битовых полей:
  - разделитель CRC
  - разделитель подтверждения
  - конец кадра
  - разделитель ошибки
  - разделитель перегрузки
- **AERR**: Ошибка подтверждения (только передача). Доминантный бит не обнаружен в промежутке подтверждения.

Рисунок 19-13. Процедуры обнаружения ошибок в кадре данных



19.7.3 Обнаружение ошибок

Канал CAN может обнаружить некоторые ошибки в сети CAN.

- При передаче:

  Ошибка установлена на уровне MOb.

- При получении:
  - Идентификация успешна:

    Ошибка установлена на уровне MOb.

  - Идентификация не успешна или пока не успешна:

    Ошибка установлена на общем уровне.

После обнаружения ошибки канал CAN посылает по сети кадр ошибки. Если канал CAN обнаруживает в сети кадр ошибки, он посылает свой собственный кадр ошибки.

19.8 Прерывания

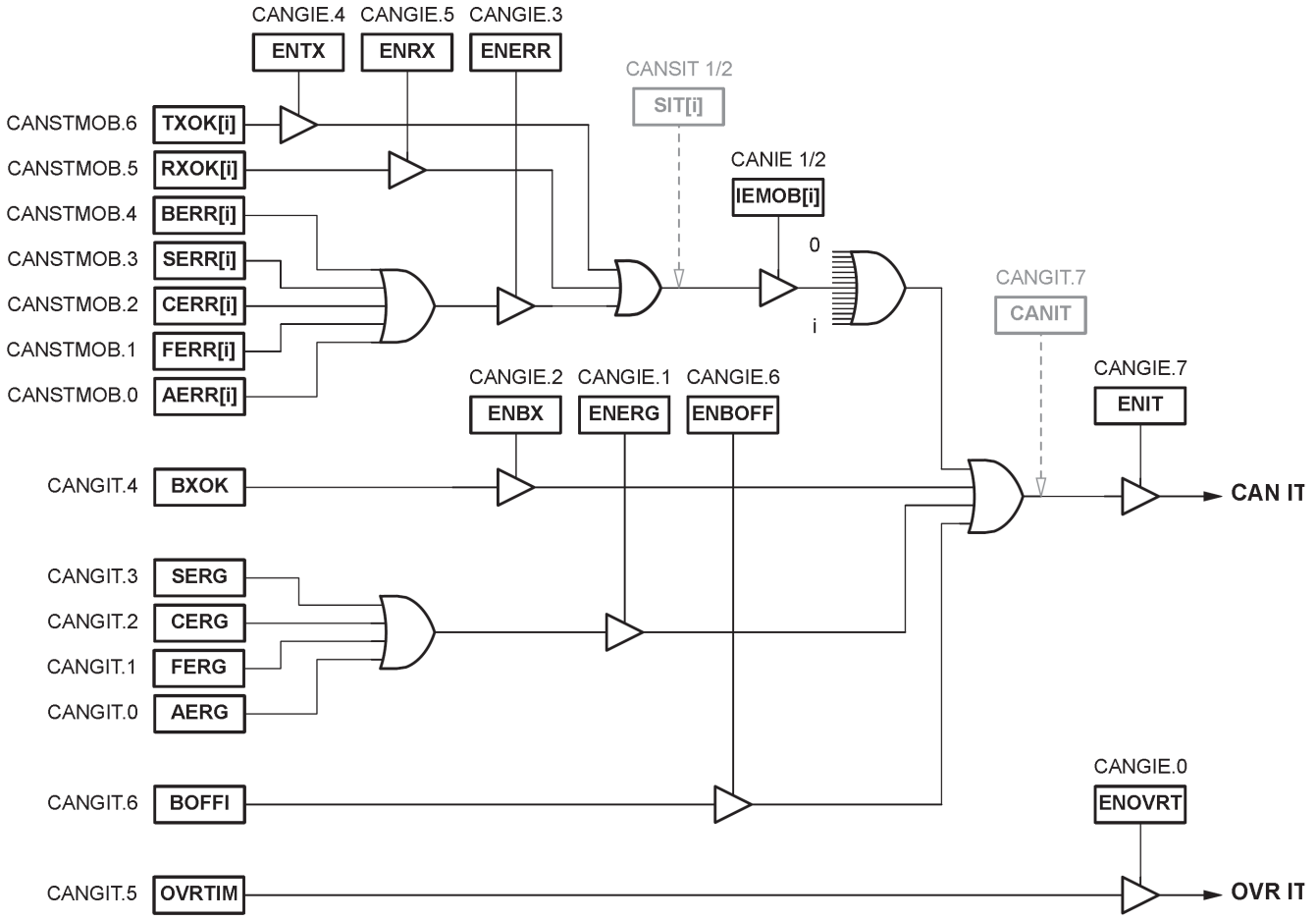
19.8.1 Организация прерываний

Имеются различные прерывания:

- Прерывание по завершению получения (OK),
- Прерывание по завершению передачи OK,
- Прерывание по обнаружению ошибки (ошибка бита, ошибка наполнения, ошибка CRC, ошибка формы, ошибка подтверждения),
- Прерывание по заполнению буфера кадра,
- Прерывание по отключению от шины (“Bus Off”),
- Прерывание по переполнению таймера CAN.

Общее разрешение прерываний определяется битом ENIT, а специфическое разрешение прерываний по переполнению таймера CAN определяется битом ENORVT.

Рисунок 19-14. Структура прерываний контроллера CAN



19.8.2 Поведение прерываний

При возникновении прерывания бит флага прерывания устанавливается в соответствующем регистре MOb-CANSTMOB или в общем регистре CANGIT. Если в регистре CANIE устанавливается бит ENRX / ENTX / ENERR, то в регистре CANSITn устанавливается соответствующий бит MOb.

Для подтверждения прерывания MOb соответствующие биты регистра CANSTMOB (RXOK, TXOK...) должны быть очищены приложением. Эта операция требует процедуры "чтение-модификация-запись".

Для подтверждения общего прерывания соответствующие биты регистра CANGIT (BXOK, BOFFIT...) должны быть очищены приложением. Эта операция выполняется записью логической 1 в эти флаги прерывания (запись логического нуля не изменяет значение флага прерывания).

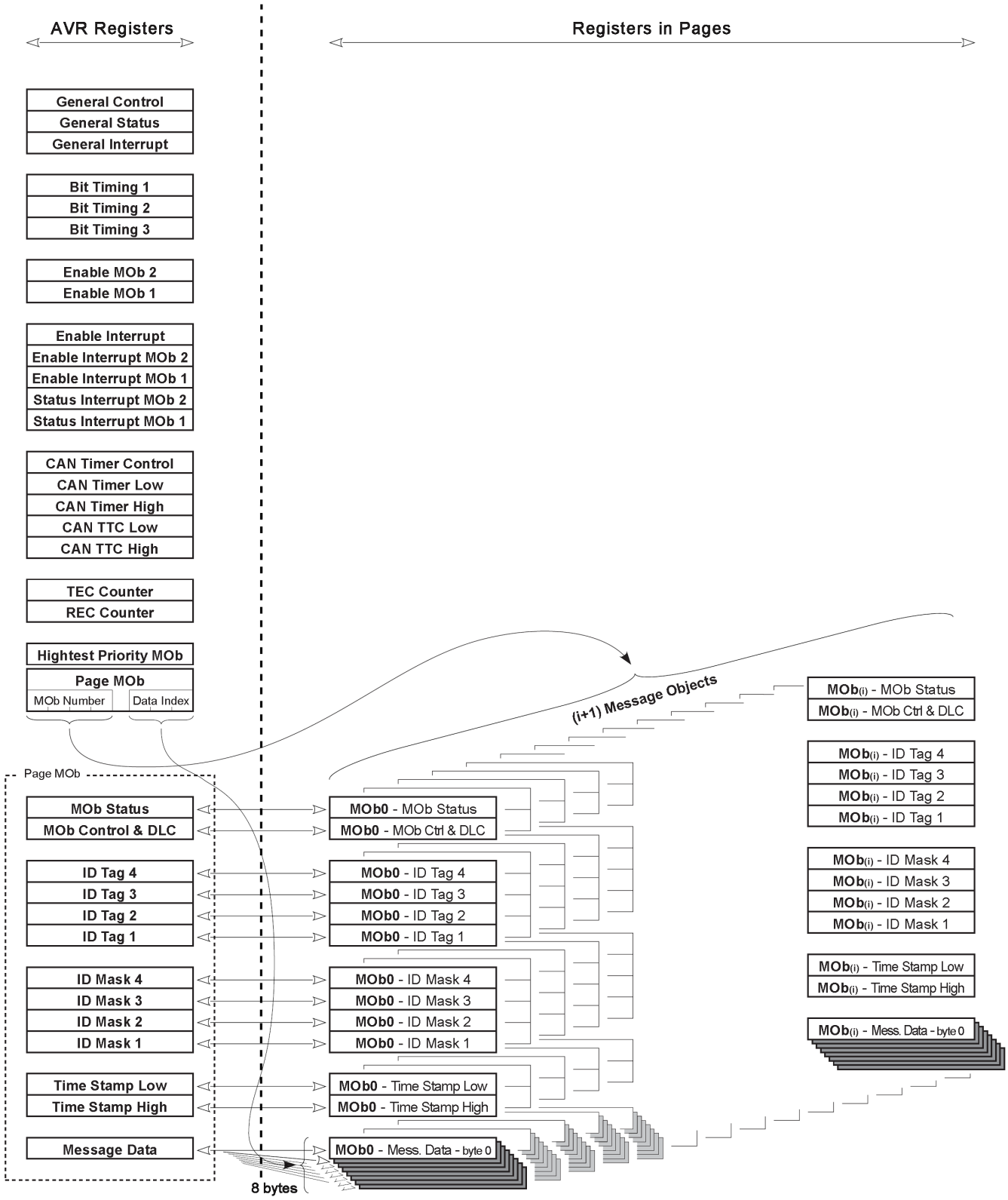
Флаг прерывания OVRTIM сбрасывается и как другие источники прерывания регистра CANGIT, и также при входе в его особый обработчик прерывания.

Когда узел CAN передает и обнаруживает в кадре ошибку формы, также будет установлена ошибка бита. Следовательно, из-за одной и той же ошибки могут произойти два последовательных прерывания.

Когда ошибка MOb и устанавливается в ее собственном регистре CANSTMOB, ни одна общая ошибка в регистре CANGIT не устанавливается.

19.9 Описание регистров CAN

Рисунок 19-15. Организация регистров



19.10 Общие регистры CAN

19.10.1 CANGCON: CAN General Control Register – общий регистр управления CAN

Bit	7	6	5	4	3	2	1	0	
	ABRQ	OVRQ	TTC	SYNTTC	LISTEN	TEST	ENA/$\overline{\text{STB}}$	SWRES	CANGCON
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ABRQ: Abort Request – запрос аварийного завершения**

Этот бит не сбрасывается автоматически.

  – 0 – нет запроса.
  – 1 – запрос аварийного завершения: сбрасываются регистры CANEN1 и CANEN2. Ожидающая обработки связь немедленно запрещается, а выполняющаяся завершится нормально, устанавливая соответствующие флаги состояния.

  Замечание: Регистр CANCDMOB не изменяется.

- **Bit 6 – OVRQ: Overload Frame Request – запрос кадра перегрузки**

Этот бит не сбрасывается автоматически.

  – 0 – нет запроса.
  – 1 – запрос кадра перегрузки: передача кадра перегрузки после следующего полученного кадра.

Кадр перегрузки может быть отслежен наблюдением OVFG в регистре CANGSTA (см. рисунок 19-9 на стр. 243).

- **Bit 5– TTC: Time Trigger Communication – связь по расписанию**
  – 0 – нет TTC.
  – 1 – режим TTC.
- **Bit 4 – SYNTTC: Synchronization of TTC – синхронизация связи по** расписанию

Этот бит используется только в режиме TTC.

  – 0 – таймер TTC захватывается на SOF.
  – 1 – таймер TTC захватывается на последнем бите EOF.

- **Bit 3 – LISTEN: Listening Mode – режим прослушивания**
  – 0 – нет режима прослушивания.
  – 1 – режим прослушивания.
- **Bit 2 – TEST: Test Mode – режим тестирования**
  – 0 – нет режима тестирования
  – 1 – режим тестирования: предназначен для производственного тестирования и не используется потребителями.

  Замечание: При установке этого бита CAN может потерпеть сбой.

- **Bit 1 – ENA/STB: Enable / Standby Mode – режим разрешение / ожидание**

Поскольку этот бит является командой и не дает немедленного эффекта, бит ENFG в регистре CANGSTA дает истинное состояние выбранного режима.

– 0 – режим ожидания: Текущая передача нормально заканчивается, и канал CAN замораживается (биты CONMOB каждого MOb не изменяются). Передатчик постоянно поддерживает рецессивный уровень. В этом режиме приемник не разрешен, но все регистры и почтовый ящик остаются доступными из центрального процессора.

Замечание: Режим ожидания, активированный во время приема, может повредить текущий прием или установить контроллер в неправильное состояние. Контроллер корректно перезапустится из этого состояния, если будет применен программный сброс (SWRES). Если не подразумевается никакого сброса, возможное решение состоит в том, чтобы ожидать занятости приемника (RXBSY) перед входом в режим ожидания. Лучшее решение состоит в том, чтобы сначала применить команду запроса аварийного завершения (ABRQ) а затем ожидать занятости приемника (RXBSY) перед входом в режим ожидания. В любом случае поведение режима ожидания не производит никакого эффекта на целостность шины CAN.

– 1 – режим разрешения: канал CAN входит в режим разрешения, как только считывает 11 рецессивных битов.

- **Bit 0 – SWRES: Software Reset Request – запрос программного сброса**

Этот автоматически сбрасываемый бит сбрасывает только контроллер CAN.

– 0 – нет сброса
– 1 – сброс: действует по логическому “ИЛИ” с аппаратным сбросом.

19.10.2 CANGSTA: CAN General Status Register – общий регистр состояния CAN

Bit	7	6	5	4	3	2	1	0	
	-	OVRG	-	TXBSY	RXBSY	ENFG	BOFF	ERRP	CANGSTA
Read/Write	-	R	-	R	R	R	R	R	
Initial Value	-	0	-	0	0	0	0	0	

- **Bit 7 – Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений.

- **Bit 6 – OVRG: Overload Frame Flag – флаг кадра перегрузки**

Этот флаг не генерирует прерывание.

– 0 – нет кадра перегрузки.
– 1 – кадр перегрузки: устанавливается аппаратно, пока посылается произведенный кадр перегрузки.

- **Bit 5 – Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений.

- **Bit 4 – TXBSY: Transmitter Busy – передатчик занят**

Этот флаг не генерирует прерывание.

– 0 – передатчик не занят.
– 1 – передатчик занят: устанавливается аппаратно, пока передается кадр (кадр данных, удаленный, перегрузки или ошибки) или поле ACK. Также устанавливается, когда передается межкадровый промежуток.

- **Bit 3 – RXBSY: Receiver Busy – приемник занят**

Этот флаг не генерирует прерывание.

– 0 – приемник не занят.
– 1 – приемник занят: устанавливается аппаратно, пока принимается или наблюдается кадр.

- **Bit 2 – ENFG: Enable Flag – флаг разрешения**

Этот флаг не генерирует прерывание.

  – 0 – контроллер CAN запрещен: поскольку команда разрешение / ожидание не производит немедленного эффекта, это состояние дает истинное состояние выбранного режима.
  – 1 – контроллер CAN разрешен.

- **Bit 1 – BOFF: Bus Off Mode – режим отключения от шины**

BOFF дает информацию о состоянии канала CAN. Только вход в режим отключения от шины генерирует прерывание BOFFIT.

  – 0 – нет режима отключения от шины.
  – 1 – режим отключения от шины.

- **Bit 0 – ERRP: Error Passive Mode – пассивный режим ошибки**

ERRP дает информацию о состоянии канала CAN. Этот флаг не генерирует прерывание.

  – 0 – нет пассивного режима ошибки.
  – 1 – пассивный режим ошибки.

19.10.3 CANGIT: CAN General Interrupt Register – общий регистр прерываний

Bit	7	6	5	4	3	2	1	0	
	CANIT	BOFFIT	OVRTIM	BXOK	SERG	CERG	FERG	AERG	CANGIT
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – CANIT: General Interrupt Flag – флаг общих прерываний**

Этот бит только для чтения.

  – 0 – Прерывание отсутствует.
  – 1 – Прерывание CAN: образ всех прерываний контроллера CAN за исключением прерывания OVRTIM. Этот бит может использоваться методом опроса (поллинг).

- **Bit 6 – BOFFIT: Bus Off Interrupt Flag – флаг прерывания отключения от шины**

Запись логической 1 сбрасывает этот флаг прерывания. Флаг BOFFIT устанавливается только когда CAN входит в режим отключения от шины (вызываемый режимом пассивной ошибки).

  – 0 – Прерывание отсутствует.
  – 1 – Прерывание отключения от шины, когда CAN входит в режим отключения от шины.

- **Bit 5 – OVRTIM: Overrun CAN Timer – переполнение таймера CAN**

Запись логической 1сбрасывает этот флаг прерывания. Вход в обработчик прерывания переполнения таймера CAN также сбрасывает этот флаг прерывания.

  – 0 – Прерывание отсутствует.
  – 1 – Прерывание по переполнению таймера CAN: устанавливается, когда таймер CAN переходит из 0xFFFF в 0.

- **Bit 4 – BXOK: Frame Buffer Receive Interrupt – прерывание приема в буфер кадра**

Запись логической 1сбрасывает этот флаг прерывания. Флаг BXOK может быть очищен, только если все поля CONMOB MOb буфера были ранее перезаписаны.

  – 0 – Прерывание отсутствует.

– 1 – Прерывание получения пакета: устанавливается, когда завершается прием в буфер кадра.

- **Bit 3 – SERG: Stuff Error General – общая ошибка заполнения**

Запись логической 1сбрасывает этот флаг прерывания.

– 0 – Прерывание отсутствует.
– 1 – Прерывание по ошибке заполнения: обнаружено более 5 последовательных бит одной полярности.

- **Bit 2 – CERG: CRC Error General – общая ошибка CRC**

Запись логической 1сбрасывает этот флаг прерывания.

– 0 – Прерывание отсутствует.
– 1 – Прерывание по ошибке CRC: проверенная CRC незаполненного сообщения не соответствует полю CRC.

- **Bit 1 – FERG: Form Error General – общая ошибка формы**

Запись логической 1сбрасывает этот флаг прерывания.

– 0 – Прерывание отсутствует.
– 1 – Прерывание по ошибке формы: одно или более нарушений фиксированной формы в разделителе CRC, разделителе подтверждения или в EOF.

- **Bit 0 – AERG: Acknowledgment Error General – общая ошибка подтверждения**

Запись логической 1сбрасывает этот флаг прерывания.

– 0 – Прерывание отсутствует.
– 1 – Прерывание по ошибке подтверждения: не обнаружен доминантный бит в промежутке подтверждения.

19.10.4 CANGIE: CAN General Interrupt Enable Register – регистр разрешения общих прерываний CAN

Bit	7	6	5	4	3	2	1	0	
	ENIT	ENBOFF	ENRX	ENTX	ENERR	ENBX	ENERG	ENOVRT	CANGIE
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ENIT: Enable all Interrupts – разрешение всех прерываний за исключением прерывания по переполнению таймера CAN**
  – 0 – Прерывание запрещено.
  – 1 – Прерывания CANIT разрешены.
- **Bit 6 – ENBOFF: Enable Bus Off Interrupt – разрешение прерывания по отключению от шины**
  – 0 – Прерывание запрещено.
  – 1 – Прерывание по отключению от шины разрешено.
- **Bit 5 – ENRX: Enable Receive Interrupt – разрешение прерывания по приему**
  – 0 – Прерывание запрещено.
  – 1 – Прерывание по приему разрешено.
- **Bit 4 – ENTX: Enable Transmit Interrupt – разрешение прерывания по передаче**
  – 0 – Прерывание запрещено.
  – 1 – Прерывание по передаче разрешено.
- **Bit 3 – ENERR: Enable MOb Errors Interrupt – разрешение прерывания по ошибкам MOb**
  – 0 – Прерывание запрещено.

– 1 – Прерывание по ошибкам MOb разрешено.

- **Bit 2 – ENBX: Enable Frame Buffer Interrupt – разрешение прерывания по буферу кадра**
  – 0 – Прерывание запрещено.
  – 1 – Прерывание по буферу кадра разрешено.
- **Bit 1 – ENERG: Enable General Errors Interrupt – разрешение прерывания по общим ошибкам**
  – 0 – Прерывание запрещено.
  – 1 – Прерывание по общим ошибкам разрешено.
- **Bit 0 – ENOVRT: Enable CAN Timer Overrun Interrupt – разрешение прерывания по переполнению таймера CAN**
  – 0 – Прерывание запрещено.
  – 1 – Прерывание по переполнению таймера CAN разрешено.

19.10.5 CANEN2, CANEN1: CAN Enable MOb Registers – регистры разрешения MOb CAN

Bit	7	6	5	4	3	2	1	0	
	ENMOB7	ENMOB6	ENMOB5	ENMOB4	ENMOB3	ENMOB2	ENMOB1	ENMOB0	CANEN2
	-	ENMOB14	ENMOB13	ENMOB12	ENMOB11	ENMOB10	ENMOB9	ENMOB8	CANEN1
Bit	15	14	13	12	11	10	9	8	
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
Read/Write	-	R	R	R	R	R	R	R	
Initial Value	-	0	0	0	0	0	0	0	

- **Bits 14:0 - ENMOB14:0: Enable Mob – разрешение MOb**

Эти биты отражают доступность MOb.

Они установлены в 1, если MOb разрешены (т.е. CONMOB1:0 регистра CANCDMOB).

Когда TXOK или RXOK устанавливается в 1 (TXOK для автоматического ответа), соответствующий ENMOB сбрасывается. ENMOB также устанавливается в нуль настраивая MOb в запрещенный режим, выполняя аварийное завершение или режим ожидания.

– 0 – Объект сообщения запрещен: MOb доступен для новой передачи или приема.
– 1 - Объект сообщения разрешен: MOb используется.

- **Bit 15 – Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений.

19.10.6 CANIE2, CANIE1: CAN Enable Interrupt MOb Registers – регистры разрешения прерывания MOb CAN

Bit	7	6	5	4	3	2	1	0	
	IEMOB7	IEMOB6	IEMOB5	IEMOB4	IEMOB3	IEMOB2	IEMOB1	IEMOB0	CANIE2
	-	IEMOB14	IEMOB13	IEMOB12	IEMOB11	IEMOB10	IEMOB9	IEMOB8	CANIE1
Bit	15	14	13	12	11	10	9	8	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	0	0	0	0	0	0	0	

- **Bits 14:0 – IEMOB14:0: Interrupt Enable by MOb**
  – 0 – Прерывание запрещено.
  – 1 – Прерывание MOb разрешено.

Пример: CANIE2 = 0000 1100_b – разрешение прерываний MOb 2 и 3.

- **Bit 15 – Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений. Для совместимости с будущими устройствами должен устанавливаться в 0 при записи CANIE1.

19.10.7 CANSIT2, CANSIT1: CAN Status Interrupt MOb Registers – регистры состояния прерываний MOb CAN

Bit	7	6	5	4	3	2	1	0	
	SIT7	SIT6	SIT5	SIT4	SIT3	SIT2	SIT1	SIT0	CANSIT2
	-	SIT14	SIT13	SIT12	SIT11	SIT10	SIT9	SIT8	CANSIT1
Bit	15	14	13	12	11	10	9	8	
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
Read/Write	-	R	R	R	R	R	R	R	
Initial Value	-	0	0	0	0	0	0	0	

- **Bits 14:0 - SIT14:0: Status of Interrupt by MOb – состояние прерываний MOb**
  - 0 – Прерывание отсутствует.
  - 1 – Прерывание MOb.

Пример: CANSIT2 = 0010 0001_b : прерывания MOb 0 и 5.

- **Bit 15 – Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений.

19.10.8 CANBT1: CAN Bit Timing Register 1 – регистр 1 битовой синхронизации CAN

Bit	7	6	5	4	3	2	1	0	
	-	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	-	CANBT1
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	-	
Initial Value	-	0	0	0	0	0	0	-	

- **Bit 7– Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений. Для совместимости с будущими устройствами должен устанавливаться в 0 при записи CANBT1.

- **Bit 6:1 – BRP5:0: Baud Rate Prescaler – предделитель скорости в бодах**

Период системы синхронизации контроллера CAN Tscl программируется и определяет индивидуальную битовую синхронизацию.

$$Tscl = \frac{BRP[5:0]+1}{clk_{10}\,frequency}$$

Если BRP [5.. 0]=0, см. раздел 19.4.3" Скорость в бодах” на стр. 242 и раздел ”Бит 0 - SMP: Точки выборки” на стр. 259.

- **Bit 0 – Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений. Для совместимости с будущими устройствами должен устанавливаться в 0 при записи.

19.10.9 CANBT2: CAN Bit Timing Register 2 – регистр 2 битовой синхронизации CAN

Bit	7	6	5	4	3	2	1	0	
	-	SJW1	SJW0	-	PRS2	PRS1	PRS0	-	CANBT2
Read/Write	-	R/W	R/W	-	R/W	R/W	R/W	-	
Initial Value	-	0	0	-	0	0	0	-	

- **Bit 7– Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений. Для совместимости с будущими устройствами должен устанавливаться в 0 при записи CANBT2 is written.

- **Bit 6:5 – SJW1:0: Re-Synchronization Jump Width – ширина перехода ресинхронизации**

Для компенсации сдвига фазы между генераторами синхросигналов различных контроллеров шины, контроллер должен ресинхронизироваться по каждому соответствующему фронту сигнала текущей передачи.

Ширина перехода синхронизации определяет максимальное число тактовых циклов. Период бита может быть сокращен или удлинен ресинхронизацией.

$$Tsjw = Tscl \times (SJW[1:0] + 1)$$

- **Bit 4 – Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений. Для совместимости с будущими устройствами должен устанавливаться в 0 при записи.

- **Bit 3:1 – PRS2:0: Propagation Time Segment – сегмент времени распространения**

Эта часть периода бита используется для компенсации времени физической задержки в сети. Представляет двойную сумму времени распространения сигнала по линиям шины, задержку входного компаратора и задержку драйвера выхода.

$$Tprs = Tscl \times (PRS[2:0] + 1)$$

- **Bit 0 – Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений. Для совместимости с будущими устройствами должен устанавливаться в 0 при записи CANBT2 is written.

19.10.10 CANBT3: CAN Bit Timing Register 3 – регистр 3 битовой синхронизации CAN

Bit	7	6	5	4	3	2	1	0	
	-	PHS22	PHS21	PHS20	PHS12	PHS11	PHS10	SMP	CANBT3
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	0	0	0	0	0	0	0	

- **Bit 7– Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений. Для совместимости с будущими устройствами должен устанавливаться в 0 при записи CANBT3 is written.

- **Bit 6:4 – PHS22:0: Phase Segment 2 – сегмент фазы 2**

Эта фаза используется для компенсации ошибки фазы фронта. Этот сегмент может быть сокращен шириной перехода ресинхронизации. PHS2 [2.. 0] будет ≥ 1 и $\leq$ PHS1 [2.. 0] (см. раздел 19.2.3"Битовая синхронизация CAN" на стр. 236 и раздел 19.4.3 "Скорость в бодах" на стр. 242).

$$Tphs2 = Tscl \times (PHS2[2:0] + 1)$$

- **Bit 3:1 – PHS12:0: Phase Segment 1 – сегмент фазы 1**

Эта фаза используется для компенсации ошибки фазы фронта. Этот сегмент может быть удлинен шириной перехода ресинхронизации.

$$Tphs1 = Tscl \times (PHS1[2:0] + 1)$$

- **Bit 0 – SMP: Sample Point(s) – точки выборки**

Эта опция позволяет фильтровать шумы на входном выводе TxCAN.

– 0 – Выборка производится однократно в конфигурируемой пользователем точке выборки **SP**.
– 1 – Конфигурация с выборкой в трех точках. Первая выборка производится за два такта $Tclk_{IO}$ до конфигурируемой пользователем точки выборки **SP**, затем за один такт $Tclk_{IO}$ до **SP** и затем в точке **SP**. Значение бита будет определено мажоритацией из трех выборок.

Конфигурация ‘SMP=1’ несовместима с ‘BRP[5:0]=0’ поскольку TQ = $Tclk_{IO}$.

Если BRP = 0, SMP должно быть очищено.

19.10.11 CANTCON: CAN Timer Control Register – регистр управления таймера CAN

Bit	7	6	5	4	3	2	1	0	
	TPRSC7	TPRSC6	TPRSC5	TPRSC4	TPRSC3	TPRSC2	TRPSC1	TPRSC0	CANTCON
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:0 – TPRSC7:0: CAN Timer Prescaler – предделитель таймера CAN**

Верхний диапазон счетчика предделителя таймера CAN от 0 до 255. Он обеспечивает тактовый сигнал для таймера CAN, если контроллер CAN разрешен.

$$Tclk_{CANTIM} = Tclk_{IO} \times 8 \times (CANTCON\ [7:0] + 1)$$

19.10.12 CANTIML, CANTIMH: CAN Timer Registers – регистры таймера CAN

Bit	7	6	5	4	3	2	1	0	
	CANTIM7	CANTIM6	CANTIM5	CANTIM4	CANTIM3	CANTIM2	CANTIM1	CANTIM0	CANTIML
	CANTIM15	CANTIM14	CANTIM13	CANTIM12	CANTIM11	CANTIM10	CANTIM9	CANTIM8	CANTIMH
Bit	15	14	13	12	11	10	9	8	
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 15:0 - CANTIM15:0: CAN Timer Count – отсчет таймера CAN**

Таймер CAN считает от 0 до 65535.

19.10.13 CANTTCL, CANTTCH: CAN TTC Timer Registers – регистры таймера TTC CAN

Bit	7	6	5	4	3	2	1	0	
	TIMTTC7	TIMTTC6	TIMTTC5	TIMTTC4	TIMTTC3	TIMTTC2	TIMTTC1	TIMTTC0	CANTTCL
	TIMTTC15	TIMTTC14	TIMTTC13	TIMTTC12	TIMTTC11	TIMTTC10	TIMTTC9	TIMTTC8	CANTTCH
Bit	15	14	13	12	11	10	9	8	
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 15:0 - TIMTTC15:0: TTC Timer Count– отсчет таймера TTC**

Диапазон таймера-счетчика CAN TTC от 0 до 65535.

19.10.14 CANTEC: CAN Transmit Error Counter Register – регистр счетчика ошибок передачи CAN

Bit	7	6	5	4	3	2	1	0	
	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	CANTEC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:0 – TEC7:0: Transmit Error Count – число ошибок передачи**

Диапазон счетчика ошибок передачи CAN от 0 до 255.

19.10.15 CANREC: CAN Receive Error Counter Register – регистр счетчика ошибок приема CAN

Bit	7	6	5	4	3	2	1	0	
	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	CANREC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:0 – REC7:0: Receive Error Count – число ошибок приема**

Диапазон счетчика ошибок приема CAN от 0 до 255.

19.10.16 CANHPMOB: CAN Highest Priority MOb Register – регистр MOb высшего приоритета CAN

Bit	7	6	5	4	3	2	1	0	
	HPMOB3	HPMOB2	HPMOB1	HPMOB0	CGP3	CGP2	CGP1	CGP0	CANHPMOB
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	0	0	0	0	

- **Bit 7:4 – HPMOB3:0: Highest Priority MOb Number – номер MOb с высшим приоритетом**

MOb, имеющий самый высокий приоритет в регистрах CANSIT.

Если CANSIT = 0 (нет MOb), возвращаемое значение равно 0xF.

Замечание: Не смешивать “приоритет MOb” и “приоритет ID сообщения”. См. “Объекты сообщений ” на стр. 243.

- **Bit 3:0 – CGP3:0: CAN General Purpose Bits – биты общего назначения CAN**

Эти биты могут быть предварительно запрограммированы для соответствия требуемой конфигурации регистра CANPAGE (то есть, установки AINC и INDX2:0).

19.10.17 CANPAGE: CAN Page MOb Register – регистр страницы MOb CAN

Bit	7	6	5	4	3	2	1	0	
	MOBNB3	MOBNB2	MOBNB1	MOBNB0	$\overline{\text{AINC}}$	INDX2	INDX1	INDX0	CANPAGE
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:4 – MOBNB3:0: MOb Number – номер MOb**

Выбор номера MOb. Допустимые номера от 0 до 14.

- **Bit 3 – AINC: Auto Increment of the FIFO CAN Data Buffer Index (Active Low) – автоинкремент индекса FIFO-буфера данных CAN (активный низкий)**
  - 0 – Автоинкремент индекса (по умолчанию).
  - 1 – Нет автоинкремента индекса.
- **Bit 2:0 – INDX2:0: FIFO CAN Data Buffer Index**

Местоположение байта данных CAN в FIFO для определенного MOb.

19.11 Регистры объектов сообщений

Регистры MOb **НЕ** имеют инициализирующих значений по умолчанию после сброса RESET.

19.11.1 CANSTMOB: CAN MOb Status Register – регистр состояния объектов сообщений

Bit	7	6	5	4	3	2	1	0	
	DLCW	TXOK	RXOK	BERR	SERR	CERR	FERR	AERR	CANSTMOB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	-	-	-	-	-	-	-	

- **Bit 7 – DLCW: Data Length Code Warning – предупреждение о коде длины данных**

Входящее сообщение не имеет ожидаемый DLC. Независимо от типа кадра поле DLC регистра CANCDMOB обновляется полученным DLC.

- **Bit 6 – TXOK: Transmit OK – удачное завершение передачи**

Этот флаг может генерировать прерывание. Он должен быть очищен процедурой типа "чтение-модификация-запись" всего регистра CANSTMOB.

Связь разрешается завершением передачи.

TxOK устанавливается в конце поля EOF и затем MOb запрещается (соответствующий бит ENMOB регистров CANEN очищается). Когда контроллер готов послать кадр, и при этом два или более объектов сообщения разрешены как производители, первым обеспечивается более низкий индекс MOb.

- **Bit 5 – RXOK: Receive OK – удачное завершение передачи**

Этот флаг может генерировать прерывание. Он должен быть очищен процедурой типа "чтение-модификация-запись" всего регистра CANSTMOB.

Связь разрешается завершением приема.

RxOK устанавливается в конце 6-го бита поля EOF и затем MOb запрещается (соответствующий бит ENMOB регистров CANEN очищается). В случае удачного приема двух или более объектов сообщений первым обновляется более низкий индекс MOb.

- **Bit 4 – BERR: Bit Error (Only in Transmission) – битовая ошибка (только при передаче)**

Этот флаг может генерировать прерывание. Он должен быть очищен процедурой типа "чтение-модификация-запись" всего регистра CANSTMOB.

Значение наблюдаемого бита отличается от переданного значения.

Исключения: наблюдаемый рецессивный бит послан как доминантный бит в период поля арбитража, и в промежутке подтверждения обнаружен доминантный бит во время посылки кадра ошибки.

Установка этого флага не запрещает MOb (соответствующий бит ENMOB регистров CANEN не очищается). Следующий правильный кадр обновит флаг BERR.

- **Bit 3 – SERR: Stuff Error – ошибка наполнения**

Этот флаг может генерировать прерывание. Он должен быть очищен процедурой типа "чтение-модификация-запись" всего регистра CANSTMOB.

Обнаружение более чем пяти последовательных бит одной полярности. Этот флаг может генерировать прерывание.

Установка этого флага не запрещает MOb (соответствующий бит ENMOB регистров CANEN не очищается). Следующий правильный кадр обновит флаг SERR.

- **Bit 2 – CERR: CRC Error – ошибка CRC**

Этот флаг может генерировать прерывание. Он должен быть очищен процедурой типа "чтение-модификация-запись" всего регистра CANSTMOB.

Приемник проверяет CRC каждого незаполненного полученного сообщения от начала кадра до поля данных. Если проверка не дает соответствия с незаполненным полем CRC, устанавливается ошибка CRC.

Установка этого флага не запрещает MOb (соответствующий бит ENMOB регистров CANEN не очищается). Следующий правильный кадр обновит флаг CERR.

- **Bit 1 – FERR: Form Error – ошибка формы**

Этот флаг может генерировать прерывание Он должен быть очищен процедурой типа "чтение-модификация-запись" всего регистра CANSTMOB.

Ошибка формы приводит к одному или более нарушению фиксированной формы в следующих битовых полях:

- Разделитель CRC.
- Разделитель подтверждения.
- EOF.

Установка этого флага не запрещает MOb (соответствующий бит ENMOB регистров CANEN не очищается). Следующий правильный кадр обновит флаг FERR.

- **Bit 0 – AERR: Acknowledgment Error – ошибка подтверждения**

Этот флаг может генерировать прерывание. Он должен быть очищен процедурой типа "чтение-модификация-запись" всего регистра CANSTMOB.

Не обнаружен доминантный бит в промежутке подтверждения.

Установка этого флага не запрещает MOb (соответствующий бит ENMOB регистров CANEN не очищается). Следующий правильный кадр обновит флаг CERR.

19.11.2 CANCDMOB: CAN MOb Control and DLC Register – регистр управления и DLC

Bit	7	6	5	4	3	2	1	0	
	CONMOB1	CONMOB0	RPLV	IDE	DLC3	DLC2	DLC1	DLC0	CANCDMOB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	-	-	-	-	-	-	-	

- **Bit 7:6 – CONMOB1:0: Configuration of Message Object – конфигурация объекта сообщения**

Эти биты заставляют производить коммуникации (**не** существует никакого начального значения после сброса RESET).

– 00 – запрет.
– 01 – разрешение передачи.
– 10 – разрешение приема.
– 11 – разрешение приема буфера кадра.

Эти биты **не** очищаются при завершении связи. Пользователь должен перезаписать конфигурацию для разрешения новой связи.

– Эта операция необходима для возможности сброса флага BXOK.
– Эта операция также устанавливает соответствующий бит в регистрах CANEN.

- **Bit 5 – RPLV: Reply Valid – ответ допустим**

Используется в режиме автоматического ответа после получения удаленного кадра.

– 0 – ответ не готов.
– 1 – ответ готов и допустим.

- **Bit 4 – IDE: Identifier Extension – расширение идентификатора**

Бит IDE удаленного кадра или кадра данных для передачи.

Этот бит обновляется соответствующим значением полученного удаленного кадра или кадра данных.

– 0 – Стандарт CAN ред. 2.0 A (длина идентификаторов = 11 бит).
– 1 – Стандарт CAN ред. 2.0 B (длина идентификаторов = 29 бит).

- **Bit 3:0 – DLC3:0: Data Length Code – код длины данных**

Число байт в поле данных сообщения.

Поле DLC удаленного кадра или кадра данных для передачи. Диапазон DLC от 0 до 8. Если поле DLC> 8 тогда принимается DLC=8.

Это поле обновляется соответствующим значением полученного удаленного кадра или кадра данных. Если ожидаемый DLC отличается от входящего DLC, в регистре CANSTMOB появляется предупреждение DLC.

19.11.3 CANIDT1, CANIDT2, CANIDT3, CANIDT4: CAN Identifier Tag Registers – регистры тэга идентификатора CAN

Ред. 2.0 часть A

Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
	-	-	-	-	-	RTRTAG	-	RB0TAG	CANIDT4
	-	-	-	-	-	-	-	-	CANIDT3
	IDT2	IDT1	IDT0	-	-	-	-	-	CANIDT2
	IDT10	IDT9	IDT8	IDT7	IDT6	IDT5	IDT4	IDT3	CANIDT1
Bit	31/23	30/22	29/21	28/20	27/19	26/18	25/17	24/16	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	-	-	-	-	-	-	-	

Ред. 2.0 часть B

Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
	IDT4	IDT3	IDT2	IDT1	IDT0	RTRTAG	RB1TAG	RB0TAG	CANIDT4
	IDT12	IDT11	IDT10	IDT9	IDT8	IDT7	IDT6	IDT5	CANIDT3
	IDT20	IDT19	IDT18	IDT17	IDT16	IDT15	IDT14	IDT13	CANIDT2
	IDT28	IDT27	IDT26	IDT25	IDT24	IDT23	IDT22	IDT21	CANIDT1
Bit	31/23	30/22	29/21	28/20	27/19	26/18	25/17	24/16	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	-	-	-	-	-	-	-	

Ред. 2.0 часть A

- **Bit 31:21 – IDT10:0: Identifier Tag – тэг идентификатора**

Поле идентификатора удаленного кадра или кадра данных для передачи.

Это поле обновляется соответствующим значением полученного удаленного кадра или кадра данных.

- **Bit 20:3 – Reserved Bits – зарезервированные биты**

Эти биты зарезервированы для будущего использования. Для совместимости с будущими устройствами при записи CANIDTn они должны иметь значение 0.

При получении удаленного кадра или кадра данных эти биты не участвуют в сравнении полей, но обновляются случайными значениями.

- **Bit 2 – RTRTAG: Remote Transmission Request Tag – тэг запроса удаленной передачи**

Бит RTR удаленного кадра или кадра данных для передачи.

Этот тэг обновляется соответствующим значением удаленного полученного кадра или кадра данных. В режиме автоматического ответа этот бит автоматически сбрасывается перед посылкой ответа.

- **Bit 1 – Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений. Для совместимости с будущими устройствами при записи CANIDTn они должны иметь значение 0.

При получении удаленного кадра или кадра данных эти биты не участвуют в сравнении полей, но обновляются случайными значениями.

- **Bit 0 – RB0TAG: Reserved Bit 0 Tag – тэг зарезервированного бита 0**

Бит RB0 удаленного кадра или кадра данных для передачи.

Этот тэг обновляется соответствующим значением удаленного полученного кадра или кадра данных.

Ред. 2.0 часть B

- **Bit 31:3 – IDT28:0: Identifier Tag – тэг идентификатора**

Поле идентификатора удаленного кадра или кадра данных для передачи.

Это поле обновляется соответствующим значением полученного удаленного кадра или кадра данных.

- **Bit 2 – RTRTAG: Remote Transmission Request Tag – тэг запроса удаленной передачи**

Бит RTR удаленного кадра или кадра данных для передачи.

Этот тэг обновляется соответствующим значением удаленного полученного кадра или кадра данных. В режиме автоматического ответа этот бит автоматически сбрасывается перед посылкой ответа.

- **Bit 1 – RB1TAG: Reserved Bit 1 Tag – тэг зарезервированного бита 1**

Бит RB1 удаленного кадра или кадра данных для передачи.

Этот тэг обновляется соответствующим значением удаленного полученного кадра или кадра данных.

- **Bit 0 – RB0TAG: Reserved Bit 0 Tag – тэг зарезервированного бита 0**

Бит RB0 удаленного кадра или кадра данных для передачи.

Этот тэг обновляется соответствующим значением удаленного полученного кадра или кадра данных.

19.11.4 CANIDM1, CANIDM2, CANIDM3, CANIDM4: CAN Identifier Mask Registers – регистры маски идентификатора CAN

Ред. 2.0 часть A

Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
	-	-	-	-	-	RTRMSK	-	IDEMSK	CANIDM4
	-	-	-	-	-	-	-	-	CANIDM3
	IDMSK2	IDMSK1	IDMSK0	-	-	-	-	-	CANIDM2
	IDMSK10	IDMSK9	IDMSK8	IDMSK7	IDMSK6	IDMSK5	IDMSK4	IDMSK3	CANIDM1
Bit	31/23	30/22	29/21	28/20	27/19	26/18	25/17	24/16	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	-	-	-	-	-	-	-	

Ред. 2.0 часть B

Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
	IDMSK4	IDMSK3	IDMSK2	IDMSK1	IDMSK0	RTRMSK	-	IDEMSK	CANIDM4
	IDMSK12	IDMSK11	IDMSK10	IDMSK9	IDMSK8	IDMSK7	IDMSK6	IDMSK5	CANIDM3
	IDMSK20	IDMSK19	IDMSK18	IDMSK17	IDMSK16	IDMSK15	IDMSK14	IDMSK13	CANIDM2
	IDMSK28	IDMSK27	IDMSK26	IDMSK25	IDMSK24	IDMSK23	IDMSK22	IDMSK21	CANIDM1
Bit	31/23	30/22	29/21	28/20	27/19	26/18	25/17	24/16	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	-	-	-	-	-	-	-	

Ред. 2.0 часть A

- **Bit 31:21 – IDMSK10:0: Identifier Mask – маска идентификатора**
  - 0 – принудительный положительный результат сравнения – см. “Фильтр приема” на стр. 246.
  - 1 – сравнение битов разрешено – см.“Фильтр приема” на стр. 246.
- **Bit 20:3 – Reserved Bits – зарезервированные биты**

Эти биты зарезервированы для будущего использования. Для совместимости с будущими устройствами при записи CANIDTn они должны иметь значение 0.

- **Bit 2 – RTRMSK: Remote Transmission Request Mask – маска запроса удаленной передачи**
  - 0 – принудительный положительный результат сравнения.
  - 1 – сравнение битов разрешено.
- **Bit 1 – Reserved Bit – зарезервированный бит**

Этот бит зарезервирован для будущих применений. Для совместимости с будущими устройствами при записи CANIDTn они должны иметь значение 0.

- **Bit 0 – IDEMSK: Identifier Extension Mask – маска расширения идентификатора**
  - 0 – принудительный положительный результат сравнения.
  - 1 – сравнение битов разрешено.

Ред. 2.0 часть B

- **Bit 31:3 – IDMSK28:0: Identifier Mask – маска идентификатора**
  - 0 – принудительный положительный результат сравнения – см. “Фильтр приема” на стр. 246.
  - 1 – сравнение битов разрешено – см.“Фильтр приема” на стр. 246.
- **Bit 2 – RTRMSK: Remote Transmission Request Mask – маска запроса удаленной передачи**

– 0 – принудительный положительный результат сравнения.
– 1 – сравнение битов разрешено.

- **Bit 1 – Reserved Bit – зарезервированный бит**

Рекомендуется записывать в этот бит 0.

- **Bit 0 – IDEMSK: Identifier Extension Mask – маска расширения идентификатора**
  – 0 – принудительный положительный результат сравнения.
  – 1 – сравнение битов разрешено.

19.11.5 CANSTML, CANSTMH: CAN Time Stamp Registers – регистры временной отметки CAN

Bit	7	6	5	4	3	2	1	0	
	TIMSTM7	TIMSTM6	TIMSTM5	TIMSTM4	TIMSTM3	TIMSTM2	TIMSTM1	TIMSTM0	CANSTML
	TIMSTM15	TIMSTM14	TIMSTM13	TIMSTM12	TIMSTM11	TIMSTM10	TIMSTM9	TIMSTM8	CANSTMH
Bit	15	14	13	12	11	10	9	8	
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	-	-	-	-	-	-	-	-	

- **Bits 15:0 - TIMSTM15:0: Time Stamp Count – счетчик временной отметки**

Диапазон счетчика временной отметки CAN от 0 до 65535.

19.11.6 CANMSG: CAN Data Message Register – регистр данных сообщения CAN

Bit	7	6	5	4	3	2	1	0	
	MSG 7	MSG 6	MSG 5	MSG 4	MSG 3	MSG 2	MSG 1	MSG 0	CANMSG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	-	-	-	-	-	-	-	

- **Bit 7:0 – MSG7:0: Message Data – данные сообщения**

Этот регистр содержит байт данных CAN, адресуемый регистром страницы MOb.

После записи в регистр страницы MOb этот байт равен значению в сообщении, адресуемому предопределенным идентификатором + индекс. Если используется автоматическое приращение, после записи регистра данных или цикла чтения индекс автоматически инкрементируется.

Модуль счета составляет 8 в бесконечном цикле (0, 1,..., 7, 0,...).

19.12 Примеры установок скорости CAN

Шина CAN требует очень точной синхронизации, особенно на высоких скоростях обмена. Для работы с CAN рекомендуется использовать только внешний кварцевый резонатор.

(См. “Битовая синхронизация” на стр. 241, “Скорость в бодах” на стр. 242 и “Регистры синхронизации битов CAN” на стр. 257...258).

Таблица 19-2. Примеры установок скорости CAN для часто применяемых частот

$Fclk_{io}$ (MHz)	Скорость CAN (Kbps)	Описание			Сегменты				Регистры		
		Точка выборки	TQ (µs)	Tbit (TQ)	Tprs (TQ)	Tph1 (TQ)	Tph2 (TQ)	Tsjw (TQ)	CANBT1	CANBT2	CANBT3
16.000	1000	69.5%(1)	0.0625	16	7	4	4	1	0x00	0x0C	0x36(2)
		75%	0.125	8	3	2	2	1	0x02	0x04	0x13
	500	75%	0.125	16	7	4	4	1	0x02	0x0C	0x37
			0.250	8	3	2	2	1	0x06	0x04	0x13
	250	75%	0.250	16	7	4	4	1	0x06	0x0C	0x37
			0.500	8	3	2	2	1	0x0E	0x04	0x13
	200	75%	0.3125	16	7	4	4	1	0x08	0x0C	0x37
			0.625	8	3	2	2	1	0x12	0x04	0x13
	125	75%	0.500	16	7	4	4	1	0x0E	0x0C	0x37
			1.000	8	3	2	2	1	0x1E	0x04	0x13
	100	75%	0.625	16	7	4	4	1	0x12	0x0C	0x37
			1.250	8	3	2	2	1	0x26	0x04	0x13
$Fclk_{io}$ (MHz)	**Скорость CAN (Kbps)**	**Описание**			**Сегменты**				**Регистры**		
		Точка выборки	**TQ (µs)**	**Tbit (TQ)**	**Tprs (TQ)**	**Tph1 (TQ)**	**Tph2 (TQ)**	**Tsjw (TQ)**	**CANBT1**	**CANBT2**	**CANBT3**
12.000	1000	67%(1)	0.083333	12	5	3	3	1	0x00	0x08	0x24(2)
				x	- - - нет данных - - -						
	500	75%	0.166666	12	5	3	3	1	0x02	0x08	0x25
			0.250	8	3	2	2	1	0x04	0x04	0x13
	250	75%	0.250	16	7	4	4	1	0x04	0x0C	0x37
			0.500	8	3	2	2	1	0x0A	0x04	0x13
	200	75%	0.250	20	8	6	5	1	0x04	0x0E	0x4B
			0.416666	12	5	3	3	1	0x08	0x08	0x25
	125	75%	0.500	16	7	4	4	1	0x0A	0x0C	0x37
			1.000	8	3	2	2	1	0x16	0x04	0x13
	100	75%	0.500	20	8	6	5	1	0x0A	0x0E	0x4B
			0.833333	12	5	3	3	1	0x12	0x08	0x25
$Fclk_{io}$ (MHz)	**Скорость CAN (Kbps)**	**Описание**			**Сегменты**				**Регистры**		
		Точка выборки	**TQ (µs)**	**Tbit (TQ)**	**Tprs (TQ)**	**Tph1 (TQ)**	**Tph2 (TQ)**	**Tsjw (TQ)**	**CANBT1**	**CANBT2**	**CANBT3**
8.000	1000	63%(1)		x	- - - нет данных - - -						
			0.125	8	3	2	2	1	0x00	0x04	0x12(2)
	500	69%(1)	0.125	16	7	4	4	1	0x00	0x0C	0x36(2)
		75%	0.250	8	3	2	2	1	0x02	0x04	0x13
	250	75%	0.250	16	7	4	4	1	0x02	0x0C	0x37
			0.500	8	3	2	2	1	0x06	0x04	0x13
	200	75%	0.250	20	8	6	5	1	0x02	0x0E	0x4B
			0.625	8	3	2	2	1	0x08	0x04	0x13
	125	75%	0.500	16	7	4	4	1	0x06	0x0C	0x37
			1.000	8	3	2	2	1	0x0E	0x04	0x13
	100	75%	0.625	16	7	4	4	1	0x08	0x0C	0x37
			1.250	8	3	2	2	1	0x12	0x04	0x13

Таблица 19-2. Примеры установок скорости CAN для часто применяемых частот (продолжение)

$Fclk_{io}$ (MHz)	Скорость CAN (Kbps)	Описание			Сегменты				Регистры		
		Точка выборки	TQ (µs)	Tbit (TQ)	Tprs (TQ)	Tph1 (TQ)	Tph2 (TQ)	Tsjw (TQ)	CANBT1	CANBT2	CANBT3
6.000	1000	- - - не применимо - - -									
	500	67%(1)	0.166666	12	5	3	3	1	0x00	0x08	0x24(2)
			- - - нет данных - - -								
	250	75%	0.333333	12	5	3	3	1	0x02	0x08	0x25
			0.500	8	3	2	2	1	0x04	0x04	0x13
	200	80%	0.333333	15	7	4	3	1	0x02	0x0C	0x35
			0.500	10	4	3	2	1	0x04	0x06	0x23
	125	75%	0.500	16	7	4	4	1	0x04	0x0C	0x37
			1.000	8	3	2	2	1	0x0A	0x04	0x13
	100	75%	0.500	20	8	6	5	1	0x04	0x0E	0x4B
			0.833333	12	5	3	3	1	0x08	0x08	0x25
$Fclk_{io}$ (MHz)	Скорость CAN (Kbps)	Описание			Сегменты				Регистры		
		Точка выборки	TQ (µs)	Tbit (TQ)	Tprs (TQ)	Tph1 (TQ)	Tph2 (TQ)	Tsjw (TQ)	CANBT1	CANBT2	CANBT3
4.000	1000	- - - не применимо - - -									
	500	63%(1)		x	- - - нет данных - - -						
			0.250	8	3	2	2	1	0x00	0x04	0x12(2)
	250	69%(1)	0.250	16	7	4	4	1	0x00	0x0C	0x36(2)
		75%	0.500	8	3	2	2	1	0x02	0x04	0x13
	200	70%(1)	0.250	20	8	6	5	1	0x00	0x0E	0x4A(2)
				x	- - - нет данных - - -						
	125	75%	0.500	16	7	4	4	1	0x02	0x0C	0x37
			1.000	8	3	2	2	1	0x06	0x04	0x13
	100	75%	0.500	20	8	6	5	1	0x02	0x0E	0x4B
			1.250	8	3	2	2	1	0x08	0x04	0x13

Замечание: 1. См. раздел 19.4.3 ”Скорость в бодах ” на стр. 242.
2. См. раздел ”Бит 0 – SMP: Точки выборки” на стр. 259

Перевод: Андрей Шлеенков, http://andromega.narod.ru, mailto:andromega@narod.ru 7679G–CAN–03/08