27. Ядро интервального таймера

Общее представление о ядре

Ядро интервального таймера с интерфейсом Avalon® - это интервальный таймер для процессорных систем, основанных на Avalon, таких как процессорная система Nios® II. Ядро предлагает следующие средства:

- 32-битные и 64-битные счётчики.
- Контроль над стартом, остановом и сбросом таймера.
- Два режима счёта: однократный и непрерывный обратный счёт.
- Регистр периода обратного счёта.
- Опции для разрешения или запрещения запроса прерывания (IRQ), когда таймер достигает нуля.
- Опциональный сторожевой (watchdog) таймер, который сбрасывает систему, когда таймер достигает нуля.
- Опциональный генератор периодических импульсов, который выводит импульсы, когда таймер достигает нуля.
- Совместимость с 32-битными и 64-битными процессорами.

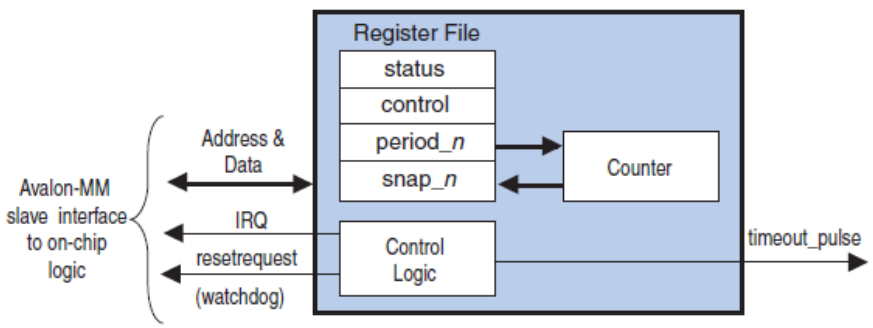
Драйверы устройств предоставляются системной библиотекой HAL для процессора Nios II. Ядро интервального таймера предназначено для SOPC Builder и легко интегрируется в любую систему, генерируемую SOPC Builder. Эта глава состоит из следующих секций:

- "Функциональное описание"
- "Поддержка чипов" на странице 27-2
- "Инсталляция ядра в SOPC Builder" на странице 27-3
- "Программная модель" на странице 27-5

Функциональное описание

На рис. 27-1 показана блок-схема ядра интервального таймера.

Figure 27–1. Interval Timer Core Block Diagram



Ядро интервального таймера имеет два видимых пользователю средства:

- Интерфейс Avalon с распределением в памяти (Avalon-MM), который предоставляет доступ к шести 16-битным регистрам.
- Опциональный выход импульсов, который может быть использован как генератор импульсов.

Все регистры имеют ширину 16-бит, делая ядро совместимым с 16-битными и 32-битными процессорами. Общие регистры реализуются аппаратно только под выбранную конфигурацию. Например, если ядро сконфигурировано под фиксированный период, регистры периода аппаратно отсутствуют.

Следующие последовательности описывают общее поведение ядра интервального таймера:

- Мастер периферия на Avalon-MM, например процессор Nios II, пишет в регистр ядра control, чтобы выполнить следующие задачи:
  - Запускать и останавливать таймер
  - Разрешать или запрещать IRQ
  - Задать одиночный или непрерывный режим обратного счёта
- Процессор читает status регистр, чтобы узнать текущую активность таймера.
- Процессор может задать период таймера, записав значение в регистры периода (period).
- Интервальный счётчик считает до нуля, и когда достигается нуль, он последовательно перезагружает значение из регистра периода (period).
- Процессор может прочитать текущее значение счётчика, сначала записав единицу в регистры привязки (snap) для запроса снимка счётчика, а затем прочитать регистры привязки (snap), чтобы узнать полное значение счётчика.
- Когда счётчик достигает нуля, запускается одно или несколько следующих событий:
  - Если разрешено IRQ – генерируется IRQ.
  - Опциональный выход генератора импульсов утверждается на один тактовый период.
  - Опциональный выход watchdog сбрасывает систему.

Слейв интерфейс Avalon-MM

Ядро интервального таймера реализует простой слейв интерфейс Avalon-MM для предоставления доступа к регистровому файлу. Слейв порт Avalon-MM использует сигнал resetrequest для реализации поведения сторожевого (watchdog) таймера. Этот сигнал является немаскируемым сигналом сброса, он поступает на вход сброса всей периферии Avalon-MM в системе SOPC Builder. Когда утверждён сигнал resetrequest, он форсирует перезагрузку всего подключенного к процессору в системе. За дополнительной информацией обратитесь к секции "Конфигурация таймера в качестве сторожевого таймера" на странице 27-4.

Поддержка чипов

Ядро интервального таймера поддерживается всеми чипами семейств Altera®.

Инсталляция ядра в SOPC Builder

Используйте интерфейс MegaWizard™ для ядра интервального таймера в SOPC Builder, чтобы задать аппаратные свойства. В этой секции описываются доступные опции в интерфейсе MegaWizard.

Период таймаута

Настройка **Timeout Period** определяет начальное значение регистров периода (period). Когда опция **Writeable period** включена, процессор может изменять значение периода записью в регистры периода (period). Когда опция **Writeable period** выключена, период фиксирован и не может быть изменён на стадии прогона. Посмотрите секцию "Аппаратные опции" на странице 27-3 с информацией об опциях регистра.

Timeout Period умножение **Timer Frequency** в целое число раз. **Timer Frequency** фиксирована в качестве настройки частоты системного тактового сигнала, ассоциированного с таймером. Настройка **Timeout Period** может быть задана в единицах: **µs** (микросекунды), **ms** (милисекунды), **seconds** или **clocks** (количество тактовых циклов системного тактового сигнала, ассоциированного с таймером). Текущий период зависит от частоты системного тактового сигнала, ассоциированного с таймером. Если период задан в **µs**, **ms** или **seconds**, то реальный период – это наименьшее количество тактовых циклов, которое больше или равно заданному значению **Timeout Period**. Например, если ассоциированный системный тактовый сигнал имеет период 30 **ns**, а заданное значение **Timeout Period** равно 1 **µs**, то реальный период таймаута будет 1.020 **µs**.

Размер счётчика

Настройка **Counter Size** определяет ширину таймера, которая может быть либо 32, либо 64 бита. 32-битный таймер имеет два 16-битных регистра, а 64-битный таймер имеет четыре 16-битных регистра. Эта опции применяется также к регистрам привязки (snap).

Аппаратные опции

Следующие опции влияют на аппаратную структуру ядра интервального таймера. Для удобства, список **Preset Configurations** предлагает несколько предустановленных аппаратных конфигураций:

- **Simple periodic interrupt** (простое периодическое прерывание) — эта конфигурация прекрасно подходит для систем, которым требуется только периодический генератор IRQ. Период фиксирован и таймер не может быть остановлен, но IRQ можно запретить.
- **Full-featured** (полнофункциональный) — эта конфигурация прекрасно подходит для встроенных процессорных систем, которым требуется таймер с переменным периодом, который может запускаться и останавливаться под управлением процессора.
- **Watchdog** (сторожевой) — эта конфигурация прекрасно подходит для систем, которым требуется сторожевой таймер для сброса системы на случай, когда система перестаёт отвечать. Обратитесь к секции "Конфигурация таймера в качестве сторожевого таймера" на странице 27-4.

Опции регистра

В табл. 27-1 показаны настройки, влияющие на регистры ядра интервального таймера.

Табл. 27-1. Опции регистра

Опция	Описание
Перезаписываемый период	Когда эта опция разрешена, мастер периферия может измерять период обратного счёта записью в регистры периода (period). Когда запрещена, период обратного счёта фиксирован в качестве заданного **Timeout Period**, а регистры периода аппаратно отсутствуют.
Читаемый кадр	Когда эта опция разрешена, мастер периферия может читать кадр текущего обратного счёта. Когда запрещена, статус счётчика детектируется только через другие индикаторы, такие как регистр status или сигнал IRQ. В этом случае, регистр snap аппаратно отсутствует, а чтение этого регистра возвращает неопределённое значение.
Контрольные биты старт/стоп	Когда эта опция разрешена, мастер периферия может запускать и останавливать таймер записью битов START и STOP в регистр control. Когда запрещена, таймер работает непрерывно. Когда разрешена опция **System reset on timeout (watchdog),** бит START также существует, в зависимости от опции **Start/Stop control bits**.

Опции выходного сигнала

В табл. 27-2 показаны настройки, влияющие на выходные сигналы ядра интервального таймера.

Табл. 27-2. Опции выходного сигнала

Опция	Описание
Импульс таймаута (шириной 1 такт)	Когда эта опция включена, ядро выводит сигнал timeout_pulse. Этот сигнал принимает значение 1 на один тактовый цикл, когда таймер достигает нуля. Когда эта опция выключена, сигнал timeout_pulse не существует.
Системный сброс по таймауту (watchdog)	Когда эта опция включена, ядро слейв порта Avalon-MM добавляет сигнал resetrequest. Этот сигнал принимает значение 1 на один тактовый цикл, когда таймер достигает нуля, вызывая сброс всей системы. Интервальный таймер останавливается по сбросу. Явная запись START бита в регистре control запускает таймер. Когда эта опция выключена, сигнал resetrequest не существует. Обратитесь к секции "Конфигурация таймера в качестве сторожевого таймера".

Конфигурация таймера в качестве сторожевого таймера

Чтобы сконфигурировать ядро под watchdog, в интерфейсе MegaWizard выберите **Watchdog** в списке **Preset Configurations** или укажите следующие настройки:

- Установите **Timeout Period** (период таймаута) в необходимый "watchdog" период
- Выключите **Writeable period** (Перезаписываемый период).
- Выключите **Readable snapshot** (Читаемый кадр).
- Выключите **Start/Stop control bits** (Контрольные биты старт/стоп).
- Выключите **Timeout pulse** (Импульс таймаута).
- Включите **System reset on timeout (watchdog)** (Системный сброс по таймауту).

Сторожевой таймер пробуждается (выходит из сброса) остановленным. Процессор снова запускает таймер, записывая 1 в START бит регистра control. После запуска таймер может никогда не остановиться. Если внутренний счётчик когда-нибудь достигнет нуля, сторожевой таймер сбросит систему, сгенерировав импульс resetrequest на своём выходе. Чтобы защитить систему от сброса, процессор должен периодически сбрасывать значение обратного счёта таймера, записывая что-либо в один из регистров периода (period) (записываемое число игнорируется). Если процессор пропускает доступ к таймеру, потому что, например, программа перестала нормально функционировать, сторожевой таймер сбрасывает систему и возвращает систему в исходное состояние.

Программная модель

В следующей секции описывается программная модель ядра интервального таймера, включающая карту регистров и программные объявления для доступа к аппаратной части. Для пользователей процессора Nios II Altera предлагает драйверы системной библиотеки слоя аппаратной абстракции (HAL), которые предоставляют вам доступ к ядру интервального таймера, используя функции HAL программного интерфейса приложения (API).

Поддержка системной библиотеки HAL

Поставляемые Altera драйверы интегрируются в системную библиотеку HAL для системы Nios II. Когда это возможно, пользователи HAL получают доступ к ядру через HAL API, что предпочтительнее, чем прямой доступ к регистрам ядра.

Altera поставляет драйвер для двух моделей HAL таймера: таймера системного тактового сигнала и таймер временной метки.

Драйвер системного тактового сигнала

Если ядро таймера сконфигурировано в качестве системного тактового сигнала, оно работает непрерывно в периодическом режиме, используя период по умолчанию, установленный в SOPC Builder. Системные службы таймера запущены как часть обслуживающей программы прерываний этого таймера. Драйвер управляет прерыванием, и поэтому должен иметь сигнал прерывания, подключенный в аппаратной части системы.

Интегрированная среда разработки (IDE) Nios II позволяет вам задать свойства системной библиотеки, которые определяют, какое устройство - таймер будет использовано в качестве таймера системного тактового сигнала.

Драйвер временной метки

Ядро интервального таймера может быть использовано в качестве устройства временной метки, если удовлетворяет следующим требованиям:

- Таймер имеет перезаписываемый регистр периода, сконфигурированный в SOPC Builder.
- Таймер не выбран в качестве системного тактового сигнала.

Nios II IDE позволяет вам задать свойства системной библиотеки, которые определяют, какое устройство - таймер будет использовано в качестве таймера временной метки.

Если аппаратно таймер не сконфигурирован с перезаписываемый регистром периода, вызов функции API *alt_timestamp_start()* не сможет сбросить счётчик временной метки. Все другие вызовы HAL API выполняются как следует.

За дополнительной информацией об использовании средств системного тактового сигнала и временной метки, которые используют эти драйверы, обратитесь к настольной книге программиста Nios II. Nios II EDS содержит несколько примеров проектов, использующих ядро интервального таймера.

Ограничения

Драйвер HAL для ядра интервального таймера не поддерживает средство сторожевого сброса ядра.

Программные файлы

Ядро интервального таймера сопровождается следующими программными файлами. Эти файлы определяют аппаратный интерфейс и предлагают HAL драйверы. Разработчики приложений не могут изменять эти файлы.

- **altera_avalon_timer_regs.h** – этот файл определяет карту регистров ядра, предлагая символьные константы для аппаратного доступа.
- **altera_avalon_timer.h**, **altera_avalon_timer_sc.c, altera_avalon_timer_ts.c, altera_avalon_timer_vars.c** – эти файлы реализуют драйверы устройства таймера для системной библиотеки HAL.

Карта регистра

Вы не имеете доступа к ядру интервального таймера напрямую через его регистры, если используете стандартные средства, предлагаемые системной библиотекой HAL для процессора Nios II. В основном, карта регистра применяется программистами для написания драйвера устройства.

Поставляемый Altera HAL драйвер устройства предоставляет прямой доступ к регистрам устройства. Если вы пишите драйвер устройства, а HAL драйвер активизирован для того же устройства, ваш драйвер вступит в конфликт и не будет работать корректно.

В табл. 27-3 показана карта регистров для 32-битного таймера. Ядро интервального таймера использует исходное выравнивание адресов. Например, для доступа к значению регистра control, используйте офсет 0x1.

Табл. 27-3. Карта регистров для 32-битного таймера

Офсет	Имя	R/W	Описание битов						
			15	**...**	**4**	**3**	**2**	**1**	**0**
0	status	RW	*(1)*					RUN	TO
1	control	RW	*(1)*			STOP	START	CONT	ITO
2	periodl	RW	Timeout Period – 1 (bits [15:0]) (период таймаута - 1)						
3	periodh	RW	Timeout Period – 1 (bits [31:16])						
4	snapl	RW	Counter Snapshot (bits [15:0]) (кадр счётчика)						
5	snaph	RW	Counter Snapshot (bits [31:16])						

Примечание к табл. 27-3:

(1) Зарезервировано. При чтении - неопределённое значение. Пишется нуль.

За подробной информацией об исходном выравнивании адресов, обратитесь к главе "Система структуры внутренних соединений для интерфейса с распределением в памяти".

Табл. 27-4. Карта регистров для 64-битного таймера

Офсет	Имя	R/W	Описание битов						
			15	**...**	**4**	**3**	**2**	**1**	**0**
0	status	RW	*(1)*					RUN	TO
1	control	RW	*(1)*			STOP	START	CONT	ITO
2	period_0	RW	Timeout Period – 1 (bits [15:0]) (период таймаута - 1)						
3	period_1	RW	Timeout Period – 1 (bits [31:16])						
4	period_2	RW	Timeout Period – 1 (bits [47:32])						
5	period_3	RW	Timeout Period – 1 (bits [63:48])						
6	snap_0	RW	Counter Snapshot (bits [15:0]) (кадр счётчика)						
7	snap_1	RW	Counter Snapshot (bits [31:16])						
8	snap_2	RW	Counter Snapshot (bits [47:32])						
9	snap_3	RW	Counter Snapshot (bits [63:48])						

Примечание к табл. 27-4:

(1) Зарезервировано. При чтении - неопределённое значение. Пишется нуль.

Регистр status

Регистр status имеет два бита для определений, как показано в табл. 27-5.

Табл. 27-5. Биты регистра status

Бит	Имя	R/W/C	Описание
0	TO	RC	Бит TO (timeout) установлен в 1, когда внутренний счётчик достигнет нуля. После установки события таймаута, бит TO остаётся установленным, пока явно не будет сброшен мастер периферией. Для сброса бита TO запишите нуль в регистр status.
1	RUN	R	Бит RUN читается как 1, когда внутренний счётчик запущен; иначе этот бит читается как 0. Бит RUN не изменяется операцией записи в регистр status.

Регистр control

Регистр control имеет четыре бита для определений, как показано в табл. 27-6.

Табл. 27-6. Биты регистра control

Бит	Имя	R/W/C	Описание
0	ITO	RW	Если бит ITO – 1, ядро интервального таймера генерирует IRQ, когда бит TO регистра status – 1. Когда бит ITO – 0, таймер не генерирует IRQ.
1	CONT	RW	Бит CONT (непрерывно) определяет то, как ведёт себя счётчик, когда достигнет нуля. Если бит CONT – 1, счётчик работает непрерывно, пока не будет остановлен битом STOP. Если бит CONT – 0, счётчик остановится, когда достигнет нуля. Когда счётчик достигнет нуля, он перезагружается значением, хранящемся в регистрах period, в зависимости от состояния бита CONT.
2	START *(1)*	W	Запись 1 в бит START запускает внутренний счетчик (обратного счёта). Бит START – это бит события, который разрешает счётчик, когда выполнена операция записи. Если таймер остановлен, запись 1 в бит START возобновляет счёт с последнего сохранённого значения в счётчике. Если счётчик уже запущен, запись 1 в бит START не имеет эффекта. Запись 0 в бит START тоже не имеет эффекта.
3	STOP *(1)*	W	Запись 1 в бит STOP останавливает внутренний счетчик. Бит STOP – это бит события, который останавливает счётчик, когда выполнена операция записи. Если таймер уже остановлен, запись 1 в бит STOP не имеет эффекта. Запись 0 в бит STOP тоже не имеет эффекта. Если таймер аппаратно сконфигурирован без средства **Start/Stop control bits,** запись в бит STOP не имеет эффекта.

Примечание к табл. 27-6:

(1) Запись 1 в оба бита START и STOP приводит к непредвиденным результатам.

Регистры period_n

Регистры period_*n* совместно сохраняют значения периода таймаута. Внутренний счётчик загружается значением, хранящемся в этих регистрах, при следующих событиях:

- Операция записи в один из регистров period_*n*
- Внутренний счётчик достигает нуля

Реальный период таймера на один тактовый цикл больше, чем значение, хранящееся в регистрах period_*n*, поскольку счётчик учитывает значение нуль за один тактовый цикл.

Запись в один из регистров period_*n* останавливает внутренний счётчик, за исключением аппаратно сконфигурированных без средства **Start/Stop control bits**. Если средство **Start/Stop control bits** выключено, запись в любой регистр не может остановить счётчик. Когда устройство сконфигурировано без средства **Writeable period**, запись в один из регистров period_*n* вызывает сброс счётчика на фиксированное значение **Timeout Period**, заданное на стадии генерации системы.

Регистры snap_n

Мастер периферия может запрашивать когерентный кадр текущего значения внутреннего счётчика, выполняя операции записи (запись данных игнорируется) в один из регистров snap_*n.*

Когда происходит запись, значение счётчика копируется в snap_*n* регистры. Кадр делается при запущенном или остановленном счётчике. Запрос кадра не влияет на работу внутреннего счётчика.

Свойство прерывания

Ядро интервального таймера генерирует IRQ, когда внутренний счётчик достигает нуля, а бит ITO в регистре control установлен в 1. Подтверждение IRQ делается двумя способами:

- Сбросом бита TO в регистре status
- Запретом прерываний сбросом бита ITO в регистре control

Пропуск подтверждения IRQ приводит к неожиданным результатам.