**20.3 RTC\_SS**

**20.3.1 Введение**

Часы реального времени - это точный таймер, который может генерировать прерывания на интервалах, указанных пользователем. Прерывания могут происходить каждую секунду, минуту, час или день. Часы сами могут отслеживать прохождение реального времени в течение нескольких лет, при условии, что он имеет достаточный источник питания все время.

Основной целью RTC является учет времени суток. Другая не менее важная цель RTC – управление цифровыми правами. Некоторая степень защиты от несанкционированного доступа необходима чтобы простая остановка, сброс или повреждение RTC не остались незамеченными, так что, если это произойдет, приложение может повторно получить время суток из надежного источника. Конечная цель RTC - разбудить остальную часть процессора из состояния отключения питания.

Аварийные сигналы доступны для прерывания CPU в определенное время или с периодическими интервалами времени, такими как один раз в минуту или один раз в день. Кроме того, RTC может прерывать CPU каждый раз, когда регистры календаря и времени обновляются или с программируемыми периодическими интервалами.

**20.3.1.1 Особенности**

Часы реального времени (RTC) обеспечивают следующие функции:

• 100-летний календарь (с xx00 по xx99)

• Подсчитывает секунды, минуты, часы, день недели, дату, месяц и год с компенсацией високосного года

• Двоично-кодированное десятичное (BCD) представление времени, календаря и тревоги

• 12-часовой режим (с AM и PM) или 24-часовой режим

• Аварийное прерывание

• Периодическое прерывание

• Одиночное прерывание CPU

• Поддерживает внешний 32.768-kHz кристалл или внешний источник синхросигнала той же частоты

**20.3.1.2 Неподдерживаемые функции RTC**

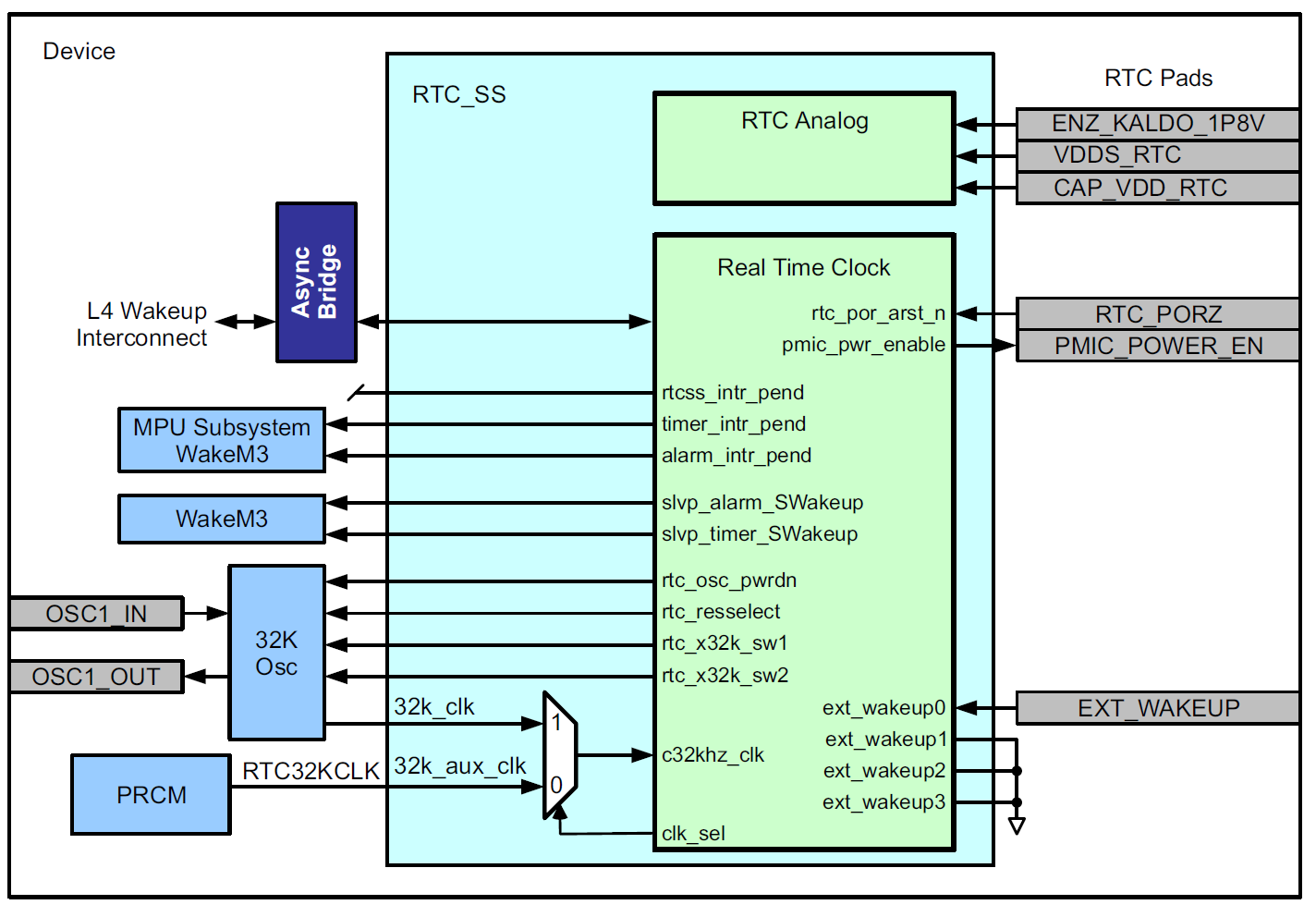
Это устройство поддерживает только одно внешнее событие пробуждения RTC.

**20.3.2 Интеграция**

Это устройство включает в себя модуль подсистемы часов реального времени (RTCSS), позволяющий легко отслеживать время и дату и генерацию аварийных сигналов в реальном времени.

Интеграция RTC показана на рис. 20-55.

**Рис. 20-55. Интеграция RTC**



**20.3.2.1 Атрибуты подключения RTC**

Общая связность для модуля RTC в устройстве показана в таблице 20-56.

Таблица 20-56. Атрибуты связности модуля RTC

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| Power Domain | RTC |
| Clock Domain | PD\_RTC\_L4\_RTC\_GCLK (Interface/OCP)  PD\_RTC\_RTC32KCLK (Func)  CLK\_32K\_RTC (Func) |
| Reset Signals | RTC\_DOM\_RST\_N |
| Idle/Wakeup Signals | Smart Idle/Wakeup |
| Interrupt Requests | Interrupts  Alarm interrupt to MPU Subsystem (RTCINT) and WakeM3  Timer interrupt to MPU Subsystem (RTCALARMINT) and  WakeM3  Alarm wakeup to WakeM3  Timer wakeup to WakeM3 |
| DMA Requests | None |
| Physical Address | L4 Wakeup slave port |

**20.3.2.2 Управление синхронизацией и сбросом RTC**

Функциональный тактовый синхросигнал RTC (c32khz\_clk входной сигнал) по умолчанию поступает из CLK32\_KHZ источника тактового синхросигнала из периферийной PLL. Он также может быть получен от 32-KHz осциллятора через мультиплексор тактовых импульсов внутри и контролируется RTC\_SS.

Таблица 20-57. Тактовые сигналы RTC

|  |  |  |  |
| --- | --- | --- | --- |
| **Clock Signal** | **Max Freq** | **Reference / Source** | **Comments** |
| ocp\_clk  (Interface clock) | 100 MHz | CORE\_CLKOUTM4 / 2 | pd\_rtc\_l4\_rtc\_gclk  From PRCM |
| rtc\_32K\_clk\_rtc\_32k\_clk  (Oscillator functional clock) | 32.768 KHz | OSC1\_IN | CLK\_32K\_RTC  From OSC1\_IN |
| rtc\_32k\_clk\_rtc\_32k\_aux\_clk  (Internal functional clock) | 32.768 KHz | PER\_CLKOUTM2 / 5859.3752 | pd\_rtc\_rtc\_32kclk  From PRCM |

**20.3.2.3 Список выводов RTC**

Модуль RTC не содержит контактов внешнего интерфейса.

Таблица 20-58. Список выводов RTC

|  |  |  |
| --- | --- | --- |
| **Pin** | **Type** | **Description** |
| RTC\_PORz | I | RTC Power On Reset |
| EXT\_WAKEUP | I | External wakeup |
| PMIC\_POWER\_EN | O | Power enable control for external power management IC |
| **Analog Signals** | | |
| ENZ\_KALDO\_1P8V | I | Enable 1.8V LDO |
| VDDS\_RTC | P | 1.8V Voltage Supply |
| CAP\_VDD\_RTC | A | Decoupling Cap when internal 1.8 V LDO is enabled, 1.1  V supply when internal 1.8 V LDO is disabled. |

**20.3.3 Функциональное описание**

В этом разделе определяются возможности и требования к прерыванию модуля.

**20.3.3.1 Функциональная блок-схема**

На рис. 20-56 показана блок-схема модуля RTC. На рис. 20-57 показана функциональная блок-схема RTC.

Рис. 20-56. Блок-схема RTC

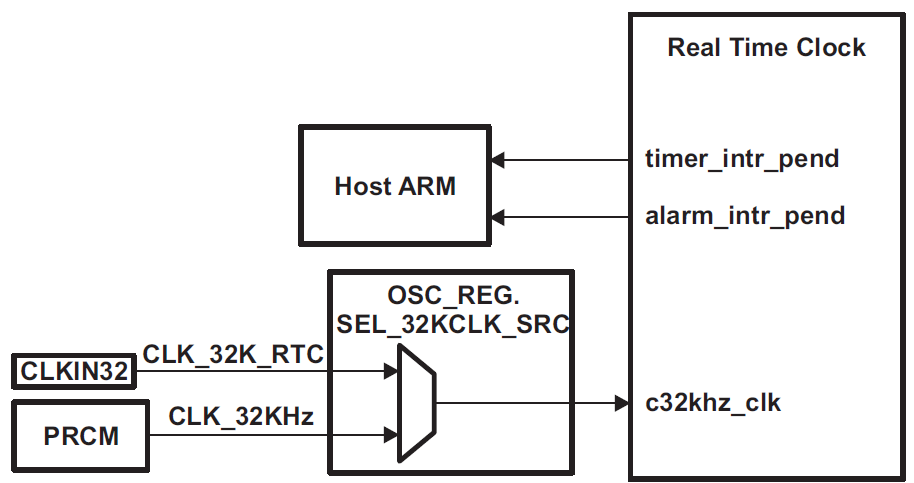
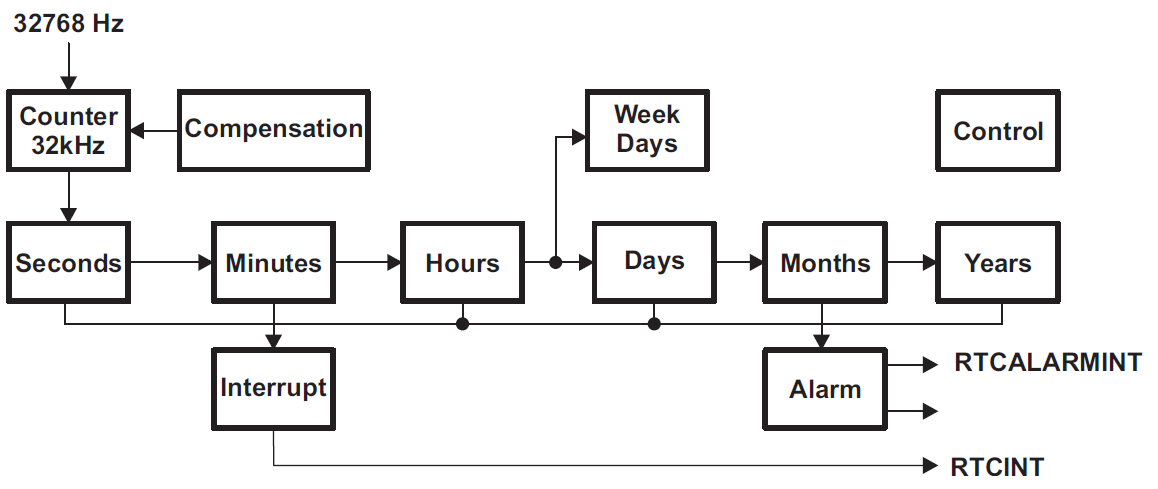


Рис. 20-57. Функциональная блок-схема RTC



**20.3.3.2 Источник синхросигнала**

Опорный синхросигнал для RTC может быть получен от внешнего кристалла (используется с осциллятором 32K RTC), внешний осциллятор 32KHz или от периферийной PLL. RTC имеет внутренний буфер осциллятора для поддержки прямой работы с кристаллом. Кристалл подключен между контактами RTC\_XTALIN и RTC\_XTALOUT. RTC\_XTALIN - это вход встроенного генератора, а RTC\_XTALOUT – выход генератора обратно в кристалл. Генератор может быть включен или отключен с помощью регистра RTC\_OSC\_REG. Дополнительные сведения о подключении к кристаллу RTC см. в руководстве по данным для конкретного устройства.

Вместо кристалла можно использовать внешний тактовый генератор 32.768-kHz. В таком случае источник синхросигнала подключается к RTC\_XTALIN, а RTC\_XTALOUT остается не подключенным.

Источник синхросигнала 32-KHz выбирается с помощью бита OSC\_CLK.SEL\_32KCLK\_SRC.

Если RTC не используется, RTC\_XTALIN контакт должен удерживаться на низком уровне, а RTC\_XTALOUT должен быть оставлен не подключен. Бит RTC\_disable в управляющем регистре (CTRL\_REG) может быть установлен для экономии энергии; однако бит RTC\_disable не должен быть очищен после его установки. Если приложение требует чтобы модуль RTC был остановлен и продолжил работу, то следует использовать STOP\_RTC бит в CTRL\_REG.

**20.3.3.3 Описание сигналов**

В таблице 20-59 перечислены сигналы и их описания для RTC.

Таблица 20-59. Сигналы RTC

|  |  |  |
| --- | --- | --- |
| **Signal** | **I/O** | **Description** |
| RTC\_XTALIN | I | Входной сигнал временной базы РТК. RTC\_XTALIN может управляться с помощью опорного сигнала 32.768-kHz, или RTC\_XTALIN и RTC\_XTALOUT могут быть подключены к внешнему кристаллу. Этот сигнал является входным сигналом для внутреннего генератора RTC. |
| RTC\_XTALOUT | O | Выходной сигнал временной базы РТК. RTC\_XTALOUT - это вывод из внутреннего RTC осциллятора. Если кристалл не используется в качестве временной базы для RTC\_XTALIN, RTC\_XTALOUT  должен быть отключен. |

**20.3.3.4 Поддержка прерываний**

**20.3.3.4.1 Прерывания ЦП**

RTC генерирует два выходных сигнала прерывания:

• timer\_intr (RTCINT) - это прерывание таймера.

• alarm\_intr (RTCALARMINT) является аварийным прерыванием.

ПРИМЕЧАНИЕ: Оба выхода прерывания поддерживают высокий уровень и высокий импульс.

**20.3.3.4.2 Описание прерывания**

**20.3.3.4.2.1 Прерывание таймера RTCINT (timer\_intr)**

Прерывание таймера может генерироваться периодически: каждую секунду, каждую минуту, каждый час или каждый день (см. INTERRUPTS\_REG[1:0] для описания того, как это настроить). IT\_TIMER бит регистра прерывания активирует это прерывание. Прерывание таймера активно - низкий уровень.

RTC\_STATUS\_REG[5:2] обновляются только при каждом новом прерывании и происходят в соответствии с таблицей 20-60.

Например, бит 2 (SEC) всегда будет установлен, когда пройдет одна секунда. Он также будет установлен, когда одна минута прошла с момента завершения одной минуты также знаменует собой завершение одной секунды (из 59 секунд до 60 секунд). То же самое верно для часов и дней: каждый из них также будет соответствовать прохождению секунды.

И наоборот, бит 5 (ДЕНЬ) всегда будет установлен, когда день прошел. Он также может быть установлен, когда прошел час, минута или секунда. Однако это происходит только тогда, когда истекший час, минута или секунда соответствует началу нового дня.

Таблица 20-60. События триггера прерывания

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | One day has passed | One hour has passed | One minute has passed | One second has passed |
| STATUS\_REG[5] (DAY) | 1 | 0/1(1) | 0/1(1) | 0/1(1) |
| STATUS\_REG[4]  (HOUR) | 1 | 1 | 0/1(1) | 0/1(1) |
| STATUS\_REG[3] (MIN) | 1 | 1 | 1 | 0/1(1) |
| STATUS\_REG[2] (SEC) | 1 | 1 | 1 | 1 |

*(1) Это событие инициируется только тогда, когда прошедшая единица времени (например, день) соответствует прохождению другой единицы (например, секунды). Например, когда часы тикают с 00:23:59:59 (дни: часы: минуты: секунды) до 01:00:00:00.*

**20.3.3.4.2.2 Аварийное прерывание RTCALARMINT (alarm\_intr)**

Аварийное прерывание может быть сгенерировано, когда время, установленное в регистрах TC ALARM, точно такое же, как в реестрах TC. Это прерывание затем генерируется, если установлен IT\_ALARM бит регистра прерываний.

Это прерывание является низкоуровневым. RTC\_STATUS\_REG[6] указывает, что прерывание IRQ\_ALARM\_CHIP произошло. Это прерывание отключается путем записи «1» в RTC\_STATUS\_REG[6].

Для установки аварийного сигнала:

• Изменение ALARM\_SECONDS, ALARM\_MINUTES, ALARM\_HOURS, ALARM\_DAY, ALARM\_WEEK, ALARM\_MONTH, и ALARM\_YEAR регистрируется в точное время, в которое должен генерироваться аварийный сигнал.

• Установите бит IT\_ALARM в регистре RTC\_INTERRUPTS, чтобы включить аварийное прерывание.

**20.3.3.5 Руководство по программированию/использованию**

**20.3.3.5.1 Формат данных времени/календаря**

Данные времени и календаря в RTC хранятся в формате двоичного десятичного кода (BCD). В формате BCD, десятичные числа с 0 по 9 кодируются их двоичным эквивалентом. Хотя большая часть регистров времени/календаря имеют 4 бита, назначенные каждой цифре BCD, некоторые из полей регистров короче поскольку диапазон допустимых чисел может быть ограничен. Например, только 3 бита требуются для представления дня недели (WEEKS\_REG), так как требуются только номера BCD с 1 по 7.Следующие регистры поддерживают формат BCD:

Следует отметить, что регистры ALARM, которые совместно используют имена, указанные выше, также совместно используют одно и то же форматирование BCD.

• SECOND - второй счет (00-59)

• MINUTE - Количество минут (00-59)

• HOUR - Счетчик часов (12HR: 01-12; 24HR: 00-23)

• DAY - Подсчет дней месяца (01-31)

• WEEK - День недели (0-6: SUN = 0)

• MONTH - Количество месяцев (01-12; JAN = 1)

• YEAR - Количество лет (00-99)

**20.3.3.5.2 Доступ к регистрам**

Существуют следующие три типа регистров, каждый из которых имеет свои ограничения доступа:

• Регистры аварийных сигналов TC и TC

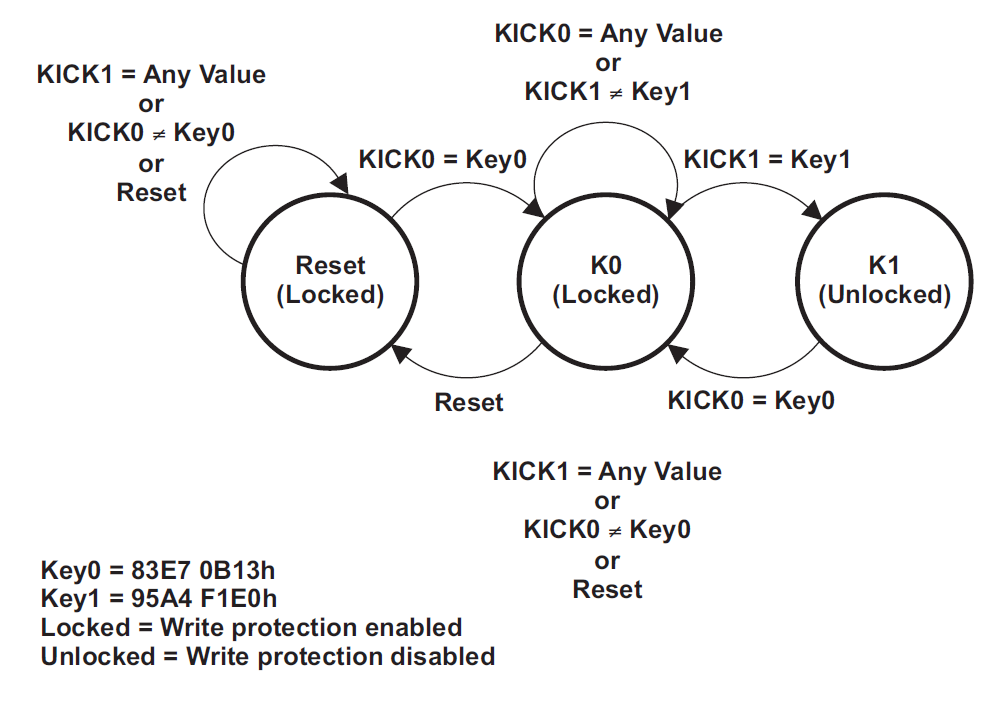
• Общие регистры

• регистры компенсаций

**20.3.3.5.3 Защита OCP MMR от ложных срабатываний**

Модуль также содержит механизм запуска (рис. 20-58), предотвращающий любые ложные записи из изменение значений регистров. Этот механизм требует двух записей MMR в регистры Kick0 и Kick1с точными значениями данных до освобождения механизма блокировки кикера. После освобождения MMR может быть записан. Данные Kick0 83E7 0B13h; Kick1 данные 95A4 F1E0h. Обратите внимание, что он остается в разблокированном состояние до тех пор, пока OCP-сброс или недопустимый шаблон данных не будет записан в один из регистров Kick0 или Kick1.

Рис. 20-58. Схема конечного автомата Kick Register



• S0 - состояние сброса/ожидания;

• S1 - цикл OCP wrt 83E7 0B13h Kick0 завершенном состоянии

• S2 - состояние UNLOCK MMR WRT;

• S0 - > S1, когда OCP wrt цикл 83E7 0B13h на Kick0

• S1 - > S2 при ОВП по циклу 95A4 F1E0h при Kick1

• S1 - > S0 при сбросе OCP

• S2 - > S0 при событии сброса OCP ИЛИ OCP wrt цикла NOT 83E7 0B13h при Kick0 ИЛИ OCP wrt цикла при Kick1

• S2 - > S1 при ОВП по циклу 83E7 0B13h при Kick0

**20.3.3.5.4 Чтение регистров таймера/календаря (TC)**

Регистры TC имеют регистр чтения-отображения. Чтение реестра секунд обновит все регистры TC. Например, год будет обновляться только при чтении регистра секунд. Регистры времени/календаря обновляются каждую секунду по мере изменения времени. Во время считывания СЕКУНД RTC копирует текущие значения регистров времени/даты в теневые регистры считывания. Эта изоляция гарантирует, что CPU может захватить все значения времени/даты в момент считывания SECONDS и не подвергаться изменению значений регистров из обновлений времени.

При необходимости RTC также предоставляет функцию однократного минутного округления для округления MINUTE: SECOND регистров с точностью до минуты (с нулевыми секундами). Эта функция включена путем установки бита ROUND\_30S в управляющем регистре (CTRL); RTC автоматически округляет значения времени до ближайшей минуты после следующего считывания регистра SECONDS.

*ПРИМЕЧАНИЕ: Программное обеспечение всегда должно сначала прочитать регистр секунд. Однако программное обеспечение не должно опрашивать любой бит состояния для определения времени считывания регистров TC. Таблица 20-61 определяет набор TC, для которого выполняется затенение.*

Таблица 20-61. Имена и значения регистров RTC

|  |  |  |
| --- | --- | --- |
| **Time Unit** | **Range** | **Remarks** |
| Year | 00 to 99 |  |
| Month | 01 to 12 |  |
| Day | 01 to 31 | Months 1, 3, 5, 7, 8, 10, 12 |
| 01 to 30 | Months 4, 6, 9, 11 |
| 01 to 29 | Month 2 (leap year) |
| 01 to 28 | Month 2 (common year) |
| Week | 00 to 06 | Day of week |
| Hour | 00 to 23 | 24 hour mode |
| 01 to 12 | AM/PM mode |
| Minute | 00 to 59 |  |
| Seconds | 00 to 59 |  |

**20.3.3.5.4.1 Секунды округления**

Время можно округлить до ближайшей минуты, установив бит ROUND\_30S управляющего регистра. Когда этот бит устанавливается, значения TC устанавливаются на ближайшее значение минуты на следующей секунде. Бит ROUND\_30S будет автоматически сбрасывается при выполнении округления.

Пример:

• Если текущее время 10H59M45S, операция округления изменит время на 11H00M00S.

• Если текущее время 10H59M29S, операция округления изменит время на 10H59M00S.

**20.3.3.5.5 Изменение регистров ТС**

Для записи правильных данных из/в регистры сигналов TC и TC alarm и считывания регистров в TC ,ARM необходимо сначала считать бит BUSY регистра STATUS, пока BUSY не станет равным нулю. После установки флага BUSY нуль, существует период доступа 15 μ s, в течение которого ARM может программировать регистры аварийных сигналов TC и TC. Как только период доступа 15 μ s проходит, флаг BUSY должен быть считан снова из регистра STATUS как описано ранее. Если ARM обращается к регистрам TC за пределами периода доступа, то доступ не гарантируется.

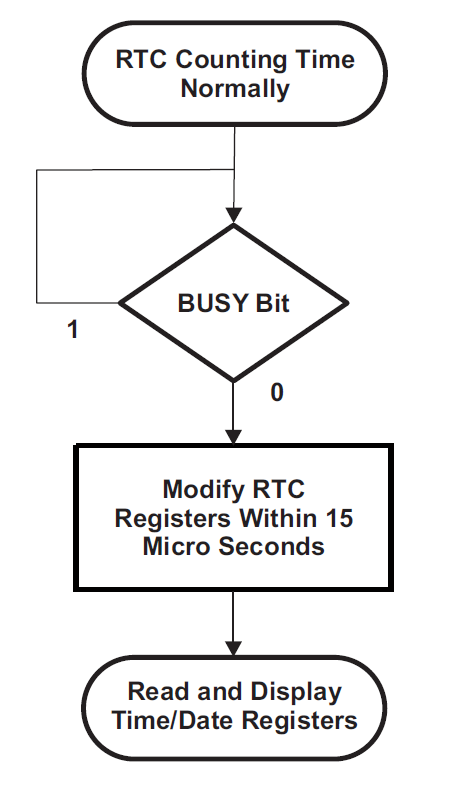
ARM может получить доступ к STATUS\_REG и CTRL\_REG в любое время, за исключением регистра CTRL\_REG[5], который может быть изменен только при остановке RTC. ARM может остановить RTC путем очистки бита STOP\_RTC управляющего регистра. После очистки этого бита бит RUN в STATUS\_REG (бит 1) необходимо проверить, чтобы убедиться, что RTC фактически остановлен. После подтверждения значения TC можно обновить. После обновления значений RTC может быть перезапущен путем сброса Бита STOP\_RTC.

*ПРИМЕЧАНИЕ: После записи в регистр TC пользователь должен подождать 4 тактовых цикла OCP, прежде чем читать значение из регистра. Если это время ожидания не соблюдается и осуществляется доступ к регистру TC, то из регистра будут считаны старые данные.*

.

|  |
| --- |
| **Внимание!!!**  Чтобы исключить любую возможность прерывания процесса чтения реестра, таким образом, создается потенциальный риск нарушения авторизованного 15-микросекундного период доступа, настоятельно рекомендуется отключить все входящие прерывания во время процесса чтения регистра |

Рис. 20-59. Управление потоком для обновления регистров RTC



**20.3.3.5.5.1 Общие регистры**

ARM может получить доступ к STATUS\_REG и CTRL\_REG в любое время (за исключением бита CTRL\_REG[5] который должны быть изменены только при остановке RTC). Для INTERRUPTS\_REG ARM должен соблюдать доступный период доступа для предотвращения ложных прерываний.

Бит RTC\_DISABLE регистра CTRL должен использоваться только для полного отключения функции RTC.

Когда этот бит установлен, тактовый сигнал 32 кГц блокируется, и RTC замораживается. С этого момента сброс этого бита в ноль может привести к неожиданному поведению. Для экономии электроэнергии этот бит следует использовать только в том случае, если RTC функция нежелательна в приложении.

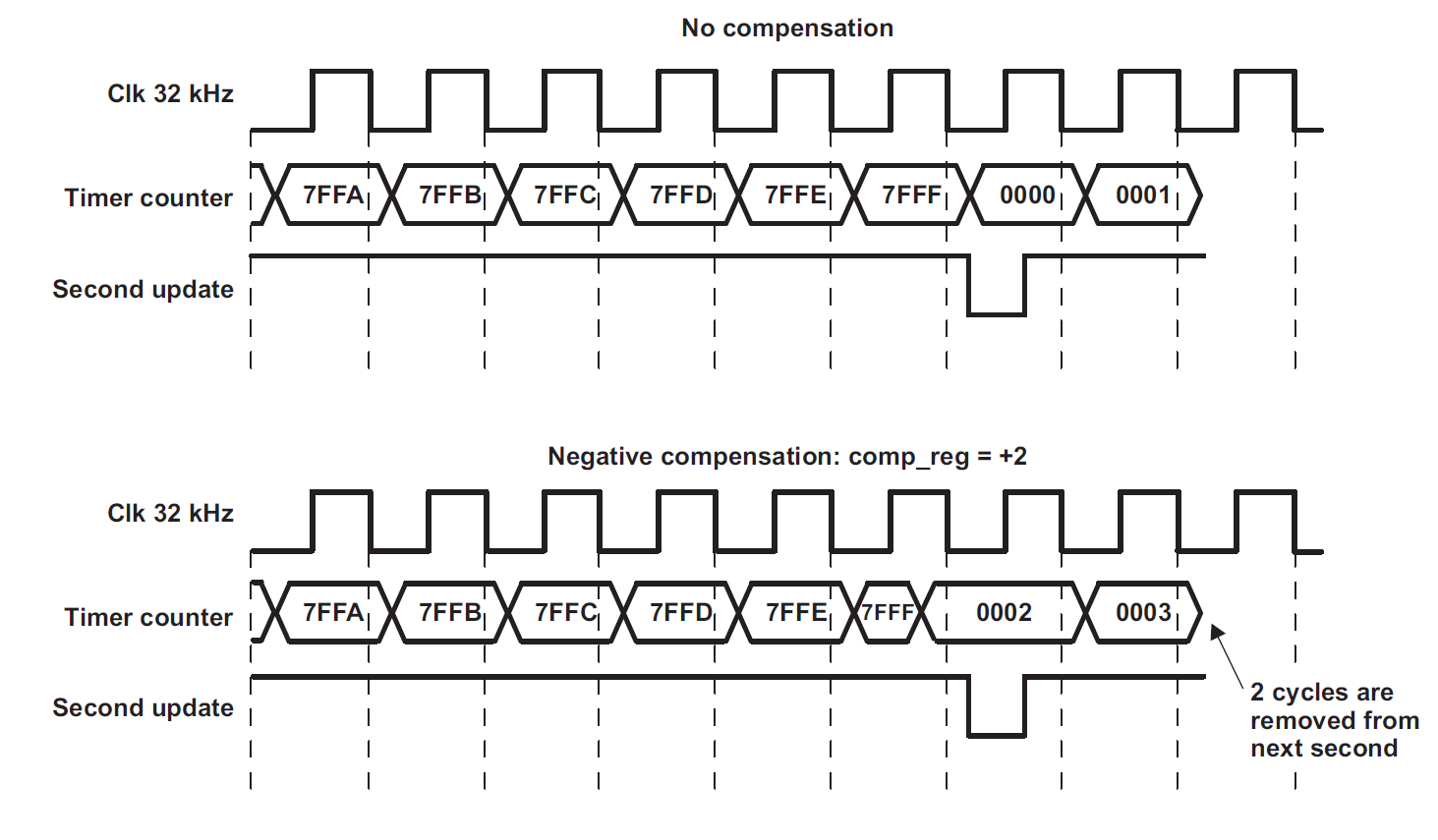
**20.3.3.5.6 Компенсация кристалла**

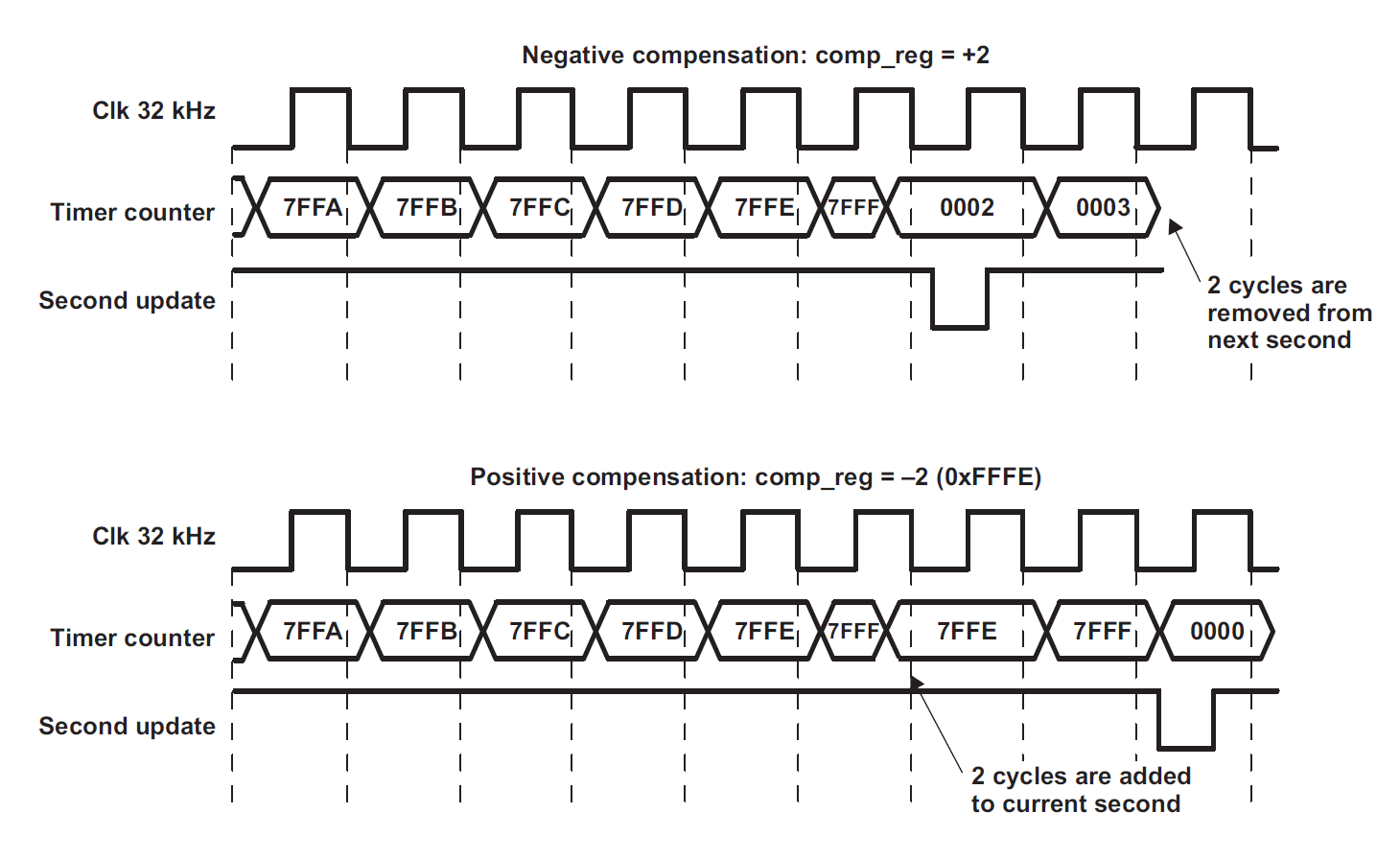
Чтобы компенсировать любую неточность генератора 32 кГц, ARM может выполнить калибровку частоты генератора, рассчитать компенсацию дрейфа по сравнению с одночасовым периодом и загрузить регистры компенсации

значениями компенсации дрейфа. Автоматическая компенсация включается битом AUTO\_COMP\_EN в регистре RTC\_CTRL. Если значение COMP\_REG положительное, компенсация происходит событие изменения секунды. COMP\_REG циклы снимаются со следующей секунды. Если значение COMP\_REG отрицательное, компенсация происходит до наступления новой секунды. COMP\_REG циклы добавляются к текущей секунде. Это обеспечивает компенсацию с точностью 1 32-kHz период каждый час. Форма сигнала ниже обобщает положительный и отрицательный компенсационный эффект.

Доступ к регистрам COMP\_MSB\_REG и COMP\_LSB\_REG должен осужествляться доступный период доступа. Эти регистры не должны обновляться во время компенсации (первая секунда каждого часа), но это нормально обновлять их во время второго предшествующего компенсационного события. Например, ARM может загружать значение компенсации в эти регистры после каждого часового события в течение доступного периода доступа.

Рис. 20-60. Иллюстрация компенсации





**20.3.3.6 Регистры повреждения**

RTC предоставляет три универсальных регистра (SCRATCHx\_REG), которые могут использоваться для хранения 32-битных слов -- эти регистры не имеют функционального назначения для RTC. Программное обеспечение, использующее RTC, может найти регистры SCRATCHx полезныме для указания состояний RTC. Например, SCRATCHx\_REG регистры может использоваться для указания состояния блокировки защиты от записи или непреднамеренного отключения питания. Чтобы указать защиту от записи, программное обеспечение должно записывать уникальное значение в один из регистров SCRATCHx\_REG при отлкюченной защите от записи, и другое уникальное значение при повторном включении защиты от записи. Таким образом, состояние блокировки регистров может быть быстро определено путем считывания регистра SCRATCH. Чтобы указать непреднамеренное выключение питания, программное обеспечение должно записать уникальное значение в один из регистров SCRATCHx\_REG , когда RTC сконфигурирован и включен. Если RTC непреднамеренно выключен, значение записанное в регистр SCRATCH очищается.

**20.3.3.7 Управление электропитанием**

RTC поддерживает протокол Power Idle. Он имеет два порта SWakeup: один для аварийного события и один для события таймера.

Когда RTC находится в режиме IDLE, тактовый сигнал OCP выключается, а тактовый сигнал 32 кГц остается включенным. Время и календарь продолжают считать в режиме IDLE. Когда RTC возвращается в режим FUNCTIONAL, регистры ТС могут быть прочитаны.

Событие Alarm SWakeup может использоваться для активизации RTC в состоянии IDLE. Для этого аварийный сигнал должен быть установлен и активизирован до перехода RTC в состояние IDLE. Как только это будет сделано, SWakeup происходит при срабатывании аварийного события.

*ПРИМЕЧАНИЕ: Поскольку SWakeup не является периодическим, его использование для пробуждения RTC в состоянии IDLE не рекомендуется. Вместо этого используйте Alarm SWakeup.*

**20.3.3.8 Управление питанием - системный уровень (режим PMIC)**

RTC генерирует pmic\_power\_en управление, которое может использоваться для управления внешней PMIC.

Таблица 20-62. Описание pmic\_power\_en

|  |  |  |
| --- | --- | --- |
| **Port** | **Direction** | **Function** |
| rtc\_pwronrstn | Input | Не опционально.  Истинное питание RTC при сбросе домена.  Утверждайте только тогда, когда RTC потерял мощность.  Всегда отключать, если напряжение RTC больше Vmin.  Порт остается отключенным во время нормальной работы. |
| pmic\_power\_en | Output | Необязательно.  Может использоваться для управления внешним PMIC.  0 = OFF  1 = ON (reset state)  ON → OFF (Turn OFF)  By ALARM2 event  OFF → ON (Turn ON)  By ALARM event OR ext\_wakeup event |
| ext\_wakeup | Input |  |

**20.3.4 Сценарии использования**

Следующий список включает высокоуровневые шаги для начала использования RTC:

1. Включите домены тактовых сигналов модуля (для получения подробной информации о домене тактовых сигналов см. раздел 20.3.2 «Интеграция»).

2. Включите модуль RTC с помощью CTRL\_REG.RTC\_disable.

3. Активизируйте синхросигнал 32K из PER PLL, если используется внутренний генератор RTC.

4. Запись в регистры ударов (KICK0R, KICK1R) в RTC.

5. Настройте таймер в RTCSS для требуемого приложения (установите время и дату, будильник и т. д.).

6. Запустите RTC (в CTRL\_REG.STOP\_RTC).