**8.1.1 Введение**

    Архитектура управления питанием устройства обеспечивает максимальную производительность и время работы для удовлетворения пользователя(поддержка аудио/видео), предлагая универсальные методы управления питанием для максимальной гибкости приложения в зависимости от требований. Это введение содержит следующую информацию:

- стандартные блоки архитектуры управления питанием для устройства.

- современные методы управления питанием, поддерживаемые архитектурой управления питанием устройства.

**8.1.2 Строительные блоки архитектуры управления питанием устройств**

      Чтобы обеспечить универсальную архитектуру, поддерживающую несколько методов управления питанием, фреймворк построен с тремя уровнями управления ресурсами: тактированием, питанием и управлением напряжением.

Эти уровни управления реализуются путем определения управляемых объектов или стандартных блоков архитектуры управления питанием

, называемую доменами тактирования, питания и напряжения. Домен — это группа модулей или подразделов устройства, которые имеют общий объект (например, общий источник синхронизации, общий источник напряжения или общий выключатель питания). Группа, образующая домен, управляется менеджером политик. Например, тактирование для домена тактирования управляется выделенным менеджером тактирования в пределах домена управления питанием, сбросом и тактированием в модуле (**PRCM**). Менеджер тактирования считает совместную синхронизацию ограничения всех модулей, принадлежащих этому домену тактирования (и, следовательно, получающих эти такты).

**8.1.3 Управление тактированием**

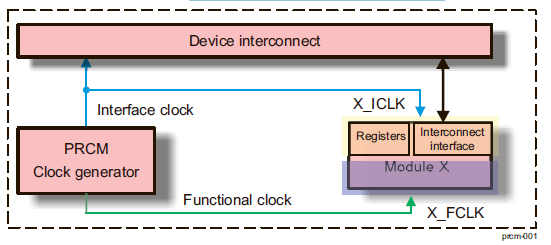
Модуль **PRCM** вместе с модулем управления управляет стробированием (т.е. выключением) и включением тактирования , поступающего к модулям устройства. Тактирование управляется на основе ограничений к связанным модулям. Следующие разделы определяют характеристики тактирования модуля, политику управления,

тактовые домены и управление тактовыми доменами.

**8.1.3.1 Интерфейс модуля и функциональное тактирование**

Каждый модуль внутри устройства имеет определенные требования к входным тактовым сигналам. На основе характеристики тактирования, поставляемого в модули, тактирование делится на две категории: **интерфейсное тактирование** и **функциональное тактирование**.

**Рис.8-1.Функциональное и интефейсное тактирование**

****

Интерфейсное тактирование имеет следующие характеристики:

- Оно обеспечивает правильную связь между любым модулем/подсистемой и межсоединением.

- В большинстве случаев оно обеспечивает интерфейс межсоединения системы и регистров модуля.

- Типичный модуль имеет один тактовый сигнал интерфейса, но также могут существовать модули с несколькими тактовыми генераторами интерфейса.

(то есть при подключении к нескольким соединительным шинам).

- Управление тактированием интерфейса осуществляется на уровне устройства.

- С точки зрения модуля **PRCM**, интерфейсное тактирование  идентифицируются суффиксом **\_ICLK**.

Функциональное тактирование имеет следующие характеристики:

- Оно обеспечивает функциональную часть модуля или подсистемы.

- Модуль может иметь один или несколько источников функционального тактирования. Некоторые источники являются обязательными, в то время как другие

являются необязательными. Модуль нуждается в своем обязательном тактировании, чтобы работать. Дополнительное тактиторвание используются для специфических функции и может быть отключено без остановки работы модуля.

- С точки зрения модуля **PRCM**, функциональное тактирование распределяется непосредственно на соответствующий модуль через специальное дерево тактирования. Оно определяется суффиксом **\_FCLK**.

**8.1.3.2 Управление тактированием на уровне модуля**

Каждый модуль в устройстве также может иметь определенные требования к тактовой частоте. Тактирование некоторых модулей должно быть активно при работе в определенных режимах или должно быть отключено в противном случае. Глобально активация и отключение тактирования модуля управляются модулем **PRCM**. Следовательно, модуль **PRCM** должен знать, когда активировать и когда стробировать тактирование модуля. Модуль **PRCM** различает управление тактированием поведения модулей устройств в зависимости от того, может ли модуль инициировать транзакции на устройстве межсоединений (называемых главным модулем или инициаторами) или не может инициировать транзакции и отвечает только на транзакции, инициированные мастером (называемым подчиненным модулем или целями). Таким образом, используется два аппаратных протокола управления питанием :

- Основной резервный протокол: протокол управления тактированием между **PRCM** и ведущими модулями.

- Протокол ожидания подчиненного устройства: протокол управления тактированием между **PRCM** и подчиненными модулями.

**8.1.3.2.1 Основной резервный протокол**

Основной резервный протокол используется для указания того, что главный модуль должен инициировать транзакцию на устройстве межсоединения и запросить определенное (функциональное и интерфейсное) тактирование для этой цели. Модуль **PRCM** гарантирует, что требуемое тактирование активно, когда главный модуль запрашивает модуль **PRCM** включить его. Это называется переходом модуля в режим пробуждения, и можно сказать, что по окончании этого  перехода модуль работает.

Точно так же, когда мастер-модуль больше не нуждается в тактировании, он информирует об этом модуль **PRCM**, который затем может прервать тактирование модуля. В этом случае принято считать, что главный модуль находится в режиме ожидания. Хотя протокол полностью аппаратно контролируется, программное обеспечение должно конфигурировать поведение управления тактированием для модуля. Это делается установкой битового поля регистра модуля <Модуль>**\_SYSCONFIG.MIDLEMODE** или <Модуль>**\_SYSCONFIG.STANDBYMODE**. Поведение, определяемое значениями режима ожидания, должно быть настроено.

**Таблица 8-1. Настройки режима ожидания главного модуля**

|  |  |  |
| --- | --- | --- |
| Standby Mode Value | Selected Mode | Description |
| 0x0 | Force-standby | Модуль безоговорочно утверждает  резервный запрос к модулю PRCM,  независимо от его внутренних операций.  Модуль PRCM может блокировать функциональное и интефейсное тактирование к модулю. Этот  режим следует использовать осторожно, поскольку он не предотвращает потерю данных в то время как тактирование отключено. |
| 0x1 | No-standby | Модуль никогда не устанавливает режим ожидания запроса к модулю PRCM. Этот режим безопасен с модульной точки зрения потому что это гарантирует, что тактирование остается активным. Однако он не эффективен с точки зрения энергосбережения, потому что он  никогда не разрешит стробирование выходного тактирования модуля PRCM. |
| 0x2 | Smart-standby | Модуль утверждает резервный запрос  исходя из состояния его внутренней активности. Cигнал ожидания подается только тогда, когда все текущие транзакции завершены и модуль простаивает. Модуль PRCM затем может стробированть тактирование модуля. |
| 0x3 | Smart-standbywakeup-capable mode | Модуль утверждает резервный запрос  исходя из состояния его внутренней активности. Сигнал ожидания подается только тогда, когда всетекущие транзакции завершены и модуль простаивает. Модуль PRCM может  затем стробировать тактирование модуля. Модуль может генерировать (связанные с мастером) события пробуждения в момент нахождения в состоянии STANDBY. Режим актуален только в том случае, если выход соответствующего модуля mwakeup реализован. |

Состояние ожидания ведущего модуля обозначается С**M\_**<Power\_domain>\_<Module>**\_CLKCTRL**[**x**]. Бит **STBYST** в модуле **PRCM**.

**Таблица 8-2. Состояние ожидания главного модуля**

|  |  |
| --- | --- |
| STBYST Bit Value | Description |
| 0x0 | **Модуль функционален.** |
| 0x1 | **Модуль в режиме ожидания.** |

**8.1.3.2.2 Протокол ожидания подчиненного устройства**

Этот аппаратный протокол позволяет модулю **PRCM** управлять состоянием ведомого модуля. **PRCM** модуль информирует ведомый модуль посредством подтверждения запроса на бездействие, когда его тактирование (интерфейсное и функциональное) может быть закрытым. Затем ведомое устройство может подтвердить запрос от модуля **PRCM** и затем модулю **PRCM** разрешает передавать тактирование модулю. Говорят, что ведомый модуль находится в состоянии **IDLE** ,когда его тактирование стробируются модулем **PRCM**. Аналогичным образом, может потребоваться разбудить неиспользуемый подчиненный модуль из-за запроса на обслуживание от главного модуля или в результате события (так называемого пробуждения; к примеру , прерывание или запрос DMA), полученного ведомым модулем. В этой ситуации **PRCM**

модуль включает тактирование для модуля, а затем отменяет запрос на бездействие, чтобы сигнализировать модулю о пробуждении. Хотя протокол полностью аппаратно контролируется, программное обеспечение должно настроить поведение управления тактированием ведомого модуля. Это делается установкой битового поля регистра модуля <Модуль>**\_SYSCONFIG. SIDLEMODE** или <Module>**\_SYSCONFIG. IDLEMODE**. Поведение, указанное в столбеце **Idle Mode Value** должно быть настроено програмно.

**Таблица 8-3. Настройки режима ожидания модуля**

|  |  |  |
| --- | --- | --- |
| Idle Mode Value | Selected Mode | Description |
| 0x0 | Force-idle | Модуль безоговорочно признает  незанятый запрос от модуля PRCM,  независимо от его внутренних операций. Этот режим следует использовать осторожно, поскольку он  не предотвращает потерю данных во время отключения тактирования. |
| 0x1 | No-idle | Модуль никогда не признает запроса на бездействие от модуля PRCM. Этот  режим безопасен с модульной точки зрения, потому что это гарантирует, что тактирование остется активным. Однако он не эффективен с перспективы энергосбережения, потому что это  не позволяет выходному тактированию модуля PRCM быть отключенным, и, таким образом, установить домен питания в состояние пониженного энергопотребления. |
| 0x2 | Smart-idle | Модуль подтверждает запрос на холостой ход, основанный на его внутренней активности. А именно, сигнал подтверждения утверждается только тогда, когда все ожидающие  транзакции, прерывания или запросы на прямой доступ к памяти (DMA) обрабатываются.  Это лучший подход к эффективному  управление питанием системы. |
| 0x3 | Smart-idle wakeup-capable mode | Модуль подтверждает запрос на холостой ход, основанный на его внутренней  активности в режиме пробуждения. А именно, сигнал подтверждения подается толькокогда все незавершенные транзакции, прерывания,  или запросы DMA обрабатываются. Это  лучший подход к эффективной системе  управление энергопотреблением. Модуль может генерировать (связанные с запросом IRQ или DMA) события пробуждения в состоянии IDLE. Режим актуален только в том случае, если соответствующие выход(ы) пробуждения модуля реализованы. |

Статус бездействия ведомого модуля указывается в битовом поле **IDLEST** CM\_<Powerdomain>\_<Module>**\_CLKCTRL** в модуле **PRCM**.

**Таблица 8-4. Состояния простоя для ведомого модуля**

|  |  |  |
| --- | --- | --- |
| IDLEST Bit VALUE | Idle Status | Description |
| 0x0 | Functional | Модуль полностью функционален.  интерфейсное и функциональное тактирование активны. |
| 0x1 | In transition | Модуль выполняет пробуждение или  Переход в сон. |
| 0x2 | Interface idle | Интерфейсное тактирование модуля находится в режиме ожидания(простоя).  Модуль может оставаться работоспособным при использовании  отдельного функционального тактирования. |
| 0x3 | Full idle | Модуль полностью бездействует. Интерфейсное и функциональное тактирование включены. |

Для управления незанятым протоколом на стороне модуля **PRCM** поведение модуля **PRCM** настраивается в битовом поле **CM\_<Power domain>\_<module>\_CLKCTRL[x]** **MODULEMODE**. На основе настроенного поведения, модуль **PRCM** безоговорочно подтверждает запрос на бездействие к модулю (то есть, сразу же, когда программа запросит).

**Таблица 8-5. Настройки режима ведомого модуля в PRCM**

|  |  |  |
| --- | --- | --- |
| MODULEMODE Bit VALUE | Selected Mode | Description |
| 0x0 | Disabled | Модуль PRCM безоговорочно утверждает  запрос на бездействие модуля. Этот запрос относится к стробированию функционального и интерфейсного тактирования модуля. Если  подтверждается модулем, PRCM  модуль может отключить все тактирование на модуль  (то есть модуль полностью  деактивирован).. |
| 0x1 | Reserved | NA |
| 0x2 | Enabled | Этот режим применяется к модулю, когда  Модуль PRCM управляет своими интерфейсным и функциональным тактированием. Функциональное тактирование для модуля остается активным безоговорочно,  в то время как модуль PRCM автоматически подтверждает/отменяет запрос на бездействие модуля на основе переходов в домена-тактирования. Если  подтверждено модулем, PRCM  модуль может стробировать только интерфейсное тактирование модуля. |
| 0x3 | Reserved | NA |

В дополнение к протоколам **IDLE** и **STANDBY**, **PRCM** также предлагает возможность управления необязательным тактирование, через прямое управление ПО: бит «**OptFclken**» из регистра программирования.

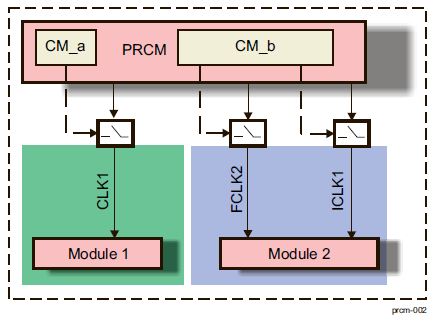
**Таблица 8-6. Условие включения тактирования модуля**

|  |  |  |  |
| --- | --- | --- | --- |
| Clock Enabling | | | |
| Тактирование, связанное с  протоколом STANDBY | AND | Домен тактирования готов | |
| OR | MStandby is de-asserted |
| Mwakeup is asserted |
| Тактирование, связанное с протоколом IDLE, как интерфейсное тактирование. | AND | Домен тактирования готов | |
| OR | Idle status = FUNCT |
| Idle status = TRANS |
| SWakeup is asserted |
| Тактирование, связанное с протоколом IDLE, как функциональное тактирование. | AND | Домен тактирования готов | |
| OR | Idle status = FUNCT |
| Idle status = IDLE |
| Idle status = TRANS |
| SWakeup is asserted |
| Опциональное тактирование | AND | Домен тактирования готов | |
| OptFclken=Enabled ('1') | |

**8.1.3.3 Домен тактирования**

Тактовый домен представляет собой группу модулей, питаемых тактовым сигналом, управляемым одним и тем же диспетчером тактовых импульсов модуля **PRCM** , путем стробирования тактирования в домене тактирования для всех модулей, принадлежащих этому домену. Тактирование можно отключить, чтобы снизить их активное энергопотребление (то есть устройство включено, а тактирование модулей динамически переключается в ACTIVE или INACTIVE(GATED) состояния). Таким образом, домен тактирования позволяет контролировать динамическое энергопотребление устройства. Устройство разделено на несколько  доменов тактирования, и каждый домен контролируется соответствующим диспетчером тактирования в  модуле PRCM . Это позволяет модулю **PRCM** индивидуально активировать и блокировать каждый домен тактирования устройства.

**Рисунок 8-2. Общий домен тактирования**

****

На рисунке выше показан пример двух диспетчеров тактирования: CM\_a и CM\_b. Каждый диспетчер тактирования управляет доменом тактирования. Домен тактирования состоит из двух источников синхронизации: функционального тактирования(**FCLK2**) и интерфейсного тактирования (**ICLK1**), в то время как область тактирования **CM\_a** состоит из тактирования(**CLK1**), которое  используются модулем в качестве функционального  и интерфейсного тактирования. Тактирование к модулю 2 может быть подключены независимо от тактирования  модуля 1, тем самым обеспечивая экономию энергии, когда модуль 2 не используется. Модуль **PRCM** позволяет программному обеспечению проверять состояние функционального тактирования домена тактирования. **CM\_**<Часовой домен>**\_CLKSTCTRL**[**x**] Бит тактирования<**FCLK**/**Clock name\_FCLK**> в модуле **PRCM** идентифицирует состояние функционального тактирования в домене тактирования. В таблице показаны возможные состояния функционального тактирования:

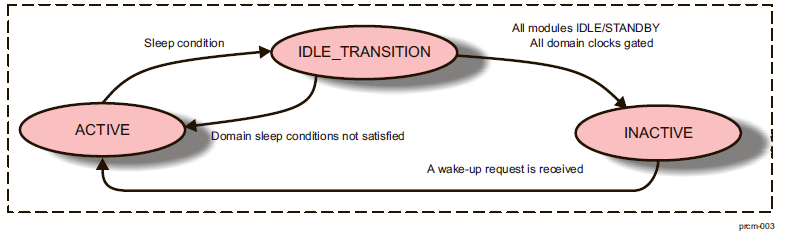
**Таблица 8-7. Функциональные состояния тактирования домена тактирования**

|  |  |  |
| --- | --- | --- |
| CLKACTIVITY BIT Value | Status | Description |
| 0x0 | Gated | Функциональное тактирование домена-тактирования неактивно. |
| 0x1 | Active | Функциональное тактирование домена-тактирования запущено. |

**8.1.3.3.1 Управление тактированием на уровне домена**

Диспетчер тактирования  домена может автоматически (т. е. в зависимости от состояния оборудования) и совместно управлять тактированием интерфейса находящегося в домене тактирования. Функциональное тактирование в домене тактирования  управляется через настройки ПО. Домен тактирования  может переключаться между тремя возможными состояниями: **ACTIVE**, **IDLE\_TRANSITION** и **INACTIVE**. На рис. 8-3 показаны переходы тактирования  в режим сна и пробуждения в домене тактирования при переходе между состояниями **ACTIVE** и **INACTIVE**.

**Рисунок 8-3. Возможные состояния домена тактирования**

****

**Таблица 8-8. Состояния домена тактирования**

|  |  |
| --- | --- |
| State | Description |
| ACTIVE | Каждый неотключенный ведомый модуль (т.е. значение MODULEMODE не отключено) выводится из режима IDLE.  Предоставляетия все интерфейсное тактирование к неотключенным ведомым модулям в домене тактирования предоставляются. В домен предоставляются все функциональное и интерфейсное тактирование к  активным ведущим модулям (то есть тем кто не в режиме ожидания). Предоставляется возможность включения всего опционального тактирования в домене тактирования. |
| IDLE\_TRANSITION | Это переходное состояние.  Каждый главный модуль в домене тактирования находится в состоянии STANDBY.  Утверждается каждый бездействующий запрос ко всем ведомым модулям в домене тактирования. Функциональное тактирование подчиненного модуля включено(то есть те, у которых значения MODULEMODE установлены на вкл.) остаются активными.  Предоставляется возможность включения всего опционального тактирования для модулей в домене тактирования. |
| INACTIVE | Все тактирование в домене тактирования закрыто.  Каждый ведомый тактируемый модуль находится в состоянии IDLE и деактивирован.  Каждый ведомый модуль в домене тактирования (то есть те, чьи  MODULEMODE установлены на disabled) находится в состоянии IDLE и деактивированы.  Возможность управления опциональным тактированием стробирована(отключена). |

Каждое поведение перехода домена тактирования управляется соответствующим битовым полем в регистре **CM\_**<**Clockdomain**>**\_CLKSTCTRL**[x] **CLKTRCTRL** модуля **PRCM**

**Таблица 8-9. Настройки режима перехода тактирования**

|  |  |  |
| --- | --- | --- |
| CLKTRCTRL Bit Value | Selected Mode | Description |
| 0x0 | NO\_SLEEP | Переход в спящий режим не может быть инициирован.Однако может произойти переход к пробуждению. |
| 0x1 | SW\_SLEEP | Программно-принудительный переход в сон. Переход инициируется, когда связанные аппаратные условия удовлетворены. |
| 0x2 | SW\_WKUP | Программно-принудительное пробуждение домена тактирования,  переход инициирован. |
| 0x3 | Reserved | NA |

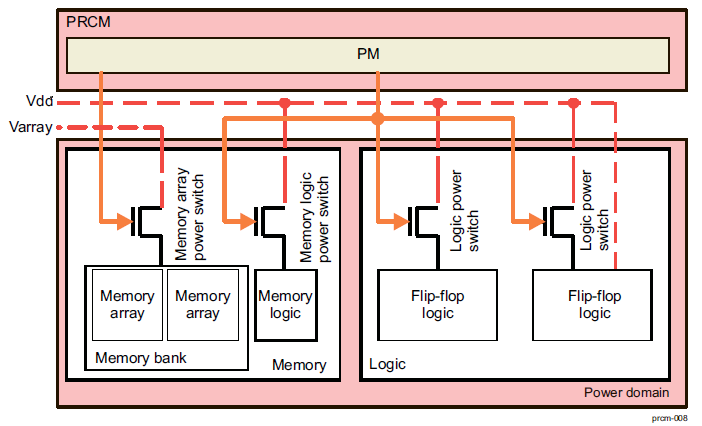
**8.1.4 Управление питанием**

Модуль **PRCM** управляет включением и выключением питания модулей устройств. Для минимизации энергопотребление устройства, питание модулей может быть отключено, когда они не используются . Независимое управление питанием секций устройства позволяет модулю **PRCM** включать и выключать определенные разделы устройства, не затрагивая другие.

**8.1.4.1 Домен питания**

Домен питания — это часть (то есть группа модулей) устройства с независимым и выделенным диспетчером питания (см. рисунок). Домен питания можно включать и выключать, не затрагивая другие части устройства.

**Рисунок 8-4. Общая архитектура Power Domain**

****

Для минимизации энергопотребления устройства модули сгруппированы в домены питания. Домен питания можно разделить на логическую область и область памяти.

**Таблица 8-10. Состояния области памяти в домене питания.**

|  |  |
| --- | --- |
| State | Description |
| ON | Массив памяти включен и полностью функционален |
| OFF | Массив памяти выключен |

**Таблица 8-11. Состояния логической области в домене питания.**

|  |  |
| --- | --- |
| State | Description |
| ON | Логика работает на полную катушку. |
| OFF | Логические выключатели питания выключены. Вся логика (DFF) потеряна. |

**8.1.4.2 Управление доменом питания**

Менеджеру питания, связанному с каждым доменом питания, назначается задача управления силовыми ключами домена. Он гарантирует, что все аппаратные условия выполнены, прежде чем он сможет инициировать переход  домена питания из исходного в целевое состояние питания.

**Таблица 8-12. Регистры контроля и состояния Power Domain**

|  |  |  |
| --- | --- | --- |
| Register/Bit Field | Type | Description |
| PM\_<Power domain>\_PWRSTCTRL[1:0] POWERSTATE | Control | Выбирает целевое состояние питания  область питания среди значений OFF, ON или  RETENTION(УДЕРЖАНИЕ). |
| PM\_<Power domain>\_PWRSTST[1:0]  POWERSTATEST | Status | Определяет текущее состояние домена питания. Он может быть OFF, ON или RETENTION(УДЕРЖАНИЕ). |
| PM\_<Power domain>\_PWRSTST[2]  LOGICSTATEST | Status | Идентифицирует текущее состояние логики  домена питания. Это может быть OFF  или ON. |
| PM\_<Power domain>\_PWRSTST[5:4]  MEMSTATEST | Status | Определяет текущее состояние памяти в домене питания. Он может OFF, ON или RETENTION(УДЕРЖАНИЕ). |

**8.1.4.2.1 Методы управления питанием**

В следующем разделе описываются современные методы управления питанием, поддерживаемые устройством.

**8.1.4.2.1.1 Адаптивное масштабирование напряжения**

**AVS** — это метод управления питанием, основанный на **Smart Reflex**, который используется для автоматического управления рабочими напряжениями устройства для снижения активной потребляемой мощности. Со **Smart Reflex**, напряжение блока питания адаптируется к производительности кремния, либо статически (на основе точек производительности, предварительно определенных в производственном процессе данного устройства) или динамически (на основе работоспособность устройства). Сравнение этих предустановленных показателей производительности с производительностью в реальном времени на кристалле

определяет, следует ли повышать или понижать напряжение питания. **AVS** достигает оптимальное соотношение производительности и мощности для всех устройств по всему спектру технологических процессов и изменение температуры. Напряжение устройства автоматически адаптируется для поддержания производительности устройства.

**8.1.4.3 Режимы питания**

Ниже приводится общее описание различных режимов питания устройства. Они перечислены в порядке от наибольшего энергопотребления, наименьшей задержки пробуждения (в режиме ожидания) до наименьшего энергопотребления, самая высокая задержка пробуждения (только для **RTC**). Если ваше приложение требует какого-либо управления питанием, вам необходимо определить, какой уровень мощности, описанный ниже, удовлетворяет вашим требованиям. Каждый уровень должен быть оценен на основе потребляемой мощности и задержки (время, необходимое для пробуждения в активном режиме). Специфический значения подробно описаны в спецификации конкретного устройства. Обратите внимание, что не все режимы поддерживаются программным обеспечением пакетов поставляемых Texas Instruments.

**Таблица 8-13. Типичные режимы питания**

|  |  |  |
| --- | --- | --- |
| Power Modes | Application State | Power Domains, Clocks, and Voltage  Supply States |
| Active | Все функции | **Источники питания**:  Все источники питания в состоянии ON.  VDD\_MPU = 1.1 V (nom)  VDD\_CORE = 1.1 V (nom)  **Такирование**:  Main Oscillator (OSC0) = ON  Все DPLL в запломбированы  **Домены питания**:  PD\_PER = ON  PD\_MPU = ON  PD\_GFX = ON or OFF (зависит от случая использования)  PD\_WKUP = ON  DDR активно. |
| Standby | Память DDR находится в режиме самообновления и ее содержимое сохраняется. Пробуждение от любого  GPIO. Содержимое контекста/регистров CortexA8 теряются и должны быть сохранены до входа в режим ожидания(standby). При выходе контекст должен быть восстановлен из DDR. Для пробуждения, выполняется boot ROM и система продолжает работу. | **Источники питания**:  Все источники питания в состоянии ON.  VDD\_MPU = 0.95 V (nom)  VDD\_CORE = 0.95 V (nom)  **Такирование**:  Main Oscillator (OSC0) = ON  Все DPLL в режиме обхода  **Домены питания**:  PD\_PER = ON  PD\_MPU = OFF  PD\_GFX = OFF  PD\_WKUP = ON  DDR в самообновлении. |
| Deepsleep1 | Встроенные периферийные регистры  сохранены. Контекст/регистры Cortex-A8  теряются, поэтому приложение нужно сохранить их в ОЗУ L3 OCMC или DDR  перед входом в DeepSleep. DDR находится в самообновлении. Для пробуждения, выполняется boot ROM и система возобновляет работу. | **Источники питания**:  Все источники питания в состоянии ON.  VDD\_MPU = 0.95 V (nom)  VDD\_CORE = 0.95 V (nom)  **Такирование**:  Main Oscillator (OSC0) = OFF  Все DPLL в режиме обхода  **Домены питания**:  PD\_PER = ON  PD\_MPU = OFF  PD\_GFX = OFF  PD\_WKUP = ON  DDR в самообновлении. |
| Deepsleep0 | Информация в периферийном регистре PD\_PER и Cortex-A8/MPU будет потеряна. Информация встроенного периферийного регистра (контекст)  домена PD-PER должна быть сохранена  приложением в SDRAM, прежде чем войти в этот режим. DDR находится в состоянии самообновления. Для пробуждения, выполняется boot ROM и система возобновляет работу. | **Источники питания**:  Все источники питания в состоянии ON.  VDD\_MPU = 0.95 V (nom)  VDD\_CORE = 0.95 V (nom)  **Такирование**:  Main Oscillator (OSC0) = OFF  Все DPLL в режиме обхода  **Домены питания**:  PD\_PER = OFF  PD\_MPU = OFF  PD\_GFX = OFF  PD\_WKUP = ON  DDR в самообновлении. |
| RTC-Only | Таймер RTC остается активным, а вся остальная функциональность устройства отключена. | **Источники питания**:  Все источники питания выключены, кроме VDDS\_RTC.  VDD\_MPU = 0 V  VDD\_CORE = 0 V  **Такирование**:  Main Oscillator (OSC0) = OFF  **Домены питания**:  Все домены питания в состоянии OFF. |

**8.1.4.3.1 Активный(Active)**

В активном режиме должно поддерживаться питание всех шин напряжения. Все домены питания переходят в состояние ON и устройство полностью функционально.

**8.1.4.3.2 Режим ожидания(Standby)**

Устройство можно перевести в режим ожидания, чтобы снизить энергопотребление при низком уровне активности. Этот первый уровень управления питанием позволяет поддерживать контекст устройства для быстрого возобновления работы. Главный характеристики этого режима, которые отличают его от активного режима:

-Все модули синхронизированы, кроме GPIO.

- PLL могут быть переведены в режим обхода, если синхронизация в нисходящем направлении не требует полной производительности.

- Области напряжения уровни напряжения **VDD\_MPU** и **VDD\_CORE** могут быть снижены до уровней **OPP50**, поскольку снижается требуемая производительность всего устройства.

- Домен мощности **MPU** (**PD\_MPU**) находится в состоянии **OFF**.

- Память **DDR** находится в режиме самообновления с низким энергопотреблением.

В этом режиме можно добиться дальнейшего снижения мощности, если функция **RTC** не требуется. Смотрите Раздел 8.1.4.3.6, Внутренний LDO RTC.

Вышеупомянутые условия приводят к более низкому энергопотреблению, чем в активном режиме, но требуют от пользователя сохранения контекста **MPU** в **OCMC RAM** или **DDR** для правильного возобновления после пробуждения. Содержимое внутренней **SRAM** потеряно, потому что **PD\_MPU** выключен. Пробуждение в режиме ожидания достигается с помощью любого **GPIO**. Пробуждение  **GPIO** возможно путем переключения ножки в режим **GPIO** и настройки соответствующего банка **GPIO** для

генерации прерывания для **MPUSS**. Обратите внимание, что ножки контроллера без режима мультиплексирования **GPIO** (например, **ADC** или **USB**), не могут вызвать эти пробуждения. Если требуются дополнительные или другие источники пробуждения, синхронизация связанного периферийного модуля и домен синхронизации межсоединений должны оставаться включенными (это может потребовать, чтобы соответствующий **PLL** оставался заблокированным), и модуль должен быть настроен соответствующим образом для пробуждения , если его настроить для генерации прерывания на **MPUSS**.

**8.1.4.3.3 Глубокий сон1(Deepsleep1)**

Режим **DeepSleep1** обеспечивает более низкое энергопотребление, чем режим ожидания. Основная характеристика этого режима, отличающая его от режима ожидания, заключается в том, что основной генератор (**OSC0**) отключен. **DeepSleep1** — это самый низкий спящий режим, необходимый для определенных режимов пробуждения USB. См. Раздел 8.1.4.3.7, Поддерживаемые сценарии пробуждения **USB** с низким энергопотреблением, для получения дополнительной информации.

В этом режиме можно добиться дальнейшего снижения мощности, если функция RTC не требуется. Смотрите раздел 8.1.4.3.6, Внутренний **LDO RTC**.

Как и в режиме ожидания, содержимое внутренней **SRAM** теряется, поскольку **PD\_MPU** выключен. Кроме того, содержимое **SDRAM** сохраняется за счет перевода **SDRAM** в режим самообновления. Активность на пробуждение периферийных устройств с помощью событий пробуждения позволяет задающему кварцевому генератору использовать схему управления генератором. События пробуждения также прерывают **Cortex-M3**. См. раздел 8.1.4.5 «Источники/события пробуждения».

подробности об источниках пробуждения.

**8.1.4.3.4 Глубокий сон0(Deepsleep0)**

Режим **DeepSleep0** обеспечивает более низкое энергопотребление, чем **DeepSleep1**. Основные характеристики этого режима, которые отличают его от других режимов более высокой мощности:

- Все внутренние домены питания отключены (кроме **PD\_WKUP** и **PD\_RTC  -** остаются включенными) для снижения утечки энергии.

- Питание **VDD\_CORE** (кроме аналога **VDDA**) для **DPLL** отключается с помощью регистра dpll\_pwr\_sw\_ctrl (только PG2.x)

- **VDDS\_SRAM\_CORE\_BG** хранится с использованием **SMA2**.**vsldo\_core\_auto\_ramp\_en** (только PG2.x).

Режим **DeepSleep0** обычно используется в периоды бездействия, когда пользователю требуется очень мало энергии при ожидании события, требующего обработки или более высокой производительности. Это режим минимальной мощности который по-прежнему включает **DDR** в самообновление, поэтому события пробуждения не требуют полной холодной перезагрузки, что значительно уменьшает задержки пробуждения по сравнению с режимом только **RTC**. В этом режиме можно добиться дальнейшего снижения мощности, если функция **RTC** не требуется. Смотрите Раздел 8.1.4.3.6, Внутренний **LDO RTC**.

Как и в режиме **DeepSleep1**, содержимое внутренней **SRAM** теряется, поскольку **PD\_MPU** выключен. Перед входом в режим **DeepSleep0** контекст периферии и **MPU** должен быть сохранен в **DDR**. Во время пробуждения, boot ROM выполняется и проверяет, не вышло ли устройство из состояния **DeepSleep0**. Если так, то boot ROM

перенаправляет работу в **DDR** для продолжения процесса возврата в работу. Поскольку питание PD\_WKUP постоянно включено во время **DeepSleep0**, подача питания на ключевые модули, такие как **GPIO0**, **I2C** и другие, поддерживается, чтобы разрешить события пробуждения для выхода из этого режима. Кроме того, поддерживается питание оперативной памяти **OCMC** для внутреннего хранения информации во время **DeepSleep0**.

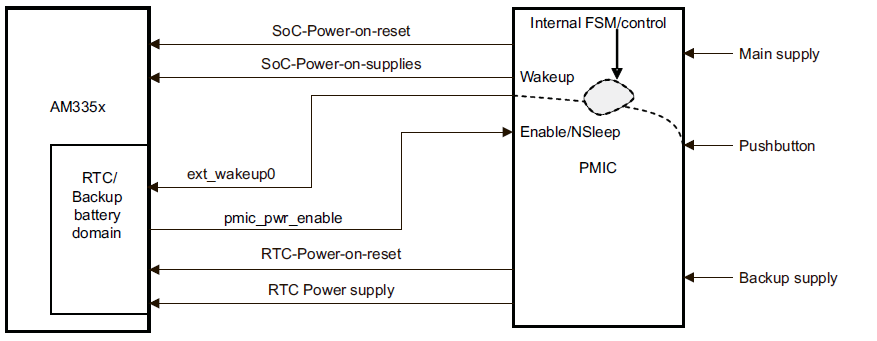
Активность периферийных устройств пробуждения через события пробуждения позволяет задающему кварцевому генератору использовать цепь управления генераторм. События пробуждения также прерывают **Cortex-M3**, который контролирует правильное включение доменов питания и тактирования в **PRCM**. См. раздел 8.1.4.5 «Источники/события пробуждения» для получения подробной информации о источниках во время **DeepSleep0** и других упомянутых режимах с низким энергопотреблением.

**8.1.4.3.5 Только RTC(RTC-Only)**

Режим «только **RTC**» — это режим сверхнизкого энергопотребления, который позволяет пользователю поддерживать питание и синхронизацию с доменом часов реального времени (**RTC**), в то время как остальная часть устройства выключена. Весь контекст и память будут быть потеряны, и единственная часть чипа, которая будет поддерживаться, — это **RTC**. Только источник питания RTC должен быть включен. Все остальные источники питания должны быть ВЫКЛЮЧЕНЫ. Домен резервного аккумулятора RTC состоит из подсистемы **RTC** (**RTCSS**), специальный встроенный кварцевый генератор с частотой **32,768** Гц и входы/выходы, связанные с

**RTCSS**: **pmic\_power\_en** и **ext\_wakeup**. На рис. 8-5 представлен общий вид системы, реализующей режим только **RTC**.

**Рисунок 8-5. Системный вид высокого уровня для режима только RTC**

****

Пробуждение из режима только RTC может быть достигнуто только с помощью сигнала **ext\_wakeup0** или **RTC Alarm**(ТРЕВОГА). Как только пробуждение запускается с использованием любого из этих источников, устройство запускает **pmic\_pwr\_enable** чтобы инициировать последовательность включения питания с помощью **PMIC**. Устройство должно пройти полную холодную перезагрузку после пробуждения из режима только **RTC**.

**8.1.4.3.6 Внутренний LDO RTC**

Устройство содержит внутренний регулятор **LDO** (с малым падением напряжения), который питает цифровое ядро ​​**RTC**. В зависимости от вашего приложения вы можете отключить этот регулятор для экономии энергии при использовании в случаях с низким энергопотреблением . Если ваше приложение никогда не использует функции **RTC**, подключите выводы **RTC\_KALDO\_ENn** к **VDDS\_RTC**, **CAP\_VDD\_RTC** на **VDD\_CORE** и **RTC\_PWRONRSTn** на землю. Эти соединения отключают внутренний **LDO RTC**, потому что когда на **RTC\_KALDO\_ENn** высокий уровень, и они начинают использовать внешний источник **VDD\_CORE** для питание цифрового ядра **RTC**. **RTC LDO** должен быть отключен для внутренней последовательности питания, даже если **RTC**не используется. Заземление сигнала сброса гарантирует, что RTC останется в состоянии сброса. Отключение внутреннего **LDO** позволит приложению снизить энергопотребление во всех режимах с низким энергопотреблением.

Если ваше приложение использует функции **RTC** и никогда не нуждается в режиме только-**RTC**, аппаратный сценарий схож с тем что описан ранее, но к устройству нужно дополнительно подключить сигнал сброса **RTC PWRONRSTn**. Обратите внимание, что **PWRONRSTn** и **RTC\_PWRONRSTn** могут иметь разные уровни напряжения, поэтому **PWRONRSTn** может потребовать сдвига уровня перед подключением к **RTC\_PWRONRSTn**. Это соединение делает возможной полную функциональность подсистемы **RTC** без потребления энергии внутренним **RTC LDO**.

Если ваше приложение использует функциональные возможности **RTC** и требует режима только **RTC**, внутренний **LDO** должен включить правильную сигнализацию пробуждения из домена **RTC**. Правильная сигнализация пробуждения требует следующих соединений:

- RTC\_KALDO\_ENn заземлен

- CAP\_VDD\_RTC подключен к развязывающему конденсатору 1 мкФ на землю.

- RTC\_PWRONRSTn подключен к источнику питания RTC 1,8 В при сбросе.

- PMIC\_POWER\_EN подключен к входу питания PMIC

- EXT\_WAKEUP0 подключен к источнику пробуждения

См. datashet устройства для получения дополнительной информации об этих сигналах.

**8.1.4.3.7 Поддерживаемые сценарии пробуждения USB с низким энергопотреблением**

В Таблице 8-14 приведены различные варианты пробуждения через **USB**, которые поддерживаются в каждом состоянии сна системы.

(**DeepSleep0**, **DeepSleep1** или **Режим ожидания**). Существует три сценария использования:

- Подключение USB: Пробуждение происходит при физическом подключении кабеля USB.

- Отключение USB: Пробуждение вызывается физическим отсоединением USB-кабеля.

- Приостановка/возобновление USB: Пробуждение вызывается командой приостановки или возобновления USB. Например, щелчок мышью USB может вызвать команду возобновления USB.

В каждом варианте использования пробуждения каждая строка описывает, поддерживается ли этот тип пробуждения в каждом спящем режиме системы. Также учитывается режим USB (хост или устройство).

Генерируются два возможных события пробуждения:

- **PHY WKUP**: это внутренний сигнал пробуждения для Cortex M3, который генерируется USB PHY на основе USB-сигнализации.

- **VBUS2GPIO**: это внешний сигнал пробуждения, поступающий из-за изменения уровня напряжения **VBUS**. Для этого события требуется внешняя плата, которая подводить **VBUS** к **GPIO** на устройстве. Убедитесь, что вы сдвинули уровень напряжения для соответствия требованиям ввода/вывода. Когда VBUS переходит с 0 В на 5 В (или наоборот), переход на **GPIO** вызовет пробуждение.

**Таблица 8-14. Случаи пробуждения от USB, поддерживаемые в спящих состояниях системы**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | USB Wakeup  Use Case | System Sleep  State | USB Controller  State | USB Mode | Supported | USB Wakeup  Event |
| 1 | USB Connect | DS0 | POWER OFF | Host | No | N/A |
| 2 | DS0 | POWER OFF | Device | Yes | VBUS2GPIO |
| 3 | DS1/ Standby | Clock Gated | Host | Yes | PHY WKUP |
| 4 | DS1/ Standby | Clock Gated | Device | Yes | VBUS2GPIO |
| 5 | USB Suspend /  Resume | DS0 | POWER OFF | Host | No | N/A |
| 6 | DS0 | POWER OFF | Device | No | N/A |
| 7 | DS1/ Standby | Clock Gated | Host | Yes | PHY WKUP |
| 8 | DS1/ Standby | Clock Gated | Device | Yes | PHY WKUP |
| 9 | USB Disconnect | DS0 | POWER OFF | Host | No | N/A |
| 10 | DS0 | POWER OFF | Device | No | N/A |
| 11 | DS1/ Standby | Clock Gated | Host | Yes | PHY WKUP |
| 12 | DS1/ Standby | Clock Gated | Device | Yes | VBUS2GPIO |

**8.1.4.4 Управление главным генератором во время глубокого сна**

Схема генератора **Deepsleep** используется для управления основным генератором, отключая его во время глубокого сна и включая во время активности/пробуждения. По умолчанию во время сброса осциллятор включен, а схема управления осциллятором - отключена (неактивна). Чтобы активировать схему управления генератором для глубокого сна, должен быть установлен бит **DSENABLE** в регистре **DEEPSLEEP\_CTRL**. Как только он установлен и всякий раз, когда включается пробуждение **M3**

в режиме ожидания, управление генератором отключит генератор, что приведет к отключению тактирования. Любое асинхронное событие от источников пробуждения приведет к тому, что управление генератором повторно включит генератор после периода **DSCOUNT** , который настраивается в регистре **DEEPSLEEP\_CTRL**.

**8.1.4.5 Источники/события пробуждения**

Следующие события выведут устройство из режимов глубокого сна (с низким энергопотреблением). Это часть домена **Wakeup Power** и всегда остается включенным.

**Примечание**. Различия в работе в зависимости от версии микросхемы **AM335x** см. в Разделе 1.2, Версия микросхемы. Функциональные отличия и улучшения.

- Банк GPIO0

- dmtimer1\_1ms (пробуждение по таймеру)

- USB2PHY (сигнализация возобновления USB из режима ожидания) — поддерживаются оба порта USB.

- TSC (контроллер сенсорного экрана, функции монитора ADC)

- UART0 (поддержка инфракрасного порта)

• I2C0

• Тревога часов реального времени

Эти события пробуждения применимы к любому из режимов глубокого сна и режима ожидания.

**8.1.4.6 Функциональная последовательность управления питанием с помощью Cortex M3**

Устройство AM335x содержит специальный процессор Cortex M3 для управления переключением питания. Он является частью домена **Wake up Power** (**PD\_WKUP**). Реализация режимов питания является частью

процессоров MPU и Cortex A8. Последовательность управления питанием начинается с того, что MPU Cortex A8 выполняет инструкцию **WFI** в следующем порядке:

1. В режиме активного питания микропроцессор Cortex A8 выполняет инструкцию **WFI** для перехода в режим **IDLE**.

2. Cortex M3 получает прерывания и становится активным. Он отключает питание **MPU** (при необходимости).

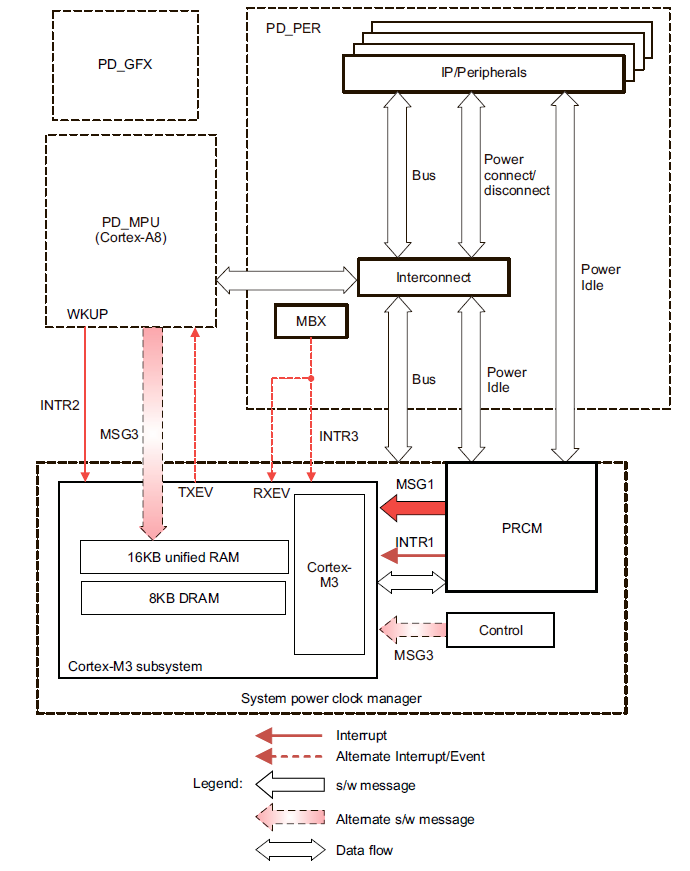
3. Регистрирует прерывание для периферийного устройства пробуждения (которое указано в источниках пробуждения в предыдущем разделе).

4. Выполняет WFI и переходит в состояние ожидания.

5. Событие пробуждения вызывает прерывание системы Cortex M3 и пробуждает микропроцессор Cortex A8.

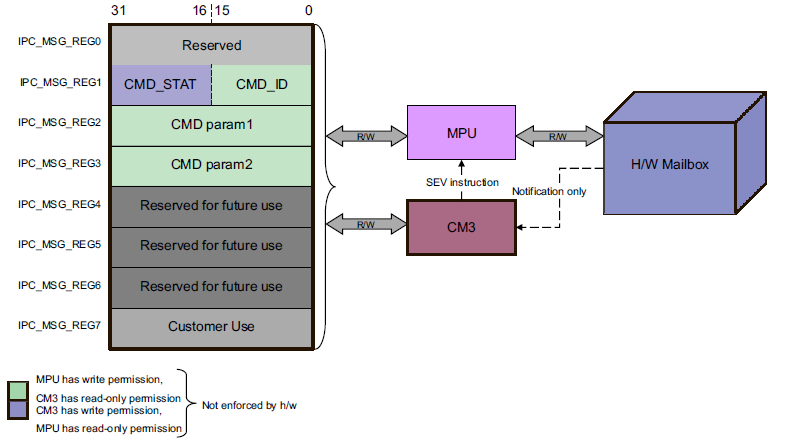
Как правило, не ожидается, что A8 и Cortex M3 будут активны одновременно, Cortex M3вместе с **PRCM** являетеся менеджером питания в первую очередь для **PD\_MPU** и **PD\_PER**. Другие домены мощности (например, **PD\_GFX**) могут обрабатываться напрямую с помощью программного обеспечения Cortex A8 MPU. На рис. 8-6 показано представление на системном уровне системы управления питанием между Cortex A8 MPU и Cortex M3.

**Рисунок 8-6. Представление на системном уровне управления питанием Cortex A8 MPU и Cortex M3**

****

Cortex-M3 обрабатывает все управление питания низкого уровня для AM335x. Бинарный файл прошивки предоставляется компанией Texas Instruments и включает в себя все необходимые функции для работы в режимах с низким энергопотреблением. Регистры межпроцессорной связи (**IPC**) (ipc\_msg\_regx, расположены в регистрах модуля управления) доступны для связи с **Cortex-M3**, поэтому пользователь может указать определенные параметры конфигурации в зависимости от требуемого уровня малой мощности. На рис. 8-7 показано отображение этих регистров.

**Рисунок 8-7. Механизм IPC**

****

IPC\_MSG\_REG1 содержит параметры CMD\_STAT и CMD\_ID, как описано в таблице 8-15 и таблице 8-16.

**Таблица 8-15. Поле CMD\_STAT**

|  |  |  |
| --- | --- | --- |
| CMD\_STAT | Value | Description |
| PASS | 0x0 | На этапе инициализации PASS (0x1) означает, что CM3 был успешно  инициализирован. |
| FAIL | 0x1 | На этапе инициализации 0x2 означает, что CM3 не может правильно инициализироваться. Когда необходимо выполнить другие задачи, FAIL (0x3) указывает на некоторую ошибку при выполнении задачм.  Подробности смотрите в векторе трассировки. |
| WAIT4OK | 0x2 | CM3 INTC перехватит следующий WFI A8 и продолжит работу с предварительно определенной последовательностью. |

**Таблица 8-16. Поле CMD\_ID**

|  |  |  |
| --- | --- | --- |
| CMD\_ID | Value | Description |
| CMD\_RTC | 0x1 | 1. Инициирует force\_sleep на тактах межсоединений.  2. Отключает домены мощности MPU и PER.  3. Программирует регистр аварийных сигналов RTC для отключения pmic\_pwr\_enable. |
| CMD\_RTC\_FAST | 0x2 | Программирует регистр аварийных сигналов RTC для отключения pmic\_pwr\_enable. |
| CMD\_DS0 | 0x3 | 1. Инициирует force\_sleep на тактах межсоединений.  2. Отключает домены питания MPU и PER.  3. Настраивает систему для отключения MOSC, когда CM3 выполняет WFI. |
| CMD\_DS1 | 0x5 | 1. Инициирует force\_sleep на тактах межсоединений.  2. Отключает домены питания MPU.  3. Настраивает систему для отключения MOSC, когда CM3 выполняет WFI. |

**8.1.4.6.1 Периодический холостой ход микропроцессора Cortex A8**

Для реализации периодического ВКЛ/ВЫКЛ MPU Cortex A8 в варианте использования можно реализовать поток управления в соответствии со следующими шагами:

1. MPU Cortex A8 выполняет инструкцию **WFI**

2. Любое периферийное прерывание на любом из следующих шагов вызовет прерывание пробуждения Cortex M3 через сигнал **WKUP** подсистемы **MPU**  (на схеме показан INTR2). Cortex M3 отключает питание **MPU**(**PD\_MPU**)

3. При получении прерывания Cortex M3 включает область питания **MPU**, включая **PD\_MPU**.

4. Cortex M3 переходит в режим ожидания с помощью инструкции **WFI**.

**8.1.4.6.2 Последовательность сна**

В этом разделе приведены рекомендации системного уровня для определения последовательности сна. Рекомендации могут служить примером для реализации последовательности спящего режима. Пользователь может реализовать последовательность с определенных шагов  между **MPU** и процессором **Cortex M3**.

1. Приложение сохраняет контекст периферийных устройств в память, поддерживающую сохранение этого контекста и в **DDR** — этот шаг только требуется для **Deepsleep0**.

2. **MPU OCMC\_RAM** остается в резерве

3. Неиспользуемые домены питания отключаются - запрограммируйте домены тактирования/питания в **PWRSTCTRL**, сохраните контексты так же как и ранее.

4. Программное обеспечение заполняет **L3\_OCMC\_RAM** для восстановления пробуждения, а именно: сохраняет настройки **EMIF**, общедоступные/защищенные указатели восстановления и т. д.

5. Выполнить **WFI** из **SRAM**

6. Любое периферийное прерывание вызовет прерывание пробуждения Cortex M3 через сигнал WKUP Cortex A8 MPU. (**INTR2** показан на схеме).

7. После того, как область питания **MPU** будет синхронизирована, **PRCM** предоставит прерывание Cortex M3 (использование **INTR1** показано на блок-схеме)

8. Cortex M3 начинает выполнение и выполняет последовательность команд снижения уровня мощности, чтобы отключить определенный домен питания, и в конечном итоге выполняет **WFI**.

9. Схема управления аппаратным генератором отключает генератор, как только Cortex M3 переходит в WFI.

**8.1.4.6.3 Последовательность пробуждения**

В этом разделе приведены рекомендации по последовательности пробуждения.

1. Один из триггеров события пробуждения (который был настроен во время последовательности сна) инициирует последовательность пробуждения

2. Событие пробуждения включит осциллятор (если он был настроен на выключение во время сна).

3. Событие пробуждения также вызовет прерывание Cortex M3.

4. В случае пробуждения из-за прерывания Cortex M3 выполните следующие действия.

- Восстановите нормальное напряжение. Рабочее напряжение.

- Включить блокировку PLL

- Cortex M3 включит домены питания и/или включит тактирование для PD\_PER.

- Cortex M3 включит домены питания и/или включит тактирование для PD\_MPU.

- Выполняет **WFI**

5. Cortex A8 MPU начинает выполнение с вектора сброса ROM.

6. Восстанавливается контекст приложения (только для **Deep sleep 0**)

**8.1.5 Обзор модуля PRCM**

PRCM структурирован с использованием архитектурных концепций, представленных во фреймворке серии 5000x Power Management. Этот фрэймворк обеспечивает:

Из набора модульных, повторно используемых блоков конечных автоматов(FSM), которые собираются в законченную систему управления тактовой частотой и механизмом питания. Из набора регистров и связаннай с ними модель программирования. Из определения функциональных подблоков для управления тактированием, управления питанием, генерации источника системных тактовых импульсов и генерация основных тактовых импульсов.

Устройство поддерживает расширенную схему управления питанием, основанную на четырех функциональных доменах питания:

**Общие домены**

- WAKEUP

- MPU

- PER

- RTC

PRCM обеспечивает следующие функциональные возможности:

- Программное обеспечение, настраиваемое для прямого, автоматического (или их комбинации) управления состоянием перехода функционального питания.

- Контроль последовательности включения устройства

- Управление последовательностью сна/пробуждения устройства

- Централизованное создание и управление сбросом

- Централизованная генерация и управление тактированием

Модули PRCM реализуют следующие общие функциональные интерфейсы:

- Порты конфигурации OCP

- Прямой интерфейс к границе устройства

- Сигналы управления выключателем питания

- Сигналы управления устройством

- Сигналы управления тактированием

- Сигналы сброса

- Набор сигналов протокола управления питанием для каждого модуля для управления и мониторинга режима ожидания, простоя и режимы пробуждения (**CM** и **PRM**)

- Сигналы эмуляции

**8.1.5.1 Описание интерфейса**

В этом разделе перечислены и кратко описаны различные интерфейсы, которые позволяют PRCM взаимодействовать с другими модулями или внешними устройствами:

**8.1.5.1.1 Интерфейсы OCP**

PRCM имеет 1 целевой интерфейс OCP, соответствующий стандарту OCP/IP2. Порт ОСР, для модуля PRCM используется для управления питанием, сбросом и пробуждением.

**8.1.5.1.2 Ведомые интерфейсы OCP**

PRCM реализует 32-битный целевой интерфейс OCP, совместимый со стандартом OCP/IP2.0.

**8.1.5.1.3 Интерфейс управления питанием**

Устройство имеет переключатели домена питания над устройством, этот интерфейс обеспечивает управление PRCM над коммутаторами домена питания и получает ответы от доменов питания, которые указывают состояние коммутатора. Он также управляет сигналами изоляции. Управление коммутаторами домена питания будет зафиксировано в регистрах состояния PRCM.

**8.1.5.1.4 Интерфейс управления устройством**

Этот интерфейс обеспечивает управление **PRM** несколькими функциями на уровне устройства, которые не являются специфическими для какого-либо

единого энергетического домена. Этот интерфейс PRM управляет сигналами к/от устройства для глобального управления:

- Кодировка типа устройства

- Контроль изоляции IOs

**8.1.5.1.5 Интерфейс тактирования**

Этот интерфейс собирает все тактовые входы и выходы, управляемые модулями PRCM.

**8.1.5.1.6 Интерфейс сброса**

Этот интерфейс собирает все входы и выходы сброса, управляемые модулем PRCM.

**8.1.5.1.7 Интерфейс управления питанием модулей**

Модули или подсистемы в устройстве делятся на 2 категории:

- Инициатор: инициатор — это модуль, способный генерировать трафик на соединениях устройств (обычно: процессоры, MMU, EDMA).

- Цель: цель — это модуль, который не может генерировать трафик на соединениях устройств, но может генерировать прерывания или запросы DMA к системе (обычно: периферийные устройства). **PRCM** обрабатывает протокол рукопожатия  при управлении питанием с каждым модулем или подсистемой. Этот протокол позволяет выполнять правильное тактирование и переключение питания с учетом активности или состояния каждого модуля.

**8.1.5.1.8 Интерфейс инициирующих модулей**

Модуль PRCM обрабатывает все интерфейсы управления питанием модулей-инициаторов: MStandby сигнал и MWait сигнал.

**8.1.5.1.9 Интерфейс целевых модулей**

Модуль PRCM обрабатывает все интерфейсы управления питанием целевых модулей: сигнал SIdleReq сигнал SIdleAck сигнал FCLKEN

Примечание. Поддержка USB для SWakeUp

**8.1.6 Генерация тактовых импульсов и управление ими**

PRCM обеспечивает централизованное управление генерацией, распределением и стробированием большинства тактовых импульсов в устройстве. PRCM стробирует внешние тактовые импульсы и внутренние тактовые импульсы для распределения по другим модулям в устройстве. PRCM управляет генерацией системного тактирования.

**8.1.6.1 Терминология**

PRCM производит 2 типа тактов:

Интерфейсное тактирование: эти такты  первую очередь обеспечивают синхронизацию модулей межсоединений системы и

части функциональных модулей устройства, которые взаимодействуют с модулями межсоединений системы. В большинстве случаем интерфейсные такты обеспечивают интерфейс между межсоединениями системы функционального модуля и регистрами. В некоторых модулях интерфейсное тактирование также используются как функциональные тактирование. В этом документе интерфейсное тактирование представлено синими линиями.

Функциональное тактирование: это тактирование обеспечивают функциональную часть модуля или подсистемы. В некоторых случаях модулю или подсистеме может потребоваться несколько функциональных тактов: 1 или несколько основных функциональных тактов, 1 или несколько дополнительных тактов. Модуль нуждается в своих основных тактах, чтобы работать. Дополнительное тактирование используются для специфические функции и могут быть отключены без остановки модуля.

**8.1.6.2 Структура тактирования**

Для генерации высокочастотных тактов устройство поддерживает несколько встроенных **DPLL**, управляемых непосредственно модулем **PRCM**. В это документе они относятся к двум типам PLL, называемым ADPLLS и ADPLLLJ.

Модуль **ADPLLS** используется для ядра, дисплея, подсистемы **ARM** и **PLL DDR**.

Модуль **ADPLLLJ** используется для функционального тактирования периферии.

Устройство имеет два опорных тактовых сигнала, которые генерируются встроенными генераторами или внешними генераторами. Это для основного тактирования и блока RTC соответственно. В случае внешнего генератора  источник тактов может быть напрямую подключен к выводу **XTALIN**, и генератор будет переведен в режим байпасс. 32-кГц кварцевый генератор управляется и настраивается с помощью **RTC IP**. Это устройство также содержит встроенный RC-генератор. Этот осциллятор не настраивается и всегда включен.

Основной генератор на устройстве (см. Главу 26, Инициализация, для возможных частот) производит главные высокочастотные такты **CLK\_M\_OSC**.

**8.1.6.3 ADPLLS**

**ADPLLS** представляет собой синтезатор частоты с высоким разрешением и встроенными регуляторами уровня, что позволяет генерации захваченных частот **PLL** до 2 ГГц. **ADPLLS** имеет функцию предварительного деления частоты, которая позволяет пользователю разделить, например, опорную тактовую частоту 24 или 26 МГц на 1 МГц, а затем умножить максимум до 2 ГГц. Все **PLL** переходят в режим байпаса при сбросе. Программное обеспечение должно правильно запрограммировать все настройки **PLL** , а затем дождаться  блокировки **PLL**. Для получения дополнительной информации см. процедуру конфигурации для каждого PLL.

Существуют следующие **PLL**:

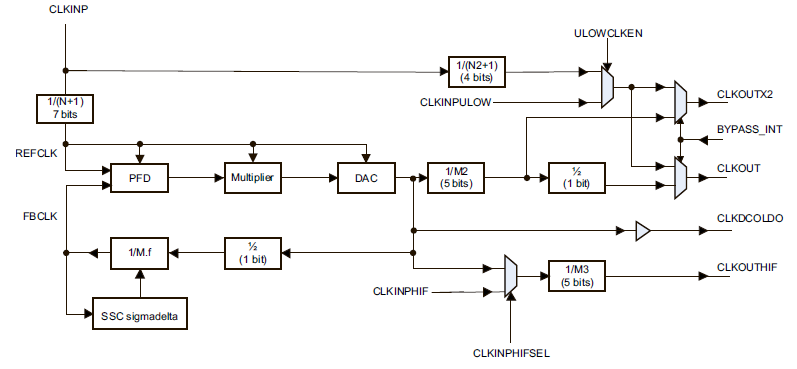
- MPU PLL

- Core PLL

- Display PLL

- DDR PLL

**Рисунок 8-8. ADPLLS**

****

ADPLLS имеет три входных тактовых сигнала:

- CLKINP: Опорный входной тактовый сигнал

- CLKINPULOW: Синхронизация низкочастотного входа только для режима байпаса.

- CLKINPHIF: Высокочастотный входной тактовый сигнал для постделителя M3.

ADPLLS имеет четыре выходных тактовых сигнала:

- CLKOUTHIF: Высокочастотный выходной тактовый сигнал от постделителя M3.

- CLKOUTX2: Дополнительный выход 2x

- CLKOUT: Синхронизация основного выхода.

- CLKDCOLDO: выходной такт генератора (DCO) без обхода.

DPLL имеет два внутренних тактовых сигнала:

- REFCLK (внутренний эталонный такт): генерируется путем деления входного тактового сигнала **CLKINP** на запрограммированное значение N+1. Весь цикл **PLL** работает на **REFCLK**.

Здесь **REFCLK** = **CLKINP**/(N+1).

•-BCLK: тактовый сигнал шины, который используются для программирования различных настроек с помощью регистров.

Частота блокировки **ADPLLS** определяется следующим образом: fDPLL = **CLKDCOLDO**

**8.1.6.3.1 Функции тактирования.**

**Таблица 8-17. Выходные тактовые сигналы в заблокированном состоянии**

|  |  |  |
| --- | --- | --- |
| Pin Name | Frequency | Comments |
| REGM4XEN='0' | | |
| CLKOUT | [M / (N+1)] \* CLKINP \* [1/M2] |  |
| CLKOUTX2 | 2 \* [M / (N+1)] \* CLKINP \* [1/M2] |  |
| CLKDCOLDO | 2 \* [M / (N+1)] \* CLKINP |  |
| CLKOUTHIF | CLKINPHIF / M3 | CLKINPHIFSEL='1' |
| 2 \* [M / (N+1)] \* CLKINP \* [1/M3] | CLKINPHIFSEL='0' |
| REGM4XEN='1' | | |
| CLKOUT | [4M / (N+1)] \* CLKINP \* [1/M2] |  |
| CLKOUTX2 | 2 \* [4M / (N+1)] \* CLKINP \* [1/M2] |  |
| CLKDCOLDO | 2 \* [4M / (N+1)] \* CLKINP |  |
| CLKOUTHIF | CLKINPHIF / M3 | CLKINPHIFSEL='1' |
| 2 \* [4M / (N+1)] \* CLKINP \* [1/M3] | CLKINPHIFSEL='0' |

**Таблица 8-18. Выходные тактовые сигналы перед блокировкой и во время режимов повторной блокировки**

|  |  |  |
| --- | --- | --- |
| Pin Name | Frequency | Comments |
| CLKOUT | CLKINP / (N2+1) | ULOWCLKEN='0' |
| CLKINPULOW | ULOWCLKEN='1' |
| CLKOUTX2 | CLKINP / (N2+1) | ULOWCLKEN='0' |
| CLKINPULOW | ULOWCLKEN='1' |
| CLKDCOLDO | Low |  |
| CLKOUTHIF | CLKINPHIF/M3 | ULOWCLKEN='1' |
| Low | ULOWCLKEN='0' |

Примечание. Поскольку делитель M3 работает во внутреннем домене LDO, в случае, когда **CLKINPHIFSEL**=’1’, **CLKOUTHIF** может быть активен только тогда, когда внутренний **LDO** включен. Следовательно, всякий раз, когда **LDOPWDN** переходит с низкого на высокий уровень для понижения мощности **LDO** (это происходит при активации **TINITZ** / при входе в режим обхода медленной повторной блокировки), вывод **CLKOUTHIF** даст сбой и остановится. Чтобы избежать этого сбоя, рекомендуется гейтировать **CLKOUTHIF** с помощью контроль CLKOUTHIFEN перед утверждением TINITZ / входом в любой режим обхода медленной повторной блокировки Диапазона Частоты (МГц)

См. руководство по данным для конкретного устройства для получения подробной информации о точках рабочих характеристик (OPP), поддерживаемых

вашим устройством.

**8.1.6.4 ADPLLLJ (DPLL с низким джиттером)**

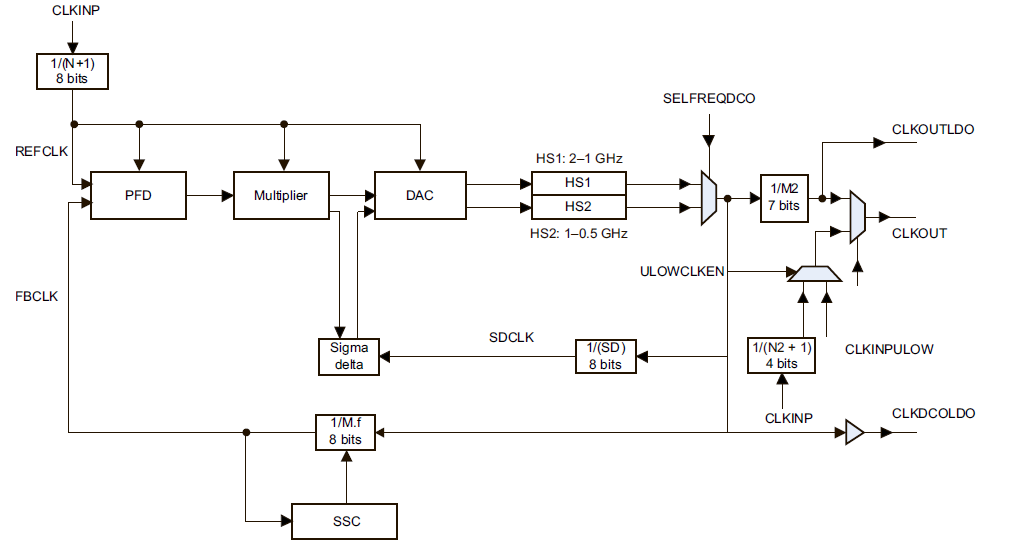
**ADPLLLJ** представляет собой систему ФАПЧ с низким джиттером и максимальной выходной частотой 2 ГГц. ADPLLLJ имеет функцию предварительного деления, которая

позволяет пользователю разделить, например, опорную частоту 24 МГц или 26 МГц на 1 МГц, а затем умножить до максимум 2 ГГц.

Все PLL переходят в режим байпас при сбросе. Программное обеспечение должно правильно запрограммировать все настройки PLL.

а затем дождитесь блокировки PLL. Для получения дополнительной информации см. процедуру конфигурации для каждого PLL.

**Рисунок 8-9. Базовая структура ADPLLLJ**

****

Периферийные PLL относится к типу ADPLLLJ:

DPLL имеет два входных тактовых сигнала:

- CLKINP: Опорный входной тактовый сигнал

- CLKINPULOW: входной тактовый сигнал обхода.

DPLL имеет два внутренних тактовых сигнала:

- **REFCLK** (внутренний эталонный такт): генерируется путем деления входного тактового сигнала CLKINP на запрограммированное значение N+1. Весь цикл PLL работает на **REFCLK**.

Здесь **REFCLK** = **CLKINP**/(N+1).

- CLKDCOLDO (тактовый сигнал внутреннего генератора): это необработанные тактовые сигналы непосредственно из осциллятора (DCO) перед постделителем. Выходной тактовый сигнал **PLL** синтезируется внутренним генератором, который привязан по фазе к refclk. В ADPLLLJ встроены два генератора. Осцилляторы

выбирается пользователем на основе требования к частоте синтезированного выходного тактового сигнала. В заблокированном состоянии, CLKDCOLDO = CLKINP \*[M/(N+1)].

Частота блокировки **ADPLLLJ** определяется следующим образом: fDPLL = CLKDCOLDO DPLL имеет три внешних тактовых сигнала:

- CLKOUTLDO: Первичные выходные тактовые сигналы в домене VDDLDOOUT. Опция обхода недоступна на этом выходе.

CLKOUTLDO = (M / (N+1))\*CLKINP\*(1/M2)

- CLKOUT: основной выходной тактовый сигнал в домене цифрового ядра.

CLKOUT = (M / (N+1))\*CLKINP\*(1/M2)

- CLKDCOLDO: выходной такт генератора (DCO) перед пост-делением в домене VDDLDOOUT. Опция обхода недоступна для этого выхода.

CLKDCOLD = (M / (N+1))\*CLKINP.

Все тактовые выходы DPLL могут быть стробированы.  Модуль управления предоставляет DPLL стробирование тактового сигнала управляющим сигналом для включения или выключения тактовых сигналов, а DPLL обеспечивает модуль **PRCM** тактированием состояния активности, чтобы оборудование модуля **PRCM** знало, когда тактирование работает или эффективно стробировано. Выходные тактовые импульсы управляются для различными источниками тактовых импульсов:

**CLKOUTEN/CLKOUTLDOEN/CLKDCOLDOEN**.

**8.1.6.4.1 Функции тактирования**

**Таблица 8-19. Выходные тактовые импульсы в заблокированном состоянии**

|  |  |
| --- | --- |
| Pin Name | Frequency |
| CLKOUT | [M /(N+1)] \* CLKINP \* [1/M2] |
| CLKOUTLDO | [M /(N+1)] \* CLKINP \* [1/M2] |
| CLKDCOLDO | [M /(N+1)] \* CLKINP |

**Таблица 8-20. Выходные тактовые импульсы перед блокировкой и во время режимов повторной блокировки**

|  |  |  |
| --- | --- | --- |
| Pin Name | Frequency | Comments |
| CLKOUT | CLKINP/(N2+1) | ULOWCLKEN=’0’ |
| CLKINPLOW | ULOWCLKEN=’1’ |
| CLKDCOLDO | LOW |  |
| CLKOUTLDO | LOW |  |

**8.1.6.5 Изменение M2 «на лету»**

Делитель М2 предназначен для смены на лету и обеспечения безотказного переключения частоты со старой на новую. Его можно изменить, когда PLL находится в заблокированном состоянии, без необходимости переключения в режим байпас. Бит переключения состояния покажет, был ли принят новый делитель. Делитель М2 также можно изменить в режиме байпас, и новое значение делителя будет отражено на выходе после запрания PLL . Дополнительные сведения см. в процедурах настройки PLL для каждой PLL.

**8.1.6.6 Синхронизация с расширенным спектром (SSC)**

ПРИМЕЧАНИЕ. Тактовые сигналы с расширенным спектром поддерживаются только для **DISP/LCD** и **MPU PLL** на этом устройстве.

Синхронизация с расширенным спектром не поддерживается для **DDR**, **PER** и **CORE PLL**. При включении **SSC** на **MPU PLL**, убедитесь, что максимальная частота **MPU** остается ниже максимальной номинальной частоты для выбранного **OPP** (более подробную информацию см. в Руководстве по данным для конкретного устройства).

Модуль поддерживает тактирование с расширенным спектром (**SSC**) на своих выходных тактовых импульсах. **SSC** используется для распространения

спектрального пика тактовых сигналов для уменьшения любых электромагнитных помех (EMI), которые могут быть вызваны основной тональностью тактового сигнала  или любой из его гармоник. Когда SSC ​​включен, спектр тактовых импульсов расширяется на величину разброса частот, а затухание определяется отношением разброса частот (Δf) и частотой модуляции (fm), т. е. [{10\*log10(Df/fm)}-10] дБ.

**8.1.6.6.1 Определение**

Целью **SSC** является изменение частоты исходных тактовых импульсов, которое расширяет сгенерированные помехи в большей полосе частот.

Теоретически **SSC** означает, что тактовый сигнал изменяется вокруг желаемой частоты. Например, для 1 ГГц, частота может быть 999,5 МГц в один момент и 1,0005 ГГц в другой. Когда **SSC** включено, спектр тактового сигнала расширяется на величину частотного расширения. Если делать ее постоянной  мощность тона должна быть распределена по более широкой полосе узких частот (с центром в желаемый тон). Чтобы сделать это изменение постоянным , происходит модуляция с дополнительным сигналом (называемый распространяющимся сигналом).

Создание SSC путем расширения начальной тактовой частоты выполняется путем определения следующих параметров:

- Частота распространения (отклонение), которая является отношением диапазона частоты расширения к исходной тактовой частоте

- Скорость модуляции (fm), которая используется для определения скорости цикла расширения тактовой частоты и времени, в течение которого генерируемая тактовая частота изменяется на Δf и возвращается к исходной частоте

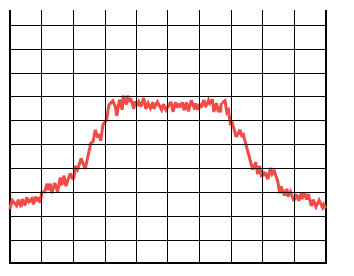
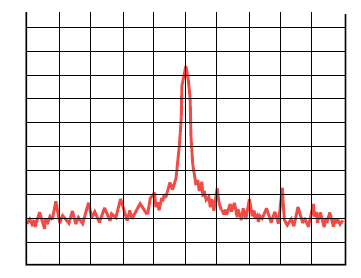
- Форма сигнала модуляции, описывающая кривую изменения во времени.

Уменьшение спектральной мощности в тактовых сигналах DPLL зависит от индекса модуляции (K), который является отношением частоты расширения, вычисленным из отклонения частоты (Δf) к скорости модуляции (fm).

**8.1.6.6.2 Влияние на тактовый сигнал**

На рис. 8-10 показан пример влияния треугольного расширения на тактовый сигнал.

**Рисунок 8-10. Влияние SSC на частоту**

****

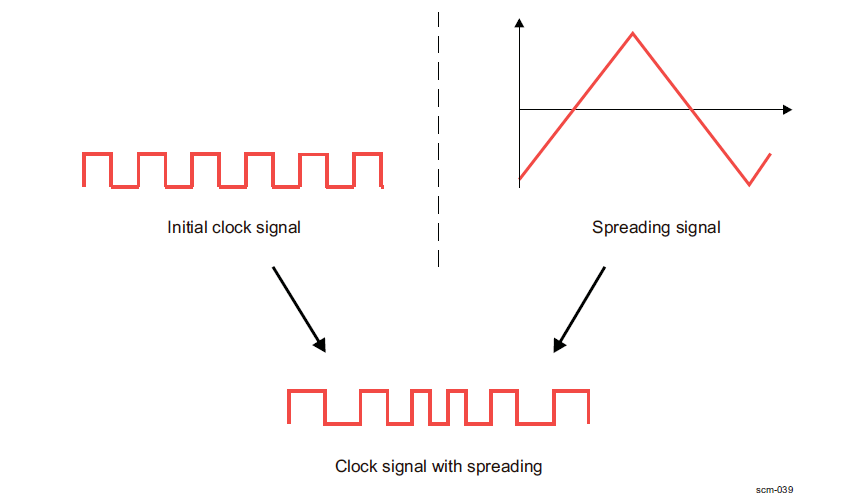
Спектр сигнала без расширения Спектр сигнала с треугольным расширением

На рис. 8-10 показано не только уменьшение мощности основного пика, но и более плоский вид модулированного сигнала. Минимальный уровень второго сигнала выше минимального уровня первого сигнала. Этот эффект является нормальным и возникает из-за шума, добавленного для модуляции.

ПРИМЕЧАНИЕ. Метод расширения рассеивает энергию пиков на других частотах, что уменьшает мощность пиков, но увеличивает общий шум сигнала.

На рис. 8-11 показано влияние треугольного расширения на тактовый сигнал во времени.

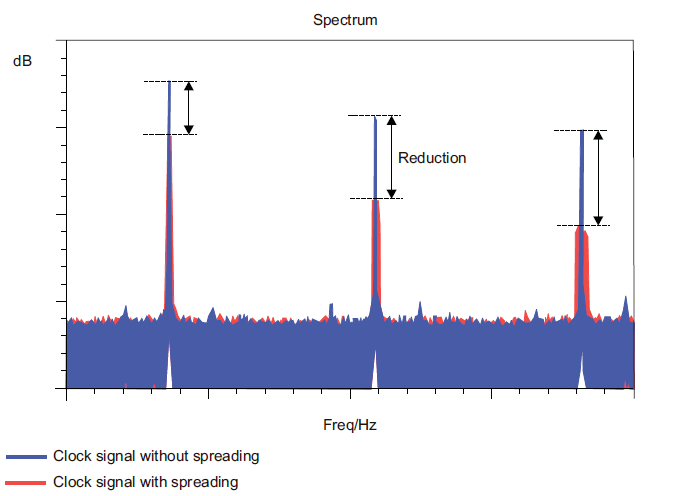
**Рисунок 8-11. Эффект SSC во временной области**



**8.1.6.6.3 Оценка уровня снижения электромагнитных помех**

На рис. 8-12 показано влияние расширения на тактовый сигнал и его гармоники.

**Рисунок 8-12. Пиковое снижение, вызванное спрейдингом по частоте**

****

Снижение электромагнитных помех можно оценить с помощью следующего уравнения:

Peak\_power\_reduction = 10 \* log ((Deviation\* fc) / fm

Где:

- Peak\_power\_reduction в дБ

• Deviationв % от начальной тактовой частоты (fc), равно Δf / fc

• fc — исходная тактовая частота в МГц.

• fm – частота распространения в МГц.

По уравнению (1) также можно вычислить отклонение, а затем и Δf, для искомого пика снижения мощности:

Отклонение = (fm / fc) \* 10 (Peak\_power\_reduction / 10)

Пример:

Для fc = 400 МГц отклонение = 1% пикового значения от fc (Δf = 4 МГц) и fm = 400 кГц; расчетное снижение пиковой мощности составляет 10 дБ.

**8.1.6.6.4 Расчет пропускной способности (правило Карсона для пропускной способности)**

Правило пропускной способности Карсона определяет приблизительные требования к пропускной способности компонентов системы связи для несущего сигнала, который частотно-модулирован непрерывным или широким спектром частот, а не одной частоты.

Правило пропускной способности Карсона выражается соотношением CBR = 2 \* (Δf + fm), где CBR — требуемая пропускная способность

, Δf — девиация пиковой частоты, а fm — самая высокая частота модулирующего сигнала. Например, ЧМ-сигнал с пиковой девиацией 5 кГц и максимальной звуковой частотой 3 кГц будет требует приблизительной полосы пропускания 2\*(5 3) = 16 кГц.

Теоретически любой FM-сигнал имеет бесконечное число боковых полос и, следовательно, бесконечную полосу пропускания, но на практике вся значительная энергия боковой полосы (98% и более) сосредоточена в пределах полосы пропускания, определяемой правилом пропускной способности Карсона.

**8.1.6.6.5 Управление генерацией SSC в устройстве**

SSC выполняется путем изменения делителя обратной связи (M) по треугольной схеме. Подразумевается, что частота выходных тактовых импульсов будут варьироваться по треугольной схеме. Частота этого паттерна будет модуляцией частоты (fm). Пик (ΔM) или амплитуда треугольного узора в процентах от M будет равна

к проценту разброса выходной частоты (Δf); то есть ΔM/M= Δf / fc. Следующая отметка с Finp частотой тактового сигнала на входе DPLL. Потому что он делится на N+1 перед входом в фазовый детектор, поэтому внутренняя опорная частота равна Fref = Finp / (N + 1).

Предположим, что центральная частота fc равна выходной частоте DPLL Fout, или fc= Fout = (Finp / (N +1))\*(М/М2). Так как это внутриполосная модуляция для DPLL, частота модуляции должна быть в полосе пропускания контура DPLL (минимальная полоса пропускания Fref / 70). Более высокая частота модуляции приведет к

меньшему распространению выходных тактовых импульсов.

SSC можно включить/отключить с помощью бита **CM\_CLKMODE\_DPLL\_xxx**.**DPLL\_SSC\_EN** (где xxx может быть любой из следующих **DPLL**: **MPU**, **DDR**, **DISP**, **CORE**, **PER**). Сигнал подтверждения **CM\_CLKMODE\_DPLL\_xxx**.DPLL **SSC\_ACK** уведомляет о точном начале и конце **SSC**. Когда SSC\_EN de-assert, SSC отключается только после завершения одного полного цикла треугольного паттерна, заданного частотой модуляции. Это сделано для того, чтобы сохранить среднюю частоту.

Частота модуляции (fm) может быть запрограммирована как отношение Fref/4; то есть значение, которое должно быть запрограммированным **ModFreqDivider** = **Fref** / **(4\*fm)**. ModFreqDivider делится на мантиссу и 2^Exponent (**ModFreqDivider = ModFreqDividerMantissa \* 2^ModFreqDividerExponent**). Мантисса управляет

7-битный сигнал ModFreqDividerMantissa через битовое поле **CM\_SSC\_MODFREQDIV\_DPLL\_xxx**.**MODFREQDEV\_MANTISSA**. Экспонента контролируется

3-битным сигналом **ModFreqDividerExponent** через битовое поле **CM\_SSC\_MODFREQDIV\_DPLL\_xxx**.**MODFREQDEV\_EXPONENT**.

ПРИМЕЧАНИЕ. Хотя одно и то же значение ModFreqDivider может быть получено с помощью различных комбинаций значения мантиссы и экспоненты, рекомендуется получить целевой **ModFreqDivider** с помощью программирования максимальной мантиссы и минимального показателя степени.

Чтобы определить разброс частот (Δf), необходимо контролировать ΔM, как объяснялось ранее. Чтобы определить ΔM, размер шага M для каждого Fref во время треугольного шаблона должен быть запрограммирован;

то есть,

ΔM = (2^ModFreqDividerExponent) \* ModFreqDividerMantissa \* DeltaMStep IF

ModFreqDividerExponent ≤ 3ΔM = 8 \* ModFreqDividerMantissa \* DeltaMStep IF

ModFreqDividerExponent > 3

DeltaMStep делится на целую часть и дробную часть. Целая часть управляется 2-битным сигналом DeltaMSTepInteger через битовое поле **CM\_SSC\_DELTAMSTEP\_DPLL\_xxx**.**DELTAMSTEP\_INTEGER**. Дробная часть управляется 18-битным сигналом **DeltaMStepFraction** через

Битовое поле **CM\_SSC\_DELTAMSTEP\_DPLL\_xxx**.**DELTAMSTEP\_FRACTION**.

Достигнутый разброс частот имеет превышение 20 процентов или погрешность +20 процентов. Если **CM\_CLKMODE\_DPLL**.**DPLL\_SSC\_DOWNSPREADis** установлен в 1, частотный разброс в нижней части вдвое больше запрограммированного значения. Разброс частоты по верхней стороне равен 0 (за исключением выброса, как

описано ранее).

Существует ограничение диапазона значений M. Ограничение: M-ΔM должно быть ≥ 20. Кроме того, M+ΔM должно быть ≤

2045. Если включена функция нисходящего расширения, M-2\*ΔM должно быть ≥ 20 и M ≤ 2045.

**8.1.6.6.6 Генерация SSC**

Настройка функции расширения не является обязательной при программировании DPLL. Эта функция обычно включается, когда тактовые  импульсы DPLL генерируют гармоники, которые потенциально могут мешать несущим частотам GSM.

Возьмем Display ADPLL с SSC и попробуем установить выходную частоту Fout = fc = 11 МГц.

Программное обеспечение, скорее всего, устанавливает DPLL выше, чтобы синхронизировать модуль DSS с более высокой функциональной частотой, а затем

устанавливает DISPC\_DIVISOR для достижения тактовой частоты пикселей 11 МГц. Но в этом примере PLL настроен на вывод 11 МГц. Частота источника входного тактового сигнала для Display ADPLL составляет Finp = 25 МГц.

1. Желаемая выходная частота может быть достигнута при следующем соотношении коэффициентов делителя: (M / M2) \* 1 / (N + 1) = Fout / Finp = 11 / 25. Делители, используемые в Display ADPLL, могут быть установлены в следующие диапазоны: N = 0..127; М = 0..2047; М2 = 1;2. Желаемая выходная частота достигается за счет

следующего выбор возможных значений делителя: M = 22; Н = 4; и M2 = 10. В этом случае рефернесная тактовая частота Fref = Finp / (N + 1) = 25 / (4 + 1) = 5 МГц.

Значение делителя обратной связи M = 22 выбрано для удовлетворения ограничения из раздела 8.1.6.6. Если, например, выбрано отклонение ΔM / M = Δf fc = 0,05 (5 %), имеем M + ΔM < 2045 и при этом время М + ΔМ > 20

После того, как регистры управления тактовой частотой настроены, можно настроить расширение спектра на сигнал тактовых импульсов.

2. Рассчитайте соотношение между центральной (выходной) частотой и частотой модуляции на основе желаемого снижения пиковой мощности (**PPR**) и выбранного относительное отклонение Δf/Fout, где Δf/Fout = fm/fc\* 10 ^(**PPR**/10). Для достижения PPR = 10 дБ при отклонении SSC (Δf / fc), выбранном равным 5 процентам, fm =

Δf/10(имп/10) = 55 кГц. Чтобы проверить, имеет ли частота модуляции соответствующее значение, проверьте находится ли он в пределах полосы пропускания контура DPLL или если fm < Fref / 70 = 5 / 70 = 71,4 кГц, что верно.

3. Вычислить содержимое битовых полей MODFREQDEV\_MANTISSA и MODFREQDEV\_EXPONENT на основе значения ModFreqDivider: ModFreqDivider = Fref / (4 \* fm) = 5 / (4 \* 0,055) = 22,73. Результирующее значение нужно представить в виде MODFREQDEV\_MANTISSA \* 2MODFREQDEV\_EXPONENT. Таким образом, мы можем аппроксимировать 23 = 23 \* 20. Аппроксимация лишь незначительно повлияет на PPR. Это означает, что мы должны написать MODFREQDEV\_EXPONENT = 0x0 и MODFREQDEV\_MANTISSA =

0x17.

4. Параметр DeltaMStep рассчитывается по формуле: DeltaMSStep = ΔM / ModFreqDivider. Поскольку ΔM = M \* (Δf / fc), DeltaMStep = M \* (Δf / fc) / ModFreqDivider. Таким образом, в этом примере DeltaMStep = 22 \* ​​0,05 / 23 = 0,047826. В этом случае запишите 0x0 в **DELTAMSTEP\_INTEGER** (биты 19:18). Чтобы выразить дробную часть 0,05 как

двоичный файл, вычислить: 0,047826 \* 2 ^ 18 = 12537,3, затем округлить до 12537, преобразовать целую часть в двоичную и запишите его в поле: **DELTAMSTEP\_FRACTIONAL** (биты 17:0) = **0x30F9**.

5. Расширение спектра должно быть разрешено с помощью бита **SSC\_EN**.

ПРИМЕЧАНИЕ. Необходимо тщательно настраивать расширение спектра на тактовые импульсы, чтобы избежать добавления шума на частоты, используемые другим модулем. Например, добавление расширения спектра по тактам для уменьшения шума на частотах GSM может «переместить» генерируемый шум на частоту

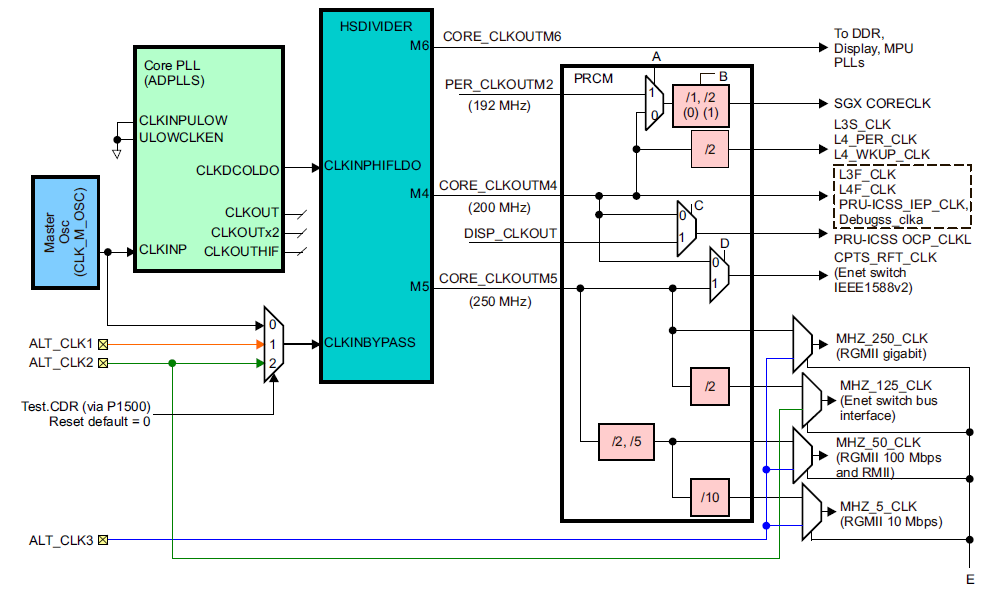
контроллера памяти и снизить его производительность.

Состояние функции модуляции можно контролировать с помощью бита **DPLL\_SSC\_ACK** соответствующего регистра.

**8.1.6.7 Описание PLL ядра**

Core PLL обеспечивает источник для большей части инфраструктуры устройств и тактирования периферии. Базовая PLL включает в себя **ADPLLS** с **HSDIVIDER** и дополнительными делителями и мультиплексорами, расположенными в **PRCM**, как показано на рисунке 8-13.

**Рисунок 8-13. Базовая ФАПЧ( PLL)**

****

**ALT\_CLK** должны использоваться для целей внутреннего тестирования и не должны использоваться в функциональном режиме.

**Таблица 8-21. PLL и тактовая частота**

|  |  |
| --- | --- |
| Mux Select | Register Bit Section 9.2.4.4 |
| A | PRCM.CLKSEL\_GFX\_FCLK[1] |
| B | PRCM.CLKSEL\_GFX\_FCLK[0] |
| C | PRCM.CLKSEL\_PRU-ICSS\_OCP\_CLK[0] |
| D | PRCM.CM\_CPTS\_RFT\_CLKSEL[0] |
| E | TEST.CDR (via P1500) |

В Таблице 8-22 приведены типичные частоты PLL и тактовые частоты. HSDIVIDER используется для создания трех разделенных тактовых сигналов М4, М5 и М6. M4 и M5 имеют номинальную частоту 200 и 250 МГц соответственно.

**Таблица 8-22. Стандартные частоты ядра PLL (МГц)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CLOCK | Source | Power-On-Reset /  HSDIVIDER Bypass | | OPP100 | | OPP50(1)(2) | |
| **DIV** | **Freq** | **DIV Value** | **Freq (MHz)** | **DIV Value** | **Freq (MHz)** |
| CLKDCOLDO (PLL  Lock frequency) | APLLS | **-** | **-** | **-** | **200** | **-** | **100** |
| CORE\_CLKOUTM4 | HSDIVIDERM4 | **-** | Mstr Xtal | **10** | **200** | **1** | **100** |
| L3F\_CLK, L4F\_CLK,  PRU-ICSS IEP CLK,  DebugSS clka,  SGX.MEMCLK,  SGX.SYSCLK | CORE\_CLKO  UTM4 | **-** | Mstr Xtal | **-** | **200** | **-** | **100** |
| L4\_PER, L4\_WKUP | CORE\_CLKO  UTM4 | **2** | Mstr Xtal / 2 | **2** | **100** | **2** | **50** |
| SGX CORECLK | CORE\_CLKO  UTM4 | **1** | Mstr Xtal | **1** | **200** | **1** | **100** |
|  |  | **2** | **100** | **2** | **50** |
| CORE\_CLKOUTM5 | HSDIVIDERM5 | **-** | Mstr Xtal | **8** | **250** | **1** | **100** |
| MHZ\_250\_CLK (Gigabit RGMII) | CORE\_CLKO  UTM5 | **-** | **NA** | **-** | **250** | **-** | **NA** |
| MHZ\_125\_CLK  (Ethernet Switch Bus Clk) | CORE\_CLKO  UTM5 | **2** | Mstr Xtal / 2 | **2** | **125** | **2** | **50** |
| MHZ\_50\_CLK (100  mbps RGMII or 10/100 RMII) | CORE\_CLKO  UTM5 | **5** | Mstr Xtal / 5 | **5** | **50** | **2** | **50** |
| MHZ\_5\_CLK (10 mbps RGMII) | MHZ\_50\_CLK | **10** | Mstr Xtal / 50 | **10** | **5** | **10** | **5** |
| CORE\_CLKOUTM6 | HSDIVIDER  M6 | **-** | Mstr Xtal | **4** | **500** | **1** | **100** |

(1) Не все интерфейсы и периферийные модули доступны в OPP50. Для получения дополнительной информации см. техническое описание конкретного устройства.

(2) Ограничения при использовании OPP50 см. в документе AM335x ARM Cortex-A8 Microprocessors (MPU) Silicon Errata (номер публикации SPRZ360).

Модуль ADPLLS поддерживает два разных режима обхода через свой внутренний режим **MNBypass** и их внешний режим обхода режима ожидания с низким энергопотреблением. PLL находятся в режиме **MNBypass** после сброса при включении питания и может быть настроен программным обеспечением для перехода в режим обхода режима ожидания с низким энергопотреблением для отключения питания. Когда Core PLL сконфигурирован в режиме обхода, **HSDIVIDER** входит в режим обхода и вход **CLKINBYPASS** подается на выходы **M4**, **M5** и **M6**. **CLKINBYPASS** по умолчанию является ведущим входом генератора (обычно 24 МГц).

**Таблица 8-23. Тактовые импульсы интерфейса шины**

|  |  |
| --- | --- |
| L3F\_CLK | SGX530 (MEMCLK & SYSCLK), LCDC, MPU Subsystem, GEMAC Switch (Ethernet), DAP, PRU-ICSS, EMIF, TPTC, TPCC, OCMC RAM, DEBUGSS, AES, SHA |
| L3S\_CLK | USB, TSC, GPMC, MMCHS2, McASP0, McASP1 |
| L4\_PER\_CLK | DCAN0, DCAN1, DES ,DMTIMER2, DMTIMER3, DMTIMER4, DMTIMER5, DMTIMER6,DMTIMER7eCAP/eQEP/ePWM0, eCAP/eQEP/ePWM1, eCAP/eQEP/ePWM2,eFuse  ELM, GPIO1, GPIO2, GPIO3, I2C1, I2C2, IEEE1500, LCD, Mailbox0,McASP0, McASP1,MMCHS0, MMCHS1, OCP Watchpoint, PKARNG, SPI0, SPI1, Spinlock,  UART1, UART2, UART3, UART4, UART5 |
| L4\_WKUP\_CLK | ADC\_TSC, Clock Manager, Control Module,DMTIMER0, DMTIMER1\_1MS, GPIO0,I2C0, M3UMEM, M3DMEM, SmartReflex0, SmartReflex1,UART0, WDT0, WDT1 |

**8.1.6.7.1 Конфигурация PLL для ядра**

1. Переключите PLL в режим байпас, установив для **CM\_CLKMODE\_DPLL\_CORE**.**DPLL\_EN** значение **0x4**.

2. Подождите, пока **CM\_IDLEST\_DPLL\_CORE**.**ST\_MN\_BYPASS** = 1, чтобы убедиться, что PLL находится в байпасе.

(**CM\_IDLEST\_DPLL\_CORE**.**ST\_DPLL\_CLK** также должен измениться на **0**, чтобы указать, что PLL разблокирован).

3. Настройте значения умножения и деления, установив **CM\_CLKSEL\_DPLL\_CORE**.**DPLL\_MULT** и **DPLL\_DIV** на нужные значения.

4. Настройте делители **M4**, **M5** и **M6**, установив биты **HSDIVIDER\_CLKOUT1\_DIV** в **CM\_DIV\_M4\_DPLL\_CORE**, **CM\_DIV\_M5\_DPLL\_CORE** и **CM\_DIV\_M6\_DPLL\_CORE** в

желаемые значения.

5. Переключитесь в режим блокировки, установив для **CM\_CLKMODE\_DPLL\_CORE**.**DPLL\_EN** значение **0x7**.

6. Подождите, пока **CM\_IDLEST\_DPLL\_CORE**.**ST\_DPLL\_CLK** = **1**, чтобы убедиться, что PLL заблокирован.

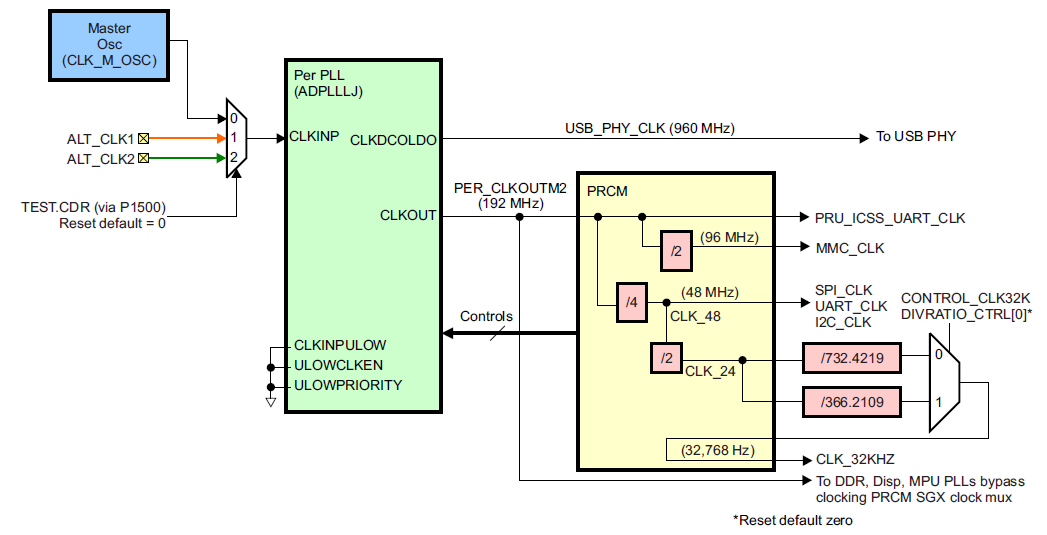
(**CM\_IDLEST\_DPLL\_CORE**.**ST\_MN\_BYPASS** также должен измениться на **0**, чтобы обозначить, что **PLL** вышел из режима байпас).

Примечание. Делители **M4**, **M5** и **M6** также можно менять «на лету», чтобы не было необходимости устанавливать **PLL** в режим байпас и обратно в режим блокировки. После изменения **CM\_DIV\_Mx\_DPLL\_CORE**.**DPLL\_CLKOUT\_DIV** проверьте **CM\_DIV\_Mx\_DPLL\_CORE**.**DPLL\_HSDIVIDER\_CLKOUT1\_DIVCHACK** для переключения (изменение с 0 на 1 или от 1 до 0), чтобы увидеть, было ли изменение подтверждено **PLL**.

**8.1.6.8 Описание периферийной PLL**

**Per PLL** обеспечивает источник для функционального тактирования периферии. **Per PLL** включает в себя **ADPLLLJ** и дополнительные делители и мультиплексоры, расположенные в **PRCM**, как показано:

**Рисунок 8-14. Периферийная структура PLL**

****

ALT\_CLK должны использоваться для целей внутреннего тестирования и не должны использоваться в функциональном режиме.

**PLL** залочен на частоте **960 МГц**. Выходной сигнал **PLL** делится на делитель **M2** для получения частоты 192 МГц - **CLKOUT**. Эти тактовые импульсы вводятся в **PRCM** для формирования тактов **PRU-ICSS UART**. Есть делитель /2 для создания 96 МГц для **MMC\_CLK**. Тактовая частота также делится внутри **PRCM** на фиксированный делитель /4 для создания тактовоай частота 48 МГц для модулей **SPI**, **UART** и **I2C**. Тактовая частота 48 МГц далее делится на фиксированный коэффициент /2 и фиксированный делитель /732,4219 для создания точной тактовой(RTC) частоты 32,768 кГц для использования в Таймерах и устранения дребезга.

**Таблица 8-24. На типичные частоты PLL (МГц)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Clock | Source | Power-On-Reset / PLL  Bypass | | OPP100 | | OPP50(1)(2) | |
| **DIV Value** | **Freq** | **DIV Value** | **Freq**  **(MHz)** | **DIV Value** | **Freq**  **(MHz)** |
| PLL Lock frequency | PLL | - | - | - | 960 | - | 960 |
| USB\_PHY\_CLK | CLKDCOLDO | - | Held Low | - | 960 | - | 960 |
| PER\_CLKOUTM2 | CLKOUT of  ADPLLLJ  CLKOUT uses PLL’s  M2 Divider when PLL  is locked and PLL’s  N2 divider when PLL  Bypass | N2 is 0 on  power-on-reset | Mstr Xtal/  (N2+1) | 5 | 192 | 10 | 96 |
| MMC\_CLK | PER\_CLKOUTM2 | 2 | Mstr Xtal/  ((N2+1)\*2) | 2 | 96 | 2 | 48 |
| SPI\_CLK,  UART\_CLK,  I2C\_CLK | PER\_CLKOUTM2 | 4 | Mstr Xtal/  ((N2+1)\*4) | 4 | 48 | 4 | 24 |
| CLK\_24 | CLK\_48 | 2 | CLK\_48 /2 | 2 | 24 | 2 | 12 |
| CLK\_32KHZ | CLK\_24  (output of CLK\_48/2) | 732.4219 | CLK\_24 /  <CLK32\_DIV> | 732.4219 | 0.032768 | 366.2109 | 0.032768 |

(1) Ограничения при использовании OPP50 см. в документе AM335x ARM Cortex-A8 Microprocessors (MPU) Silicon Errata (номер публикации SPRZ360).

(2) Не все интерфейсы и периферийные модули доступны в OPP50. Для получения дополнительной информации см. AM335x ARM Cortex-A8. Микропроцессоры (MPU) Silicon Errata (номер в литературе SPRZ360).

Модуль ADPLLLJ поддерживает два разных режима байпас через свой внутренний режим MNBypass и их внешний режим байпас режима ожидания с низким энергопотреблением. PLL находится в режиме MNBypass после сброса при включении питания и может быть настроен программным обеспечением для перехода в режим байпас режима ожидания с низким энергопотреблением для отключения питания.

**PER PLL** может использовать режим байпас с бездействием с низким энергопотреблением. Когда выбран режим внутреннего байпаса, выход **CLKOUT** управляется CLKINP/(N2+1), где N2 управляется **PRCM**. **CLKINP** по умолчанию вход задающего генератора (обычно 24 МГц)

**8.1.6.8.1 Настройка периферийного PLL**

Следующие шаги подробно описывают, как настроить периферийный PLL.

1. Переключите PLL в режим байпас, установив для **CM\_CLKMODE\_DPLL\_PER**.**DPLL\_EN** значение **0x4**.

2. Подождите, пока **CM\_IDLEST\_DPLL\_PER**.**ST\_MN\_BYPASS = 1**, чтобы убедиться, что PLL находится в байпасе.

(**CM\_IDLEST\_DPLL\_PER**.**ST\_DPLL\_CLK** также должен измениться на **0**, чтобы указать, что **PLL** разблокирован).

3. Настройте значения умножения и деления, установив **CM\_CLKSEL\_DPLL\_PER**.**DPLL\_MULT** и **DPLL\_DIV** до желаемых значений.

4. Настройте делитель **M2**, установив **CM\_DIV\_M2\_DPLL\_PER**.**DPLL\_CLKOUT\_DIV** на желаемое значение.

5. Переключитесь в режим блокировки, установив для **CM\_CLKMODE\_DPLL\_PER**.**DPLL\_EN** значение **0x7**.

6. Подождите, пока **CM\_IDLEST\_DPLL\_PER**.**ST\_DPLL\_CLK = 1**, чтобы убедиться, что PLL заблокирован.

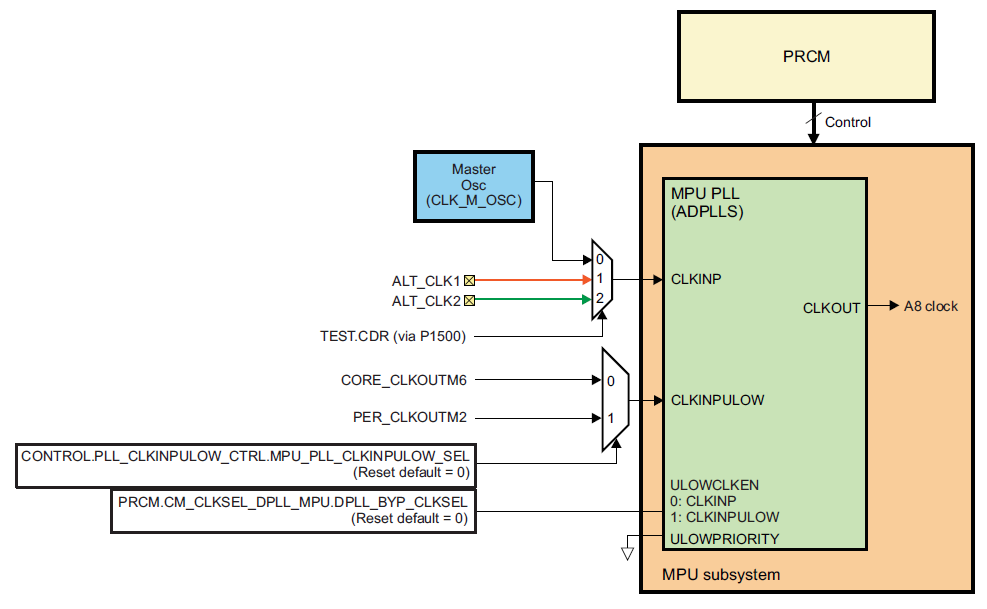
(**CM\_IDLEST\_DPLL\_PER**.**ST\_MN\_BYPASS** также должен измениться на **0**, чтобы указать, что PLL вышел из режима байпас).

***Примечание***: делитель **M2** также можно менять «на лету» (т. е. нет необходимости переводить PLL в режим байпаса и обратно в режим блокировки). После изменения **CM\_DIV\_M2\_DPLL\_PER**.**DPLL\_CLKOUT\_DIV** проверьте **CM\_DIV\_M2\_DPLL\_PER**.**DPLL\_CLKOUT\_DIVCHACK** для переключения (изменение с 0 на 1 или с 1 на 0), чтобы увидеть что изменение было подтверждено **PLL**.

**8.1.6.9 PLL MPU Описание**

Подсистема MPU Cortex A8 включает в себя внутреннюю ADPLLS для создания требуемого тактирования MPU Cortex A8. Эта PLL управляется выходом задающего генератора через регистры PRCM.

**Рисунок 8-15. Структура PLL подсистемы MPU**



Например:

Для частоты для MPU, скажем, 600 МГц, настраивается ADPLLS (PLL заблокирован на частоте 1200 МГц и Делитель M2 =1), поэтому ожидается CLKOUT = 600 МГц.

Вход ULOWCLKEN из программируемого регистра PRCM выбирает, будет CLKINP или CLKINPULOW является обходным источником синхронизации. Это безглючный переключатель. Когда выбран CLKINP, он поступает через делитель ADPLLS 1/(N2+1). Регистр PRCM по умолчанию равен 0 при включении питания, чтобы выбрать источник CLKINP.

Вход CLKINPULOW может быть получен от CORE\_CLKOUTM6 в Core PLL или от PER\_CLKOUTM2 из Per PLL. Эти выходные тактовые сигналы PLL могут использоваться в качестве альтернативных источников тактовых импульсов в вариантах активного использования с низким энергопотреблением для подсистемы тактирования MPU, когда PLL находится в режиме обхода.

**8.1.6.9.1 Настройка PLL MPU**

Следующие шаги подробно описывают, как настроить MPU PLL.

1. Переключите PLL в обходной режим, установив в **CM\_CLKMODE\_DPLL\_MPU.DPLL\_EN** значение 0x4.

2. Подождите, пока **CM\_IDLEST\_DPLL\_MPU.ST\_MN\_BYPASS** = 1, чтобы убедиться, что PLL находится в обходе. (**CM\_IDLEST\_DPLL\_MPU.ST\_DPLL\_CLK** также должен измениться на 0, чтобы указать, что PLL разблокирован).

3. Настройте значения умножения и деления, установив **CM\_CLKSEL\_DPLL\_MPU.DPLL\_MULT** и **DPLL\_DIV** на нужные значения.

4. Настройте делитель M2, установив **CM\_DIV\_M2\_DPLL\_MPU.DPLL\_CLKOUT\_DIV** на желаемое значение.

5. Переключитесь в режим блокировки, установив в **CM\_CLKMODE\_DPLL\_MPU.DPLL\_EN** значение 0x7.

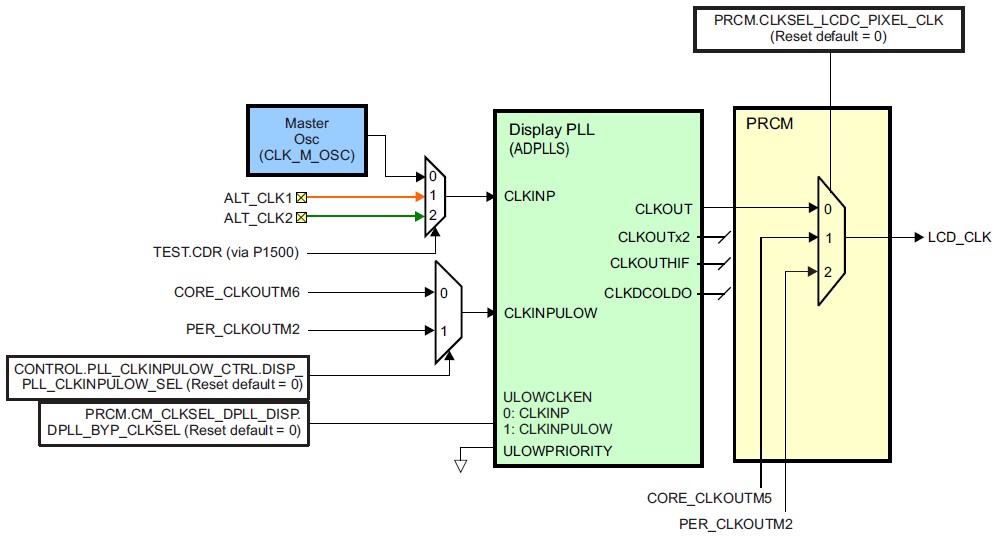
6. Подождите, пока **CM\_IDLEST\_DPLL\_MPU.ST\_DPLL\_CLK** = 1, чтобы убедиться, что PLL заблокирован. (CM\_IDLEST\_DPLL\_MPU.ST\_MN\_BYPASS также должен измениться на 0, чтобы указать, что PLL вышел из режима байпас).

***Примечание***: делитель M2 также можно менять «на лету» (т. е. нет необходимости переводить PLL в режим байпаса и обратно в режим блокировки). После изменения **CM\_DIV\_M2\_DPLL\_MPU.DPLL\_CLKOUT\_DIV** проверьте **CM\_DIV\_M2\_DPLL\_MPU.DPLL\_CLKOUT\_DIVCHACK** для переключения (изменение с 0 на 1 или с 1 на 0), чтобы увидеть если изменение было подтверждено PLL.

**8.1.6.10 Описание PLL дисплея**

PLL дисплея обеспечивает тактовую частоту пикселей, необходимую для ЖК-дисплея, и не зависит от других периферийных и инфраструктурных тактовых импульсов. PLL синхронизируется с задающим генератором. Делитель **ADPLLS M2** определяет выходную тактовую частоту, которая стробируется PRCM, как показано на рисунке 8-16.

**Рисунок 8-16. Структура PLL дисплея**

****

Например: скажем, частота для синхронизации пикселей 100 МГц, ADPLLS настроен (PLL заблокирован на частоте 200 МГц и M2 Divider =1), чтобы ожидать что CLKOUT = 100 МГц.

Вход ULOWCLKEN из программируемого регистра PRCM выбирает, будет ли CLKINP или CLKINPULOW является обходным источником синхронизации. Это безглючный переключатель. Когда выбран CLKINP, он поступает через Делитель ADPLLS 1/(N2+1). Регистр PRCM по умолчанию равен 0 при включении питания, чтобы выбрать источник CLKINP.

Вход CLKINPULOW снабжается от CORE\_CLKOUTM6 в Core PLL или от PER\_CLKOUTM2 в Per PLL. Этот выходной тактовый сигнал PLL можно использовать в качестве альтернативного источника тактового сигнала при активном использовании в случаях с низким энергопотреблением тактирования пикселей, когда PLL дисплея находится в режиме обхода.

**8.1.6.10.1 Настройка PLL дисплея**

Следующие шаги подробно описывают, как настроить PLL дисплея:

1. Переключите PLL в обходной режим, установив для **CM\_CLKMODE\_DPLL\_DISP.DPLL\_EN** значение 0x4.

2. Подождите, пока **CM\_IDLEST\_DPLL\_DISP.ST\_MN\_BYPASS** = 1, чтобы убедиться, что PLL находится в обходе.(**CM\_IDLEST\_DPLL\_DISP.ST\_DPLL\_CLK** также должен измениться на 0, чтобы указать, что **PLL** разблокирован).

3. Настройте значения умножения и деления, установив **CM\_CLKSEL\_DPLL\_DISP.DPLL\_MULT** и **DPLL\_DIV** на нужные значения.

4. Настройте делитель **M2**, установив **CM\_DIV\_M2\_DPLL\_DISP.DPLL\_CLKOUT\_DIV** на нужное значение.

5. Переключитесь в режим блокировки, установив для **CM\_CLKMODE\_DPLL\_DISP.DPLL\_EN** значение 0x7.

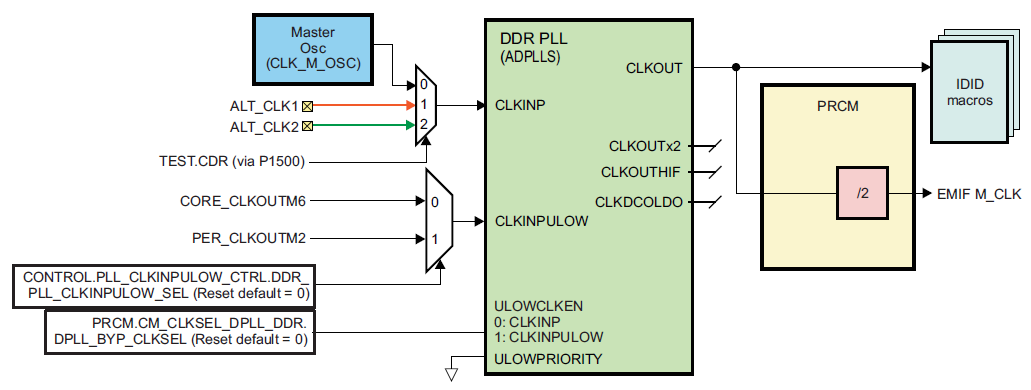
6. Подождите, пока **CM\_IDLEST\_DPLL\_DISP.ST\_DPLL\_CLK** = 1, чтобы убедиться, что PLL заблокирован.(**CM\_IDLEST\_DPLL\_DISP.ST\_MN\_BYPASS** также должен измениться на 0, чтобы указать, что PLL вышел из режима байпасс).

***Примечание***: делитель M2 также можно менять «на лету» (т. е. нет необходимости переводить PLL в режим байпаса и обратно в режим блокировки). После изменения **CM\_DIV\_M2\_DPLL\_DISP.DPLL\_CLKOUT\_DIV** проверьте переключение **CM\_DIV\_M2\_DPLL\_DISP.DPLL\_CLKOUT\_DIVCHACK** (изменение с 0 на 1 или с 1 на 0) чтобы увидеть, было ли изменение подтверждено PLL.

**8.1.6.11 Описание DDR PLL**

PLL DDR обеспечивает тактирование, необходимое для макросов DDR и EMIF, и не зависит от других периферийных и инфраструктурных тактовых импульсов. PLL синхронизируется с задающим генератором. Делитель ADPLLS M2 определяет выходную тактовую частоту, которая напрямую связана с макросами DDR. Тактовые импульсы также маршрутизируются через PRCM, где используется фиксированный делитель /2 для создания M\_CLK, используемого EMIF как показано на рис. 8-17.

**Рисунок 8-17. Структура DDR PLL**

****

Информацию об OPP см. в руководстве по данным для конкретного устройства.

Пример частоты для тактовой частоты DDR, скажем, 266 МГц, ADPLLS настроен (PLL заблокирован на 532 МГц и M2 Divider =1), чтобы получить CLKOUT = 266 МГц.

Вход ULOWCLKEN из программируемого регистра PRCM выбирает, будет ли CLKINP или CLKINPULOW является обходным источником синхронизации. Это безглючный переключатель. Когда выбран CLKINP, он поступает через делитель ADPLLS 1/(N2+1). Регистр PRCM по умолчанию равен 0 при включении питания, чтобы выбрать источник CLKINP.

Вход CLKINPULOW может быть получен от CORE\_CLKOUTM6 в Core PLL или PER\_CLKOUTM2 в Per PLL. Эти выходные тактовые сигналы PLL могут использоваться в качестве альтернативных источников тактовых импульсов в вариантах активного использования с низким энергопотреблением для тактирования DDR, когда PLL находится в режиме байпасс(обход).

**8.1.6.11.1 Настройка DDR PLL**

Следующие шаги подробно описывают, как настроить DDR PLL.

1. Переключите PLL в обходной режим, установив для **CM\_CLKMODE\_DPLL\_DDR.DPLL\_EN** значение 0x4.

2. Подождите, пока **CM\_IDLEST\_DPLL\_DDR.ST\_MN\_BYPASS** = 1, чтобы убедиться, что PLL находится в обходе. (**CM\_IDLEST\_DPLL\_DDR.ST\_DPLL\_CLK** также должен измениться на 0, чтобы указать, что PLL разблокирован).

3. Настройте значения умножения и деления, установив **CM\_CLKSEL\_DPLL\_DDR.DPLL\_MULT** и **DPLL\_DIV** на нужные значения.

4. Настройте делитель M2, установив **CM\_DIV\_M2\_DPLL\_DDR.DPLL\_CLKOUT\_DIV** на желаемое значение.

5. Переключитесь в режим блокировки, установив для **CM\_CLKMODE\_DPLL\_DDR.DPLL\_EN** значение 0x7.

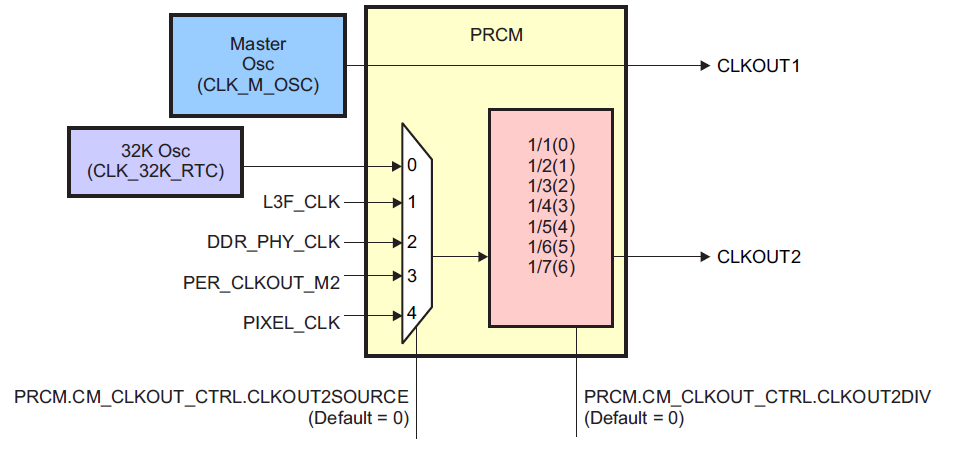
6. Подождите, пока **CM\_IDLEST\_DPLL\_DDR.ST\_DPLL\_CLK** = 1, чтобы убедиться, что PLL заблокирован.(**CM\_IDLEST\_DPLL\_DDR.ST\_MN\_BYPASS** также должен измениться на 0, чтобы указать, что PLL вышел из режима байпасс).

***Примечание***: делитель М2 также можно менять «на лету» (т. е. нет необходимости переводить PLL в режим байпаса и в режим блокировки). После изменения **CM\_DIV\_M2\_DPLL\_DDR.DPLL\_CLKOUT\_DIV** проверьте **CM\_DIV\_M2\_DPLL\_DDR.DPLL\_CLKOUT\_DIVCHACK** (изменение с 0 на 1 или с 1 на 0), чтобы увидеть если изменение было подтверждено PLL.

**8.1.6.12 Сигналы CLKOUT**

Сигналы CLKOUT1 и CLKOUT2 поступают на контактные площадки устройства и в основном должны использоваться в качестве контрольных точек отладки. Использование этих сигналов для критичных по времени внешних цепей не рекомендуется из-за непредсказуемого джиттера производительности. Для получения дополнительной информации см. техническое описание устройства, Процессоры AM335x Sitara (литература номер SPRS717). CLKOUT1 создается из задающего генератора. CLKOUT2 можно получить из 32-кГц кварцевого генератора или любой из выходов PLL (кроме MPU PLL). Выбранный выход может быть дополнительно модифицирован программируемым делителем для создания желаемой выходной частоты.

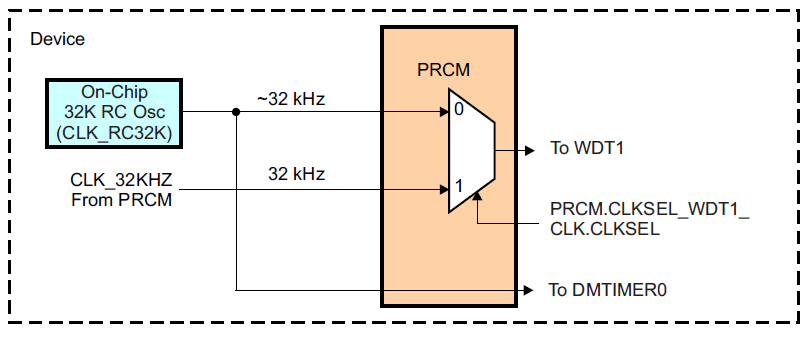
**Рисунок 8-18. Сигналы CLKOUT**

****

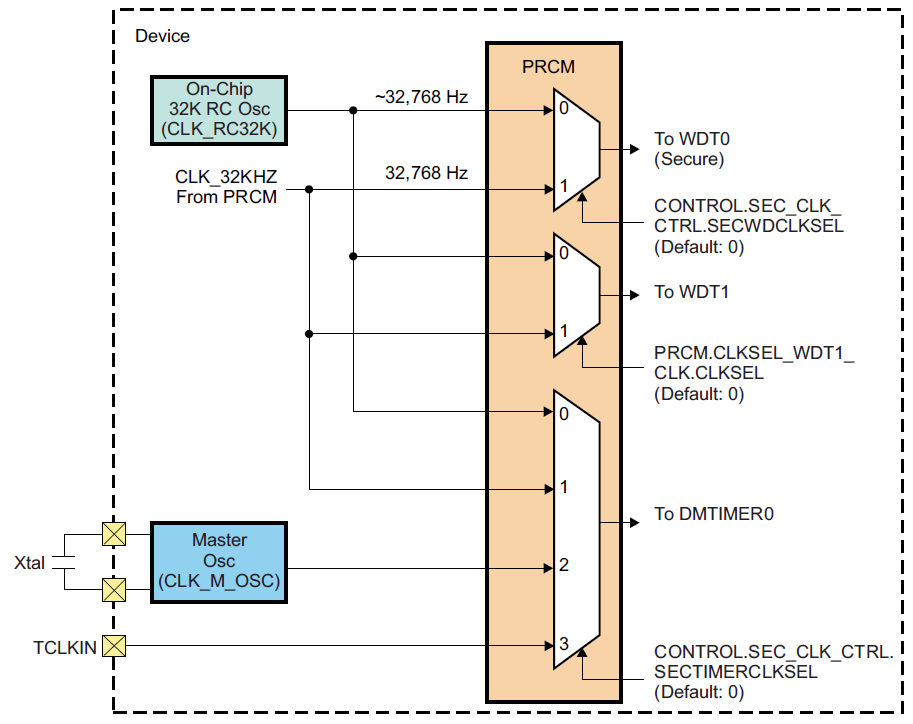
**8.1.6.13 Структура тактирования таймера**

Тактовая частота CLK\_32KHZ — это точная тактовая частота 32,768 кГц, полученная из PER PLL, и ее также можно выбирать для тактирования WDT1. DMTIMER0 может синхронизироваться только от внутреннего RC-генератора (CLK\_RC32K). Параметры тактирования показаны на рис. 8-19.

**Рисунок 8-19. Выбор тактирования сторожевого таймера**

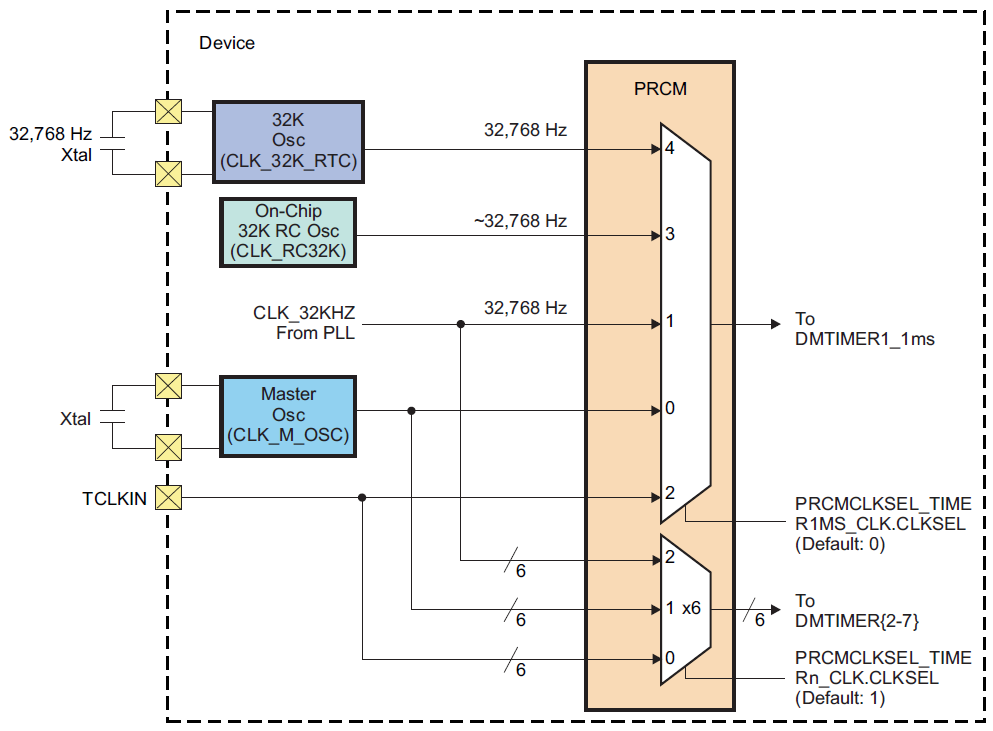


**Рисунок 8-20. Выбор тактирования сторожевого таймера и безопасного таймера (только для защищенных устройств)**



Все мультиплексоры выбираются в формате PRCM, если на диаграммах явно не указано иное. Выбор тактирования для других модулей таймера устройства показан на рис. 8-21. Задающий генерато CLK\_32KHZ и внешний вывод (TCLKIN) являются дополнительными источниками тактовых импульсов, доступными для таймеров, которые могут выбираться на основе конечного применения.

**DMTIMER1** реализован с помощью модуля DMTimer\_1ms, способного генерировать точный 1 мс с использованием тактовой частоты 32,768 кГц. В режимах пониженного энергопотребления задающий генератор отключен. CLK\_32KHZ также будет недоступен в этом сценарии, так как он получен от Master Osc на основе PER PLL. Следовательно, в режимах с низким энергопотреблением DMTIMER1 в домене WKUP может использовать генератор 32K RC для генерации тиков 1мс для ОС (операционной системы) и пробуждения по таймеру. Так как большинство приложении ожидают точный такт ОС в 1 мс, чего не может обеспечить неточный генератор 32K RC (16-60 кГц) в качестве дополнительной опции предоставляется отдельный генератор 32768 Гц (32K Osc).

**Рисунок 8-21. Выбор тактирования таймера** 

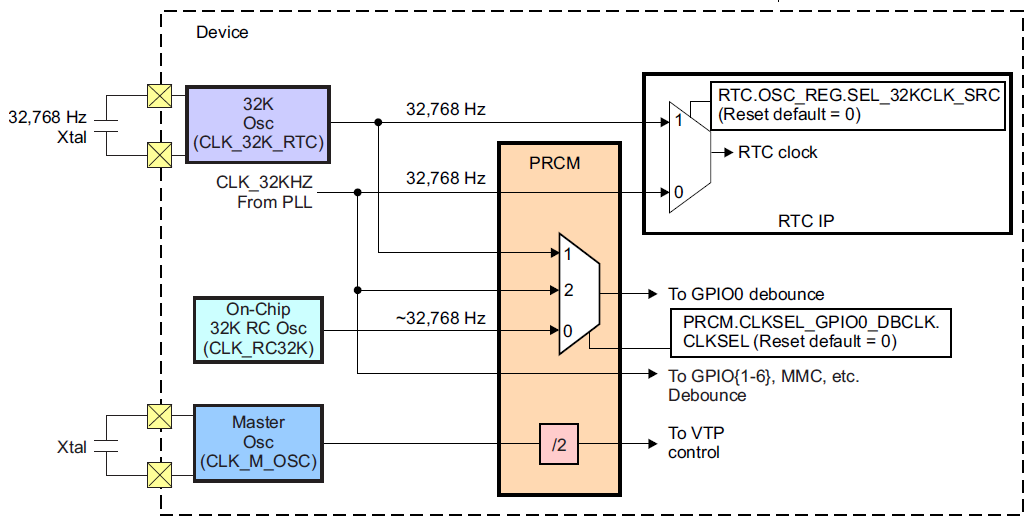
Все мультиплексоры выбираются в формате PRCM, если на диаграммах явно не указано иное.

Параметры RTC, Антидребезга и VTP показаны на рис. 8-22. В режимах пониженного энергопотребления для GPIO0 в домене WKUP можно использовать точный кварцевый генератор с частотой 32768 Гц или неточный (от 16 кГц до 60 кГц) 32K RC-генератор при выключенном главном генераторе (Master Osc).

Для 32K Osc требуется внешний кварц 32768 Гц.

Все мультиплексоры выбираются в формате PRCM, если на диаграммах явно не указано иное.

**Рисунок 8-22. Выбор тактирования RTC, VTP и Антидребезга**

****

Все мультиплексоры выбираются в формате PRCM, если на диаграммах явно не указано иное.

**8.1.7 Управление сбросом**

**8.1.7.1 Обзор**

PRCM управляет сбросом всех доменов питания внутри устройства и генерацией одного выходного сигнала через пин устройства, WARMRSTn, для внешнего использования. PRCM не знает и не контролирует сброс, генерируемый локально в модуле, например, через бит регистра конфигурации OCP **IPName\_SYSCONFIG.SoftReset**.

Все выходы сброса PRM активируются асинхронно. Эти выходы имеют активный низкий уровень, за исключением сброса PLL. Отмена подтверждения синхронна с тактовыми импульсами, которые запускают счетчик, используемый для остановки или задержки сброса отмены подтверждения при деактивации источника. Эти тактовые импульсы CLK\_M\_OSC используются всеми диспетчерами сброса. Все модули, получающие сброс, сгенерированный PRCM, обработают сброс как асинхронный и реализуют локальную повторную синхронизацию после деактивации по мере необходимости.

Для каждого домена питания требуется один или несколько диспетчеров сброса. Независимое управление множественным сбросом доменов требуется для выполнения требований последовательности сброса всех модулей в домене электропитания.

**8.1.7.2 Понятия и определения сброса**

PRCM насчитывает множество источников сброса. Ниже приведен список квалификаторов источника сброса:

- Cold reset:: влияет на всю логику данного объекта.

- Warm reset: это частичный сброс, который не влияет на всю логику данного объекта.

- Global reset: влияет на все устройство.

- Local reset: затрагивает часть устройства (например, 1 домен питания)

- S/W reset: он инициируется программным обеспечением.

- H/W reset: аппаратно управляемый.

Каждый источник сброса определяется как холодного или теплого типа. Холодные типы являются синонимами перезагрузки при включении питания (power-on-reset POR). Такие источники применяются глобально в каждом принимающем объекте (т. е. подсистеме, модуле, макроячейке) при настройке. К событиям холодного сброса относятся: включение питания устройства, включение питания домена, и сбои при программировании eFuse.

Типы теплого сброса не обязательно применяются глобально в каждом принимающем объекте. Модуль может использовать теплый сброс для сброса подмножества его логики. Это часто делается для ускорения времени восстановления сброса, т.е. для перехода в безопасное рабочее состояние по сравнению со временем, необходимым при получении холодного сброса. Теплое событие сброса включают в себя: инициированное программным обеспечениме для домена питания, по тайм-ауту сторожевого таймера, при нарушение безопасности, вызванное внешними источниками, а так же при начале эмуляции.

Источники сброса, теплого или холодного типа, предназначенные для действия всего устройства, классифицируются как глобальные источники. Источники сброса , предназначенные для регионального воздействия, относятся к локальным источникам.

Каждый менеджер сброса имеет два выхода сброса. Один из них — холодный сброс, сгенерированный из группы глобальных и локальных источников холодного сброса, которые он получает. Другой - теплый + холодный сброс, созданный из комбинированной группы, глобальных и локальных, источники холодного и теплого сброса, которые он получает.

Менеджер сброса устанавливает один или оба своих выхода сброса асинхронно при подтверждении источника сброса. Отмена подтверждения сброса продлевается за время, в течение которого источник снимается. После этого менеджер сброса продливает активный период выходов сброса после освобождения источника сброса в соответствии с внутренними ограничениями PRCM и ограничениями устройства. Некоторые длительности сброса могут быть настроены программно.

Большинство (но не все) источников сброса регистрируются в регистрах статуса сброса PRCM. Один и тот же выход сброса может быть активирован несколькими источниками сброса, и один и тот же источник сброса обычно может активировать несколько выходов сброса. Все выходные сигналы сброса PRCM имеют активный низкий уровень. Несколько соглашений используются в этом докуменет для именования сигналов и портов. Они включают:

• "\_RST" в имени сигнала или порта используется для обозначения сигнала сброса.

• "\_PWRON\_RST" в имени сигнала или порта используется для обозначения источника холодного сброса.

**8.1.7.3 Глобальный сброс при включении питания (холодный сброс)**

Существует несколько источников холодного сброса. См. Таблицу 8-25 для сводки различных источников сброса.

**8.1.7.3.1 Сброс при включении питания (PORz)**

Источником питания при сбросе является сигнал PORz на устройстве. Все на устройстве сбрасывается с утверждением питания при сбросе. Этот сброс не блокируется. **PORz** может управляться внешними устройствами управления питанием или схемой управления питанием. При включении питания, когда питание устройства увеличивается, **PORz** нужно управлять низким уровнем сигнала (Low). Когда питание устоится и достигнет своих установившихся значений, **PORz** нужно переключить в высокий уровень (High). Во время нормальной работы, когда любой из источников питания устройства выключен., PORz должен быть переведен в низкий уровень (Low).

**8.1.7.3.2 Последовательность PORz**

- Вывод **PORz** на чипе устанавливается (переходит в низкий уровень). ***Примечание***. Состояние **nRESETIN\_OUT** во время утверждения **PORz** должно быть безразлично, оно не должно влиять на **PORz** (только в том случае, если они оба подтверждены, а **nRESETIN\_OUT** отключается после **PORz**, вы получите повторную блокировку контактов конфигурации загрузки и может увидеть теплый флаг **nRESETIN\_OUT**, установленный в **PRCM** по сравнению с **POR**).

- Все входы-выходы перейдут в указанное в техническом описании устройства состояние ( см AM335x Sitara Processors (номер публикации SPRS717)

- При подаче питания значение **PORz** передается на **PRCM**.

- **PRCM** сбросит весь чип, и вс. логика, использующую асинхронный сброс. **nRESETIN\_OUT** перейдет в низкий уровень, указывая на то, что выполняется сброс.

- Внешние тактовые сигналы начнут подаваться, и PRCM передаст их чипу, поддерживая режим байпас у PLL.

- Вся логика, использующая сброс синхронизации, будет сброшена.

- Когда питание и тактовая частота микросхемы стабильны, **PORz** должен быть деактивирован.

- Выводы конфигурации загрузки фиксируются после отключения вывода **PORz**.

- Элементы управления ячейкой ввода-вывода от IP для всех вводов-выводов, за некоторыми исключениями (подробности см. в техническом описании), управляются GPIO-модулем. GPIO переводит все IO в режим ввода.

- Сброс FuseFarm будет отменен, чтобы начать сканирование eFuse.

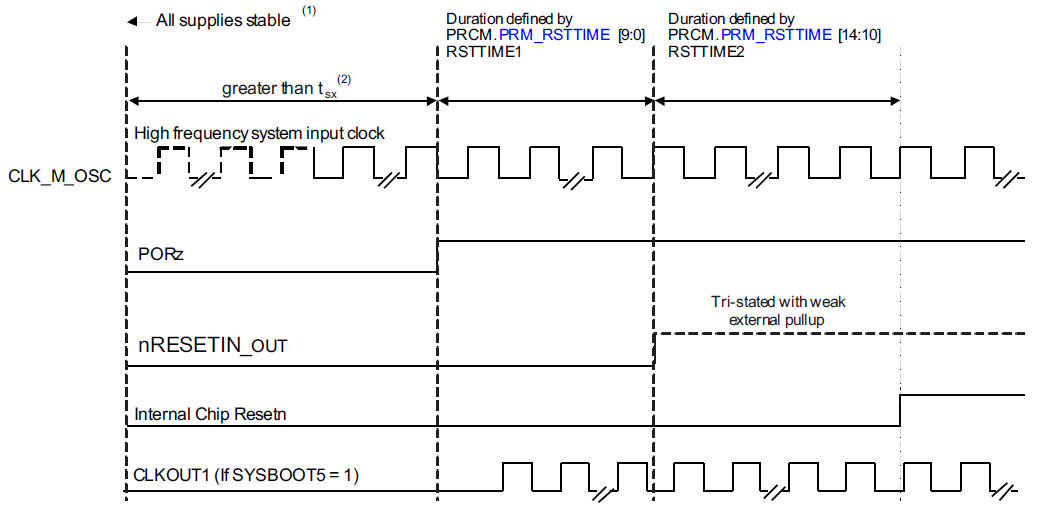
- После завершения сканирования eFuse перезагрузка хост-процессор и всех остальных периферийных устройств (периферийные устройства без локального процессора) будет отменена.

- **nRESETIN\_OUT** будет деактивирован по истечении времени, заданного параметром **PRM\_RSTTIME.RSTTIME1**.

- Как только хост-процессоры закончат загрузку, все оставшиеся периферийные устройства увидят отмену подтверждения сброса.

Обратите внимание, что все модули с локальными CPU будут иметь локальный сброс, установленный по умолчанию в **PORz**, и для возможности записи в соответствующие регистры в PRCM необходимо отменить подтверждение сброса хост-процессором.

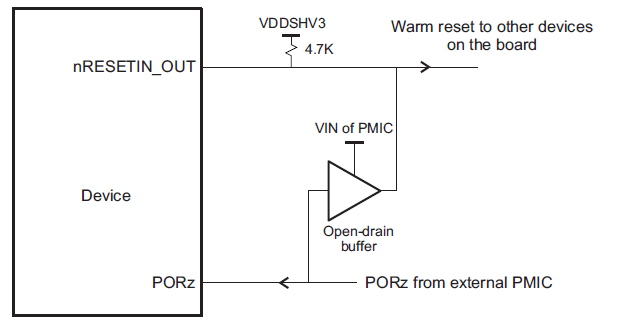
**Рисунок 8-23. PORz**



(1) **nRESETIN\_OUT** не определен (может быть либо установлен в низкий уровень, либо поднят на высокий уровень) до тех пор, пока все источники питания не будут полностью линейно изменены вверх. Для того чтобы nRESETIN\_OUT поддерживал действительное низкое состояние до тех пор, пока не будет линейно изменено питание, внешний буфер должен быть реализован, как показано на рисунке 8-24.

(2) Для получения информации о tsx см. процессоры AM335x Sitara (номер публикации SPRS717).

**Рисунок 8-24. Внешний буфер для nRESETIN\_OUT**



**8.1.7.3.3 Неверный сброс устройства**

Этот сброс устанавливается всякий раз, когда **DEVICE\_TYPE** кодирует неподдерживаемый тип устройства, например код для "плохого" устройства.

**8.1.7.3.4 ICEPICK Сброс при включении питания**

Внутренний модуль эмуляции ICE-Pick генерирует **ICEPICK\_POR\_RST**. Этот сброс используется только в режиме эмуляции. PRCM должен предоставить выходной порт **ACT\_LIKE\_SECURE**, который асинхронно устанавливается в высокий логический уровень при утверждении **ICEPICK\_POR\_RST**. Это состояние должно сохраняться до тех пор, пока происходит нормальный системный **POR**.

**8.1.7.3.5 Глобальный холодный сброс программного обеспечения (GLOBAL\_COLD\_SW\_RST)**

Источник для **GLOBAL\_COLD\_SW\_RST** создается внутри **PRM**. Активируется при настройке бит **PRM\_RSTCTRL.RST\_GLOBAL\_COLD\_SW** в карте памяти **PRM**. Этот бит самоочищается, т.е. он автоматически очищается аппаратно.

**8.1.7.4 Глобальный теплый сброс**

**8.1.7.4.1 Внешний теплый сброс**

nRESETIN\_OUT — это двунаправленный сигнал теплого сброса. В качестве входа обычно используется внешний источник как сброс устройства. Обратитесь к Таблице 8-24 для сводки различий между теплым сбросом и холодным сбросом. Вот некоторые из этих различий:

- Теплый сброс можно заблокировать для переключателя **EMAC** и его источника эталонного тактового сигнала PLL с помощью регистра **RESET\_ISO** в модуле управления.

- Теплый сброс предполагает, что тактовые частоты и питание микросхемы стабильны с момента подтверждения до снятия подтверждения, тогда как во время холодного сброса источники питания могут стать стабильными во время подтверждения.

- Некоторые регистры модуля PRCM и управления нечувствительны к горячему сбросу и сохраняют свои значения во время теплого сброса.

- Выводы SYSBOOT не фиксируются при теплом сбросе. Устройство загрузится со значениями SYSBOOT от предыдущего холодного сброса.

- Теплый сброс не влияет на большую часть логики подсистемы отладки. Это позволяет поддерживать любую отладку сеансов во время теплого сброса.

- На PLL не влияет теплый сброс

В качестве выхода сброса можно использовать **nRESETIN\_OUT** для сброса внешних устройств. **nRESET\_OUT** будет иметь низкий уровень во время холодноо сброса или внутренне сгенерированный теплый сброс. После завершения холодного или теплого сброса **nRESETIN\_OUT** будет продолжать поддерживать низкий уровень в течение периода, определенного **PRM\_RSTTIME.RSTTIME1. RSTTIME1** это таймер, который ведет обратный отсчет до нуля со скоростью, равной частоте входных тактовых импульсов **CLK\_M\_OSC**. Это позволяет удерживать внешние устройства в состоянии сброса в течение некоторого времени после выхода AM335x из состояния сброса.

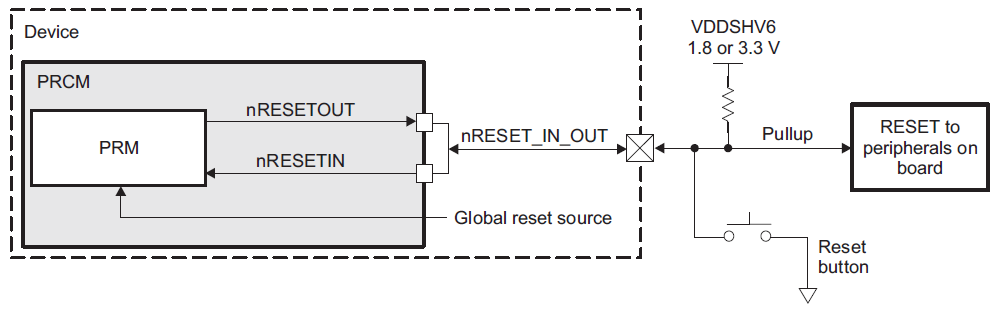
Следует соблюдать осторожность при реализации **nRESETIN\_OUT** в качестве двунаправленного сигнала сброса. Потому что из короткого максимального времени, разрешенного с использованием **RSTTIME1**, он не обеспечивает адекватного времени устранения дребезга для схема внешней тактовой кнопки. Процессор потенциально может начать работать, когда внешние компоненты все еще находятся в сбросе. Рекомендуется использовать этот сигнал только как вход (не подключать к другим устройствам в качестве сброса) для реализации схемы кнопки сброса на AM335x или выхода только для возможности сброса других устройств после завершения сброса AM335x.

**8.1.7.4.1.1 Ввод/вывод горячего сброса (nRESETIN\_OUT)**

Любой глобальный источник сброса (внутренний или внешний) приводит к тому, что **nRESETIN\_OUT** управляется и поддерживается в границах устройства в течение как минимум времени, настроенного в параметре **PRCM.PRM\_RSTTIME**. в битовом поле **RSTTIME1**. Это гарантирует, что устройство и связанные с ним периферийные устройства сбрасываются вместе. Выходной буфер **nRESETIN\_OUT** настроен как открытый сток; следовательно, требуется внешний подтягивающий резистор.

После снятия подтверждения двунаправленный контакт **nRESETIN\_OUT** переходит в третье состояние, чтобы разрешить подтверждение из выключенного состояния (внешнее подтверждение).

**Рисунок 8-25. Внешний сброс системы**



***Примечание***. Рекомендуется реализовать теплый сброс только как вход (например, кнопка) или только как выход (для

сброса внешних периферийных устройства), а не сразу оба.

Устройство будет иметь один контакт **nRESETIN\_OUT**, который отражает состояние сброса микросхемы. Этот вывод всегда будет устанавливаться асинхронно, когда происходит сброс любого сторожевого таймера микросхемы, если любое из следующих событий сброса имеет место:

- POR (только внутренняя растянутая часть события сброса после фиксации начальной загрузки(boot))

- Внешний горячий сброс (вывод **nRESETIN\_OUT**, только внутренняя растянутая часть события сброса после блокировки bootstrap)

- Сброс эмуляции (холодный или теплый от **ICEPICK**)

- Запрос сброса

- Холодный/горячий сброс программного обеспечения

Этот вывод будет оставаться активным до тех пор, пока PRCM поддерживает сброс на хост-процессор.

***Примечание***. TRST не вызывает утверждение **RSTOUTn**.

**8.1.7.4.1.2 Последовательность горячего сброса**

1. Вывод **nRESETIN\_OUT** на чипе устанавливается (переходит в низкий уровень). ПРИМЕЧАНИЕ. Для выполнения последовательности теплого сброса, как описано, ожидается что пин **PORz** всегда неактивен, иначе вы получите **PORz** функции, описанные в предыдущем разделе.

2. Все выводы IO (кроме теста и эмуляции) немедленно перейдут в тройное (Hiz) состояние.

3. Тактовые импульсы микросхемы не затрагиваются, так как и PLL, и делители не работают.

4. **nRESETIN\_OUT** сбрасывается после 30 циклов.

5. **PRCM** отменяет сброс для главного процессора и всех других периферийных устройств без локальных процессоров.

6. Обратите внимание, что все IP с локальными ЦП будут иметь локальный сброс, установленный по умолчанию при горячем сбросе и от хост-процессора потребуется подтверждение отмены сброса для возможнсти записи в соответствующие регистры PRCM.

На рис. 8-26 показана форма сигнала **nRESETIN\_OUT** при использовании **RESETIN\_OUT** в качестве источника теплого сброса. Пока замкнут внешний переключатель теплого сброса, и устройство, и микросхема будут работать на нуле.

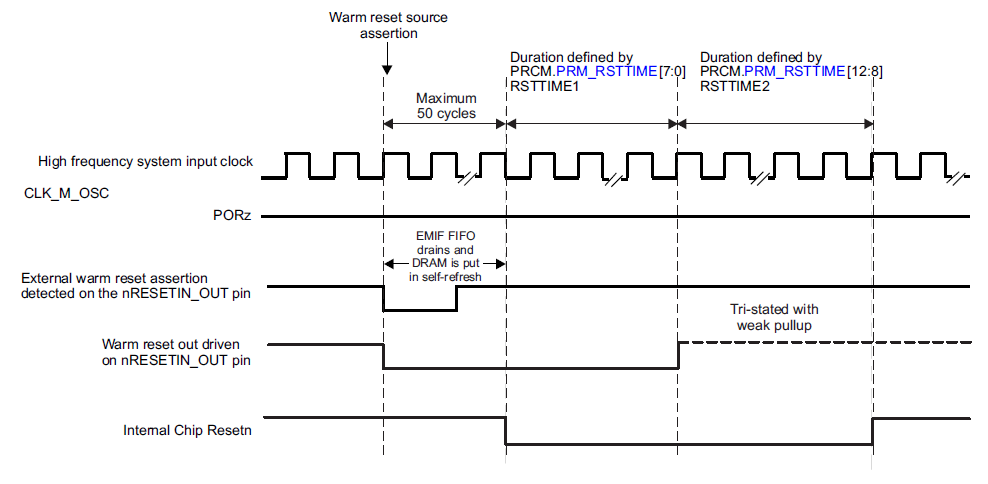
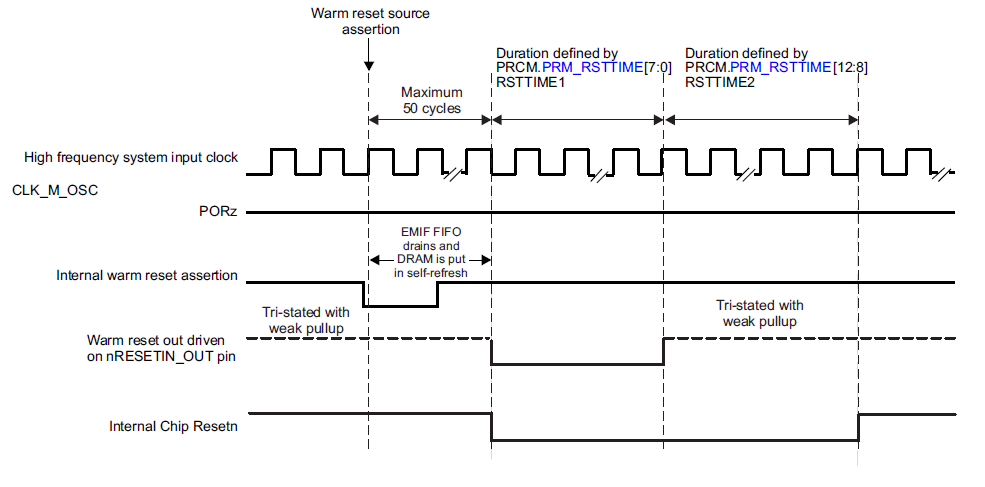
**Рисунок 8-26. Последовательность теплого сброса (внешний источник теплого сброса)** 

Рисунок 8-27 показывает форму волны **nRESETIN\_OUT**, когда любой из источников теплого сброса захватывается, за исключением использования самого **nRESETIN\_OUT** , в качестве источника горячего сброса.

**Рисунок 8-27. Последовательность теплого сброса (внутренний источник теплого сброса)**



**8.1.7.4.2 Сторожевой таймер**

На устройстве имеется один сторожевой таймер. В регистровом пространстве модуля управления есть регистр уровня микросхемы который содержит липкий бит (очищаемый только **PORz**) для каждого сброса. Сброс не блокируется. Для большей информации см. раздел **20.4 Watchdog**.

**8.1.7.4.3 Сброс программного обеспечения Global Warm (GLOBAL\_SW\_WARM\_RST)**

**GLOBAL\_WARM\_SW\_RST** генерируется внутри PRCM. Активация срабатывает после установки бита регистра с отображением в памяти, **PRM\_RSTCTRL.** **RST\_GLOBAL\_WARM\_SW**. Этот бит самоочищаемый, и автоматически очищаемый аппаратно.

**8.1.7.4.4 Тестовый сброс (TRSTz)**

Этот сброс запускается с вывода **TRSTz** на интерфейсе JTAG. Это неблокируемый сброс, и он сбрасывает тестовую логику и логику эмуляции.

***ПРИМЕЧАНИЕ***. Утверждением сброса PORz должно вызывать сброс всего устройства, включая всю логику тестирования и эмуляции независимо от состояния TRSTz. Таким образом, утверждение PORz приведет к полному сбросу устройства, даже если вывод TRSTz постоянно находится в высоком состоянии, и во время изменения питания не требуется специального переключения вывода TRSTz для достижения полного сброса POR на устройстве. Кроме того, допустимо постоянный подтяг ввода TRSTz к низкому уровню во время нормального функционального использования устройства в конечной системе, чтобы гарантировать, что вся тестовая и логика эмуляции остаются в сбросе.

**8.1.7.5 Характеристики сброса**

В следующей таблице показаны характеристики каждого источника сброса.