**23.1 Введение**

**23.1.1 Особенности DCAN**

Общие характеристики контроллера DCAN следующие:

- Поддержка протокола CAN версии 2.0 часть A, B (ISO 11898-1)

- Скорость передачи данных до 1 МБит/с

- Двойной источник синхронизации

- 16, 32, 64 или 128 объектов сообщений (в данном устройстве инстанцируется как 64)

- Индивидуальная маска идентификатора для каждого объекта сообщения

- Программируемый режим FIFO для объектов сообщений

- Программируемые режимы обратного цикла для самотестирования

- Режим приостановки для поддержки отладки

- Программный сброс модуля

- Автоматическое включение шины после состояния Bus-Off по программируемому 32-битному таймеру

- Механизм проверки четности оперативной памяти сообщений

- Прямой доступ к ОЗУ сообщений в тестовом режиме

- Выводы CAN Rx / Tx конфигурируются как выводы ввода-вывода общего назначения

- Две линии прерываний (плюс дополнительная линия прерывания по ошибке четности)

- Инициализация оперативной памяти

- Поддержка DMA

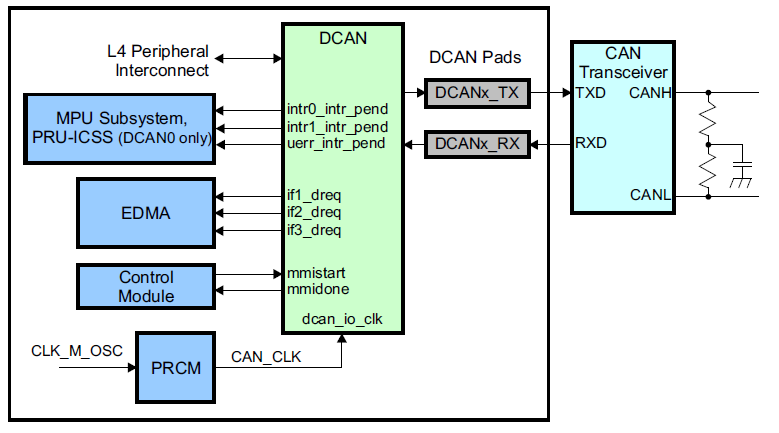
**23.1.2 Неподдерживаемые функции DCAN**

Модуль DCAN в этом устройстве не поддерживает режим работы с выводами GPIO. Вся функциональность GPIO отображается через GPIO-модули и муссируются на пины. Управляющие сигналы пинов GPIO от модулей DCAN не подключены.

**23.2 Интеграция**

CAN - это последовательный протокол связи, который эффективно поддерживает распределенное управлениe в реальном времени с высоким уровнем безопасности. Модуль DCAN поддерживает скорость передачи данных до 1 Мбит/с и соответствует спецификации протокола CAN 2.0B. Ядро IP для DCAN предоставлено компанией Bosch.

Это устройство включает в себя два варианта контроллера DCAN: DCAN0 и DCAN1. На рисунке 23-1 показанаинтеграция модуля DCAN.



**Рисунок 23-1. Интеграция DCAN**

**23.2.1 Атрибуты подключения DCAN**

Общие атрибуты подключения для модуля DCAN приведены в таблице 23-1.

**Таблица 23-1. Атрибуты подключения DCAN**

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| Power Domain | Peripheral Domain |
| Clock Domain | PD\_PER\_L4LS\_GCLK (OCP)  PD\_PER\_CAN\_CLK (Func) |
| Reset Signals | PER\_DOM\_RST\_N |
| Idle/Wakeup Signals | Smart Idle |
| Interrupt Requests | 3 Interrupts per instance  Intr0 (DCANx\_INT0) – Error, Status, Msg Object interrupt  Intr1 (DCANx\_INT1) – Msg Object interrupt  Uerr (DCANx\_PARITY) – Parity error interrupt  All DCAN0 interrupts to MPU Subsystem and PRU-ICSS  All DCAN1 interrupts to only MPU Subsystem |
| DMA Requests | 3 DMA requests per instance to EDMA (CAN\_IFxDMA) |
| Physical Address | L4 Peripheral slave port |

**23.2.2 Управление тактовыми импульсами и сбросом DCAN**

Контроллеры DCAN имеют разделенные интерфейс шины и функциональное тактирование.

**Таблица 23-2. Сигналы тактового генератора DCAN**

|  |  |  |  |
| --- | --- | --- | --- |
| **Clock Signal** | **Max Freq** | **Reference / Source** | **Comments** |
| DCAN\_ocp\_clk  Interface clock | 100 MHz | CORE\_CLKOUTM4 / 2 | pd\_per\_l4ls\_gclk  from PRCM |
| DCAN\_io\_clk  Functional clock | 26 MHz | CLK\_M\_OSC | pd\_per\_can\_clk  from PRCM |

**23.2.3 Список выводов DCAN**

Внешние сигналы модуля DCAN приведены в следующей таблице.

**Таблица 23-3. Список выводов DCAN**

|  |  |  |
| --- | --- | --- |
| **Pin** | **Type** | **Description** |
| DCAN*x*\_TX | O | DCAN transmit line |
| DCAN*x*\_RX | I | DCAN receive line |

**23.3 Функциональное описание**

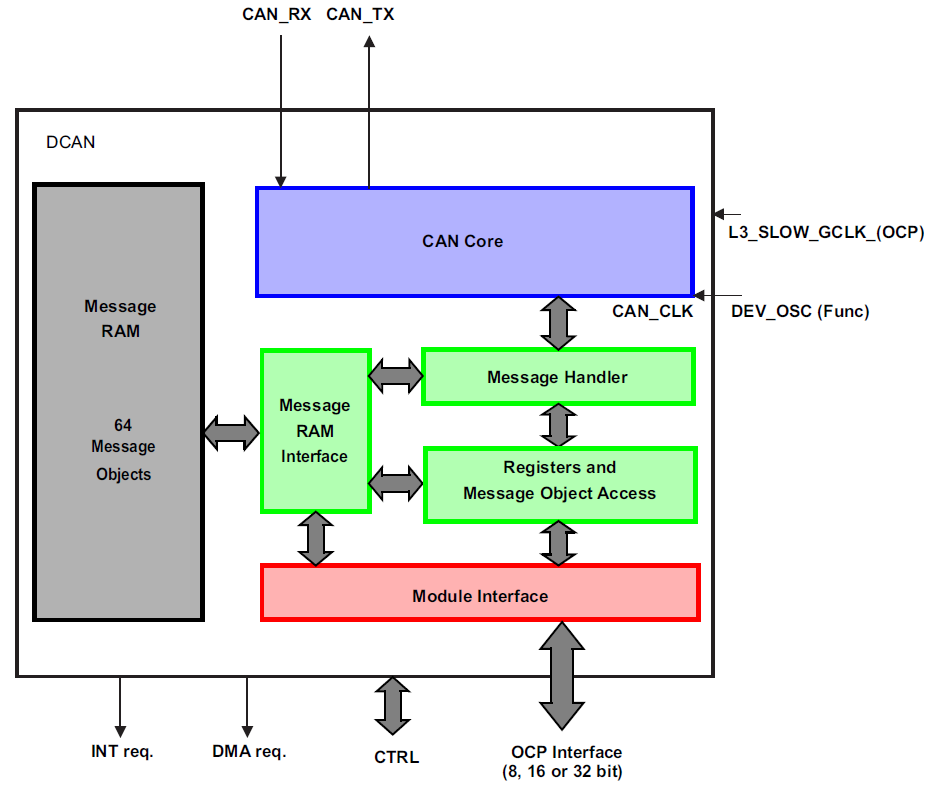
Модуль DCAN осуществляет обмен данными по протоколу CAN в соответствии с ISO 11898-1. Скорость передачи данных может быть запрограммирована на значения до 1 МБит/с. Для подключения к физическому уровню (шине CAN) требуется дополнительное приемопередающее оборудование физического уровня (шиной CAN).

Для связи в сети CAN можно сконфигурировать отдельные объекты сообщений. Объекты сообщений и маски идентификаторов хранятся в оперативной памяти сообщений. Все функции, связанные с обработкой сообщений, реализованы в обработчике сообщений. К таким функциям относятся фильтрация приема, передача сообщений между CAN-ядром и оперативной памятью сообщений(RAM), обработка запросов на передачу, а также генерация прерываний или запросов DMA.

Набор регистров модуля DCAN может быть доступен непосредственно центральному процессору через интерфейс модуля.Эти регистры используются для управления и конфигурирования CAN-ядра и обработчика сообщений, а также для доступа к оперативной памяти сообщений.

На рисунке 23-2 показана блок-схема модуля DCAN, а его особенности описаны ниже.

**Рисунок 23-2. Блок-схема DCAN**



**23.3.1 Ядро CAN**

Ядро CAN состоит из контроллера протокола CAN и регистра сдвига Rx/Tx. Оно обрабатывает все функции протокола ISO 11898-1.

**23.3.2 Обработчик сообщений**

Обработчик сообщений - это машина состояний, которая управляет передачей данных между однопортовым ОЗУ сообщений и сдвиговым регистром Rx/Tx CAN-ядра. Он также обрабатывает фильтрацию приема и генерацию запросов прерывания/DMA, как запрограммировано в регистрах управления.

**23.3.3 ОЗУ сообщений**

DCAN0 и DCAN1 позволяют хранить 64 сообщения CAN.

**23.3.4 Интерфейс ОЗУ сообщений**

Три набора интерфейсных регистров управляют доступом процессора к ОЗУ сообщений для чтения и записи. Имеются два набора интерфейсных регистров для доступа на чтение и запись, IF1 и IF2, и один набор интерфейсных регистров для доступа только на чтение, IF3. Дополнительную информацию можно найти в разделе 23.3.15.12.

Интерфейсные регистры имеют ту же длину слова, что и ОЗУ сообщений.

**23.3.5 Доступ к регистрам и объектам сообщений**

Согласованность данных обеспечивается косвенным доступом к объектам сообщений. Во время нормальной работы все доступы процессора и DMA к ОЗУ сообщений осуществляются через интерфейсные регистры. В специальном тестовом режиме, ОЗУ сообщений отображается на память и может быть напрямую доступно как процессору, так и DMA.

**23.3.6 Интерфейс модуля**

Доступ к регистрам модуля DCAN осуществляется центральным процессором или пользовательским программным обеспечением через 32-битный интерфейс периферийной шины.

**23.3.7 Двойной источник тактовых импульсов**

Модуль DCAN имеет два тактовых домена: периферийный синхронный тактовый домен

(L3\_SLOW\_GCLK) и домен периферийного асинхронного источника тактовых импульсов (CLK\_M\_OSC) для CAN\_CLK.

**23.3.8 Работа с CAN**

После аппаратного сброса бит Init в регистре управления CAN (CTL) установлен, и все функции протокола CAN отключены. Перед началом работы модуль CAN необходимо инициализировать. На рисунке 23-3 показан основной основной процесс инициализации модуля CAN.

**23.3.8.1 Инициализация модуля CAN**

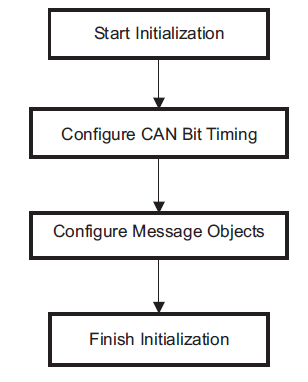
Общая инициализация CAN-модуля включает в себя следующие два важных этапа:

- Конфигурация синхронизации битов CAN

- Конфигурация объектов сообщений

Для инициализации CAN-контроллера процессор должен настроить синхронизацию битов CAN и те объекты сообщений, которые должны использоваться для связи по CAN. Объекты сообщений, которые не нужны, могут быть деактивированы.

**Рисунок 23-3. Общий поток инициализации модуля CAN**



**23.3.8.1.1 Конфигурация синхронизации битов CAN**

Для конфигурирования синхронизации битов CAN модуль должен находиться в режиме инициализации.

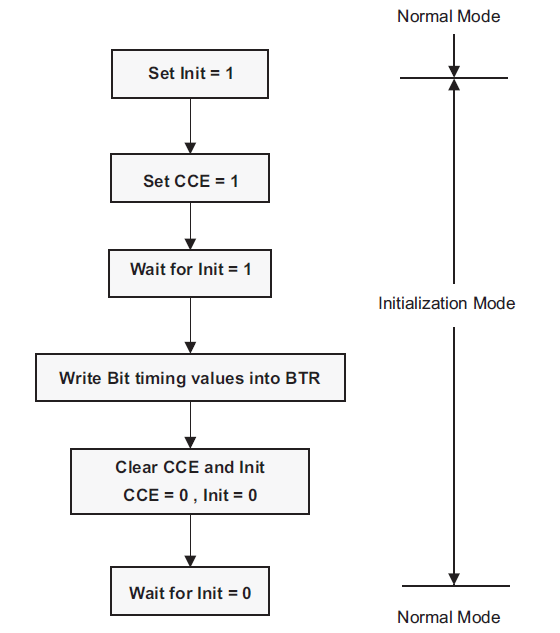
Поток программной конфигурации синхронизации битов CAN см. на рисунке 23-4.

Шаг 1: Войдите в режим инициализации, установив бит Init (Initialization) в регистре управления CAN.

Пока бит Init установлен, передача сообщений с и на шину CAN остановлена, а состояние выхода CAN\_TX является рецессивным (высоким). Счетчики ошибок CAN не обновляются. Установка бита Init не изменяет никаких других конфигурационных регистров.

Также следует учитывать, что CAN-модуль находится в режиме инициализации при аппаратном сбросе и во время отключения шины.

**Рисунок 23-4. Конфигурация синхронизации битов CAN**



Шаг 2: Установите бит Configure Change Enable (CCE) в регистре управления CAN.

Доступ к регистру битовой синхронизации (BTR) для конфигурирования битовой синхронизации разрешен, если установлены оба бита Init и CCE в регистре управления CAN.

Шаг 3: Дождитесь установки бита Init. Это позволит убедиться, что модуль вошел в режим инициализации.

Шаг 4: Запишите значения битовой синхронизации в регистр битовой синхронизации. См. раздел 23.3.16.2 для расчета значения BTR для заданного битового тайминга.

Шаг 5: Очистите биты CCE и Init.

Шаг 6: Дождитесь, пока бит Init очистится. Это гарантирует, что модуль вышел из режима инициализации.

После выполнения этих шагов модуль переходит в рабочий режим, синхронизируясь с шиной CAN, при условии, что BTR настроен в соответствии со скоростью передачи данных шины CAN, хотя объекты сообщений должны быть сконфигурированы перед выполнением любого обмена данными.

*ПРИМЕЧАНИЕ: Модуль не выйдет из режима инициализации, если в шаге 4 записаны неверные значения BTR.*

*ПРИМЕЧАНИЕ: Как описано в разделе 23.3.8.1, необходимые объекты сообщений должны быть сконфигурированы как объекты передачи или приема до начала передачи данных.*

**23.3.8.1.2 Конфигурация объектов сообщений**

Объекты сообщений могут быть сконфигурированы только через интерфейсные регистры; процессор не имеет прямого доступа к объекту сообщения (RAM сообщений). Ознакомьтесь с набором интерфейсных регистров (IFx) (см. раздел 23.3.17) и структурой объектов сообщений (см. раздел 23.3.18) перед конфигурированием объектов сообщений.

Более подробную информацию о процедуре конфигурирования объектов сообщений см. в разделе 23.3.14. Все объекты сообщений должны быть настроены на определенные идентификаторы или установлены на недействительные до начала передачи сообщения. Конфигурацию объектов сообщений можно изменить во время нормальной работы (то есть между передачами данных).

*ПРИМЕЧАНИЕ: Инициализация объектов сообщений не зависит от конфигурации битовой синхронизации.*

**23.3.8.1.3 Аппаратная инициализация оперативной памяти DCAN**

Аппаратная инициализация памяти для модуля DCAN включается в регистре управления устройством (DCAN\_RAMINIT), который инициализирует оперативную память нулями и устанавливает биты четности соответствующим образом. Дождитесь RAMINIT\_DONE, чтобы убедиться в успешной инициализации ОЗУ. Убедитесь, что тактирование модуля DCAN включено перед началом такой инициализации.

Более подробную информацию об аппаратной инициализации ОЗУ см. в главе 9 «Модуль управления».

**23.3.8.2 Передача сообщений CAN (нормальная работа)**

После инициализации DCAN и сброса бита Init в ноль ядро CAN синхронизируется с шиной CAN и готово к передаче сообщений в соответствии с конфигурированными объектами сообщений.

Процессор может включить линии прерываний (установив IE0 и IE1 в '1') одновременно с очисткой Init и CCE. Статусные прерывания EIE и SIE могут быть включены одновременно.

Связь по CAN может осуществляться в любом из следующих двух режимов: прерывание и опрос.

Регистр прерываний указывает на те объекты сообщений, для которых IntPnd = '1'. Он обновляется, даже если линии прерывания линии прерываний для CPU отключены (IE0/IE1 равны нулю).

CPU может параллельно опрашивать все биты NewDat и TxRqst объекта сообщения из регистров NewData X и регистров запроса передачи X (TXRQ X). Опрос может быть упрощен, если все объекты передачи сгруппированы по младшим номерам, а все объекты приема - по старшим.

Принятые сообщения сохраняются в соответствующих объектах сообщений, если они прошли фильтрацию приема.

Все сообщение (включая все арбитражные биты, DLC и до восьми байтов данных) сохраняется в объекте сообщений. Как следствие (например, при использовании маски идентификатора), арбитражные биты, которые маскируются маской «неважно», могут измениться в объекте сообщения при сохранении принятого сообщения.

Центральный процессор может читать или записывать каждое сообщение в любое время через интерфейсные регистры, поскольку обработчик сообщений гарантирует согласованность данных при одновременном доступе.

Если для сообщения существует постоянный объект сообщения (арбитражные и управляющие биты, установленные при конфигурировании и остающиеся для сообщения), то у него можно обновить только байты данных.

Если одному объекту сообщения должно быть назначено несколько передаваемых сообщений, весь объект сообщения должен быть сконфигурирован до того, как будет запрошена передача этого сообщения.

Одновременно может быть запрошена передача нескольких объектов сообщений. Впоследствии они передаются в соответствии с их внутренним приоритетом.

Сообщения могут быть обновлены или признаны недействительными в любой момент, даже если запрошенная передача еще не завершена. Однако байты данных будут отброшены, если сообщение будет обновлено до того, как будет начата ожидающая передача.

В зависимости от конфигурации объекта сообщения, передача может быть запрошена автоматически при получении удаленного кадра с соответствующим идентификатором.

**23.3.8.2.1 Автоматическая ретрансляция**

В соответствии со спецификацией CAN (ISO11898), DCAN обеспечивает механизм автоматической повторной передачи кадров, которые потеряли арбитраж или были нарушены ошибками во время передачи. Услуга передачи кадра не будет подтверждена пользователю до успешного завершения передачи.

По умолчанию эта автоматическая ретрансляция включена. Ее можно отключить, установив параметр disable automatic retransmission (DAR) в регистре CTL. Более подробная информация об этом режиме приведена в Разделе 23.3.15.3.

**23.3.8.2.2 Автоматическое включение шины**

По умолчанию после перехода DCAN в состояние Bus-Off процессор может запустить последовательность Bus-Off-Recovery путем сброса бита Init. Если этого не сделать, модуль останется в состоянии Bus-Off.

В DCAN предусмотрена функция автоматического включения шины, которая включается битом ABO в регистре CTL. Если он установлен, модуль DCAN автоматически запустит последовательность Bus-Off-Recovery. Последовательность может быть отложена на заданное пользователем количество циклов L3\_SLOW\_GCLK, которое может быть определено в регистре Auto-Bus-On Time register (ABOTR).

*ПРИМЕЧАНИЕ: Если DCAN переходит в состояние Bus-Off из-за массового появления ошибок шины CAN, он останавливает все действия на шине и автоматически устанавливает бит Init. После того как бит Init будет сброшен процессором или благодаря функции Auto-Bus-On, устройство будет ждать 129 событиев простоя шины (равных 129 \* 11 последовательных рецессивных битов) перед возобновлением нормальной работы. В конце последовательности восстановления Bus-Off счетчики ошибок будут сброшены.*

**23.3.8.3 Режимы тестирования**

Модуль DCAN обеспечивает несколько режимов тестирования, которые в основном предназначены для производственных испытаний или самотестирования.

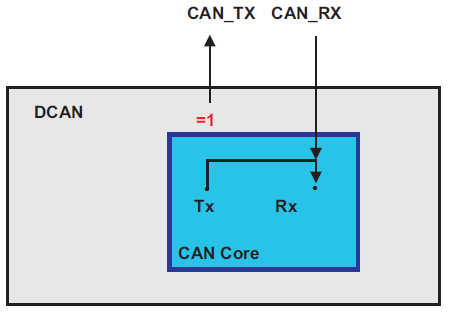
Для всех режимов тестирования бит Test в регистре CTL должен быть установлен в единицу. Это разрешает доступ на запись к регистру тестирования (TEST).

**23.3.8.3.1 Бесшумный режим**

Тихий режим может использоваться для анализа трафика на шине CAN, не влияя на него путем отправки доминирующих битов (например, бит подтверждения, флаг перегрузки, флаг активной ошибки). DCAN по-прежнему может принимать действительные кадры данных и действительные удаленные кадры, но не будет посылать никаких доминирующих битов. Однако они внутренне направляются в ядро CAN.

На рисунке 23-5 показано подключение сигналов CAN\_TX и CAN\_RX к ядру CAN в тихом режиме. Бесшумный режим может быть активирован установкой бита Silent в регистре TEST в единицу. В стандарте ISO 11898-1 бесшумный режим называется режимом мониторинга шины.

**Рисунок 23-5. Ядро CAN в бесшумном режиме**



**23.3.8.3.2 Режим обратной связи**

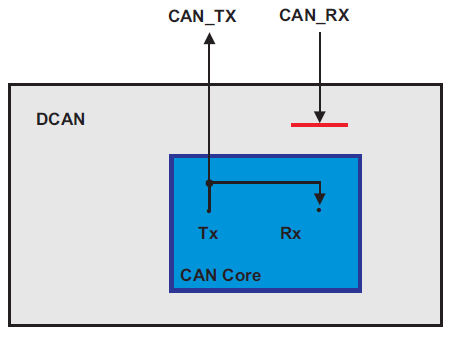
Режим обратной связи предназначен в основном для функций самотестирования оборудования. В этом режиме ядро CAN использует внутреннюю обратную связь с выхода Tx на вход Rx. Переданные сообщения рассматриваются как принятые и могут быть сохранены в объектах сообщений, если они прошли приемочную фильтрацию. Фактическое значение входного вывода CAN\_RX игнорируется ядром CAN. Переданные сообщения по-прежнему можно отслеживать на выводе CAN\_TX.

Для того чтобы быть независимым от внешней стимуляции, ядро CAN игнорирует сэмплированное в слот подтверждения кадра данных/дистанционного кадра.

На рисунке 23-6 показано подключение сигналов CAN\_TX и CAN\_RX к CAN-ядру в режиме loopback. Режим обратной связи может быть активирован установкой бита LBack в регистре TEST в единицу.

*ПРИМЕЧАНИЕ: В режиме обратной связи путь сигнала от ядра CAN к контакту Tx, сам контакт Tx и путь сигнала обратно к CAN-ядру не учитываются. Для включения их в тестирование см.Раздел 23.3.8.3.3.*

**Рисунок 23-6. Ядро CAN в режиме обратной связи**



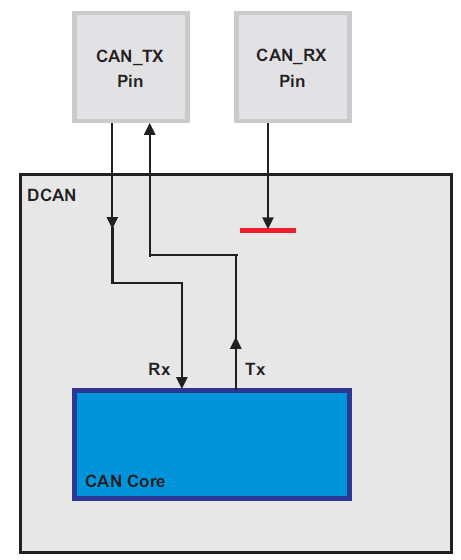
**23.3.8.3.3 Режим внешней обратной связи**

Режим внешней обратной связи аналогичен режиму обратной связи; однако он включает в себя путь сигнала от ядра CAN к контакту Tx, сам контакт Tx и путь сигнала от контакта Tx обратно к ядру CAN. Когда выбрана внешняя обратная связь, вход CAN-ядра подключается к входному буферу Tx-контакта. При такой конфигурации можно протестировать цепь ввода-вывода вывода Tx.

Режим внешней обратной связи может быть активирован установкой бита EXL в регистре TEST в единицу.На рисунке 23-7 показано подключение сигналов CAN\_TX и CAN\_RX к CAN-ядру в режиме внешнего шлейфа.

*ПРИМЕЧАНИЕ: Если режим обратной связи активен (установлен бит LBack), бит ExL будет игнорироваться.*

**Рисунок 23-7. CAN-ядро в режиме внешнего шлейфа**

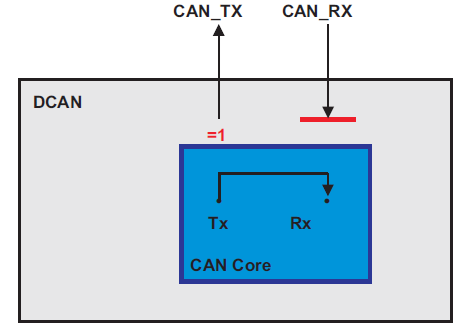


**23.3.8.3.4 Режим обратной связи в сочетании с бесшумным режимом**

Также можно совместить режим обратной связи и режим молчания, установив биты LBack и Silent одновременно. Этот режим может быть использован для «горячего самотестирования», то есть аппаратное обеспечение DCAN может быть протестировано без затрагивания сети CAN. В этом режиме контакт CAN\_RX отключается от ядра CAN, и никакие доминирующие биты не будут отправляться на контакт CAN\_TX.

На рисунке 23-8 показано подключение сигналов CAN\_TX и CAN\_RX к CAN-ядру в случае комбинации режима loopback с режимом silent.

**Рисунок 23-8. Ядро CAN в режиме внешней обратной связи в сочетании с бесшумным режимом**



**23.3.8.3.5 Программное управление выводом CAN\_TX**

Для вывода CAN-передачи CAN\_TX доступны четыре функции вывода. В дополнение к своей функции по умолчанию (вывод последовательных данных), вывод CAN\_TX может передавать постоянные доминантные или рецессивные значения, или может передавать сигнал CAN sample point signal для контроля синхронизации битов CAN-ядра.

В сочетании с читаемым значением вывода CAN\_RX эта функция может быть использована для проверки физического уровня шины CAN.

Режим вывода сигнала CAN\_TX выбирается путем программирования битов Tx[1:0] регистра TEST.

*ПРИМЕЧАНИЕ: Программное управление выводом CAN\_TX вмешивается в функции протокола CAN. Для CAN передачи сообщений или любого из тестовых режимов (режим с обратной связью, режим внешнего обратной связи или режим «тишины), контакт CAN\_TX должен работать в режиме по умолчанию.*

**23.3.9 Двойной источник тактовых импульсов**

Для модуля DCAN предусмотрено два тактовых домена: периферийный синхронный тактовый домен (L3\_SLOW\_GCLK) в качестве общего источника тактовых импульсов модуля, и периферийный асинхронный источник тактовых импульсов домен (CLK\_M\_OSC), передаваемый ядру CAN (как источник CAN\_CLK) для генерации битов CAN синхронизации.

Оба тактовых домена могут быть получены от одного и того же источника (так что L3\_SLOW\_GCLK = CLK\_M\_OSC).

Дополнительную информацию о том, как настроить соответствующие регистры источников тактовых импульсов в системном модуле, см. Главу 8, Управление питанием и тактовыми частотами.

Между двумя тактовыми доменами в модуле DCAN реализован механизм синхронизации для обеспечения корректной передачи данных.

*ПРИМЕЧАНИЕ: Если используется функция двойного тактового генератора, то L3\_SLOW\_GCLK всегда должен быть выше или равен CAN\_CLK (CLK\_M\_OSC) (получаемый от источника асинхронного тактового сигнала), чтобы достичь стабильной функциональности DCAN. Здесь также учитывается сдвиг частоты модулированного сигнала L3\_SLOW\_GCLK:*

*f0, L3\_SLOW\_GCLK(OCP) ± ΔfFM, L3\_SLOW\_GCLK(OCP) ≥ fCANCLK.*

*ПРИМЕЧАНИЕ: Ядро CAN должно быть запрограммировано как минимум на 8 тактов для отсчета периода получения бита. Для достижения скорости передачи данных 1 Мбод при использовании асинхронного тактового домена в качестве источника тактового сигнала для CAN\_CLK (CLK\_M\_OSC), необходимо использовать частоту генератора 8 МГц или выше.*

**23.3.10 Функциональность прерываний**

Прерывания могут генерироваться на двух линиях прерываний: DCANINT0 и DCANINT1. Эти линии могут быть активированы установкой битов IE0 и IE1 соответственно в регистре CTL. Прерывания запускаются по уровню на уровне микросхемы.

DCAN обеспечивает три группы источников прерываний: прерывания объектов сообщений, прерывания изменения состояния, прерывания изменения состояния и прерывания ошибок (см. Рисунок 23-9 и Рисунок 23-10).

Источник прерывания можно определить по идентификаторам прерывания Int0ID/Int1ID в регистре прерываний (см. Регистр INT). Если прерывание не ожидается, регистр будет иметь нулевое значение.

Каждая линия прерывания остается активной до тех пор, пока выделенное поле (Int0ID или Int1ID) в регистре прерываний (INT) снова достигнет нуля, что означает, что причина прерывания сброшена, или пока IE0 или IE1 не будут сброшены.

Значение 0x8000 в поле Int0ID указывает на то, что прерывание ожидается, поскольку ядро CAN обновило (не обязательно изменило) регистр ошибок и состояния (ES). Это прерывание имеет наивысший приоритет. Центральный процессор может обновить (сбросить) биты состояния WakeUpPnd, RxOk, TxOk и LEC, прочитав регистр ES, но доступ процессора на запись никогда не сгенерирует и не сбросит прерывание.

Значения между 1 и номером последнего объекта сообщения указывают на то, что источником прерывания является один из объектов сообщения, Int0ID или Int1ID будет указывать на ожидающее прерывание сообщения с наивысшим приоритетом. Объект сообщения 1 имеет наивысший приоритет; последний объект сообщения имеет наименьший приоритет.

Подпрограмма обслуживания прерываний, которая считывает сообщение, являющееся источником прерывания, может прочитать его сообщение и сбросить IntPnd объекта сообщения в одно и то же время (бит ClrIntPnd в регистре IF1CMD или IF2CMD).Когда IntPnd будет сброшен, регистр прерываний будет указывать на следующий объект сообщения , ожидающий прерывания.

**23.3.10.1 Прерывания объектов сообщений**

Прерывания объектов сообщений генерируются событиями от объектов сообщений. Они управляются флагами IntPND, TxIE и RxIE, которые описаны в разделе 23.3.18.1.

Прерывания объектов сообщений могут быть направлены на линию DCANINT0 или DCANINT1, управляемую регистром мультиплексора прерываний (INTMUX12 - INTMUX78).

**23.3.10.2 Прерывания по изменению состояния**

События WakeUpPnd, RxOk, TxOk и LEC в регистре ES относятся к прерываниям изменения состояния. Группа прерываний изменения состояния может быть включена битом в регистре CTL.

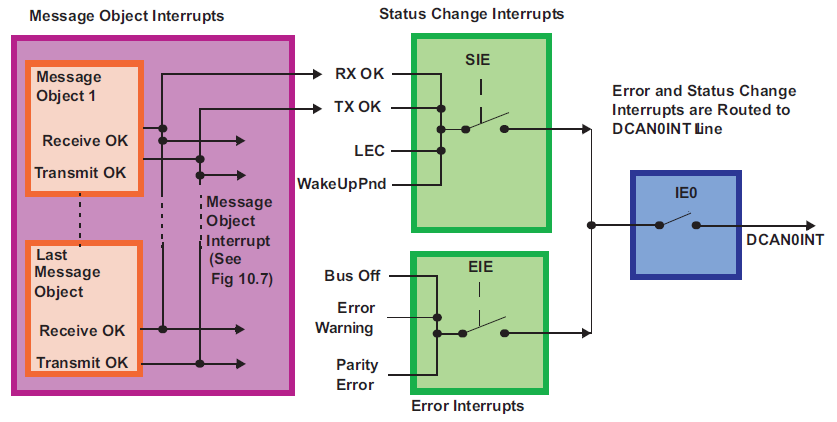
Если SIE установлен, прерывание по изменению состояния будет генерироваться в каждом CAN-кадре, независимо от ошибок шины и также независимо от конфигурации RAM сообщений. Прерывания изменения состояния могут быть направлены только на линию прерывания DCAN0INT, которая должна быть разрешена установкой бита IE0 в регистре CTL.

*ПРИМЕЧАНИЕ: Чтение регистра ошибок и состояния снимает флаг WakeUpPnd. Если в режиме глобального отключения питания, флаг WakeUpPnd будет сброшен таким доступом на чтение до того, как модуль DCAN будет разбужен системой. DCAN может снова установить флаг WakeUpPnd и произойдет второе прерывание.*

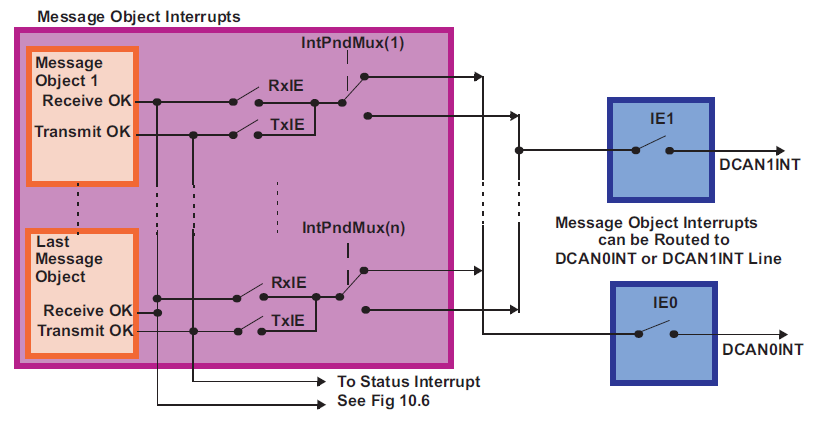
**23.3.10.3 Прерывания по ошибкам**

События PER, BOff и EWarn, отслеживаемые в регистре ES, относятся к прерываниям ошибок. Группа прерываний по ошибкам может быть включена установкой бита EIE в регистре CTL. Прерывания по ошибкам могут быть направлены только на линию прерывания DCAN0INT, которая должна быть включена установкой бита IE0 в регистре CTL.

**Рисунок 23-9. Топология прерываний CAN 1**



**Рисунок 23-10. Топология прерываний CAN 2**



**23.3.11 Режим локального отключения питания**

DCAN поддерживает режим локального отключения питания, которым можно управлять с помощью регистра CTL.

**23.3.11.1 Вход в режим локального отключения питания**

Локальный режим отключения питания запрашивается установкой бита PDR в регистре CTL.

После этого DCAN завершает все запросы на передачу объектов сообщений. Когда все запросы выполнены, модуль DCAN ждет, пока не будет распознано состояние простоя шины. Модуль автоматически устанавливает бит Init в регистре CTL, чтобы предотвратить дальнейшую передачу данных по CAN, а также установит бит PDA в регистре ошибок и состояния CAN (ES). Установкой бита PDA модуль DCAN указывает, что был введен режим локального отключения питания.

В режиме локального отключения питания внутренние часы модуля DCAN выключены, но имеется логика пробуждения (см. раздел 23.3.11.2), которая может быть активна, если разрешена. Кроме того также может быть считано фактическое содержимое управляющих регистров.

*ПРИМЕЧАНИЕ: В режиме локального пониженного энергопотребления приложение не должно сбрасывать бит Init, пока установлен PDR. Если в RAM сообщений есть сообщения, которые сконфигурированы как сообщения передачи, а приложение сбрасывает бит init, эти сообщения могут быть отправлены.*

**23.3.11.2 Пробуждение после локального отключения питания**

Существует два способа пробуждения DCAN из режима локального отключения питания:

- Приложение может разбудить модуль DCAN вручную, сбросив бит PDR, а затем сбросив бит Init в регистре CTL.

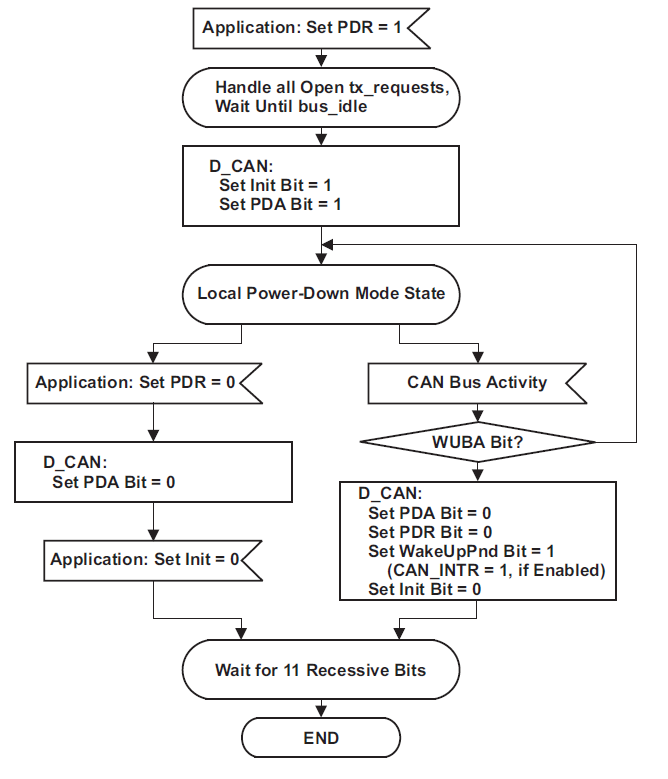
- В качестве альтернативы можно активировать схему обнаружения активности шины CAN, установив бит wakeup on bus activity (WUBA) в регистре CTL. Если эта схема активна, то при появлении доминирующего уровня на шине DCAN автоматически запустит последовательность пробуждения. При этом будет сброшен бит PDR в регистре CTL и также сбросит бит PDA в регистре ошибок и состояния. Бит WakeUpPnd в регистре ES будет установлен. Если включены статусные прерывания, также будет сгенерировано прерывание. Наконец, бит Init в регистре CTL будет очищен.

После сброса бита Init модуль ждет, пока не обнаружит 11 последовательных рецессивных битов на выводе CAN\_RX, а затем снова переходит в режим активной шины.

*ПРИМЕЧАНИЕ: Схема приемопередатчика CAN должна оставаться активной, чтобы обнаружить любую активность шины CAN, пока DCAN находится в режиме локального отключения питания. Первое сообщение CAN, которое инициирует активность шины не может быть получено. Это означает, что первое сообщение, полученное в режиме отключения питания и в режиме автоматического пробуждения, будет потеряно.*

На рисунке 23-11 показана блок-схема входа и выхода из режима локального отключения питания.

**Рисунок 23-11. Блок-схема режима локального отключения питания**



**23.3.12 Механизм проверки четности**

В DCAN предусмотрен механизм проверки четности для обеспечения целостности данных в оперативной памяти сообщений. Для каждого слова (32 бита) в RAM сообщений вычисляется один бит четности. Формирование различных слов осуществляется в соответствии с представлением RAM сообщений в режиме RDA, см. раздел 23.3.18.4.

Информация о четности сохраняется в RAM сообщений при доступе на запись и сверяется с сохраненным битом четности из RAM сообщений при доступе на чтение. Функциональность проверки четности может быть включена или отключена битовым полем PMD в регистре CTL. В случае отключенной проверки четности, биты четности в оперативной памяти сообщений будут оставлены без изменений при доступе к области данных на запись , а при доступе на чтение проверка не будет производиться.

Если проверка четности включена, биты четности будут автоматически генерироваться и проверяться DCAN. Биты четности могут быть считаны в режиме отладки/приостановки (см. раздел 23.3.18.3) или в режиме RDA (см.Раздел 23.3.18.4). Однако прямой доступ к битам четности возможен только в этих двух режимах при отключенной проверке четности.

Бит четности будет установлен, если сумма битов данных по модулю 2 равна 1. Это определение эквивалентно следующему: Бит четности будет установлен, если количество битов 1 в данных нечетно.

*ПРИМЕЧАНИЕ: Схема четности привязана к четности на уровне устройства.*

**23.3.12.1 Поведение при ошибке четности**

При любом доступе на чтение к оперативной памяти сообщений (например, при начале передачи кадра CAN) проверяется четность объекта сообщения. Если обнаружена ошибка четности, в регистре ES будет установлен бит PER. Если прерывания ошибок, то также будет сгенерировано прерывание. Чтобы избежать передачи недопустимых данных по шине CAN, бит D объекта сообщения будет сброшен.

Данные объекта сообщения могут быть прочитаны центральным процессором независимо от ошибок четности. Таким образом приложение должно убедиться, что считанные данные действительны, например, путем немедленной проверки регистра кода ошибок четности (PERR) при прерывании по ошибке четности.

*ПРИМЕЧАНИЕ: При инициализации ОЗУ проверка на четность не производится.*

**23.3.12.2 Тестирование четности**

Проверка механизма контроля четности может быть выполнена путем включения бита RamDirectAccess (RDA) и ручной записи битов четности непосредственно в выделенные места RAM. Таким образом, данные и биты четности могут быть проверены при чтении непосредственно из оперативной памяти.

*ПРИМЕЧАНИЕ: Если проверка четности была отключена, приложение должно обеспечить правильную обработку битов четности, чтобы предотвратить ошибки четности в дальнейшем, когда проверка четности будет включена.*

**23.3.13 Режим отладки/приостановки**

Модуль поддерживает использование внешнего отладочного устройства, предоставляя такие функции, как приостановка работы DCAN и доступ к содержимому ОЗУ сообщений через интерфейс OCP.

Перед переходом в режим отладки/приостановки схема либо дожидается завершения начатой передачи , и будет распознано состояние простоя шины, либо немедленно прервет текущую передачу или прием. Это зависит от бита IDS в регистре CTL. После этого DCAN переходит в режим отладки/приостановки, на что указывает флаг InitDbg в регистре CTL.

В режиме отладки/приостановки возможен доступ ко всем регистрам DCAN. Чтение зарезервированных битов вернет '0'. Запись в зарезервированные биты не будет иметь никакого эффекта.

Кроме того, RAM сообщений будет отображено в памяти. Это позволяет внешнему отладочному устройству считывать сообщение RAM. Об организации памяти см. раздел 23.3.18.3).

ПРИМЕЧАНИЕ: В режиме отладки/приостановки доступ к RAM сообщений через наборы регистров IFx невозможен.Запись в управляющие регистры в режиме отладки/приостановки может повлиять на работу машины состояний CAN и дальнейшую обработку сообщений.

Для поддержки отладки функция автоматической очистки следующих регистров DCAN отключена:

- Регистр ES (очистка флагов состояния при чтении)

- IF1CMD и IF2CMD (сброс флага DMAActive при чтении/записи).

**23.3.14 Конфигурирование объектов сообщений**

Вся оперативная память сообщений должна быть сконфигурирована до конца инициализации, однако можно также изменить конфигурацию объектов сообщений во время CAN-коммуникации.

Библиотека драйверов программного обеспечения CAN должна предлагать подпрограммы, которые:

- Передают полную структуру сообщения в объект сообщения. (Конфигурация)

- Передают байты данных сообщения в объект сообщения и устанавливать TxRqst и NewDat. (Начать новую передачу)

- Получить байты данных сообщения из объекта сообщения и очистить NewDat (и IntPnd). (Прочитать полученные данные)

- Получить полное сообщение из объекта сообщения и очистить NewDat (и IntPnd). (Прочитать полученное сообщение, включая идентификатор, из объекта сообщения с UMask = '1')

Параметрами подпрограмм являются номер сообщения и указатель на полную структуру сообщения или на байты данных структуры сообщения.

Здесь показаны два примера назначения наборов регистров интерфейса IFx этим подпрограммам:

В первом случае задачи прикладной программы, которые могут обращаться к модулю, делятся на две группы. Каждая группа ограничивается использованием одного из наборов интерфейсных регистров. Задачи одной группы могут прерывать задачи другой группы, но не одной и той же группы.

Во втором методе, который может быть частным случаем первого метода, в прикладной программе есть только две задачи, которые обращаются к модулю прикладной программы, которые обращаются к модулю. Задача Read\_Message, которая использует IF2 или IF3 для получения принятых сообщений из оперативной памяти сообщений, и задача Write\_Message, которая использует IF1 для записи сообщений для передаваемых (или конфигурируемых) в оперативную память сообщений. Обе задачи могут прерывать друг друга.

**23.3.14.1 Конфигурация объекта передачи для кадров данных**

В таблице 23-4 показано, как может быть инициализирован объект передачи.

**Таблица 23-4. Инициализация объекта передачи**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MsgVal** | **Arb** | **Data** | **Mask** | **EoB** | **Dir** | **NewDat** | **MsgLst** | **RxIE** | **TxIE** | **IntPnd** | **RmtEn** | **TxRqst** |
| 1 | appl. | appl. | appl. | 1 | 1 | 0 | 0 | 0 | appl. | 0 | appl. | 0 |

Биты арбитража (ID[28:0] и бит Xtd) задаются приложением. Они определяют идентификатор и тип исходящего сообщения. Если используется 11-битный идентификатор (стандартный кадр) (Xtd = '0'), он программируется на ID[28:18]. В этом случае ID[17:0] можно игнорировать.

Регистры данных (DLC[3:0] и Data0-7) задаются приложением. TxRqst и RmtEn не должны быть установлены до того, как данные станут действительными.

Если установлен бит TxIE, то бит IntPnd будет установлен после успешной передачи объекта сообщения.

Если бит RmtEn установлен, то соответствующий полученный удаленный кадр вызовет установку бита TxRqst; удаленный кадр будет автономно отвечен кадром данных.

Биты маски (Msk[28:0], UMask, MXtd и MDir) могут быть использованы (UMask='1') для того, чтобы группы удаленных кадров с одинаковыми идентификаторами могли установить бит TxRqst. Бит Dir не должен быть замаскирован. Подробнее, см. раздел 23.3.15.8. Маскировка идентификаторов должна быть отключена (UMask = '0'), если не разрешается удаленным кадрам устанавливать бит TxRqst (RmtEn = '0').

**23.3.14.2 Конфигурация объекта передачи для удаленных кадров**

Нет необходимости конфигурировать объекты передачи для передачи удаленных кадров. Установка TxRqst для объекта приема вызывает передачу удаленного кадра с тем же идентификатором, что и кадр данных для которого сконфигурирован этот объект приема.

**23.3.14.3 Конфигурация одного объекта приема для кадров данных**

В таблице 23-5 показано, как может быть инициализирован объект приема для кадров данных.

**Таблица 23-5. Инициализация одного объекта приема для кадров данных**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MsgVal** | **Arb** | **Data** | **Mask** | **EoB** | **Dir** | **NewDat** | **MsgLst** | **RxIE** | **TxIE** | **IntPnd** | **RmtEn** | **TxRqst** |
| 1 | appl. | appl. | appl. | 1 | 0 | 0 | 0 | appl. | 0 | 0 | 0 | 0 |

Биты арбитража (ID[28:0] и бит Xtd) задаются приложением. Они определяют идентификатор и тип принимаемых сообщений. Если используется 11-битный идентификатор (стандартный кадр) (Xtd = '0'), он программируется в ID[28:18]. В этом случае ID[17:0] можно игнорировать. При получении кадра данных с 11-битным идентификатором, ID[17:0] устанавливается в '0'.

Код длины данных (DLC[3:0]) задается приложением. Когда обработчик сообщения сохраняет кадр данных в объект сообщения, он сохраняет полученный код длины данных и восемь байтов данных. Если код длины данных меньше 8, оставшиеся байты объекта сообщения могут быть перезаписаны не определенными значениями.

Биты маски (Msk[28:0], UMask, MXtd и MDir) могут быть использованы (UMask = '1') для того, чтобы группы кадров данных с одинаковыми идентификаторами. Бит Dir не следует маскировать в обычных приложениях. Если некоторые биты маски установлены в значение «безразлично», соответствующие биты регистра арбитража будут перезаписаны битами хранимого кадра данных.

Если бит RxIE установлен, бит IntPnd будет установлен, когда принятый кадр данных будет принят и сохранен в объекте сообщения.

Если установлен бит TxRqst, то будет инициирована передача удаленного кадра с тем же идентификатором, что и в арбитражных битах. Содержимое арбитражных битов может измениться, если используются биты маски (UMask = '1') для фильтрации приема.

**23.3.14.4 Конфигурация одного объекта приема для удаленных кадров**

В таблице 23-6 показано, как можно инициализировать объект приема для удаленных кадров.

**Таблица 23-6. Инициализация одного объекта приема для удаленных кадров**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MsgVal** | **Arb** | **Data** | **Mask** | **EoB** | **Dir** | **NewDat** | **MsgLst** | **RxIE** | **TxIE** | **IntPnd** | **RmtEn** | **TxRqst** |
| 1 | appl. | appl. | appl. | 1 | 1 | 0 | 0 | appl. | 0 | 0 | 0 | 0 |

Объект приема удаленных кадров может использоваться для мониторинга удаленных кадров на шине CAN. Удаленный кадр, сохраненный в объекте приема, не будет вызывать передачу кадра данных. Объекты приема для удаленных кадров могут быть расширены до буфера FIFO (см. раздел 23.3.14.5). UMask должен быть установлен в '1'. Биты маски (Msk[28:0], UMask, MXtd и MDir) могут быть установлены в значение «mustmatch» или «все равно», чтобы обеспечить прием групп удаленных кадров с одинаковыми идентификаторами. Бит Dir не должен маскироваться в обычных приложениях. Подробнее см. раздел 23.3.15.8.

Биты арбитража (ID[28:0] и бит Xtd) могут быть заданы приложением. Они определяют идентификатор и тип принимаемых удаленных кадров. Если некоторые биты маски установлены в значение «неважно», то соответствующие биты арбитражных битов будут перезаписаны битами сохраненного удаленного кадра. Если 11- бит идентификатора (стандартный кадр) (Xtd = '0'), он программируется в ID[28:18]. В этом случае ID[17:0] можно игнорировать. При получении удаленного кадра с 11-битным идентификатором ID[17:0] будет установлен в '0'.

Код длины данных (DLC[3:0]) может быть задан приложением. Когда обработчик сообщения сохраняет удаленный кадр в объект сообщения, он сохранит полученный код длины данных. Байты данных объекта сообщения остаются неизменными.

Если установлен бит RxIE, то бит IntPnd будет установлен, когда принятый удаленный кадр будет принят и сохранен в объекте сообщения.

**23.3.14.5 Конфигурация буфера FIFO**

За исключением бита EoB, конфигурация объектов приема, принадлежащих буферу FIFO, такая же, как и конфигурация одного объекта приема.

Чтобы объединить несколько объектов сообщений в буфер FIFO, идентификаторы и маски (если используются) этих объектов должны быть запрограммированы на совпадающие значения. Из-за неявного приоритета объектов сообщений объект сообщения с наименьшим номером будет первым объектом сообщения в буфере FIFO.

Биты EoB всех объектов сообщений буфера FIFO, кроме последнего, должны быть запрограммированы на нуль.

Бит EoB последнего объекта сообщения буфера FIFO устанавливается в единицу, конфигурируя его как конец блока.

**23.3.15 Обработка сообщений**

После завершения инициализации модуль DCAN синхронизируется с трафиком на шине CAN. Он выполняет фильтрацию принятых сообщений и сохраняет те кадры, которые были приняты, в назначенные объекты сообщений. Приложению необходимо обновить данные передаваемых сообщений, разрешить и запрашивать их передачу. Передача запрашивается автоматически при получении соответствующего удаленного кадра.

Приложение может читать сообщения, которые были получены и приняты. Сообщения, которые не были прочитаны до того, как было принято следующее сообщение для того же объекта сообщения, будут перезаписаны.

Сообщения могут быть прочитаны по прерыванию или после опроса NewDat.

**23.3.15.1 Обзор обработчика сообщений**

Машина состояний обработчика сообщений управляет передачей данных между сдвиговым регистром Rx/Tx ядра CAN и оперативной памятью сообщений. Он выполняет следующие задачи:

- Передача данных из RAM сообщений в CAN-ядро (передаваемые сообщения)

- Передача данных из CAN-ядра в RAM сообщений (принятые сообщения)

- Передача данных от CAN-ядра к блоку фильтрации приема

- Сканирование оперативной памяти сообщений в поисках соответствующего объекта сообщения (приемная фильтрация)

- Сканирование того же объекта сообщения после его изменения регистрами IF1/IF2, если приоритет такой же или выше, чем у объекта сообщения, найденного при последнем сканировании

- Обработка флагов TxRqst

- Обработка флагов прерываний

Регистры обработчика сообщений содержат флаги состояния всех объектов сообщений, сгруппированные в следующие темы:

- Флаги запроса на передачу

- Флаги новых данных

- Флаги ожидания прерывания

- Регистры валидности сообщений

Вместо того чтобы собирать перечисленную выше информацию о состоянии каждого объекта сообщения через регистры IFx отдельно, эти регистры обработчика сообщений обеспечивают быстрый и простой способ получить обзор, например, обо всех ожидающих запросах на передачу.

Все регистры обработчика сообщений доступны только для чтения.

**23.3.15.2 Приоритет приема/передачи**

Приоритет приема/передачи для объектов сообщений привязан к номеру сообщения, а не к CAN идентификатору. Объект сообщения 1 имеет наивысший приоритет, а последний реализованный объект сообщения имеет самый низкий приоритет. Если ожидается несколько запросов на передачу, они обслуживаются в соответствии с приоритетом соответствующего объекта сообщения, поэтому сообщения с наивысшим приоритетом, например, могут быть помещены в объекты сообщений с наименьшими номерами.

Фильтрация приема принятых кадров данных или удаленных кадров также выполняется в порядке возрастания объектов сообщений, поэтому кадр, принятый объектом сообщения, не может быть принят другим объектом сообщения с более высоким номером. Последний объект сообщения может быть настроен на прием любого кадра данных или удаленного кадр, который не был принят ни одним другим объектом сообщения, для узлов, которым необходимо регистрировать весь трафик сообщений на шине CAN.

**23.3.15.3 Передача сообщений в CAN-коммуникации, управляемой событиями**

Если сдвиговый регистр CAN-ядра готов к загрузке и если нет передачи данных между IFx и RAM сообщений, биты D в регистре Message Valid (MSGVAL12 - MSGVAL78) и биты TxRqst в регистре запроса передачи оцениваются. Действительный объект сообщения с наивысшим приоритетом, ожидающий запроса на передачу, загружается обработчиком сообщений в сдвиговый регистр и начинается передача. Бит NewDat объекта сообщения сбрасывается.

После успешной передачи и если с момента начала передачи в объект сообщения не было записано новых данных (NewDat = '0'), бит TxRqst будет сброшен. Если установлен TxIE, то после успешной передачи будет установлен IntPnd. Если модуль DCAN потерял арбитраж или во время передачи произошла ошибка, сообщение будет передано повторно, как только шина CAN снова освободится. Если тем временем идет передача сообщения с более высоким приоритетом, сообщения будут переданы в порядке их приоритета.

Если режим автоматической ретрансляции отключен установкой бита DAR в регистре CTL, то поведение битов TxRqst и битов TxRqst и NewDat в регистре управления сообщениями набора регистров интерфейса будет следующим:

- Когда начинается передача, бит TxRqst соответствующего набора регистров интерфейса сбрасывается, а бит NewDat остается установленным.

- После успешного завершения передачи бит NewDat сбрасывается.

При неудачной передаче (потеря арбитража или ошибка) бит NewDat остается установленным. Чтобы перезапустить передачу, приложение должно снова установить TxRqst.

Полученные удаленные кадры не требуют объекта приема. Они автоматически вызывают передачу кадра данных, если в соответствующем объекте передачи установлен бит RmtEn.

**23.3.15.4 Обновление объекта передачи**

Центральный процессор может обновлять байты данных объекта передачи в любое время через интерфейсные регистры IF1 и IF2, при этом ни D, ни TxRqst не должны быть сброшены перед обновлением.

Даже если обновлению подлежит только часть байтов данных, все четыре байта в соответствующем регистре IF1 или IF2 Data A (IF1DATA или IF2DATA) или регистре IF1 или IF2 Data B (IF1DATB или IF2DATB) должны быть действительными прежде чем содержимое этого регистра будет передано в объект сообщения. Либо процессор должен записать все четыре байта в регистр данных IF1/IF2, либо объект сообщения передается в регистр данных IF1/IF2 до того, как процессор записывает новые байты данных.

Когда обновляются только байты данных, первые 0x87 могут быть записаны в биты [23:16] регистров команд IF1 и IF2 (IF1CMD и IF2CMD), а затем номер объекта сообщения записывается в биты [7:0] регистра команд, одновременно обновляя байты данных и устанавливая TxRqst с NewDat.

Чтобы предотвратить сброс TxRqst в конце передачи, которая может быть уже в процессе, пока данные обновляются, NewDat должен быть установлен вместе с TxRqst при передаче данных по CAN, управляемой событиями. Подробнее,см. раздел 23.3.15.3.

Если NewDat установлен вместе с TxRqst, NewDat будет сброшен, как только начнется новая передача.

**23.3.15.5 Изменение объекта передачи**

Если количество реализованных объектов сообщений недостаточно для использования их только в качестве постоянных объектов сообщений то объекты передачи могут управляться динамически. Центральный процессор может записать все сообщение (арбитраж, управление и данные) в регистр интерфейса. Биты [23:16] регистра команд могут быть установлены в 0xB7 для передачи всего содержимого объекта сообщения в объект сообщения. Ни D, ни TxRqst не должны быть сброшены перед этой операцией.

Если ранее запрошенная передача данного объекта сообщения не завершена, но уже идет, она будет продолжена; однако она не будет повторена, если она была нарушена.

Чтобы обновить только байты данных передаваемого сообщения, биты [23:16] регистра команд должны быть установлены на 0x87.

*ПРИМЕЧАНИЕ: После обновления объекта передачи набор регистров интерфейса будет содержать копию фактического содержимого объекта, включая ту часть, которая не была обновлена.*

**23.3.15.6 Фильтрация принятых сообщений**

Когда арбитражные и управляющие биты (идентификатор + IDE + RTR + DLC) входящего сообщения полностью сдвинуты в сдвиговый регистр CAN-ядра, обработчик сообщений начинает сканирование RAM сообщений в поисках подходящего объекта сообщения:

- Блок фильтрации приема загружается арбитражными битами из сдвигового регистра ядра CAN.

- Затем биты арбитража и маски (включая MsgVal, UMask, NewDat и EoB) объекта сообщения 1 загружаются в блок фильтрации приема и сравниваются с арбитражными битами из сдвигового регистра. Эта процедура повторяется для всех последующих объектов сообщений до тех пор, пока не будет найден подходящий объект сообщения, или пока не будет достигнут конец оперативной памяти сообщений.

- Если совпадение произошло, сканирование останавливается, и обработчик сообщения продолжает работу в зависимости от типа полученного кадра (кадр данных или удаленный кадр).

**23.3.15.7 Прием кадров данных**

Обработчик сообщений сохраняет сообщение из сдвигового регистра CAN-ядра в соответствующий объект сообщения в соответствующий объект сообщения в оперативной памяти. Не только байты данных, но и все арбитражные биты и код длины данных сохраняются в соответствующем объекте сообщения. Это гарантирует, что байты данных останутся связанными с идентификатором, даже если используются регистры маски арбитража.

Бит NewDat устанавливается, чтобы указать, что были получены новые данные (еще не видимые процессором). Центральный процессор должен сбросить бит NewDat, когда считывает объект сообщения. Если в момент приема бит NewDat уже был установлен, то устанавливается MsgLst, указывая на то, что предыдущие данные (предположительно не замеченные процессором) потеряны. Если бит RxIE установлен, устанавливается бит IntPnd, заставляя регистр прерываний (INT) указывать на этот объект сообщения.

Бит TxRqst этого объекта сообщения сбрасывается, чтобы предотвратить передачу удаленного кадра, в то время как запрошенный кадр данных только что был получен.

**23.3.15.8 Прием удаленных кадров**

При приеме удаленного кадра необходимо учитывать три различные конфигурации объекта сообщения:

- Dir = '1' (направление = передача), RmtEn = '1', UMask = '1' или '0'.

Бит TxRqst этого объекта сообщения устанавливается при приеме совпадающего удаленного кадра. Остальная часть объекта сообщения остается неизменным.

- Dir = '1' (направление = передача), RmtEn = '0', UMask = '0'

Удаленный кадр игнорируется, данный объект сообщения остается без изменений.

- Dir = '1' (направление = передача), RmtEn = '0', UMask = '1'

Удаленный кадр обрабатывается так же, как и принятый кадр данных. При получении соответствующего удаленного кадра Remote Message Frame, бит TxRqst этого объекта сообщения сбрасывается. Биты арбитража и управления (Identifier + IDE + RTR + DLC) из сдвигового регистра сохраняются в объекте сообщения в оперативной памяти сообщения RAM, а бит NewDat этого объекта сообщения устанавливается. Байты данных объекта сообщения остаются неизменными.

**23.3.15.9 Чтение принятых сообщений**

Центральный процессор может в любой момент прочитать полученное сообщение через регистр интерфейса IFx. Согласованность данных гарантируется машиной состояния обработчика сообщений. Обычно процессор записывает сначала 0x7F в биты [23:16] , а затем номер объекта сообщения в биты [7:0] регистра команд. Эта комбинация передаст все полученное сообщение из оперативной памяти сообщений в набор регистров интерфейса. Кроме того, биты NewDat и IntPnd очищаются в ОЗУ сообщений (но не в наборе регистров интерфейса). Значения этих битов в регистре управления сообщениями всегда отражают состояние до сброса битов. Если объект сообщения использует маски для фильтрации приема, биты арбитража показывают, какое из различных соответствующих сообщений было получено.

Фактическое значение NewDat показывает, было ли получено новое сообщение с момента последнего чтения данного объекта сообщения. Фактическое значение MsgLst показывает, было ли получено более одного сообщения с момента последнего считывания этого объекта сообщения.MsgLst не сбрасывается автоматически.

**23.3.15.10 Запрос новых данных для объекта приема**

С помощью удаленного кадра CPU может запросить у другого узла CAN новые данные для объекта приема. Установка бита TxRqst объекта приема вызовет передачу удаленного кадра с идентификатором объекта приема. Этот удаленный кадр заставляет другой узел CAN начать передачу соответствующего кадра данных. Если совпадающий кадр данных получен до того, как удаленный кадр успел быть передан то бит TxRqst автоматически сбрасывается. Установка бита TxRqst без изменения содержимого объекта сообщения требует значения 0x84 в битах [23:16] регистра IFxCMD.

**23.3.15.11 Хранение принятых сообщений в буферах FIFO**

Несколько объектов сообщений могут быть сгруппированы для формирования одного или нескольких буферов FIFO. Каждый буфер FIFO, сконфигурированный для хранения принятых сообщений с определенным (группой) идентификатором (идентификаторами). Регистры арбитража и маски объектов сообщений буфера FIFO идентичны. Биты конца буфера (EoB) всех объектов сообщений буфера FIFO, кроме последнего равны '0'; в последнем объекте бит EoB равен '1'.

Принятые сообщения с идентификаторами, соответствующими буферу FIFO, сохраняются в объекте сообщения этого FIFO-буфера, начиная с объекта сообщения с наименьшим номером сообщения. Когда сообщение сохраняется в объекте сообщения буфера FIFO, устанавливается бит NewDat этого объекта сообщения. При установке NewDat, когда EoB равен '0', объект сообщения блокируется для дальнейшего доступа обработчика cообщения на запись до тех пор, пока центральный процессор не снимет бит NewDat.

Сообщения хранятся в буфере FIFO до тех пор, пока не будет достигнут последний объект сообщения в этом буфере FIFO. Если ни один из предыдущих объектов сообщений не освобожден путем записи NewDat в '0', все последующие сообщения для этого FIFO-буфера будут записаны в последний объект сообщений FIFO-буфера (EoB = '1') и, следовательно перезапишет предыдущие сообщения в этом объекте сообщений.

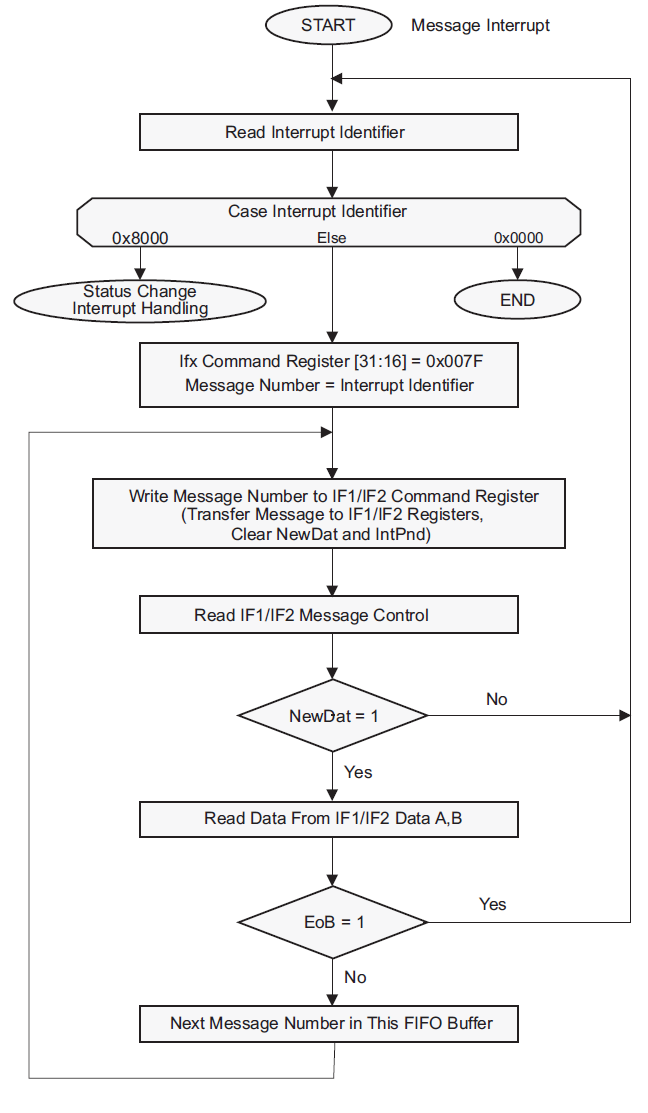
**23.3.15.12 Чтение из буфера FIFO**

Несколько сообщений могут быть накоплены в наборе объектов сообщений, которые объединяются для формирования FIFO-буфера, прежде чем прикладная программа должна (во избежание потери данных) опустошить буфер.Буфер FIFO длины N будет хранить -1 плюс последнее полученное сообщение с момента его последней очистки.

Буфер FIFO очищается путем чтения и сброса битов NewDat всех его объектов сообщений, начиная с объекта FIFO с наименьшим номером сообщения. Это должно быть сделано в подпрограмме, как показано на рисунке 23-12.

*ПРИМЕЧАНИЕ: Все объекты сообщений буфера FIFO должны быть считаны и очищены, прежде чем будет сохранена следующая партия сообщений. В противном случае истинная функциональность FIFO не может быть гарантирована, поскольку объекты сообщений частично прочитанного буфера будут заполнены заново в соответствии с обычным (по убыванию) приоритетом. Чтение из объекта сообщения буфера FIFO и сброс его бита NewDat выполняется так же, как и чтение из отдельного объекта сообщения.*

**Рисунок 23-12. Обработка буфера FIFO процессором (управляемый прерыванием)**



**23.3.16 Временные характеристики битов CAN**

DCAN поддерживает скорость передачи данных от менее 1 кБит/с до 1000 кБит/с.

Каждый узел сети CAN имеет собственный тактовый генератор, обычно получаемый от кристаллического осциллятора. Параметры синхронизации битов могут быть настроены индивидуально для каждого узла CAN, создавая общую скорость передачи данных даже если периоды генераторов (fosc) узлов CAN могут быть разными.

Частоты этих осцилляторов не являются абсолютно стабильными. Небольшие колебания вызваны изменениями температуры или напряжения, а также из-за износа компонентов. До тех пор, пока эти отклонения остаются в пределах определенного диапазона допуска осциллятора (df), узлы CAN способны компенсировать разную скорость передачи данных путем повторной синхронизации с потоком битов.

Во многих случаях синхронизация битов CAN исправляет ошибочную конфигурацию синхронизации битов CAN до такой степени, что лишь изредка генерируется кадр ошибки. Однако в случае арбитража, когда два или более узлов CAN одновременно пытаются передать кадр, неправильно выбранная точка выборки может привести к тому, что один из передатчиков может стать пассивным к ошибкам.

Анализ таких спорадических ошибок требует детального знания битовой синхронизации CAN внутри узла CAN и взаимодействия узлов CAN на шине CAN.

Даже если незначительные ошибки в конфигурации синхронизации битов CAN не приводят к немедленному отказу, производительность сети CAN может быть значительно снижена.

**23.3.16.1 Битовое время и скорость передачи данных**

Согласно спецификации CAN, битовое время делится на четыре сегмента (см. Рисунок 23-13):

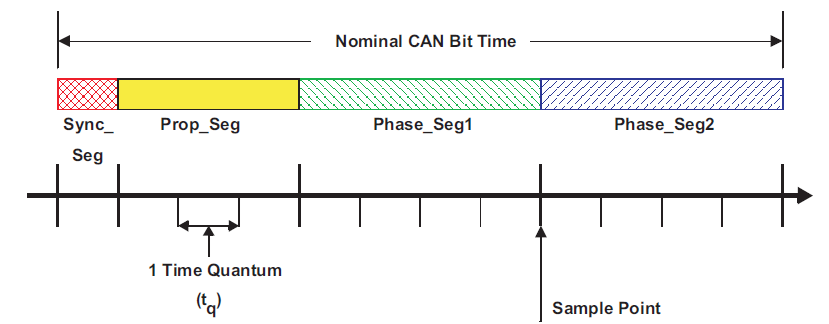
- Сегмент синхронизации (Sync\_Seg)

- Сегмент времени распространения (Prop\_Seg)

- Сегмент фазового буфера 1 (Phase\_Seg1)

- Сегмент фазового буфера 2 (Phase\_Seg2)

**Рисунок 23-13. Временные характеристики битов**



Каждый сегмент состоит из определенного количества квантов времени. Длительность одного кванта времени (tq), который является базовой единицей битового времени, задается CAN\_CLK и прескалерами скорости передачи данных (BRPE и BRP). С помощью этих двух прескалеров скорости передачи данных можно запрограммировать значения делителя от 1 до 1024:

tq = Baud Rate Prescaler / CAN\_CLK

Кроме фиксированной длины сегмента синхронизации, эти числа являются программируемыми. Таблица 23-7 описывает минимальные программируемые диапазоны, требуемые протоколом CAN. Заданная скорость передачи данных может быть обеспечена различными конфигурациями битового времени.

**Таблица 23-7. Параметры битового времени CAN**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Range** | **Remark** |
| Sync\_Seg | 1 tq (fixed) | Синхронизация входа шины с CAN\_CLK |
| Prop\_Seg | [1 … 8] tq | Компенсирует время физической задержки |
| Phase\_Seg1 | [1 … 8] tq | Может быть временно удлинена за счет синхронизации |
| Phase\_Seg2 | [1 … 8] tq | Может быть временно сокращена за счет синхронизации |
| Synchronization Jump Width  (SJW) | [1 … 4] tq | Не может быть длиннее, чем один из сегментов фазового буфера |

*ПРИМЕЧАНИЕ: Для правильной работы сети CAN необходимо учитывать физическое время задержки и диапазон допуска осциллятора.*

**23.3.16.1.1 Сегмент синхронизации**

Сегмент синхронизации (Sync\_Seg) - это часть битового времени, в которой ожидается появление фронтов уровня шины CAN. Если фронт возникает за пределами Sync\_Seg, его расстояние до Sync\_Seg называется фазовой ошибкой этого фронта.

**23.3.16.1.2 Сегмент времени распространения**

Эта часть битового времени используется для компенсации времени физической задержки в сети CAN. Эти задержки состоят из времени распространения сигнала по шине и времени внутренней задержки узлов CAN.

Любой узел CAN, синхронизированный с битовым потоком на шине CAN, может находиться вне фазы с передатчиком битового потока, что вызвано временем распространения сигнала между двумя узлами. Протокол CAN предоставляет приемникам CAN неразрушающий побитовый арбитраж и доминирующий бит подтверждения и требует, чтобы CAN-узел, передающий битовый поток, также мог принимать доминирующие биты передаваемых другими узлами CAN, которые синхронизированы с этим битовым потоком. Пример на рисунке 23-14 показывает фазовый сдвиг и время распространения между двумя узлами CAN.

**Рисунок 23-14. Сегмент времени распространения**

