**20.1 DMTimer**

**20.1.1 Введение**

**20.1.1.1 Обзор**

Модуль таймера содержит свободно работающий восходящий счетчик с возможностью автоматической перезагрузки при переполнении. Счетчик таймера можно читать и записывать в режиме реального времени (во время подсчета). Модуль таймера включает логику сравнения , позволяющую выполнить прерывание при совпадении запрограммированного значения со значением счетчика.

Специальный выходной сигнал может быть импульсным или переключаться на событие переполнения и совпадения. Этот вывод может выводить

триггерные сигналы с определенным временным шаблоном или ШИМ (широтно-импульсная модуляция). Специальный выходной сигнал может

использоваться для общего назначения **PORGPOCFG** (непосредственно управляется битом 14 регистра **TCLR**). Специальный входной сигнал используется для автоматического  запуска выборки счетчика таймера и события прерывания,  на программируемом входе  сигнала переходного типа. Программируемый делитель тактовой частоты (пределитель) позволяет уменьшить входную тактовую частоту таймера. Все внутренние источники прерываний таймера объединены в одну линию прерывания модуля и одну линию пробуждения.  Каждый внутренний источник прерывания может быть независимо включен/выключен.

Этот модуль управляется через периферийную шину **OCP**.

Поскольку внутри этого модуля имеются два домена тактирования, ресинхронизация выполняется по специальной логике между доменом тактирования **OCP** и доменом тактирования Timer'а. После сброса логика синхронизации позволяет использование всех соотношений между тактированием **OCP** и тактированием Timer'а. Недостаток этого режима в том, что путь полной ресинхронизация используется с влиянием задержки доступа на производительность с точки зрения тактовых циклов **OCP**. В порядке снижения влияния  задержки доступа к модулю, а также в условиях ограниченных тактовых частот, опубликовать запись можно установив бит **POSTED** в регистре управления системой (**TSCR**). В этом режиме, режим публикации записи включен, что означает, что команда записи **OCP** предоставляется до  завершения процесса записи  в домене такирования Timer'а.  Этот режим позволяет программному обеспечению выполнять одновременную запись в двойном режиме регистров таймера и наблюдать за завершением процесса записи (синхронизацией) на программном уровне путем чтения независимых битов состояния записи в регистре состояния записи (**TWPS**).

**20.1.1.2 Особенности**

Таймер состоит из следующих функций:

- Таймер счетчика с режимами сравнения и захвата

- Режим автоперезагрузки

- Режим старт-стоп

- Программируемый делитель источника тактирования

- 16-32-битная адресация

- Чтение/запись регистров «на лету»

- Прерывания, генерируемые при переполнении, сравнении и захвате

- Разрешение прерываний

- Включение побудки(только для **Timer0**)

- Режим записи записей

- Специальный входной триггер для режима захвата и специальный выходной триггер/ШИМ-сигнал

- Специальный выходной сигнал общего назначения **PORGPOCFG**

- Совместимость с интерфейсом **OCP**

Разрешение таймера и период прерывания зависят от выбранного входного тактового сигнала и предварительного делителя тактового сигнала.

Примеры разрешения для общих значений тактирования показаны в таблице 20-1.

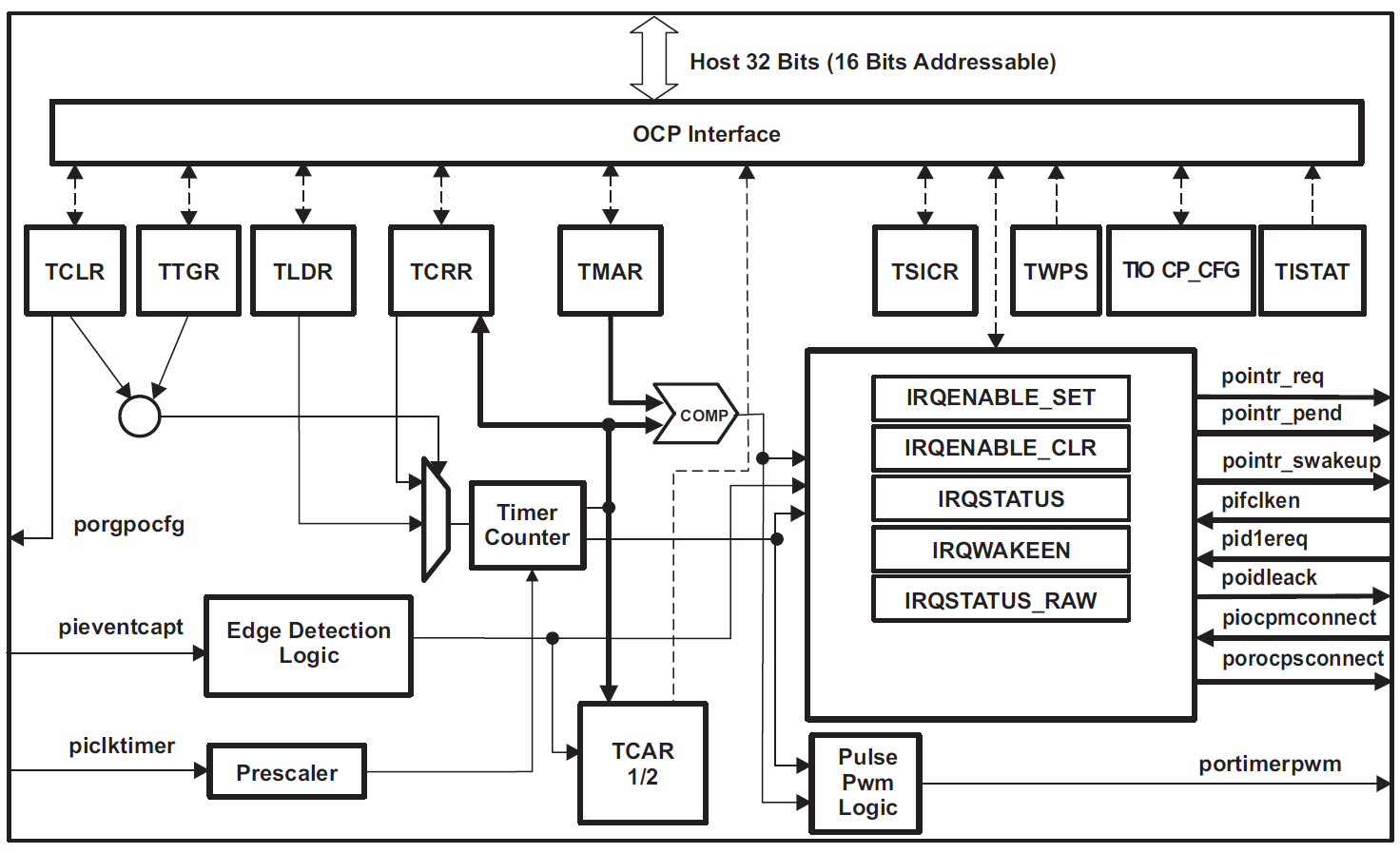
Таблица 20-1. Разрешение таймера и максимальный диапазон

|  |  |  |  |
| --- | --- | --- | --- |
| **Clock** | **Prescaler** | **Resolution** | **Interrupt Period Range** |
| 32.768 KHz | 1 (min) | 31.25 us | 31.25 us to ~36h 35m |
| 256 (max) | 8 ms | 8 ms to ~391d 22h 48m |
| 25 MHz | 1 (min) | 40 ns | 40 ns to ~171.8s |
| 256 (max) | 10.24 us | ~20.5 us to ~24h 32m |

**20.1.1.3 Функциональная блок-схема**

На рис. 20-1 показана блок-схема таймера.

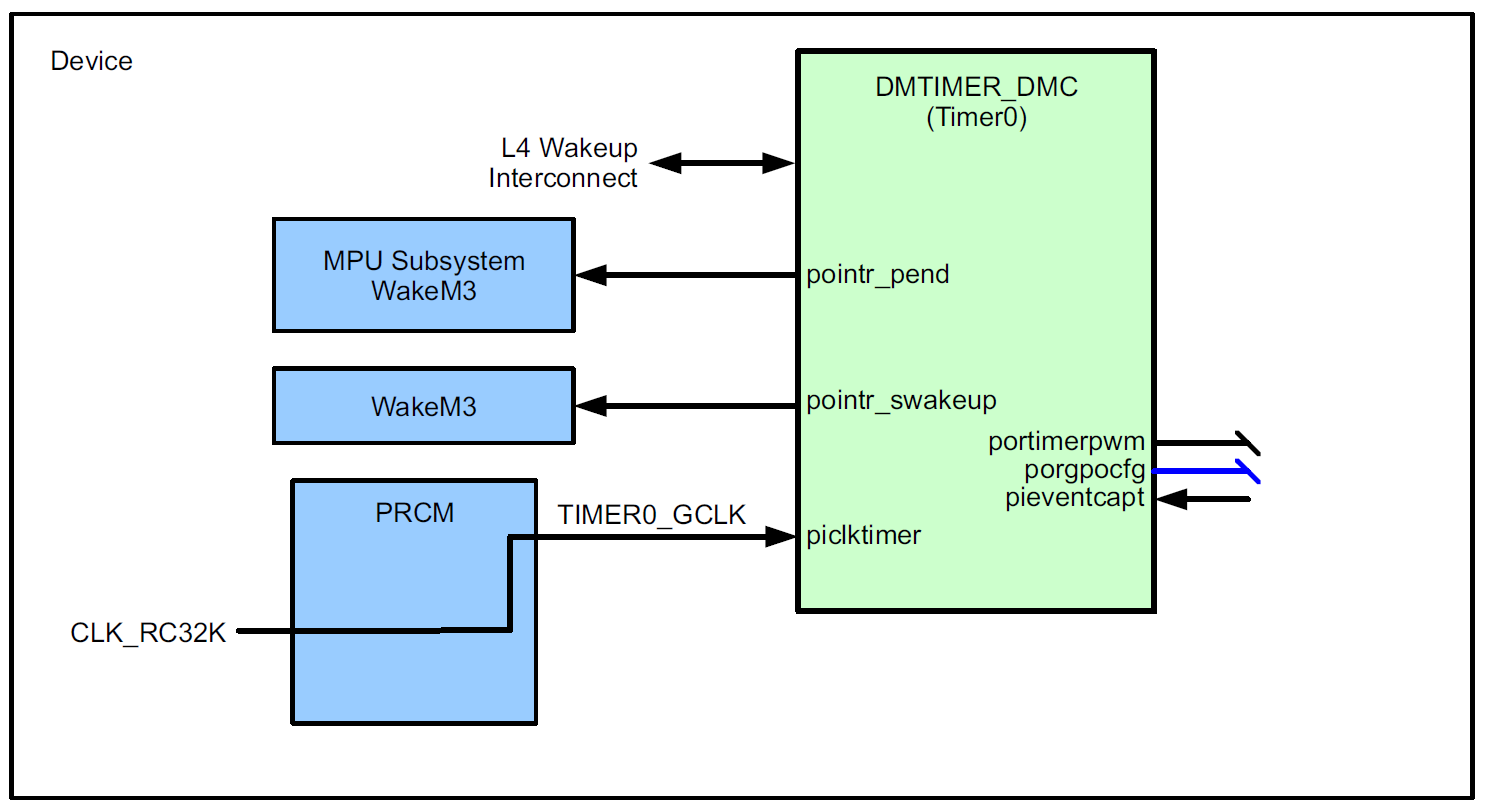
Рис. 20-1. Блок-схема таймера



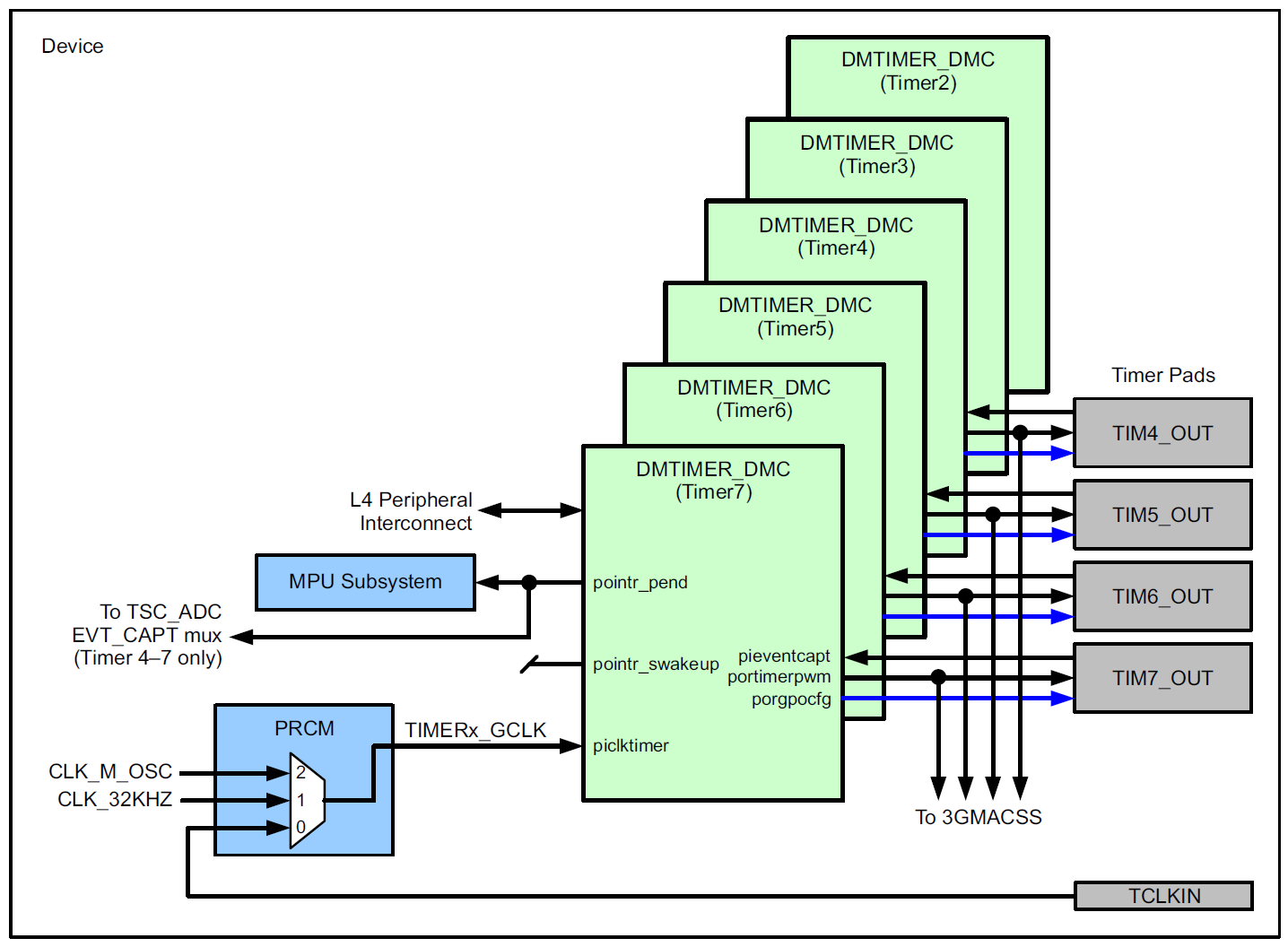
**20.1.2 Интеграция**

Интеграция **Timer0** и **Timer2-7** показана на рисунках 20-2 и 20-3.

**Рис. 20-2. Timer0 интеграция**



**Рисунок 20-3. Timer2-7 интеграция**

****

**20.1.2.1 Атрибуты подключения таймера**

**Таблица 20-2. Атрибуты подключения по таймеру [0]**

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| Power domain | Wakeup domain |
| Clock Domain | PD\_WKUP\_L4\_WKUP\_GCLK (Interface/OCP)  PD\_WKUP\_TIMER0\_GCLK (Func) |
| Reset Signals | WKUP\_DOM\_RST\_N |
| Idle/Wakeup Signals | Idle  Slave Wakeup |
| Interrupt Requests | 1 to MPU Subsystem (TINT0), 1 to WakeM3 |
| DMA Requests | None |
| Physical Address | L4 Wakeup slave port |

**Таблица 20-3. Параметры подключения по таймеру [2-7]**

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| Power Domain | Peripheral Domain |
| Clock Domain | PD\_PER\_L4LS\_GCLK (OCP)  Functional Clocks:  PD\_PER\_TIMER2\_GCLK (Timer 2)  PD\_PER\_TIMER3\_GCLK (Timer 3)  PD\_PER\_TIMER4\_GCLK (Timer 4)  PD\_PER\_TIMER5\_GCLK (Timer 5)  PD\_PER\_TIMER6\_GCLK (Timer 6)  PD\_PER\_TIMER7\_GCLK (Timer 7) |
| Reset Signals | PER\_DOM\_RST\_N |
| Idle/Wakeup Signals | Smart Idle (No wakeup capabilities) |
| Interrupt Requests | 1 per timer module to MPU Subsystem (TINT2 - TINT7)  Also to TSC\_ADC event capture mux for Timer 4–Timer7 |
| DMA Requests | Interrupt requests are redirected as DMA requests: 1 per  instance (TINTx) |
| Physical Address | L4 Peripheral slave port |

**20.1.2.2 Таймер и управление сбросом**

Каждое функциональное тактированин **DTimer**[2–7] выбирается в **PRCM** с использованием связанного

регистра **CLKSEL\_TIMERx\_CLK** из трех возможных источников:

- Системные такты(**CLK\_M\_OSC**)

- **PER PLL** генерирует тактовую частоту 32,768 кГц (**CLK\_32KHZ**).

- Внешние тактовые импульсы **TCLKIN**.

Функциональное тактирование **DMTimer** 0 завязано на использование внутреннего тактирования RC 32 кГц (**CLK\_RC32K**).

**20.1.2.3 Сигналы тактирования таймера**

**Таблица 20-4. Синхросигналы таймера**

|  |  |  |  |
| --- | --- | --- | --- |
| **Clock Signal** | **Max Freq** | **Reference / Source** | **Comments** |
| **Timer[0] Clock Signals** | | | |
| PICLKOCP  Interface clock | 100 MHz | CORE\_CLKOUTM4 / 2 | pd\_wkup\_l4\_wkup\_gclk  from PRCM |
| PICLKTIMER  Functional clock | 26 MHz(1) | CLK\_RC32K  CLK\_M\_OSC | pd\_wkup\_timer0\_gclk  from PRCM |
| **Timer[2–7] Clock Signals** | | | |
| PICLKOCP  Interface clock | 100 MHz | CORE\_CLKOUTM4 / 2 | pd\_per\_l4ls\_gclk  from PRCM |
| PICLKTIMER  Functional clock | 26 MHz(1) | CLK\_M\_OSC  CLK\_32KHZ  (PER\_CLKOUTM2 / 5859.375)  TCLKIN | pd\_per\_timer2\_gclk  pd\_per\_timer3\_gclk  pd\_per\_timer4\_gclk  pd\_per\_timer5\_gclk  pd\_per\_timer6\_gclk  pd\_per\_timer7\_gclk  from PRCM |

**20.1.2.4 Список выводов таймера**

Входной сигнал таймера **PIEVENTCAPT** и выходные сигналы **PORTIMERPWM** мультиплексируются в одину ножку **TIMER** настроенную на вход/выход.

Направление пина (и, следовательно, функция контакта) управляется из модуля **DMTimer** с помощью сигнала **PORGPOCFG** в качестве разрешения выхода.

**Таблица 20-5. Список выводов таймера**

|  |  |  |
| --- | --- | --- |
| **Pin** | **Type** | **Description** |
| TCLKIN | I | External timer clock source |
| TIMER4 | I/O | Timer 4 trigger input or PWM output |
| TIMER5 | I/O | Timer 5 trigger input or PWM output |
| TIMER6 | I/O | Timer 6 trigger input or PWM output |
| TIMER7 | I/O | Timer 7 trigger input or PWM output |

**20.1.3 Функциональное описание**

Таймер общего назначения представляет собой восходящий счетчик. Он поддерживает 3 функциональных режима:

• Режим таймера

• Режим захвата

• Режим сравнения

По умолчанию после сброса ядра режимы захвата и сравнения отключены.

**20.1.3.1 Функциональность режима таймера**

Таймер представляет собой восходящий счетчик, который можно запускать и останавливать в любое время с помощью регистра управления таймером

(бит **TCLR ST**). Регистр таймера-счетчика (**TCRR**) может быть загружен при остановке или на лету(при подсчете). **TCRR** может быть загружен непосредственно при записи в **TCRR** с нового значения таймера. **TCRR** также может быть загружен значением, хранящимся в регистре загрузки таймера (**TLDR**), с помощи доступа для записи через регистр триггера (**TTGR**) . Загрузка **TCRR** выполняется независимо от значения, записанного в **TTGR**. Значение регистра таймера счетчика (**TCRR**) может быть прочитано при остановке или захвате на лету с помощью доступа на чтение через регистр **TCRR**. Таймер останавливается, и значение счетчика сбрасывается до «**0**», когда подтверждается сброс модуля. Таймер сохраняется в состоянии остановки после сброса. Когда таймер остановлен, **TCRR** замораживается. Таймер может быть перезапущен из замороженного значения, если **TCRR** не был перезагружен с новым значением.

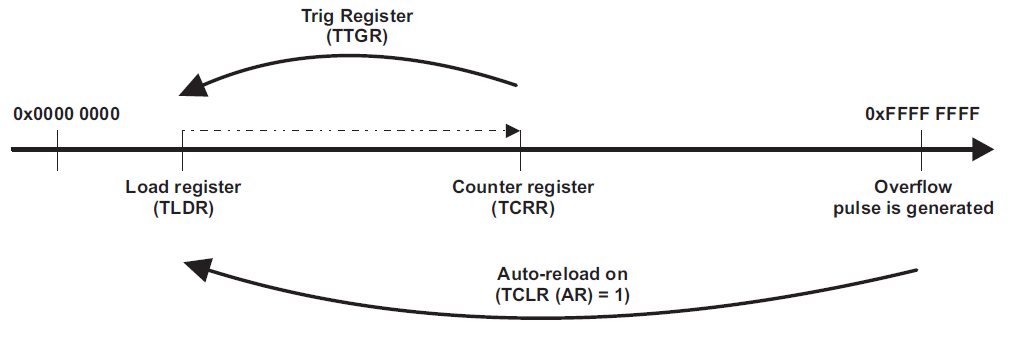
В однократном режиме (бит **TCLR AR = 0**) счетчик останавливается после переполнения счета (значение счетчика остается на нуле).

Когда режим автоматической перезагрузки включен (бит **TCLR AR = 1**), **TCRR** перезагружается с загрузкой регистра значения таймера (**TLDR**) после переполнения счетчика.

Не рекомендуется помещать значение переполнения (**0xFFFF FFFF**) в **TLDR**, поскольку это может привести к нежелательным результатам.

Прерывание может быть выдано при переполнении, если в прерывании таймера установлен бит разрешения прерывания от переполнения в регистре разрешения прерывания **IER** (бит **IRQENABLE\_SET** **OVF\_IT\_FLAG** = **1**). Специальный выходной контакт (**PORTIMERPWM**) программируется через **TCLR** (биты **TRG** и **PT**) для генерации одного положительного импульса (задается предделителем) или инвертирования текущее значения (режим переключения) при переполнении.

**Рис. 20-4. Значение синхронизации TCRR**



**20.1.3.2 Функциональность режима захвата**

Значение таймера в **TCRR** может быть захвачено и сохранено в функции **TCAR1** или **TCAR2** спосбом, который выбран в **TCLR** через поле **CAPT\_MODE** при обнаружении перехода на входном контакте модуля (**PIEVENTCAPT**). Схема обнаружения фронта отслеживает переходы на входном контакте (**PIEVENTCAPT**).

Восходящий переход, нисходящий переход или оба могут быть выбраны в **TCLR** (бит **TCM**) для запуска захвата счетчика таймера. Модуль устанавливает **IRQSTATUS** (бит **TCAR\_IT\_FLAG**) при обнаружении активного перехода и при этом значение счетчика **TCRR** сохраняется в одном из регистров захвата таймера **TCAR1** или **TCAR2** следующим образом:

- Если поле **CAPT\_MODE TCLR** равно **0**, то при первом активированном событии захвата значение счетчика регистра сохраняется в регистре **TCAR1**, а все последующие события игнорируются (нет обновления в **TCAR1** и нет срабатывания прерывания) до тех пор, пока логика обнаружения не будет сброшена или регистр статуса прерывания не будет очищен в **TCAR** , записью в него **1**.

- Если поле **CAPT\_MODE** **TCLR** равно **1**, то при первом разрешенном захваченном событии значение счетчика сохраняется в регистре **TCAR1** и при втором включенном событии захвата значение регистра-счетчика сохраняется в регистре **TCAR2**. Все остальные события игнорируются (отсутствие обновления на **TCAR1/2** и срабатывание прерывания) пока логика обнаружения не будет сброшена или регистр состояния прерывания не будет очищен в  **TCAR**, записью в него 1. Этот механизм полезен для расчета периода тактирования, если эти тактовые импульсы подключены к входному контакту **PIEVENTCAPT**.

Логика обнаружения фронта сбрасывается (разрешается новый захват), когда обслуживается активное прерывание захвата.

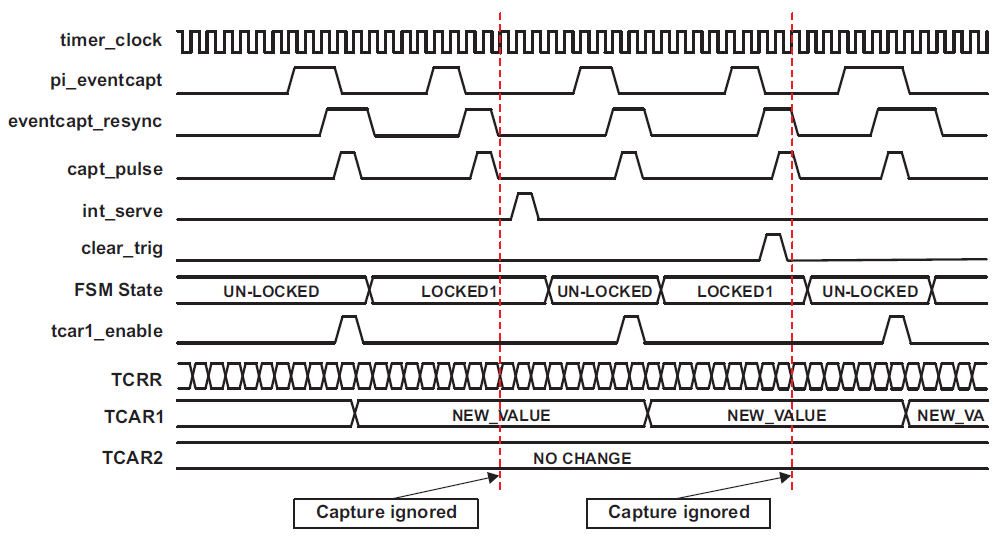
Бит **TCAR\_IT\_FLAG** в **IRQSTATUS** (ранее 1) очищается. Функциональное тактирование таймера (вход в **prescaler**) используется для выборки входного вывода (**PIEVENTCAPT**). Отрицательные или положительные импульсы могут быть обнаружены когда время импульса превышает период функционального тактирования . Прерывание может быть выдано при обнаружении перехода если бит разрешения прерывания захвата установлен в регистре разрешения прерывания таймера **IRQENABLE\_SET** (бит **TCAR\_IT\_FLAG**).

На Рисунке 20-5 значение **TCM** равно **01**, а **CAPT\_MODE** равно **0** — будет работать только нарастающий фронт сигнала захвата **PIEVENTCAPT** в **TCAR**, и только **TCAR1** будет обновляться.

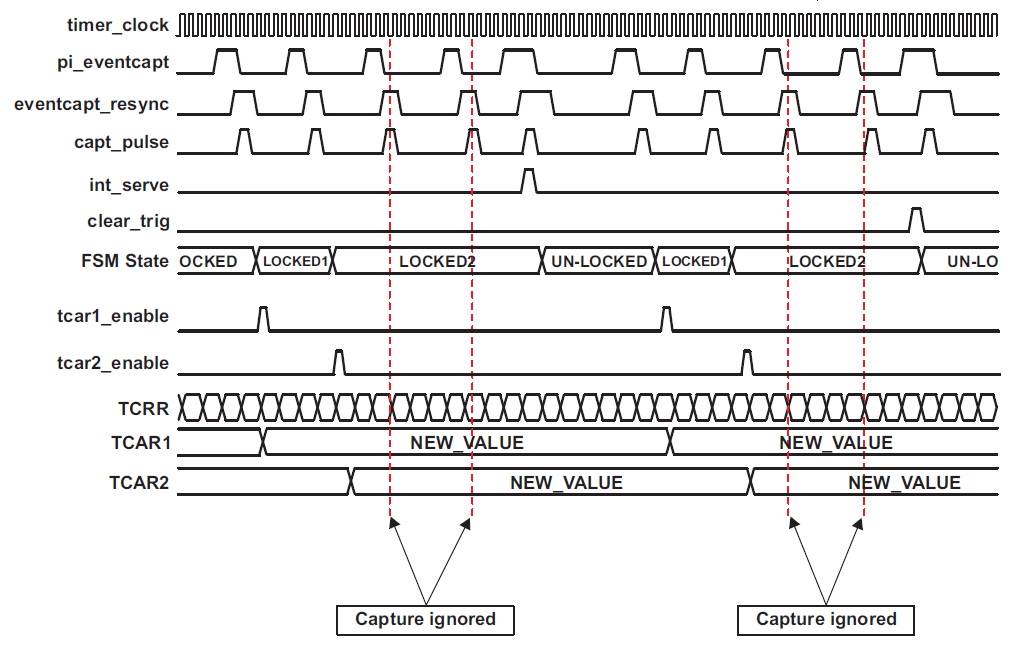
На Рисунке 20-6 значение **TCM** равно **01**, а **CAPT\_MODE** равно **1** — только нарастающий фронт сигнала **PIEVENTCAPT** будет

инициировать захват в TCAR1 при первом включенном событии, а **TCAR2** обновится при втором включенном событии.

**Рис. 20-5. Пример волны захвата для CAPT\_MODE = 0**



**Рис. 20-6. Пример волны захвата для CAPT\_MODE = 1**

****

**20.1.3.3 Функциональность режима сравнения**

Когда срвненине разрешено в  **TCLR**  (бит **CE**) установлено значение 1, значение таймера (**TCRR**) постоянно сравнивается с

значение, хранящееся в регистре соответствия таймера (**TMAR**). Значение **TMAR** может быть загружено в любое время (таймер считает или

остановлен). Когда значения **TCRR** и **TMAR** совпадают, прерывание может быть выдано, если установлен бит в **IRQENABLE\_SET**(бит **MAT\_EN\_FLAG**) . Правильное использование  - записать значение сравнения в регистр **TMAR** перед установкой **TCLR** (бит CE), чтобы избежать любых нежелательных прерываний из-за эффекта согласования значений сброса.

Специальный выходной контакт (**PORTIMERPWM**) может быть запрограммирован через **TCLR** (биты **TRG** и **PT**) на генерацию одного положительного импульса (длительностью одного такта ТАЙМЕРА) или инвертировать текущее значение (режим переключения), когда происходит переполнение и совпадение.

**20.1.3.4 Функциональность предварительного делителя**

Счетчик предварительного делителя можно использовать для деления входной тактовой частоты счетчика таймера. Предделитель включен, когда бит **5 TCLR** установлен (**PRE**). Значение коэффициента деления **2n** (**PTV**) можно настроить в регистре **TCLR**.Счетчик предварительного делителя сбрасывается, когда счетчик таймера останавливается или перезагружается на лету.

**Таблица 20-6. Функциональность прескалера**

|  |  |  |
| --- | --- | --- |
| **Contexts** | **Prescaler Counter** | **Timer Counter** |
| Overflow (when Auto-reload on) | Reset | TLDR |
| TCRR Write | Reset | TCRR |
| TTGR Write | Reset | TLDR |
| Stop | Reset | Frozen |

**20.1.3.5 Широтно-импульсная модуляция**

Таймер можно настроить для обеспечения программируемой широтно-импульсной модуляции (**PORTIMERPWM**) на выходе. Выходной контакт **PORTIMERPWM** можно настроить на переключение по определенному событию. Регистр **TCLR** (**TRG** бит) определяет, на какое значение регистра переключается вывод **PORTIMERPWM**. Либо переполнение, либо совпадение могут использоваться для переключения вывода **PORTIMERPWM** при возникновении условия сравнения.

В случае режима переполнения и совпадения событие совпадения будет игнорироваться с момента установки режима до тех пор, пока не произойдет первое событие переполнения (см. рис. 20-6).

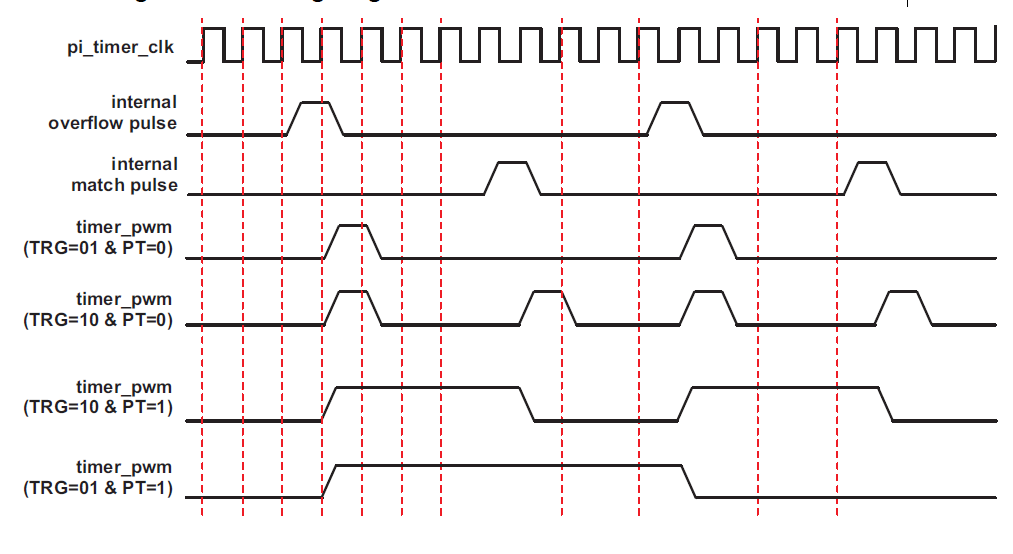
**TCLR** (бит **SCPWM**) можно запрограммировать для установки или сброса выходного сигнала **PORTIMERPWM**, в то время как счетчик остановлен или отключен только триггер. Это позволяет зафиксировать детерминированное состояние выходного вывода при остановке модуляции. Модуляция синхронно останавливается, когда бит TRG очищается и происходит переполнение.

На следующей временной диаграмме внутренний импульс переполнения устанавливается каждый раз когда (0xFFFF FFFFF — TLDR + 1) значение достигается, и внутренний импульс совпадения устанавливается, когда счетчик достигает значения регистра **TMAR**. В соответствии со значением запрограммированным в  **TCLR** (биты **TRG** и **PT**), таймер выдает импульс или ШИМ на выходе (**PORTIMERPWM**).

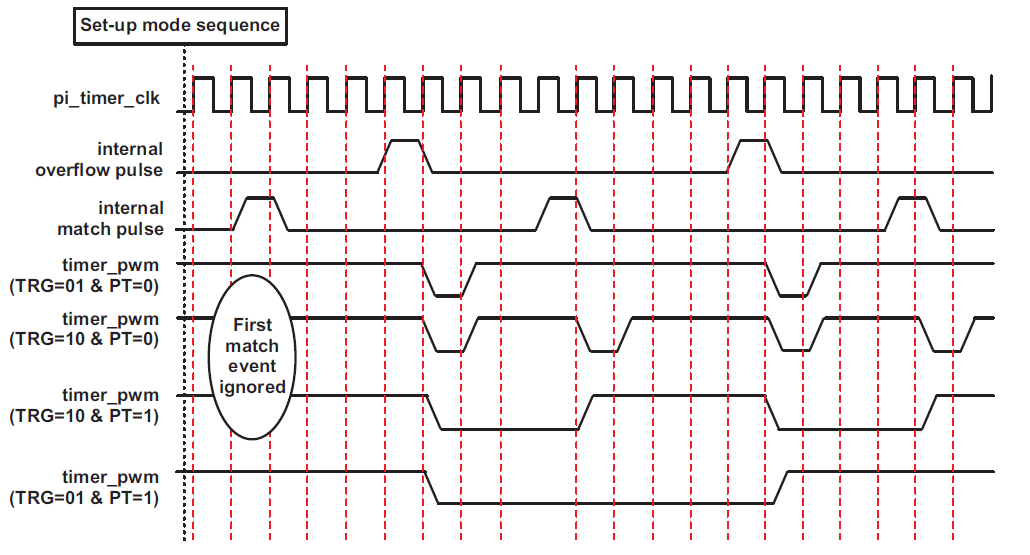
Регистры **TLDR** и **TMAR** должны сохранять значения меньше значения переполнения (0xFFFF FFFF) на минимум 2 единицы(0xFFFF FFFD). В случае, если триггерные события **PWM** являются одновременно переполнением и совпадением, разница между значениями хранятся в регистре **TMAR**, а значение в **TLDR** должно быть не менее 2 единиц. Когда используется событие совпадения, должен быть установлен режим сравнения **TCLR** (**CE**).

На рисунке 20-7 TCLR (бит **SCPWM**) сброшен в **0**. На рисунке 20-8 TCLR (бит **SCPWM**) установлен в **1**.

**Рис. 20-7. Временная диаграмма широтно-импульсной модуляции с SCPWM = 0**

****

**Рис. 20-8. Временная диаграмма широтно-импульсной модуляции с SCPWM = 1**

****

**20.1.3.6 Скорость счета таймера**

Счетчик таймера состоит из каскада предварительного делителя и счетчика таймера. Предварительный делитель синхронизируется с

тактированием таймера и действует как делитель частоты для каскада счетчика таймера. Соотношением можно управлять  через доступ к полю определения отношения регистра управления (**PTV** и **PRE** **TCLR**). См. Таблицу 20-7.

Скорость таймера определяется:

- Значением полей предделителя (**PRE** и **PTV** регистра **TCLR**)

- Значением, загруженным в регистр загрузки таймера (**TLDR**).

**Таблица 20-7. Значение тактовых коэффициентов прескалера**

|  |  |  |
| --- | --- | --- |
| **PRE** | **PTV** | **Divisor (PS)** |
| 0 | X | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 4 |
| 1 | 2 | 8 |
| 1 | 3 | 16 |
| 1 | 4 | 32 |
| 1 | 5 | 64 |
| 1 | 6 | 128 |
| 1 | 7 | 256 |

Уравнение расчета скорости таймера выглядит следующим образом:

(0xFFFF FFFF – TLDR + 1) × **период\_синхронизации\_таймера** × **делитель\_тактовой\_частоты** (**PS**)

С периодом тактирования таймер = 1/Тактовая частота и PS = 2e(PTV + 1).

В качестве примера, если мы рассмотрим входной сигнал таймера с частотой 32 кГц, с полем **PRE**, равным **0**, выходной сигнал таймера

период:

**Таблица 20-8. Значение и соответствующий период прерывания**

|  |  |
| --- | --- |
| **TLDR** | **Interrupt period** |
| 0000 0000h | 37 h |
| FFFF 0000h | 2 s |
| FFFF FFF0h | 500 us |
| FFFF FFFEh | 62.5 us |

**20.1.3.7 Таймер двойного режима в режиме эмуляции**

Чтобы настроить таймер на остановку во время событий приостановки эмуляции (например, точек останова отладчика),  настройте таймер и подсистему отладки:

1. Установите **TIOCP\_CFG.EMUFREE=0**. Это позволит сигналу **Suspend\_Control** от подсистемы отладки (глава 27) остановить и запустить таймер. Обратите внимание, что если **EMUFREE=1**, сигнал **Suspend\_Control** игнорируется, и таймер работает независимо от любого события приостановки отладки. Этот бит **EMUFREE** дает локальное управление с точки зрения модуля для блокировки сигнала приостановки, поступающего от  подсистемы отладки.

2. Установите соответствующий регистр **xxx\_Suspend\_Control** = **0x9**, как описано в Разделе 27.1.1.1, Остановка поддержки периферийных устройств во время Отладки. Выберите регистр, соответствующий периферийному устройству, которое вы хотите приостановить во время события приостановки.

**20.1.3.8 Доступ к регистрам**

Все регистры 32-битные, доступны через интерфейс **OCP** с 16-битным или 32-битным доступом OCP (чтение/запись). Обновление записи 32-битных регистров при 16-битном доступе должно быть сначала **LSB16**, а второе обращение записи должно быть **MSB16**. Для операции записи модуль позволяет пропустить доступ к **MSB**, если пользователю не нужно обновлять **16 MSB**  бит регистра, но только для регистров **OCP** (**TIDR**, **TIOCP\_CFG**, **IRQSTATUS\_RAW**, **IRQSTATUS**, **IRQENABLE\_SET**, **IRQENABLE\_CLR**, **IRQWAKEEN** и **TSICR**). Операция записи в любой функциональный регистр (**TCLR**, **TCRR**, **TLDR**, **TTGR** и **TMAR**) должна быть полной ( **MSB** должен быть записан, даже если данные **MSB** не используются).

**20.1.3.8.1 Программирование регистров таймера**

Хост осуществляет запись в регистры **TLDR**, **TCRR**, **TCLR**, **TIOCP\_CFG**, **IRQSTATUS**, **IRQENABLE\_SET**, **IRQENABLE\_CLR**,  **IRQWAKEEN**, **TTGR**, **TSICR** и **TMAR** синхронно с тактированием  **OCP**,с применеием протокола шины **OCP**.

**20.1.3.8.2 Чтение регистров таймера**

Регистр счетчика (**TCRR**) представляет собой 32-битные «атомарные данные», а 16-битный захват выполняется сначала на 16-битном **LSB**(младшие биты)чтобы сделать возможным атомарный захват LSB16 + MSB16. Атомный захват также выполняется для регистры TCAR1 и TCAR2 , так как они могут измениться из-за внутренних процессов.

**20.1.3.8.3 Генерация ошибок OCP**

Модуль таймера выдает сообщение об ошибке в следующих случаях:

**Ошибка при записи транзакций**

- Установите сигнал **PORSRESP** = **ERR** в том же цикле, что и **PORSCMDACCEPTED**.

- Использование кода **ERR** для **PORSRESP** на этапе реагирования.

**Ошибка при чтении транзакций**

- Установите сигнал **PORSRESP** = **ERR** в том же цикле, что и **PORSCMDACCEPTED**.

- Использование кода **ERR** для **PORSRESP** на этапе реагирования. **PORSDATA** в этом случае недействительна.

**Таблица 20-9. Отчеты об ошибках OCP**

|  |  |
| --- | --- |
| **Error Type** | **Response: SRESP = ERR** |
| Неподдерживаемая команда PIOCPMCMD | Yes |
| Ошибка адреса: чтение или запись на несуществующий внутренний адрес | No |
| Чтение в регистры только для записи и запись в регистры только для чтения | No |
| Невыровненный адрес (PIOCPMADDR ≠ 00) при выполнении чтении/записи | Yes |
| Неподдерживаемый PIOCPMBYTEEN в транзакции чтения/записи | Yes |

**ПРИМЕЧАНИЕ**. Разрешение байта «0000» — это поддерживаемое разрешение байта.

**20.1.3.9 Выбор режима публикации**

Выбор между двумя режимами синхронизации будет сделан с учетом соотношения частот и периодов простоя, которые могут поддерживаться системой, не влияя на общую производительность. Выбор режима публикации применяется только к функциональным регистрам, которые требуют синхронизации по/от домена синхронизации таймера. Для операции записи регистрами, затрагиваемыми этим выбором отправленных/не отправленных сообщений являются: **TCLR**, **TLDR**, **TCRR**, **TTGR** и **TMAR**. Для операции чтения регистрами, затрагиваемыми этим выбором отправленный/не отправленный являются: **TCRR**, **TCAR1** и **TCAR2**.

Синхронные регистры домена тактирования OCP **TIDR**, **TIOCP\_CFG**, **TISTAT**, **IRQSTATUS**, **IRQSTATUS\_RAW**, **IRQENABLE\_SET**, **IRQENABLE\_CLR**, **IRQWAKEEN**, **TWPS** и **TSICR** не зависят от выбора режима публикации/не публикации; операция записи/чтения эффективна и подтверждается(команда принята) после одного такта **OCP** с момента утверждения команды.

Режим без сообщений можно использовать, когда частота (таймера) < частоты**(OCP)/4**, но рекомендуется использовать режим с сообщениями, потому что режим без публикации задержит «команду принять», и задержка транзакции будет такой же  как описано в следующих главах. Режим публикации предлагает уменьшение задержки интерфейса **OCP**, когда частота(таймера) < частоты**(OCP)/4**. Обратитесь к эррате AM335x для рекомендаций, связанных с опубликованным режимом.

**20.1.3.10 Доступ к записи регистров**

**20.1.3.10.1 Запись с публикацией**

Этот режим можно использовать только в том случае, если рабочий диапазон частот равен частоте (таймера) < частоты**(OCP)/4**.

Этот режим используется, если в регистре управления таймером **TSICR** (бит POSTED) установлен в **1**.

В этом режиме используется схема отложенной записи для обновления любого внутреннего регистра. Транзакция записи

немедленно подтверждается на интерфейсе **OCP**, хотя фактическая операция записи произойдет позже, из-за ресинхронизации в домене тактирования таймера. Преимущество этого заключается в том, что система межсоединений или ЦП , запросивший транзакцию записи, не простаивают. Для каждого регистра присутствует бит состояния, который устанавливается при условии, что  есть отложенный доступ для записи в этот регистр.

В этом режиме необходимо, чтобы ЦП проверял бит состояния перед любым доступом для записи. В случае, если запись

применяется к регистру с ожиданием предыдущего доступа, предыдущий доступ отбрасывается без уведомления

(это также может привести к неожиданным результатам).

На каждый регистр приходится один бит состояния, доступный в регистре состояния записи таймера(**TWPS**). Когда модуль таймера работает в этом режиме, происходит автоматическая выборка текущего значения счетчика таймера, в регистре захвата синхронизированном-OCP. Следовательно, любой доступ для чтения к регистру счетчика таймера не

способен добавить задержку повторной синхронизации; текущее значение всегда доступно.

Регистр, считанный после регистра с публикацией(write posted) (в том же регистре), не гарантирует чтение предыдущей записи, если процесс публикации записи не завершен. Следует использовать программную синхронизацию, чтобы избежать не когерентного чтения.

Недостаток этого механизма автоматического обновления заключается в том, что он предполагает связь между

частотой интерфейса OCP и функциональной частотой таймера.

Этот период публикации определяется как интервал между отправленным запросом на публикацию для записи и сбросом

бита публикации в регистре **TWPS** и может быть определен количественно:

**T (reset posted max.) = 3 OCP clock + 2.5 TIMER clock**

Время завершения записи:

**T (write accomplish) = 1 OCP clock + 2.5 TIMER clock**

**20.1.3.10.2 Запись без публикации**

Этот режим работает независимо от соотношения между частотой интерфейса OCP и функциональной тактовой частототой. Рекомендуемый функциональный диапазон частот: частота (таймера) >= частота **(OCP)/4**.

Этот режим используется, если **TSICR** (бит **POSTED**) сброшен в 0 в регистре управления таймером.

В этом режиме используется схема без публикации для обновления любого внутреннего регистра. Это означает, что транзакция записи не будет подтверждена на интерфейсе OCP, пока не произойдет эффективная операция записи, после

повторной синхронизации в домене часов таймера. Недостатком является то, что как система межсоединений, так и

ЦП в этот период останавливаются.

- Задержка обслуживания прерывания увеличивается, так как система межсоединений и ЦП останавливаются.

- Логика межсоединений, включая логику тайм-аута для обнаружения ошибочных транзакций, может генерировать

нежелательное событие прерывания работы системы.

Период простоя определяется как интервал между запросом доступа на запись без публикации и повышением команда принимает сигнал и может быть определена количественно:

T (stall max.) = 3 OCP clock + 2.5 TIMER clock

Время завершения записи:

T (write accomplish) = 1 OCP clock + 2.5 TIMER clock

Регистр, считанный после записи в тот же регистр, всегда согласован.

**20.1.3.11 Доступ к чтению регистров**

**20.1.3.11.1 Чтение с публикацией**

Этот режим можно использовать только в том случае, если рабочий диапазон частот равен частоте (таймера) < частота (OCP)/4.

Этот режим используется, если в регистре управления таймером **TSICR** (бит **POSTED**) установлен в 1 .

В этом режиме используется схема отложенного чтения для чтения любого внутреннего регистра. Транзакция чтения

немедленно подтверждено на интерфейсе OCP, и значение, которое должно быть прочитано, предварительно подвергается ресинхронизации. Преимущество этого заключается в том, что ни система межсоединений, ни ЦП не простаивают при запросе транзакции чтения. Обратитесь к эррате AM335x для рекомендаций, связанных с режимом публикации.

**20.1.3.11.2 Чтение без публикации**

Этот режим работает независимо от соотношения между частотой интерфейса OCP и функциональной тактовой частотой. Рекомендуемый функциональный диапазон частот: частота (таймера) >= частоты (OCP)/4.

Этот режим используется, если **TSICR** (бит **POSTED**) сброшен в 0 в регистре управления таймером.

В этом режиме используется схема чтения без публикации для чтения любого внутреннего регистра. Транзакция чтения не может быть подтверждена на интерфейсе OCP до тех пор, пока не произойдет эффективная операция чтения, после

ресинхронизация в домене часов таймера. Недостатком является то, что как система межсоединений, так и

ЦП останавливается в этот период.

- Задержка обслуживания прерывания увеличивается, так как система межсоединений и ЦП останавливаются.

- Система межсоединений, включающая логику тайм-аута для обнаружения ошибочных транзакций, может генерировать

нежелательное событие прерывания работы системы.

Этот режим применяется только к трем регистрам: **TCRR**, **TCAR1** и **TCAR2**, которые нуждаются в ресинхронизации из

функционала OCP домена тактирования .

Период простоя определяется как интервал между запросом на доступ для чтения без публикации и восходящим уровнем сигнала команды на прием и может быть определен количественно:

T (stall max.) = 3 OCP clock + 2.5 TIMER clock

Время выборки значения:

T (read sample) = 1 OCP clock + 2.5 TIMER clock