MSP430x1xxFamily

Семейство микроконтроллеров MSP430x1xx

Руководство пользователя

С12 **Семейство микроконтроллеров MSP430x1xx**.
Руководство пользователя: Пер. с англ. – М.:
Серия «Библиотека Компэла». ЗАО «Компэл», 2004. – 368 с.

ISBN 5-98730-001-0

Данное руководство представляет собой перевод User's Guide MSP430x1xx Family (slau049d) выпущенного компанией Texas Instruments в январе 2004 года. В переводе учтены и исправлены все опечатки отмеченные в Errata MSP430x1xx Family User's Guide (slaz007) (июнь 2004), а также ряд ошибок, обнаруженных в процессе перевода.

Руководство пользователя MSP430x1xx – необходимый инструмент инженера-разработчика, работающего над созданием электронных приборов на базе популярных микроконтроллеров серии MSP430 со сверхнизким потреблением энергии.

ISBN 5-98730-001-0

MSP430x1xxFamily

Содержание

Texas Instruments

К ЧИТАТЕЛЮ

Основанная в 1930 г., компания **Texas Instruments Incorporated** сегодня известна как мировой лидер в области цифровых сигнальных процессоров и технологий обработки аналоговых сигналов.

С момента создания в 1954 году компанией Texas Instruments первого коммерческого транзистора, в 1958 году первой интегральной микросхемы, в 1971 г. первого однокристального микроконтроллера и за годы своего существования, компания оказала и продолжает оказывать существенное влияние на направления мирового развития электронной техники и технологии производства электронной аппаратуры.

Компания Texas Instruments на протяжении многих лет находится в пятёрке ведущих производителей электронных компонентов и создателей технологии обработки сигналов.

Рубеж 20 – 21 века – это эпоха бурного развития Интернета, и компания Texas Instruments была, есть и старается оставаться лидером в области современных технологий связи. Компания Texas Instruments инвестировала и продолжает инвестировать значительные средства в развитие Интернет технологий, системы беспроводной связи, в такие новые быстрорастущие рынки как цифровые видеокамеры и цифровое аудио. Так, компания Texas Instruments инвестировала в исследования и разработку новых технологий 1 млрд. долл. США в 2002 г., 1,75 млрд. долл.США в 2003, а ориентировочная сумма инвестиций в 2004 г. превысит 2,1 млрд. долл. США. На предприятиях компании трудится в общей сложности 36 000 высококвалифицированных специалистов на всех континентах, за исключением Австралии и Антарктиды.

С создания компанией Texas Instruments в 1982 г первого однокристального цифрового сигнального процессора началось триумфальное шествие технологии цифровой обработки аналоговых сигналов по земному шару. Ныне, почти каждый сотовый телефон или цифровой фотоаппарат, который Вы берёте в руки, каждое интернет-соединение, и даже каждая мелодия, льющаяся из радиоприёмника являются результатом работы цифровых сигнальных процессоров и аналоговых технологий компании Texas Instruments. Более 40% мирового объёма производимых цифровых сигнальных процессоров выпускается компанией Texas Instruments. Остальные 60% цифровых сигнальных процессоров производятся компаниями Analog Devices, Agere

Systems, Freescale Semiconductors, STMicroelectronics и рядом более мелких фирм.

В 1999 году компания Texas Instruments начала производство семейства флэш-RISC 16-битных, программно/конструктивно совместимых сверхмалопотребляющих микроконтроллеров MSP430. За очень короткое время новый микроконтроллер стал поистинне «золотым камнем» в изделиях с батарейным питанием. Сегодня, мировую популярность микроконтроллеров семейства MSP430 трудно переоценить. Ни один из признанных мирововых лидеров в производстве микроконтроллеров не может предложить рынку изделие, способное конкурировать с MSP430 по соотношению цена/качество, уровню управления энергопотреблением, набору размещённых на кристалле узлов.

Благодаря своим уникальным качествам, микроконтроллеры серии MSP430 завоевали широчайшую популярность у разработчиков электронной техники. Однако эта популярность в нашей стране была бы еще более высокой, если бы вся обширная документация, выпущенная компанией Texas Instruments, была бы доступна на русском языке. Этой книгой мы начинаем серию публикаций, посвященных микроконтроллерам серии MSP430. На очереди выход первого сборника переводов Примеров применения (Application Reports), показавшихся нам наиболее интересными для отечественных инженеров, а также Руководство пользователя MSP430x4xx.

Ваши пожелания и замечания
просим присылать по адресу:
E-mail: TI@compel.ru
Бренд-менеджер
по продукции Texas Instruments
компании «Компэл» – Илья Фурман.

MSP430x1xxFamily

Предисловие

Texas Instruments

Введение

Об этом руководстве

В настоящем руководстве рассматриваются модули и периферийные устройства семейства микроконтроллеров MSP430x1xx. Представлен обобщенный обзор каждого модуля и периферийного устройства. Не все микроконтроллеры обладают полным набором функций и особенностей модулей и периферийных устройств, рассмотренных здесь. Кроме того, конкретная реализация модулей и периферии может различаться в разных устройствах семейства, либо они могут быть реализованы не в полном объеме. Назначение выводов, подключение источников внутренних сигналов и рабочие параметры отличаются от устройства к устройству. Пользователю необходимо изучить информацию по конкретному микроконтроллеру для выяснения подробностей его работы.

Дополнительная документация

Дополнительную информацию по рассматриваемой теме можно найти на сайте ***http://www.ti.com/msp430***.

Предупреждение FCC

Это оборудование предназначено для тестирования только в лабораторной среде. Оно генерирует, использует и может излучать радиочастотную энергию, и не было протестировано на соответствие с ограничениями для вычислительных устройств, предусмотренными подразделом J раздела 15 правил FCC, разработанных для обеспечения разумной защиты от радиочастотной интерференции. Функционирование этого оборудования в других средах может вызвать взаимные помехи с системами радиокоммуникаций, из-за чего пользователь может понести расходы, связанные с необходимостью проведения каких-либо измерений для снижения помех.

Принятые обозначения

Примеры программ показаны *особым шрифтом*.

Глоссарий

Сокращение	Значение	Раздел с подробным описанием
ACLK	Auxiliary Clock (вспомогательное тактирование)	«Модуль основного тактирования»
ADC	Analog-to-Digital Converter (аналого-цифровой преобразователь, АЦП)	
BOR	Brown-Out Reset (Сброс при пониженном питающем напряжении)	«Системы сброса, прерываний и режимы работы»
BSL	Bootstrap Loader (начальный загрузчик программной памяти или ОЗУ)	*www.ti.com/msp430*
CPU	Central Processing Unit (центральное процессорное устройство, ЦПУ)	«16-разрядное RISC CPU»
DAC	Digital-to-Analog Converter (цифро-аналоговый преобразователь, ЦАП)	
DCO	Digitally Controlled Oscillator (осциллятор с цифровым управлением)	«Модуль основного тактирования»
dst	Destination (Назначение)	«16-разрядное RISC CPU»
FLL	Frequency Locked Loop (система автоматической подстройки частоты)	
GIE	General Interrupt Enable (общее разрешение прерываний)	«Системы сброса, прерываний и режимы работы»
INT (N/2)	Integer portion of N/2 (целая часть N/2)	
I/O	Input/Output (вход / выход)	«Цифровые входы/выходы»
ISR	Interrupt Service Routine (процедура обработки прерывания)	
LSB	Least-Significant Bit (младший бит)	
LSD	Least-Significant Digit (младший разряд)	
LPM	Low-Power Mode (режим пониженного энергопотребления)	«Системы сброса, прерываний и режимы работы»
MAB	Memory Address Bus (адресная шина памяти)	
MCLK	Master Clock (главное тактирование)	«Модуль основного тактирования»
MDB	Memory Data Bus (шина данных памяти)	
MSB	Most-Significant Bit (старший бит)	
MSD	Most-Significant Digit (старший разряд)	
NMI	(Non)-Maskable Interrupt (немаскируемое прерывание)	«Системы сброса, прерываний и режимы работы»
PC	Program Counter (программный счетчик)	«16-разрядное RISC CPU»

Сокращение	Значение	Раздел с подробным описанием
POR	Power-On Reset (сброс при включении питания)	«Системы сброса, прерываний и режимы работы»
PUC	Power-up clear (очистка при включении питания)	«Системы сброса, прерываний и режимы работы»
RAM	Random Access Memory (оперативное запоминающее устройство, ОЗУ)	
SCG	System Clock Generator (генератор системного тактирования)	«Системы сброса, прерываний и режимы работы»
SFR	Special Function Register (регистр специального назначения)	
SMCLK	Sub-System Master Clock (подсистема главного тактирования)	«Модуль основного тактирования»
SP	Stack Pointer (указатель стека)	«16-разрядное RISC CPU»
SR	Status Register (регистр статуса)	«16-разрядное RISC CPU»
src	Source (источник)	«16-разрядное RISC CPU»
TOS	Top-of-Stack (вершина стека)	«16-разрядное RISC CPU»
WDT	Watchdog Timer (сторожевой таймер)	«Сторожевой таймер»

Соглашения в обозначениях состояния битов регистров

Каждый регистр показывается с ключом, означающим тип доступа к каждому индивидуальному биту и его исходное состояние:

Тип доступа к битам регистра и исходное состояние

Ключ	Тип доступа к биту
rw	Чтение / запись
r	Только чтение
r0	Читается как «0»
r1	Читается как «1»
w	Только запись
w0	Записывается как «0»
w1	Записывается как «1»
(w)	Бит в регистре не реализован; запись 1 приводит к импульсу. Всегда читается как «0»
h0	Очищается аппаратно
h1	Устанавливается аппаратно
-0, -1	Состояние после сигнала PUC
-(0), -(1)	Состояние после сигнала POR

MSP430x1xxFamily

Введение

Раздел I.

Texas Instruments

Введение

В этом разделе описывается архитектура MSP430.

1.1. Архитектура

Микроконтроллеры семейства MSP430 содержат 16-разрядное RISC CPU, периферийные модули и гибкую систему тактирования, соединенные через фон Неймановскую общую адресную шину (MAB) памяти и шину памяти данных (MDB). Объединяя современное CPU с отображаемыми в памяти аналоговыми и цифровыми периферийными устройствами, семейство MSP430 предлагает решения для приложений со смешанными сигналами.

Семейство MSP430 обладает следующими ключевыми особенностями:

- Архитектура с ультранизким потреблением, увеличивающая время работы при питании от батарей:
  – для сохранности содержимого ОЗУ необходим ток не более 0,1 мкА;
  – модуль тактирования реального времени потребляет 0,8 мкА;
  – ток потребления при максимальной производительности составляет 250 мкА;
- Высококачественная аналоговая периферия для выполнения точных измерений:
  – встроенные модули 12-разрядного или 10-разрядного АЦП скоростью 200 ksps;
  – имеется температурный датчик и источник опорного напряжения VRef;
  – сдвоенный 12-разрядный ЦАП;
  – таймеры, управляемые компаратором для измерения резистивных элементов;
  – схема слежения (супервизор) за напряжением питания;
- 16-разрядное RISC CPU, допускающее новые приложения к фрагментам кода:
  – большой регистровый файл снимает проблему «узкого файлового горлышка»;
  – компактное ядро имеет пониженное энергопотребление и стоимость;
  – оптимизировано для современного высокоуровневого программирования;
  – набор команд состоит из 27 инструкций, поддерживается семь режимов адресации;
  – расширенные возможности векторных прерываний;

- Возможность внутрисхемного программирования Flash-памяти позволяет гибко изменять и обновлять программный код, производить регистрацию данных.

1.2. Гибкая система тактирования

Система тактирования разработана специально для использования в приложениях с питанием от батарей. Вспомогательная низкочастотная система тактирования (ACLK) работает непосредственно от обычного 32 кГц часового кристалла. Модуль ACLK может использоваться в качестве фоновой системы реального времени с функцией самостоятельного «пробуждения». Интегрированный высокоскоростной осциллятор с цифровым управлением (DCO) может быть источником основного тактирования (MCLK) для ЦПУ и высокоскоростных периферийных устройств. Модуль DCO становится активным и стабильным менее чем через 6 мкС после запуска. Решения на основе архитектуры MSP430 позволяют эффективно использовать высокопроизводительное 16-разрядное RISC CPU в очень малые промежутки времени:

- низкочастотная вспомогательная система тактирования обеспечивает работу микроконтроллера в режиме ультранизкого потребления мощности;
- активизация основного высокоскоростного модуля тактирования позволяет выполнить быструю обработку сигналов.

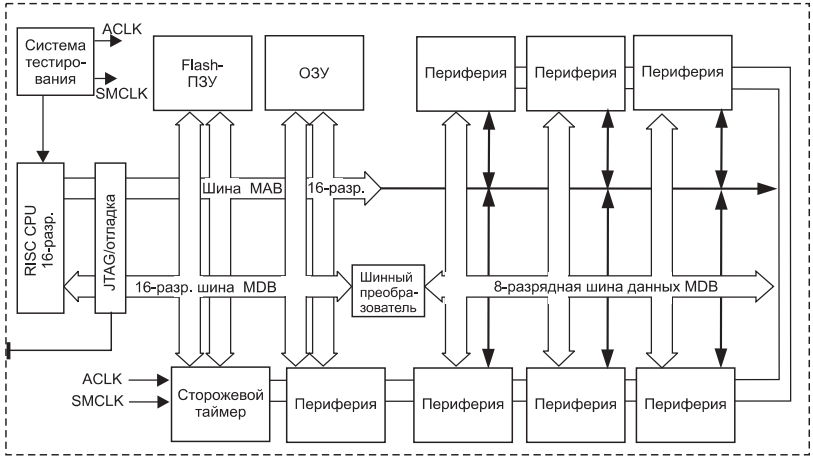


Рис. 1-1. Архитектура MSP430

1.3. Встроенная эмуляция

Специальная встроенная логическая подсистема эмуляции находится непосредственно в устройстве и доступна через JTAG без использования дополнительных системных ресурсов.

Выгоды встроенной эмуляции состоят в следующем:

- возможна фоновая разработка и отладка на полной рабочей скорости выполнения программы;

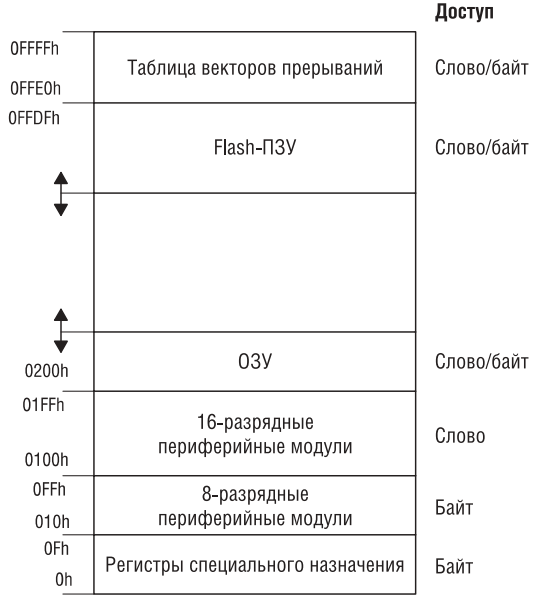


Рис. 1-2. Карта памяти

- поддерживается использование контрольных точек и пошаговое выполнение программы;
- объект внутрисхемной разработки имеет те же характеристики, что и в конечном устройстве;
- сохраняется целостность смешанных сигналов, на которую не влияют помехи кабельной разводки.

1.4. Адресное пространство

Семейство MSP430 имеет фон Ньюмановскую архитектуру с единым адресным пространством для регистров специального назначения (SFR), пери-

ферии, ОЗУ и Flash-памяти программ, в соответствии с рис. 1.2. Конкретное распределение памяти можно узнать из справочных данных на интересующее устройство. Доступ к программному коду выполняется всегда по четным адресам. Данные могут быть доступны как байты или как слова.

Общий объем адресуемой памяти составляет 64 кБ, с учетом предполагаемого расширения.

1.4.1. Flash-память программ

Начальный адрес Flash-памяти зависит от объема имеющейся памяти и различается для разных устройств. Конечный адрес Flash-памяти всегда 0FFFFh. Flash- память может использоваться как для программного кода, так и для данных. Байты или слова таблиц данных могут сохраняться и использоваться непосредственно в Flash-памяти, что исключает необходимость копировать эти таблицы в ОЗУ перед дальнейшим использованием.

Таблица векторов прерываний занимает верхние 16 слов адресного пространства Flash-памяти, при этом вектор прерывания с наивысшим приоритетом находится в самом верхнем адресном слове Flash-памяти (0FFFEh).

1.4.2. ОЗУ

ОЗУ начинается с адреса 0200h. Конечный адрес ОЗУ зависит от объема представленной памяти и различается для каждого конкретного устройства. ОЗУ может использоваться как для программного кода, так и для данных.

1.4.3. Периферийные модули

Периферийные модули отображены в адресном пространстве. Адреса с 0100 до 01FFh зарезервированы для 16-разрядных периферийных модулей. Они будут доступны с помощью команд-слов. Если используются однобайтные команды, допустимы только четные адреса, при этом старший байт результата всегда будет содержать «0».

Адресное пространство с 010h по 0FFh зарезервировано для 8-разрядных периферийных модулей. Эти модули доступны с помощью однобайтных команд. Чтение байтов модулей с помощью команд-слов приведет к появлению в старшем байте непредсказуемого содержимого. Если в байт модуля будут записываться данные в виде слова, то в регистре периферийного модуля сохранится только младший байт этого слова, старший будет проигнорирован.

1.4.4. Регистры специального назначения (SFRs)

Некоторые функции периферии конфигурируются в SFRs. Регистры специального назначения расположены в низших 16-ти байтах адресного пространства и организованы в виде байтов. Обращение к регистрам SFRs производится только с использованием однобайтных команд. Назначение отдельных

битов регистров SFRs описано в техническом руководстве на каждое конкретное устройство.

1.4.5. Организация памяти

Байты расположены в четных или нечетных адресах. Слова располагаются только в четных адресах, как показано на рис.1.3. При работе с командами-словами должны использоваться только четные адреса. Младший байт слова всегда расположен по четному адресу. Старший байт – в следующем нечетном адресе. Например, если слово данных расположено по адресу xxx4h, то младший байт слова данных будет иметь адрес xxx4h, а старший байт слова адрес xxx5h.

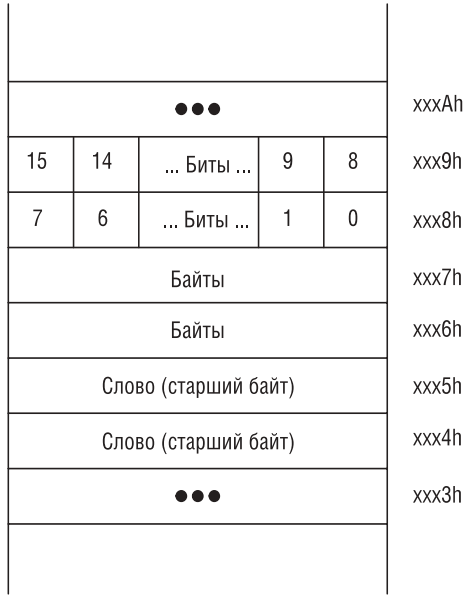


Рис. 1-3. Биты, байты и слова в памяти, организованной побайтно

MSP430x1xxFamily

Системный сброс, прерывания и рабочие режимы

Раздел II.

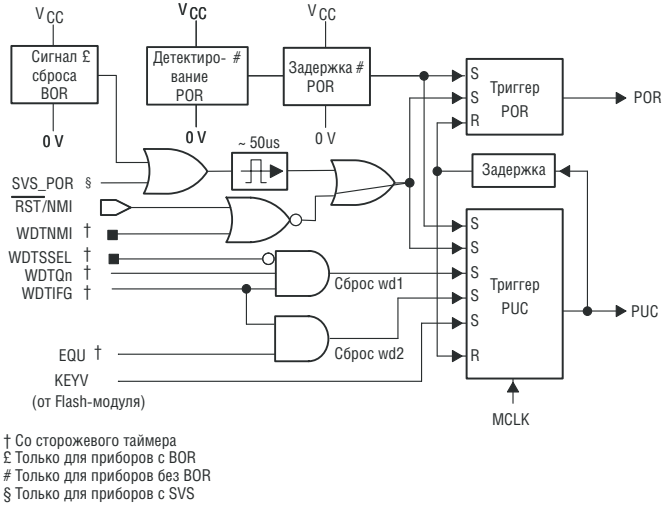
Texas Instruments

Системный сброс, прерывания и рабочие режимы

Этот раздел описывает системный сброс, прерывания и рабочие режимы семейства MSP430x1xx.

2.1. Системный сброс и инициализация

В схеме системного сброса, показанной на рис. 2.1 источниками сброса могут быть сигналы сброса при включении (POR) и очистки при включении



† Со сторожевого таймера
£ Только для приборов с BOR
Только для приборов без BOR
§ Только для приборов с SVS

Рис. 2-1. Схема сброса (POR) и очистки (PUC) при включении

(PUC). Различные события и исходные условия определяют, какой именно из этих сигналов будет сгенерирован.

Сигнал POR сбрасывает устройство. Он может быть сгенерирован в следующих трех случаях:

- включение устройства;
- появление сигнала низкого уровня на выводе RST/NMI, когда он сконфигурирован как вход сигнала «сброса»;
- низкий уровень питания при PORON = 1.

Сигнал PUC генерируется всегда при появлении сигнала POR, но сигнал POR не генерирует PUC. Следующие события приводят к появлению сигнала PUC:

- сигнал POR;
- срабатывание «сторожевого» таймера (только если сторожевой таймер активирован);

- произошло нарушение ключа безопасности «сторожевого» таймера;
- произошло нарушение ключа безопасности Flash-памяти.

2.1.1. Сброс при включении питания (POR)

Когда напряжение Vcc повышается медленно, детектор POR удерживает сигнал POR в активном состоянии до тех пор, пока Vcc не превысит уровень V(POR), как показано на рис. 2.2. Когда питающее напряжение Vcc повышается быстрее задержки POR, сигнал POR удерживается в активном состоянии в течение времени t(POR_DELAY) для корректной инициализации MSP430.

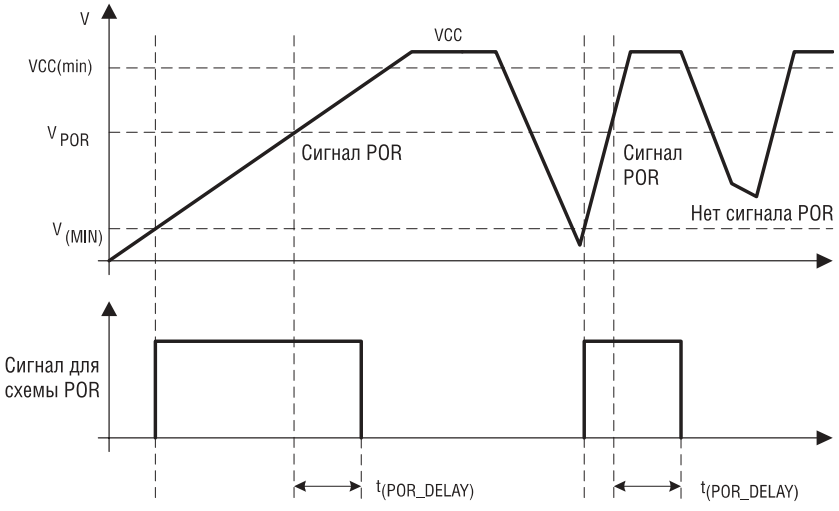


Рис. 2-2. Временные диаграммы сигнала POR

Если питающее напряжение циклически изменяется, его уровень должен упасть ниже значения V(min), чтобы гарантировано появился новый сигнал POR, когда напряжение восстановится снова. Если напряжение Vcc не падает ниже уровня V(min) в течение цикла или появляется кратковременная импульсная помеха, сигнал POR не генерируется и состояние после включения может быть некорректным. Точные параметры можно узнать из руководства по конкретному устройству.

2.1.2. Сброс при пониженном напряжении питания (BOR)

Некоторые устройства имеют схему сброса при пониженном напряжении питания (см. соответствующие справочные данные), вместо схемы детектирования и задержки POR. Схема сброса при пониженном напряжении питания детектирует понижение питающего напряжения, например, напряжение, подава-

емое или снимаемое с вывода Vcc. Схема сброса при пониженном напряжении сбрасывает устройство, вызывая появление POR-сигнала, когда напряжение прикладывается или снимается. Рабочие уровни показаны на рис. 2.3.

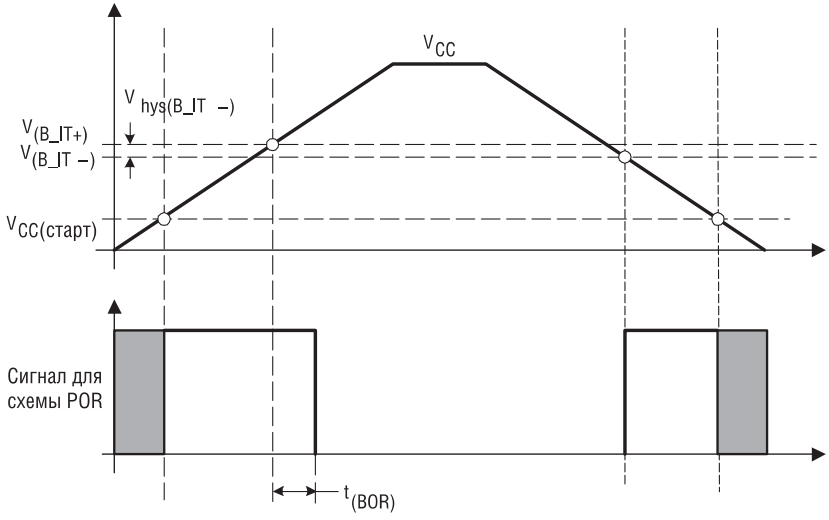


Рис. 2-3. Временные диаграммы схемы сброса при пониженном напряжении питания

Сигнал POR становится активным, когда напряжение Vcc достигает уровня Vcc(start). И остается активным до тех пор, пока Vcc не пересечет порог V(B_IT+) и не закончится выдержка t(BOR). Адаптивная задержка t(BOR) бывает больше при медленном изменении Vcc. Гистерезис VHys(B_IT-) введен, чтобы гарантировать, что питающее напряжение должно снизится ниже уровня V(B_IT-), прежде чем схемой сброса будет сгенерирован другой сигнал POR.

Поскольку уровень V(B_IT-) значительно выше уровня V(min) схемы POR, система BOR обеспечивает сброс при сбоях в источнике питания, когда напряжение Vcc не падает ниже уровня V(min). Точные параметры можно узнать из руководства по конкретному устройству.

2.1.3. Исходное состояние устройства после системного сброса

После снятия сигнала POR, MSP430 переходит в следующее состояние:

- Вывод RST/NMI конфигурируется как вход «сброса»
- Выводы ввода/вывода переключаются в режим ввода в соответствии с описанием в разделе «Цифровые входы/выходы»
- Другие периферийные модули и регистры инициализируются так, как описано в соответствующих разделах этого руководства

- Регистр статуса (SR) сбрасывается
- Сторожевой таймер активизируется в сторожевом режиме
- В программный счетчик загружается адрес, содержащийся в вектора сброса (0FFFEh). ЦПУ начинает выполнять команды с этого адреса.

Программная инициализация

После системного сброса пользовательское программное обеспечение должно инициализировать MSP430 в соответствии с требованиями конкретного приложения. Необходимо выполнить следующие действия:

- Инициализировать указатель стека SP (как правило, указывается вершина ОЗУ)
- Инициализировать сторожевой таймер в зависимости от требований приложения
- Сконфигурировать периферийные модули в зависимости от требований приложения

Дополнительно можно оценить состояние флагов сторожевого таймера, флэш-памяти и неисправности осциллятора для определения источника сброса.

2.2. Прерывания

Приоритеты прерываний показаны на рис. 2.4. Приоритеты определяются порядком расположения модулей в соединяющей их цепи. Чем ближе модуль к ЦПУ/NMIRS, тем выше его приоритет.

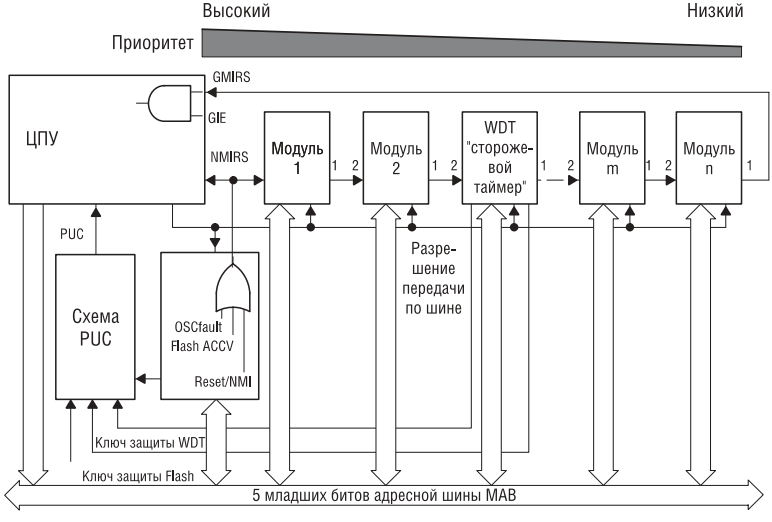


Рис. 2-4. Приоритеты прерываний

Прерывания делятся на три типа:

- Системное (системный сброс)
- Немаскируемое (NMI)
- Маскируемое

2.2.1. Немаскируемые прерывания

Немаскируемые прерывания NMI не маскируются общим битом разрешения прерываний (GIE), но могут управляться индивидуальными битами включения прерывания (ACCVIE, NMIIE, OFIE). Когда происходит немаскируемое прерывание NMI, все биты разрешения NMI-прерываний автоматически сбрасываются. Выполнение программы продолжается с адреса, содержащегося в векторе немаскируемого прерывания (0FFFCh). Программное обеспечение пользователя должно установить необходимые биты NMI-прерывания, чтобы оно было разрешено вновь. Блок-схема источников NMI-прерываний показана на рис. 2.5.

Немаскируемое прерывание NMI может быть вызвано тремя событиями:

- Появление фронта сигнала на выводе RST/NMI
- Появление неисправности осциллятора
- Нарушение доступа к флэш-памяти

Вывод Reset/NMI

При включении микроконтроллера вывод RST/NMI конфигурируется как вывод сброса. Его функциональное назначение определяется в регистре управления сторожевым таймером WDTCTL. Если вывод RST/NMI запрограммирован на функцию сброса, ЦПУ будет находиться в состоянии сброса до тех пор, пока на этом выводе присутствует сигнал низкого уровня. После смены уровня на этом входе на лог.«1», ЦПУ начинает выполнять программу с команды, адрес которой хранится в векторе сброса (0FFFEh).

Если вывод RST/NMI сконфигурирован программой пользователя как вход вызова немаскируемого прерывания, фронт сигнала, выбранного битом WDTNMIES вызовет NMI-прерывание, если установлен бит NMIIE. Также будет установлен флаг NMIIFG.

Примечание:

Удержание вывода RST/NMI в состоянии лог.«0».

Когда вывод RST/NMI сконфигурирован в NMI-режиме, сигнал, вызывающий NMI-прерывание, не должен удерживаться на выводе RST/NMI в состоянии лог.«0». Если появится сигнал PUC от какого-либо источника, когда NMI-сигнал имеет низкий уровень, микроконтроллер будет сброшен, поскольку сигнал PUC изменит назначение вывода RST/NMI и он станет входом сигнала сброса.

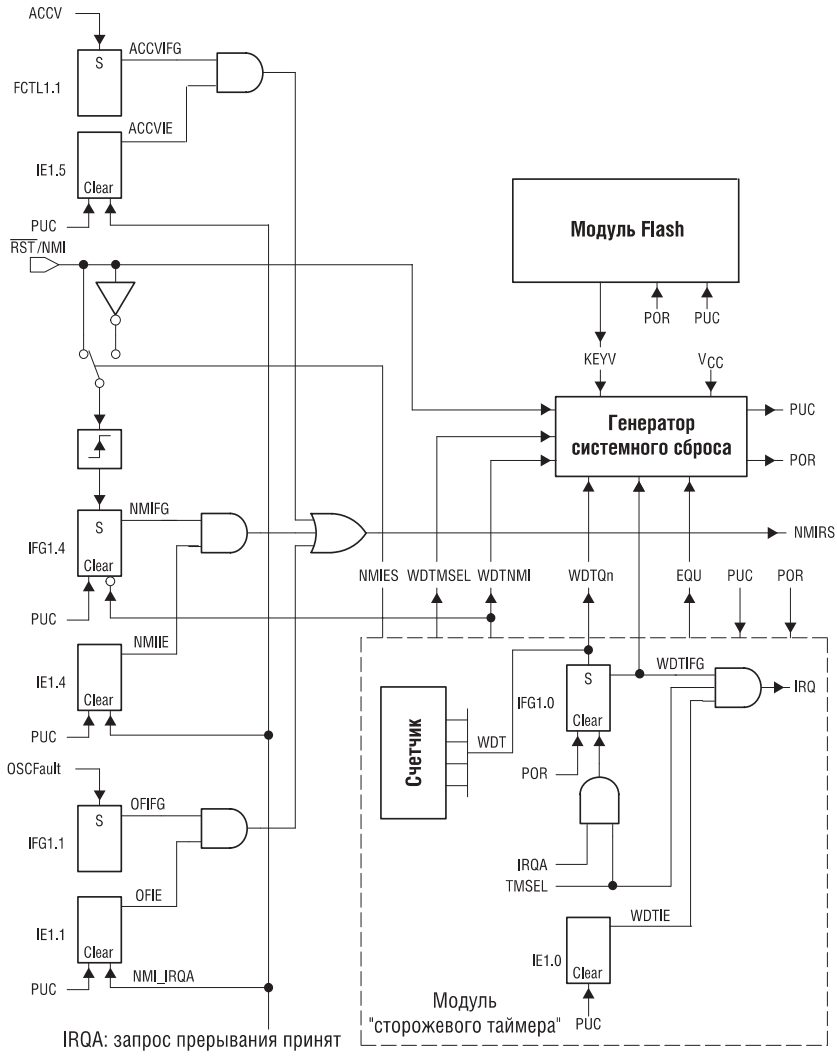


Рис. 2-5. Блок-схема источников немаскируемого прерывания

Примечание:

Модификация WDTNMIES.

Когда выбран режим NMI и изменен бит WDTNMIES, появление NMI-прерывания определяется уровнем сигнала на выводе RST/NMI. Если бит выбора

фронта NMI-сигнала изменен до выбора режима NMI, немаскируемое прерывание NMI не генерируется.

Нарушение доступа к Flash-памяти

Флаг ACCVIFG устанавливается, когда происходит нарушение доступа к Flash-памяти. Генерация NMI-прерывания при нарушении доступа к Flash-памяти происходит при установленном бите ACCVIE. Проверкой флага ACCVIFG в процедуре обработки NMI-прерывания можно определить, было ли вызвано прерывание нарушением доступа к Flash-памяти.

Неисправность осциллятора

Сигнал неисправности осциллятора позволяет предотвратить ошибки, связанные с неправильным функционированием осциллятора. Установкой бита OFIE можно разрешить генерацию NMI-прерывания при неисправности осциллятора. С помощью флага OFIFG процедура обработки NMI-прерывания может проверить, было ли NMI-прерывание вызвано неисправностью осциллятора.

Сигнал неисправности осциллятора может быть вызван PUC-сигналом, поскольку он переводит осциллятор LFXT1 из режима HF в режим LF. Сигнал PUC также отключает осциллятор XT2.

Пример программы обработки немаскируемого прерывания NMI

NMI-прерывание имеет много возможных источников. NMI-прерывание автоматически сбрасывает биты разрешения прерываний NMIIE, OFIE и ACCVIE. Пользовательская процедура обработки NMI-прерывания сбрасывает флаги прерывания и включает биты разрешения прерываний в соответствии с требованиями приложения так, как показано на рис. 2.6.

Примечание:

Разрешение NMI-прерывания с помощью ACCVIE, NMIIE и OFIE

Установка битов разрешения NMI-прерывания ACCVIE, NMIIE и OFIE не должна производится в теле процедуры обработки NMI-прерывания, за исключением случая, когда это происходит непосредственно перед командой RETI. В противном случае могут появиться вложенные NMI-прерывания, что приведет к переполнению стека и дальнейшей непредсказуемой работе.

2.2.2. Маскируемые прерывания

Маскируемые прерывания вызываются периферийными устройствами, имеющими возможность вызова прерываний, включая ситуацию переполнения сторожевого таймера в активном режиме. С помощью индивидуальных битов разрешения прерывания можно отключать источники прерываний как по

отдельности, так и все сразу с использованием общего бита разрешения всех прерываний (GIE) в регистре статуса (SR).

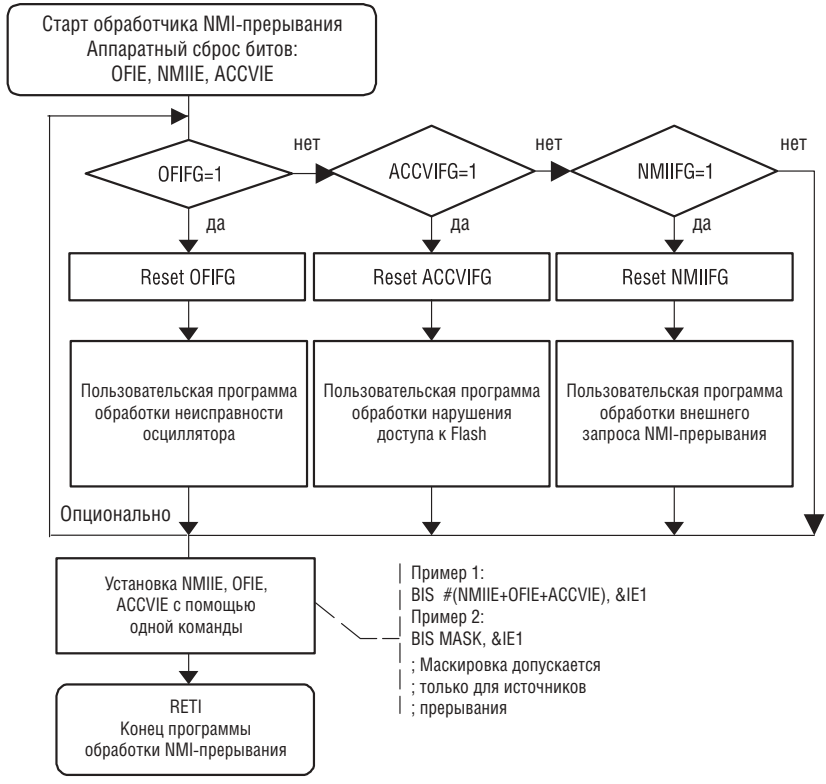


Рис. 2-6. Алгоритм процедуры обработки NMI-прерывания

Каждое конкретное периферийное прерывание будет рассмотрено в этом руководстве в разделе описания соответствующего периферийного устройства.

2.2.3. Обработка прерывания

Если периферийное устройство запросило прерывание и включены биты общего разрешения прерываний GIE и индивидуальный бит разрешения прерывания от этого устройства, будет вызвана процедура обработки прерывания. Для вызова немаскируемого (NMI) прерывания достаточно установки только индивидуального бита разрешения прерывания.

Получение прерывания

Время задержки вызова прерывания составляет 6 машинных циклов, с момента приема запроса на прерывание и до начала выполнения первой команды процедуры обработки прерывания, как показано на рис. 2.7. Логика обработки запроса прерывания имеет следующую последовательность:

1) Любая текущая команда выполняется до конца;

2) Содержимое программного счетчика PC, указывающего на следующую команду, помещается в стек;

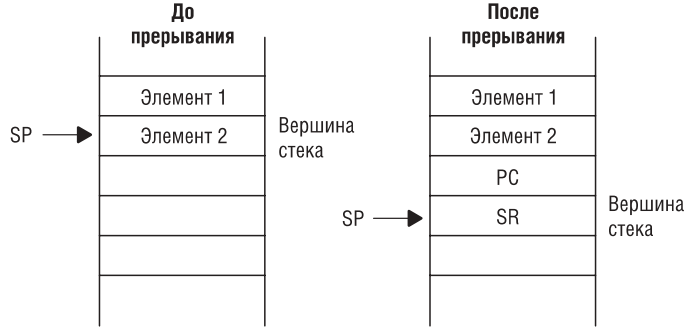


Рис. 2-7. Процесс прерывания

3) Содержимое регистра статуса SR помещается в стек;

4) Если поступило несколько прерываний во время выполнения последней команды, обрабатывается прерывание с наивысшим приоритетом, остальные ожидают обслуживания;

5) Автоматически сбрасывается флаг одного источника прерывания. Флаги запроса остальных прерываний остаются установленными в ожидании обслуживания программным обеспечением.

6) Регистр SR очищается, за исключением бита SCG0, остающегося неизменным. В результате прекращается работа в любом режиме пониженного энергопотребления;

7) Содержимое вектора прерывания загружается в PC и начинается выполнение процедуры обработки прерывания с загруженного адреса.

Возврат из прерывания

Подпрограмма обработки прерывания заканчивается такой командой:

RETI (*возврат из подпрограммы обработки прерывания*)

Для возврата из прерывания необходимо 5 машинных циклов, чтобы выполнить действия, показанные на рис. 2.8.

1) Восстанавливается из стека содержимое регистра SR. Становятся актуальными все предыдущие установки GIE, CPUOFF и пр., в не зависимости от установок, использовавшихся в процедуре обработке прерывания.

2) Восстанавливается из стека содержимое программного счетчика PC и начинается выполнение программы с того места, где она была прервана.

Рис. 2-8. Возврат из прерывания

Вложенные прерывания

Разрешаются вложенные прерывания, если бит GIE установлен во время выполнения процедуры обработки прерывания. Когда вложенные прерывания разрешены, любое прерывание, возникающее во время выполнения одной подпрограммы обработки прерывания, вызовет выполнение своей подпрограммы, несмотря на приоритеты прерываний.

2.2.4. Векторы прерываний

Векторы прерываний и стартовые адреса расположены в адресном диапазоне с 0FFFFh по 0FFE0h, как показано в таблице 2.1. Вектор программируется пользователем с помощью указания 16-разрядного стартового адреса соответствующей процедуры обработки прерывания. Полный перечень векторов прерываний приводится в справочном руководстве каждого конкретного устройства.

Таблица 2.1. Источники прерываний, флаги и векторы

Источник прерывания	Флаг прерывания	Характер прерывания	Адрес слова	Приоритет
Включение питания, внешний сброс, сигнал сторожевого таймера, проверка пароля Flash-памяти	WDTIFG KEYV	Сброс	0FFFEh	15, наивысший
NMI-прерывание, неисправность осциллятора, нарушение доступа к Flash-памяти	NMIIFG OFIFG ACCVIFG	Немаскируемое Немаскируемое Немаскируемое	0FFFCh	14

Таблица 2.1 (Окончание)

Источник прерывания	Флаг прерывания	Характер прерывания	Адрес слова	Приоритет
Определяется устройством			0FFFAh	13
Определяется устройством			0FFF8h	12
Определяется устройством			0FFF6h	11
Сторожевой таймер	WDTIFG	Маскируемое	0FFF4h	10
Определяется устройством			0FFF2h	9
Определяется устройством			0FFF0h	8
Определяется устройством			0FFEEh	7
Определяется устройством			0FFECh	6
Определяется устройством			0FFEAh	5
Определяется устройством			0FFE8h	4
Определяется устройством			0FFE6h	3
Определяется устройством			0FFE4h	2
Определяется устройством			0FFE2h	1
Определяется устройством			0FFE0h	0, низший

В регистрах SFRs расположены биты доступа к некоторым модулям, биты разрешения прерываний и флаги прерываний. Регистры SFRs занимают начало адресного пространства и реализованы в однобайтном формате. Доступ к ним производится также с помощью однобайтных команд. Конфигурация регистров SFRs описывается индивидуально для каждого конкретного устройства.

2.3. Режимы работы

Семейство MSP430 разработано для приложений с ультранизким потреблением мощности и имеет различные режимы работы, показанные на рис. 2.10.

Режимы работы учитывают три различные потребности:

- ультранизкое потребление
- скорость и пропускную способность
- минимизацию потребления тока конкретной периферией

Типичное потребление тока микроконтроллерами семейства MSP430 показано на рис. 2.9.

Режимы низкого энергопотребления 0-4 конфигурируются с помощью битов CPUOFF, OSCOFF, SCG0 и SCG1 в регистре статуса. Преимущество включения битов управления режимом CPUOFF, OSCOFF, SCG0 и SCG1 в состав регистра статуса SR состоит в том, что текущий режим работы может быть сохранен, путем помещения содержимого SR в стек во время работы процедуры обработки прерывания. Выполняемая программа возвращается к предыдущему режиму

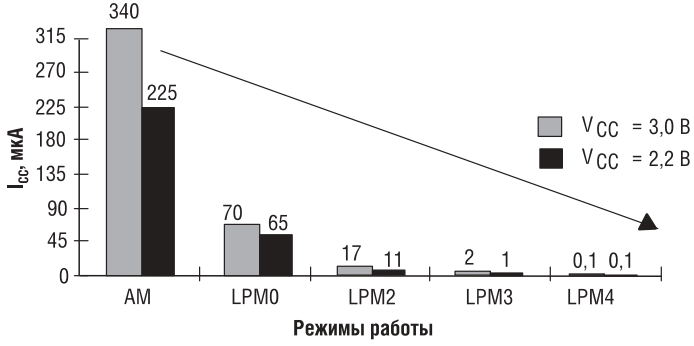


Рис. 2-9. Типичное потребление тока устройствами 13x и 14x в зависимости от режима работы

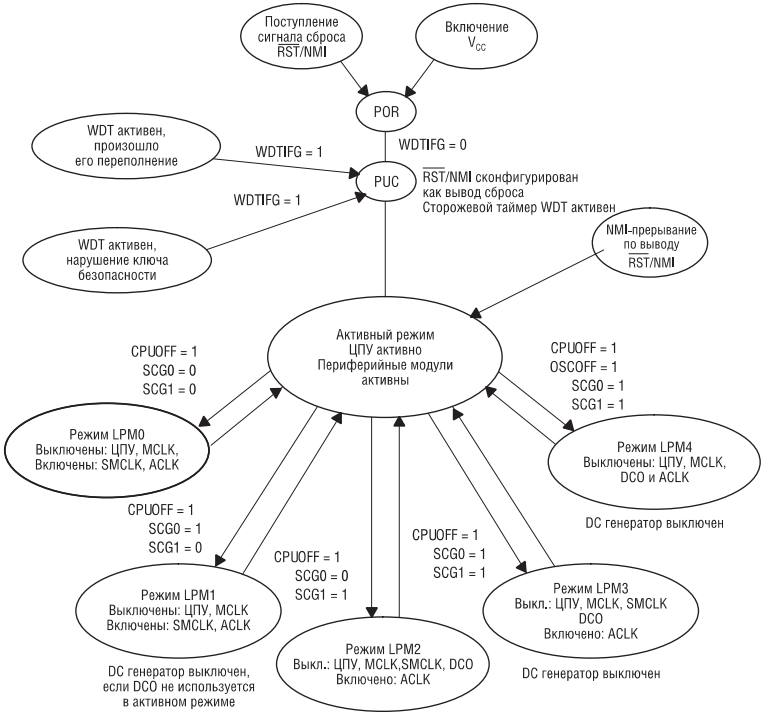


Рис. 2-10. Режимы работы основной системы тактирования MSP430

работы, если сохраненное содержимое регистра SR не было изменено процедурой обработки прерывания. Выполнение программы может продолжится в другом рабочем режиме, если процедура обработки прерывания изменит значение регистра SR в стеке. Обращение к битам управления режимом и стеку может производиться с помощью любой команды.

При изменении любого бита управления режимом, выбранный режим работы активизируется немедленно. При отключении любой системы тактирования, блокируются также периферийные устройства, работающие от этой системы. Периферийные устройства также могут отключаться с помощью соответствующих им индивидуальных управляющих регистров. Состояние всех выводов портов ввода/вывода и ячеек ОЗУ остается неизменным. «Пробуждение» возможно через все разрешенные прерывания.

SCG1	SCG0	OSCOFF	CPUOFF	Режим	Состояние ЦПУ и систем тактирования
0	0	0	0	Активный	ЦПУ и все системы тактирования активны
0	0	0	1	LPM0	ЦПУ и MCLK отключены; SMCLK и ACLK активны
0	1	0	1	LPM1	ЦПУ, MCLK и DCO-осцил. отключены;DC генератор отключен, если DCO не используется для MCLK или SMCLK в активном режиме; SMCLK и ACLK активны
1	0	0	1	LPM2	ЦПУ, MCLK, SMCLK и DCO-осцил. отключены;DC генератор остается включенным; ACLK активно
1	1	0	1	LPM3	ЦПУ, MCLK, SMCLK и DCO-осцил. отключены; DC отключен; ACLK активно
1	1	1	1	LPM4	ЦПУ и все системы тактирования отключены

2.3.1. Вход и выход из режимов пониженного энергопотребления

Появление прерывания выводит микроконтроллер семейства MSP430 из любого режима пониженного энергопотребления.

Программный поток выглядит так:

- Вход в процедуру обработки прерывания:
  - Содержимое регистров PC и SR сохраняется в стеке;
  - Биты CPUOFF, SCG1 и OSCOFF автоматически сбрасываются;
- Параметры для возвращения из процедуры обработки прерывания:
  - Исходное содержимое регистра SR восстанавливается из стека, что приводит к возобновлению работы устройства в предыдущем режиме;
  - Биты регистра SR, сохраненного в стеке, могут быть модифицированы процедурой обработки прерывания, что приведет к переходу в другой рабочий режим после выполнения команды RETI.

```
;Пример входа в режим LPM0
   BIS  #GIE+CPUOFF, SR                    ;Вход в режим LPM0
;...                                       ;Программа останавлива-
                                           ется в этом месте
;
;Выход из режима LPM0 по процедуре обработки прерывания
   BIC  #CPUOFF, 0(SP)                     ;Выход и режим LPM0
                                           по команде RETI
   RETI
;Пример входа в режим LPM3
   BIS  #GIE+CPUOFF+SCG1+SCG0, SR          ;Вход в режим LPM3
;...                                       ;Программа останавлива-
                                           ется в этом месте
;
;Выход из режима LPM3 по процедуре обработки прерывания
   BIC  #CPUOFF+SCG1+SCG0,0(SP)            ;Выход и режим LPM3
                                           по команде RETI
   RETI
```

«Растяжение» времени в режимах пониженного энергопотребления

Когда модуль DCO длительное время отключен при работе устройства в режиме пониженного энергопотребления, следует принимать во внимание его отрицательный температурный коэффициент. Если изменения температуры значительны, частота модуля DCO при включении после выхода из режима пониженного энергопотребления может сильно отличаться от исходной частоты и даже выходить за пределы заданного рабочего диапазона. Избежать этого можно, если устанавливать DCO на самое нижнее значение перед входом в режим пониженного энергопотребления на длительное время, когда температура может изменяться.

```
;Пример входа в режим LPM4 с низшими установками DCO
   BIC  #RSEL2+RSEL1+RSEL0,&BCSCTL1        ;Установка низших
                                           значений RSEL
   BIS  #GIE+CPUOFF+OSCOFF+SCG1+SCG0,
        SR                                 ; Вход в режим LPM4
;...                                       ; Останов программы
;
;Процедура обработки прерывания
   BIC  #CPUOFF+OSCOFF+SCG1+SCG0, 0(SR)    ;Выход в режим LPM4
                                           по команде RETI
   RETI
```

2.4. Принципы создания приложений с низким энергопотреблением

Часто, наиболее важным фактором для снижения энергопотребления является использование системы тактирования MSP430 для увеличения времени пребывания микроконтроллера в режиме LPM3. Типичное потребление тока в этом режиме составляет менее 2 мкА при функционирующей схеме тактирования реального времени и активированной системе прерываний. Модуль ACLK использует часовый кристалл 32 кГц, а ЦПУ тактируется от DCO (выключенного в нормальном режиме), который имеет время «пробуждения» 6 мкС.

- Использование прерываний, «пробуждающих» процессор и управляющий программный поток;
- Периферийные устройства должны включаться только при необходимости;
- Следует использовать интегрированные периферийные модули с низким энергопотреблением вместо функций, реализуемых программными методами. К примеру таймер A и таймер B могут автоматически генерировать сигнал ШИМ и делать захват внешней синхронизации без использования ресурсов ЦПУ;
- Вместо опроса флагов и длительных программных вычислений следует использовать рассчитываемое ветвление и быстрые таблицы преобразований;
- Нужно избегать частого вызова подпрограмм и функций, увеличивающих непроизводительные издержки;
- В длинных программных процедурах необходимо использовать однотактные регистры ЦПУ.

2.5. Подключение неиспользуемых выводов

Правильное подключение всех неиспользуемых выводов приведено в таблице 2.2.

Таблица 2.2. Подключение неиспользуемых выводов

Вывод	Потенциал	Комментарии
AV_{CC}	DV_{CC}	
AV_{SS}	DV_{SS}	
V_{REF+}	Свободный	
Ve_{REF+}	DV_{SS}	
V_{REF-}/Ve_{REF-}	DV_{SS}	
XIN	DV_{SS}	
XOUT	Свободный	
XT2IN	DV_{SS}	Устройства 13x, 14x, 15x и 16x
XT2OUT	Свободный	Устройства 13x, 14x, 15x и 16x
C Px.0 по Px.7	Свободный	Переключены к функции порта, направленного на вывод
RST/NMI	DV_{SS} или V_{CC}	«Подтягивающий» резистор 47 кОм с понижающей емкостью 10 нФ
Test/V_{PP}	DV_{SS}	Устройства P11x
Test	DV_{SS}	Притянутый к нулевому потенциалу через резистор 30K (микросхемы 11x1-серии)
	Свободный	11x1A, 11x2, 12x, 12x2 - серии
TDO	Свободный	
TDI	Свободный	
TMS	Свободный	
TCK	Свободный	

MSP430x1xxFamily

16-разрядное RISC CPU

Раздел III.

Texas Instruments

16-разрядное RISC CPU

В этом разделе описывается ЦПУ MSP430, режимы адресации и набор команд.

3.1. Введение в ЦПУ

ЦПУ включает возможности, специально созданные для современных технологий программирования, таких как вычисляемое ветвление, обработка таблиц и использование языков высокого уровня, подобных языку С. ЦПУ может выполнять адресацию в полном адресном диапазоне без использования страниц памяти.

ЦПУ обладает следующими возможностями:

- RISC-архитектура с 27 командами и 7 режимами адресации;
- Ортогональная архитектура, при которой каждая команда пригодна для каждого режима адресации;
- Полный доступ ко всем регистрам, включая программный счетчик, регистры статуса и указатель стека;
- Однотактные регистровые операции;
- Большой 16-разрядные регистровый файл, уменьшающий количество обращений к памяти;
- 16-разрядная адресная шина, обеспечивающая прямой доступ и ветвление во всем диапазоне памяти;
- 16-разрядная шина данных, позволяющая напрямую манипулировать параметрами шириной в слово;
- Генератор констант немедленно предоставляет шесть используемых наиболее часто значений, уменьшая размер кода;
- Прямой обмен между ячейками памяти без промежуточной записи в регистр;
- Команды и адресация в форматах «слово» и «байт».

Блок-схема ЦПУ показана на рис. 3.1.

3.2. Регистры ЦПУ

ЦПУ включает шестнадцать 16-разрядных регистров. Регистры R0, R1, R2 и R3 имеют специальное назначение. Регистры с R4 по R15 являются рабочими регистрами общего назначения.

3.2.1. Программный счетчик (PC)

16-разрядный программный счетчик (PC/R0) указывает на следующую команду, которая будет выполняться. Каждая команда состоит из четного числа

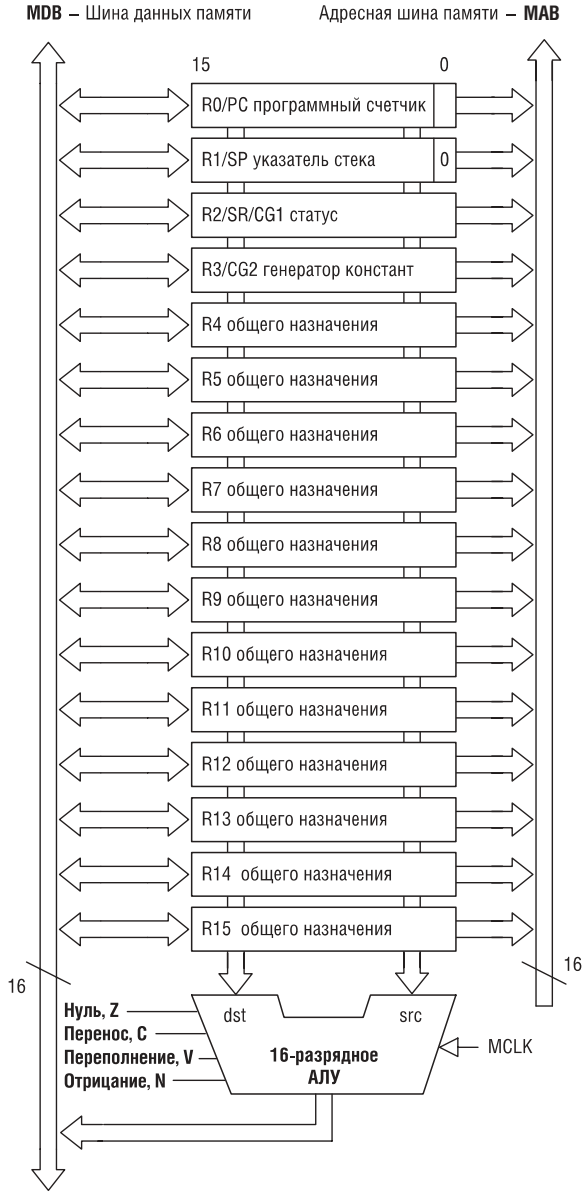


Рис. 3-1. Блок-схема ЦПУ

байтов (два, четыре или шесть), поэтому PC инкрементируется соответственно. Команды доступа в адресном пространстве 64 кБайт выполняются к границам слов, поэтому PC выравнивается к четным адресам. На рис. 3.2 показана организация программного счетчика.

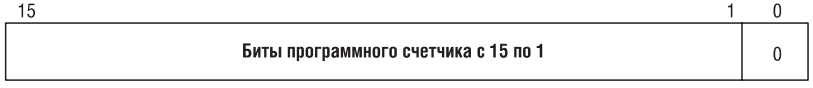


Рис. 3-2. Программный счетчик

Программный счетчик PC может быть адресован всеми командами и во всех адресных режимах. **Некоторые примеры:**

```
MOV    #LABEL, PC   ;Переход к адресу с меткой LABEL
MOV    LABEL, PC    ;Переход к адресу, содержащемуся в пере-
                    менной LABEL
MOV    @R14, PC     ;Косвенный переход по косвенному содер-
                    жимому R14
```

3.2.2. Указатель стека (SP)

Указатель стека (SP/R1) используется ЦПУ для хранения адресов возврата из подпрограмм и прерываний. Стек основан на предекрементной постинкрементной схеме. Кроме того, указатель стека SP может использоваться со всеми командами и во всех адресных режимах. На рис. 3.3 показана организация SP. Указатель стека SP инициализируется в ОЗУ пользователем и выравнивается к четным адресам.

Рис. 3-3. Указатель стека

```
MOV    2(SP),R6     ;Элемент стека I2 в R6
MOV    R7,0(SP)     ;Перезапись в вершину стека (TOS) содер-
                    жимого R7
PUSH   #0123h       ;Помещение числа 0123h на вершину стека
                    (TOS)
POP    R8           ;R8 = 0123h
```

Особенности использования «SP» в качестве аргумента команд PUSH и POP описаны и показаны на рис. 3.5.

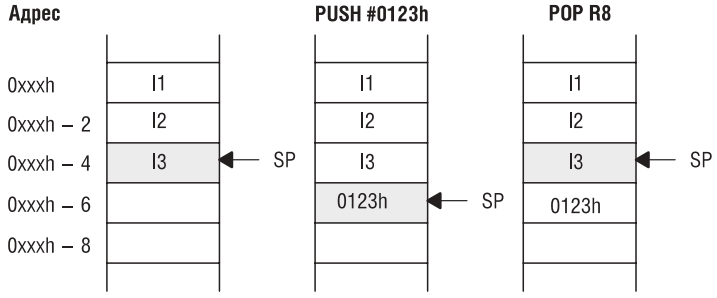


Рис. 3-4. Использование стека

Рис. 3-5. Последовательность PUSH SP – POP SP

Указатель стека изменяется после выполнения команды PUSH SP.

Указатель стека не изменяется после команды POP SP. Команда POP SP помещает SP1 в указатель стека SP (SP2=SP1).

3.2.3. Регистр статуса (SR)

Регистр статуса (SR/R2), используемый как регистр источника или получателя, может адресоваться в регистровом режиме только с помощью команд-слов. Прочие комбинации режимов адресации используются для поддержки генератора констант. На рис. 3.6 показаны биты регистра статуса SR.

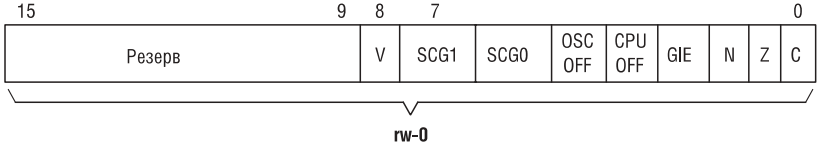


Рис. 3-6. Биты регистра статуса

В таблице 3.1 приведено описание битов регистра статуса.

Таблица 3.1. Описание битов регистра статуса

Бит	Описание	
V	Бит переполнения. Этот бит устанавливается, если результат арифметической операции имеет переполнение в области знаковых переменных.	
	ADD(.B), ADDC(.B)	Устанавливается, когда: Положительный + Положительный = Отрицательный Отрицательный + Отрицательный = Положительный в противном случае сбрасывается
	SUB(.B), SUBC(.B), CMP(.B)	Устанавливается, когда: Положительный – Отрицательный = Отрицательный Отрицательный – Положительный = Позитивный в противном случае сбрасывается
SCG1	Системный тактовый генератор 1. Когда этот бит установлен, SMCLK выключен.	
SCG0	Системный тактовый генератор 0. Когда этот бит установлен, генератор DCO выключен, если DCOCLK не используется для MCLK или SMCLK.	
OSCOFF	Выключение осциллятора. Когда этот бит установлен, осциллятор LFXT1, использующий кристалл, выключен, если LFXT1CLK не используется для MCLK или SMCLK.	
CPUOFF	Выключение ЦПУ. Когда этот бит установлен, ЦПУ выключено.	
GIE	Общий бит разрешения прерываний. Когда этот бит установлен, маскируемые прерывания разрешены. Когда сброшен, все маскируемые прерывания запрещены.	
N	Бит отрицательного результата. Этот бит устанавливается, когда результат операции с байтом или словом отрицательный и сбрасывается, когда результат не отрицательный. Операции со словами: N устанавливается по значению бита 15 результата Операции с байтами: N устанавливается по значению бита 7 результата	
Z	Бит нуля. Этот бит устанавливается, когда результат операции с байтом или словом равен «0» и очищается, если результат не равен «0».	
C	Бит переноса. Этот бит устанавливается, когда результат операции с байтом или словом имеет перенос и очищается, когда переноса нет.	

3.2.4. Регистры генератора констант CG1 и CG2

Шесть обычно используемых констант генерируются с помощью регистров R2 и R3 генератора констант, что исключает необходимость использования дополнительного 16-разрядного слова в программном коде. Константы выбираются путем изменения режима адресации (As) регистра-источника, в соответствии с таблицей 3.2.

Таблица 3.2. Значения генераторов констант CG1, CG2

Регистр	As	Константа	Комментарий
R2	00	- - - - -	Регистровый режим
R2	01	(0)	Режим абсолютной (безусловной) адресации
R2	10	00004h	+4, побитовая обработка
R2	11	00008h	+8, побитовая обработка
R3	00	00000h	0, обработка по словам
R3	01	00001h	+1
R3	10	00002h	+2, побитовая обработка
R3	11	0FFFFh	-1, обработка по словам

Генератор констант обладает следующими преимуществами:

- Не требуются особые команды
- Код не содержит дополнительного слова для шести констант
- Не требуется код (команда) доступа к памяти для получения константы

Ассемблер автоматически использует генератор констант, если одна из шести констант используется как непосредственный исходный операнд. При использовании регистров R2 и R3 в режиме генерации констант, адресация к ним не может быть явной – они действуют только как регистры-источники.

Генератор команд – расширенная система команд

Набор RISC-команд семейства MSP430 состоит только из 27 команд. Однако, генератор констант позволяет поддерживать MSP430-ассемблеру 24 дополнительные эмулированные команды. К примеру, команда с одним операндом:

```
CLR    dst
```

эмулируется командой с двумя операндами такой же длины:

```
MOV    R3,dst
```

где #0 замещается ассемблером, а R3 используется в режиме As=00
Команда `INC dst` замещается командой `ADD 0(R3),dst`

3.2.5. Регистры общего назначения R4-R15

Двенадцать регистров с R4 по R15 являются регистрами общего назначения. Все эти регистры могут быть использованы в качестве регистров данных, указателей адресов или индексных значений и доступны с помощью команд работы с байтами или словами, как показано на рис. 3.7.

Рис. 3-7. Операции регистр-байт/байт-регистр

Пример операции регистр-байт	Пример операции байт-регистр
R5=0A28Fh *R6=0203h* *Mem(0203h)=012h* *ADD.B* *R5,0(R6)* *08Fh* *+012h* *0A1h* *Mem(0203h)=0A1h* *C=0, Z=0, N=1* *(младший байт регистра)+(адресуемый байт)* *->(адресуемый байт)*	*R5=01202Fh* *R6=0223h* *Mem(0223h)=05Fh* *ADD.B* *@R6,R5* *05Fh* *+002h* *00061h* *R5=00061h* *C=0, Z=0, N=0* *(адресуемый байт)+(младший байт регистра)* *->(младший байт регистра, ноль в старшем байте)*

3.3. Режимы адресации

Семь режимов адресации для операнда источника и четыре режима адресации для операнда назначения могут адресовать полное адресное пространство без исключений. В таблице 3.3 приводится конфигурация битов для режимов As (источник) и Ad (назначение).

Таблица 3.3 Режимы адресации операндов источника/получателя

As/Ad	Режим адресации	Синтаксис	Описание
00 / 0	Регистровый режим	Rn	Содержимое регистра является операндом
01 / 1	Индексный режим	X(Rn)	Значение (Rn+X) указывает на операнд. X сохранен в следующем слове

Таблица 3.3 (Окончание)

As/Ad	Режим адресации	Синтаксис	Описание
01 / 1	Символьный режим	ADDR	Значение (PC+X) указывает на операнд. X сохранен в следующем слове. Использован индексный режим X(PC)
01 / 1	Абсолютный (безусловный) режим	&ADDR	Слово, следующее за командой, содержит абсолютный адрес. X сохранен в следующем слове. Использован индексный режим X(SR)
10 / -	Косвенный регистровый режим	@Rn	Содержимое Rn использовано как указатель на операнд
11 / -	Косвенный автоинкремент	@Rn+	Содержимое Rn использовано как указатель на операнд. Содержимое Rn впоследствии увеличивается на 1 для байтовых команд и на 2 для команд-слов.
11 / -	Прямой (непосредственный) режим	#N	Слово, следующее за командой, содержит непосредственную константу N. Использован косвенный автоинкрементный режим @PC+

Семь упомянутых способов адресации подробно рассматриваются в следующих разделах. В большинстве примеров показаны схожие режимы адресации для источника и получателя, но в команде возможны любые правильные комбинации способов адресации источника и получателя.

Примечание: использование меток EDE и TONI

Везде в документации по семейству MSP430 используются универсальные метки EDE и TONI. Они являются только метками, не имеющими никакого специального назначения.

3.3.1. Регистровый режим

Регистровый режим описан в таблице 3.4.

Таблица 3.4. Описание регистрового режима

Код ассемблера	Содержимое ПЗУ
`MOV R10,R11`	`MOV R10, R11`
Длина: Одно или два слова **Операция:** Пересылка содержимого R10 в R11. Содержимое R10 не изменяется. **Комментарий:** Действительно для источника и получателя. **Пример:** `MOV R10,R11`	

Примечание: данные в регистрах

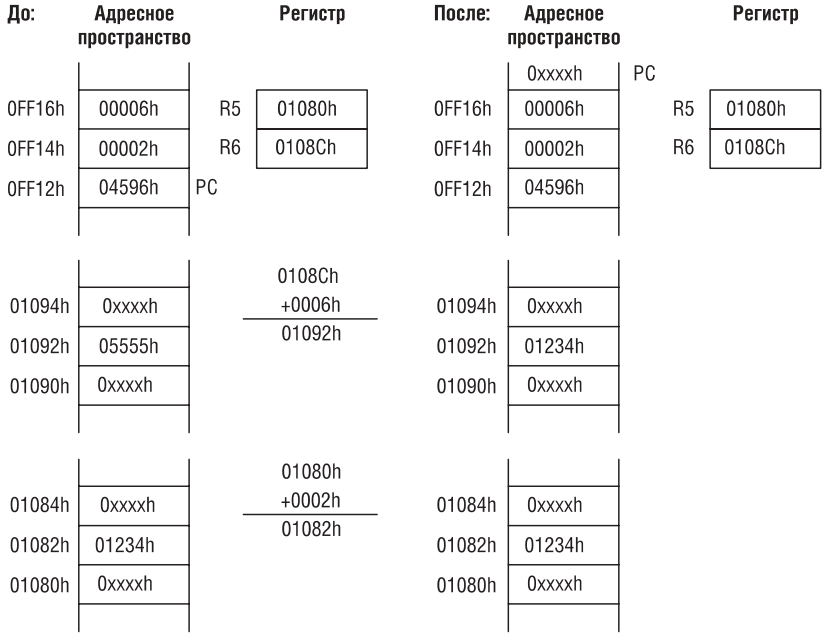
Данные в регистре могут быть доступны с помощью байтовых команд или команд-слов. Если используются байтовые команды, старший байт всегда будет содержать в результате «0». Биты статуса обрабатываются согласно результату байтовой команды.

3.3.2. Индексный режим

Индексный режим описан в таблице 3.5.

Таблица 3.5. Описание индексного режима

Код ассемблера	Содержимое ПЗУ
`MOV 2(R5), 6(R6)`	`MOV X(R5), Y(R6) X=2 Y=6`
Длина: Два или три слова	
Операция: Пересылка содержимого с исходного адреса (равного сумме содержимого R5 + 2) по адресу назначения (содержимое R6 + 6). Регистры источника и получателя (R5 и R6) не изменяются. В индексном режиме программный счетчик автоматически инкрементируется таким образом, что выполнение программы продолжается со следующей команды.	
Комментарий: Действительно для источника и получателя	
Пример: `MOV 2(R5),6(R6)` :	

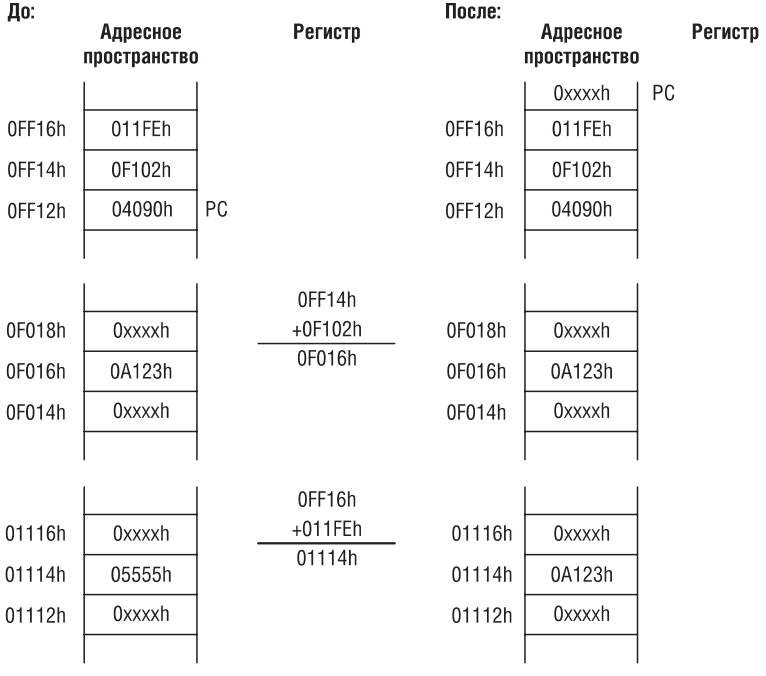


3.3.3. Символьный режим

Символьный режим описан в таблице 3.6.

Таблица 3.6. Описание символьного режима.

Код ассемблера	Содержимое ПЗУ
`MOV EDE, TONI`	`MOV X(PC), Y(PC)` `X=EDE-PCY=TONI-PC`
Длина: Два или три слова	
Операция: Пересылка содержимого с исходного адреса EDE (равного сумме содержимого PC + X) по адресу назначения TONI (содержимое PC + Y). Слова после команды содержат разницу между PC и адресами источника или получателя соответственно. Ассемблер автоматически вычисляет и вставляет смещения X и Y. В символьном режиме программный счетчик автоматически инкрементируется так, что выполнение программы продолжается со следующей команды.	
Комментарий: действительно для источника и получателя	
Пример: `MOV EDE,TONI ;Адрес источника EDE=0F016h ;Адрес получателя TONI=01114h`	

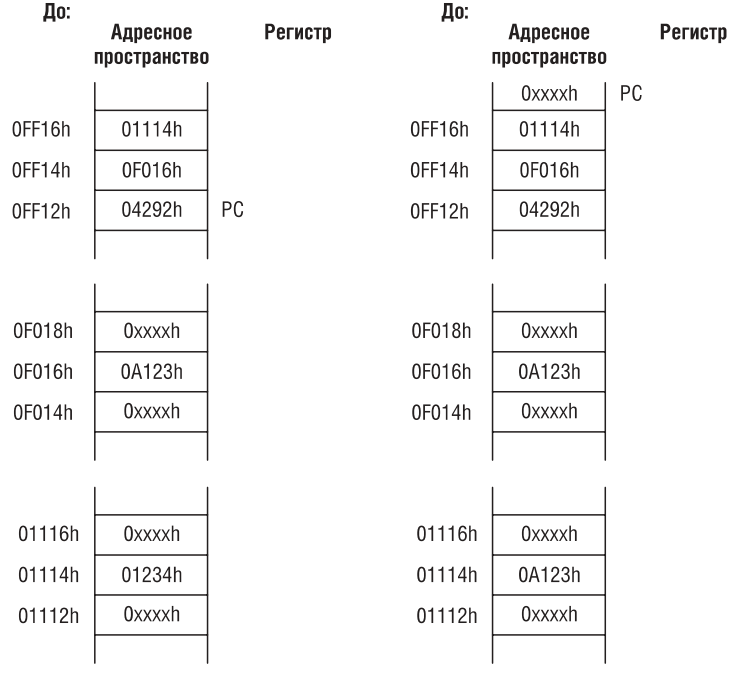


3.3.4. Абсолютный режим

Абсолютный режим описан в таблице 3.7.

Таблица 3.7. Описание абсолютного режима

Код ассемблера	Содержимое ПЗУ
`MOV &EDE, &TONI`	`MOV X(0), Y(0) X=EDEY=TONI`
Длина: Два или три слова	
Операция: Пересылка содержимого с исходного адреса EDE по адресу назначения TONI. Слова после команды содержат абсолютные адреса источника и получателя. В абсолютном режиме программный счетчик автоматически инкрементируется так, что выполнение программы продолжается со следующей команды.	
Комментарий: Действительно для источника и получателя	
Пример: `MOV &EDE,&TONI ;Адрес источника EDE=0F016h ;Адрес получателя TONI=01114h`	



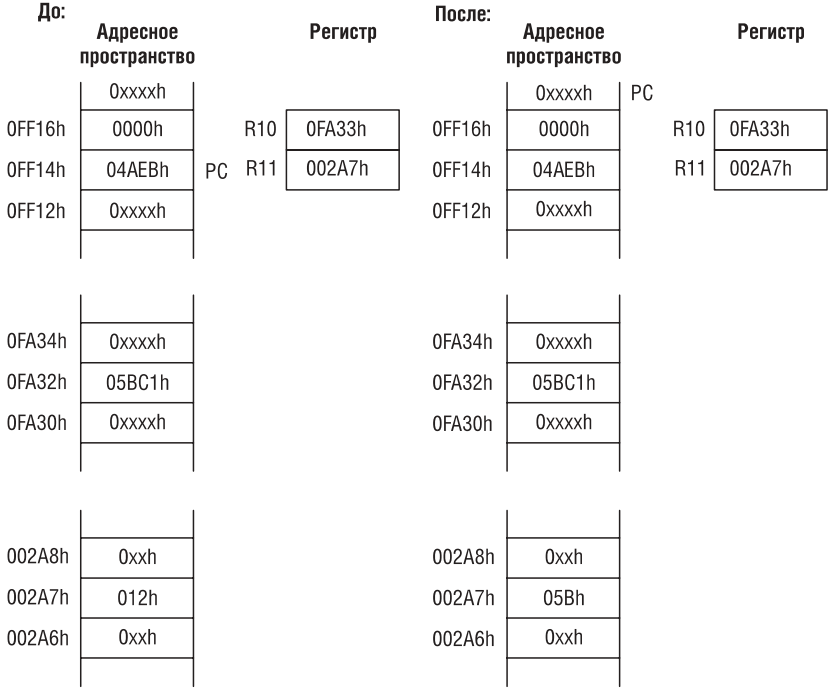
Этот режим адресации предназначен главным образом для аппаратных периферийных модулей, расположенных по абсолютным, фиксированным адресам. Они адресуются в абсолютном режиме, что гарантирует переносимость программы (например, при написании позиционно-независимого, переносимого кода).

3.3.5. Косвенный регистровый режим

Косвенный регистровый режим описан в таблице 3.8.

Таблица 3.8. Описание косвенного режима

Код ассемблера	Содержимое ПЗУ
`MOV @R10, 0(R11)`	`MOV @R10, 0(R11)`
Длина: Одно или два слова	
Операция: Пересылка содержимого с исходного адреса (содержится в R10) по адресу назначения (содержится в R11). Регистры не изменяются.	
Комментарий: Действительно только для операнда источника. В качестве операнда получателя подставляется 0(Rd)	
Пример: `MOV.B @R10, 0(R11)`	

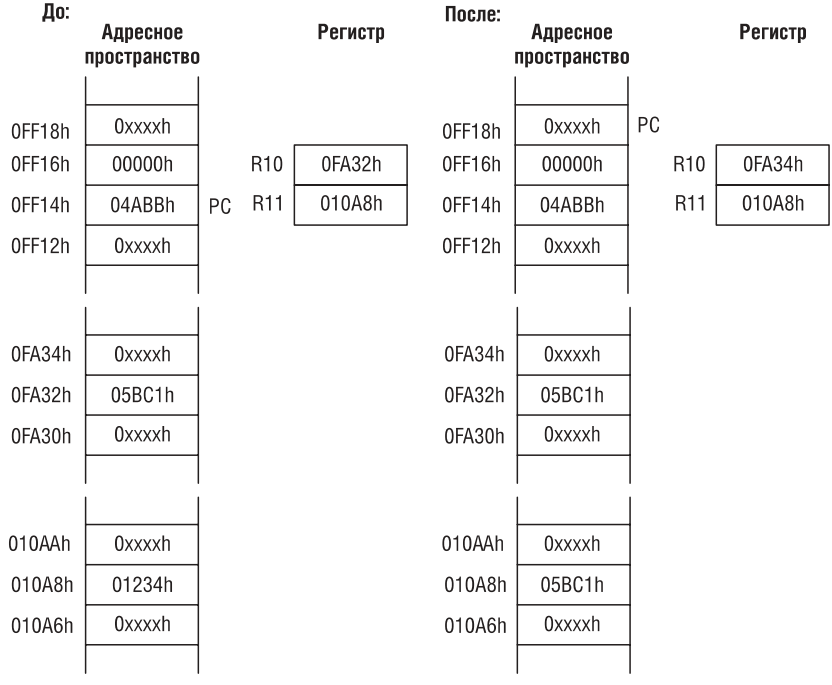


3.3.6. Косвенный автоинкрементный режим

Косвенный автоинкрементный режим описан в таблице 3.9.

Таблица 3.9. Описание косвенного автоинкрементного режима

Код ассемблера	Содержимое ПЗУ
`MOV @R10+, 0(R11)`	`MOV @R10+, 0(R11)`
Длина: Одно или два слова	
Операция: Пересылка содержимого с исходного адреса (содержится в R10) по адресу назначения (содержится в R11). Регистр R10 инкрементируется после выборки на 1 для байтовых операций или на 2 для команд-слов, таким образом указывается следующий адрес без дополнительных действий. Это полезно для обработки таблиц.	
Комментарий: Действительно только для операнда источника. В качестве операнда получателя подставляется 0(Rd) плюс вторая команда INCD Rd.	
Пример: `MOV.B @R10+, 0(R11)`	



Автоинкремент содержимого регистра происходит после выборки операнда. Этот процесс показан на рис. 3.8.

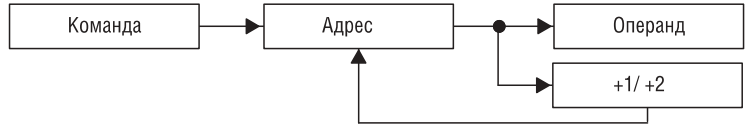


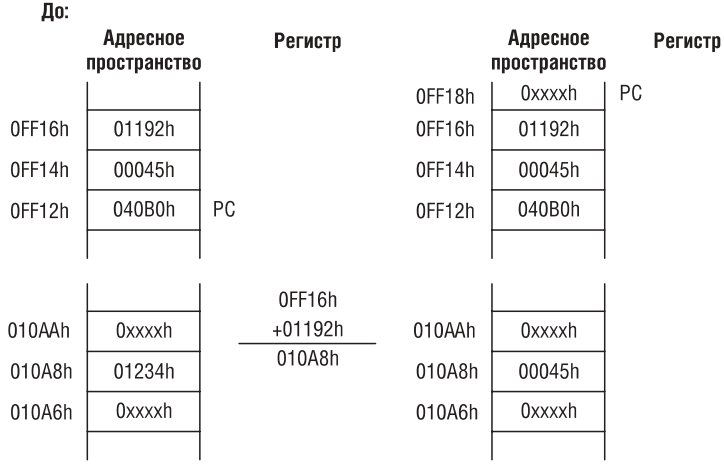
Рис. 3-8. Операция выборки операнда

3.3.7. Прямой режим

Прямой (непосредственный) режим описан в таблице 3.10.

Таблица 3.10. Описание прямого режима

Код ассемблера	Содержимое ПЗУ
`MOV #45h, TONI`	`MOV @PC+, X(PC)45X=TONI-PC`
Длина: Два или три слова. На одно слово меньше, если может использоваться константа генераторов CG1 или CG2.	
Операция: Пересылка непосредственной константы 45h, находящейся в слове, следующем за командой, по адресу назначения TONI. Когда происходит выборка источника, программный счетчик указывает на слово, следующее за командой, и выполняется пересылка содержимого по назначению.	
Комментарий: Действительно только для операнда источника	
Пример: `MOV #45h,TONI`	



3.4. Набор команд

Полный набор команд семейства MSP430 содержит 27 команд ядра и 24 эмулированные команды. Команды ядра – это команды, имеющие уникальный код операции, декодируемый ЦПУ. Эмулированные команды представляют собой инструкции, облегчающие чтение и написание кода, но не имеющие собственного кода операции, поэтому ассемблер автоматически меняет их на эквивалентные команды ядра. Использование эмулированных команд не приводит к увеличению объема кода или снижению производительности.

Существует три формата команд ядра:

- С двойным операндом
- С одиночным операндом
- Команды перехода

Все команды с одним и двумя операндами могут быть командами для работы с байтами или командами для работы со словами, используя, соответственно, расширения «.B» или «.W». Байтовые команды используются для доступа к данным байта или к байту периферийного устройства. Команды-слова используются для доступа к данным слова или к слову периферийного устройства. Если никакое расширение не используется, команда является командой-словом.

Источник и получатель в команде определяются следующими полями:

src	Операнд источника определяется As и S-reg
dst	Операнд получателя определяется Ad D-reg
As	Адресные биты, задающие режим адресации, используемые для источника (src)
S-reg	Рабочий регистр, используемый в качестве источника (src)
Ad	Адресные биты, задающие режим адресации, используемые для получателя (dst)
D-reg	Рабочий регистр, используемый в качестве получателя (dst)
B/W	Операция с байтом или словом:
	0: операция со словом
	1: операция с байтом

Примечание: адрес получателя

Адрес получателя действителен в любом месте карты распределения памяти. Однако, при использовании команды, изменяющей содержимое получателя, пользователь должен быть уверен, что по адресу назначения можно производить запись. К примеру, маскированное ПЗУ имеет правильный адрес назначения, но его содержимое не может модифицироваться, поэтому команда изменения его содержимого не будет правильно выполнена.

Примечание: использование меток EDE и TONI

Везде в документации по семейству MSP430 используются универсальные метки EDE и TONI. Они являются только метками, не имеющими никакого специального назначения.

3.4.1. Команды с двойным операндом (Формат I)

На рис. 9 показана структура формата команды с двойным операндом.

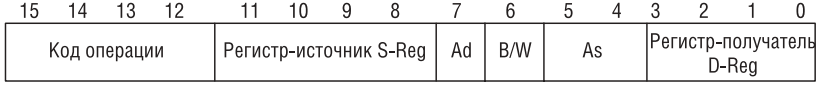


Рис. 3-9. Формат команды с двойным операндом

В таблице 3.11 приведено описание и перечень команд с двойным операндом.

Таблица 3.11. Команды с двойным операндом

Мнемоника	S-Reg, D-Reg	Операция	Биты статуса			
			V	N	Z	C
MOV(.B)	src,dst	src → dst	-	-	-	-
ADD(.B)	src,dst	src + dst → dst	*	*	*	*
ADDC(.B)	src,dst	src + dst + C → dst	*	*	*	*
SUB(.B)	src,dst	dst + .not.src + 1 → dst	*	*	*	*
SUBC(.B)	src,dst	dst + .not.src + C → dst	*	*	*	*
CMP(.B)	src,dst	dst – src	*	*	*	*
DADD(.B)	src,dst	src + dst + C → dst (десятичное)	*	*	*	*
BIT(.B)	src,dst	src .and. dst	**0**	*	*	*
BIC(.B)	src,dst	.not.src .and. dst → dst	-	-	-	-
BIS(.B)	src,dst	src .or. dst → dst	-	-	-	-
XOR(.B)	src,dst	src .xor. dst → dst	*	*	*	*
AND(.B)	src,dst	src .and. dst → dst	**0**	*	*	*
*	Влияет на бит статуса					
-	Не влияет на бит статуса					
0	Бит статуса очищается					
1	Бит статуса устанавливается					

Примечание: Команды CMP и SUB

Команды CMP и SUB идентичны, за исключением сохранения результата. Это также справедливо для команд BIT и AND.

3.4.2. Команды с одним операндом (Формат II)

На рис. 3.10 показана структура формата команды с одним операндом.

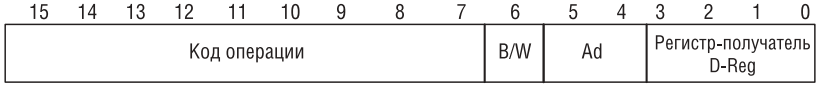


Рис. 3-10. Формат команды с одним операндом

В таблице 3.12 приведено описание и перечень команд с одним операндом.

Таблица 3.12. Команды с одним операндом

Мнемоника	S-Reg, D-Reg	Операция	Биты статуса			
			V	N	Z	C
RRC(.B)	dst	C → MSB → … LSB → C	*	*	*	*
RRA(.A)	dst	MSB → MSB → … LSB → C	0	*	*	*
PUSH(.B)	src	SP-2 → SP, src → @SP	-	-	-	-
SWPB	dst	Обмен байтами	-	-	-	-
CALL	dst	SP-2 → SP, PC+2 → @SP	-	-	-	-
		dst → PC				
RETI		TOS → SR, SP+2 → SP	*	*	*	*
		TOS → PC, SP+2 → SP				
SXT	dst	Бит7 → Бит8 … … Бит15	0	*	*	*
*	Влияет на бит статуса					
-	Не влияет на бит статуса					
0	Бит статуса очищается					
1	Бит статуса установляется					

Для команды CALL возможны все способы адресации. Если используется символический режим (Адрес), прямой режим (#N), абсолютный режим (&EDE) или индексный режим x(RN), следующее за командой CALL слово должно содержать информацию об адресе.

3.4.3. Команды перехода

На рис. 3.11 показан формат команды условного перехода.

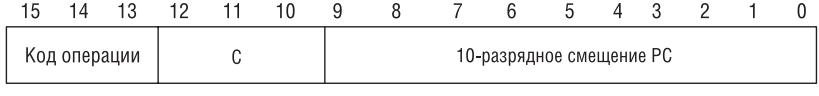


Рис. 3-11. Формат команды условного перехода

В таблице 3.13 приведено описание и перечень команд переходов.

Таблица 3.13. Команды переходов

Мнемоника	S-Reg, D-Reg	Операция
JEQ/JZ	Метка	Переход к метке, если бит нуля (Z) установлен
JNE/JNZ	Метка	Переход к метке, если бит нуля (Z) сброшен
JC	Метка	Переход к метке, если бит переноса (C) установлен
JNC	Метка	Переход к метке, если бит переноса (C) сброшен
JN	Метка	Переход к метке, если бит отрицательного результата (N) установлен
JGE	Метка	Переход к метке, если (N.XOR.V)=0
JL	Метка	Переход к метке, если (N.XOR.V)=1
JMP	Метка	Безусловный переход к метке

Условные переходы обеспечивают ветвление программы относительно программного счетчика PC и не оказывают влияния на биты статуса. Возможный диапазон переходов с помощью команды перехода составляет от -511 до +512 слов относительно текущего значения PC. 10-разряное смещение программного счетчика обрабатывается как 10-разрядное значение со знаком: удваивается и складывается с содержимым программного счетчика:

$$PCnew = PCold + 2 + PCoffset \times 2$$

где: PCnew – новое содержимое программного счетчика;
PCold – исходное содержимое программного счетчика;
PCoffset – 10-разрядная величина смещения программного счетчика.

ADC[.W]	**Сложить бит переноса с получателем**	
ADC.B	**Сложить бит переноса с получателем**	
Синтаксис	`ADC dst или ADC.W dst`	
	`ADC.B dst`	
Операция	`dst + C → dst`	
Эмуляция	`ADDC #0,dst`	
	`ADDC.B #0,dst`	
Описание	Бит переноса (C) складывается с операндом получателя. Предыдущее содержимое получателя теряется.	
Биты статуса	N:	Устанавливается, если результат отрицательный, сбрасывается, если положительный
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается

Биты статуса	C:	Устанавливается, если содержимое получателя dst инкрементируется от 0FFFFh к 0000, в противном случае сбрасывается; Устанавливается, если содержимое получателя dst инкрементируется от 0FFh к 00, в противном случае сбрасывается;
	V:	Устанавливается, если произошло арифметическое переполнение, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Сложение содержимого 16-разрядного счетчика, указанного в R13, с 32-разрядным счетчиком, указанным в R12:
	`ADD @R13,0(R12)  ;сложение LSD`
	`ADC 2(R12)  ;сложение переноса с MSD`	

Пример	Сложение содержимого 8-разрядного счетчика, указанного в R13, с 16-разрядным счетчиком, указанным в R12:
	`ADD.B @R13,0(R12)  ;сложение LSD`
	`ADC.B 1(R12)  ;сложение переноса с MSD`	

ADD [.W]	**Сложение содержимого источника с содержимым получателя**	
ADD.B	**Сложение содержимого источника с содержимым получателя**	
Синтаксис	`ADD src,dst` или `ADD.W src,dst`	
	`ADD.B src,dst`	
Операция	`src + dst → dst`	
Описание	Операнд источника складывается с операндом получателя. Операнд источника не изменяется. Предыдущее содержимое получателя теряется	
Биты статуса	N:	Устанавливается, если результат отрицательный, сбрасывается, если положительный
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается
	C:	Устанавливается, если в результате происходит перенос; очищается, если переноса нет
	V:	Устанавливается, если произошло арифметическое переполнение, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Прибавление 10 к содержимому R5. Выполняется переход к метке TONI, если произошел перенос (установлен бит C):
	`ADD #10, R5`
	`JC  TONI  ;произошел перенос`
	`…  ;переноса нет`	

Пример	Прибавление 10 к содержимому R5. Выполняется переход к метке TONI, если произошел перенос (установлен бит C):
	`ADD.B #10, R5 ;прибавление 10 к младшему байту R5`
	`JC TONI  ;перенос произошел, если (R5) ≥ 246`
	`;[0Ah+0F6h]`
	`…  ;переноса нет`	

ADDC[.W]	**Сложение содержимого источника и переноса с содержимым получателя**	
ADDC.B	**Сложение содержимого источника и переноса с содержимым получателя**	
Синтаксис	`ADDC src,dst` или `ADDC.W src,dst`	
	`ADDC.B src,dst`	
Операция	`src + dst + C` → `dst`	
Описание	Операнд источника и бит переноса (C) складываются с операндом получателя. Операнд источника не изменяется. Предыдущее содержимое получателя теряется	
Биты статуса	N:	Устанавливается, если результат отрицательный, сбрасывается, если положительный
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается
	C:	Устанавливается, если произошел перенос из MSB результата; сбрасывается, если переноса нет
	V:	Устанавливается, если произошло арифметическое переполнение, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Прибавление содержимого 32-разрядного счетчика, указанного в R13, к 32-разрядному счетчику, расположенному на одиннадцать слов (20/2 + 2/2) выше указанного в R13:
	`ADD @R13+, 20(R13) ;сложение LSD-байтов без учета`
	`;переноса`
	`ADDC @R13+, 20(R13) ;сложение MSD с учетом переноса`
	`;в результате`
	`… ;предыдущей команды сложения LSD`	

Пример	Прибавление содержимого 24-разрядного счетчика, указанного в R13, к 24-разрядному счетчику, расположенному на одиннадцать слов выше указанного в R13:
	`ADD.B @R13+, 10(R13) ;сложение LSD-байтов без учета`
	`;переноса`
	`ADDC.B @R13+, 10(R13) ;сложение средних битов`
	`;с переносом`
	`ADDC.B @R13+, 10(R13) ;сложение MSD с учетом переноса`
	`;в результате`
	`… ;предыдущей команды сложения LSD`	

AND[.W]	**Логическое «И» источника и получателя**
AND.B	**Логическое «И» источника и получателя**
Синтаксис	`AND src,dst` или `AND.W src,dst`
	`AND.B src,dst`
Операция	`src .AND. dst` → `dst`
Описание	Над операндом источника и операндом получателя выполняется операция логического «И» (логическое умножение). Результат остается в получателе.

Биты статуса	N:	Устанавливается, если в результате устанавливается MSB, сбрасывается, если не устанавливается
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается
	C:	Устанавливается, если результат не «0»; в противном случае сбрасывается (=.NOT. Zero)
	V:	Сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Установка битов в R5 с использованием маски (#0AA55h) для слова, адресованного меткой TOM. Если результат «0», выполняется переход к метке TONI:
	`MOV #0AA55h, R5 ;загрузка маски в регистр R5`
	`AND R5, TOM ;маскирование слова, адресованного`
	`;TOM, с помощью регистра R5`
	`JZ TONI ;`
	`… ;результат не «0»`
	`;`
	`;`
	`;или`
	`;`
	`;`
	`AND #0AA55h,TOM`
	`JZ TONI`	

Пример	Логическое перемножение битов маски #0A5h с младшим байтом TOM. Если результат «0», выполняется переход к метке TONI:
	`AND.B #0A5,TOM ;маскирование младшего байта маской`
	`;#0A5h`
	`JZ TONI ;`
	`… ;результат не «0»`	

BIC[.W]	**Очистка битов получателя**
BIC.B	**Очистка битов получателя**
Синтаксис	`BIC src,dst` или `BIC.W src,dst`
	`BIC.B src,dst`
Операция	`.NOT.src .AND. dst → dst`
Описание	Инвертированный операнд источника и операнд получателя логически перемножаются. Результат помещается в получатель. Операнд источника не изменяется.
Биты статуса	Биты статуса не изменяются
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются

Пример	Очистка шести битов MSB слова LEO в ОЗУ:
	`BIC #0FC00h,LEO ;очистка 6-ти битов MSB в слове`
	`;памяти (LEO)`

Пример	Очистка пяти битов MSB байта LEO в ОЗУ:
	`BIC.B #0F8h,LEO ;очистка 5-ти битов MSB в ОЗУ`
	`;в байте LEO`

BIS[.W]	**Установка битов получателя**
BIS.B	**Установка битов получателя**
Синтаксис	`BIS src,dst` или `BIS.W src,dst`
	`BISC.B src,dst`
Операция	`src .OR. dst` → `dst`
Описание	Над операндом источника и операндом получателя выполняется операция логического «ИЛИ» (логическое сложение). Результат помещается в получатель. Операнд источника не изменяется.
Биты статуса	Биты статуса не изменяются
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются

Пример	Установка шести LSB-битов слова TOM в ОЗУ:
	`BIS #003Fh, TOM ;установка 6-ти LSB-битов слова TOM`
	`;в ОЗУ`

Пример	Установка трех MSB-битов байта LEO в ОЗУ:
	`BIS.B #0E0h,TOM ;установка 3-х MSB-битов в байте`
	`;TOM в ОЗУ`

BIT[.W]	**Проверка битов получателя**	
BIT.B	**Проверка битов получателя**	
Синтаксис	`BIT src,dst` или `BIT.W src,dst`	
Операция	`src .AND. dst`	
Описание	Над операндом источника и операндом получателя выполняется операция логического «И» (логическое умножение). Результат влияет только на биты статуса. Операнды источника и получателя не изменяются.	
Биты статуса	N:	Устанавливается, если установлен MSB результата, иначе сбрасывается
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается
	C:	Устанавливается, если результат не «0»; в противном случае сбрасывается (.NOT. Zero)
	V:	Сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Если бит 9 регистра R8 установлен, выполняется переход к метке TOM:
	`BIT #0200h, R8 ;бит 9 регистра R8 установлен?`
	`JNZ TOM ;Да, переход к метке TOM`
	`… ;Нет, продолжение программы`	

Пример	Если бит 3 регистра R8 установлен, выполняется переход к метке TOM: `BIT #8, R8JC TOM`	

Пример	Проверяется бит приема (RCV) при последовательной передаче данных. Поскольку при использовании команды BIT для проверки одного бита содержимое бита переноса эквивалентно состоянию проверяемого бита, оно используется в следующей команде; прочитанная информация сдвигается в регистр RECBUF:
	`;`
	`;Последовательная передача данных, начиная с младшего бита (LSB), сдвиг которого происходит в первую очередь:`
	`;xxxx xxxx xxxx xxxx`
	`BIT.B #RCV,RCCTL ;Информационный бит в бите переноса`
	`RRC RECBUF ;Бит переноса → в MSB регистра`
	`;RECBUF`
	`;cxxx xxxx`
	`… ;повтор двух предыдущих команд`
	`… ;8 раз`
	`;cccc cccc`
	`;^ ^ ;`
	`MSB LSB ;Последовательная передача данных,`
	`;начиная со старшего бита (MSB),`
	`;сдвиг которого происходит в первую`
	`;очередь:`
	`BIT.B #RCV, RCCTL ;Информационный бит в бите переноса`
	`RLC.B RECBUF ;Бит переноса → в LSB регистра`
	`;RECBUF`
	`;xxxx xxxc`
	`… ;повтор двух предыдущих команд`
	`… ;8 раз`
	`;cccc cccc`
	`;LSB`
	`;MSB`

***BR, BRANCH**	**Переход к … месту назначения**
Синтаксис	`BR dst`
Операция	`dst → PC`
Эмуляция	`MOV dst, PC`
Описание	Безусловный переход выполняется в любое место 64 кБайт адресного пространства. Могут использоваться все способы адресации. Команда перехода – это команда-слово.
Биты статуса	Биты статуса не изменяются

Пример	Примеры для всех режимов адресации:
	`BR #EXEC ;Переход к метке EXEC или прямой переход (например, #0A4h)`
	`;Команда ядра MOV @PC+,PC`
	`BR EXEC ;Переход по адресу, содержащемуся в EXEC`
	`;Команда ядра MOV X(PC),PC`
	`;Косвенный адрес`

Пример	`BR &EXEC ;Переход по адресу, содержащемуся`
	`;в абсолютном адресе EXEC`
	`;Команда ядра MOV X(0),PC`
	`;Косвенный адрес`
	`BR R5 ;Переход по адресу, содержащемуся в R5`
	`;Команда ядра MOV R5,PC`
	`;Косвенная адресация по содержимому R5BR @R5`
	`;Переход по адресу, содержащемуся в слове,`
	`;указанном врегистре R5`
	`;Команда ядра MOV @R5,PC`
	`;Косвенная адресация по косвенному`
	`;содержимому R5`
	`BR @R5+ ;Переход по адресу, содержащемуся в слове,`
	`;указанном в регистре R5 и последующий`
	`;инкремент указателя в R5.`
	`;При следующем использовании указателя R5`
	`;программным потоком выполнение программы`
	`;может измениться, поскольку будет`
	`;использован следующей адрес`
	`;в таблице, указанной регистром R5`
	`;Команда ядра MOV @R5,PC`
	`;Косвенная адресация по косвенному`
	`;содержимому R5 с автоинкрементом`
	`BR X(R5) ;Переход по адресу, содержащемуся в адресе,`
	`;указанном выражением R5+X (например,`
	`;таблица со стартовым адресом X). «X» может`
	`;быть адресом или меткой`
	`;Команда ядра MOV X(R5),PC`
	`;Косвенная адресация по косвенному`
	`;содержимому R5 + X`

CALL	**Вызов подпрограммы**
Синтаксис	`CALL dst`
Операция	`dst → tmp dst оценивается и сохраняется`
	`SP – 2 → SP`
	`PC → @SP  PC сохраняется на вершине стека (TOS)`
	`tmp → PC dst записывается в PC`
Описание	Вызов подпрограммы может производиться по любому адресу в пределах 64 кБайт адресного пространства. Могут использоваться все способы адресации. Адрес возврата (адрес следующей команды) сохраняется в стеке. Команда вызова подпрограммы – это команда-слово.
Биты статуса	Биты статуса не изменяются

Пример	Примеры для всех режимов адресации:
	`CALL #EXEC ;Вызов с метки EXEC или прямая адресация`
	`;(например, #0A4h)`
	`;SP-2 → SP, PC+2 → @SP, @PC+ → PC`

Пример	CALL EXEC ;Вызов по адресу, содержащемуся в EXEC ;SP-2 → SP, PC+2 → @SP, X(PC) → PC ;Косвенная адресация CALL &EXEC ;Вызов по адресу, содержащемуся ;в абсолютном адресе EXEC ;SP-2 → SP, PC+2 → @SP, X(0) → PC ;Косвенная адресация CALL R5 ;Вызов по адресу, содержащемуся в R5 ;SP-2 → SP, PC+2 → @SP, R5 → PC ;Косвенная адресация по содержимому R5 CALL @R5 ;Вызов по адресу, содержащемуся в слове, ;указанном в регистре R5 ;SP-2 → SP, PC+2 → @SP, @R5 → PC ;Косвенная адресация по косвенному ;содержимому R5 CALL @R5+ ;Вызов по адресу, содержащемуся ;в слове, указанном в регистре R5 ;и последующий инкремент указателя в R5. ;При следующем использовании указателя R5 ;программным потоком выполнение программы ;может измениться, поскольку будет ;использован следующей адрес в таблице, ;указанной регистром R5 ;SP-2 → SP, PC+2 → @SP, @R5 → PC ;Косвенная адресация по косвенному ;содержимому R5 с автоинкрементом CALL X(R5) ;Вызов по адресу, содержащемуся в адресе, ;указанном выражением R5+X (например, ;таблица со стартовым адресом X). ;«X» может быть адресом или меткой. ;SP-2 → SP, PC+2 → @SP, X(R5) → PC ;Косвенная адресация по косвенному ;содержимому R5 + X

***CLR[.W]**	**Очистка получателя**
***CLR.B**	**Очистка получателя**
Синтаксис	CLR dst или CLR.W dst
	CLR.B dst
Операция	0 → dst
Эмуляция	MOV #0, dst
	MOV.B #0, dst
Описание	Операнд получателя очищается
Биты статуса	Биты статуса не изменяются
Пример	Очистка слова TONI в ОЗУ: CLR TONI ;0 → TONI
Пример	Очистка регистра R5: CLR R5
Пример	Очистка байта TONI в ОЗУ: CLR.B TONI ;0 → TONI

***CLRC**	**Очистка бита переноса**	
Синтаксис	CLRC	
Операция	0 → C	
Эмуляция	BIC #1, SR	
Описание	Бит переноса (C) очищается. Команда очистки переноса – это команда-слово.	
Биты статуса	N:	Не изменяется
	Z:	Не изменяется
	C:	Очищается
	V:	Не изменяется
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Прибавление содержимого 16-разрядного десятичного счетчика, указанного в R13 к содержимому 32-разрядного счетчика, указанного в R12:
	`CLRC ;C=0: определение исходного`
	`;состояния`
	`DADD @R13,0(R12) ;сложение 16-разрядного счетчика`
	`;с младшим словом 32-разрядного`
	`;счетчика`
	`DADC 2(R12) ;прибавление переноса к старшему`
	`;слову 32-разрядного счетчика`	

***CLRN**	**Очистка бита отрицания**	
Синтаксис	CLRN	
Операция	0 → N или (.NOT.src .AND. dst → dst)	
Эмуляция	BIC #4, SR	
Описание	Константа 04h инвертируется (0FFFBh) и логически умножается (AND) с операндом получателя. Результат помещается в получатель. Команда очистки бита отрицания – это команда-слово.	
Биты статуса	N:	Сбрасывается в «0»
	Z:	Не изменяется
	C:	Не изменяется
	V:	Не изменяется
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	
Пример	Бит отрицания в регистре статуса очищается. Это позволяет избежать специальной обработки отрицательных чисел вызываемой подпрограммы.	

SUBR	`CLRN`
	`CALL SUBR`
	`…`
	`JN SUBRET ;если при входе – отрицательное значение,`
	`;ничего не делается и происходит выход`
	`;из подпрограммы`
	`…`	

SUBRET	RET	

*CLRZ	Очистка бита нулевого результата	
Синтаксис	CLRZ	
Операция	0 → Z или (.NOT.src .AND. dst → dst)	
Эмуляция	BIC #2, SR	
Описание	Константа 02h инвертируется (0FFFDh) и логически умножается (AND) с операндом получателя. Результат помещается в получатель. Команда очистки бита нуля – это команда-слово.	
Биты статуса	N:	Не изменяется
	Z:	Сбрасывается в «0»
	C:	Не изменяется
	V:	Не изменяется
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	
Пример	Бит нуля в регистре статуса очищается. CLRZ	

CMP[.W]	Сравнение источника и получателя	
CMP.B	Сравнение источника и получателя	
Синтаксис	CMP src,dst или CMP.W src,dst	
	CMP.B src, dst	
Операция	dst + .NOT.src + 1 или (dst – src)	
Описание	Операнд источника вычитается из операнда получателя. Это выполняется прибавлением дополнения до единицы операнда источника плюс 1. Оба операнда не изменяются, а результат не сохраняется, изменяются только биты статуса.	
Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный (src >= dst)
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается (src = dst)
	C:	Устанавливается, если произошел перенос из MSB результата, в противном случае сбрасывается
	V:	Устанавливается, если произошло арифметическое переполнение, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	
Пример	Сравнивается содержимое R5 и R6. Если оно одинаково, выполнение программы продолжается с метки EQUAL. CMP R5,R6 ;R5=R6? JEQ EQUAL ;Да, переход к метке EQUAL	
Пример	Сравниваются два блока в ОЗУ. Если они не эквивалентны, программа переходит к метке ERROR. L$1 MOV #NUM, R5 ;количество слов, которые ;будут сравниваться MOV #BLOCK1,R6 ;начальный адрес BLOCK1 ;в регистр R6	

Пример	`L$1 MOV #BLOCK2,R7 ;начальный адрес BLOCK2`
	`;в регистр R7`
	`CMP @R6+,0(R7) ;сравнение содержимого`
	`;ячеек и инкремент R6`
	`JNZ ERROR ;если не равны, переход`
	`;к метке ERROR`
	`INCD R7 ;если равны инкремент R7`
	`DEC R5 ;декрементировать R5`
	`JNZ L$1 ;если сравнение не заверше-`
	`;но – продолжить`

Пример	Сравниваются байты в ОЗУ, адресованные метками EDE и TONI. Если они одинаковы, выполнение программы продолжается с метки EQUAL.
	`CMP.B EDE, TONI ;MEM(EDE)=MEM(TONI)?`
	`JEQ EQUAL ;Да,переход к метке EQUAL`

***DADC[.W]**	**Десятичное сложение переноса с получателем**	
***DADC.B**	**Десятичное сложение переноса с получателем**	
Синтаксис	`DADC dst или DADC.W src,dst`	
	`DADC.B dst`	
Операция	`dst + C → dst (десятичное)`	
Эмуляция	`DADD #0, dst`	
	`DADD.B #0, dst`	
Описание	Бит переноса (C) десятично прибавляется к получателю	
Биты статуса	N:	Устанавливается, если MSB равен «1»
	Z:	Устанавливается, если dst равен «0»; в противном случае сбрасывается
	C:	Устанавливается, если получатель инкрементируется от 9999 до 0000; в противном случае сбрасывается. Устанавливается, если получатель инкрементируется от 99 до 00; в противном случае сбрасывается
	V:	Не определено
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Десятичное число из четырех цифр, содержащееся в регистре R5 прибавляется к десятичному числу из восьми цифр, указанному в регистре R8.
	`CLRC ;сброс переноса`
	`;стартовое условие для следующих`
	`;команд задано`
	`DADD R5,0(R8) ;сложение LCDs и переноса`
	`DADC 2(R8) ;прибавление переноса к MSD`	

Пример	Десятичное число из двух цифр, содержащееся в регистре R5 прибавляется к десятичному числу из четырех цифр, указанному в регистре R8.
	`CLRC ;сброс переноса`
	`;стартовое условие для следующих`
	`;команд задано`
	`DADD.B R5,0(R8) ;сложение LCDs и переноса`
	`DADC 1(R8) ;прибавление переноса к MSD`	

DADD[.W]	Десятичное сложение источника, переноса и получателя	
DADD.B	Десятичное сложение источника, переноса и получателя	
Синтаксис	DADD src,dst или DADD.W src,dst	
	DADD.B src, dst	
Операция	src + dst + C → dst (десятичное)	
Описание	Операнды источника и получателя обрабатываются как четыре двоично-десятичных числа (BCD – Binary Coded Decimal) с положительными знаками. Операнд источника и бит переноса (C) десятично прибавляются к операнду получателя. Операнд источника не изменяется. Предыдущее содержимое получателя теряется. Для чисел, представленных не в BCD-формате, результат не определен.	
Биты статуса	N:	Устанавливается, если MSB равен «1»; сбрасывается в противном случае
	Z:	Устанавливается, если результат равен «0»; в противном случае сбрасывается
	C:	Устанавливается, если результат превышает 9999Устанавливается, если результат превышает 99
	V:	Не определено
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	BCD-число из восьми цифр, содержащееся в регистрах R5 и R6, десятично прибавляется к BCD-числу из восьми цифр, содержащемуся в регистрах R3 и R4 (регистры R6 и R4 содержат MSD).
	`CLRC ;очистка переноса`
	`DADD R5,R3 ;сложение LSDs`
	`DADD R6,R4 ;сложение MSDs и переноса`
	`JC OVERFLOW ;если произошел перенос, выполняется`
	`;переход в подпрограмму обработки`
	`;ошибок`	

Пример	Десятичный счетчик из двух цифр в байте ОЗУ с меткой «CNT» инкрементируется на единицу.
	`CLRC ;сброс переноса`
	`DADD.B #1,CNT ;инкремент десятичного счетчика`
	или
	`SETC`
	`DADD.B #0,CNT ;≡ DADC.B CNT`	

*DEC[.W]	Декремент получателя
*DEC.B	Декремент получателя
Синтаксис	DEC dst или DEC.W dst
	DEC.B dst
Операция	dst - 1 → dst
Эмуляция	SUB #1, dst
	SUB.B #1, dst

Описание	Операнд получателя уменьшается (декрементируется) на единицу. Исходное содержимое теряется.	
Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный
	Z:	Устанавливается, если dst содержал «1»; в противном случае сбрасывается
	C:	Сбрасывается, если получатель содержал «0»; в противном случае устанавливается
	V:	Устанавливается, если произошло арифметическое переполнение; в противном случае сбрасывается; Устанавливается, если исходное значение получателя было 08000h, в противном случае сбрасывается; Устанавливается, если исходное значение получателя было 080h, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	
Пример	Содержимое регистра R10 декрементируется на 1. `DEC R10 ;декремент R10`	

Пересылка блока из 255 байт, расположенного в памяти начиная с адреса, указанного меткой EDE, в область памяти, начало которой указано меткой TONI. Таблицы не должны наложиться: стартовый адрес назначения TONI должен находиться вне диапазона от EDE до EDE+0FEh.

```
    MOV    #EDE,R6
    MOV    #255,R10
L$1 MOV.B  @R6+,TONI-EDE-1(R6)
    DEC    R10
    JNZ    L$1
```

Не следует перемещать таблицы, используя приведенную выше подпрограмму; с перекрытием, показанным на рис. 3.12.

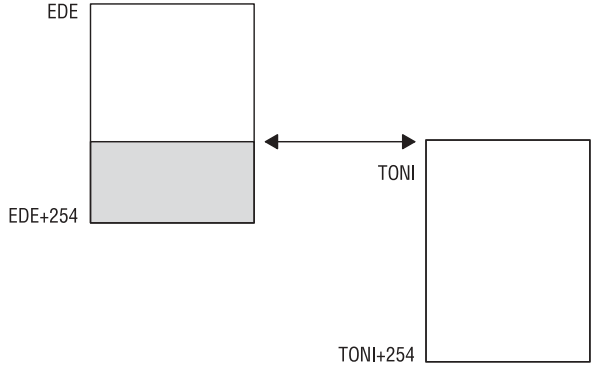


Рис. 3-12. Перекрытие (наложение) при декременте

*DECD[.W]	Двойной декремент получателя	
*DECD.B	Двойной декремент получателя	
Синтаксис	DECD dst или DECD.W dst	
	DECD.B dst	
Операция	dst - 2 → dst	
Эмуляция	SUB #2, dst	
	SUB.B #2, dst	
Описание	Операнд получателя уменьшается (декрементируется) на два. Исходное содержимое теряется.	
Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный
	Z:	Устанавливается, если dst содержал «2»; в противном случае сбрасывается
	C:	Сбрасывается, если получатель содержал «0»; в противном случае устанавливается
	V:	Устанавливается, если произошло арифметическое переполнение; в противном случае сбрасывается; Устанавливается, если исходное значение получателя было 08001h или 08000h, в противном случае сбрасывается; Устанавливается, если исходное значение получателя было 081h или 080h, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	
Пример	Содержимое регистра R10 декрементируется на 2. DECD R10 ;декремент R10 на два	

Пересылка блока из 255 слов, расположенного в памяти начиная с адреса, указанного меткой EDE, в область памяти, начало которой указано меткой TONI. Таблицы не должны наложиться: стартовый адрес назначения TONI должен находиться вне диапазона от EDE до EDE+0FEh.

```
     MOV    #EDE, R6
     MOV    #510, R10
L$1  MOV    @R6+, TONI-EDE-2(R6)
     DECD   R10
     JNZ    L$1
```

Пример	Содержимое ячейки памяти LEO декрементируется на два. DECD.B LEO ;декремент MEM(LEO) Декремент байта статуса STATUS на два. DECD.B STATUS

*DINT	Запрещение (общее) прерываний
Синтаксис	DINT
Операция	0 → GIE или (0FFF7h .AND. SR → SR/.NOT.src .AND. dst → dst)
Эмуляция	BIC #8,SR

Описание	Все прерывания запрещаются. Константа 08h инвертируется и логически перемножается с регистром статуса (SR). Результат помещается в регистр статуса SR.
Биты статуса	Биты статуса не изменяются
Биты режима	GIE сбрасывается. Биты OSCOFF и CPUOFF не изменяются.

Пример	Бит общего разрешения прерываний в регистре статуса очищается, что позволяет без повреждения переслать содержимое 32-разрядного счетчика. Это гарантирует, что содержимое счетчика не будет изменено во время пересылки возникновением какого-либо прерывания.
	`DINT ;с помощью бита GIE запрещаются все`
	`;прерывания`
	`NOP`
	`MOV COUNTHI, R5 ;копирование счетчика`
	`MOV COUNTLO, R6`
	`EINT ;с помощью бита GIE разрешаются все`
	`;прерывания`

Примечание: запрет прерываний

Если какую-либо последовательность кода нужно защитить от прерывания, после команды DINT должна быть выполнена, по крайней мере, одна команда до начала выполнения этой последовательности, или же следующей командой после DINT должна быть инструкция NOP.

***EINT**	**Разрешение (общее) прерываний**
Синтаксис	`EINT`
Операция	`1 → GIE или (0008h .OR. SR → SR / .src .OR. dst → dst)`
Эмуляция	`BIS #8,SR`
Описание	Все прерывания разрешаются. Константа #08h и регистр статуса SR логически складываются (OR). Результат помещается в регистр статуса SR.
Биты статуса	Биты статуса не изменяются
Биты режима	GIE устанавливается. Биты OSCOFF и CPUOFF не изменяются.
Пример	Бит общего разрешения прерываний (GIE) в регистре статуса устанавливается.

Подпрограмма обработки прерывания портов с P1.2 по P1.7
P1IN – это адрес регистра, в котором читаются все биты порта.
P1IFG – это адрес регистра, в котором фиксируются все события, вызывающие прерывания

```
          PUSH.B &P1IN
          BIC.B  @SP, &P1IFG  ;сброс только принятых флагов
          EINT                ;Предварительно установленные флаги
                              ;прерывания порта 0 сохранены
                              ;на стеке, поэтому допустимы другие
                              ;прерывания
          BIT    #Mask, @SP
          JEQ    MaskOK       ;переход, если флаги идентичны
                              ;представленной маске
```

```
            …
MaskOK      BIC     #Mask,@SP
            …
            INCD    SP              ;Вспомогательное действие, обратное
                                    ;команде PUSH, использованной
                                    ;в начале процедуры обработки
                                    ;прерывания. Корректирует указатель
                                    ;стека для правильного выхода из
                                    ;процедуры обработки прерывания
            RETI
```

Примечание: разрешение прерываний

Команда, следующая за командой разрешения прерываний (EINT), выполняется всегда, даже когда ранее поступивший запрос на обслуживание прерывания ожидает, когда прерывания будут разрешены.

***INC[.W]**	**Инкремент получателя**	
***INC.B**	**Инкремент получателя**	
Синтаксис	INC dst или INC.W dst	
	INC.B dst	
Операция	dst + 1 → dst	
Эмуляция	ADD #1, dst	
Описание	Операнд получателя инкрементируется на единицу. Исходное содержимое теряется.	
Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный
	Z:	Устанавливается, если dst содержал 0FFFFh, в противном случае сбрасывается; Устанавливается, если dst содержал 0FFh, в противном случае сбрасывается
	C:	Устанавливается, если dst содержал 0FFFFh, в противном случае сбрасывается; Устанавливается, если dst содержал 0FFh, в противном случае сбрасывается
	V:	Устанавливается, если dst содержал 07FFFh, в противном случае сбрасывается; Устанавливается, если dst содержал 07Fh, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	
Пример	Байт статуса процесса STATUS инкрементируется. Если результат равен 11, происходит переход к метке OVFL. INC.B STATUS CMP.B #11,STATUS JEQ OVFL	

***INCD[.W]**	**Двойной инкремент получателя**	
***INCD.B**	**Двойной инкремент получателя**	
Синтаксис	INCD dst или INCD.W dst	
	INCD.B dst	
Операция	dst + 2 → dst	
Эмуляция	ADD #2, dst	
Эмуляция	ADD.B #2, dst	
Описание	Операнд получателя инкрементируется на два. Исходное содержимое теряется.	
Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный
	Z:	Устанавливается, если dst содержал 0FFFEh, в противном случае сбрасывается; Устанавливается, если dst содержал 0FEh, в противном случае сбрасывается
	C:	Устанавливается, если dst содержал 0FFFEh или 0FFFFh, в противном случае сбрасывается; Устанавливается, если dst содержал 0FEh или 0FFh, в противном случае сбрасывается
	V:	Устанавливается, если dst содержал 07FFEh или 07FFFh, в противном случае сбрасывается; Устанавливается, если dst содержал 07Eh или 07Fh, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Удаление элемента с вершины стека (TOS) без использования регистра
	`....`
	`PUSH R5 ;содержимое регистра R5, является`
	`;результатом вычисления, сохраненного`
	`;в системном стеке`
	`INCD SP ;удаление элемента TOS путем двойного`
	`;инкрементирования стека. Команду INCD.B`
	`;для этих целей использовать нельзя,`
	`;поскольку SP – это регистр размером в`
	`;слово.`
	`RET`	

Пример	Байт на вершине стека инкрементируется на два.
	`INCD.B 0(SP) ;байт на вершине стека (TOS)`
	`;инкрементируется на два`	

***INV[.W]**	**Инвертирование получателя**
***INV.B**	**Инвертирование получателя**
Синтаксис	INV dst
	INV.B dst
Операция	.NOT.dst → dst
Эмуляция	XOR #0FFFFh, dst
Эмуляция	XOR.B #0FFh, dst

Описание	Операнд получателя инвертируется. Исходное содержимое теряется.	
Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный
	Z:	Устанавливается, если dst содержал 0FFFFh, в противном случае сбрасывается; Устанавливается, если dst содержал 0FFh, в противном случае сбрасывается
	C:	Устанавливается, если результат не ноль, в противном случае сбрасывается (= .NOT. Zero) Устанавливается, если результат не ноль, в противном случае сбрасывается (= .NOT. Zero)
	V:	Устанавливается, если исходное содержимое операнда было отрицательное, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Над содержимым регистра R5 выполняется операция отрицания (дополнение до двух).
	`MOV #00AEh,R5 ; R5=000AEh`
	`INV R5 ;инвертирование R5, R5=0FF51h`
	`INC R5 ;теперь R5 инвертирован, R5=0FF52h`	

Пример	Над содержимым байта памяти LEO выполняется операция отрицания.
	`MOV #0AEh,LEO ; MEM(LEO)=0AEh`
	`INV.B LEO ;инвертирование LEO, MEM(LEO)=051h`
	`INC.B LEO ;MEM(LEO) инвертирован, MEM(LEO)=052h`	

JC	**Переход, если перенос установлен**
JHS	**Переход, если наивысший* или равный**
Синтаксис	`JC label`
	`JHS label`
Операция	`Если C=1: PC+2 × смещение → PC`
	`Если C=0: выполняется следующая команда`
Описание	Проверяется бит переноса (C) регистра статуса. Если он установлен, 10-разрядная величина смещения со знаком, содержащаяся в младших битах (LSB) команды прибавляется к счетчику команд. Если бит переноса C сброшен, выполняется команда, следующая за инструкцией `jump`. Команда JC (переход, если перенос / наивысший или равный) используется для сравнения чисел без знака (от 0 до 65536).
Биты статуса	Биты статуса не изменяются

Пример	Использование сигнала P1IN.1 для задания и управления ходом программы.
	`BIT #01h,&P1IN ;состояние сигнала → в бит переноса`
	`JC PROGA ;Если бит переноса равен 1,`
	`;выполняется программная процедура A`
	`… ;Если бит переноса равен 0,`
	`;выполнение программы продолжается`
	`;здесь`

Пример	Содержимое R5 сравнивается с числом 15. Если содержимое наивысшее или такое же, происходит переход к метке LABEL.
	`CMP #15, R5`
	`JHS LABEL ;Если R5≥15, происходит переход`
	`… ;Продолжение с этого места, если R5<15`

* В оригинале используется термин «higher». Таким образом, подчеркивается, что эта команда позволяет выполнять сравнение чисел без знака, в отличие от команды JGE (Jump if Greater or Equal – переход, если *больше* или равно), с помощью которой можно сравнивать числа со знаком.	

JEQ, JZ	**Переход, если равно; переход, если ноль**
Синтаксис	`JEQ label, JZ label`
Операция	`Если Z=1: PC+2 × смещение → PC`
	`Если Z=0: выполняется следующая команда`
Описание	Проверяется бит нуля (Z) регистра статуса. Если он установлен, 10-разрядная величина смещения со знаком, содержащаяся в младших битах (LSB) команды прибавляется к счетчику команд. Если бит нуля Z не установлен, выполняется команда, следующая за инструкцией `jump`.
Биты статуса	Биты статуса не изменяются

Пример	Переход по адресу, содержащемуся в TONI, если R7 содержит ноль.
	`TST R7 ;Проверка содержимого регистра R7`
	`JZ TONI ;Переход, если «ноль»`

Пример	Переход по адресу LEO, если содержимое R6 равно содержимому таблицы.
	`CMP R6, Table(R5) ;Сравнение содержимого регистра R6`
	`;с содержимым памяти (адрес`
	`;таблицы содержится в R5)`
	`JEQ LEO ;Переход, если данные равны`
	`… ;Нет, данные не равны, выполнение`
	`;программы продолжается здесь`

Пример	Переход к метке LABEL, если содержимое R5 равно нулю.
	`TST R5`
	`JZ LABEL`
	…

JGE	**Переход, если больше или равно**
Синтаксис	JGE label
Операция	`Если (N .XOR. V)=0, то переход к метке: PC+2 × смещение → PC`
	`Если (N .XOR. V)=1, то выполняется следующая команда`
Описание	Проверяются бит отрицания (N) и бит переполнения (V) в регистре статуса. Если они оба установлены или сброшены, 10-разрядная величина смещения со знаком, содержащаяся в младших битах (LSB) команды прибавляется к счетчику команд. Если установлен только один бит, выполняется команда, следующая за инструкцией `jump`. Это позволяет сравнивать числа со знаком.
Биты статуса	Биты статуса не изменяются

Пример	Если содержимое регистра R6 больше или равно содержимому памяти по адресу, указанному в R7, выполнение программы продолжается с метки EDE.
	`CMP @R7, R6 ;R6≥(R7)?, сравнение чисел со знаком`
	`JGE EDE ;Да, R6≥(R7); переход к метке EDE`
	`… ;Нет, продолжение программы`
	…

JL	**Переход, если меньше**
Синтаксис	`JL label`
Операция	`Если (N .XOR. V)=1, то переход к метке: PC+2 × смещение → PC`
	`Если (N .XOR. V)=0, то выполняется следующая команда`
Описание	Проверяются бит отрицания (N) и бит переполнения (V) в регистре статуса. Если установлен только один из них, 10-разрядная величина смещения со знаком, содержащаяся в младших битах (LSB) команды прибавляется к счетчику команд. Если оба бита N и V установлены или сброшены, выполняется команда, следующая за инструкцией `jump`. Это позволяет сравнивать числа со знаком.
Биты статуса	Биты статуса не изменяются

Пример	Если содержимое регистра R6 меньше содержимого памяти по адресу, указанному в R7, выполнение программы продолжается с метки EDE.
	`CMP @R7,R6 ;R6<(R7)?, сравнение чисел со знаком`
	`JL EDE ;Да, R6<(R7) ;переход к метке EDE`
	`… ;Нет, продолжение программы`
	…
	…

JMP	**Безусловный переход**
Синтаксис	`JMP label`
Операция	`PC+2 × смещение → PC`
Описание	10-разрядная величина смещения со знаком, содержащаяся в младших битах (LSB) команды прибавляется к счетчику команд.
Биты статуса	Биты статуса не изменяются
Рекомендация	Эта команда длиной в одно слово может заменить команду BRANCH для диапазона слов от -511 до +512 относительно текущего содержимого счетчика команд.

JN	**Переход, если отрицание**
Синтаксис	`JN label`
Операция	`Если N=1: PC+2 × смещение → PC`
	`Если N=0: выполняется следующая команда`
Описание	Проверяется бит отрицания (N) регистра статуса. Если он установлен, 10-разрядная величина смещения со знаком, содержащаяся в младших битах (LSB) команды прибавляется к счетчику команд. Если N сброшен, выполняется команда, следующая за инструкцией `jump`.

Биты статуса	Биты статуса не изменяются

Пример	Результат вычисления в R5 вычитается из COUNT. Если получается отрицательная величина, COUNT очищается и выполнение программы продолжается по другому пути.
	`SUB R5, COUNT ;COUNT – R5 ® COUNT`
	`JN L$1 ;Если результат отрицательный,`
	`;тогда COUNT=0, PC=L$1`
	`… ;Продолжение, если COUNT≥0`
	`…`
	`…`
	`…`
	`L$1 CLR COUNT`
	`…`
	`…`
	`…`

JNC	**Переход, если перенос не установлен**
JLO	**Переход, если низший**
Синтаксис	`JNC label JLO label`
Операция	`Если C=0: PC+2 × смещение → PC`
	`Если C=1: выполняется следующая команда`
Описание	Проверяется бит переноса (C) регистра статуса. Если он сброшен, 10-разрядная величина смещения со знаком, содержащаяся в младших битах (LSB) команды прибавляется к счетчику команд. Если бит C установлен, выполняется команда, следующая за инструкцией `jump`. Команда JNC (переход, если нет переноса / низший) используется для сравнения чисел без знака (от 0 до 65536).
Биты статуса	Биты статуса не изменяются

Пример	Результат в R6 прибавляется к BUFFER. Если происходит переполнение, выполняется процедура обработки ошибки по адресу ERROR.
	`ADD R6,BUFFER ;BUFFER + R6 → BUFFER`
	`JNC CONT ;Переход к CONT, если переноса нет`
	`ERROR … ;Начало процедуры обработки ошибки`
	`…`
	`…`
	`…`
	`COUNT … ;Продолжение нормального хода`
	`;программы`
	`…`
	`…`

Пример	Переход к STL2, если байт STATUS содержит 1 или 0.
	`CMP.B #2,STATUS`
	`JLO STL2 ;STATUS<2`
	`… ;STATUS≥2, продолжение здесь`

JNE	Переход, если не равно
JNZ	Переход, если не ноль
Синтаксис	JNE label JNZ label
Операция	Если Z=0: PC+2 × смещение → PC
	Если Z=1: выполняется следующая команда
Описание	Проверяется бит нуля (Z) регистра статуса. Если он сброшен, 10-разрядная величина смещения со знаком, содержащаяся в младших битах (LSB) команды прибавляется к счетчику команд. Если бит Z установлен, выполняется команда, следующая за инструкцией jump.
Биты статуса	Биты статуса не изменяются

Пример	Переход по адресу TONI, если регистры R7 и R8 имеют различное содержимое.
	`CMP R7, R8 ;Сравнение R7 с R8`
	`JNE TONI ;Переход, если содержимое различное`
	`… ;Продолжение, если содержимое одинаковое`

MOV[.W]	Пересылка содержимого источника в получатель
MOV.B	Пересылка содержимого источника в получатель
Синтаксис	MOV src, dst или MOV.W src, dst MOV.B src, dst
Операция	src → dst
Описание	Операнд источника посылается в получатель.Операнд источника не изменяется. Предыдущее содержимое получателя теряется.
Биты статуса	Биты статуса не изменяются
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются

Пример	Содержимое таблицы EDE (данные в виде слов) копируется в таблицу TOM. Длина таблиц должна составлять 020h ячеек.
	`MOV #EDE,R10 ;Подготовка счетчика`
	`MOV #020h,R9 ;Подготовка счетчика`
	`Loop MOV @R10+,TOM-EDE-2(R10) ;Использование`
	`;указателя`
	`;в регистре R10 для`
	`;обеих таблиц`
	`DEC R9 ;Декремент счетчика`
	`JNZ Loop ;Содержимое`
	`;счетчика≠0,`
	`;копирование`
	`;продолжается`
	`… ;Копирование закончено`
	`…`
	`…`

Пример	Содержимое таблицы EDE (данные в виде байт) копируется в таблицу TOM. Длина таблиц должна составлять 020h ячеек.
	`MOV #EDE,R10 ;Подготовка счетчика`
	`MOV #020h,R9 ;Подготовка счетчика`
	`Loop MOV.B @R10+,TOM-EDE-1(R10) ;Использование`
	`;указателя`

Пример	;в регистре R10 для ;обеих таблиц DEC R9 ;Декремент счетчика JNZ Loop ;Содержимое ;счетчика≠0, ;копирование ;продолжается … ;Копирование закончено … …

***NOP**	**Нет операции**
Синтаксис	NOP
Операция	не выполняется
Эмуляция	MOV #0, R3
Описание	Никакая операция не выполняется. Команда может использоваться для исключения команд в ходе проверки программного обеспечения или для задания необходимого времени ожидания.
Биты статуса	Биты статуса не изменяются

Команда NOP главным образом используется в двух случаях:

– сохранение одного, двух или трех слов памяти;

– корректировка программных временных интервалов.

Примечание: эмуляция команды NOP.

Другие команды могут эмулировать функцию NOP, позволяя получать различное количество циклов команды и слов кода. Ниже представлены некоторые примеры:

```
MOV     #0,R3         ;1 цикл, 1 слово
MOV     0(R4),0(R4)   ;6 циклов, 3 слова
MOV     @R4,0(R4)     ;5 циклов, 2 слова
BIC     #0,EDE(R4)    ;4 цикла, 2 слова
JMP     $+2           ;2 цикла, 1 слово
BIC     #0,R5         ;1 цикл, 1 слово
```

Однако, нужно соблюдать осторожность при использовании этих примеров, чтобы избежать непредсказуемых результатов. К примеру, при использовании команды MOV 0(R4), 0(R4), когда R4 содержит значение 120h, произойдет нарушение защиты сторожевого таймера (адрес 120h), потому что не будет использован ключ защиты.

***POP[.W]**	**Снятие со стека слова в получатель**
***POP.B**	**Снятие со стека слова в получатель**
Синтаксис	POP dst POP.B dst
Операция	@SP → temp SP + 2 → SP temp → dst

Эмуляция	MOV @SP+, dst или MOV.W @SP+, dst MOV.B @SP+, dst
Описание	Содержимое стека, на которое указывает указатель стека (TOS) помещается в получатель. Затем указатель стека инкрементируется на два.
Биты статуса	Биты статуса не изменяются

Пример	Восстановление из стека содержимого регистра R7 и регистра статуса.
	`POP R7 ;Восстановление R7`
	`POP SR ;Восстановление регистра статуса`

Пример	Восстановление из стека содержимого байта ОЗУ LEO. `POP.B LEO ;Младший байт помещается из стека в LEO`

Пример	Восстановление из стека содержимого регистра R7.
	`POP.B R7 ;Младший байт помещается из стека в R7,`
	`;старший байт регистра R7 равен 00h`

Пример	Восстановление из стека содержимого ячейки памяти, указанной в регистре R7 и содержимого регистра статуса.
	`POP.B 0(R7) ;Младший байт помещается из стека`
	`;в байт, который указан в регистре R7`
	`;Пример: R7=203h`
	`; Mem(R7) = младший байт`
	`; системного стека`
	`;Пример: R7=20Ah`
	`; Mem(R7) = младший байт`
	`; системного стека`
	`POP SR`

Примечание: указатель системного стека

Указатель системного стека (SP) всегда инкрементируется на два, независимо от наличия суффикса байта.

PUSH[.W]	Помещение слова в стек
PUSH.B	Помещение байта в стек
Синтаксис	PUSH src или PUSH.W src PUSH.B src
Операция	SP - 2 → SP src → @SP
Описание	Указатель стека декрементируется на два, затем операнд источника помещается в слово ОЗУ, адрес которого содержит указатель стека (TOS).
Биты статуса	Биты статуса не изменяются
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются

Пример	Содержимое регистра статуса и регистра R8 сохраняются в стеке.
	`PUSH SR ;сохранение регистра статуса`
	`PUSH R8 ;сохранение регистра R8`

Пример	Сохранение содержимого периферии TCDAT в стеке.
	`PUSH.B &TCDAT ;сохранение в стеке данных`
	`;из 8-разрядного периферийного`
	`;модуля, адресованного TCDAT`

Примечание: указатель системного стека

Указатель системного стека (SP) всегда декрементируется на два, независимо от наличия суффикса байта.

***RET**	**Возврат из подпрограммы**
Синтаксис	RET
Операция	@SP → PC SP + 2 → SP
Эмуляция	MOV @SP+, PC
Описание	Адрес возврата, помещенный в стек командой CALL посылается в счетчик команд. Программа продолжается с адреса кода, следующего за адресом команды вызова подпрограммы.
Биты статуса	Биты статуса не изменяются

RETI	**Возврат из прерывания**	
Синтаксис	RETI	
Операция	TOS → SR, SP + 2 → SP, TOS → PC, SP + 2 → SP	
Описание	Регистр состояния восстанавливает свое исходное значение, существовавшее до начала процедуры обработки прерывания, замещая текущее содержимое SR содержимым TOS. Указатель стека (SP) инкрементируется на два. Счетчик команд восстанавливает свое исходное значение, имевшееся до начала обработки прерывания. Это следующий шаг после прерывания нормального хода программы. Восстановление выполняется путем замещения текущего содержимого PC содержимым TOS. Указатель стека (SP) инкрементируется.	
Биты статуса	N:	восстанавливается из системного стека
	Z:	восстанавливается из системного стека
	C:	восстанавливается из системного стека
	V:	восстанавливается из системного стека
Биты режима	Биты OSCOFF, CPUOFF и GIE восстанавливаются из системного стека	

***RLA[.W]**	**Арифметическая ротация влево**
***RLA.B**	**Арифметическая ротация влево**
Синтаксис	RLA dst или RLA.W dst
	RLA.B dst
Операция	C ← MSB ← MSB–1 … LSB+1 ← LSB ← 0
Эмуляция	ADD dst, dst
	ADD.B dst, dst
Описание	Операнд получателя сдвигается влево на одну позицию, как показано на рис. 3.14. Старший бит MSB сдвигается в бит переноса (C), а в младший бит LSB записывается «0». Команда RLA действует как умножение со знаком на 2. Переполнение происходит, если dst ≥ 04000h и dst < 0C000h перед выполнением операции: результат меняет знак.

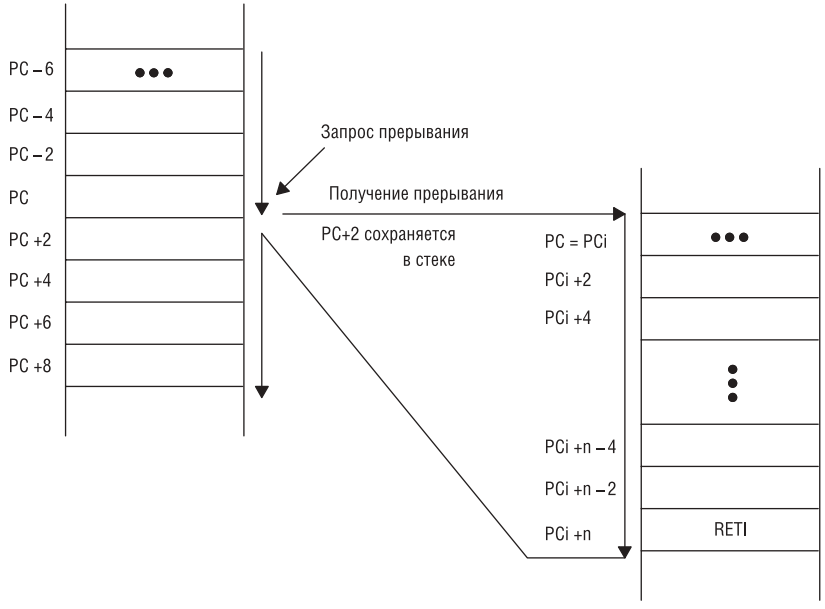


Рис. 3-13. Прерывание главной программы

Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается
	C:	Загружается из MSB
	V:	Устанавливается, если произошло арифметическое переполнение: исходное значение 04000h ≤ dst < 0C000h; в противном случае сбрасывается Устанавливается, если произошло арифметическое переполнение: исходное значение 040h ≤ dst <0C0h; в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

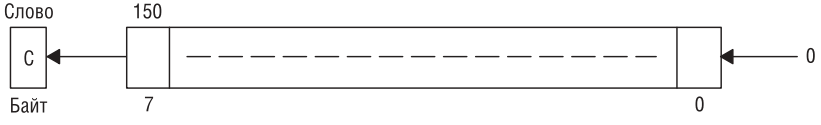


Рис. 3-14. Операнд получателя – арифметический сдвиг влево

Пример	Содержимое регистра R7 умножается на 2. `RLA R7 ;Сдвиг влево R7 (умножение на 2)`

Пример	Младший байт регистра R7 умножается на 4.
	`RLA.B R7 ;Сдвиг влево младшего байта R7`
	`;(умножение на 2)`
	`RLA.B R7 ;Сдвиг влево младшего байта R7`
	`;(умножение на 4)`

Примечание: замена RLA

Ассемблер не распознает команду:

RLA @R5+ и RLA.B @R5+

Вместо неё должна использоваться следующая команда:

ADD @R5+,-2(R5) и ADD.B @R5+,-1(R5)

***RLC[.W]**	**Арифметическая ротация влево через перенос**
***RLC.B**	**Арифметическая ротация влево через перенос**
Синтаксис	RLC dst или RLC.W dst
	RLC.B dst
Операция	C ← MSB ← MSB–1 … LSB+1 ← LSB ← C
Эмуляция	ADDC dst, dst
Описание	Операнд получателя сдвигается влево на одну позицию, как показано на рис. 3.15. Бит переноса (C) сдвигается в младший бит LSB, а старший бит MSB сдвигается в бит переноса (C).

Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается
	C:	Загружается из MSB
	V:	Устанавливается, если произошло арифметическое переполнение исходное значение 04000h ≤ dst < 0C000h; в противном случае сбрасывается Устанавливается, если произошло арифметическое переполнение исходное значение 040h ≤ dst < 0C0h; в противном случае сбрасывается

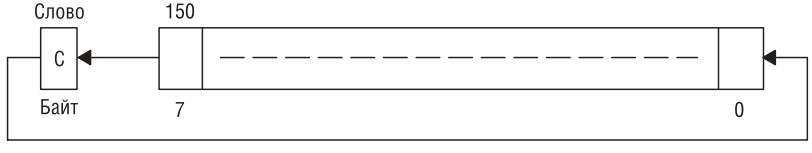


Рис. 3-15. Операнд получателя – сдвиг влево через перенос

Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются
Пример	Содержимое регистра R5 сдвигается влево на одну позицию. `RLC R5 ;(R5×2)+C → R5`

Пример	Данные с входа P1IN.1 сдвигаются в младший бит LSB регистра R5.
	`BIT.B #2,&P1IN ;данные → в бит переноса (Carry)`
	`RLC R5 ;Carry=P0in.1 → в LSB регистра R5`

Пример	Содержимое MEM (LEO) сдвигается влево на одну позицию. `RLC.B LEO ;Mem(LEO)×2+C → Mem(LEO)`

Примечание: эмуляция RLC и RLC.B

Ассемблер не распознает команду:

RLC @R5+

Вместо неё должна использоваться следующая команда:

ADDC @R5+,-2(R5)

RRA[.W]	**Арифметическая ротация вправо**
RRA.B	**Арифметическая ротация вправо**
Синтаксис	`RRA dst или RRA.W dst`
	`RRA.B dst`
Операция	`MSB → MSB, MSB → MSB-1, … LSB+1 → LSB, LSB → C`
Описание	Операнд получателя сдвигается вправо на одну позицию, как показано на рис. 3.16. Старший бит MSB сдвигается сам в себя и в бит MSB-1, бит LSB+1 сдвигается в младший бит LSB.

Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается
	C:	Загружается из LSB
	V:	Сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	
Пример	Содержимое регистра R5 сдвигается вправо на одну позицию. Старший бит MSB сохраняет старое значение. Эта операция эквивалентна арифметическому делению на 2. `RRA R5 ;R5/2 → R5`	

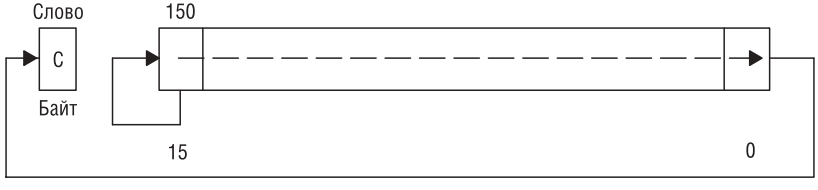


Рис. 3-16. Операнд получателя – арифметический сдвиг вправо

;
;
	Содержимое R5 умножается на 0.75 (0.5 + 0.25).
	`PUSH R5 ;Временное сохранение R5 с помощью стека`
	`RRA R5 ;R5×0.5 → R5`
	`ADD @SP+,R5 ;R5×0.5 + R5 = 1.5×R5 → R5`
	`RRA R5 ;(1.5×R5)×0.5 = 0.75×R5 → R5`
	…

Пример	Содержимое младшего байта регистра R5 сдвигается вправо на одну позицию. Старший бит MSB сохраняет старое значение. Эта операция эквивалентна арифметическому делению на 2.
	`RRA.B R5 ;R5/2 → R5: операция производится`
	`;только с младшим байтом, старший байт`
	`;R5 сброшен`
	`PUSH.B R5 ;R5×0.5 → TOS`
	`RRA.B @SP ;TOS×0.5 = 0.5×R5×0.5 = 0.25×R5 → TOS`
	`ADD.B @SP+,R5 ;R5×0.5 + R5×0.25 = 0.75×R5 → R5`
	…

RRC[.W]	**Ротация вправо через перенос**
RRC.B	**Ротация вправо через перенос**
Синтаксис	`RRC dst или RRC.W dst`
	`RRC.B dst`
Операция	`C → MSB → MSB-1 … LSB+1 → LSB → C`
Описание	Операнд получателя сдвигается вправо на одну позицию, как показано на рис. 3.17. Бит переноса (C) сдвигается в старший бит MSB, младший бит LSB сдвигается в бит переноса (C).

Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается
	C:	Загружается из LSB
	V:	Устанавливается, если исходное содержимое положительно и бит переноса перед выполнением операции установлен, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

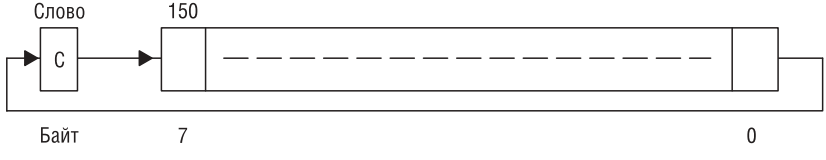


Рис. 3-17. Операнд получателя – сдвиг вправо через перенос

Пример	Содержимое регистра R5 сдвигается вправо на одну позицию. В старший бит MSB загружается «1».
	`SETC ;Подготовка бита переноса для MSB`
	`RRC R5 ;R5/2 + 8000h → R5`

Пример	Содержимое регистра R5 сдвигается вправо на одну позицию. В старший бит MSB загружается «1».
	`SETC ;Подготовка бита переноса для MSB`
	`RRC.B R5 ;R5/2 + 80h → R5 ;используется младший байт R5`

***SBC[.W]**	**Вычитание заема/.NOT. переноса из получателя**	
***SBC.B**	**Вычитание заема/.NOT. переноса из получателя**	
Синтаксис	`SBC dst или SBC.W dst`	
	`SBC.B dst`	
Операция	`dst + 0FFFFh + C → dst`	
	`dst + 0FFh + C → dst`	
Эмуляция	`SUBC #0, dst`	
	`SUBC.B #0, dst`	
Описание	Бит переноса (C) прибавляется к операнду получателя минус один. Предыдущее содержимое получателя теряется.	
Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается
	C:	Устанавливается, если есть перенос из старшего бита MSB результата, сбрасывается в противном случае. Устанавливается в «1», если заема нет; сбрасывается, если заем есть.
	V:	Устанавливается, если произошло арифметическое переполнение, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Содержимое 16-разрядного счетчика, указанного в R13, вычитается из 32-разрядного счетчика, указанного в R12:
	`SUB @R13,0(R12) ;Вычитание LSD`
	`SBC 2(R12) ;Вычитание переноса из MSD`	

Пример	Содержимое 8-разрядного счетчика, указанного в R13, вычитается из 16-разрядного счетчика, указанного в R12:
	`SUB.B @R13,0(R12) ;Вычитание LSD`
	`SBC.B 1(R12) ;Вычитание переноса из MSD`	

Примечание: реализация заема

Заем обрабатывается как .NOT. переноса:

Заем	*Да*	*Нет*
Бит переноса	*0*	*1*

***SETC**	**Установка бита переноса**	
Синтаксис	SETC	
Операция	1 → C	
Эмуляция	BIS #1, SR	
Описание	Устанавливается бит переноса (C).	
Биты статуса	N:	Не изменяется
	Z:	Не изменяется
	C:	Устанавливается
	V:	Не изменяется
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Эмуляция десятичного вычитания: Десятичное вычитание R5 из R6. Принимается, что R5=3987 и R6=4137:
	`DSUB ADD #6666h,R5 ;Пересылка содержимого R5`
	`;от 0-9 к 6-0Fh`
	`;R5=03987+6666=09FEDh`
	`INV R5 ;Инвертирование R5`
	`;(результат назад к 0-9)`
	`;R5=.NOT. R5=06012h`
	`SETC ;Подготовка переноса`
	`;carry=1`
	`DADD R5,R6 ;Эмулирование вычитания`
	`;сложением:`
	`;(1000-R5-1)`
	`;R6=R6+R5+1`
	`;R6=4137+06012+1=1 0150=0150`	

***SETN**	**Установка бита отрицания**	
Синтаксис	SETN	
Операция	1 → N	
Эмуляция	BIS #4, SR	
Описание	Устанавливается бит отрицания (N).	
Биты статуса	N:	Устанавливается
	Z:	Не изменяется
	C:	Не изменяется
	V:	Не изменяется
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

***SETZ**	**Установка бита нуля**
Синтаксис	SETZ
Операция	1 → Z
Эмуляция	BIS #2, SR
Описание	Устанавливается бит нуля (Z).

Биты статуса	N:	Не изменяется
	Z:	Устанавливается
	C:	Не изменяется
	V:	Не изменяется
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

SUB[.W]	**Вычитание источника из получателя**	
SUB.B	**Вычитание источника из получателя**	
Синтаксис	SUB src, dst или SUB.W src,dst	
	SUB.B src,dst	
Операция	dst + .NOT.src + 1 → dst или [(dst – src → dst)]	
Описание	Операнд источника вычитается из операнда получателя путем прибавления дополнения до единицы операнда источника и константы «1» к операнду получателя. Операнд источника не изменяется. Предыдущее содержимое получателя теряется.	
Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный.
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается.
	C:	Устанавливается, если есть перенос из старшего бита MSB результата, в противном случае сбрасывается. Устанавливается в «1», если нет заема; сбрасывается, если был заем.
	V:	Устанавливается, если произошло арифметическое переполнение, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	
Пример	См. пример для команды SBC	
Пример	См. пример для команды SBC.B	

Примечание: обработка заема как .NOT.

Заем обрабатывается как операция .NOT. переноса:

Заем	*Да*	*Нет*
Бит переноса	*0*	*1*

SUBC[.W], SBB[.W]	**Вычитание источника и заема/.NOT. переноса из получателя**
SUBC.B, SBB.B	**Вычитание источника и заема/.NOT. переноса из получателя**
Синтаксис	SUBC src, dst или SUBC.W src, dst или
	SBB src, dst или SBB.W src, dst
	SUBC.B src, dst или SBB.B src, dst
Операция	dst + .NOT.src + C → dst или [(dst – src – 1 + C → dst)]

Описание	Операнд источника вычитается из операнда получателя путем прибавления дополнения до единицы операнда источника и бита переноса (C) к операнду получателя. Операнд источника не изменяется. Предыдущее содержимое получателя теряется.	
Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный.
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается.
	C:	Устанавливается, если есть перенос из старшего бита MSB результата, в противном случае сбрасывается.Устанавливается в «1», если нет заема; сбрасывается, если был заем.
	V:	Устанавливается, если произошло арифметическое переполнение, в противном случае сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Вычитаются две мантиссы (24-разрядные) с плавающей точкой. Младшие байты LSB находятся в R13 и R10, старшие байты MSB находятся в R12 и R9.
	`SUB.W R13,R10 ;16-разрядная часть, LSB`
	`SUBC.B R12,R9 ;8-разрядная часть, MSB`	

Пример	Содержимое 16-разрядного счетчика, указанного в R13, вычитается из 16-разрядного счетчика, находящегося в регистрах R10 и R11(MSD).
	`SUB.B @R13+,R10 ;Вычитание младших байтов LSB без ;переноса`
	`SUBC.B @R13,R11 ;Вычитание старших байтов MSB ;с переносом,`
	`… ;возникшим в результате ;выполнения`
	`… ;операции над младшими байтами LSB`	

Примечание: реализация заема

Заем обрабатывается как операция .NOT. переноса:

```
Заем          Да    Нет
Бит переноса  0     1
```

SWPB	**Обмен байтами**
Синтаксис	`SWPB dst`
Операция	Биты с 15 по 8 « биты с 7 по 0
Описание	Старший и младший байты операнда получателя меняется местами, как показано на рис. 3.18
Биты статуса	Биты статуса не изменяются
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются
Пример	`MOV #040BFh,R7 ;0100000010111111 → R7` `SWPB R7 ;1011111101000000 в R7`

Пример	Содержимое R5 умножается на 255. Результат сохраняется в R5, R4.
	`SWPB R5 ;`
	`MOV R5, R4 ;копирование значения после обмена`
	`;в R4`
	`BIC #0FF00h, R5 ;корректировка результата`
	`BIC #00FFh, R4 ;корректировка результата`

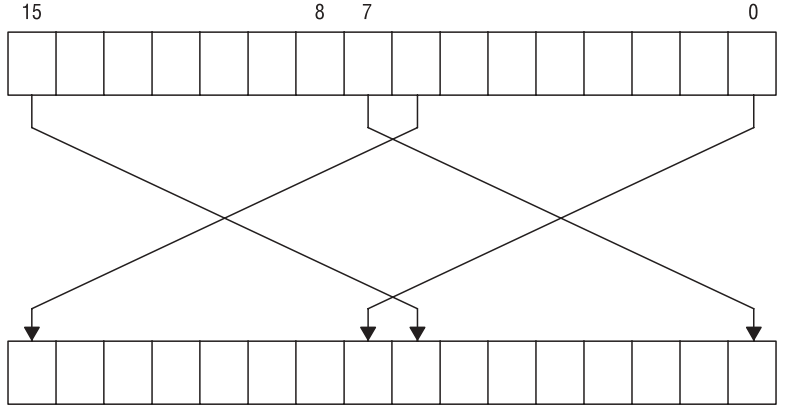


Рис. 3-18. Обмен байтов в операнде получателя

SXT	**Распространение знака**	
Синтаксис	SXT dst	
Операция	Бит 7 → в биты с 8 по 15	
Описание	Знак младшего байта распространяется в старшем байте, как показано на рис. 3.19.	
Биты статуса	Биты статуса не изменяются	
Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный.
	Z:	Устанавливается, если результат «0», в противном случае сбрасывается.
	C:	Устанавливается, если результат не ноль, в противном случае сбрасывается (.NOT. Zero)
	V:	Сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	В R7 загружается значение P1IN. Команда распространения знака выполняет операцию развертывания значения бита 7 в биты с 8 по 15.
	`MOV.B &P1IN,R7 ;P1IN = 080h: .... .... 1000 0000`
	`SXT R7 ;R7 = 0FF80h: 1111 1111 1000 0000`	

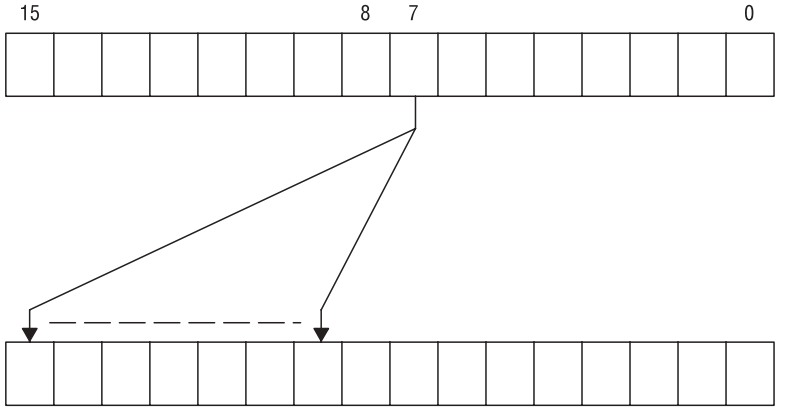


Рис. 3-19. Распространение знака операнда получателя

***TST[.W]**	**Проверка получателя**	
***TST.B**	**Проверка получателя**	
Синтаксис	`TST dst или TST.W dst`	
	`TST.B dst`	
Операция	`dst + 0FFFFh + 1`	
	`dst + 0FFh + 1`	
Эмуляция	`CMP  #0, dst`	
	`CMP.B #0, dst`	
Описание	Операнд получателя сравнивается с нулем. Биты статуса устанавливаются в соответствии с результатом сравнения. Получатель не изменяется.	
Биты статуса	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный.
	Z:	Устанавливается, если результат содержит «0», в противном случае сбрасывается.
	C:	Устанавливается
	V:	Сбрасывается
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	
Пример	Проверяется R7. Если его содержимое отрицательно, программа продолжается с метки R7NEG; если положительно, но не равно нулю, выполняется переход к метке R7POS.	

```
        TST  R7      ;проверка R7
        JN   R7NEG   ;содержимое R7 отрицательно
        JZ   R7ZERO  ;R7 содержит ноль
R7POS   …            ;содержимое R7 положительное, но не 0
R7NEG   …            ;содержимое R7 отрицательное
R7ZERO  …            ;R7 содержит ноль
```

Пример	Проверяется младший байт регистра R7. Если его содержимое отрицательно, программа продолжается с метки R7NEG; если положительно, но не равно нулю, выполняется переход к метке R7POS.
	`TST.B R7 ;проверка младшего байта R7`
	`JN R7NEG ;младший байт R7 отрицателен`
	`JZ R7ZERO ;младший байт R7 содержит ноль`
	`R7POS … ;младший байт R7 положителен,`
	`;но не 0`
	`R7NEG … ;младший байт R7 отрицателен`
	`R7ZERO … ;младший байт R7 содержит ноль`

XOR[.W]	**Исключающее «ИЛИ» источника и получателя**	
XOR.B	**Исключающее «ИЛИ» источника и получателя**	
Синтаксис	`XOR src, dst или XOR.W src, dst`	
	`XOR.B src, dst`	
Операция	`src .XOR. dst → dst`	
Описание	Над операндами источника и получателя выполняется операция логического «ИЛИ» (OR). Результат помещается в получатель. Операнд источника не изменяется.	
Биты статуса	N:	Устанавливается, если установлен MSB результата; сбрасывается, если не установлен.
	Z:	Устанавливается, если результат содержит «0», в противном случае сбрасывается.
	C:	Устанавливается, если результат не ноль, в противном случае сбрасывается (= .NOT. Zero)
	V:	Устанавливается, если оба операнда отрицательны
Биты режима	Биты OSCOFF, CPUOFF и GIE не изменяются	

Пример	Биты, установленные в регистре R6 переключают биты в слове ОЗУ TONI.
	`XOR R6,TONI ;Переключение битов слова TONI`
	`;в соответствии с битами,`
	`;установленными в R6`	

Пример	Биты, установленные в регистре R6 переключают биты в байте ОЗУ TONI.
	`XOR.B R6,TONI ;Переключение битов байта TONI`
	`;в соответствии с битами,`
	`;установленными в младшем байте`
	`;регистра R6`	

Пример	Обнуление битов в младшем байте регистра R7, которые отличаются от соответствующих битов байта ОЗУ EDE.
	`XOR.B EDE,R7 ;Установка отличающихся битов в «1»`
	`INV.B R7 ;Инвертирование младшего байта R7,`
	`;в старшем байте «0h»`	

3.4.4. Командные циклы и длина команд

Число тактовых циклов ЦПУ, требуемых для выполнения команды, определяется форматом команды и используемым режимом адресации, и не зависит, собственно, от команды. Понятие количества тактовых циклов относится к MCLK.

Циклы прерывания и сброса

В таблице 3.14 приведено количество циклов ЦПУ для обслуживания прерывания и сброса.

Таблица 3.14. Циклы прерывания и сброса

Действие	Количество циклов	Длина команды
Возврат из прерывания (RETI)	5	1
Получение прерывания	6	–
Сброс WDT	4	–
Сброс (nonRST/NMI)	4	–

Циклы команд формата-II (один операнд) и их длина

В таблице 3.15 приводится длина и необходимое количество циклов ЦПУ для всех адресных режимов команд формата-II.

Таблица 3.15. Количество циклов и длина команд формата-II

Режим адресации	Действие			Длина команды	Пример
	RRA, RRC, SWPB, SXT	PUSH	CALL		
Rn	1	3	4	1	SWPB R5
@Rn	3	4	4	1	RRC @R9
@Rn+	3	4	5	1	SWPB @R10+
#N	См. прим.	4	5	2	CALL #0F000h
X(Rn)	4	5	5	2	CALL 2(R7)
EDE	4	5	5	2	PUSH EDE
&EDE	4	5	5	2	SXT &EDE

Примечание: команда формата-II в непосредственном режиме адресации

Не следует использовать команды RRA, RRC и SXT с непосредственным режимом в поле получателя. Их использование в непосредственном режиме приведет к выполнению непредсказуемой программной операции.

Циклы команд формата-III (команды перехода) и их длина

Все команды перехода требуют одно слово кода и при выполнении используют два цикла ЦПУ, независимо от того, сделан переход или нет.

Циклы команд формата-I (двойной операнд) и их длина

В таблице 3.16 приводится длина и необходимое количество циклов ЦПУ для всех адресных режимов команд формата-I.

Таблица 3.16. Количество циклов и длина команд формата-I

Режим адресации		Количество циклов	Длина команды	Пример
Src	**Dst**			
Rn	Rm	1	1	`MOV R5,R8`
	PC	2	1	`BR R9`
	x(Rm)	4	2	`ADD R5,4(R6)`
	EDE	4	2	`XOR R8,EDE`
	&EDE	4	2	`MOV R5,&EDE`
@Rn	Rm	2	1	`AND @R4,R5`
	PC	3	1	`BR @R8`
	x(Rm)	5	2	`XOR @R5,8(R6)`
	EDE	5	2	`MOV @R5,EDE`
	&EDE	5	2	`XOR @R5,&EDE`
@Rn+	Rm	2	1	`ADD @R5+,R6`
	PC	3	1	`BR @R9+`
	x(Rm)	5	2	`XOR @R5,8(R6)`
	EDE	5	2	`MOV @R9+,EDE`
	&EDE	5	2	`XOR @R9+,&EDE`
#N	Rm	2	2	`MOV #20,R9`
	PC	3	2	`BR #2AEh`
	x(Rm)	5	3	`MOV #0300h,0(SP)`
	EDE	5	3	`ADD #33,EDE`
	&EDE	5	3	`ADD #33,&EDE`
x(Rn)	Rm	3	2	`MOV 2(R5),R7`
	PC	3	2	`BR 2(R6)`
	TONI	6	3	`MOV 4(R7),TONI`
	x(Rm)	6	3	`ADD 4(R4),6(R9)`
	&TONI	6	3	`MOV 2(R4),&TONI`

Таблица 3.16. (Окончание)

Режим адресации		Количество циклов	Длина команды	Пример
Src	**Dst**			
EDE	Rm	3	2	`AND EDE,R6`
	PC	3	2	`BR EDE`
	TONI	6	3	`CMP EDE,TONI`
	x(Rm)	6	3	`MOV EDE,0(SP)`
	&TONI	6	3	`MOV EDE,&TONI`
&EDE	Rm	3	2	`MOV &EDE,R8`
	PC	3	2	`BRA &EDE`
	TONI	6	3	`MOV &EDE,TONI`
	x(Rm)	6	3	`MOV &EDE,0(SP)`
	&TONI	6	3	`MOV &EDE,&TONI`

3.4.5. Описание набора команд

Карта команд показана на рис. 3.20, а полный набор команд приведен в таблице 3.17.

	000	040	080	0C0	100	140	180	1C0	200	240	280	2C0	300	340	380	3C0
0xxx																
4xxx																
8xxx																
Cxxx																
1xxx	RRC	RRC.B	SWPB		RRA	RRA.B	SXT		PUSH	PUSH.B	CALL		RETI			
14xx																
18xx																
1Cxx																
20xx	JNE/JNZ															
24xx	JEQ/JZ															
28xx	JNC															
2Cxx	JC															
30xx	JN															
34xx	JGE															
38xx	JL															
3Cxx	JMP															
4xxx	MOV, MOV.B															
5xxx	ADD, ADD.B															
6xxx	ADDC, ADDC.B															
7xxx	SUBC, SUBC.B															
8xxx	SUB, SUB.B															
9xxx	CMP, CMP.B															
Axxx	DADD, DADD.B															
Bxxx	BIT, BIT.B															
Cxxx	BIC, BIC.B															
Dxxx	BIS, BIS.B															
Exxx	XOR, XOR.B															
Fxxx	AND, AND.B															

Рис. 3-20. Карта команд ядра

Таблица 3.17. Набор команд MSP430

Мнемоника		Описание		V	N	Z	C
ADC(.B)*	dst	Сложение бита C с получателем	dst + C → dst	*	*	*	*
ADD(.B)	src,dst	Сложение источника с получателем	src + dst → dst	*	*	*	*
ADDC(.B)	src,dst	Сложение источника и бита C с получателем	src + dst + C → dst	*	*	*	*
AND(.B)	src,dst	Операция «И» источника и получателя	src .and. dst → dst	0	*	*	*
BIC(.B)	src,dst	Очистка битов в получателе	.not.src .and. dst → dst	-	-	-	-
BIS(.B)	src,dst	Установка битов в получателе	src .or. dst → dst	-	-	-	-
BIT(.B)	src,dst	Проверка битов в получателе	src .and. dst	0	*	*	*
BR*	dst	Переход по назначению	dst → PC	-	-	-	-
CALL	dst	Вызов получателя	PC + 2 → stack, dst → PC	-	-	-	-
CLR(.B)*	dst	Очистка получателя	0 → dst	-	-	-	-
CLRC*		Очистка бита C	0 → C	-	-	-	0
CLRN*		Очистка бита N	0 → N	-	0	-	-
CLRZ*		Очистка бита Z	0 → Z	-	-	0	-
CMP(.B)	src,dst	Сравнение источника и получателя	dst - src	*	*	*	*
DADC(.B)*	dst	Десятичное сложение бита C с получателем	dst + c → dst (десятичное)	*	*	*	*
DADD(.B)	src,dst	Десятичное сложение источника и бита C с получателем	src + dst + C → dst (десятичное)	*	*	*	*
DEC(.B)*	dst	Декремент получателя	dst - 1 → dst	*	*	*	*

Таблица 3.17. (Продолжение)

Мнемоника		Описание		V	N	Z	C
DECD(.B)*	dst	Двойной декремент получателя	dst - 2 → dst	*	*	*	*
DINT*		Запрещение прерываний	0 → GIE	-	-	-	-
EINT*		Разрешение прерываний	1 → GIE	-	-	-	-
INC(.B)*	dst	Инкремент получателя	dst + 1 → dst	*	*	*	*
INCD(.B)*	dst	Двойной инкремент получателя	dst + 2 → dst	*	*	*	*
INV(.B)*	dst	Инвертирование получателя	.not.dst → dst	*	*	*	*
JC/JHS	label	Переход, если C установлен / переход если наивысший или такой же		-	-	-	-
JEQ/JZ	label	Переход, если равно / переход если Z установлен		-	-	-	-
JGE	label	Переход, если больше или равно		-	-	-	-
JL	label	Переход, если меньше		-	-	-	-
JMP	label	Переход	PC + 2 × смещение → PC	-	-	-	-
JN	label	Переход, если N установлен		-	-	-	-
JNC/JLO	label	Переход, если C не установлен / переход если низший		-	-	-	-
JNE/JNZ	label	Переход, если не равно, переход если Z не установлен		-	-	-	-
MOV(.B)	src,dst	Пересылка источника в получатель	src → dst	-	-	-	-
NOP*		Нет операции		-	-	-	-

Таблица 3.17. (Окончание)

Мнемоника		Описание		V	N	Z	C
POP(.B)*	dst	Снятие элемента со стека в получатель	@SP → dst, SP + 2 → SP	-	-	-	-
PUSH(.B)	src	Помещение источника в стек	SP - 2 → SP, src → @SP	-	-	-	-
RET*		Возврат из подпрограммы	@SP → PC, SP + 2 → SP	-	-	-	-
RETI		Возврат из прерывания		*	*	*	*
RLA(.B)*	dst	Арифметическая ротация влево		*	*	*	*
RLC(.B)*	dst	Ротация влево через C		*	*	*	*
RRA(.B)	dst	Арифметическая ротация вправо		0	*	*	*
RRC(.B)	dst	Ротация вправо через C		*	*	*	*
SBC(.B)*	dst	Вычитание not(C) из получателя	dst + 0FFFFh + C → dst	*	*	*	*
SETC*		Установка C	1 → C	-	-	-	1
SETN*		Установка N	1 → N	-	1	-	-
SETZ*		Установка Z	1 → C	-	-	1	-
SUB(.B)	src,dst	Вычитание источника из получателя	dst + .not.src + 1 → dst	*	*	*	*
SUBC(.B)	src,dst	Вычитание источника и not(C) из получателя	dst + .not.src + C → dst	*	*	*	*
SWPB	dst	Обмен байтов		-	-	-	-
SXT	dst	Распространение знака		0	*	*	*
TST(.B)*	dst	Проверка получателя	dst + 0FFFFh + 1	0	*	*	1
XOR(.B)	src,dst	Исключающее «ИЛИ» источника и получателя	src .xor. dst → dst	*	*	*	*

***Эмулированные команды**

MSP430x1xxFamily

Основной модуль тактирования

Раздел IV.

Texas Instruments

Основной модуль тактирования

Основной модуль тактирования обеспечивает тактирование устройств семейства MSP430x1xx. В этом разделе описывается работа с основным модулем тактирования. Этот модуль реализован во всех устройствах семейства MSP430x1xx.

4.1. Введение в основой модуль тактирования

Основной модуль тактирования имеет низкую стоимость и ультранизкое энергопотребление. Используя три внутренних тактовых сигнала, пользователь может выбрать наилучший баланс соотношения производительности и малой потребляемой мощности. Основной модуль тактирования может быть программно сконфигурирован на работу без использования внешних компонент, с одним внешним резистором, с одним или двумя внешними кристаллами или резонаторами.

Основной модуль тактирования включает в себя два или три источника тактовых импульсов:

- LFXT1CLK: низкочастотный / высокочастотный осциллятор, который может использоваться как с низкочастотными часовыми кристаллами на 32768 Гц или стандартными кристаллами или резонаторами в диапазоне от 450 кГц до 8 МГц.
- XT2CLK: дополнительный высокочастотный осциллятор может использоваться со стандартными кристаллами, резонаторами или внешними источниками тактовых сигналов в диапазоне от 450 кГц до 8 МГц.
- DCOCLK: встроенный осциллятор с цифровым управлением (DCO) с характеристикой RC-типа.

От основного модуля тактирования можно получить три тактовых сигнала:

- ACLK: вспомогательное тактирование. Модуль ACLK – это буферизированный LFXT1CLK источник тактовых импульсов с делителем на 1, 2, 4 или 8. ACLK программно выбирается для конкретных периферийных модулей.
- MCLK: основное тактирование. Модуль MCLK программно выбирается как LFXT1CLK, XT2CLK (если доступен) или DCOCLK. MCLK делится на 1, 2, 4 или 8. MCLK используется ЦПУ и системой.
- SMCLK: второстепенное тактирование. Модуль SMCLK программно выбирается как LFXT1CLK, XT2CLK (если доступен) или DCOCLK. SMCLK делится на 1, 2, 4 или 8. SMCLK программно выбирается для конкретных периферийных модулей.

Блок-схема основного модуля тактирования показана на рис. 4.1.

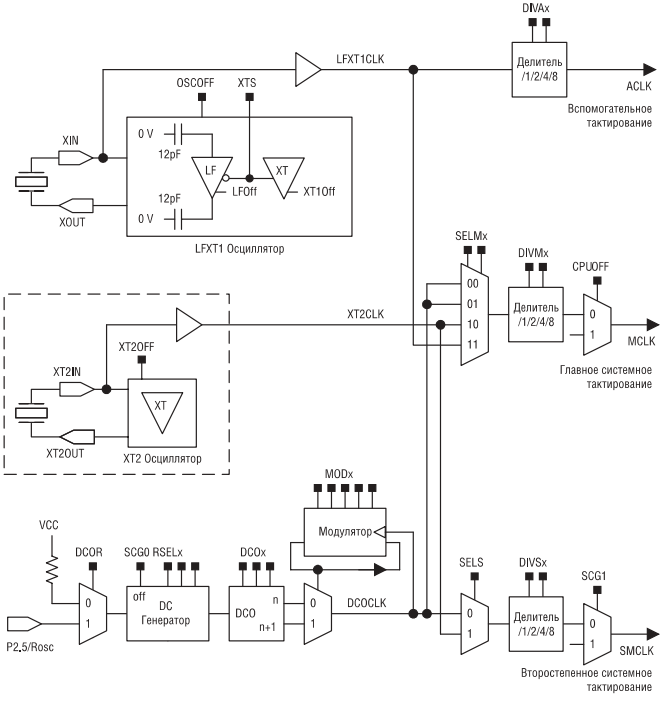


Рис. 4-1. Блок-схема основного тактирования

Примечание: XT2 осциллятор

Осциллятор XT2 отсутствует в устройствах MSP430x11xx и MSP430x12xx. Вместо XT2CLK используется LFXT1CLK.

4.2. Функционирование основного модуля тактирования

После сигнала PUC источником для модулей MCLK и SMCLK является DCOCLK с частотой около 800 кГц (см. параметры в справочном руководстве конкретного устройства) и LFXT1 для модуля ACLK в режиме LF.

Управляющие биты регистра статуса SCG0, SCG1, OSCOFF и CPUOFF конфигурируют рабочие режимы MSP430 и позволяют включать или отключать отдельные части основного модуля тактирования. См. раздел «Сброс, прерывания и рабочие режимы». С помощью регистров DCOCTL, BCSCTL1 и BCSCTL2 осуществляется конфигурирование основного модуля тактирования.

Основное тактирование может конфигурироваться и реконфигурироваться программным обеспечением в любой момент времени в ходе выполнения программы, например:

```
BIS.B #RSEL2+RSEL1+RSEL0,&BCSCTL1
BIS.B #DCO2+DCO1+DCO0,&DCOCTL ;установка
                              ;максимальной
                              ;частоты DCO
```

4.2.1. Возможности основного модуля тактирования в приложениях с малым потреблением мощности

В приложениях на основе MSP430x1xx с питанием от батарей обычно существуют следующие противоречивые требования:

- Низкая тактовая частота для экономии энергии и увеличения времени работы от батарей;
- Высокая тактовая частота для быстрой реакции на события и обеспечения возможности быстрой обработки информации

Основной модуль тактирования позволяет пользователю обходить вышеперечисленные противоречия путем выбора наиболее оптимального из трех возможных сигналов тактирования: ACLK, MCLK и SMCLK. Для оптимальной производительности с низким энергопотреблением модуль ACLK может быть сконфигурирован на работу от часового кристалла на 32768 Гц, обеспечивающим стабильное тактирование для системы и малое потребление в режиме ожидания. MCLK может настраиваться на работу от интегрированного модуля DCO, который активируется только при появлении запроса на обработку прерывания. SMCLK можно конфигурировать на работу как от часового кристалла, так и от DCO, в зависимости от требований периферии. Гибкое распределение тактовых сигналов и наличие система деления тактовой частоты обеспечивает тонкую настройку индивидуальных потребностей тактирования.

4.2.2. Осциллятор LFXT1

В режиме LF (XTS=0) осциллятор LFXT1 обеспечивает ультранизкое потребление тока при использовании часового кристалла на 32768 Гц. Часовой кристалл подключается к выводам XIN и XOUT без каких-либо дополнительных компонентов. При работе осциллятора LFXT1 в режиме LF используются внутренние нагрузочные конденсаторы на 12 пФ. Они включаются последовательно, обеспечивая нагрузку 6 пФ, необходимую для стандартного кристалла 32768 Гц. При необходимости могут быть добавлены дополнительные конденсаторы.

Осциллятор LFXT1 также поддерживает высокочастотные кристаллы или резонаторы, когда находится в режиме HF (XTS=1). Высокоскоростные кристаллы или резонаторы подключаются к выводам XIN и XOUT и нуждаются в использовании внешних конденсаторов на обоих выводах. Параметры конденсаторов должны соответствовать требованиям, приведенным в спецификациях кристаллов или резонаторов.

LFXT1 может использоваться с внешним источником тактового сигнала, который подаётся на вывод XIN и в LF и в HF режимах. В этом случае, частота внешнего тактового сигнала должна соответствовать параметрам, указанным для выбранного режима.

Программное обеспечение может отключить осциллятор LFXT1 установкой OSCOFF, если этот сигнал не используется в качестве источника для SMCLK или MCLK, как показано на рис. 4.2.

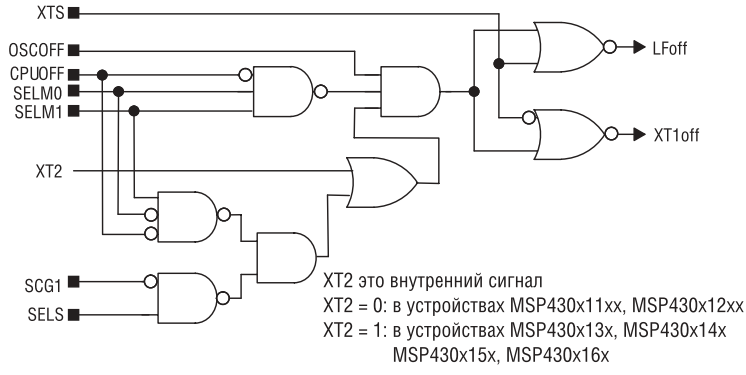


Рис. 4-2. Сигналы выключения осциллятора LFXT1

Примечание: характеристики осциллятора LFXT1

Низкочастотным кристаллам, в зависимости от типа, часто требуются сотни миллисекунд для старта. Это допускается для осциллятора LFXT1 в LF режиме.

Осцилляторы с ультранизким потреблением, такие как LFXT1 в режиме LF необходимо защищать от шумов других источников. Кристалл следует размещать как можно ближе к MSP430 с заземленной площадкой под ним, а трассировку проводников от кристалла выполнять с защитными заземляющими проводниками.

При работе осциллятора LFXT1 в режиме LF требуется подключение резистора 5,1 МОм между выводами Xout и Vss, когда Vcc < 2,5 В.

4.2.3. Осциллятор XT2

Некоторые устройства имеют второй кристаллический осциллятор XT2. XT2 является источником сигнала XT2CLK, а его характеристики идентичны LFXT1 в режиме HF. Бит XT2OFF отключает осциллятор XT2, если XT2CLK не используется для MCLK или SMCLK, как показано на рис. 4.3.

XT может быть использован с внешним источником тактирования, который подключается к выводу XT2IN. Частота внешнего тактового сигнала выбирается в соответствии с параметрами XT2.

4.2.4. Осциллятор с цифровым управлением (DCO)

DCO представляет собой интегрированный автогенератор с характеристикой RC-типа. Как у любого осциллятора RC-типа, его частота зависит от температуры, напряжения и отличается от устройства к устройству. Частота DCO может подстраиваться программным обеспечением с помощью битов DCOx, MODx и

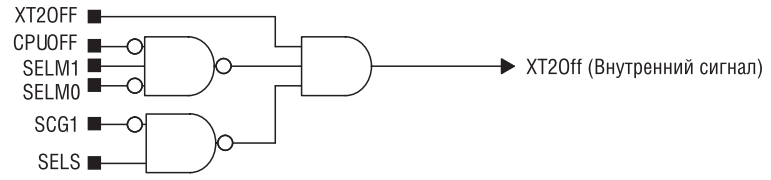


Рис. 4-3. Сигналы выключения осциллятора XT2

RSELx. Цифровое управление осциллятором позволяет стабилизировать частоту, несмотря на характеристику RC-типа.

Отключение DCO

Программное обеспечение может отключать DCOCLK, когда он не используется в качестве источника для SMCLK или MCLK, как показано на рис. 4.4.

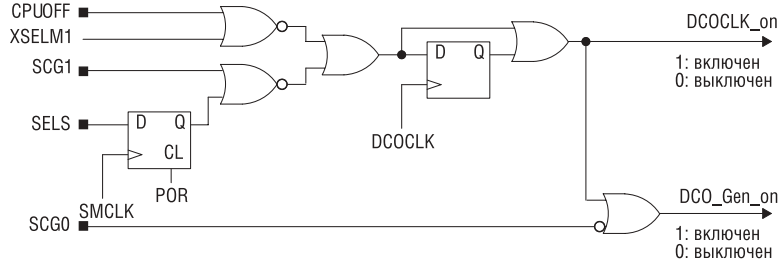


Рис. 4-4. Включение/выключение DCO

Подстройка частоты DCO

После сигнала PUC для DCO генератора выбирается встроенный резистор, устанавливаются значения RSELx=4 и DCOx=3, в результате DCO стартует с усредненной частоты. В качестве источника для MCLK и SMCLK используется DCOCLK. Поскольку при выполнении кода ЦПУ тактируется от сигнала MCLK, который использует быстростартующий DCO, выполнение кода начинается менее чем через 6 мкС после сигнала PUC. На рисунке 4-5 приведена зависимость частоты DCO от значений DCOx и RSELx

Частота DCOCLK устанавливается следующими способами:

- Фундаментальная частота определяется инжекцией тока в DC генератор через внутренний либо внешний резистор. Бит DCOR позволяет выбрать внутренний или внешний резистор.
- Три бита RSELx позволяют выбрать для DCO один из восьми номинальных диапазонов частот. Эти диапазоны определены для конкретного устройства в соответствующем ему справочном руководстве.
- Три бита DCOx делят диапазон DCO, выбранный с помощью битов RSELx на 8 уровней частоты, различающихся примерно на 10%.

- Пять битов MODx выполняют переключение между частотой, устанавливаемой битами DCOx и следующей более высокой частотой, устанавливаемой DCOx+1. В случае установки DCOx = 07h значение MODx не будет влиять на частоту DCO, так как для DCOx уже установлено максимальное значение.Диапазоны DCOx и RSELx, а также возможные шаги изменения частоты показаны на рис. 4.5.

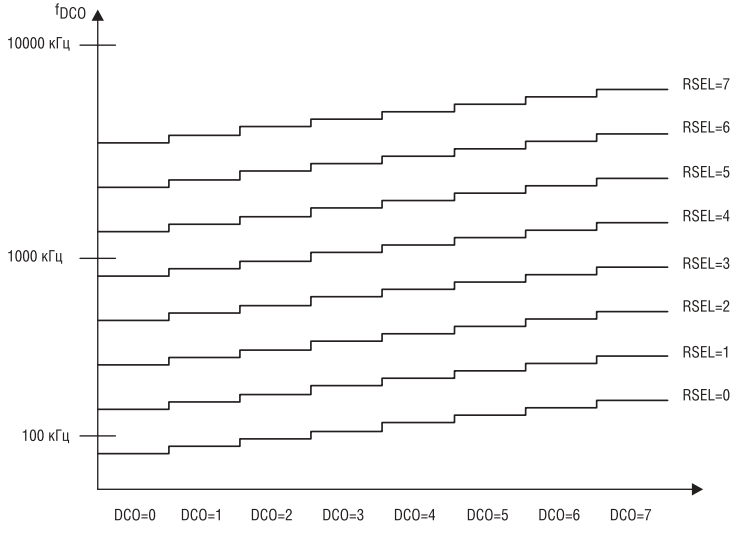


Рис. 4-5. Диапазон DCOx и шаги RSELx

Использование внешнего резистора (Rosc) для DCO

Температурный коэффициент DCO может быть уменьшен при использовании внешнего резистора Rosc в качестве источника тока для DC генератора. На рис. 4.6 показана типичная зависимость частоты DCO от температуры для встроенного и внешнего резисторов. Использование внешнего резистора Rosc уменьшает температурный коэффициент DCO примерно на -0.05%/C. Подробные характеристики приведены в справочном руководстве на конкретное устройство.

Резистор Rosc также позволяет работать DCO на высоких частотах. К примеру, встроенный резистор с номинальным сопротивлением около 300 кОм позволяет работать модулю DCO на частоте приблизительно до 5 МГц. Когда используется внешний резистор Rosc сопротивлением около 100 кОм, DCO может работать на частотах до 10 МГц. Пользователю необходимо соблюдать осторожность, чтобы не превысить максимальную частоту MCLK, указанную в справочных данных, даже если DCO способен работать на более высоких частотах.

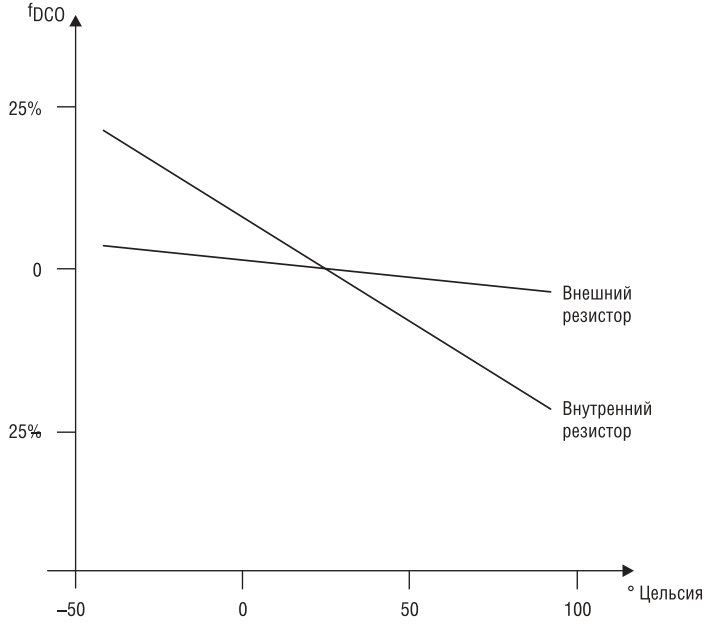


Рис. 4-6. Зависимость частоты модуля DCO от температуры

4.2.5. DCO модулятор

Модулятор смешивает две DCO-частоты: fDCO и fDCO+1, вырабатывая промежуточную эффективную частоту между fDCO и fDCO+1 и распределяет энергию тактирования, что приводит к снижению электромагнитных помех (EMI[1]). Модулятор смешивает частоты fDCO и fDCO+1 для 32 тактовых циклов DCOCLK и может конфигурироваться с помощью битов MODx. Когда MODx=0, модулятор выключен.

Смешивание частот модулятором происходит согласно следующей формуле:

$$t=(32\text{-}MODx) \times tDCO + MODx \times tDCO+1$$

Поскольку fDCO меньше эффективной частоты, а fDCO+1 выше, погрешность эффективной частоты в сумме равна нулю. Накопления погрешности не происходит. Погрешность эффективной частоты равна нулю каждые 32 цикла DCOCLK. На рис. 4.7 показана работа модулятора.

Параметры настройки модулятора и управления DCO конфигурируются программно. Сигнал DCOCLK может сравниваться со стабильной, заранее известной частотой и подстраиваться с помощью битов DCOx, RSELx и MODx. Заме-

[1] EMI – ElectroMagnetic Interference

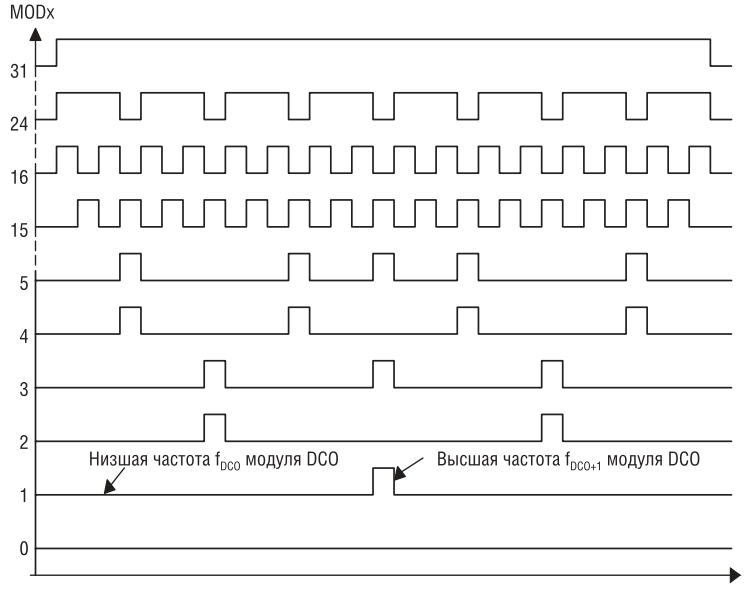


Рис. 4-7. Диаграмма сигналов модулятора

чания по применению и примеры кода для конфигурирования DCO можно найти на сайте **http://www.ti.com/sc/msp430**.

4.2.6. Надежность работы основного модуля тактирования

В основном модуле тактирования имеется возможность определения возникновения неисправности осциллятора. Детектор неисправности осциллятора представляет собой аналоговую схему мониторинга сигналов LFXT1CLK (в режиме HF) и XT2CLK. Неисправность осциллятора определяется, когда любой из этих тактовых сигналов отсутствует в течение приблизительно 50 мкС. Когда обнаруживается неисправность осциллятора, а источником для сигнала MCLK выступает либо LFXT1 в режиме HF, либо XT2, происходит автоматическое переключение MCLK на работу от DCO, как от источника тактовых импульсов. Это позволяет продолжить выполнение программного кода, даже в ситуации, когда кварцевый генератор остановился.

Если установлены флаги OFIFG и OFIE, происходит запрос немаскируемого прерывания NMI. Процедура обработки NMI-прерывания может проверить флаг OFIFG, что позволит выявить возникшую неисправность осциллятора. Очистка флага OFIFG должна производиться программным обеспечением.

Примечание: определение неисправности осциллятора LFXT1 в режиме LF не производится.

Определение неисправности осциллятора выполняется только для LFXT1 в режиме HF и для XT2. Детектирование неисправности осциллятора модуля LFXT1 в режиме LF не производится.

Флаг OFIFG устанавливается сигналом неисправности осциллятора XT_OscFault. Сигнал XT_OscFault устанавливает POR, когда модули XT2 или LFXT1 в режиме HF имеют неисправность осциллятора. Когда XT2 или LFXT1 в режиме HF останавливаются программным обеспечением, сигнал XT_OscFault вырабатывается немедленно, и остается активным пока осциллятор не будет перезапущен, и снимается примерно через 50 мкС после рестарта осциллятора, как показано на рис. 4.8.

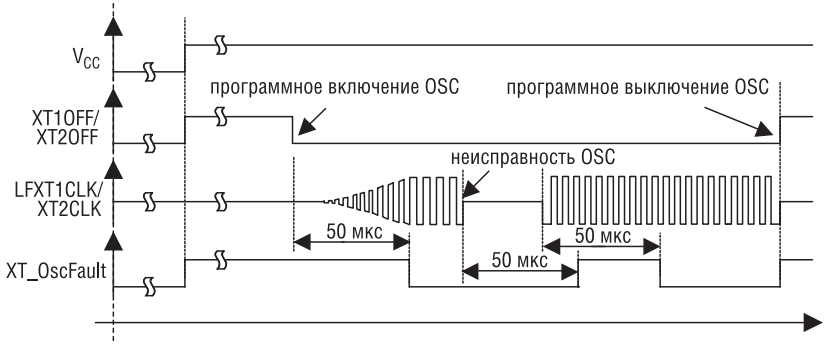


Рис. 4-8. Сигнал неисправности осциллятора

Определение неисправности осциллятора

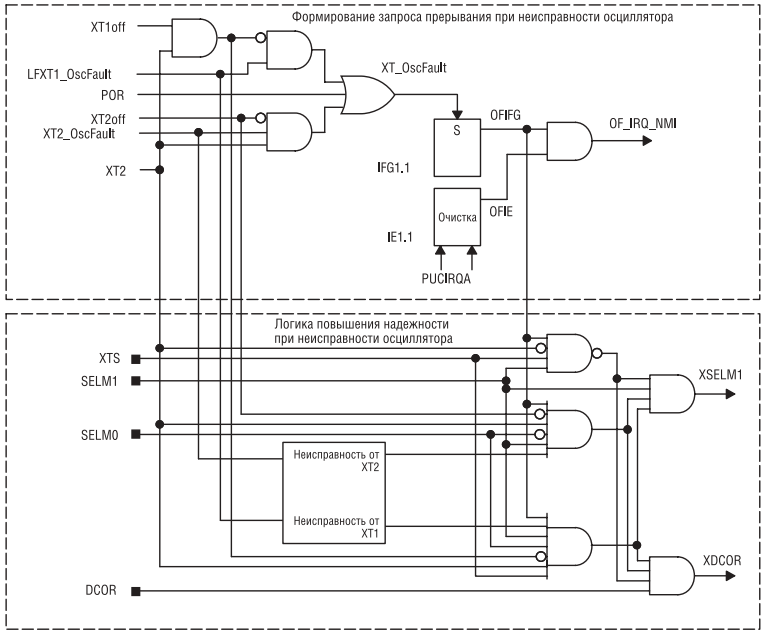
Сигнал XT_OscFault переключает флаг OFIFG так, как показано на рис. 4.9. Сигнал LFXT1_OscFault имеет низкий уровень, когда LFXT1 находится в LF режиме.

В устройствах, у которых модуль XT2 отсутствует, флаг OFIFG не может быть очищен, когда LFXT1 в режиме LF. Источником для сигнала MCLK может являться LFXT1CLK в режиме LF при установке битов SELMx, даже если флаг OFIFG остается поднятым.

В устройствах, имеющих XT2, флаг OFIFG может очищаться программно, когда LFXT1 находится в режиме LF и далее остается очищенным. Источником для сигнала MCLK может являться LFXT1CKL в режиме LF независимо от состояния флага OFIFG.

Использование кварцевого резонатора для формирования MCLK

После сигнала PUC основной модуль тактирования использует DCOCLK для формирования MCLK. Если необходимо, в качестве источника сигнала для MCLK можно использовать LFXT1 или XT2. Для смены источника тактирования сигнала MCLK с модуля DCO на тактирование от кварцевого резонатора (LFXT1CLK или XT2CLK) используется следующая последовательность команд:



XT2 - внутренний сигнал, XT2=0 в устройствах без модуля XT2 (MSP430x11xx и MSP430x12xx).
XT2 = 1 в устройствах, имеющих XT2 (MSP430F13x, MSP430F14x, MSP430F15x и MSP430F16x).
IRQA: прием запроса прерывания
LFXT1_OscFault: применим только к осциллятору LFXT1 в режиме HF.

Рис. 4-9 Прерывание при неисправности осциллятора

1) Переключение на кварцевый резонатор
2) Очистка флага OFIFG
3) Ожидание в течение примерно 50 мкС
4) Проверка OFIFG и повторение шагов с 1-4 до тех пор, пока OFIFG остается очищенным.

```
;Выбор LFXT1 (в режиме HF) для MCLK
   BIC   #OSCOFF,SR              ;включение осциллятора
   BIS.B #XTS,BCSCTL1            ;установка режима HF
L1 BIC.B #OFIFG,&IFG1            ;очистка OFIFG
   MOV   #0FFh,R15               ;задержка
L2 DEC   R15                     ;
   JNZ   L2                      ;
   BIT.B #OFIFG,&IFG1            ;повторная проверка
                                 ;OFIFG
   JNZ   L1                      ;повторение проверки,
                                 ;если необходимо
   BIS.B #SELM1+SELM0,&BCSCTL2   ;выбор LFXT1CLK
```

4.2.7. Синхронизация сигналов тактирования

Когда происходит переключение MCLK или SMCLK на другой источник опорной тактовой частоты, переключатель синхронизируется, чтобы избежать критических состояний «гонки» сигналов. Это показано на рис. 4.10:

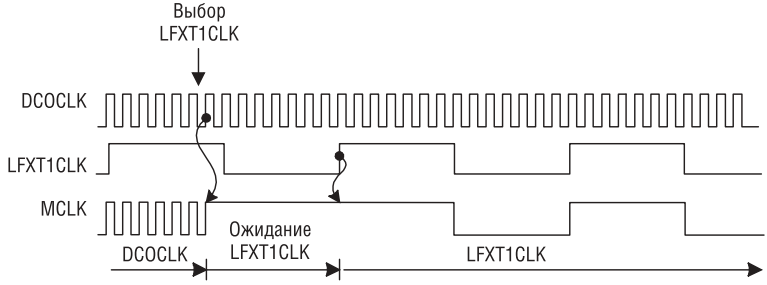


Рис. 4-10. Переключение MCLK с DCOCLK на LFXT1CLK

1) Текущий тактовый цикл продолжается до следующего фронта сигнала исходного источника.
2) Уровень тактового сигнала (MCLK) остается высоким до следующего фронта сигнала нового источника.
3) Выбирается новый источник тактирования и далее MCLK продолжает работать от него, начиная с полного периода нового источника.

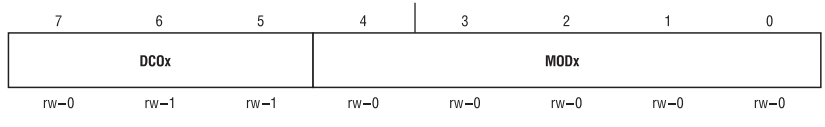
4.3. Регистры основного модуля тактирования

Перечень регистров основного модуля тактирования приведен в таблице 4.1.

Таблица 4-1. Регистры основного модуля тактирования

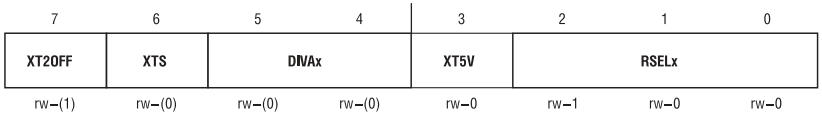
Регистр	Обозначение	Тип регистра	Адрес	Исходное состояние
Регистр управления DCO	DCOCTL	Чтение / запись	056h	060h после PUC
Регистр 1 управления системой основного тактирования	BCSCTL1	Чтение / запись	057h	084h после PUC
Регистр 2 управления системой основного тактирования	BCSCTL2	Чтение / запись	058h	Сбрасывается после POR
Регистр 1 разрешения прерываний SFR	IE1	Чтение / запись	000h	Сбрасывается после PUC
Регистр 1 флагов прерываний SFR	IFG1	Чтение / запись	002h	Сбрасывается после PUC

DCOCTL, регистр управления DCO



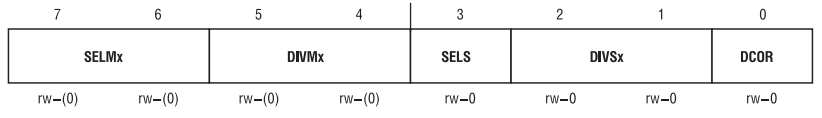
DCOx	Биты 7-5	Выбор частоты DCO. Эти биты определяют, какая из восьми дискретных частот DCO будет использоваться при выбранных установках RSELx.
MODx	Биты 4-0	Выбор модулятора. Эти биты определяют, как часто частота fDCO+1 используется в течение периода 32 циклов DCOCLK. В течение оставшихся циклов (32-MOD) используется частота fDCO. Эти биты не задействуются, когда DCOx=7.

BCSCTL1, регистр 1 управления системой основного тактирования



XT2OFF	Бит 7	Выключение XT2. Этот бит выключает осциллятор XT2. 0 – включение XT2 1 – выключение XT2, если он не используется для MCLK или SMCLK.
XTS	Бит 6	Выбор режима LFXT1 0 – режим низкой частоты 1 – режим высокой частоты
DIVAx	Биты 5-4	Делитель для ACLK 00 – /1 01 – /2 10 – /4 11 – /8
XT5V	Бит 3	Не задействован. XT5V должен быть всегда сброшен.
RSELx	Биты 2-0	Выбор резистора. Внутренний резистор имеет восемь различных величин сопротивления. Значение сопротивления резистора определяет номинальную частоту. Низшая номинальная частота выбирается при установке RSELx=0.

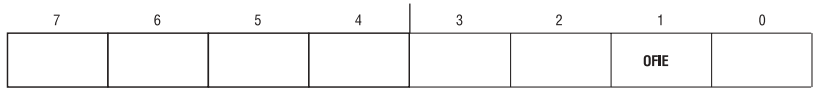
BCSCTL2, регистр 2 управления системой основного тактирования



SELMx	Биты 7-6	Выбор MCLK. Эти биты позволяют выбрать источник для MCLK. 00 – DCOCLK 01 – DCOCLK 10 – XT2CLK, когда XT2 имеется в микросхеме. LFXT1CLK, когда XT2 отсутствует. 11 – LFXT1CLK
DIVMx	Биты 5-4	Делитель для MCLK 00 – /1 01 – /2 10 – /4 11 – /8

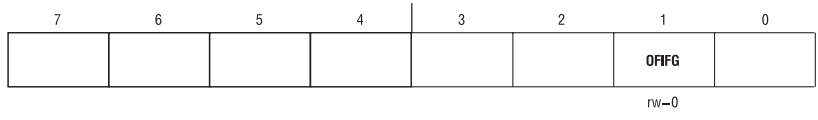
SELS	Бит 3	Выбор SMCLK. Эти биты позволяют выбрать источник для SMCLK. 0 – DCOCLK 1 – XT2CLK, когда XT2 имеется в микросхеме. LFXT1CLK, когда XT2 отсутствует.
DIVSx	Биты 2-1	Делитель для SMCLK 00 – /1 01 – /2 10 – /4 11 – /8
DCOR	Бит 0	Выбор резистора для DCO. 0 – Внутренний резистор 1 – Внешний резистор

IE1, регистр 1 разрешения прерываний



	Биты 7-2	Эти биты могут быть использованы другими модулями. См. справочное руководство конкретной микросхемы.
OFIE	Бит 1	Разрешение прерывания при возникновении ошибки осциллятора. Этот бит разрешает прерывание OFIFG. Поскольку другие биты в регистре IE1 могут использоваться для других устройств, рекомендуется вместо команд `MOV.B` или `CLR.B` применять команды `BIS.B` или `BIC.B`. 0 – Прерывание запрещено 1 – Прерывание разрешено
	Бит 0	Этот бит может быть использован другими модулями. См. справочное руководство конкретной микросхемы.

IFG1, регистр 1 флагов прерываний



	Биты 7-2	Эти биты могут быть использованы другими модулями. См. справочное руководство конкретной микросхемы.
OFIFG	Бит 1	Флаг прерывания при возникновении ошибки осциллятора. Поскольку остальные биты в регистре IFG1 могут использоваться для других устройств, рекомендуется вместо команд `MOV.B` или `CLR.B` применять команды `BIS.B` или `BIC.B`. 0 – Прерывание не ожидается 1 – Прерывание ожидается
	Бит 0	Этот бит может быть использован другими модулями. См. справочное руководство конкретной микросхемы.

MSP430x1xxFamily

Контроллер флэш-памяти

Раздел V.

Texas Instruments

Контроллер флэш-памяти

В этом разделе описывается работа контроллера флэш-памяти семейства MSP430.

5.1. Введение в флэш-память

Флэш-память в MSP430 адресуется побитно, побайтно или пословно и может перепрограммироваться. Модуль флэш-памяти имеет интегрированный контроллер, управляющий процессом стирания и программирования. Контроллер имеет три регистра, тактовый генератор и генератор напряжения для обеспечения напряжений стирания и программирования.

Флэш-память в MSP430 обладает следующими возможностями:

- внутренний генератор напряжения для программирования;
- программирование битов, байтов или слов;
- работа при ультранизком потреблении мощности;
- стирание сегмента или массовое (полное) стирание.

Блок-схема флэш-памяти и контроллера показана на рис. 5.1.

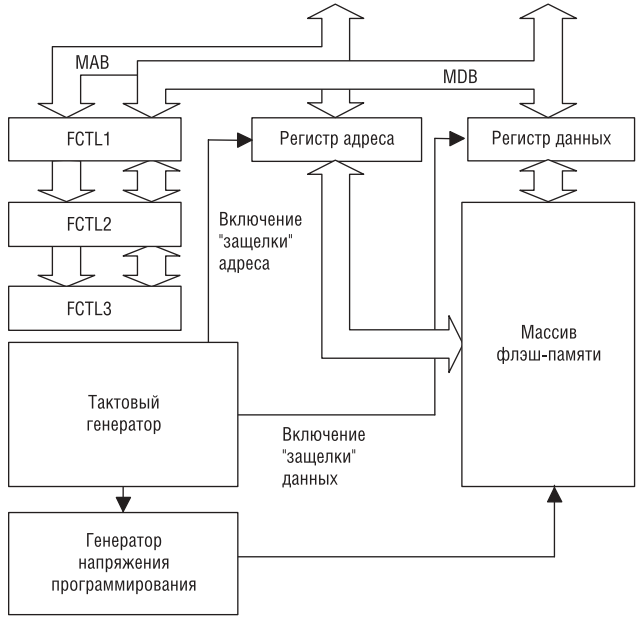


Рис. 5-1. Блок-схема модуля флэш-памяти

Примечание: Минимальное напряжение VCC во время записи или стирания флэш-памяти

Минимальное значение напряжения VCC во время записи или стирания флэш-памяти должно составлять 2,7 В. Если VCC падает ниже 2,7 В во время записи или стирания, результат записи или стирания будет непредсказуемым.

5.2. Сегментация флэш-памяти

Флэш-память в MSP430 разбита на сегменты. В неё может быть записан один бит, байт или слово, но сегмент – это минимальный размер флэш-памяти, который можно стереть. Три режима стирания позволяют стереть один сегмент, стереть все главные сегменты или стереть все сегменты (основные и информационные сегменты).

Флэш-память разделена на основной и информационный разделы памяти. Нет никаких различий в работе основного и информационного разделов памяти. Программный код или данные могут быть расположены в любом разделе. Различие между этими двумя разделами заключается в разных размерах сегмента и различных физических адресах.

Информационная память имеет два 128-байтных сегмента (в устройствах MSP430F1101 есть только один сегмент). Основная память имеет два или более 512-байтных сегмента. См. справочное руководство конкретного устройства для выяснения точной карты памяти.

На рис.5.2 показана сегментация памяти на основе примера 4 кБ флэш-памяти, имеющей восемь основных сегментов и оба информационных сегмента.

5.3. Функционирование флэш-памяти

Режим по умолчанию для флэш-памяти – режим чтения. В этом режиме флэш-память не может быть стерта или записана, тактовый генератор и генератор напряжения выключены – память работает подобно ПЗУ.

Флэш-память MSP430 поддерживает внутрисистемное программирование (ISP) и не нуждается в использовании дополнительного внешнего напряжения. ЦПУ может программировать собственную флэш-память. Приведенные ниже режимы записи/стирания флэш-памяти выбираются битами BLKWRT, WRT, MERAS, ERASE:

- запись байта/слова
- запись блока
- стирание сегмента
- массовое стирание (стирание всех сегментов основной памяти)
- полное стирание (стирание всех сегментов)

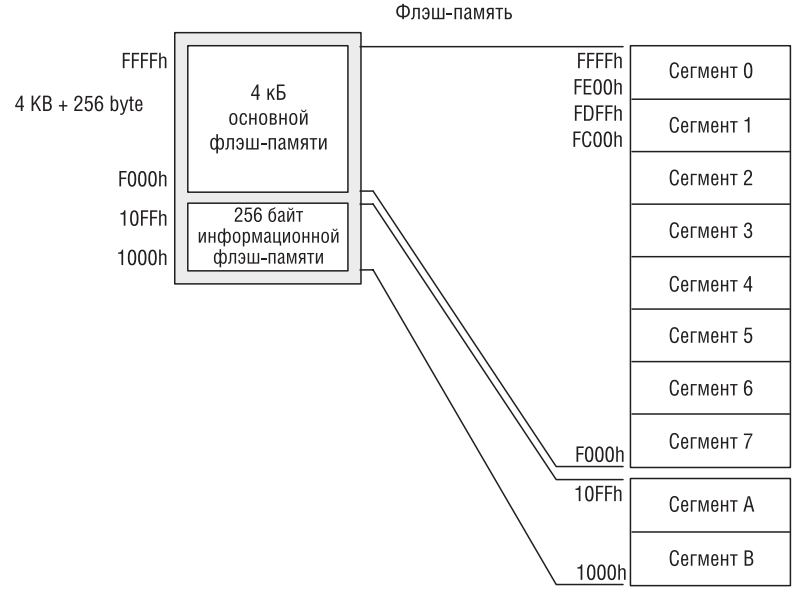


Рис. 5-2. Сегменты флэш-памяти, пример для 4кБ

Чтение или запись флэш-памяти во время программирования или стирания запрещены. Если требуется выполнение программы ЦПУ в течении записи или стирания, исполняемый код должен быть помещен в ОЗУ. Любое обновление флэш может инициироваться из флэш-памяти или ОЗУ.

5.3.1. Тактовый генератор флэш-памяти

Операции записи и стирания управляются тактовым генератором флэш-памяти, показанным на рис. 5.3. Рабочая частота f(FTG) тактового генератора

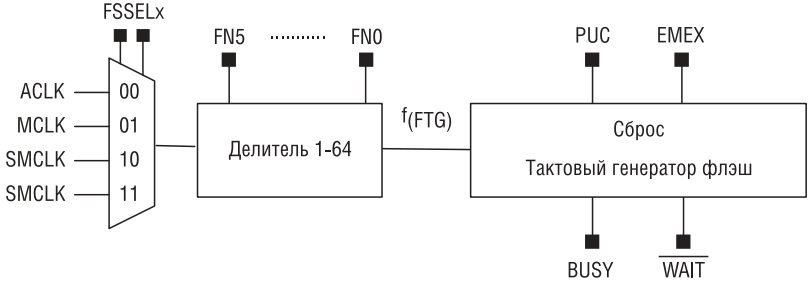


Рис. 5-3. Блок-схема тактового генератора флэш-памяти

должна лежать в диапазоне от ~ 257 кГц до ~ 476 кГц (точные данные см. в руководстве по конкретному устройству).

Тактовый генератор флэш-памяти может тактироваться от ACLK, SMCLK или MCLK. Тактовый сигнал выбранного источника должен быть поделен с помощью битов FNx для обеспечения необходимых требований к частоте f(FTG). Если появляется девиация (отклонение) частоты f(FTG) от требуемого значения в ходе записи или стирания, результат записи или стирания может быть непредсказуемым или же флэш-память окажется подвергнутой ударной перегрузке сверх допустимых пределов, гарантирующих надежную работу.

5.3.2. Стирание флэш-памяти

После стирания бит флэш-памяти читается как «1». Можно программировать индивидуально каждый бит, меняя его значение с «1» на «0», но перепрограммирование от «0» к «1» требует выполнения цикла стирания. Сегмент – это наименьшее количество флэш-памяти, которое можно стереть. Существует три режима стирания, которые могут быть выбраны с помощью битов ERASE и MERAS в соответствии с таблицей 5-1.

Таблица 5-1. Режимы стирания

MERAS	ERASE	Режим стирания
0	1	Стирание сегмента
1	0	Массовое стирание (стирание всех сегментов основной памяти)
1	1	Стирание всей флэш-памяти (основных и информационных сегментов)

Любое стирание инициируется фиктивной записью[1] в адресный диапазон, который будет стерт. Фиктивная запись запускает тактовый генератор флэш-памяти и процедуру стирания. На рис. 5.4 показан временной цикл процесса стира-

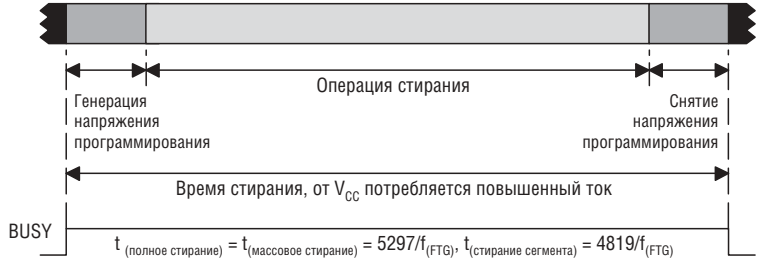


Рис. 5-4. Временная диаграмма цикла стирания

[1] Имеется ввиду выполнение реальной команды записи в флэш-память любых незначащих данных для запуска процедуры стирания.

ния. Бит BUSY устанавливается немедленно после фиктивной записи и остается установленным в течение всего цикла стирания. Биты BUSY, MERAS и ERASE автоматически очищаются, когда цикл завершен. Временные параметры цикла стирания не зависят от объема представленной в устройстве флэш-памяти. Продолжительность цикла стирания одинакова для всех устройств MSP430.

Фиктивная запись по адресу, который лежит вне диапазона стирания не приводит к запуску цикла стирания, не воздействует на флэш-память и не влияет на флаги. Такая ошибочная фиктивная запись игнорируется.

Прерывания должны быть отключены перед началом цикла стирания флэш-памяти. После завершения цикла стирания прерывания могут быть разрешены вновь. Любое прерывание, произошедшее во время цикла стирания, вызовет установку соответствующего флага, а после разрешения прерываний будет сгенерирован запрос на обработку прерывания.

Инициирование процедуры стирания из программы, находящейся в флэш-памяти

Любой цикл стирания может быть инициирован программой, находящейся как во флэш-памяти, так и в ОЗУ. Когда стирание сегмента инициировано программой из флэш-памяти, все тактирование выполняется контроллером флэш-памяти, а ЦПУ останавливается до завершения цикла стирания. После окончания цикла стирания ЦПУ продолжает выполнение программного кода с команды, следующей за фиктивной записью.

Когда цикл стирания инициируется программой их флэш-памяти, возможно стирание кода, необходимого для выполнения после завершения

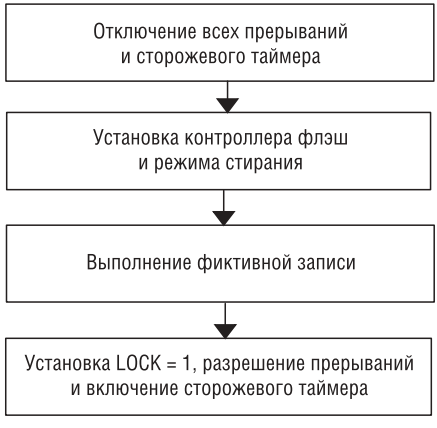


Рис. 5-5. Цикл стирания, инициируемый программой из флэш-памяти

стирания. Если это произойдет, работа ЦПУ после окончания цикла стирания будет непредсказуема.

Программный поток, инициирующий стирание из флэш-памяти, показан на рис. 5.5.

```
;Стирание сегмента из флэш. 514 кГц < SMCLK < 952 кГц
;Принимается ACCVIE = NMIIE = OFIE = 0.
MOV  #WDTPW+WDTHOLD,&WDTCTL   ;Отключение WDT
DINT                          ;Запрещение прерываний
MOV  #FWKEY+FSSEL1+FN0,&FCTL2 ;SMCLK/2
MOV  #FWKEY,&FCTL3            ;Очистка LOCK
MOV  #FWKEY+ERASE,&FCTL1      ;Разрешение стирания
                              ;сегмента
CLR  &0FC10h                  ;Фиктивная запись,
                              ;стирание S1
MOV  #FWKEY+LOCK,&FCTL3       ;Выполнено, установка
                              ;LOCK
...                           ;Повторное включение
                              ;WDT?
EINT                          ;Разрешение прерываний
```

Инициирование процедуры стирания программой из ОЗУ

Любой цикл стирания может быть инициирован из ОЗУ. В этом случае ЦПУ не приостанавливается, и может продолжать выполнять код из ОЗУ. Доступ ЦПУ к любому адресу флэш-памяти возможен после окончания цикла стирания, которое определяется путем опроса бита BUSY. Попытка доступа к флэш-памяти, когда BUSY=1 приведет к нарушению доступа с последующей установкой флага ACCVIFG и непредсказуемым результатам процедуры стирания.

Программный поток стирания из флэш-памяти программой из ОЗУ показан на рис. 5.6.

```
;Стирание сегмента программой из ОЗУ. 514 кГц<SMCLK
<952 кГц
;Принимается ACCVIE = NMIIE = OFIE = 0.
   MOV #WDTPW+WDTHOLD,&WDTCTL ;Отключение WDT
   DINT                       ;Запрещение прерываний
L1 BIT #BUSY,&FCTL3           ;Проверка BUSY
   JNZ L1                     ;Ожидание, пока занято
   MOV #FWKEY+FSSEL1+FN0,&FCTL2 ;SMCLK/2
```

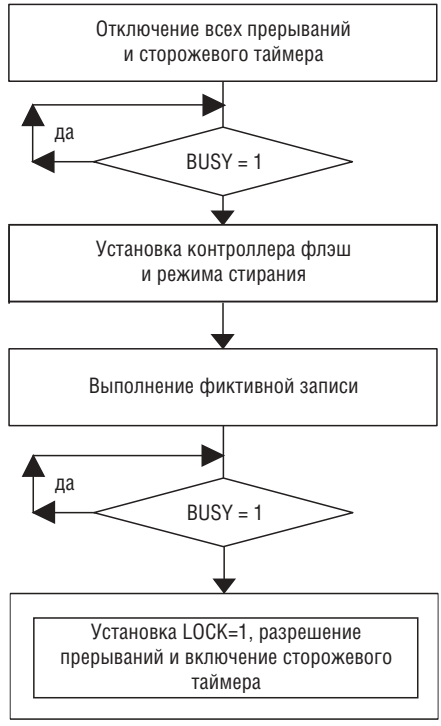


Рис. 5-6. Цикл стирания, инициируемый программой из ОЗУ

```
    MOV #FWKEY,&FCTL3           ;Очистка LOCK
    MOV #FWKEY+ERASE,&FCTL1     ;Разрешение стирания
    CLR &0FC10h                 ;Фиктивная запись,
                                ;стирание S1
L2  BIT #BUSY,&FCTL3            ;Проверка BUSY
    JNZ L2                      ;Ожидание, пока занято
    MOV #FWKEY+LOCK,&FCTL3      ;Завершено, установка
                                ;LOCK
    ...                         ;Повторное включение
                                ;WDT?
    EINT                        ;Разрешение прерываний
```

5.3.3. Запись в флэш-память

Режимы записи, задаваемые битами WRT и BLKWRT приведены в таблице 5.2.

Таблица 5-2. Режимы записи

BLKWRT	WRT	Режим записи
0	1	Запись байта/слова
1	1	Запись блока

Каждый из режимов записи использует последовательность собственных команд записи, но режим блочной записи позволяет выполнять запись примерно вдвое быстрее по сравнению с режимом байт/слово, поскольку генератор напряжения остается включенным до завершения записи блока. Любая команда, модифицирующая получателя может использоваться для изменения месторасположения в флэш-памяти как в режиме записи байта/слова, так и в режиме блочной записи.

Бит BUSY установлен, пока активна процедура записи и очищается, когда запись завершена. Если операция записи инициирована из ОЗУ, ЦПУ не должен обращаться к флэш-памяти, пока BUSY=1. В противном случае произойдет нарушение прав доступа, будет установлен флаг ACCVIFG, а результат записи окажется непредсказуем.

Запись байта/слова

Операция записи байта/слова может инициироваться программой из флэш-памяти или из ОЗУ. Когда инициирование происходит из флэш-памяти, все тактирование осуществляется контроллером флэш-памяти, а ЦПУ ожидает завершения записи. После выполнения записи ЦПУ продолжает выполнение кода с команды, следующей за командой записи. Временная диаграмма процедуры записи байта/слова показана на рис. 5.7.

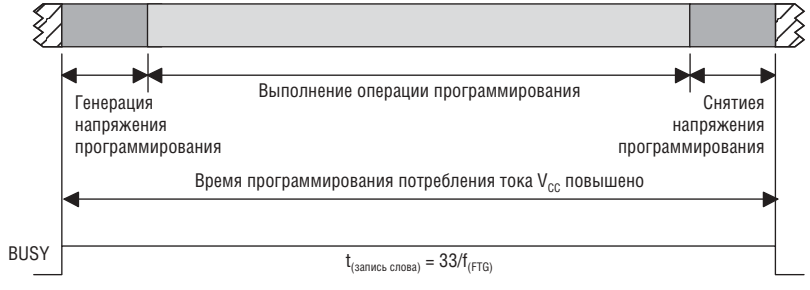


Рис. 5-7. Временная диаграмма операции записи байта/слова

Когда запись байта/слова выполняется из ОЗУ, ЦПУ продолжает выполнять код из ОЗУ. Бит BUSY должен стать равным нулю, прежде чем ЦПУ обратится к флэш-памяти снова, иначе произойдет нарушение прав доступа и установка флага ACCVIFG, а результат записи будет непредсказуем.

Инициирование записи байта/слова программой из флэш-памяти

Программный поток, инициирующий запись байта/слова из флэш-памяти показан на рис. 5.8.

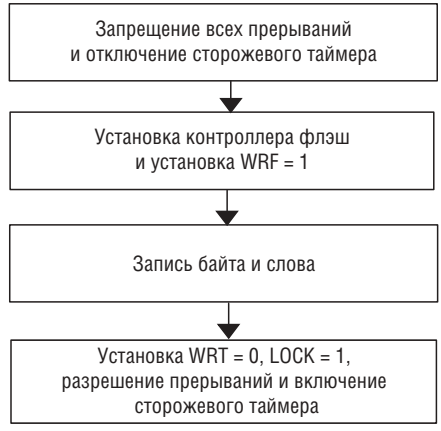


Рис. 5-8. Инициирование записи байта/слова из флэш-памяти

```
;Запись байта/слова из флэш. 514 кГц<SMCLK<952 кГц
;Принимается, что 0FF1Eh уже стерто
;Принимается ACCVIE = NMIIE = OFIE = 0.
MOV  #WDTPW+WDTHOLD,&WDTCTL  ;Отключение сторожевого
                             ;таймера
DINT                         ;Запрещение прерываний
MOV  #FWKEY+FSSEL1+FN0,&FCTL2 ;SMCLK/2
MOV  #FWKEY,&FCTL3           ;Очистка LOCK
MOV  #FWKEY+WRT,&FCTL1       ;Разрешение записи
MOV  #0123h,&0FF1Eh          ;0123h -> 0FF1Eh
MOV  #FWKEY,&FCTL1           ;Выполнено. Очистка WRT
MOV  #FWKEY+LOCK,&FCTL3      ;Установка LOCK
...                          ;Повторный запуск
                             ;сторожевого таймера?
EINT                         ;Разрешение прерываний
```

Инициирование записи байта/слова программой из ОЗУ

Программный поток, инициирующий запись байта/слова из ОЗУ показан на рис. 5.9.

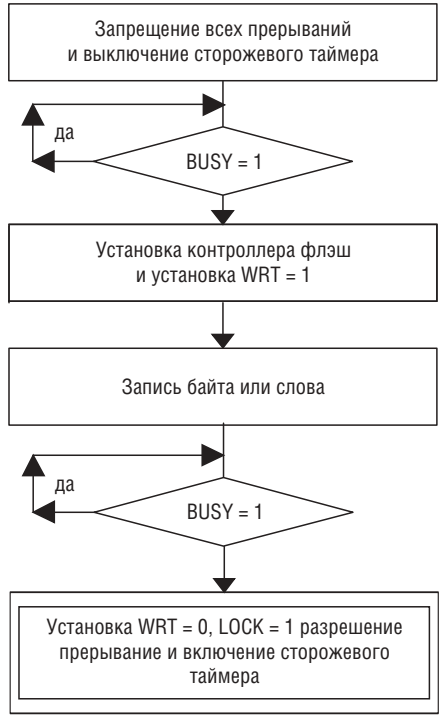


Рис. 5-9. Инициирование записи байта/слова из ОЗУ

```
;Запись байта/слова из ОЗУ. 514 кГц < SMCLK < 952 кГц
;Принимается, что 0FF1Eh уже стерто
;Принимается ACCVIE = NMIIE = OFIE = 0.
    MOV #WDTPW+WDTHOLD,&WDTCTL ;Отключение сторожевого
                               ;таймера
    DINT                       ;Запрещение прерываний
L1  BIT #BUSY,&FCTL3           ;Проверка BUSY
    JNZ L1
    MOV #FWKEY+FSSEL1+FN0,&FCTL2 ;SMCLK/2
    MOV #FWKEY,&FCTL3          ;Очистка LOCK
```

```
    MOV #FWKEY+WRT,&FCTL1       ;Разрешение записи
    MOV #0123h,&0FF1Eh          ;0123h -> 0FF1Eh
L2  BIT #BUSY,&FCTL3            ;Проверка BUSY
    JNZ L2                      ;Ожидание в цикле,
                                ;пока занято
    MOV #FWKEY,&FCTL1           ;Очистка WRT
    MOV #FWKEY+LOCK,&FCTL3      ;Установка LOCK
...                             ;Повторный запуск
                                ;сторожевого таймера?
    EINT                        ;Разрешение прерываний
```

Запись блока

Блочную запись можно использовать для ускорения процесса записи во флэш-память большой последовательности байт или слов. Блок – это 64 байта, начиная с 0xx00h, 0xx40h, 0xx80h или 0xxC0h и заканчивая 0xx3Fh, 0xx7Fh, 0xxBFh или 0xxFFh, как показано на рис. 5.10. Напряжение программирования флэш-памяти остается поданным в течение записи блока из 64-байт.

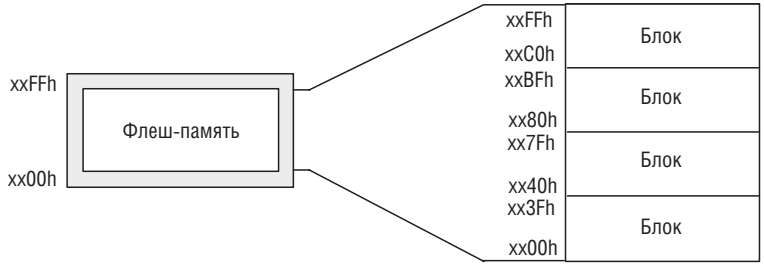


Рис. 5-10. Блоки флэш-памяти

Блочная запись не может быть инициирована из флэш-памяти. Блочная запись должна инициироваться только из ОЗУ или ПЗУ. Бит BUSY остается установленным в течение всего цикла записи блока. Бит WAIT должен проверяться между записью каждого байта или слова в блоке. Очередной байт или слово блока могут быть записаны, когда бит WAIT установлен. При записи последовательности блоков бит BLKWRT необходимо очищать после завершения записи текущего блока. Бит BLKWRT может быть установлен для инициирования записи следующего блока после выдержки заданного времени восстановления флэш t(end). Бит BUSY очищается после завершения записи каждого блока, информируя о возможности записи следующего блока. На рис. 5.11 показана временная диаграмма процедуры блочной записи.

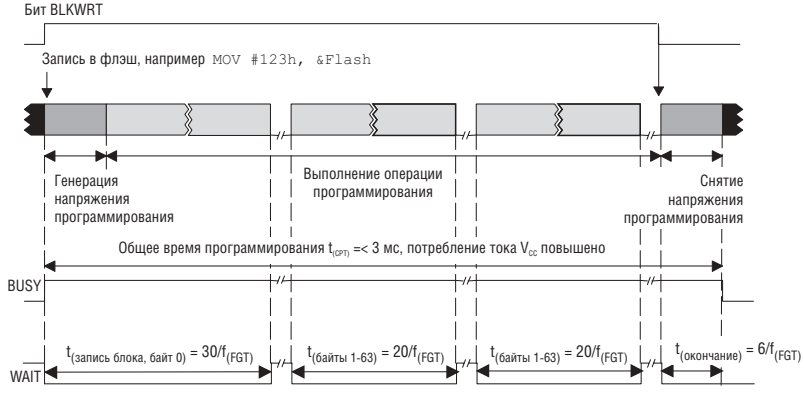


Рис. 5-11. Временная диаграмма цикла блочной записи

Программный поток записи блока и пример

Программный поток записи блока показан на рис. 5.12, а ниже приводится соответствующий пример.

```
;Запись одного блока, начиная с адреса 0F000h.
;Запись должна выполняться из ОЗУ; предполагается,
что флэш-память уже стерта.
;514 кГц < SMCLK < 952 кГц
;Принимается ACCVIE = NMIIE = OFIE = 0.
    MOV #32,R5                  ;Используется как
                                ;счетчик записи
    MOV #0F000h,R6              ;Указатель записи
    MOV #WDTPW+WDTHOLD,&WDTCTL ;Отключение
                                ;сторожевого таймера
    DINT                        ;Запрещение прерываний
L1  BIT #BUSY,&FCTL3            ;Проверка BUSY
    JNZ L1                      ;Ожидание в цикле,
                                ;пока занято
    MOV #FWKEY+FSSEL1+FN0,&FCTL2 ;SMCLK/2
    MOV #FWKEY,&FCTL3           ;Очистка LOCK
    MOV #FWKEY+BLKWRT+WRT,&FCTL1 ;Разрешение записи
                                 ;блока
L2  MOV Write_Value,0(R6)       ;Месторасположение
                                ;записи
L3  BIT #WAIT,&FCTL3            ;Проверка WAIT
```

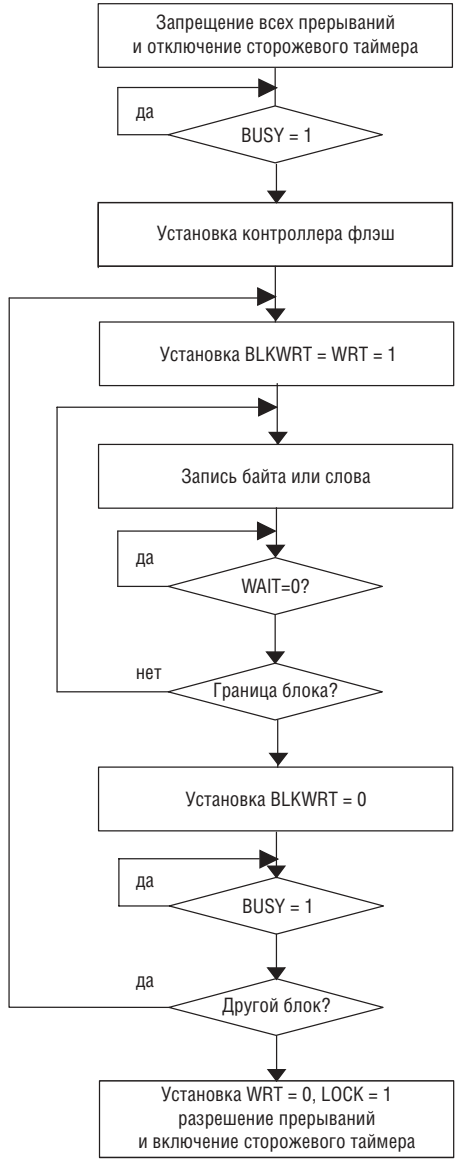


Рис. 5-12. Программный поток блочной записи

```
    JZ L3                          ;Ожидание в цикле,
                                   ;пока WAIT=0
    INCD R6                  ;Указание на следующее слово
    DEC R5                   ;Декремент счетчика записи
    JNZ L2                   ;Конец блока?
    MOV #FWKEY,&FCTL1        ;Очистка WRT,BLKWRT
L4  BIT #BUSY,&FCTL3         ;Проверка BUSY
    JNZ L4                   ;Ожидание в цикле,
                             ;пока занято
    MOV #FWKEY+LOCK,&FCTL3 ;Установка LOCK
...                          ;Повторный запуск
                             ;сторожевого таймера,
                             ;если необходимо
    EINT                     ;Разрешение прерываний
```

5.3.4. Доступ к флэш-памяти во время записи или стирания

Когда выполняется любая операция записи или стирания, инициированная из ОЗУ и BUSY=1, ЦПУ не может выполнять чтение или запись любой ячейки флэш-памяти. В противном случае произойдет нарушение прав доступа, будет установлен флаг ACCVIFG и результат окажется непредсказуемым. Также, если запись во флэш-память предпринята с WRT=0, устанавливается флаг прерывания ACCVIFG, а содержимое флэш-памяти не изменяется.

Когда инициируется запись байта/слова или любая операция стирания программой из флэш-памяти, контроллер флэш возвращает ЦПУ код операции 03FFFh при выборке следующей команды. Код операции 03FFFh – это команда JMP PC. Это приведет к зацикливанию ЦПУ, пока работа с флэш не будет закончена. Когда операция с флэш-памятью закончена и BUSY=0, контроллер флэш позволяет ЦПУ выполнить выборку правильного кода операции и выполнение программы возобновляется.

Условия доступа к флэш-памяти, когда BUSY=1 приведены в таблице 5.3.

Таблица 5-3. Доступ к флэш-памяти при BUSY=1

Операция с флэш-памятью	Доступ к флэш-памяти	WAIT	Результат
Любой режим стирания или запись байта/слова	Чтение	0	ACCVIFG = 1, читается значение 03FFFh
	Запись	0	ACCVIFG = 1. Запись игнорируется
	Выборка команды	0	ACCVIFG = 0. CPU считывает код 03FFFh. Это команда JMP PC.

Операция с флэш-памятью	Доступ к флэш-памяти	WAIT	Результат
Запись блока	Любой	0	ACCVIFG = 1, LOCK = 1
	Чтение	1	ACCVIFG = 0, читается значение 03FFFh
	Запись	1	ACCVIFG = 0. Запись игнорируется
	Выборка команды	1	ACCVIFG = 1, LOCK = 1

Все источники прерываний необходимо заблокировать перед инициализацией любой операции с флэш-памятью. Если бы разрешенное прерывание произошло во время операции с флэш-памятью, ЦПУ сделало бы выборку кода 03FFFh в качестве адреса процедуры обработки прерывания. ЦПУ выполнило бы команду JMP PC при BUSY=1. После завершения операции с флэш-памятью, ЦПУ начало бы выполнение кода с адреса 03FFFh, который не является правильным адресом процедуры обработки прерывания.

5.3.5. Останов цикла записи или стирания

Любая операция записи или стирания может быть остановлена до момента нормального завершения путем установки бита аварийного выхода EMEX. Установка бита EMEX немедленно останавливает активную операцию и контроллер флэш-памяти. Все операции с флэш-памятью прекращаются, она возвращается в режим чтения, а все биты в регистре FCTL1 сбрасываются. Результат предполагавшейся операции с флэш-памятью будет непредсказуем.

5.3.6. Конфигурирование и доступ к контроллеру флэш-памяти

FCTLx – это 16-разрядные регистры записи/чтения, защищенные паролем. Любая операция чтения или записи доступна только при использовании команды-слова, а запись возможна только при наличии в старшем байте пароля записи 0A5h. Любая запись в любой FCTLx регистр с любым значением в старшем байте, отличном от 0A5h вызовет нарушение ключа защиты, установку флага KEYV и запуск системного сброса PUC. При любом чтении любого регистра FCTLx результат содержит в старшем байте значение 096h.

Любая запись в FCTL1 во время стирания или операции записи байта/слова приведет к нарушению прав доступа и установке флага ACCVIFG. Запись в FCTL1 возможна в режиме блочной записи, когда WAIT=1, однако запись в FCTL1 в режиме блочной записи, когда WAIT=0 приведет к нарушению прав доступа и установке флага ACCVIFG.

Любая запись в FCTL2, когда BUSY=1 приведет к нарушению прав доступа.

Любой FCTLx регистр может быть прочитан, когда BUSY=1. Чтение не приведет к нарушению прав доступа.

5.3.7. Прерывания контроллера флэш-памяти

Контроллер флэш имеет два источника прерывания: KEYV и ACCVIFG. Флаг ACCVIFG устанавливается, когда происходит нарушение прав доступа. Когда бит ACCVIE устанавливается вновь после записи или стирания флэш-памяти, установленный флаг ACCVIFG будет генерировать запрос прерывания. Флаг ACCVIFG – источник вектора немаскируемого прерывания NMI, поэтому нет необходимости устанавливать GIE для запроса прерывания по флагу ACCVIFG. Помимо этого, ACCVIFG можно проверить программно, чтобы определить, было ли нарушение прав доступа. Флаг ACCVIFG должен сбрасываться программно.

Флаг нарушения ключа KEYV устанавливается, когда выполняется запись в любой управляющий регистр контроллера флэш с неправильным паролем. Когда это происходит, генерируется сигнал PUC, немедленно сбрасывая устройство.

5.3.8. Программирование устройств с флэш-памятью

Имеется три способа программирования флэш-устройств MSP430. Все способы поддерживают внутрисистемное программирование (ISP):

- Программирование через JTAG[1]
- Программирование через самозагрузчик
- Программирование через пользовательское решение

Программирование флэш-памяти через JTAG

Устройства MSP430 могут программироваться через JTAG-порт. Для JTAG-интерфейса нужны четыре сигнальных линии (5 сигнальных линий у 20 и 28-выводных устройств), общий провод и опционально VCC и nonRST/NMI.

JTAG-порт защищен с помощью предохранителей. Перегорание предохранителей явление необратимое – в результате срабатывания предохранителя JTAG-порт отключается. Последующий доступ к устройству через JTAG-порт становится невозможен. Подробности см. в приложении *«Programming a Flash-Based MSP430 Using the JTAG Interface[2]»* на сайте *www.ti.com/sc/msp430*.

Программирование флэш-памяти через самозагрузчик (BSL)

Каждое MSP430 устройство с флэш-памятью содержит самозагрузчик BSL. Он позволяет пользователю читать или программировать флэш-память или ОЗУ с помощью последовательного интерфейса UART[3]. Доступ к флэш-памяти MSP430 через BSL защищен 256-разрядным паролем, определяе-

[1] JTAG (Joint Test Automation Group) interface – интерфейс «объединенной рабочей группы по автоматизации тестирования»

[2] «Программирование MSP430 с флэш-памятью через JTAG-интерфейс»

[3] UART (Universal Asynchronous Receiver / Transmitter) - универсальный асинхронный приемопередатчик

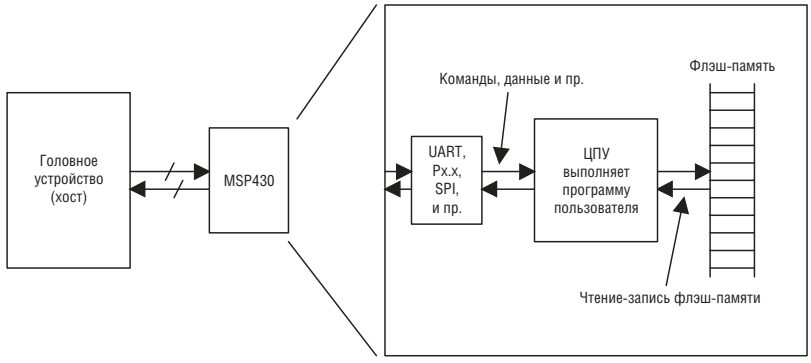


Рис. 5-13. Решение по программированию, разработанное пользователем

мым пользователем. Подробности см. в приложении «*Features of the MSP430 Bootstrap Loader*[1]» на сайте *www.ti.com/sc/msp430*.

Программирование флэш-памяти через пользовательское решение

Способность ЦПУ в MSP430 записывать собственную флэш-память позволяет реализовать внутрисистемное программирование внешними пользовательскими решениями, как показано на рис. 5.13. Пользователь может выбрать, каким образом данные будут поступать в MSP430 с использованием любого имеющегося доступного способа (UART, SPI и пр.). Разработанное пользователем программное обеспечение может получать данные и программировать флэш-память. Так как этот тип решения разработан пользователем, его можно настроить таким образом, чтобы наиболее полно удовлетворялись потребности в программировании, стирании и обновлении флэш-памяти.

5.4. Регистры флэш-памяти

Перечень регистров флэш-памяти приведен в таблице 5.4.

Таблица 5-4. Регистры флэш-памяти

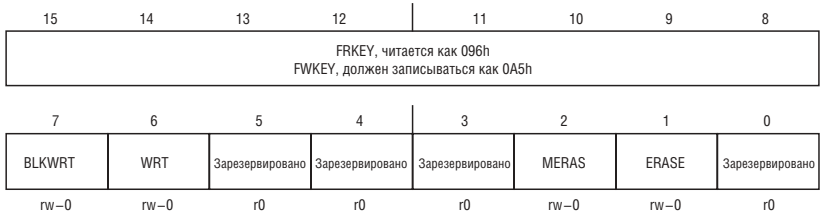
Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр 1 управления флэш-памятью	FCTL1	Чтение/запись	0128h	09600h с PUC

[1] «Возможности самозагрузчика MSP430»

Таблица 5-4. (Окончание)

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр 2 управления флэш-памятью	FCTL2	Чтение/запись	012Ah	09642h с PUC
Регистр 3 управления флэш-памятью	FCTL3	Чтение/запись	012Ch	09618h с PUC
Регистр 1 разрешения прерывания	IE1	Чтение/запись	000h	Сброс с PUC

FCTL1, регистр управления флэш-памятью



FRKEY/FWKEY	Биты 15-8	Пароль FCTLx. Всегда читается как 096h. Должен записываться как 0A5h, в противном случае будет генерироваться сигнал PUC.
BLKWRT	Бит 7	Режим блочной записи. Для режима блочной записи также должен быть установлен WRT. Бит BLKWRT автоматически сбрасывается при установке EMEX. 0 – Режим блочной записи выключен 1 – Режим блочной записи включен
WRT	Бит 6	Запись. Этот бит используется для выбора любого режима записи. Бит WRT автоматически сбрасывается при установке EMEX. 0 – Режим записи выключен 1 – Режим записи включен
Зарезервировано	Биты 5-3	Зарезервировано. Всегда читается как 0.
MERAS	Бит 2	Массовое стирание и обычное стирание. Эти биты используются совместно для выбора режима стирания. Биты MERAS и ERASE автоматически сбрасываются, когда устанавливается EMEX.
ERASE	Бит 1	
Зарезервировано	Бит 0	Зарезервировано. Всегда читается как 0.

MERAS	ERASE	Цикл стирания
0	0	Нет стирания
0	1	Стирание только конкретного сегмента
1	0	Стирание всех сегментов основной памяти
1	1	Стирание всех сегментов основной и информационной памяти

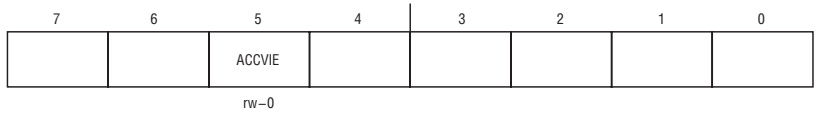
FCTL2, регистр управления флэш-памятью

FWKEY	**Биты 15-8**	Пароль FCTLx. Всегда читается как 096h. Должен записываться как 0A5h, в противном случае будет генерироваться сигнал PUC.
FSSELx	**Биты 7-6**	Выбор источника тактирования контроллера флэш 00 – ACLK 01 – MCLK 10 – SMCLK 11 – SMCLK
FNx	**Биты 5-0**	Делитель тактовой частоты для контроллера флэш. Эти шесть битов позволяют установить необходимый коэффициент деления для тактирования контроллера флэш. Значение коэффициента деления равно FNx+1. К примеру, когда FNx=00h, коэффициент деления равен 1. Когда FNx=02Fh, коэффициент деления равен 64.

FCTL3, регистр управления флэш-памятью

FWKEY	**Биты 15-8**	Пароль FCTLx. Всегда читается как 096h. Должен записываться как 0A5h, в противном случае будет генерироваться сигнал PUC.
Зарезервировано	**Биты 7-6**	Зарезервировано. Всегда читается как 0.
EMEX	**Бит 5**	Аварийный выход 0 – Нет аварийного выхода 1 – Аварийный выход
LOCK	**Бит 4**	Блокировка. Этот бит разблокирует флэш-память для выполнения записи или стирания. Бит LOCK может быть установлен в любой момент во время записи байта/слова или операции стирания, при этом выполняемая операция будет нормально завершена. В режиме блочной записи, если бит LOCK устанавливается, когда BLKWRT=WAIT=1, биты BLKWRT и WAIT сбрасываются и режим нормально заканчивается. 0 – Разблокировано 1 – Заблокировано
WAIT	**Бит 3**	Ожидание. Указывает, что происходит запись флэш-памяти. 0 – Флэш-память не готова для записи следующего байта/слова. 1 – Флэш-память готова для записи следующего байта/слова.
ACCVIFG	**Бит 2**	Флаг прерывания при нарушении прав доступа 0 – Прерывание не ожидается 1 – Ожидание прерывание
KEYV	**Бит 1**	Ключ нарушения безопасности флэш. Этот бит показывает, что был записан неправильный пароль FCTLx в любой регистр управления флэш-памятью и при его установке генерируется сигнал PUC. Бит KEYV должен быть сброшен программно. 0 – Был записан корректный пароль FCTLx 1 – Был записан некорректный пароль FCTLx
BUSY	**Бит 0**	Занято. Этот бит показывает состояние тактового генератора флэш. 0 – Не занят 1 – Занят

IE1, регистр 1 разрешения прерывания



	Биты 7-6, 4-0	Эти биты могут быть использованы для других модулей. См. справочной руководство конкретного устройства.
ACCVIE	**Бит 5**	Разрешение прерывания при нарушении доступа к флэш-памяти. Этот бит разрешает прерывание от ACCVIFG. Поскольку остальные биты в IE1 могут быть использованы для других модулей, рекомендуется устанавливать и очищать этот бит с помощью команд `BIS.B` или `BIC.B`, вместо команд `MOV.B` или `CLR.B`. 0 – Прерывание запрещено 1 – Прерывание разрешено

MSP430x1xxFamily

Супервизор напряжения питания

Раздел VI.

Texas Instruments

Супервизор напряжения питания

В этом разделе описывается работа супервизора напряжения питания (SVS[1]). Модуль SVS реализован в устройствах MSP430x15x и MSP430x16x.

6.1. Введение в SVS

Супервизор напряжения питания (SVS) используется для мониторинга напряжения питания AVCC или внешнего напряжения. SVS может быть сконфигурирован так, чтобы выполнялась установка флага или генерировался сигнал сброса POR, когда напряжение питания или внешнее напряжение снижаются ниже порога, установленного пользователем.

SVS обладает следующими возможностями:

- Мониторинг AVCC;
- Возможность генерации сигнала POR;
- Программно доступный вывод компаратора SVS;
- Программно доступное условие фиксации при низком напряжении;
- Выбор из 14 возможных пороговых уровней;
- Внешний канал мониторинга внешнего напряжения.

Блок-схема SVS показана на рис. 6.1.

6.2. Функционирование SVS

SVS определяет снижение напряжения AVCC ниже заданного уровня. Модуль SVS можно сконфигурировать на выработку сигнала POR или установку флага при снижении напряжения. После сигнала BOR модуль SVS отключается, чтобы сохранить потребление тока.

6.2.1. Конфигурирование SVS

Биты VLDx используются для включения/выключения SVS и выбора одного из 14 пороговых уровней (V(SYS_IT-)) для сравнения с AVCC. SVS выключен, когда VLDx=0 и включен, когда VLDx>0. Бит SVSON не включает SVS. Он показывает включенное/выключенное состояние модуля SVS и может использоваться для определения, включен ли SVS.

При VLDx=1111 выбирается внешний канал SVSin. Напряжение на SVSin сравнивается с внутренним уровнем напряжения, равным приблизительно 1,2 В.

6.2.2. Функционирование компаратора SVS

Состояние пониженного напряжения появляется, когда AVCC понижается меньше выбранного порога или когда внешнее напряжение снижается ниже

[1] SVS – Supply Voltage Supervisor.

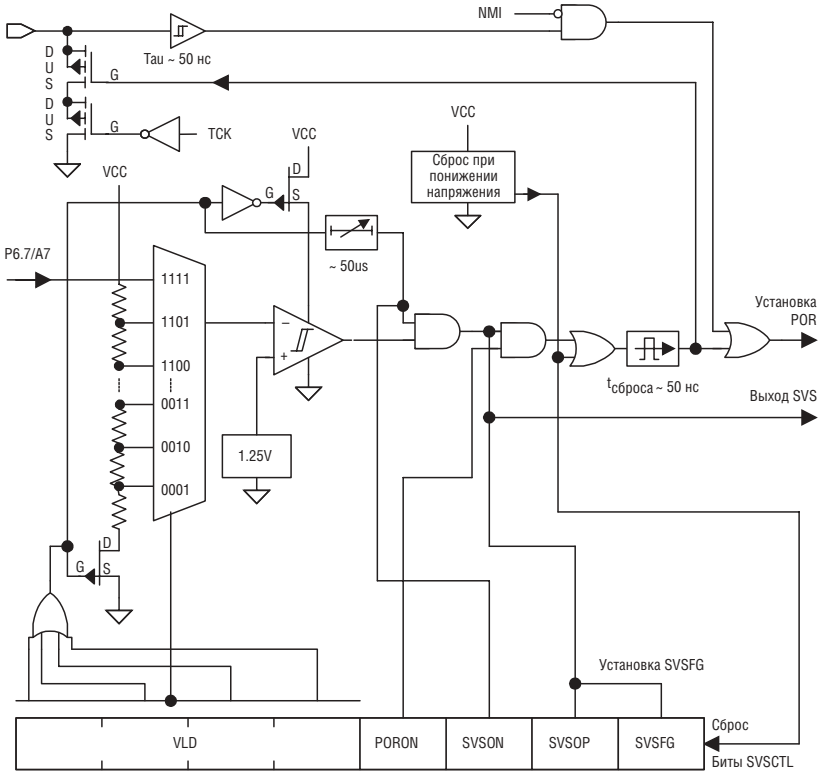


Рис. 6-1. Блок схема модуля SVS

порога в 1,2 В. Любое состояние пониженного напряжения устанавливает бит SVSFG.

Бит PORON включает или выключает функцию сброса устройства от SVS. Если PORON=1, при установке бита SVSFG генерируется сигнал POR. Если PORON=0, состояние пониженного напряжения устанавливает SVSFG, но не приводит к генерации сигнала POR.

Бит SVSFG при установке фиксируется. Благодаря этому пользователь может определить, что ранее произошло понижение напряжения. Бит SVSFG должен сбрасываться программным обеспечением пользователя. Если состояние пониженного напряжение остается в момент сброса бита SVSFG, он немедленно устанавливается снова модулем SVS.

6.2.3. Изменение битов VLDx

После изменения битов VLDx выдерживаются две установочных задержки, позволяющие установиться схеме SVS. В течение каждой задержки SVS не будет устанавливать SVSFG. Задержки td(SVSon) и tsettle показаны на рис. 6.2. Задержка td(SVSon) действует, когда VLDx изменяются от нуля к любому отличному от нуля значению, и составляет примерно 50 мкС. Задержка tsettle действует при изменении битов VLDx от любого ненулевого значения к любому другому ненулевому значению и составляет максимум ~12 мкС. Точные значения задержек см. в руководстве по конкретному устройству.

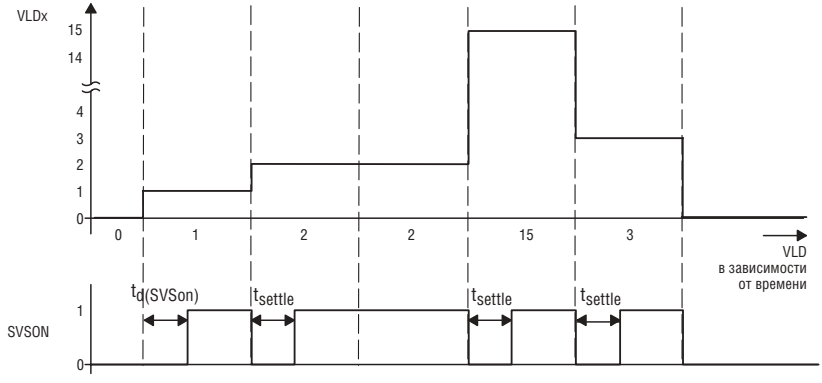


Рис. 6-2. Состояние бита SVSON при изменении VLDx

Во время задержек SVS не устанавливает флаг состояния пониженного напряжения и не сбрасывает устройство, а бит SVSON остается очищенным. Программное обеспечение может проверять бит SVSON для определения момента окончания задержки и начала достоверного мониторинга напряжения модулем SVS.

6.2.4. Рабочий диапазон SVS

Каждый уровень SVS имеет гистерезис для уменьшения чувствительности к малым изменениям питающего напряжения, когда величина AVCC близка к установленному порогу. Работа SVS и SVS/Brownout[1] взаимодействие показано на рис. 6.3.

[1] Brownout - понижение напряжения.

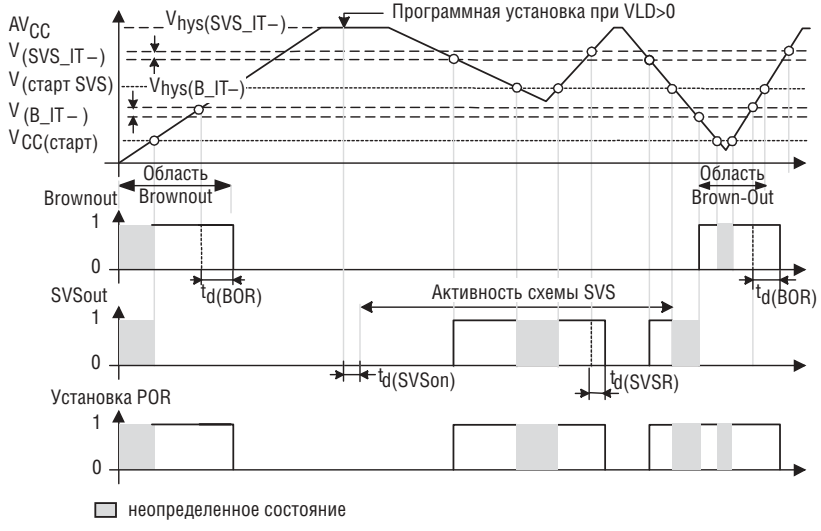


Рис. 6-3. Рабочие уровни для SVS и схемы Brownout/сброс

6.3. Регистры SVS

Перечень регистров SVS приведен в таблице 6.1.

Таблица 6-1. Регистры SVS

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Управляющий регистр SVS	SVSCTL	Чтение/запись	055h	Сбрасывается после POR

SVSCTL, регистр управления SVS

7	6	5	4	3	2	1	0
VLDx				PORON	SVSON	SVSOP	SVSFG
rw–0	rw–0	rw–0	rw–0	rw–0	r	r	rw–0

VLDx	**Биты7-4**	Детектируемый уровень напряжения. Эти биты включают SVS и позволяют выбрать номинальный пороговый уровень напряжения SVS. Точные параметры см. в руководстве на 0000 – SVS выключен

VLDx	**Биты 7-4**	0001 – 1.9В 0010 – 2.1В 0011 – 2.2В 0100 – 2.3В 0101 – 2.4В 0110 – 2.5В 0111 – 2.65В 1000 – 2.8В 1001 – 2.9В 1010 – 3.05В 1011 – 3.2В 1100 – 3.35В 1101 – 3.5В 1110 – 3.7В 1111 – Сравнение внешнего напряжения на входе SVSin со значением 1.2В
PORON	**Бит 3**	Включение POR. Этот бит разрешает флагу SVSFG вызывать сброс устройства сигналом POR. 0 – SVSFG не вызывает POR 1 – Установка SVSFG приводит к генерации POR
SVSON	**Бит 2**	Включение SVS. Этот бит отражает состояние работы SVS. Этот бит НЕ ВКЛЮЧАЕТ SVS. SVS включается установкой VLDx > 0. 0 – SVS выключен 1 – SVS включен
SVSOP	**Бит 1**	Выход SVS. Этот бит отражает выходное значение компаратора SVS. 0 – Выход компаратора SVS имеет высокий уровень 1 – Выход компаратора SVS имеет низкий уровень
SVSFG	**Бит 0**	Флаг SVS. Этот бит показывает состояние пониженного напряжения. Бит SVSFG остается установленным после устранения состояния пониженного напряжения до сброса программным обеспечением. 0 – Состояние пониженного напряжения не произошло 1 – Произошло либо уже присутствует состояние пониженного напряжения

Примечание: некорректная информация

Исходное состояние регистра SVSCTL указано неправильно. Содержимое регистра SVSCTL сбрасывается только по brownout-условию. Значение SVSCTL сохраняется при генерации сигнала POR и в случае низкого уровня на выводе RST/NMI (аппаратный сброс), и в случае, когда POR генерируется самим модулем SVS.

MSP430x1xxFamily

Аппаратный умножитель

Раздел VII.

Texas Instruments

Аппаратный умножитель

В этом разделе описывается аппаратный умножитель. Аппаратный умножитель реализован в устройствах MSP430x14x и MSP430x16x.

7.1. Введение в аппаратный умножитель

Аппаратный умножитель является периферийным устройством и не является частью ЦПУ MSP430. Это означает, что его действия не пересекаются с действиями ЦПУ. Регистры умножителя – это периферийные регистры, которые загружаются и читаются командами ЦПУ.

Аппаратный умножитель поддерживает:

- Умножение без знака;
- Умножение со знаком;
- Умножение без знака с накоплением;
- Умножение со знаком и накоплением;
- 16×16 бит, 16×8 бит, 8×16 бит, 8×8 бит.

Блок-схема аппаратного умножителя показана на рис. 7.1.

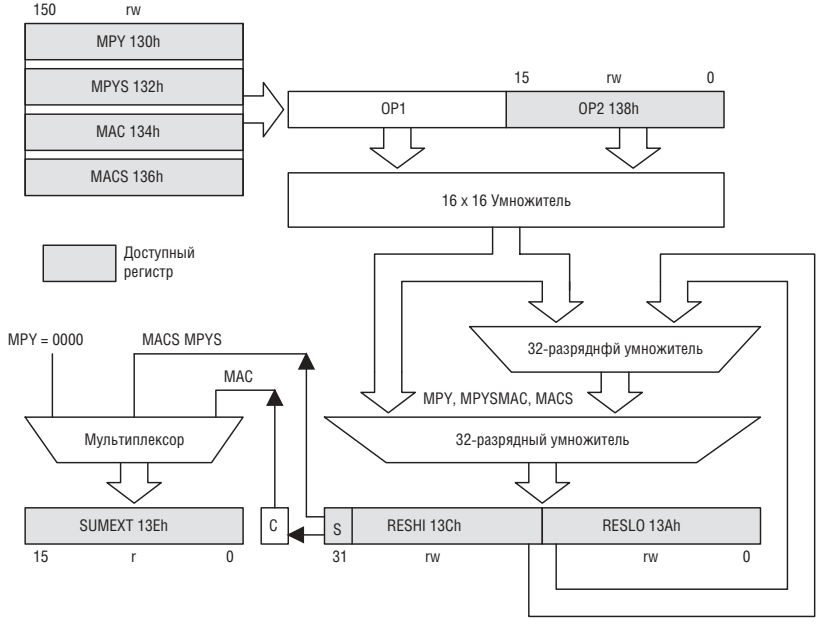


Рис. 7-1. Блок-схема аппаратного умножителя

7.2. Функционирование аппаратного умножителя

Аппаратный умножитель поддерживает операции умножения без знака, умножения со знаком, умножения без знака с накоплением и умножение со знаком и накоплением. Тип операции выбирается адресом, в который записан первый операнд.

Аппаратный умножитель имеет два 16-разрядных регистра OP1 и OP2 и три регистра результата RESLO, RESHI и SUMEXT. В регистре RESLO содержится младшее слово результата, в RESHI – старшее слово результата, а в регистре SUMEXT находится информация о результате. Для появления результата необходимо 3 такта MCLK. Результат может быть прочитан следующей командой после записи в OP2. Исключение составляет случай,когда используется косвенный режим адресации к регистру результата. В этом случае необходимо вставить команду NOP перед чтением результата.

7.2.1. Операнд регистров

Регистр OP1 первого операнда имеет четыре адреса, показанные в таблице 7.1, используемые при выборе режима умножения. Запись первого операнда по желаемому адресу позволит выбрать тип операции умножения, но не приведет к началу выполнения какой-либо операции. Запись второго операнда в регистр OP2 второго операнда инициирует операцию умножения. Запись в OP2 стартует выбранную операцию над значениями, сохраненными в OP1 и OP2. Результат записывается в три регистра результата RESLO, RESHI и SUMEXT.

Повторение операций умножения может выполняться без перезагрузки OP1, если значение в OP1 используется для последовательных операций. Нет необходимости перезаписывать значение в OP1 для выполнения операций.

Таблица 7-1. Адреса OP1

Адрес OP1	Имя регистра	Операция
0130h	MPY	Умножение без знака
0132h	MPYS	Умножение со знаком
0134h	MAC	Умножение без знака с накоплением
0136h	MACS	Умножение со знаком и накоплением

7.2.2. Регистры результата

Младший регистр результата RESLO содержит младшие 16 разрядов вычисленного результата. Содержимое старшего регистра результата RESHI зависит от операции умножения. Различные варианты содержимого RESHI приведены в таблице 7.2.

Таблица 7-2. Возможные варианты содержимого регистра RESHI

Режим	Содержимое RESHI
MPY	Старшие 16 разрядов результата
MPYS	В старшем бите MSB регистра находится знак результата. Оставшиеся биты содержат старшие 15 разрядов результата. Используется представление результата с дополнением до двух.
MAC	Старшие 16 разрядов результата
MACS	Старшие 16 разрядов результата. Используется представление результата с дополнением до двух.

Содержимое регистра расширенного суммирования SUMEXT зависит от выполненной операции умножения. Различные варианты содержимого SUMEXT приведены в таблице 7.3.

Таблица 7-3. Возможные варианты содержимого регистра SUMEXT

Режим	SUMEXT
MPY	SUMEXT всегда содержит 0000h
MPYS	SUMEXT содержит расширенный знак результата 00000h результат был положительный 0FFFFh результат был отрицательный
MAC	SUMEXT содержит перенос результата 0000h результат не содержит переноса 0001h результат имеет перенос
MACS	SUMEXT содержит расширенный знак результата 00000h результат был положительный 0FFFFh результат был отрицательный

Потеря значащих разрядов и переполнение в режиме MACS

Умножитель не может автоматически определить потерю значащих разрядов или переполнение в режиме MACS. Диапазон аккумулятора для положительных чисел равен 0 – 7FFF FFFFh, а для отрицательных чисел 0FFF FFFh – 8000 0000h. Переполнение происходит, когда результат суммирования двух отрицательных чисел выходит за диапазон для положительного числа. Потеря значащих разрядов происходит, когда результат сложения двух положительных чисел выходит за диапазон для отрицательного числа. В обоих случаях регистр SUMEXT содержит правильный знак результата: 0FFFFh при переполнении и 0000h при потере значащих разрядов. Программное обеспечение пользователя должно определить и соответствующим образом обработать эти состояния.

7.2.3. Примеры программного обеспечения

Ниже приведены примеры для всех режимов умножителя. Все режимы 8×8 используют абсолютные адреса для регистров, поскольку ассемблер не позволит обеспечить доступ типа .B к регистрам-словам, когда используются метки из стандартного файла определений.

```
; 16x16 умножение без знака
MOV #01234h,&MPY        ;Загрузка первого операнда
MOV #05678h,&OP2        ;Загрузка второго операнда
;...                    ;Обработка результатов
;8x8 умножение без знака. Абсолютная адресация.
MOV.B #012h,&0130h      ;Загрузка первого операнда
MOV.B #034h,&0138h      ;Загрузка второго операнда
;...                    ;Обработка результатов
;16x16 умножение со знаком
MOV #01234h,&MPYS       ;Загрузка первого операнда
MOV #05678h,&OP2        ;Загрузка второго операнда
;...                    ;Обработка результатов
;8x8 умножение со знаком. Абсолютная адресация.
MOV.B #012h,&0132h      ;Загрузка первого операнда
SXT &MPYS               ;Знаковое расширение первого операнда
MOV.B #034h,&0138h      ;Загрузка второго операнда
SXT &OP2                ;Знаковое расширение второго операнда
                        ;(запуск второго умножения)
;...                    ;Обработка результатов
;16x16 умножение без знака с накоплением
MOV #01234h,&MAC        ;Загрузка первого операнда
MOV #05678h,&OP2        ;Загрузка второго операнда
;...                    ;Обработка результатов
;8x8 умножение без знака с накоплением. Абсолютная адресация.
MOV.B #012h,&0134h      ;Загрузка первого операнда
MOV.B #034h,&0138h      ;Загрузка второго операнда
;...                    ;Обработка результатов
;16x16 умножение со знаком и накоплением
MOV #01234h,&MACS       ;Загрузка первого операнда
MOV #05678h,&OP2        ;Загрузка второго операнда
;...                    ;Обработка результатов
;8x8 умножение со знаком и накоплением. Абсолютная адресация
MOV.B #012h,&0136h      ;Загрузка первого операнда
SXT &MACS               ;Знаковое расширение первого операнда
MOV.B #034h,R5          ;Временное расположение второго операнда
SXT R5                  ;Знаковое расширение второго операнда
MOV R5,&OP2             ;Загрузка второго операнда
;...                    ;Обработка результатов
```

7.2.4. Косвенная адресация RESLO

Когда используется косвенный или косвенный автоинкрементный режим адресации для доступа к регистрам результата, нужна, по крайней мере, одна команда между загрузкой второго операнда и доступа к одному из регистров результата:

```
;Доступ к результатам умножителя с косвенной адресацией
MOV #RESLO,R5      ;Загрузка адреса RESLO в R5 для косвенной
                   ;адресации
MOV &OPER1,&MPY    ;Загрузка первого операнда
MOV &OPER2,&OP2    ;Загрузка второго операнда
NOP                ;Необходим один цикл
MOV @R5+,&xxx      ;Пересылка RESLO
MOV @R5,&xxx       ;Пересылка RESHI
```

7.2.5. Использование прерываний

Если прерывание произошло после записи OP1, но до записи OP2, а умножитель используется в процедуре обработки прерывания, исходный выбранный режим умножителя будет потерян и результат станет непредсказуемым. Чтобы этого избежать, нужно отключать прерывания перед использованием аппаратного умножителя и не использовать его в процедурах обработки прерывания.

```
;Отключение прерываний перед использованием аппаратного умножителя
DINT               ;Запрещение прерываний
NOP                ;Требуется для DINT
MOV #xxh,&MPY      ;Загрузка первого операнда
MOV #xxh,&OP2      ;Загрузка второго операнда
EINT               ;Разрешение прерываний
                   ;Обработка результатов
```

7.3. Регистры аппаратного умножителя

Перечень регистров аппаратного умножителя приведен в таблице 7.4.

Таблица 7-4. Регистры аппаратного умножителя

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Операнд один – умножение	MPY	Чтение/запись	0130h	Неизменное
Операнд один – умножение со знаком	MPYS	Чтение/запись	0132h	Неизменное
Операнд один – умножение с накоплением	MAC	Чтение/запись	0134h	Неизменное
Операнд один – умножение со знаком и накоплением	MACS	Чтение/запись	0136h	Неизменное
Операнд два	OP2	Чтение/запись	0138h	Неизменное
Младшее слово результата	RESLO	Чтение/запись	013Ah	Неопределенное
Старшее слово результата	RESHI	Чтение/запись	013Ch	Неопределенное
Регистр знакового дополнения	SUMEXT	Чтение	013Eh	Неопределенное

MSP430x1xxFamily

Контроллер DMA

Раздел VIII.

Texas Instruments

Контроллер DMA

Модуль контроллера DMA переносит данные из одного адреса в другой без участия ЦПУ. Этот раздел описывает работу контроллера DMA. Контроллер DMA реализован в устройствах MSP430x15x и MSP430x16x.

8.1. Введение в контроллер DMA

Контроллер прямого доступа к памяти (DMA) переносит данные из одного адреса в другой во всем адресном диапазоне без вмешательства ЦПУ. К примеру, контроллер DMA может переместить данные из памяти преобразования АЦП12 в ОЗУ.

Использование контроллера DMA может увеличить пропускную способность периферийных модулей. Также в результате его использования снижается потребляемой системой мощности, поскольку ЦПУ может оставаться в режиме пониженного энергопотребления без пробуждения при перемещении данных в/из периферии.

Контроллер DMA обладает следующими возможностями:

- Три независимых канала переноса;
- Конфигурируемые приоритеты канала DMA;
- Необходимо только два тактовых цикла MCLK;
- Возможен перенос байтов, слов или смешанно байтов/слов;
- Размер блока до 65535 байт или слов;
- Набор конфигурируемых источников запуска переноса;
- Возможность выбора условия запуска переноса по фронту/спаду или по уровню;
- Четыре режима адресации;
- Одиночный, блочный или пакетно-блочный режимы переноса.

Блок-схема контроллера DMA показана на рис. 8.1.

8.2. Функционирование DMA

Контроллер DMA конфигурируется программным обеспечением пользователя. В следующих далее разделах описывается инициализация и функционирование DMA.

8.2.1. Режимы адресации DMA

Контроллер DMA имеет четыре режима адресации. Режимы адресации каждого канала DMA конфигурируются независимо друг от друга. Например, канал 0 может выполнять перенос между двумя фиксированными адресами, в то время как в канале 1 выполняются переносы между двумя блоками адре-

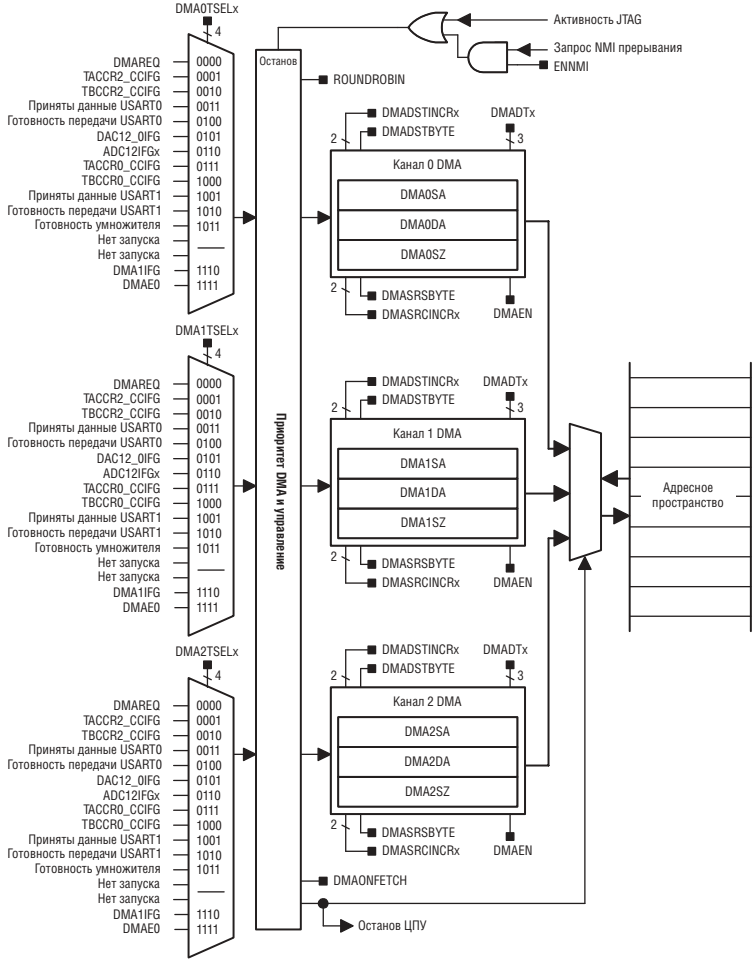


Рис. 8-1. Блок-схема контроллера DMA

сов. Режимы адресации показаны на рис. 8.2. Существуют следующие режимы адресации:

- Фиксированный адрес к фиксированному адресу;
- Фиксированный адрес к блоку адресов;
- Блок адресов к фиксированному адресу;
- Блок адресов к блоку адресов.

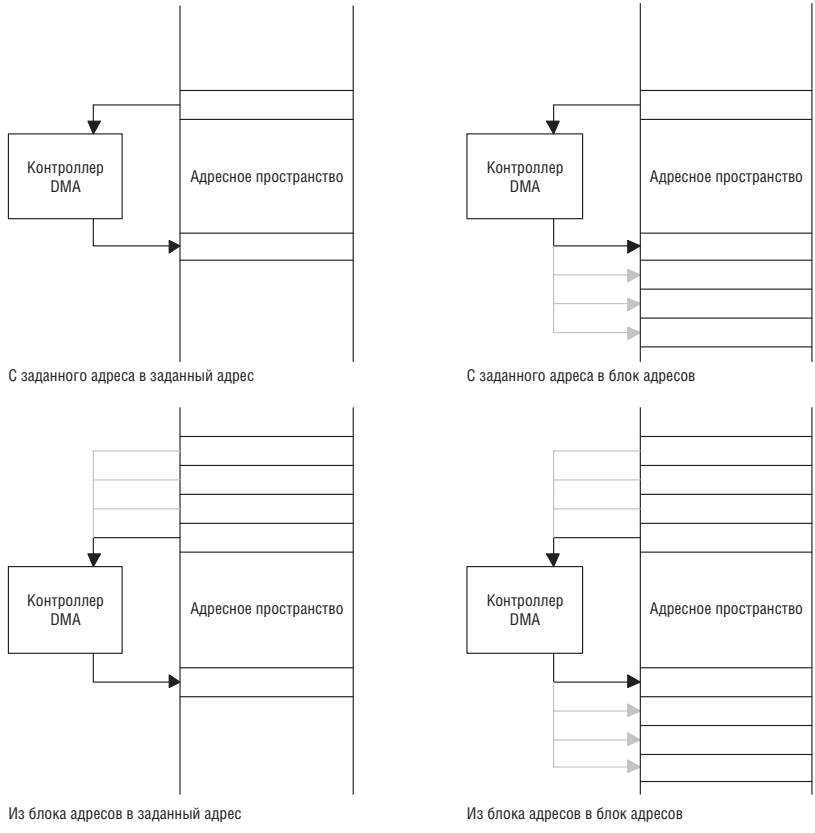


Рис. 8-2. Режимы адресации DMA

Режимы адресации конфигурируются с помощью управляющих битов DMASRCINCRx и DMADSTINCRx. Биты DMASRCINCRx выбираются, если адрес источника инкрементируется, декрементируется или не изменяется после каждого переноса. Биты DMADSTINCRx выбираются, если адрес назначения инкрементируется, декрементируется или не изменяется после каждого переноса.

Переносы могут быть такими: байт-байт, слово-слово, байт-слово или слово-байт. Когда выполняется перенос слово-байт, переносится только младший байт слова-источника. Когда выполняется перенос байт-слово, старший байт слова-получателя очищается, когда происходит перенос.

8.2.2. Режимы переноса DMA

Контроллер DMA имеет шесть режимов переноса, определяемых битами DMADTx, в соответствии с таблицей 8-1. В каждом канале режим переноса конфигурируется индивидуально. К примеру, канал 0 может быть сконфигурирован в одиночном режиме переноса, канал 1 в режиме пакетно-блочного переноса, а канал 2 работать в повторяющемся блочном режиме. Режим переноса конфигурируется независимо от режима адресации. Любой режим адресации можно использовать в любом режиме переноса.

Таблица 8-1. Режимы переноса DMA

DMADTx	Режим переноса	Описание
000	Одиночныйперенос	Каждый перенос нуждается в запуске. DMAEN автоматически очищается после выполнения количества переносов, определенного в DMAxSZ.
001	Блочныйперенос	Блок переносится полностью после одного запуска. DMAEN автоматически очищается после завершения переноса блока.
010, 011	Пакетно-блочный перенос	ЦПУ активен в промежутках между переносами блоков. DMAEN автоматически очищается после завершения пакетно-блочного переноса.
100	Повторяющийся одиночный-перенос	Каждый перенос нуждается в запуске. DMAEN остается установленным.
101	Повторяющийся блочный перенос	Блок переносится полностью после одного запуска. DMAEN остается установленным.
110, 111	Повторяющийся пакетно-блочный перенос	ЦПУ активен в промежутках между переносами блоков. DMAEN остается установленным.

Одиночный перенос

В одиночном режиме переноса пересылка каждого байта/слова требует отдельного запуска. Диаграмма состояний при одиночном переносе показана на рис. 8-3.

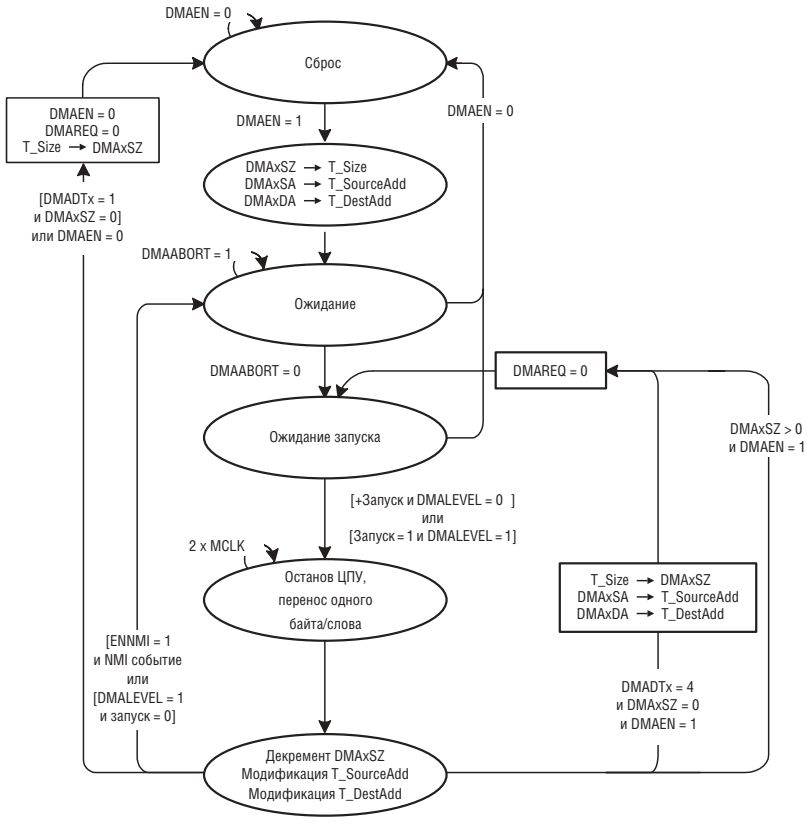


Рис. 8-3. Диаграмма состояний одиночного переноса DMA

Регистр DMAxSZ используется для задания числа переносов, которые нужно выполнить. Биты DMADSTINCRx и DMASRCINCRx выбираются, если адрес получателя и адрес источника инкрементируются или декрементируются после каждого переноса. Если DMAxSZ=0, переносы не выполняются.

Регистры DMAxSA, DMAxDA и DMAxSZ копируются во временные регистры. Временные значения DMAxSA и DMAxDA инкрементируются или декрементируются после каждого переноса. Регистр DMAxSZ декрементируется после каждого переноса. Когда регистр DMAxSZ декрементируется до нуля, он перезагружается из временного регистра и происходит установка соответствую-

щего флага DMAIFG. Когда DMADTx=0, бит DMAEN автоматически очищается, когда DMAxSZ декрементируется до нуля и он должен быть снова установлен для выполнения другого переноса.

В повторяющемся одиночном режиме переноса контроллер DMA остается включенным с DMAEN=1, и переносы выполняются каждый раз при появлении условия запуска.

Блочные переносы

В блочном режиме перенос полного блока данных выполняется после всего лишь одного запуска. Когда DMADTx=1, бит DMAEN очищается после завершения переноса блока и должен быть установлен снова перед запуском переноса другого блока. После запуска переноса блока все последующие сигналы запуска, поступающие в ходе выполнения блочного переноса, игнорируются. Диаграмма состояний блочного переноса показана на рис. 8-4.

Регистр DMAxSZ используется для задания размера блока, а биты DMADSTINCRx и DMASRCINCRx выбираются, если адрес получателя и адрес источника инкрементируется или декрементируется после каждого переноса блока. Если DMAxSZ=0, переносы не выполняются.

Регистры DMAxSA, DMAxDA и DMAxSZ копируются во временные регистры. Временные значения DMAxSA и DMAxDA инкрементируются или декрементируются после каждого переноса в блоке. Регистр DMAxSZ декрементируется после каждого переноса блока и содержит количество оставшихся в блоке переносов. Когда регистр DMAxSZ декрементируется до нуля, он перезагружается из временного регистра, и устанавливается соответствующий флаг DMAIFG.

В процессе переноса блока ЦПУ приостанавливается до завершения переноса блока. Для выполнения поблочного переноса необходимо 2×MCLK×DMA×SZ тактовых циклов. После завершения переноса блока ЦПУ возобновляет работу с предыдущего состояния.

В режиме повторяющегося блочного переноса бит DMAEN остается установленным после завершения переноса блока. Следующий после завершенного повторяющегося блочного переноса сигнал запуска запустит другой блочный перенос.

Пакетно-блочные переносы

В пакетно-блочном режиме переносы блоков, чередуются с работой ЦПУ. ЦПУ выполняет 2 MCLK цикла после переноса каждых четырех байт/слов блока. В результате производительность ЦПУ составляет 20% от номинальной. После завершения пакетно-блочного переноса ЦПУ вновь начинает работать со 100% производительностью, а бит DMAEN очищается. DMAEN должен быть установ-

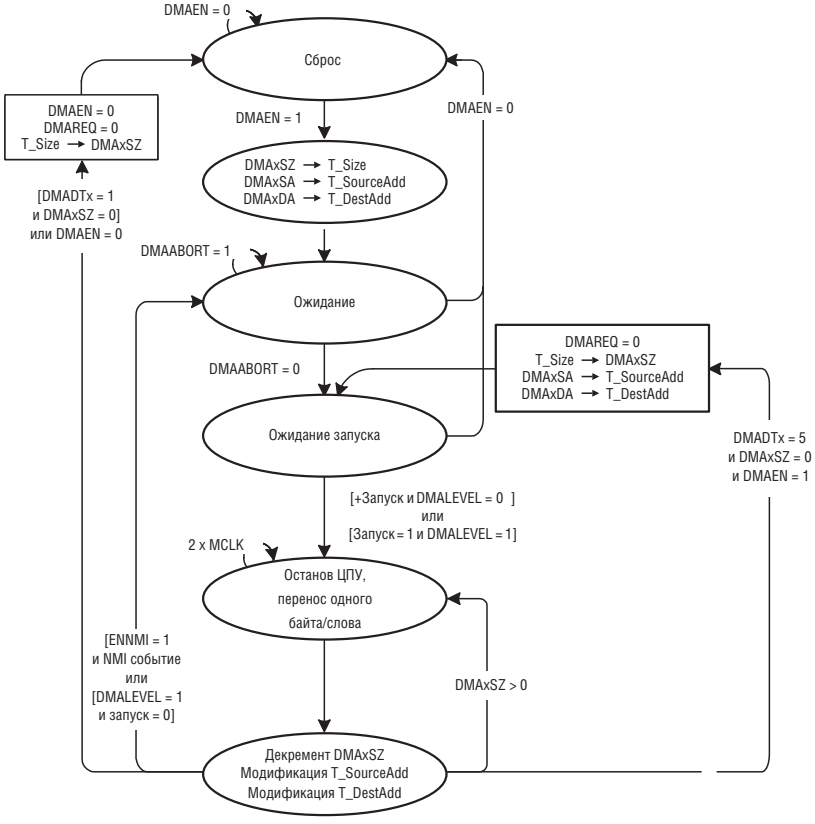


Рис. 8-4. Диаграмма состояний блочного переноса DMA

лен снова перед запуском другого пакетно-блочного переноса. После запуска пакетно-блочного переноса последующие сигналы запуска, появляющиеся во время выполнения пакетно-блочного переноса, игнорируются. Диаграмма состояний пакетно-блочного переноса показана на рис. 8-5.

Регистр DMAxSZ используется для задания размера блока, а биты DMADSTINCRx и DMASRCINCRx выбираются, если адрес получателя и адрес источника инкрементируется или декрементируется после каждого переноса блока. Если DMAxSZ=0, переносы не выполняются.

Регистры DMAxSZ, DMAxDA и DMAxSZ копируются во временные регистры. Временные значения DMAxSA и DMAxDA инкрементируются или декремен-

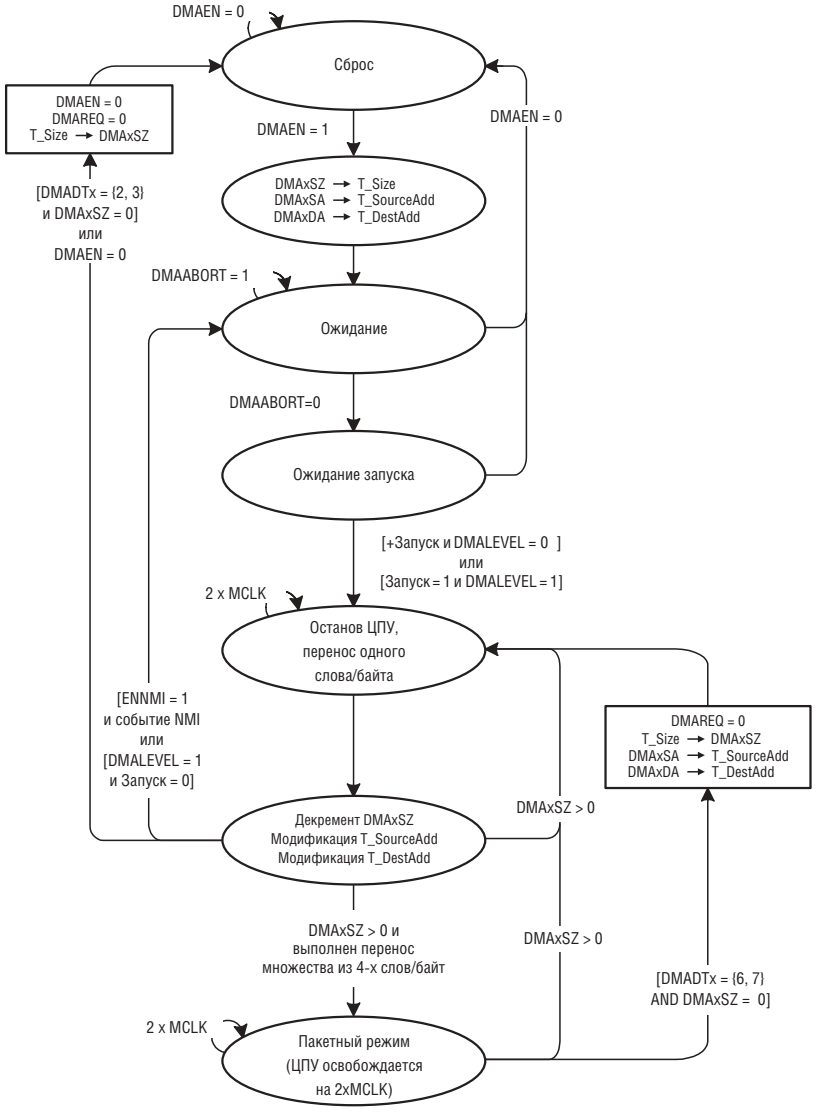


Рис. 8-5. Диаграмма состояний пакетно-блочного переноса DMA

тируются после каждого переноса в блоке. Регистр DMAxSZ декрементируется после каждого переноса блока и содержит количество оставшихся в блоке переносов. Когда DMAxSZ декрементируется до нуля, он перезагружается из временного регистра и устанавливается соответствующий флаг DMAIFG.

В повторяющемся пакетно-блочном режиме бит DMAEN остается установленным после завершения пакетно-блочного переноса и последующие сигналы запуска для инициирования пакетно-блочного переноса не нужны. Другой пакетно-блочный перенос начинается немедленно после завершенного пакетно-блочного переноса. В этом случае, переносы могут быть остановлены очисткой бит DMAEN или по NMI-прерыванию, если установлен ENNMI. В режиме повторяющегося пакетно-блочного переноса производительность будет ЦПУ составлять 20% от номинальной до момента остановки пакетно-блочного переноса.

8.2.3 Инициирование DMA-переносов

Источники запуска переноса в каждом канале DMA конфигурируются независимо с помощью битов DMAxTSELx в соответствии с таблицей 8-2. Биты DMAxTSELx должны модифицироваться только тогда, когда бит DMAEN DMACTLx равен 0. В противном случае могут произойти непредсказуемые запуски DMA.

Когда выбирается условие запуска, оно не должно быть уже выполнено, поскольку в этом случае запуск не произойдет. К примеру, если бит TACCR2 CCIFG выбран как источник запуска, и он уже был установлен, перенос не будет выполнен до момента новой установки бита TACCR2 CCIFG.

Запуски по фронту

Когда DMALEVEL=0, используется условие запуска по фронту, когда фронт сигнала запуска инициирует перенос. В режиме одиночного переноса для выполнения каждого переноса необходим собственный сигнал запуска. Когда используется блочный или пакетно-блочный режим, для инициирования блочного или пакетно-блочного переноса необходим только один сигнал запуска.

Запуски по уровню

Когда DMALEVEL=1, используется условие запуска по уровню. Для правильной работы механизм запуска по уровню может использоваться только тогда, когда в качестве условия запуска выбран внешний триггер DMAE0. DMA переносы запускаются в течение всего времени, пока сигнал запуска имеет высокий уровень и бит DMAEN остается установленным.

Для завершения блочного или пакетно-блочного переноса сигнал запуска должен оставаться в состоянии высокого уровня. Если во время выполнения

блочного или пакетно-блочного переноса сигнал запуска переходит в состояние низкого уровня, контроллер DMA удерживается в текущем состоянии до тех пор, пока сигнал запуска вновь не примет высокий уровень или пока регистры DMA не будут модифицированы программным обеспечением. Если регистры DMA программно не модифицировались, когда сигнал запуска снова перешел в состояние высокого уровня, перенос возобновляется с того состояния, в котором он оставался, когда сигнал запуска стал низкого уровня.

Когда DMALEVEL=1, рекомендуется выбирать режимы переноса при DMADTx={0, 1, 2, 3}, поскольку бит DMAEN автоматически сбрасывается после конфигурирования переноса.

Останов выполнения команд для DMA переносов

Бит DMAONFETCH определяет, когда ЦПУ остановится для выполнения DMA переноса. Когда DMAONFETCH=0, ЦПУ останавливается немедленно и перенос начинается, когда получен сигнал запуска. Когда DMAONFETCH=1, ЦПУ заканчивает выполнение текущей команды, затем контроллер DMA останавливает ЦПУ и начинает перенос.

Примечание: DMAONFETCH обязательно должен использоваться, когда DMA записывает в флэш.

Если контроллер DMA используется для записи во флэш-память, бит DMAONFETCH должен быть установлен. В противном случае результат работы может быть непредсказуем.

Таблица 8-2. Источники запуска DMA

DMAxTSELx	Действие
0000	Установка бита DMAREQ запускает DMA-перенос. Бит DMAREQ автоматически сбрасывается, когда начинается перенос.
0001	Перенос запускается, когда устанавливается флаг TACCR2 CCIFG. Флаг TACCR2 CCIFG автоматически сбрасывается, когда начинается перенос. Если бит TACCR2 CCIE установлен, флаг TACCR2 CCIFG не запустит перенос.
0010	Перенос запускается, когда устанавливается флаг TBCCR2 CCIFG. Флаг TBCCR2 CCIFG автоматически сбрасывается, когда начинается перенос. Если бит TBCCR2 CCIE установлен, флаг TBCCR2 CCIFG не запустит перенос.
0011	Перенос запускается, когда модуль USART0 принимает новые данные. В режиме I2C запуск происходит при условии приема данных, но не при установке флага RXRDYIFG. RXRDYIFG не очищается, когда начинается перенос, а программная установка RXRDYIFG не приводит к запуску переноса. Если RXRDYIE установлен, условие приема данных не вызовет запуска переноса. В UART или SPI режимах перенос запускается, когда установлен флаг URXIFG0. URXIFG0 автоматически сбрасывается, когда начинается перенос. Если URXIE0 установлен, флаг URXIFG0 не запускает перенос.

Таблица 8-2. (Окончание)

DMAxTSELx	Действие
0100	Перенос запускается, когда модуль USART0 готов передавать новые данные. В режиме I2C запуск происходит при условии готовности передачи данных, но не при установке флага TXRDYIFG. TXRDYIFG не очищается, когда начинается перенос, а программная установка TXRDYIFG не приводит к запуску переноса. Если TXRDYIE установлен, условие готовности передачи данных не вызовет запуска переноса. В UART или SPI режимах перенос запускается, когда установлен флаг UTXIFG0. UTXIFG0 автоматически сбрасывается, когда начинается перенос. Если UTXIE0 установлен, флаг UTXIFG0 не запускает перенос.
0101	Перенос запускается, когда устанавливается флаг DAC12_0CTL DAC12IFG. Флаг DAC12_0CTL DAC12IFG автоматически очищается, когда начинается перенос. Если DAC12_0CTL DAC12IE установлен, флаг DAC12_0CTL DAC12IFG не запускает перенос.
0110	Перенос запускается флагом ADC12IFGx. Когда выполняется преобразование в одном канале, соответствующий флаг ADC12IFGx запускает перенос. Когда выполняется повторяющаяся последовательность преобразований, перенос запускается флагом ADC12IFGx последнего преобразования в последовательности. Перенос запускается, когда преобразование завершено и ADC12IFGx установлен. Установка ADC12IFGx программным обеспечением перенос не запускает. Все флаги ADC12IFGx сбрасывается автоматически, когда контроллер DMA обращается к соответствующему регистру ADC12MEMx.
0111	Перенос запускается, когда устанавливается флаг TACCR0 CCIFG. Флаг TACCR0 CCIFG автоматически сбрасывается, когда перенос стартует. Если бит TACCR0 CCIE установлен, флаг TACCR0 CCIFG не запускает перенос.
1000	Перенос запускается, когда устанавливается флаг TBCCR0 CCIFG. Флаг TBCCR0 CCIFG автоматически сбрасывается, когда перенос стартует. Если бит TBCCR0 CCIE установлен, флаг TBCCR0 CCIFG не запускает перенос.
1001	Перенос запускается, когда устанавливается флаг URXIFG1. URXIFG1 автоматически сбрасывается, когда перенос стартует. Если URXIE1 установлен, флаг URXIFG1 не запускает перенос.
1010	Перенос запускается, когда устанавливается флаг UTXIFG1. UTXIFG1 автоматически сбрасывается, когда перенос стартует. Если UTXIE1 установлен, флаг UTXIFG1 не запускает перенос.
1011	Перенос запускается, когда аппаратный умножитель готов для нового операнда.
1100	Перенос не запускается.
1101	Перенос не запускается.
1110	Перенос запускается, когда устанавливается флаг DMAxIFG. DMA0IFG запускает канал 1, DMA1IFG запускает канал 2, а DMA2IFG запускает канал 0. Ни один из флагов DMAxIFG автоматически не сбрасывается, когда перенос стартует.
1111	Перенос запускается внешним сигналом от DMAE0.

8.2.4. Останов DMA-переносов

Есть два способа остановить выполняющийся DMA-перенос:

- Одиночный, блочный или пакетно-блочный перенос может быть остановлен NMI-прерыванием, если установлен бит ENNMI в регистре DMACTL1.
- Пакетно-блочный перенос может быть остановлен очисткой бита DMAEN.

8.2.5. Приоритеты каналов DMA

По умолчанию приоритеты DMA-каналов такие: DMA0-DMA1-DMA2. Если одновременно появляются или находятся в ожидании два или три сигнала запуска, первым завершает перенос (одиночный, блочный или пакетно-блочный перенос) канал с наивысшим приоритетом, затем канал со вторым приоритетом и в завершение канал с третьим приоритетом. Выполняющиеся переносы не приостанавливаются, если запускается перенос в канале с более высоким приоритетом. Канал с высшим приоритетом ожидает завершения выполняющегося переноса, и только затем стартует.

Приоритеты DMA-каналов конфигурируются с помощью бита ROUNDROBIN. Когда бит ROUNDROBIN установлен, низший приоритет получает канал, завершивший перенос. Последовательность приоритетов каналов всегда остается подобной DMA0-DMA1-DMA2, например:

Приоритет DMA	Завершенный перенос	Новый приоритет DMA
DMA0 – DMA1 – DMA2	DMA1	DMA2 – DMA0 – DMA1
DMA2 – DMA0 – DMA1	DMA2	DMA0 – DMA1 – DMA2
DMA0 – DMA1 – DMA2	DMA0	DMA1 – DMA2 – DMA0

Когда бит ROUNDROBIN очищен, приоритеты каналов возвращается к приоритетам по умолчанию.

8.2.6. Длительность цикла DMA-переноса

Контроллер DMA нуждается в одном или двух тактовых циклах MCLK для синхронизации перед каждым одиночным переносом или полным блочным или пакетно-блочным переносом. Для переноса каждого байта/слова нужно два цикла MCLK после синхронизации и один цикл времени ожидания после переноса. Поскольку контроллер DMA использует MCLK, продолжительность цикла DMA определяется режимом работы MSP430 и установками системы тактирования.

Если источник MCLK активен, но ЦПУ выключено, контроллер DMA будет использовать источник MCLK для каждого переноса без включения ЦПУ. Если источник MCLK выключен, контроллер DMA временно перезапустит MCLK с

тактированием от DCOCLK для выполнения одиночного переноса или полного блочного или пакетно-блочного переноса. ЦПУ остается выключенным, а после завершения переноса выключается MCLK. Максимальная длительность цикла DMA для всех режимов работы показана в таблице 8-3.

Таблица 8-3. Максимальная длительность цикла одиночного DMA-переноса

Режим работы ЦПУ	Источник тактирования	Максимальная продолжительность цикла DMA
Активный режим	MCLK=DCOCLK	4 цикла MCLK
Активный режим	MCLK=LFXT1CLK	4 цикла MCLK
Режим пониженного потребления LPM0/1	MCLK=DCOCLK	5 циклов MCLK
Режим пониженного потребления LPM3/4	MCLK=DCOCLK	5 циклов MCLK + 6 мкС*
Режим пониженного потребления LPM0/1	MCLK=LFXT1CLK	5 циклов MCLK
Режим пониженного потребления LPM3	MCLK=LFXT1CLK	5 циклов MCLK
Режим пониженного потребления LPM4	MCLK=LFXT1CLK	5 циклов MCLK + 6 мкС*

* Дополнительные 6 мкС необходимы для запуска DCOCLK. Этот параметр в справочном руководстве называется t(LPMx).

8.2.7. Использование DMA с системными прерываниями

DMA переносы не прерываются системными прерываниями. Системные прерывания ожидают завершения переноса. Немаскируемые NMI-прерывания могут прервать работу DMA-контроллера, если установлен бит ENNMI.

Процедуры обработки системного прерывания прерываются DMA переносами. Если процедура обработки прерывания или какая-либо другая подпрограмма должны выполняться без прерываний, контроллер DMA необходимо отключить перед выполнением такой подпрограммы.

8.2.8. Прерывания контроллера DMA

Каждый канал DMA имеет собственный флаг DMAIFG. Каждый флаг DMAIFG устанавливается в любом режиме, когда соответствующий регистр DMAxSZ досчитывает до нуля. Если соответствующие биты DMAIE и GIE установлены, генерируется запрос прерывания.

Все флаги DMAIFG – источники только одного вектора прерывания контроллера DMA, а вектор прерывания общий с модулем DAC12. Программное обеспечение должно проверить флаги DMAIFG и DAC12IFG, чтобы определить источник прерывания. Флаги DMAIFG автоматически не сбрасываются и должны быть сброшены программно.

8.2.9. Использование модуля I²C с контроллером DMA

Модуль I²C может стать источником двух условий запуска для контроллера DMA. Модуль I²C может запустить перенос, когда приняты новые данные I²C и когда появилась необходимость в передаче данных.

Биты TXDMAEN и RXDMAEN разрешают или запрещают использование контроллера DMA с модулем I²C. Когда RXDMAEN=1, контроллер DMA может быть использован для переноса данных из модуля I²C после приема данных модулем I²C. Когда RXDMAEN=1, RXRDYIE игнорируется и RXRDYIFG не генерирует прерывание.

Когда TXDMAEN=1, контроллер DMA может быть использован для переноса данных в модуль I²C для передачи. Когда TXDMAEN=1, TXRDYIE игнорируется и TXRDYIFG не генерирует прерывание.

8.2.10. Использование АЦП12 с контроллером DMA

Устройства MSP430 с интегрированным контроллером DMA могут автоматически перемещать данные из любого регистра ADC12MEMx в другое место. Переносы DMA выполняются без вмешательства ЦПУ и независимо от любого режима пониженного энергопотребления. Контроллер DMA увеличивает пропускную способность модуля АЦП12 и расширяет возможные сферы применения MSP430 в малопотребляющих приложениях, позволяя ЦПУ оставаться выключенным при выполнении переноса данных.

DMA переносы могут быть запущены от любого флага ADC12IFGx. Когда CONSEQx={0,2}, флаг ADC12IFGx для ADC12MEMx, используемого при преобразовании, может запустить DMA перенос. Когда CONSEQx={1, 3}, флаг ADC12IFGx для последнего в последовательности ADC12MEMx может запустить DMA перенос. Любой флаг ADC12IFGx автоматически очищается, когда контроллер DMA обращается к соответствующему регистру ADC12MEMx.

8.2.11. Использование ЦАП12 с контроллером DMA

Устройства MSP430 с интегрированным контроллером DMA могут автоматически перемещать данные в регистр DAC12_xDAT. Переносы DMA выполняются без вмешательства ЦПУ и независимо от любого режима пониженного энергопотребления. Контроллер DMA увеличивает пропускную способность модуля ЦАП12 и расширяет возможные сферы применения MSP430 в малопотребляющих приложениях, позволяя ЦПУ оставаться выключенным при выполнении переноса данных.

Приложения, в которых требуется генерировать периодические колебания, могут получить существенную выгоду от использования контроллера DMA с ЦАП12. Например, приложение, создающее синусоидальное колебание, может хранить значения синусоиды в таблице. Контроллер DMA способен авто-

матически непрерывно переносить эти значения в ЦАП12 через заданные интервалы времени, создавая синусоиду при остановленном ЦПУ. Флаг DAC12IFG DAC12_xCTL автоматически очищается, когда контроллер DMA обращается к регистру DAC12_xDAT.

8.3. Регистры DMA

Перечень регистров DMA приведен в таблице 8-4.

Таблица 8-4. Регистры DMA

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр 0 управления DMA	DMACTL0	Чтение/запись	0122h	Сбрасывается с POR
Регистр 1 управления DMA	DMACTL1	Чтение/запись	0124h	Сбрасывается с POR
Регистр управления канала 0 DMA	DMA0CTL	Чтение/запись	01E0h	Сбрасывается с POR
Регистр адреса источника канала 0 DMA	DMA0SA	Чтение/запись	01E2h	Не изменяется
Регистр адреса получателя канала 0 DMA	DMA0DA	Чтение/запись	01E4h	Не изменяется
Регистр объема переноса канала 0 DMA	DMA0SZ	Чтение/запись	01E6h	Не изменяется
Регистр управления канала 1 DMA	DMA1CTL	Чтение/запись	01E8h	Сбрасывается с POR
Регистр адреса источника канала 1 DMA	DMA1SA	Чтение/запись	01EAh	Не изменяется
Регистр адреса получателя канала 1 DMA	DMA1DA	Чтение/запись	01ECh	Не изменяется
Регистр объема переноса канала 1 DMA	DMA1SZ	Чтение/запись	01EEh	Не изменяется
Регистр управления канала 2 DMA	DMA2CTL	Чтение/запись	01F0h	Сбрасывается с POR
Регистр адреса источника канала 2 DMA	DMA2SA	Чтение/запись	01F2h	Не изменяется
Регистр адреса получателя канала 2 DMA	DMA2DA	Чтение/запись	01F4h	Не изменяется
Регистр объема переноса канала 2 DMA	DMA2SZ	Чтение/запись	01F6h	Не изменяется

DMACTL0, регистр 0 управления DMA

15	14	13	12	11	10	9	8
Зарезервировано				**DMA2TSELx**			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
DMA1TSELx				**DMA0TSELx**			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Зарезервировано	Биты 15-12	Зарезервировано
DMA2TSELx	Биты 11-8	Выбор источника сигнала запуска DMA. Эти биты определяют источник сигнала запуска DMA-переноса. 0000 – Бит DMAREQ (программный запуск) 0001 – Бит TACCR2 CCIFG 0010 – Бит TBCCR2 CCIFG 0011 – URXIFG0 (режим UART/SPI), данные приняты USART0 (режим I^2C) 0100 – UTXIFG0 (режим UART/SPI), готовность передачи USART0 (режим I^2C) 0101 – Бит DAC12IFG DAC12_0CTL 0110 – Бит ADC12IFGx ADC12 0111 – Бит TACCR0 CCIFG 1000 – Бит TBCCR2 CCIFG 1001 – Бит URXIFG 11010 – Бит UTXIFG 11011 – Готовность умножителя 1100 – Действие не производится 1101 – Действие не производится 1110 – Бит DMA0IFG запускает канал 1 DMA Бит DMA1IFG запускает канал 2 DMA Бит DMA2IFG запускает канал 0 DMA 1111 – Внешний запуск DMAE0
DMA1TSELx	Биты 7-4	Подобно DMA2TSELx
DMA0TSELx	Биты 3-0	Подобно DMA2TSELx

DMACTL1, регистр 1 управления DMA

15	14	13	12	11	10	9	8
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r0	r0	r0	r0	r0	r0	r0	r0

7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**DMA ONFETCH**	**ROUND ROBIN**	**ENNMI**
r0	r0	r0	r0	r0	rw-(0)	rw-(0)	rw-(0)

Зарезервировано	Биты 15-3	Зарезервировано. Только чтение. Всегда читаются как 0.
DMAONFETCH	Бит 2	Выборка DMA 0 – DMA перенос происходит немедленно 1 – DMA перенос происходит при выборке следующей команды после запуска

ROUNDROBIN	Бит 1	Этот бит разрешает циклическое движение приоритетов каналов DMA. 0 – Устанавливается следующий приоритет DMA каналов: DMA0-DMA1-DMA2 1 – Приоритет DMA каналов изменяется с каждым переносом
ENNMI	Бит 0	Разрешение NMI. Этот бит разрешает прерывание DMA переноса немаскируемым прерыванием NMI. Когда NMI прерывает DMA перенос, текущий перенос завершается нормально, но последующие переносы прекращаются и устанавливается флаг DMAABORT. 0 – NMI прерывание не прерывает DMA перенос. 1 – NMI прерывание прерывает DMA перенос.

DMAxCTL, DMA регистр управления каналом x

15	14	13	12	11	10	9	8
Зарезер-вировано	**DMADTx**			**DMADSTINCRx**		**DMASRCINCRx**	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
DMA DSTBYTE	**DMA SRCBYTE**	**DMA LEVEL**	**DMAEN**	**DMAIFG**	**DMAIE**	**DMA ABORT**	**DMAREQ**
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Зарезервировано	Бит 15	Зарезервирован
DMADTx	Биты 14-12	Режим переноса DMA. 000 – Одиночный перенос 001 – Блочный перенос 010 – Пакетно-блочный перенос 011 – Пакетно-блочный перенос 100 – Повторный одиночный перенос 101 – Повторный блочный перенос 110 – Повторный пакетно-блочный перенос 111 – Повторный пакетно-блочный перенос
DMADSTINCRx	Биты 11-10	Инкремент DMA получателя. Этот бит позволяет выбрать автоматическое инкрементирование или декрементирование адреса получателя после переноса каждого байта или слова. Когда DMADSTBYTE=1, адрес получателя инкрементируется/декрементируется на единицу. Когда DMADSTBYTE=0, адрес получателя инкрементируется/декрементируется на 2. DMAxDA копируется во временный регистр и уже временный регистр инкрементируется или декрементируется. DMAxDA не инкрементируется и не декрементируется. 00 – Адрес получателя не изменяется 01 – Адрес получателя не изменяется 10 – Адрес получателя декрементируется 11 – Адрес получателя инкрементируется

DMASRCINCRx	Биты 9-8	Инкремент DMA источника. Этот бит позволяет выбрать автоматическое инкрементирование или декрементирование адреса источника после переноса каждого байта или слова. Когда DMASRCBYTE=1, адрес источника инкрементируется/декрементируется на единицу. Когда DMASRCBYTE=0, адрес источника инкрементируется/декрементируется на 2. DMAxSA копируется во временный регистр и уже временный регистр инкрементируется или декрементируется. DMAxSA не инкрементируется и не декрементируется. 00 – Адрес источника не изменяется 01 – Адрес источника не изменяется 10 – Адрес источника декрементируется 11 – Адрес источника инкрементируется
DMADSTBYTE	Бит 7	Байт DMA получателя. Этот бит определяет формат получателя: байт или слово. 0 – Слово. 1 – Байт.
DMASRCBYTE	Бит 6	Байт DMA источника. Этот бит определяет формат источник: байт или слово. 0 – Слово. 1 – Байт.
DMALEVEL	Бит 5	Уровень DMA. Этот бит позволяет выбрать условие запуска переноса: по перепаду или по уровню. 0 – Чувствительность к перепаду (фронт сигнала) 1 – Чувствительность к уровню (высокий уровень)
DMAEN	Бит 4	Разрешение DMA 0 – Запрещено 1 – Разрешено
DMAIFG	Бит 3	Флаг DMA прерывания 0 – Прерывание не ожидается 1 – Ожидается прерывание
DMAIE	Бит 2	Разрешение DMA прерывания 0 – Запрещено 1 – Разрешено
DMAABORT	Бит 1	Прекращение DMA переносов. Этот бит показывает, что DMA перенос был прерван NMI прерыванием. 0 – DMA перенос не прерывался 1 – DMA перенос был прерван NMI прерыванием
DMAREQ	Бит 0	Запрос DMA. Программно управляемый старт DMA. Бит DMAREQ сбрасывается автоматически. 0 – Нет DMA старта 1 – Старт DMA

DMAxSA, регистр адреса источника DMA

15	14	13	12	11	10	9	8
DMAxSAx							
rw	rw	rw	rw	rw	rw	rw	rw

7	6	5	4	3	2	1	0
DMAxSAx							
rw	rw	rw	rw	rw	rw	rw	rw

DMAxSAx	Биты 15-0	Адрес DMA источника. Регистр адреса источника указывает адрес источника DMA для одиночных переносов или первый адрес источника для блочных переносов. Регистр адреса источника остается неизменным во время блочных или пакетно-блочных переносов.

DMAxDA, регистр адреса получателя DMA

15	14	13	12	11	10	9	8
DMAxDAx							
rw	rw	rw	rw	rw	rw	rw	rw

7	6	5	4	3	2	1	0
DMAxDAx							
rw	rw	rw	rw	rw	rw	rw	rw

DMAxDAx	Биты 15-0	Адрес DMA получателя. Регистр адреса получателя указывает адрес получателя для одиночных переносов или первый адрес получателя для блочных переносов. Регистр DMAxDA остается неизменным во время блочных или пакетно-блочных переносов.

DMAxSZ, адресный регистр размера DMA

15	14	13	12	11	10	9	8
DMAxSZx							
rw	rw	rw	rw	rw	rw	rw	rw

7	6	5	4	3	2	1	0
DMAxSZx							
rw	rw	rw	rw	rw	rw	rw	rw

DMAxSZx	Биты 15-0	Объем (размер) DMA. Регистр объема DMA определяет количество байт/слов данных при переносе блока. Регистр DMAxSZ декрементируется при каждом переносе слова или байта. Когда DMAxSZ декрементируется до нуля, в него немедленно автоматически перезагружается предыдущее значение инициализации. 00000h – Перенос запрещен 00001h – Перенос одного байта или слова 00002h – Перенос двух байт или слов . . 0FFFFh – Перенос 65535 байт или слов

MSP430x1xxFamily

Цифровые входы/выходы

Раздел IX.

Texas Instruments

Цифровые входы/выходы

В этом разделе описывается работа портов цифровых входов/выходов. Порты P1-P2 имеются в устройствах MSP430x11xx. Порты P1-P3 реализованы в устройствах MSP430x12xx. Порты P1-P6 реализованы в устройствах MSP430x14, MSP430x15x и MSP430x16x.

9.1. Введение в цифровые входы/выходы

Устройства MSP430 имеют до 6 портов цифровых входов/выходов от P1 до P6. Каждый порт имеет 8 выводов входа/выхода. Каждый вывод индивидуально конфигурируется как вход или выход и каждая линия ввода/вывода может быть индивидуально считана или записана.

Порты P1 и P2 имеют возможность вызывать прерывание. Для каждой линии ввода/вывода портов P1 и P2 можно индивидуально разрешить прерывания и сконфигурировать их так, чтобы прерывание происходило по фронту или спаду входного сигнала. Все линии ввода/вывода порта P1 являются источником одного вектора прерывания, а все линии ввода/вывода порта P2 – источник другого вектора прерывания.

Цифровые входы/выходы обладают следующими возможностями:

- Независимые индивидуально программируемые входы/выходы;
- Любые комбинации входа или выхода;
- Индивидуально конфигурируемые прерывания от P1 и P2;
- Раздельные регистры данных для входов и выходов.

9.2. Функционирование цифровых входов/выходов

Цифровые входы/выходы конфигурируются программным обеспечением пользователя. Настройка и работа цифровых входов/выходов описывается в нижеследующих разделах.

9.2.1. Регистры ввода PxIN

Каждый бит в каждом регистре PxIN отражает величину входного сигнала на соответствующей ножке ввода/вывода, когда она сконфигурирована на функцию ввода/вывода.

Бит = 0: Входной сигнал имеет низкий уровень;

Бит = 1: Входной сигнал имеет высокий уровень.

Примечание: Запись в регистры «только для чтения» PxIN

Запись в эти регистры «только для чтения» приводит к увеличению потребления тока на время выполнения попытки записи.

9.2.2. Регистры вывода PxOUT

Каждый бит в каждом регистре PxOUT содержит значение, которое будет выведено на соответствующую ножку ввода/вывода, сконфигурированную на функцию ввода/вывода и имеющую направление на вывод.

Бит = 0: Выходной сигнал имеет низкий уровень;

Бит = 1: Выходной сигнал имеет высокий уровень.

9.2.3. Регистры направления PxDIR

Каждый бит в каждом регистре PxDIR позволяет выбрать направление соответствующей ножки ввода/вывода, независимо от выбранной для этой ножки функции. Биты PxDIR для ножек ввода/вывода, выбранные для других функций модуля должны быть установлены так, как это требуется для другой функции.

Бит = 0: Ножка порта переключается на ввод;

Бит = 1: Ножка порта переключается на вывод.

9.2.4. Регистры выбора функции PxSEL

Ножки порта часто мультиплексированы с другими функциями периферийных модулей. См. справочное руководство по конкретному устройству для выяснения возможных функций вывода. Каждый бит PxSEL определяет, как будет использована ножка – в качестве порта ввода/вывода или в качестве функции периферийного модуля.

Бит = 0: Для ножки выбирается функция ввода/вывода

Бит = 1: Для ножки выбирается функция периферийного модуля

Установка PxSEL=1 автоматически не определяет направление движения информации для ножки. Некоторые функции периферийных модулей требуют конфигурирования битов PxDIR для выбора направления, необходимого для правильной работы этой функции.

```
;Вывод ACLK на P2.0 в устройстве MSP430F11x1
BIS.B #01h,&P2SEL ;Выбор функции ACLK для ножки
BIS.B #01h,&P2DIR ;Установка направления на вывод
                  ;(необходимо)
```

Примечание: Отключение прерываний от P1 и P2 при PxSEL=1

Когда какой-либо бит P1SELx или P2SELx установлен, функция прерывания от соответствующей ножки отключена. Поэтому сигналы на этих ножках не будут генерировать прерывания P1 или P2, независимо от состояния соответствующего бита P1IE или P2IE.

Когда вывод порта работает как вход периферии, входным сигналом периферии является зафиксированное в защелке представление сигнала на выводе устройства. Когда PxSELx=1, внутренний входной сигнал соответствует сигналу на ножке. Однако, если PxSELx=0, на входе периферии сохраняется значе-

ние входного сигнала на выводе устройства, имевшееся перед сбросом бита PxSELx.

9.2.5. Прерывания P1 и P2

Каждая ножка портов P1 и P2 имеет возможность вызова прерывания, конфигурируемую регистрами PxIFG, PxIE и PxIES. Все ножки P1 – источник одного вектора прерывания, а все выводы P2 – источник другого одиночного вектора прерывания. Определить источник прерывания – P1 или P2 можно путем проверки регистра PxIFG.

Регистры флагов прерывания P1IFG, P2IFG

Каждый бит PxIFG – это флаг прерывания соответствующей ножки ввода/вывода, устанавливаемый, когда происходит перепад выбранного входного сигнала на ножке. Все флаги прерывания PxIFG запрашивают прерывание, когда установлен их соответствующий бит PxIE и установлен бит GIE. Каждый флаг PxIFG должен быть сброшен программно. Программное обеспечение также может устанавливать каждый флаг PxIFG, обеспечивая возможность генерации программно-инициированного прерывания.

Бит = 0: Прерывание не ожидается

Бит = 1: Прерывание ожидается

Прерывания вызывают только перепады уровней, а не статические уровни. Если любой флаг PxIFG оказывается установленным во время выполнения процедуры обработки прерывания Px или устанавливается после команды RETI выполняемой процедуры обработки прерывания Px, установка флага PxIFGx генерирует другое прерывание. Таким образом, гарантируется, что каждый перепад уровня будет учтен.

Примечание: Состояние флагов PxIFG при изменении PxOUT или PxDIR

Запись в P1OUT, P1DIR, P2OUT или P2DIR может привести к установке соответствующих флагов P1IFG или P2IFG.

Примечание: Длительность события вызова прерывания на ножке ввода/вывода

Любое событие вызова внешнего прерывания должно иметь длительность, по крайней мере, равную 1,5 MCLK или дольше, чтобы быть гарантировано принятым и вызвать установку соответствующего флага прерывания.

Регистры выбора фронта прерывания P1IES, P2IES

Каждый бит PxIES позволяет выбрать, по какому фронту сигнала будет происходить прерывание для соответствующей ножки ввода/вывода.

Бит = 0: Флаг PxIFG устанавливается при изменении уровня сигнала с низкого на высокий;

Бит = 1: Флаг PxIFG устанавливается при изменении уровня сигнала с высокого на низкий.

Примечание: Запись в PxIESx

Запись в P1IES или P2IES может привести к установке соответствующих флагов прерывания.

PxIESx	PxINx	PxIFGx
0 → 1	0	Может быть установлен
0 → 1	1	Не изменяется
1 → 0	0	Не изменяется
1 → 0	1	Может быть установлен

Разрешение прерываний P1IE, P2IE

Каждый бит PxIE разрешает прерывание от соответствующего флага прерываний регистра PxIFG.

Бит = 0: Прерывание запрещено

Бит = 1: Прерывание разрешено

9.2.6. Конфигурирование неиспользуемых выводов порта

Неиспользуемые ножки ввода/вывода должны быть сконфигурированы на функцию ввода/вывода, в направлении вывода и оставаться неподключенными на печатной плате для уменьшения потребляемой мощности. Значение бита PxOUT может быть любым, поскольку ножка не подключена. См. раздел «Системный сброс, прерывания и режимы работы» для уточнения вопросов подключения неиспользуемых выводов.

9.3. Регистры цифровых входов/выходов

Для конфигурирования P1 и P2 используются семь регистров. Четыре регистра необходимы для конфигурирования портов P3-P6. Регистры цифровых входов/выходов приведены в таблице 9-1.

Таблица 9-1. Регистры цифровых входов-выходов.

Порт	Регистр	Краткое обозначение	Адрес	Тип регистра	Исходное состояние
P1	Ввод	P1IN	020h	Только чтение	–
	Вывод	P1OUT	021h	Чтение/запись	Не изменяется
	Направление	P1DIR	022h	Чтение/запись	Сброс с PUC
	Флаг прерывания	P1IFG	023h	Чтение/запись	Сброс с PUC
	Выбор фронта прерывания	P1IES	024h	Чтение/запись	Не изменяется
	Разрешение прерывания	P1IE	025h	Чтение/запись	Сброс с PUC
	Выбор порта	P1SEL	026h	Чтение/запись	Сброс с PUC

Таблица 9-1. (Окончание)

Порт	Регистр	Краткое обозначение	Адрес	Тип регистра	Исходное состояние
P2	Ввод	P2IN	028h	Только чтение	–
	Вывод	P2OUT	029h	Чтение/запись	Не изменяется
	Направление	P2DIR	02Ah	Чтение/запись	Сброс с PUC
	Флаг прерывания	P2IFG	02Bh	Чтение/запись	Сброс с PUC
	Выбор фронта прерывания	P2IES	02Ch	Чтение/запись	Не изменяется
	Разрешение прерывания	P2IE	02Dh	Чтение/запись	Сброс с PUC
	Выбор порта	P2SEL	02Eh	Чтение/запись	Сброс с PUC
P3	Ввод	P3IN	018h	Только чтение	–
	Вывод	P3OUT	019h	Чтение/запись	Не изменяется
	Направление	P3DIR	01Ah	Чтение/запись	Сброс с PUC
	Выбор порта	P3SEL	01Bh	Чтение/запись	Сброс с PUC
P4	Ввод	P4IN	01Ch	Только чтение	–
	Вывод	P4OUT	01Dh	Чтение/запись	Не изменяется
	Направление	P4DIR	01Eh	Чтение/запись	Сброс с PUC
	Выбор порта	P4SEL	01Fh	Чтение/запись	Сброс с PUC
P5	Ввод	P5IN	030h	Только чтение	–
	Вывод	P5OUT	031h	Чтение/запись	Не изменяется
	Направление	P5DIR	032h	Чтение/запись	Сброс с PUC
	Выбор порта	P5SEL	033h	Чтение/запись	Сброс с PUC
P6	Ввод	P6IN	034h	Только чтение	–
	Вывод	P6OUT	035h	Чтение/запись	Не изменяется
	Направление	P6DIR	036h	Чтение/запись	Сброс с PUC
	Выбор порта	P6SEL	037h	Чтение/запись	Сброс с PUC

MSP430x1xxFamily

Сторожевой таймер

Раздел X.

Texas Instruments

Сторожевой таймер

Сторожевой таймер – это 16-разряный таймер, который можно использовать как в качестве сторожевого, так и в качестве «интервального» таймера. В этом разделе описывается модуль сторожевого таймера. Сторожевой таймер реализован во всех устройствах MSP430x1xx.

10.1. Введение в сторожевой таймер

Первичная функция модуля сторожевого таймера (WDT) – выполнять рестарт управляемой системы при возникновении проблемы с программным обеспечением. Если установленный временной интервал истек, генерируется системный сброс. Если сторожевая функция в приложении не нужна, модуль может быть сконфигурирован как интервальный таймер для генерации прерываний через установленные интервалы времени.

Сторожевой таймер обладает следующими возможностями:

- Восемь программно настраиваемых временных интервалов
- Режим сторожевого таймера
- Режим интервального отсчета
- Доступ к регистру управления WDT защищен паролем
- Управление функцией вывода nonRST/NMI
- Возможность выбора источника тактовых импульсов
- Возможность останова для уменьшения потребляемой мощности

Блок схема модуля WDT показана на рис. 10-1.

Примечание: Работа сторожевого таймера после подачи питания

После сигнала PUC, модуль WDT автоматически конфигурируется в сторожевом режиме с начальным интервалом сброса ~32 мС с использованием DCOCLK. Пользователь должен выполнить необходимую настройку или остановить WDT до истечения начального интервала сброса.

10.2. Функционирование сторожевого таймера

Модуль WDT можно сконфигурировать с помощью регистра WDTCTL как сторожевой либо интервальный таймер. Регистр WDTCTL также содержит управляющие биты для конфигурирования вывода nonRST/NMI. WDTCTL – это 16-разрядный, защищенный паролем регистр чтения/записи. Любое чтение или попытка записи должны использовать команды-слова, а попытка записи должна содержать пароль записи 05Ah в старшем байте. Любая запись в WDTCTL любого значения, отличного от 05Ah в старшем байте приведет к нарушению

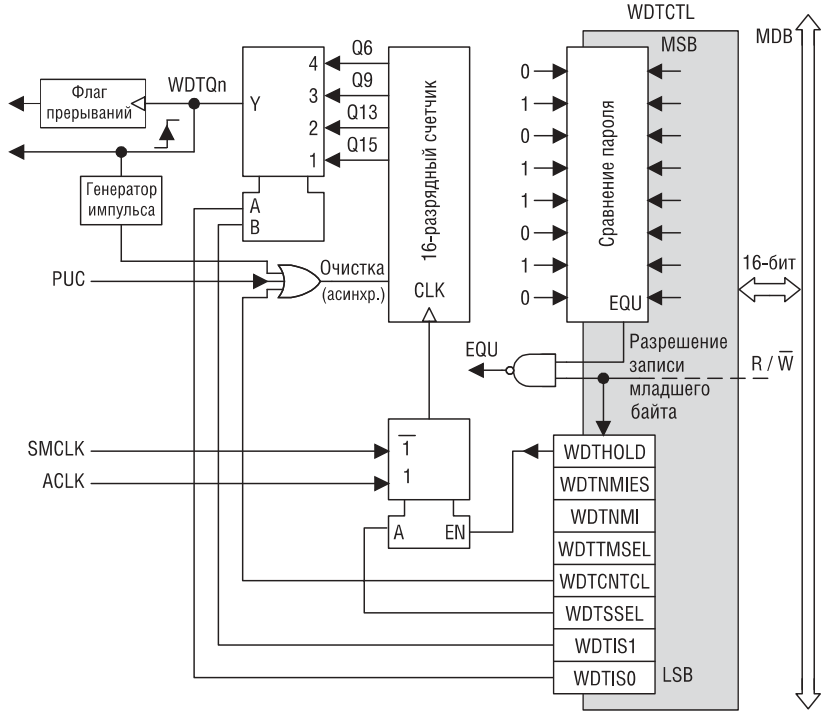


Рис. 10-1. Блок-схема сторожевого таймера

ключа безопасности и запуску системного сброса PUC независимо от режима таймера. При любом чтении WDTCTL в старшем байте читается 069h.

10.2.1. Счетчик сторожевого таймера

Счетчик сторожевого таймера (WDTCNT) – это 16-разрядный суммирующий счетчик, не имеющий прямого доступа из программы. Управление WDTCNT и выбор временных интервалов производится через регистр управления сторожевым таймером WDTCTL.

WDTCNT может тактироваться от ACLK или SMCLK. Источник тактирования выбирается с помощью бита WDTSSEL.

10.2.2. Сторожевой режим

После состояния PUC, модуль WDT конфигурируется в сторожевом режиме с начальным интервалом сброса ~32 мС с использованием DCOCLK. Пользо-

ватель должен настроить, остановить или очистить WDT до истечения начального интервала сброса, в противном случае будет сгенерирован новый сигнал PUC. Когда WDT сконфигурирован в сторожевом режиме, запись в WDTCTL неправильного пароля или истечение выбранного интервала времени приведет к запуску PUC. PUC сбросит WDT к исходному состоянию и сконфигурирует вывод nonRST/NMI на режим сброса.

10.2.3. Режим интервального таймера

Установка бита WDTTMSEL в «1» приводит к выбору режима интервального таймера. Этот режим можно использовать для получения периодических прерываний. В режиме интервального таймера флаг WDTIFG устанавливается по истечении выбранного интервала времени. PUC не генерируется в режиме интервального таймера по истечении установленного временного интервала, а WDTIFG разрешает биту WDTIE оставаться неизменным.

Когда биты WDTIE и GIE установлены, флаг WDTIFG запрашивает прерывание. Флаг прерывания WDTIFG автоматически сбрасывается, когда обслуживается его запрос на прерывание, либо же он может быть сброшен программно. Адреса векторов прерывания различаются для интервального и сторожевого режимов таймера.

Примечание: Модификация сторожевого таймера

Интервал WDT должен быть изменен совместно с WDTCNTCL=1 в одной команде, чтобы избежать неожиданной немедленной генерации PUC или прерывания. Модуль WDT должен быть приостановлен перед сменой источника тактирования для предотвращения возможности установки некорректного интервала.

10.2.4. Прерывания сторожевого таймера

WDT использует два бита в SFR для управления прерыванием.

- Флаг прерывания WDT, WDTIFG, расположенный в IFG1.0
- Бит разрешения прерывания от WDT, WDTIE, расположенный в IE1.0

Если WDT используется в сторожевом режиме, флаг WDTIFG является источником вектора прерывания по сбросу. WDTIFG может быть использован процедурой обработки прерывания по сбросу для определения, был ли сторожевой таймер причиной сброса устройства. Если флаг установлен, состояние сброса было инициировано сторожевым таймером по истечении времени, либо произошло нарушение ключа безопасности. Если WDTIFG очищен, сброс был вызван другим источником.

При использовании WDT в режиме интервального таймера, флаг WDTIFG устанавливается после выбранного временного интервала и запрашивает прерывание интервального таймера WDT, если установлены биты WDTIE и GIE. Вектор прерывания интервального таймера отличается от вектора сброса, используемого в сторожевом режиме. В режиме интервального таймера флаг WDTIFG сбрасывается автоматически при обработке прерывания, либо же он может быть сброшен программно.

10.2.5. Работа в режимах пониженного энергопотребления

Устройства MSP430 имеют несколько режимов пониженного энергопотребления. Различные сигналы тактирования доступны в различных режимах пониженного энергопотребления. Потребности пользовательского приложения и тип используемой системы тактирования определяют, как WDT должен быть сконфигурирован. К примеру, WDT не должен конфигурироваться в сторожевом режиме с SMCLK в качестве источника тактирования, если пользователь хочет использовать 3-й режим пониженного потребления, поскольку SMCLK не активен в режиме LPM3 и WDT не будет функционировать. Если сторожевой таймер не нужен, с помощью бита WDTHOLD можно остановить WDTCNT, чтобы уменьшить энергопотребление.

10.2.6. Примеры программного обеспечения

Любая операция записи в WDTCTL должна быть операцией-словом со значением 05Ah (WDTPW) в старшем байте:

```
;Периодическая отчистка активного сторожевого таймера
MOV #WDTPW+WDTCNTCL,&WDTCTL
;
;Изменение интервала сторожевого таймера
MOV #WDTPW+WDTCNTL+SSEL,&WDTCTL
;
;Останов сторожевого таймера
MOV #WDTPW+WDTHOLD,&WDTCTL
;
;Переключение WDT в интервальный режим с интервалом clock/8192
MOV #WDTPW+WDTCNTCL+WDTTMSEL+WDTIS0,&WDTCTL
```

10.3. Регистры сторожевого таймера

Регистры модуля сторожевого таймера приведены в таблице 10-1.

Таблица 10-1. Регистры сторожевого таймера.

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр управления сторожевым таймером	WDTCTL	Чтение/запись	0120h	06900h после PUC
SFR регистр 1 разрешения прерываний	IE1	Чтение/запись	0000h	Сброс с PUC
SFR регистр 1 флагов прерываний	IFG1	Чтение/запись	0002h	Сброс с PUC1

1) WDTIFG сбрасывается с POR

WDTCTL, регистр сторожевого таймера

WDTPWx	**Биты 15-8**	Пароль сторожевого таймера. Всегда читается как 069h. Должен записываться как 05Ah, в противном случае будет сгенерирован PUC.
WDTHOLD	**Бит 7**	Останов сторожевого таймера. Этот бит останавливает сторожевой таймер. Установка WDTHOLD=1, когда WDT не используется, позволяет снизить энергопотребление. 0 – Сторожевой таймер не остановлен 1 – Сторожевой таймер остановлен
WDTNMIES	**Бит 6**	Выбор фронта NMI сторожевого таймера. Этот бит позволяет выбрать фронт прерывания для NMI прерывания при WDTNMI=1. Изменение этого бита может вызвать NMI. Чтобы избежать случайного запуска NMI следует изменять этот бит при WDTNMI=0. 0 – NMI прерывание происходит по переднему фронту 1 – NMI прерывание происходит по спаду

WDTNMI	**Бит 5**	Выбор NMI сторожевого таймера. Этот бит позволяет установить режим функционирования вывода nonRST/NMI. 0 – Функция сброса 1 – Функция NMI
WDTTMSEL	**Бит 4**	Выбор режима сторожевого таймера 0 – Сторожевой режим 1 – Режим интервального таймера
WDTCNTCL	**Бит 3**	Очистка счетчика сторожевого таймера. Установкой WDTCNTCL=1 производится очистка счетчика до значения 0000h. Бит WDTCNTCL автоматически сбрасывается. 0 – Действие не производится 1 – WDTCNT = 0000h
WDTSSEL	**Бит 2**	Выбор источника тактирования сторожевого таймера 0 – SMCLK 1 – ACLK
WDTISx	**Биты 1-0**	Выбор интервала сторожевого таймера. Эти биты определяют интервал времени сторожевого таймера, по истечении которого устанавливается флаг WDTIFG и/или генерируется сигнал PUC. 00 – Частота источника тактирования сторожевого таймера /32768 01 – Частота источника тактирования сторожевого таймера /8192 10 – Частота источника тактирования сторожевого таймера /512 11 – Частота источника тактирования сторожевого таймера /64

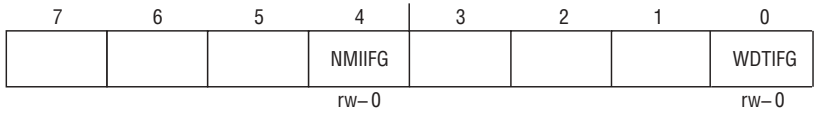
IE1, регистр 1 разрешения прерываний

7	6	5	4	3	2	1	0
			NMIIE				WDTIE
			rw–0				rw–0

	Биты 7-1	Эти биты могут быть использованы другими модулями. См. справочное руководство конкретного устройства.
NMIIE	**Бит 4**	Разрешение прерывания NMI. Этот бит разрешает прерывание NMI. Поскольку другие биты в IE1 могут быть использованы другими модулями, рекомендуется устанавливать и очищать этот бит с помощью команд BIS.B или BIC.B, а не командами MOV.B или CLR.B. 0 – Прерывание запрещено 1 – Прерывание разрешено

WDTIE	**Бит 0**	Разрешение прерывания от сторожевого таймера. Этот бит разрешает прерывание WDTIFG в режиме интервального таймера. Нет необходимости устанавливать этот бит в режиме сторожевого таймера. Поскольку другие биты в IE1 могут быть использованы другими модулями, рекомендуется устанавливать и очищать эти биты с помощью команд `BIS.B` или `BIC.B`, а не командами `MOV.B` или `CLR.B`. 0 – Прерывание запрещено 1 – Прерывание разрешено

IFG1, регистр 1 флагов прерываний



	Биты 7-1	Эти биты могут быть использованы другими модулями. См. справочное руководство конкретного устройства.
NMIIFG	**Бит 4**	Флаг прерывания NMI. NMIIFG должен быть сброшен программно. Поскольку другие биты в IFG1 могут быть использованы другими модулями, рекомендуется очищать WDTIFG с помощью команд BIS.B или BIC.B, а не командами MOV.B или CLR.B. 0 – Прерывание не ожидается 1 – Прерывание ожидается
WDTIE	**Бит 0**	Флаг прерывания сторожевого таймера. В сторожевом режиме WDTIFG остается установленным до сброса программным обеспечением. В интервальном режиме бит WDTIFG сбрасывается автоматически при обслуживании прерывания или же может быть сброшен программно. Поскольку другие биты в IE1 могут быть использованы другими модулями, рекомендуется устанавливать и очищать эти биты с помощью команд `BIS.B` или `BIC.B`, а не командами `MOV.B` или `CLR.B`. 0 – Прерывание не ожидается 1 – Прерывание ожидается

MSP430x1xxFamily

Таймер A

Раздел XI.

Texas Instruments

Таймер A

Таймер A – это 16-разрядный таймер/счетчик с тремя регистрами захвата/сравнения. В этом разделе описывается таймер A. Он реализован во всех устройствах MSP430x1xx.

11.1. Введение в таймер A

Таймер A – это 16-разрядный таймер/счетчик с тремя регистрами захвата/сравнения. Таймер A может обеспечить множество захватов/сравнений, выходов ШИМ и выдержку временных интервалов. Таймер A также имеет обширные возможности прерывания. Прерывания могут быть сгенерированы от счетчика при переполнении и от каждого из регистров захвата/сравнения.

Таймер A обладает следующими возможностями:

- Асинхронный 16-разрядный таймер/счетчик с четырьмя режимами работы;
- Выбираемый и конфигурируемый источник тактирования;
- Три конфигурируемых регистра захвата/сравнения;
- Конфигурируемые выходы с возможностью ШИМ;
- Асинхронная фиксация (защелка) входа и выхода;
- Регистр вектора прерываний для быстрого декодирования всех прерываний таймера A.

Блок-схема таймера A показана на рис. 11-1.

Примечание: Использование слова «счет»

«Счет» используется везде в этом разделе. Это способ показать, что счетчик должен быть в процессе подсчета для выполняемого действия. Если в счетчик напрямую записывается конкретное значение, соответствующе действие не происходит.

11.2. Функционирование таймера A

Модуль таймера A конфигурируется программным обеспечением пользователя. Настройка и работа таймера A рассматриваются в нижеследующих разделах.

11.2.1. 16-разрядный таймер-счетчик

Регистр 16-разрядного таймера/счетчика TAR, инкрементируется или декрементируется (в зависимости от режима работы) с каждым нарастающим фронтом тактового сигнала. TAR может быть программно прочитан и записан. Кроме того, таймер может генерировать прерывание при переполнении.

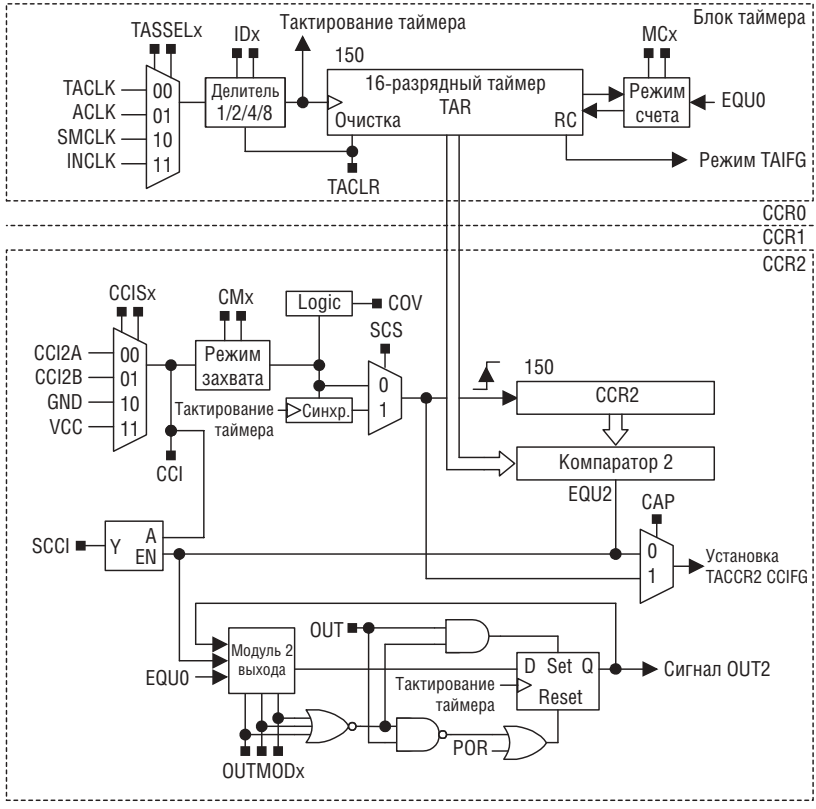


Рис. 11-1. Блок-схема таймера A

TAR можно очистить установкой бита TACLR. Установка TACLR также очищает делитель тактовой частоты и направление счета для режима вверх/вниз.

Примечание: Изменение регистров таймера A

Рекомендуется останавливать таймер перед изменением режима его работы (за исключением операций с флагом прерывания и разрешения прерывания, и TACLR), чтобы предотвратить его некорректную работу.

Когда сигнал TACLK асинхронен с сигналом тактирования ЦПУ, любое чтение из TAR должно происходить, когда таймер не работает, в противном случае результаты могут быть непредсказуемы. Любая запись в TAR приведет к немедленному результату.

Выбор источника тактирования и делитель

Тактирование таймера TACLK может производиться от ACLK, SMCLK или от внешнего источника через TACLK или INCLK. Источник тактовых импульсов выбирается с помощью битов TASSELx. Выбранный источник тактирования может быть подключен к таймеру напрямую или через делитель на 2, 4 или 8 с помощью битов IDx. Делитель TACLK сбрасывается при установке бита TACLR.

11.2.2. Запуск таймера

Таймер может быть запущен или перезапущен следующими способами:

- Таймер считает, когда MCx > 0 и источник тактовых импульсов активен
- Когда таймер в режиме «вверх» или «вверх/вниз», он может быть остановлен путем записи 0 в TACCR0. Затем таймер может быть перезапущен путем записи ненулевого значения в TACCR0. В этом случае таймер начинает инкрементироваться от нуля.

11.2.3. Управление режимом таймера

Таймер имеет четыре режима работы, описанных в таблице 11-1: «стоп», «вверх», «непрерывный» и «вверх/вниз». Режимы работы выбираются с помощью битов MCx.

Таблица 11-1. Режимы таймера.

MCx	Режим	Описание
00	Стоп	Останов таймера
01	Вверх	Таймер многократно считает от нуля до значения в TACCR0
10	Непрерывный	Таймер многократно считает от нуля до значения 0FFFFh
11	Вверх/вниз	Таймер многократно считает от нуля вверх до значения в TACCR0 и назад до нуля.

Режим «Вверх»

Режим «вверх» используется, если период таймера должен отличаться от количества отсчетов 0FFFFh. Таймер многократно считает вверх до значения, содержащегося в регистре сравнения TACCR0, который задает период, как показано на рис.11-2. Количество отсчетов таймера в периоде равно TACCR0+1. Когда значение таймера становится равно содержимому TACCR0, таймер перезапускается, начиная счет от нуля. Если режим «вверх» выбран, когда значение таймера больше содержимого TACCR0, таймер немедленно перезапускается, начиная считать от нуля.

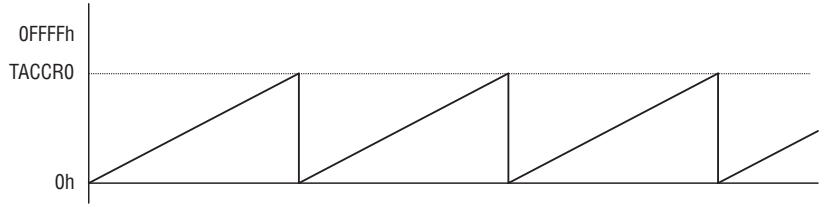


Рис. 11-2. Режим «Вверх»

Флаг прерывания TACCR0 CCIFG устанавливается, когда значение таймера равно содержимому TACCR0. Флаг прерывания TAIFG устанавливается, когда таймер считает от TACCR0 к нулю. На рис. 11-3 показан цикл установки флагов.

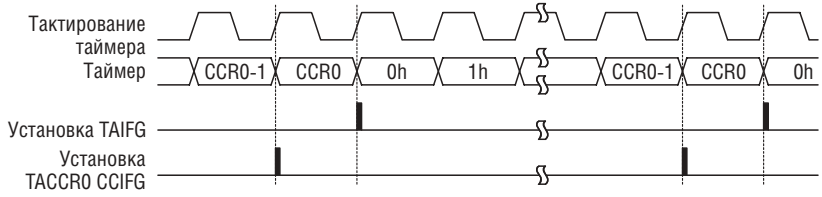


Рис. 11-3. Установка флагов в режиме «Вверх»

Изменение регистра периода TACCR0

Когда изменяется TACCR0 во время работы таймера, таймер продолжает отсчет вверх до новой границы периода, если новый период больше или равен старому или больше текущего значения счета. Если новый период меньше величины текущего значения счета, таймер обнуляется. Однако до обнуления счетчика может произойти один дополнительный отсчет.

Непрерывный режим

В непрерывном режиме таймер многократно считает вверх до 0FFFFh и перезапускается от нуля, как показано на рис. 11-4. Регистр захвата/сравнения TACCR0 работает подобно другим регистрам захвата/сравнения.

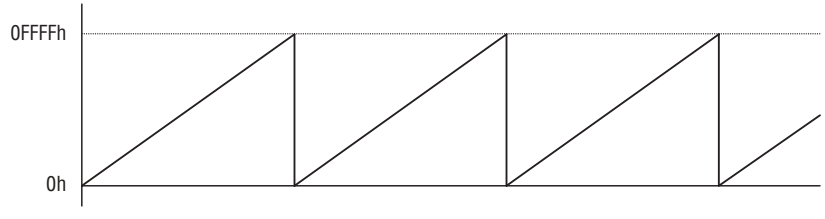


Рис. 11-4. Непрерывный режим

Флаг прерывания TAIFG устанавливается, когда таймер считает от 0FFFFh к нулю. На рис. 11-5 показан цикл установки флага.

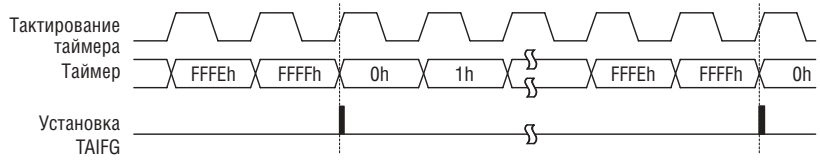


Рис. 11-5. Установка флага в непрерывном режиме

Использование непрерывного режима

Непрерывный режим может использоваться для генерации независимых временных интервалов и выходных частот. Каждый раз по завершении интервала генерируется прерывание. Следующий временной интервал добавляется к регистру TACCRx в процедуре обработки прерывания. На рис. 11-6 показаны два раздельных временных интервала t0 и t1, добавляемые к регистрам захвата/сравнения. При таком использовании временные интервалы управляются аппаратно, без программного обеспечения, без влияния времени задержки прерывания. При использовании всех трех регистров захвата/сравнения можно сгенерировать до трех независимых временных интервалов или выходных частот.

Временные интервалы можно реализовать также в других режимах, где TACCR0 используется как регистр периода. Их обработка более комплексна, поскольку сумма старого значения TACCRx и нового периода может быть выше

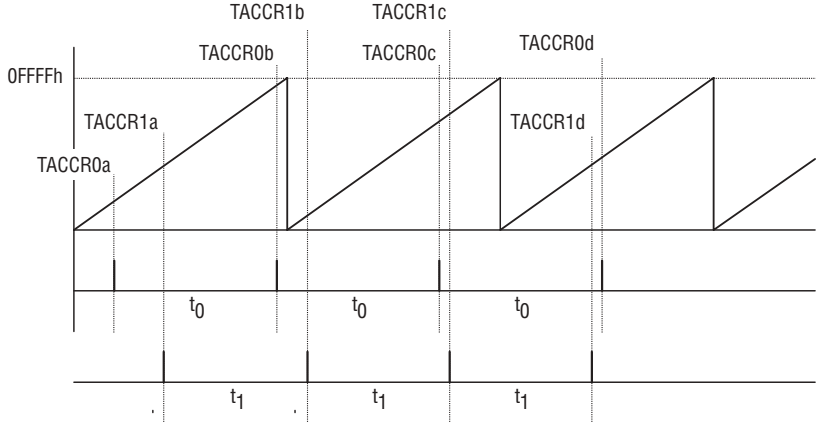


Рис. 11-6. Временные интервалы непрерывного режима

значения TACCR0. Когда предыдущее значение TACCRx плюс tx больше величины TACCR0, значение TACCR0 должно быть вычтено для получения правильного временного интервала.

Режим «вверх/вниз»

Режим «вверх/вниз» используется, если период таймера должен отличаться от значения 0FFFFh и необходима генерация симметричных импульсов. Таймер непрерывно считает вверх до значения, находящегося в регистре сравнения TACCR0 и назад к нулю, как показано на рис. 11-7. Период составляет удвоенную величину значения в TACCR0.

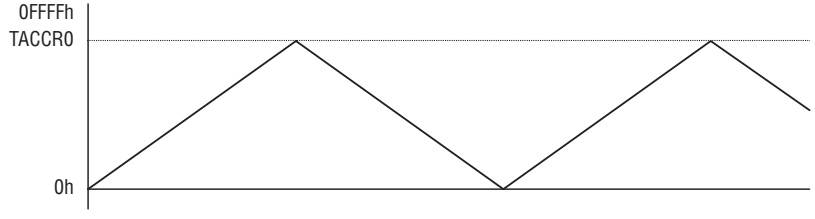


Рис. 11-7. Режим «вверх/вниз»

Направление счета запоминается. Это позволяет таймеру останавливаться и запускаться в том же направлении отсчета, которое было до останова. Если это нежелательно, для очистки направления нужно установить бит TACLR. Бит TACLR также очищает значения в TAR и в делителе TACLK.

В режиме «вверх/вниз» флаг прерывания TACCR0 CCIFG и флаг прерывания TAIFG устанавливаются только однажды во время периода, разделяясь 1/2 периода таймера. Флаг прерывания TACCR0 CCIFG устанавливается, когда таймер считает от TACCR0-1 к TACCR0, а флаг TAIFG устанавливается, когда таймер завершает счет вниз от 0001h к 0000h. На рис. 11-8 показан цикл установки флагов.

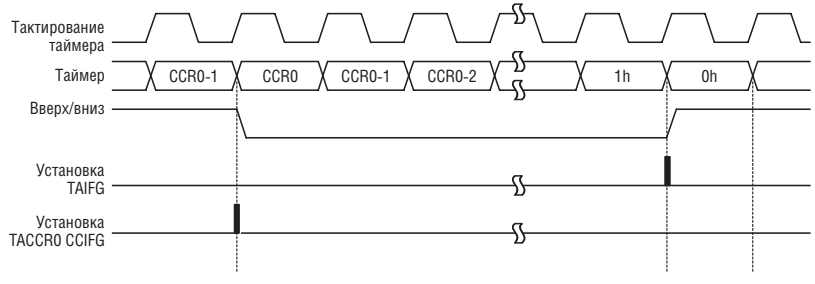


Рис. 11-8. Установка флагов в режиме вверх/вниз

Изменение регистра периода TACCR0

При изменении TACCR0 во время работы таймера и счете вниз, таймер продолжает счет до нуля. Новый период начинает действовать после завершения отсчета вниз до нуля.

Когда таймер выполняет отсчет вверх и новый период больше или равен старому периоду или же больше текущего отсчитанного значения, таймер считает вверх до конца нового периода перед отсчетом вниз. Когда же таймер производит отсчет вверх, а величина нового периода меньше текущего отсчитанного значения, таймер начинает считать вниз. Однако может произойти один дополнительный отсчет до начала счета вниз.

Использование режима вверх/вниз

Режим «вверх/вниз» поддерживает приложения, требующие наличия «мертвого» времени между выходными сигналами (см. раздел «Модуль вывода Таймера А»). К примеру, чтобы избежать перегрузки, два выхода, управляющие Н-мостом никогда не должны одновременно иметь высокий уровень. В примере, показанном на рис.11-9 величина времени простоя t_{dead} составляет:

$t_{dead} = t_{timer} \times (TACCR1 – TACCR2)$, где:

t_{dead} – интервал времени, в течение которого оба выхода должны быть неактивны

t_{timer} – время цикла тактирования таймера

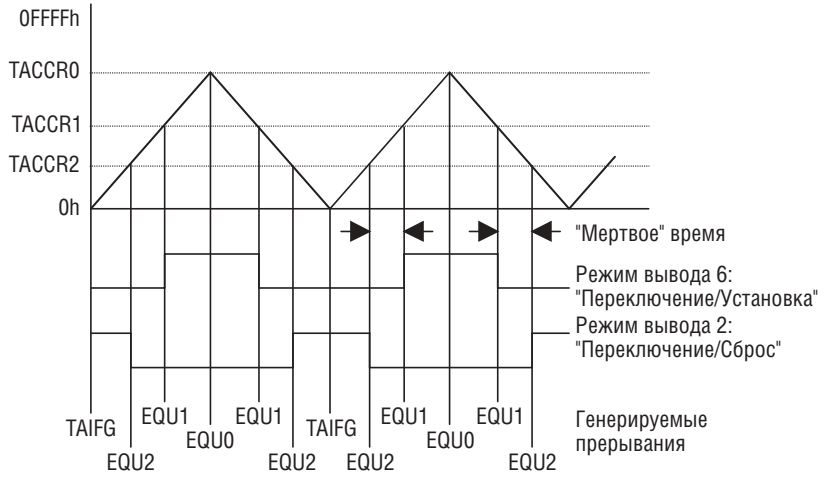


Рис. 11-9. Модуль вывода в режиме «вверх/вниз»

TACCRx – содержимое регистра захвата/сравнения x

Регистры TACCRx не буферизированы. Когда в них производится запись, они немедленно модифицируются. Поэтому, любое требуемое «мертвое» время не будет поддерживаться автоматически.

11.2.4. Блоки захвата/сравнения

В таймере A представлено три одинаковых блока захвата/сравнения TACCRx. Любой из блоков может быть использован для фиксации (захвата) данных таймера или для генерации временных интервалов.

Режим захвата

Режим захвата выбирается, когда CAP=1. Режим захвата используется для регистрации временных событий. Это может потребоваться для быстрых вычислений или для измерения времени. Входы захвата CCIxA и CCIxB подключаются к внешним выводам или к внутренним сигналам и выбираются с помощью битов CCISx. Биты CMx определяют, как будет происходит захват: по фронту входного сигнала, по его спаду или в обоих случаях. Захват происходит по выбранному фронту входного сигнала. Если захват произошел, тогда:

- Значение таймера копируется в регистр TACCRx
- Устанавливается флаг прерывания CCIFG

Уровень входного сигнала может быть прочитан в любое время через бит CCI. К входам CCIxA и CCIxB в устройствах семейства MSP430x1xx могут подключаться различные сигналы. См. справочное руководство конкретного устройства для выяснения особенностей подключения этих сигналов.

Сигнал захвата может быть асинхронен тактовой частоте таймера и вызывать состояние гонки сигналов. Установка бита SCS будет синхронизировать захват со следующим тактовым импульсом таймера. Рекомендуется устанавливать бит SCS для синхронизации сигнала захвата с тактовыми импульсами таймера. Это иллюстрируется на рис. 11-10.

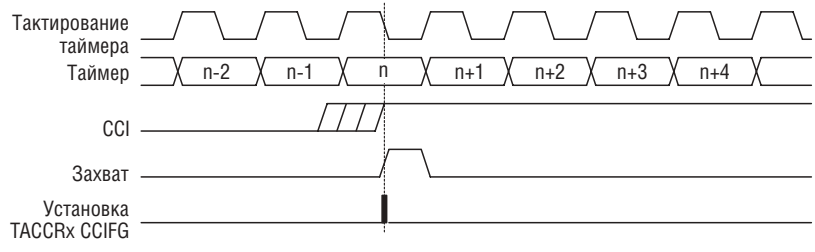


Рис. 11-10. Сигнал захвата (SCS=1)

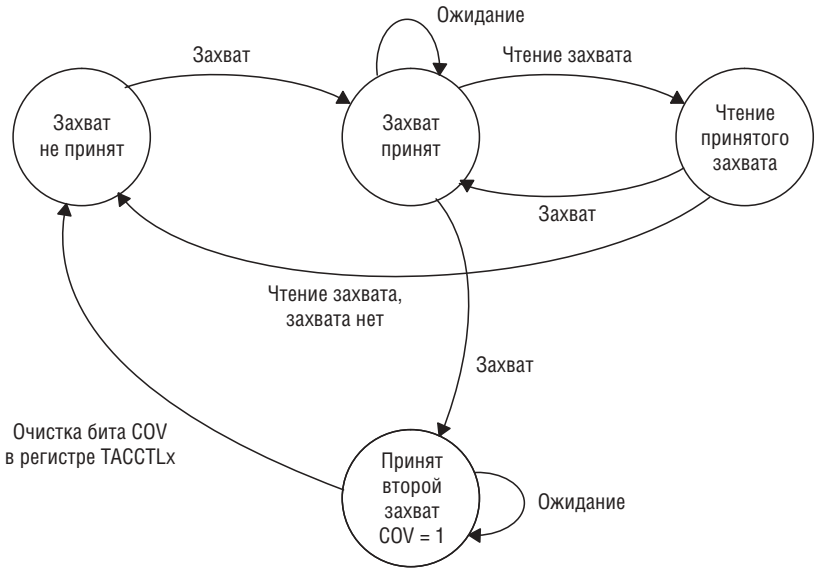


Рис. 11-11. Цикл захвата

Логика переполнения предусмотрена в каждом регистре захвата/сравнения для индикации в случае, если произошел второй захват перед прочтением значения первого захвата. Когда это происходит, устанавливается бит COV, как показано на рис. 11-11. Бит COV должен сбрасываться программно.

Инициирование захвата программным обеспечением

Захваты могут быть инициированы программно. Биты CMx могут быть установлены для выполнения захвата по обоим фронтам. В этом случае программное обеспечение устанавливает CCI=1 и переключает бит CCISO для переключения сигнала захвата между VCC и GND, инициируя захват каждый раз, когда CCISO изменяет состояние:

```
MOV #CAP+SCS+CCIS1+CM_3,&TACCTLx     ;Настройка TACCTLx
XOR #CCIS0,&TACCTLx                  ;TACCTLx = TAR
```

Режим сравнения

Режим сравнения выбирается, когда CAP=0. Режим сравнения используется для генерации выходных ШИМ-сигналов или прерываний через конкретные временные интервалы. Когда TAR досчитывает до значения в TACCRx, происходит следующее:

- Устанавливается флаг прерывания CCIFG
- Внутренний сигнал EQU=1
- EQUx воздействует на выход согласно режиму вывода
- Входной сигнал CCI фиксируется в SCCI

11.2.5. Модуль вывода

Каждый блок захвата/сравнения содержит модуль вывода. Модуль вывода используется для генерации выходных сигналов, в т.ч. таких, как ШИМ-сигналы. Каждый модуль вывода имеет восемь рабочих режимов, которые генерируют сигналы, основываясь на сигналах EQU0 и EQUx.

Режимы вывода

Режимы вывода устанавливаются битами OUTMODx, их описание приведено в таблице 11-2. Сигнал OUTx изменяется с нарастающим фронтом тактового сигнала таймера во всех режимах, кроме режима 0. Режимы вывода 2, 3, 6 и 7 не используются для модуля вывода 0, потому что EQUx=EQU0.

Таблица 11-2. Режимы вывода.

OUTMODx	Режим	Описание
000	Вывод	Выходной сигнал OUTx определяется битом OUTx. Сигнал OUTx изменяется немедленно при изменении OUTx.
001	Установка	Выход устанавливается, когда таймер досчитывает до значения в TACCRx. Он остается установленным до сброса таймера или до выбора другого режима вывода и воздействия на выход.
010	Переключение/Сброс	Выход переключается, когда таймер досчитывает до значения TACCRx. Он сбрасывается, когда таймер досчитывает до значения TACCR0.
011	Установка/Сброс	Выход устанавливается, когда таймер досчитывает до значения TACCRx. Он сбрасывается, когда таймер досчитывает до значения TACCR0.
100	Переключение	Выход переключается, когда таймер досчитывает до значения TACCRx. Период выходного сигнала равен удвоенному периоду таймера.
101	Сброс	Выход сбрасывается, когда таймер досчитывает до значения TACCRx. Это остается сброшенным до выбора другого режима вывода и воздействия на выход.
110	Переключение/Установка	Выход переключается, когда таймер досчитывает до значения TACCRx. Он устанавливается, когда таймер досчитывает до значения TACCR0.

Таблица 11-2. (Окончание)

OUTMODx	Режим	Описание
111	Сброс/Установка	Выход сбрасывается, когда таймер досчитывает до значения TACCRx. Он устанавливается, когда таймер досчитывает до значения TACCR0.

Пример вывода – таймер в режиме «вверх»

Сигнал OUTx изменяется, когда таймер досчитывает вверх до значения TACCRx и обратно от TACCR0 к нулю, в зависимости от режима вывода. Пример с использованием TACCR0 и TACCR1 показан на рис. 11-12.

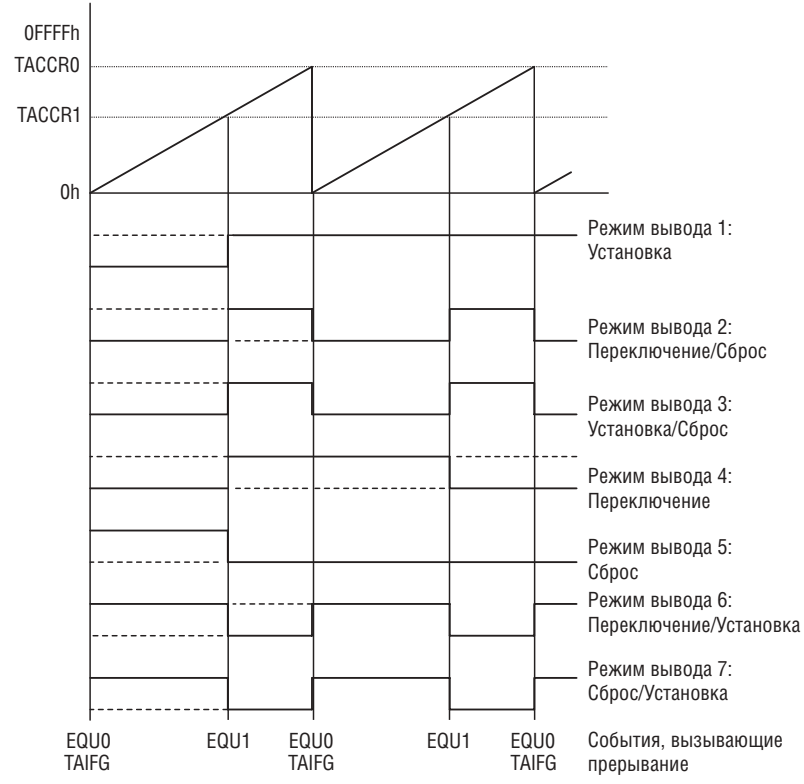


Рис. 11-12. Пример вывода – таймер в режиме «вверх»

Пример вывода – таймер в непрерывном режиме

Сигнал OUTx изменяется, когда таймер достигает значений TACCRx и TACCR0, в зависимости от режима вывода. Пример с использованием TACCR0 и TACCR1 показан на рис. 11-13.

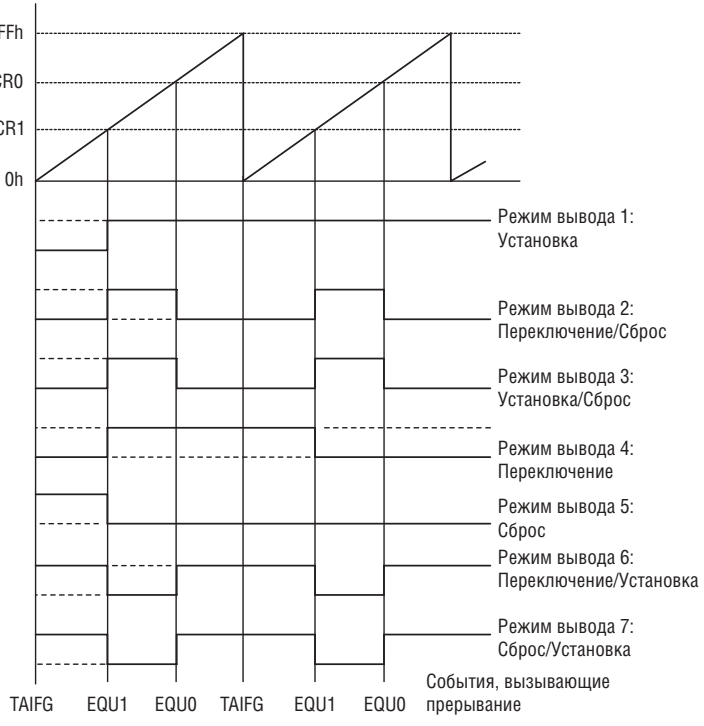


Рис. 11-13. Пример вывода – таймер в непрерывном режиме

Пример вывода – таймер в режиме «вверх/вниз»

Сигнал OUTx изменяется, когда таймер становится равным TACCRx в каждом направлении счета, а также когда таймер равен TACCR0, в зависимости от режима вывода. Пример с использованием TACCR0 и TACCR2 показан на рис. 11-14.

Примечание: Переключение между режимами вывода

При переключении между режимами вывода один из битов OUTMODx должен оставаться установленным во время перехода, кроме переключения в режим 0. В противном случае может произойти сбой, поскольку режим вывода 0 декодирует элемент NOR (НЕ-ИЛИ). Безопасный метод переключения между

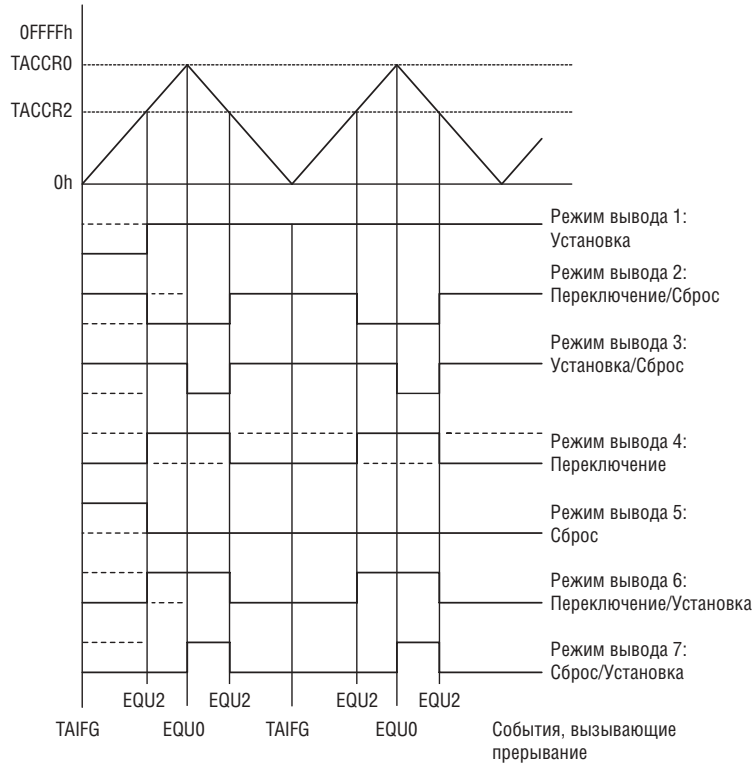


Рис. 11-14. Пример вывода – таймер в режиме «вверх/вниз»

режимами вывода заключается в использовании режима вывода 7 как переходного состояния:

```
BIS #OUTMOD_7,&TACCTLx       ;Установка режима вывода 7
BIC #OUTMODx,&TACCTLx        ;Очистка лишних битов
```

11.2.6. Прерывания Таймера A

С 16-разрядным модулем таймера A связаны два вектора прерываний:

- Вектор прерывания TACCR0 для TACCR0 CCIFG
- Вектор прерывания TAIV для всех других флагов CCIFG и TAIFG

В режиме захвата любой флаг CCIFG устанавливается, когда значение таймера зафиксировано в соответствующем регистре TACCRx. В режиме сравнения устанавливается любой флаг CCIFG, если TAR досчитал до соответствую-

щего значения TACCRx. Программное обеспечение может также устанавливать или очищать любой флаг CCIFG. Все флаги CCIFG запрашивают прерывание, когда установлены их соответствующие биты CCIE и бит GIE.

Прерывание TACCR0

Флаг TACCR0 CCIFG обладает наивысшим приоритетом прерывания Таймера A и имеет специализированный вектор прерывания, как показано на рис. 11-15. Флаг TACCR0 CCIFG автоматически сбрасывается, когда обслуживается запрос на прерывание TACCR0.

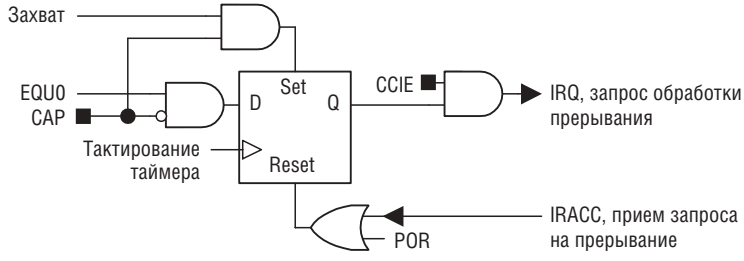


Рис. 11-15. Флаг прерывания захвата/сравнения TACCR0

Генератор вектора прерывания TAIV

Флаги TACCR1 CCIFG, TACCR2 CCIFG и TAIFG распределены по приоритетам и объединены в источник одного вектора прерывания. Регистр вектора прерывания TAIV используется для определения, какой флаг запросил прерывание.

Разрешенное прерывание с наивысшим приоритетом генерирует число в регистре TAIV (см. описание регистра). Можно оценить это число или добавить его к программному счетчику для автоматического входа в соответствующую программную процедуру. Запрещенные прерывания Таймера A не воздействуют на значение TAIV.

Любое обращение – чтение или запись регистра TAIV – автоматически сбрасывает флаг наивысшего ожидающего прерывания. Если установлен другой флаг прерывания, будет немедленно сгенерировано другое прерывание после обработки начального прерывания. Например, если флаги TACCR1 и TACCR2 CCIFG установлены, когда процедура обработки прерывания обращается к регистру TAIV, флаг TACCR1 CCIFG автоматически сбрасывается. После выполнения команды RETI процедуры обработки прерывания, флаг TACCR2 CCIFG генерирует другое прерывание.

Пример программного обеспечения, использующего TAIV

Приведенный далее пример программного обеспечения показывает рекомендуемое использование TAIV и величину издержек на управление. Значение TAIV добавляется к программному счетчику PC для автоматического перехода к соответствующей программной процедуре.

Числа в правом поле показывают необходимое количество циклов ЦПУ для каждой команды. Программные издержки различных источников прерывания включают задержки прерывания и циклы возврата из прерывания, но не учитывают собственно время обработки задачи. Задержки делятся на:

- Блок захвата/сравнения TACCR0 11 циклов
- Блоки захвата/сравнения TACCR1, TACCR2 16 циклов
- Переполнение таймера TAIFG 14 циклов

```
;Обработчик прерывания для TACCR0 CCIFG.                          Циклы
CCIFG_0_HND
;           ...   ;Начало времени задержки обработчика прерывания 6
            RETI                                                  5
;Обработчик прерывания для TAIFG, TACCR1 и TACCR2 CCIFG.
TA_HND      ...   ;Задержка прерывания                            6
            ADD &TAIV,PC ;Добавление смещения к таблице переходов 3
            RETI  ;Вектор 0: Нет прерывания                       5
            JMP CCIFG_1_HND ;Вектор 2: TACCR1                     2
            JMP CCIFG_2_HND ;Вектор 4: TACCR2                     2
            RETI            ;Вектор 6: Зарезервировано            5
            RETI            ;Вектор 8: Зарезервировано            5
TAIFG_HND                   ;Вектор 10: Флаг TAIFG
            ...             ;Задача стартует здесь
            RETI                                                  5
CCIFG_2_HND                 ;Вектор 4: TACCR2
            ...             ;Задача стартует здесь
            RETI            ;Возврат к главной программе          5
CCIFG_1_HND                 ;Вектор 2: TACCR1
            ...             ;Задача стартует здесь
            RETI            ;Возврат к главной программе          5
```

11.3. Регистры Таймера A

Перечень регистров Таймера A приведен в таблице 11-3.

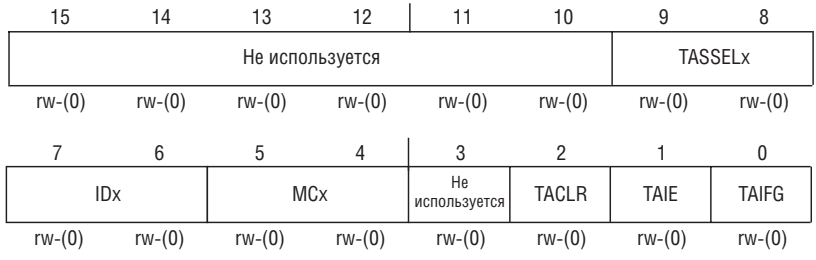
Таблица 11-3. Регистры Таймера A.

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Управление Таймером A	TACTL	Чтение/запись	0160h	Сброс с POR
Счетчик Таймера A	TAR	Чтение/запись	0170h	Сброс с POR

Таблица 11-3. (Окончание)

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр 0 управления захватом/ сравнением Таймера A	TACCTL0	Чтение/запись	0162h	Сброс с POR
Регистр 0 захвата/сравнения Таймера A	TACCR0	Чтение/запись	0172h	Сброс с POR
Регистр 1 управления захватом/ сравнением Таймера A	TACCTL1	Чтение/запись	0164h	Сброс с POR
Регистр 1 захвата/сравнения Таймера A	TACCR1	Чтение/запись	0174h	Сброс с POR
Регистр 2 управления захватом/ сравнением Таймера A	TACCTL2	Чтение/запись	0166h	Сброс с POR
Регистр 2 захвата/сравнения Таймера A	TACCR2	Чтение/запись	0176h	Сброс с POR
Вектор прерывания Таймера A	TAIV	Только чтение	012Eh	Сброс с POR

TACTL, регистр управления Таймером A



Не используются	Биты 15-10	Не используются
TASSELx	**Биты 9-8**	Выбор источника тактирования Таймера A 00 – TACLK 01 – ACLK 10 – SMCLK 11 – INCLK
IDx	**Биты 7-6**	Входной делитель. Эти биты позволяют выбрать коэффициент деления для входной тактовой частоты. 00 – /1 01 – /2 10 – /4 11 – /8

MCx	**Биты 5-4**	Выбор режима. Установка MCx=00h, когда Таймер A не используется, позволяет уменьшить потребляемую мощность. 00 – Режим остановка: таймер остановлен 01 – Режим «вверх»: таймер считает вверх к TACCR0 10 – Непрерывный режим: таймер считает вверх к 0FFFFh 11 – Режим «вверх/вниз»: таймер считает вверх к TACCR0, затем вниз к 0000h
Не используется	**Бит 3**	Не используется
TACLR	**Бит 2**	Очистка Таймера A. Установка этого бита сбрасывает TAR, IDx и выбранное направление счета. Бит TACLR автоматически сбрасывается и всегда читается как нуль.
TAIE	**Бит 1**	Разрешение прерывания от Таймера A. Этот бит разрешает запрос прерывания TAIFG. 0 – Запрещение прерывания 1 – Разрешение прерывания
TAIFG	**Бит 0**	Флаг прерывания Таймера A 0 – Прерывание не ожидается 1 – Ожидается прерывание

TAR, регистр Таймера A

15	14	13	12	11	10	9	8
TARx							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
TARx							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

TARx	**Биты 15-0**	Регистр Таймера A. Регистр TAR является счетчиком Таймера A.

TACCTLx, регистр управления захватом/сравнением

15	14	13	12	11	10	9	8
CMx		CCISx		SCS	SCCI	Не используется	CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)	r-(0)	rw-(0)

7	6	5	4	3	2	1	0
OUTMODx			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

CMx	**Биты 15-14**	Режим захвата 00 – Нет захвата 01 – Захват по нарастающему (переднему) фронту 10 – Захват по заднему фронту 11 – Захват как по переднему, так и по заднему фронтам
CCISx	**Биты 13-12**	Выбор входа захвата/сравнения. Этими битами выбирается входной сигнал TACCRx. См. справочное руководство конкретного устройства для выяснения подробностей подключения сигналов. 00 – CCIxA 01 – CCIxB 10 – GND 11 – VCC
SCS	**Бит 1**	Синхронизация источника захвата. Этот бит используется для синхронизации входного сигнала захвата с тактовым сигналом таймера. 0 – Асинхронный захват 1 – Синхронный захват
SCCI	**Бит 10**	Синхронизация входа захвата/сравнения. Выбранный входной сигнал CCI фиксируется по сигналу EQUx и может быть прочитан через этот бит.
Не используется	**Бит 9**	Не используется. Только читается. Всегда читается как 0.
CAP	**Бит 8**	Режим захвата. 0 – Режим сравнения 1 – Режим захвата
OUTMODx	**Биты 7-5**	Режим вывода. Режимы 2, 3, 6 и 7 не используются для TACCR0, поскольку EQUx=EQU0. 000 – Значение бита OUT 001 – Установка 010 – Переключение/сброс 011 – Установка/сброс 100 – Переключение 101 – Сброс 110 – Переключение/установка 111 – Сброс/установка
CCIE	**Бит 4**	Разрешение прерывания по захвату/сравнению. Этот бит разрешает запрос прерывания от соответствующего флага CCIFG. 0 – Запрещение прерывания 1 – Разрешение прерывания
CCI	**Бит 3**	Вход захвата/сравнения. Выбранный входной сигнал может быть прочитан этим битом.
OUT	**Бит 2**	Выход. Этот бит указывает состояние выхода. Если выбран режима вывода 0, этот бит напрямую управляет состоянием выхода. 0 – Низкий уровень выхода 1 – Высокий уровень выхода

COV	**Бит 1**	Переполнение захвата. Этот бит указывает, что произошло переполнение захвата. Бит COV должен быть сброшен программно 0 – Переполнения захвата не произошло 1 – Произошло переполнение захвата
CCIFG	**Бит 0**	Флаг прерывания захвата/сравнения 0 – Прерывание не ожидается 1 – Ожидается прерывание

TAIV, регистр вектора прерывания таймера A

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
r0	r0	r0	r0	r0	r0	r0	r0

7	6	5	4	3	2	1	0
0	0	0	0	TAIVx			0
r0	r0	r0	r0	r-(0)	r-(0)	r-(0)	r0

TARx	**Биты 15-0**	Значение вектора прерывания таймера A

Содержимое TAIV	Источник прерывания	Флаг прерывания	Приоритет прерывания
00h	Прерывание не ожидается	-	
02h	Захват/сравнение 1	TACCR1 CCIFG	Высший
04h	Захват/сравнение 2	TACCR2 CCIFG	
06h	Зарезервировано	-	
08h	Зарезервировано	-	
0Ah	Переполнение таймера	TAIFG	
0Ch	Зарезервировано	-	
0Eh	Зарезервировано	-	Низший

MSP430x1xxFamily

Таймер B

Раздел XII.

Texas Instruments

Таймер B

Таймер B – это 16-разрядный таймер/счетчик со несколькими регистрами захвата/сравнения. В этом разделе описывается работа таймера B. Таймер B3 (с тремя регистрами захвата/сравнения) реализован в устройствах MSP430x13x и MSP430x15x. Таймер B7 (с семью регистрами захвата/сравнения) реализован в устройствах MSP430x14x и MSP430x16x.

12.1. Введение в таймер B

Таймер B – это 16-разрядный таймер/счетчик с тремя или семью регистрами захвата/сравнения. Таймер B может поддерживать несколько режимов захвата/сравнения, вывод ШИМ-сигналов и выдержку временных интервалов. Таймер B также имеет расширенные возможности прерываний. Прерывания могут быть сгенерированы при переполнении счетчика и от каждого из регистров захвата/сравнения.

Таймер B обладает следующими возможностями:

- Асинхронный 16-разрядный таймер/счетчик с четырьмя режимами работы и четырьмя настраиваемыми длительностями
- Выбираемые и конфигурируемые источники тактирования
- Три или семь конфигурируемых регистров захвата/сравнения
- Конфигурируемые выходы с возможностью ШИМ
- Защелки сравнения с двойной буферизацией и синхронизируемой загрузкой
- Регистр вектора прерывания для быстрого декодирования всех прерываний таймера B

Блок-схема таймера B показана на рис. 12-1.

Примечание: Использование слова «счет»

«Счет» используется везде в этом разделе. Это способ показать, что счетчик должен быть в процессе подсчета для выполняемого действия во взятом месте. Если в счетчик напрямую записывается конкретное значение, соответствующе действие не происходит.

12.1.1. Сходства и различия с таймером A

Таймер B идентичен таймеру A, но со следующими исключениями:

- Длина таймера B программируется и может составлять 8, 10, 12 или 16 бит
- Регистры TBCCRx таймера B имеют двойную буферизацию и могут группироваться

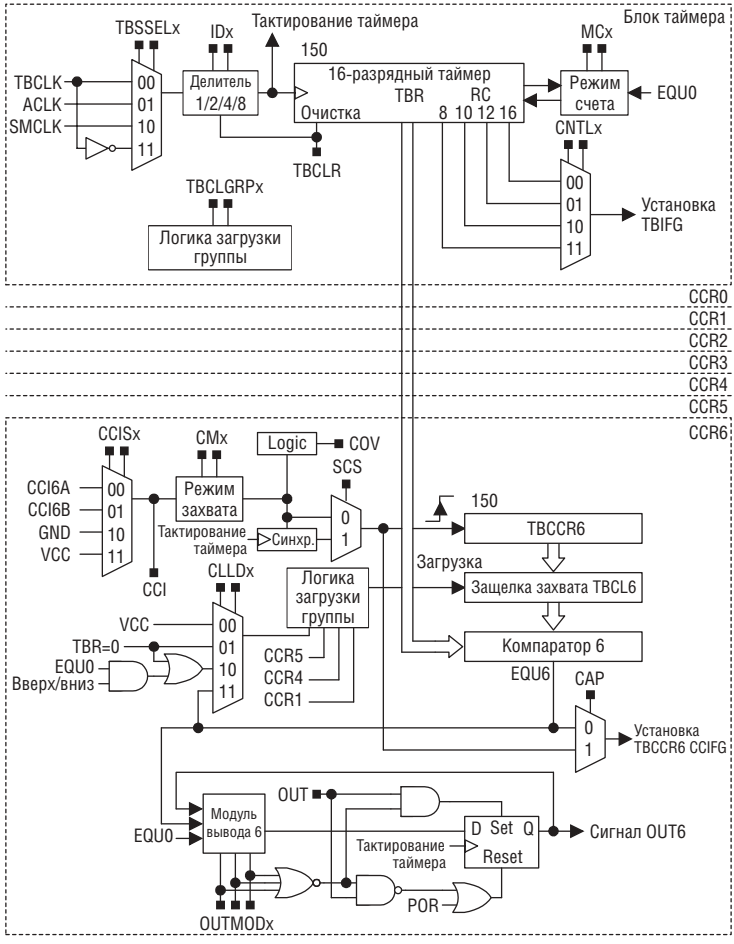


Рис. 12-1. Блок-схема таймера B

- Все выходы таймера B могут быть переведены в третье (высокоимпедансное) состояние
- Функция бита SCCI не реализована в таймере B.

12.2. Работа таймера B

Модуль таймера B конфигурируется программным обеспечением пользователя. Настройка и работа таймера B рассматривается в следующих разделах

12.2.1. 16-разрядный счетчик таймера

16-разрядный регистр таймера/счетчика TBR инкрементируется и декрементируется (в зависимости от режима работы) по каждому нарастающему фронту тактового сигнала. Регистр TBR может программно читаться и записываться. Помимо этого, таймер может генерировать прерывание при его переполнении.

Регистр TBR может быть очищен установкой бита TBCLR. Установка TBCLR также очищает делитель тактовой частоты и направление счета для режима «вверх/вниз».

Примечание: Модификация регистров таймера B

Рекомендуется останавливать таймер перед изменением режима работы (кроме операций с флагом прерывания и разрешения прерывания, и TBCLR) во избежание его ошибочной работы.

Когда TBCLK асинхронен тактовой частоте ЦПУ, любое чтение из TBR должно выполняться при неработающем таймере, в противном случае результат может оказаться непредсказуемым. Любая запись в TBR приведет к немедленному результату.

Длина TBR

Таймер B конфигурируется для работы в качестве 8, 10, 12 или 16-разрядного таймера с помощью битов CNTLx. Максимальное значение счета TBR(max) для выбранной длины соответственно составит 0FFh, 03FFh, 0FFFh и 0FFFFh. Данные, записанные в регистр TBR в 8-ми, 10-ти и 12-разрядном режиме выравниваются по правому знаку с нулями впереди.

Выбор источника тактирования и делитель

В качестве источников тактовой частоты TBCLK могут выступать ACLK, SMCLK или внешние сигналы, поступающие через TBCLK или INCLK. Источник тактирования выбирается битами TBSSELx. Выбранный источник тактирования может подключаться к таймеру напрямую или через делитель на 2, 4 или 8 с помощью битов IDx. Делитель TBCLK сбрасывается при установке бита TBCLR.

12.2.2. Старт таймера

Таймер может быть запущен или перезапущен следующими способами:

- Таймер считает, когда MCx > 0 и активен источник тактовых сигналов
- Когда таймер находится в режиме счета «вверх» или «вверх/вниз», его можно остановить загрузкой нуля в TBCL0. Таймер может быть тогда и перезапущен при загрузке ненулевого значения в TBCL0. В этом случае таймер начинает инкрементирование вверх от нуля.

12.2.3. Управление режимом таймера

Таймер имеет четыре режима работы, описанных в таблице 12-1: «стоп», «вверх», «непрерывный» и «вверх/вниз». Рабочий режим выбирается с помощью битов MCx.

Таблица 12-1. Режимы таймера.

MCx	Режим	Описание
00	Стоп	Останов таймера
01	Вверх	Таймер многократно считает от нуля до значения в регистре сравнения TBCL0
10	Непрерывный	Таймер многократно считает от нуля до значения, выбранного битами TBCNTLx.
11	Вверх/вниз	Таймер многократно считает от нуля вверх до значения в TBCL0 и назад до нуля.

Режим «вверх»

Режим «вверх» используется, если период таймера должен быть отличен от количества отсчетов TBR(max). Таймер многократно считает вверх до значения в защелке сравнения TBCL0, которое определяет период, как показано на рис. 12-2. Количество отсчетов таймера в периоде равно TBCL0+1. Когда значение таймера равно TBCL0, таймер перезапускается на счет с нуля. Если режим «вверх» выбран, когда значение таймера больше чем в TBCL0, таймер немедленно перезапускается на отсчет с нуля.

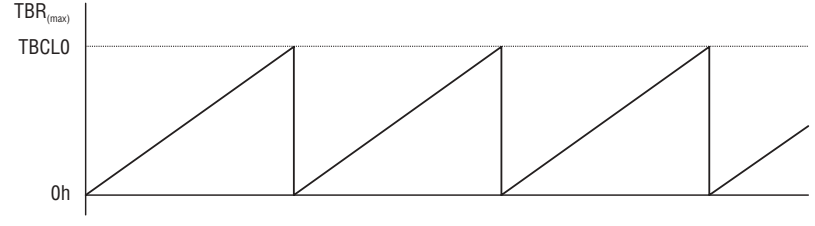


Рис. 12-2. Режим «вверх»

Флаг прерывания TBCCR0 CCIFG устанавливается, когда значение таймера равно значению TBCL0. Флаг прерывания TBIFG устанавливается, когда таймер пересчитывает от TBCL0 к нулю. На рис. 12-3 показан цикл установки флагов.

Изменение регистра периода TBCL0

При изменении TBCL0 во время работы таймера, когда TBCL0 находится в режиме непосредственной загрузки, если новый период больше старого или

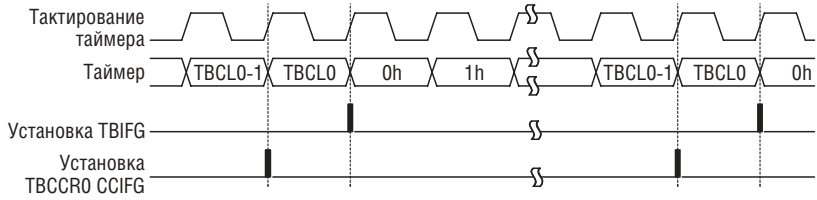


Рис. 12-3. Установка флагов в режиме «вверх»

больше текущего значения счета, таймер считает вверх к новому периоду. Если новый период меньше текущего значения счета, таймер обнуляется. Однако, может произойти один дополнительный отсчет перед обнулением счетчика.

Непрерывный режим

В непрерывном режиме таймер многократно считает вверх до значения $TBR_{(max)}$ и перезапускается от нуля, как показано на рис. 12-4. Защелка сравнения TBCL0 работает подобно другим регистрам захвата/сравнения.

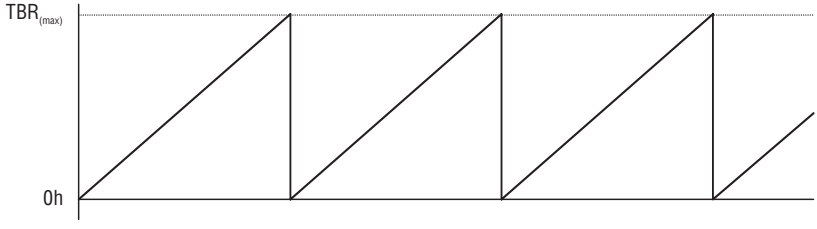


Рис. 12-4. Непрерывный режим

Флаг прерывания TBIFG устанавливается, когда таймер считает от $TBR_{(max)}$ к нулю. На рис. 12-5 показан цикл установки флагов.

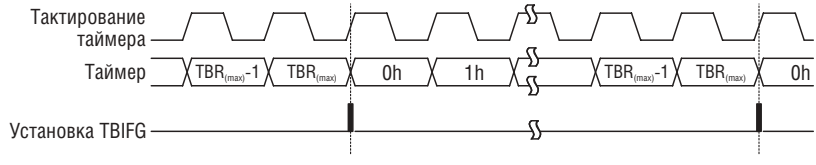


Рис. 12-5. Установка флагов в непрерывном режиме

Использование непрерывного режима

Непрерывный режим может использоваться для генерации независимых временных интервалов и выходных частот. Каждый раз по завершении интервала

генерируется прерывание. Следующий временной интервал добавляется к защелке TBCLx в процедуре обработки прерывания. На рис. 12-6 показаны два раздельных временных интервала t0 и t1, добавляемые к регистрам захвата/сравнения. Выдержка временных интервалов осуществляется аппаратно, без участия программного обеспечения и без воздействия задержек прерывания. Может быть сгенерировано до трех (таймер B3) или до семи (таймер B7) независимых временных интервалов или выходных частот при использовании регистров захвата/сравнения.

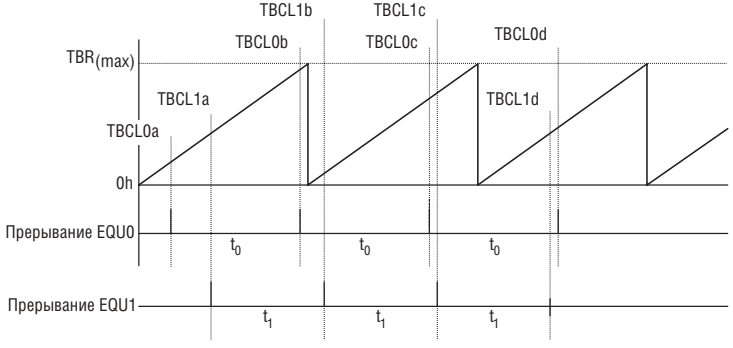


Рис. 12-6. Временные интервалы в «непрерывном» режиме

Временные интервалы могут быть реализованы также в других режимах, в которых TBCL0 используется как регистр периода. Работа с ними более комплексна, т.к. сумма старого значения TBCLx и нового периода может оказаться больше значения TBCL0. Когда сумма предыдущего значения TBCLx плюс tx больше содержимого TBCL0, для получения правильного временного интервала необходимо вычитать старое значение TBCL0.

Режим «вверх/вниз»

Режим «вверх/вниз» используется, если период таймера должен отличаться от величины отсчетов TBR(max), а также если требуется генерация симметричных импульсов. Таймер многократно считает вверх до значения в защелке сравнения TBCL0 и назад к нулю, как показано на рис. 12-7. Период в этом случае равен удвоенному значению TBCL0.

Примечание: TBCL0 > TBR(max)

Если TBCL0 > TBR(max), счетчик работает так, как если бы был сконфигурирован для непрерывного режима. Счет вниз от TBR(max) до нуля не производится.

Направление счета защелкивается. Это позволяет таймеру останавливаться и перезапускаться с тем направлением счета, которое было до его остано-

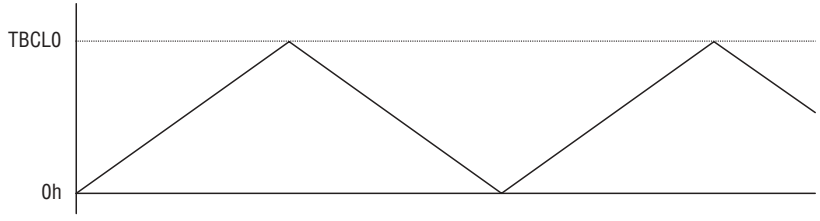


Рис. 12-7. Режим «вверх/вниз»

ва. Если это не желательно, для сброса направления нужно использовать бит TBCLR. Бит TBCLR также очищает значение TBR и делитель TBCLK.

В режиме «вверх/вниз» флаг прерывания TBCCR0 CCIFG и флаг прерывания TBIFG устанавливаются один раз в период, разделяясь 1/2-ой периода таймера. Флаг прерывания TBCCR0 CCIFG устанавливается, когда таймер считает от TBCL0-1 до TBCL0, а TBIFG устанавливается, когда таймер завершает счет вниз от 0001h к 0000h. На рис. 12-8 показан цикл установки флагов.

Изменение значение регистра периода TBCL0

Когда изменяется TBCL0 во время работы таймера при выбранном направ-

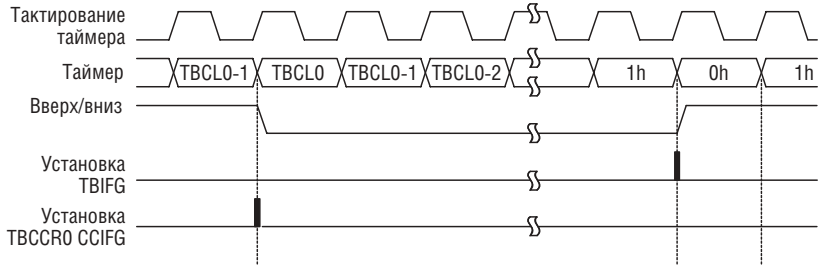


Рис. 12-8. Установка флагов в режиме «вверх/вниз»

лении счета вниз и установленном непосредственном режиме загрузке TBCL0, таймер продолжает отсчет вниз до нуля. Новый период будет активизирован после завершения счета счетчика до нуля.

Если таймер считает вверх, когда новый период защелкнут в TBCL0, и новый период больше или равен старому периоду или же больше текущего значения счета, таймер считает вверх до нового периода перед счетом вниз. Когда таймер считает вверх, а новый период меньше текущего значения счета в момент загрузки TBCL0, таймер начинает считать вниз. Однако, может произойти один дополнительный отсчет прежде, чем счетчик начнет считать вниз.

Использование режима «вверх/вниз»

Режим «вверх/вниз» поддерживает приложения, требующие наличия «мертвого» времени между выходными сигналами (см. раздел Модуль вывода Таймера B). К примеру, чтобы избежать режима перегрузки, два выхода, управляющие H-мостом никогда не должны иметь высокий уровень одновременно. В примере, показанном на рис. 12-9 «мертвое» время t_{dead} задается так:

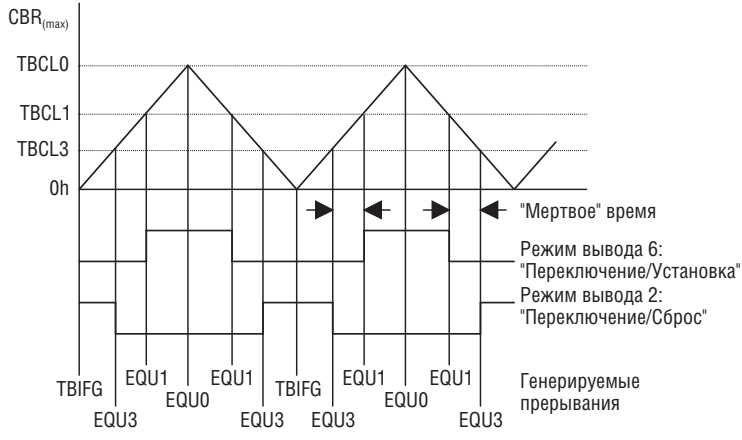


Рис. 12-9. Модуль вывода в режиме «вверх/вниз»

$t_{dead} = t_{timer} \times (TBCL1 – TBCL3)$, где:

t_{dead} – время, в течение которого оба выхода не активны;

t_{timer} – время цикла такта таймера;

TBCLx – содержимое защелки сравнения x.

Возможность одновременной загрузки сгруппированных защелок гарантирует наличие «мертвого» времени.

12.2.4. Блоки захвата/сравнения

Три или семь идентичных блоков захвата/сравнения TBCCRx представлены в таймере B. Любой из блоков может использоваться для захвата данных таймера или для генерации временных интервалов.

Режим захвата

Режим захвата выбирается, когда CAP=1. Режим захвата используется для регистрации временных событий. Он может быть использован для выполнения быстрых вычислений или измерений времени. Входы захвата CCIxA и CCIxB под-

ключаются к внешним выводам или внутренним сигналам и выбираются с помощью битов CCISx. Биты CMx позволяют задать, как будет происходить захват: по переднему, по заднему или по обеим фронтам входного сигнала. Захват происходит по выбранному фронту входного сигнала. Если захват произошел, то:

- Значение таймера копируется в регистр TBCCRx
- Устанавливается флаг прерывания CCIFG

Уровень входного сигнала может быть прочитан в любое время через бит CCI. Устройства семейства MSP430x1xx могут иметь различные сигналы, подключенные к CCIxA и CCIxB. См. справочное руководство конкретного устройства для выяснения подробностей подключения этих сигналов.

Сигнал захвата может быть асинхронен тактовой частоте таймера и вызывать состояние гонки сигналов. При установке бита SCS захват синхронизируется со следующим тактовым импульсом таймера. Рекомендуется устанавливать бит SCS для синхронизации сигнала захвата с тактовыми импульсами таймера. Это иллюстрируется на рис. 12-10.

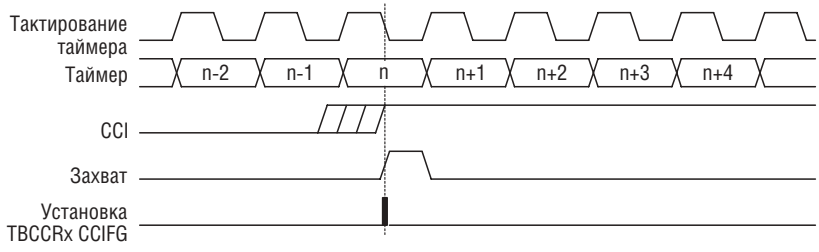


Рис. 12-10. Сигнал захвата (SCS=1)

Логика переполнения предусмотрена в каждом регистре захвата/сравнения для индикации в случае, если произошел второй захват перед прочтением значения первого захвата. Когда это происходит, устанавливается бит COV, как показано на рис. 12-11. Бит COV должен сбрасываться программно.

Захват, инициируемый программным обеспечением

Захваты могут быть инициированы программно. Биты CMx могут устанавливаться для выполнения захвата на обоих фронтах. В этом случае программное обеспечение устанавливает бит CCIS1=1 и переключает бит CCIS0 для переключения сигнала захвата между VCC и GND, инициируя захват каждый раз, когда CCISO изменяет состояние:

```
MOV #CAP+SCS+CCIS1+CM_3,&TBCCTLx          ;Настройка TBCCTLx
XOR #CCIS0,&TBCCTLx                       ;TBCCTLx = TBR
```

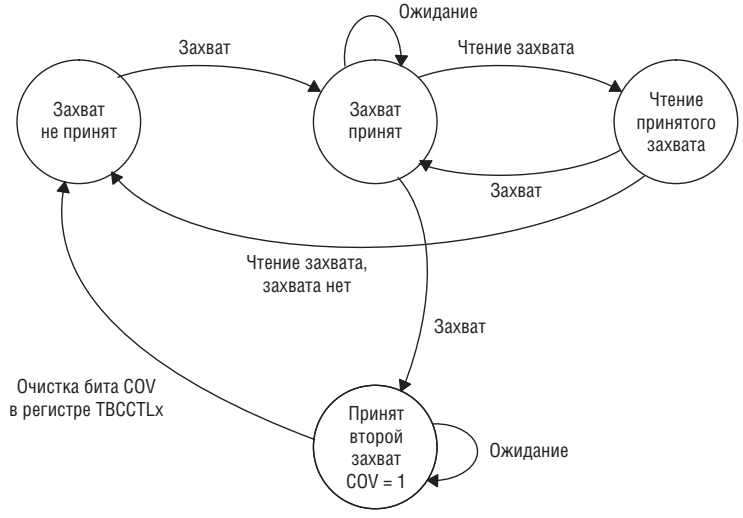


Рис. 12-11. Цикл захвата

Режим сравнения

Режим сравнения выбирается, когда CAP=0. Режим сравнения используется для генерации выходных ШИМ – сигналов или прерываний через заданные временные интервалы. Когда TAR досчитывает до значения в TBCLx, происходит следующее:

- Устанавливается флаг прерывания CCIFG
- Внутренний сигнал EQU=1
- EQU воздействует на вывод согласно режиму вывода

Защелка сравнения TBCLx

Защелка сравнения TBCCRx TBCLx хранит данные для сравнения со значением таймера в режиме сравнения. TBCLx буферизирована TBCCRx. Буферизация защелки сравнения дает пользователю контроль над обновлением периода сравнения. Пользователь не имеет прямого доступа к TBCLx. Сравниваемые данные записываются в каждый регистр TBCCRx и автоматически переносятся в TBCLx. Синхронизация переноса из TBCCRx в TBCLx выбирается пользователем с помощью битов CLLDx в соответствии с описанным в таблице 12-2.

Таблица 12-2. Варианты загрузки TBCLx

CLLDx	Описание
00	Новые данные переносятся из TBCCRx в TBCLx немедленно, когда записывается TBCCRx.
01	Новые данные переносятся из TBCCRx в TBCLx, когда TBR досчитывает до 0.

CLLDx	Описание
10	Новые данные переносятся из TBCCRx в TBCLx, когда TBR досчитывает до 0 в режимах «вверх» и «непрерывный». Новые данные переносятся из TBCCRx в TBCLx, когда TBR досчитывает до старого значения TBCL0 или до 0 в режиме «вверх/вниз».
11	Новые данные переносятся из TBCCRx в TBCLx, когда TBR досчитывает до старого значения TBCLx.

Группировка защелок сравнения

Несколько защелок сравнения могут быть сгруппированы вместе для одновременного обновления с помощью битов TBCLGRPx. При использовании групп биты CLLDx самого младшего TBCCRx в группе определяют вариант загрузки для каждой защелки группы, кроме случая, когда TBCLGRP=3, как показано в таблице 12-3. Биты CLLDx, управляющие TBCCRx, не должны устанавливаться в ноль. Когда биты CLLDx управления регистром TBCCRx установлены в ноль, все защелки сравнения немедленно обновляются при выполнении записи в их соответствующие регистры TBCCRx – группировки защелок сравнения не происходит.

При загрузке сгруппированных защелок сравнения необходимо соблюдать два условия. Во-первых, все регистры группы должны быть обновлены, даже когда новые данные TBCCRx равны старым данным TBCCRx. Во-вторых, должно произойти событие загрузки.

Таблица 12-3. Рабочие режимы защелок сравнения

TBCLGRPx	Группировка	Управление обновлением
00	Нет	Индивидуальное
01	TBCL1 + TBCL2 TBCL3 + TBCL4 TBCL5 + TBCL6	TBCCR1 TBCCR3 TBCCR5
10	TBCL1 + TBCL2 + TBCL3 TBCL4 + TBCL5 + TBCL6	TBCCR1 TBCCR4
11	TBCL0 + TBCL1 + TBCL2 +TBCL3 + TBCL4 + TBCL5 + TBCL6	TBCCR1

12.2.5. Модуль вывода

Каждый блок захвата/сравнения содержит модуль вывода. Модуль вывода используется для генерации выходных сигналов, таких как ШИМ-сигналы. Каждый модуль вывода имеет восемь рабочих режимов, которые генерируют сигналы, основываясь на сигналах EQU0 и EQUx. Функция ножки TBOUTH может использоваться для установки всех выходов таймера B в «третье» (высокоимпедансное) состояние. Когда для ножки выбрана функция TBOUTH и на вывод подан высокий лог. уровень, все выходы таймера B находятся в «третьем» состоянии.

Режимы вывода

Режимы вывода задаются битами OUTMODx, а их описание приведено в таблице 12-4. Сигнал OUTx изменяется по нарастающему фронту тактового сигнала таймера во всех режимах, кроме режима 0. Режимы вывода 2, 3, 6 и 7 не используются для модуля вывода 0, поскольку EQUx=EQU0.

Таблица 12-4. Режимы вывода.

OUTMODx	Режим	Описание
000	Вывод	Выходной сигнал OUTx определяется битом OUTx. Сигнал OUTx изменяется немедленно при изменении OUTx.
001	Установка	Выход устанавливается, когда таймер досчитывает до значения в TBCLx. Он остается установленным до сброса таймера или до выбора другого режима вывода и воздействия на выход.
010	Переключение/сброс	Выход переключается, когда таймер досчитывает до значения TBCLx. Он сбрасывается, когда таймер досчитывает до значения TBCL0.
011	Установка/сброс	Выход устанавливается, когда таймер досчитывает до значения TBCLx. Он сбрасывается, когда таймер досчитывает до значения TBCL0.
100	Переключение	Выход переключается, когда таймер досчитывает до значения TBCLx. Период выходного сигнала равен удвоенному периоду таймера.
101	Сброс	Выход сбрасывается, когда таймер досчитывает до значения TBCLx. Он остается сброшенным до выбора другого режима вывода и воздействия на выход.
110	Переключение/установка	Выход переключается, когда таймер досчитывает до значения TBCLx. Он устанавливается, когда таймер досчитывает до значения TBCL0.
111	Сброс/установка	Выход сбрасывается, когда таймер досчитывает до значения TBCLx. Он устанавливается, когда таймер досчитывает до значения TBCL0.

Пример вывода – таймер в режиме «вверх»

Сигнал OUTx изменяется, когда таймер досчитывает вверх до значения TBCLx и обратно от TBCL0 к нулю, в зависимости от режима вывода. Пример с использованием TBCL0 и TBCL1 показан на рис. 12-12.

Пример вывода – таймер в непрерывном режиме

Сигнал OUTx изменяется, когда таймер достигает значений TBCLx и TBCL0, в зависимости от режима вывода. Пример с использованием TBCL0 и TBCL1 показан на рис. 12-13.

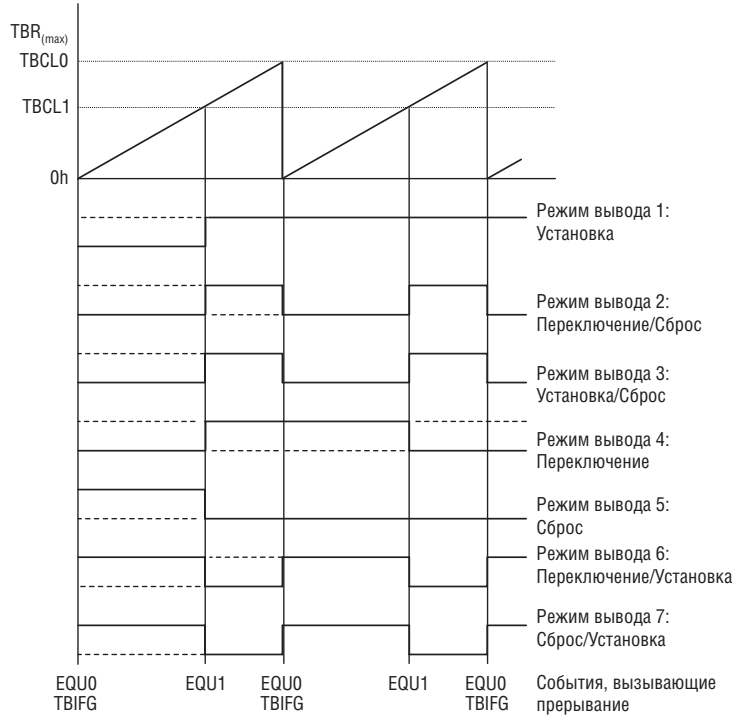


Рис. 12-12. Пример вывода – таймер в режиме «вверх»

Пример вывода – таймер в режиме «вверх/вниз»

Сигнал OUTx изменяется, когда таймер равен TBCLx при любом направлении счета, либо когда таймер равен TBCL0, в зависимости от режима вывода. Пример с использованием TBCL0 и TBCL3 показан на рис. 12-14.

Примечание: Переключение между режимами вывода

При переключении между режимами вывода один из битов OUTMODx должен оставаться установленным во время перехода между режимами, кроме переключения в режим 0. В противном случае может произойти сбой, поскольку режим вывода 0 декодирует элемент NOR (НЕ-ИЛИ). Безопасный метод переключения между режимами вывода заключается в использовании режима вывода 7 как переходного состояния:

```
BIS #OUTMOD_7,&TBCCTLx          ;Установка режима вывода =7
BIC #OUTMODx,&TBCCTLx           ;Очистка ненужных битов
```

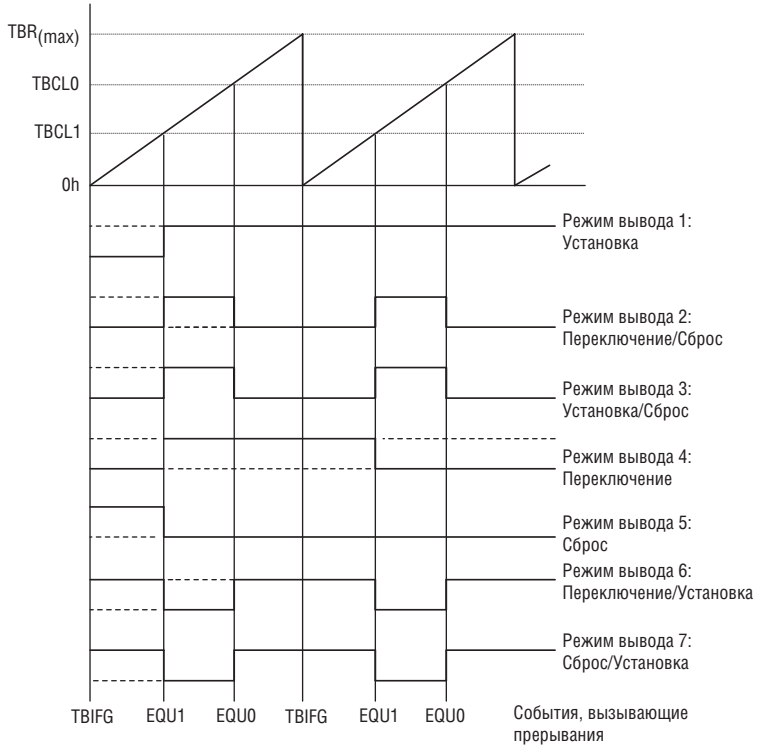


Рис. 12-13. Пример вывода – таймер в «непрерывном» режиме

12.2.6. Прерывания Таймера B

С 16-разрядным модулем таймера B связаны два вектора прерываний:

- Вектор прерывания TBCCR0 для TBCCR0 CCIFG
- Вектор прерывания TBIV для всех других флагов CCIFG и TBIFG

В режиме захвата любой флаг CCIFG устанавливается, когда значение таймера зафиксировано в соответствующем регистре TBCCRx. В режиме сравнения устанавливается любой флаг CCIFG, если TBR досчитал до соответствующего значения TBCLx. Программное обеспечение может также устанавливать или очищать любой флаг CCIFG. Все флаги CCIFG запрашивают прерывания, когда установлены их соответствующие биты CCIE и бит GIE.

Вектор прерывания TBCCR0

Флаг TBCCR0 CCIFG обладает наивысшим приоритетом прерывания Таймера B и имеет специализированный вектор прерывания, как показано на

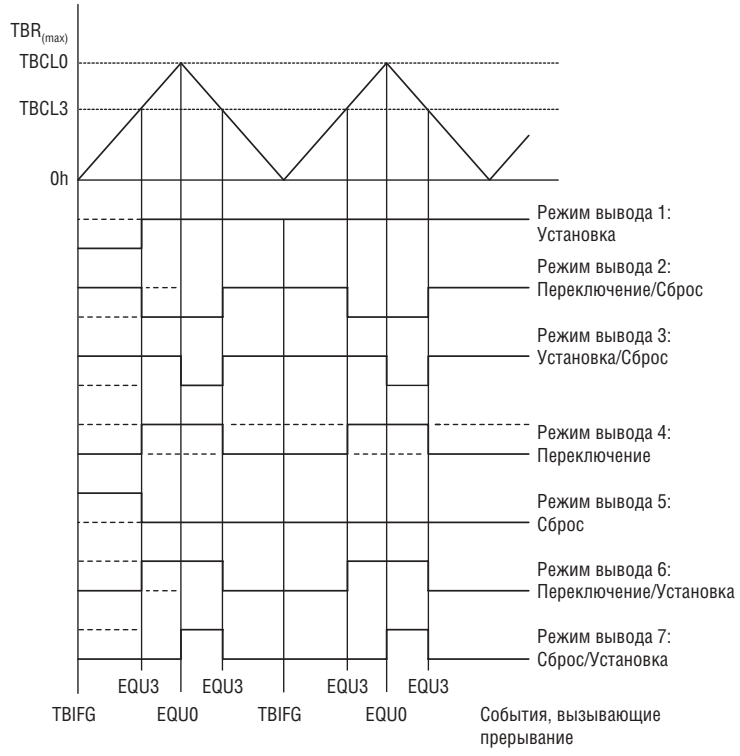


Рис. 12-14. Пример вывода – таймер в режиме «вверх/вниз»

рис. 12-15. Флаг TBCCR0 CCIFG автоматически сбрасывается, когда обслуживается запрос на прерывание TBCCR0.

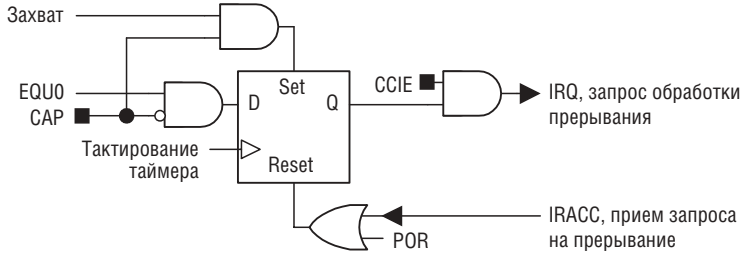


Рис. 12-15. Флаг прерывания TBCCR0 захвата/сравнения

Генератор вектора прерывания TBIV

Флаг TBIFG и флаги TBCCRx CCIFG (кроме TBCCR0 CCIFG) распределены по приоритетам и объединены в источник одного вектора прерывания. Регистр вектора прерывания TBIV используется для определения, какой флаг запросил прерывание.

Разрешенное прерывание с наивысшим приоритетом (кроме TBCCR0 CCIFG) генерирует число в регистре TBIV (см. описание регистра). Можно оценить это число или добавить его к программному счетчику для автоматического входа в соответствующую процедуру программы. Запрещенные прерывания таймера B не воздействуют на значение TBIV.

Любой тип доступа: чтение или запись регистра TBIV автоматически сбрасывает флаг наивысшего ожидающего прерывания. Если установлен другой флаг прерывания, будет немедленно сгенерировано другое прерывание после обработки изначального прерывания. К примеру, если флаги TBCCR1 и TBCCR2 CCIFG установлены, когда процедура обработки прерывания обращается к регистру TBIV, флаг TBCCR1 CCIFG автоматически сбрасывается. После выполнения команды процедуры обработки прерывания RETI, флаг TBCCR2 CCIFG генерирует другое прерывание.

Пример программного обеспечения, использующего TBIV

Приведенный далее пример программного обеспечения показывает рекомендуемое использование TBIV и величину издержек времени на управление. Значение TBIV добавляется к программному счетчику PC для автоматического перехода к соответствующей программной процедуре.

Числа в правом поле показывают необходимое количество циклов ЦПУ для каждой команды. Программные издержки различных источников прерывания включают задержку прерывания и циклы возврата из прерывания, но не учитывают собственно время обработки задачи. Задержки делятся на:

- Блок захвата/сравнения CCR0 11 циклов
- Блоки захвата/сравнения с CCR1 по CCR6 16 циклов
- Переполнение таймера TBIFG 14 циклов

Следующий пример программного обеспечения показывает рекомендуемое использование TBIV для таймера B3:

```
;Обработчик прерывания для TBCCR0 CCIFG.                    Циклы
CCIFG_0_HND
           ...   ;Начало времени задержки обработчика прерывания 6
           RETI                                                 5
;Обработчик прерывания для TBIFG, TBCCR1 и TBCCR2 CCIFG.
TB_HND     $           ;Задержка прерывания                     6
           ADD &TBIV,PC ;Добавление смещения к таблице переходов 3
           RETI        ;Вектор 0: Нет прерывания                5
           JMP CCIFG_1_HND ;Вектор 2: Модуль 1                  2
```

```
           JMP CCIFG_2_HND ;Вектор 4: Модуль 2                  2
           RETI            ;Вектор 6
           RETI            ;Вектор 8
           RETI     ;Вектор 10
           RETI     ;Вектор 12
TBIFG_HND          ;Вектор 14: Флаг TIMOV
           ...     ;Задача стартует здесь
           RETI                                                 5
CCIFG_2_HND        ;Вектор 4: Модуль 2
           ...     ;Задача стартует здесь
           RETI    ;Назад к главной программе                   5
                   ;Модуль 1 обработчика позволяет узнать,
                   ;что ожидается любое другое прерывание:
                   ;на это тратится 5 циклов, но 9 циклов можно
                   ;сэкономить, если ожидается другое прерывание
CCIFG_1_HND        ;Вектор 6: Модуль 3
           ...     ;Задача стартует здесь
JMP TB_HND         ;Просмотр ожидающих прерываний              2
```

12.3. Регистры таймера B

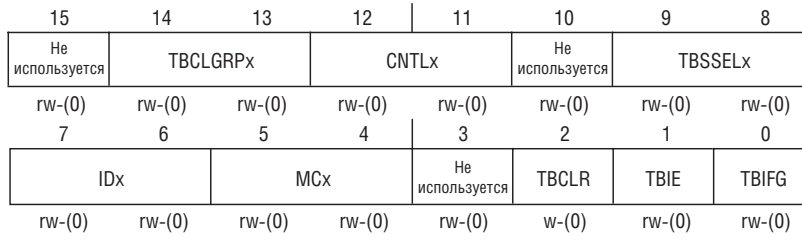
Перечень регистров таймера B приведен в таблице 12-5.

Таблица 12-5. Регистры Таймера B.

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Управление Таймером B	TBCTL	Чтение/запись	0180h	Сброс с POR
Счетчик Таймера B	TBR	Чтение/запись	0190h	Сброс с POR
Регистр 0 управления захватом/ сравнением таймера B	TBCCTL0	Чтение/запись	0182h	Сброс с POR
Регистр 0 захвата/сравнения таймера B	TBCCR0	Чтение/запись	0192h	Сброс с POR
Регистр 1 управления захватом/ сравнением таймера B	TBCCTL1	Чтение/запись	0184h	Сброс с POR
Регистр 1 захвата/сравнения таймера B	TBCCR1	Чтение/запись	0194h	Сброс с POR
Регистр 2 управления захватом/ сравнением таймера B	TBCCTL2	Чтение/запись	0186h	Сброс с POR
Регистр 2 захвата/сравнения таймера B	TBCCR2	Чтение/запись	0196h	Сброс с POR
Регистр 3 управления захватом/ сравнением таймера B	TBCCTL3	Чтение/запись	0188h	Сброс с POR
Регистр 3 захвата/сравнения таймера B	TBCCR3	Чтение/запись	0198h	Сброс с POR

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр 4 управления захватом/сравнением таймера B	TBCCTL4	Чтение/запись	018Ah	Сброс с POR
Регистр 4 захвата/сравнения таймера B	TBCCR4	Чтение/запись	019Ah	Сброс с POR
Регистр 5 управления захватом/сравнением таймера B	TBCCTL5	Чтение/запись	018Ch	Сброс с POR
Регистр 5 захвата/сравнения таймера B	TBCCR5	Чтение/запись	019Ch	Сброс с POR
Регистр 6 управления захватом/сравнением таймера B	TBCCTL6	Чтение/запись	018Eh	Сброс с POR
Регистр 6 захвата/сравнения таймера B	TBCCR6	Чтение/запись	019Eh	Сброс с POR
Вектор прерывания Таймера B	TBIV	Только чтение	011Eh	Сброс с POR

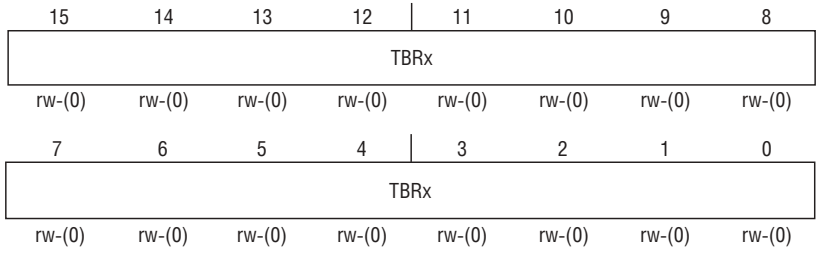
TBCTL, регистр управления таймером B



Не используется	Бит 15	Не используется
TBCLGRP	**Биты 14-13**	Группировка TBCLx 00 – Каждая защелка TBCLx загружается независимо 01 – TBCL1+TBCL2 (биты TBCCR1 CLLDx управляют обновлением) TBCL3+TBCL4 (биты TBCCR3 CLLDx управляют обновлением) TBCL5+TBCL6 (биты TBCCR5 CLLDx управляют обновлением) TBCL0 независим 10 – TBCL1+TBCL2+TBCL3 (биты TBCCR1 CLLDx управляют обновлением) TBCL4+TBCL5+TBCL6 (биты TBCCR4 CLLDx управляют обновлением) TBCL0 независим 11 – TBCL0+TBCL1+TBCL2+TBCL3+TBCL4+TBCL5+TBCL6 (биты TBCCR1 CLLDx управляют обновлением)
CNTLx	**Биты 12-11**	Длина счетчика 00 – 16-разрядный, TBR(max) = 0FFFFh 01 – 12-разрядный, TBR(max) = 0FFFh 10 – 10-разрядный, TBR(max) = 03FFh 11 – 8-разрядный, TBR(max) = 0FFh

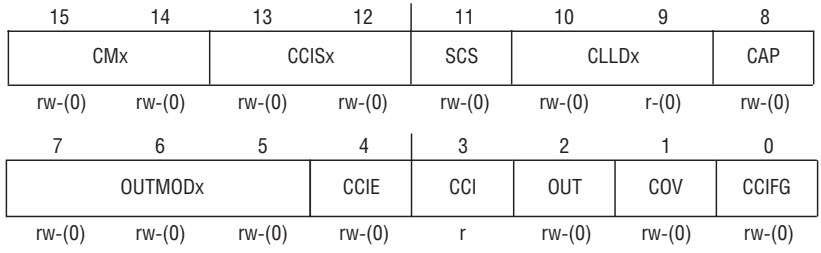
Не используется	**Бит 10**	Не используется
TBSSELx	**Биты 9-8**	Выбор источника тактирования таймера B 00 – TBCLK 01 – ACLK 10 – SMCLK 11 – инвертированный INCLK
IDx	**Биты 7-6**	Входной делитель. Эти биты позволяют выбрать коэффициент деления для входной тактовой частоты. 00 – /1 01 – /2 10 – /4 11 – /8
MCx	**Биты 5-4**	Выбор режима. Установка MCx=00h, когда таймер B не используется, позволяет уменьшить потребляемую мощность. 00 – Режим «останов»: таймер остановлен 01 – Режим «вверх»: таймер считает вверх к TBCL0 10 – Непрерывный режим: таймер считает вверх к значению, установленному TBCNTLx 11 – Режим «вверх/вниз»: таймер считает вверх к TBCL0, затем вниз к 0000h
Не используется	**Бит 3**	Не используется
TBCLR	**Бит 2**	Очистка таймера B. Установка этого бита сбрасывает TBR, IDx и выбранное направление счета. Бит TBCLR автоматически сбрасывается и всегда читается как нуль.
TBIE	**Бит 1**	Разрешение прерывания от таймера B. Этот бит разрешает запрос прерывания TBIFG. 0 – Прерывание запрещено 1 – Прерывание разрешено
TBIFG	**Бит 0**	Флаг прерывания таймера B 0 – Прерывание не ожидается 1 – Ожидается прерывание

TBR, регистр таймера B



TBRx	**Биты 15-0**	Регистр таймера B. Регистр TBR является счетчиком таймера B.

TBCCTLx, регистр управления захватом/сравнением



CMx	**Биты 15-14**	Режим захвата 00 – Нет захвата 01 – Захват по нарастающему (переднему) фронту 10 – Захват по заднему фронту 11 – Захват как по переднему, так и по заднему фронтам
CCISx	**Биты 13-12**	Выбор входа захвата/сравнения. Этими битами выбирается входной сигнал TBCCRx. См. справочное руководство конкретного устройства для выяснения подробностей подключения сигналов. 00 – CCIxA 01 – CCIxB 10 – GND 11 – VCC
SCS	**Бит 11**	Синхронизация источника захвата. Этот бит используется для синхронизациивходного сигнала захвата с тактовым сигналом. 0 – Асинхронный захват 1 – Синхронный захват
CLLDx	**Бит 10-9**	Загрузка защелки сравнения. Эти биты определяют, какое событие вызовет загрузку защелки сравнения. 00 – TBCLx загружается при записи в TBCCRx 01 – TBCLx загружается, когда TBR досчитывает к нулю 10 – TBCLx загружается, когда TBR досчитывает к нулю (непрерывный режим или режим «вверх») 11 – TBCLx загружается, когда TBR досчитывает к TBCLx
CAP	**Бит 8**	Режим захвата. 0 – Режим сравнения 1 – Режим захвата
OUTMODx	**Биты 7-5**	Режим вывода. Режимы 2, 3, 6 и 7 не используются для TBCL0, поскольку EQUx=EQU0. 000 – Значение бита OUT 001 – Установка 010 – Переключение/сброс 011 – Установка/сброс 100 – Переключение 101 – Сброс 110 – Переключение/установка 111 – Сброс/установка

CCIE	**Бит 4**	Разрешение прерывания по захвату/сравнению. Этот бит разрешает запрос прерывания от соответствующего флага CCIFG. 0 – Прерывание запрещено 1 – Прерывание разрешено
CCI	**Бит 3**	Вход захвата/сравнения. Выбранный входной сигнал может быть прочитан этим битом.
OUT	**Бит 2**	Выход. Этот бит указывает состояние вывода. Если выбран режима вывода 0, этот бит напрямую управляет состоянием выхода. 0 – Низкий уровень выхода 1 – Высокий уровень выхода
COV	**Бит 1**	Переполнение захвата. Этот бит указывает, что произошло переполнение захвата. Бит COV должен быть сброшен программно. 0 – Переполнения захвата не произошло 1 – Произошло переполнение захвата
CCIFG	**Бит 0**	Флаг прерывания захвата/сравнения 0 – Прерывание не ожидается 1 – Ожидается прерывание

TBIV, регистр вектора прерывания таймера B

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
r0	r0	r0	r0	r0	r0	r0	r0

7	6	5	4	3	2	1	0
0	0	0	0	TBIVx			0
r0	r0	r0	r0	r-(0)	r-(0)	r-(0)	r0

TBIVx	**Биты 15-0**	Значение вектора прерывания таймера B

Содержимое TBIV	Источник прерывания	Флаг прерывания	Приоритет прерывания
00h	Прерывание не ожидается	–	
02h	Захват/сравнение 1	TBCCR1 CCIFG	Высший
04h	Захват/сравнение 2	TBCCR2 CCIFG	
06h	Захват/сравнение 3*	TBCCR3 CCIFG	
08h	Захват/сравнение 4*	TBCCR4 CCIFG	
0Ah	Захват/сравнение 5*	TBCCR5 CCIFG	
0Ch	Захват/сравнение 6*	TBCCR6 CCIFG	
0Eh	Переполнение таймера	TBIFG	Низший

* Только в устройствах MSP430x14x, MSP430x16x.

MSP430x1xxFamily

Периферийный интерфейс USART, режим UART

Раздел XIII.

Texas Instruments

Периферийный интерфейс USART, режим UART

Универсальный синхронно/асинхронный приемопередающий (USART) периферийный интерфейс поддерживает два последовательных режима в одном аппаратном модуле. Этот раздел описывает работу асинхронного режима UART. USART0 реализован в устройствах MSP430x12xx, MSP430x13xx и MSP430x15x. В дополнение к USART0 в устройствах MSP430x14x и MSP430x16x реализован второй идентичный USART модуль – USART1.

13.1. Введение в USART: режим UART

В асинхронном режиме USART подключает MSP430 к внешней системе через два внешних вывода: URXD и UTXD. Режим UART выбирается при очистке бита SYNC.

Режим UART имеет следующие возможности:

- 7- или 8-разрядные данные с проверкой четности/нечетности и без контроля четности
- Независимые сдвиговые регистры передачи и приема
- Раздельные буферные регистры передачи и приема
- Передача и прием начинаются с младшего бита данных
- Встроенные коммуникационные протоколы свободной линии и адресного бита для многопроцессорных систем
- Определение в приемнике стартового фронта сигнала для автоматического пробуждения из режимов LPMx
- Программируемая скорость передачи с модуляцией для поддержки дробных величин скоростей
- Флаги статуса для обнаружения ошибок, блокировки и определения адреса
- Возможны независимые прерывания для приема и передачи

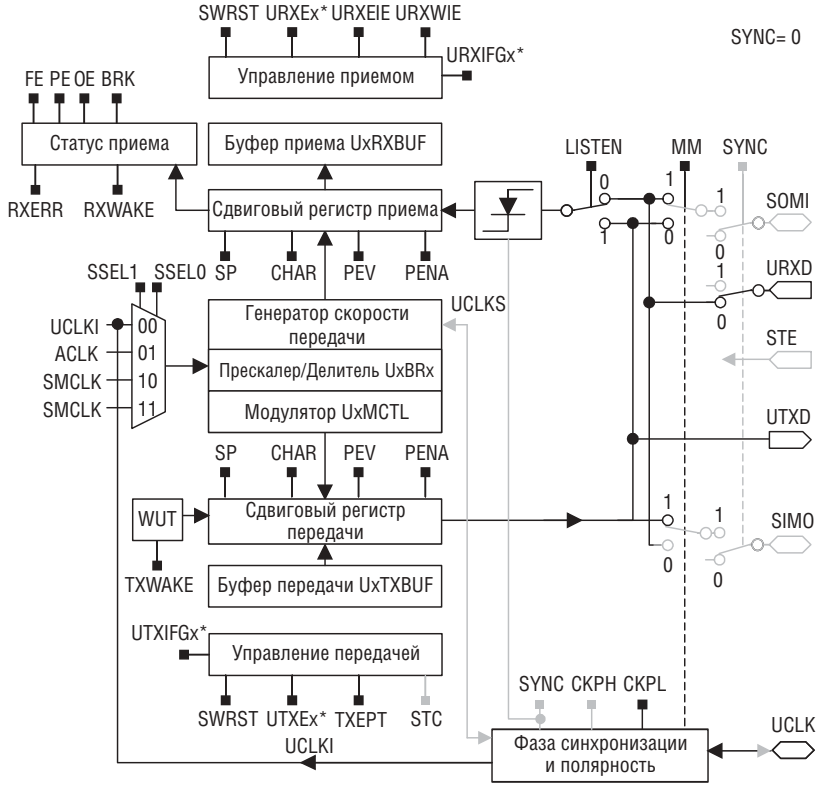
На рис. 13-1 показан USART, сконфигурированный в режиме UART.

13.2. Работа USART: режим UART

В режиме UART модуль USART передает и принимает символы на скорости, асинхронной другому устройству. Синхронизация каждого символа основана на выбранной скорости передачи USART. Для выполнения функций передачи и приема используется одинаковая скорость в бодах.

13.2.1. Инициализация и сброс USART

Модуль USART сбрасывается сигналом PUC или при установке бита SWRST. После PUC бит SWRST автоматически устанавливается, оставляя



* См. справочное руководство конкретного устройства для выяснения расположения SFR

Рис. 13-1. Блок-схема USART в режиме UART

USART в состоянии сброса. Когда бит SWRST установлен, сброшены биты URXIEx, UTXIEx, URXIFGx, RXWAKE, TXWAKE, RXERR, BRK, PE, OE, FE и установлены биты UTXIFGx и TXEPT. Флаги разрешения приема и передачи URXEx и UTXEx не изменяются битом SWRST. Очистка SWRST позволяет модулю USART функционировать. См. также раздел «Модуль USART, режим I²C» при реконфигурировании USART0 из режима I²C в режим UART.

Примечание: Инициализация и реконфигурирование модуля USART

Процесс инициализации/реконфигурирования USART необходимо выполнять так:

1) Установить SWRST (BIS.B #SWRST,&UxCTL)

2) Инициализировать все регистры USART установкой SWRST=1 (включая UxCTL)

3) Включить модуль USART через MEx SFRs (URXEx и/или UTXEx)

4) Программно очистить SWRST (BIC.B #SWRST,&UxCTL)

5) Разрешить прерывания (если необходимо) через IEx SFRs (URXIEx и/или UTXIEx)

Невыполнение этой последовательности может привести к непредсказуемому поведению USART.

13.2.2. Формат символа

Формат символа USART, показанный на рис. 13-2, содержит стартовый бит, семь или восемь битов данных, бит контроля четности, адресный бит (в адресном режиме) и один или два стоповых бит. Период битов определяется выбранным источником тактовых импульсов и настройкой регистров скорости передачи.

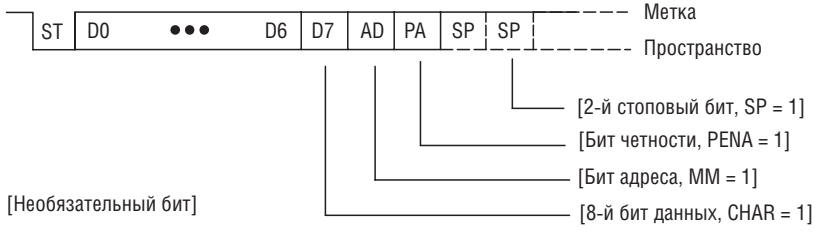


Рис. 13-2. Формат символа

13.2.3. Асинхронные коммуникационные форматы

Когда два устройства обмениваются информацией асинхронно, в качестве протокола используется формат «свободная линия». Когда связываются три или более устройств, USART поддерживает многопроцессорные коммуникационные форматы со свободной линией и формат с адресным битом.

Многопроцессорный формат со свободной линией

Когда MM=0, выбирается многопроцессорный формат со свободной линией. Блоки данных на линиях передачи или приема разделены временем простоя, как показано на рис. 13-3. Простой линии приема обнаруживается, когда приняты 10 или более непрерывных логических единиц (меток) после первого стопового бита символа. Когда для свободной линии используются два стоповых бита, второй стоповый бит принимается за первый маркерный бит периода простоя.

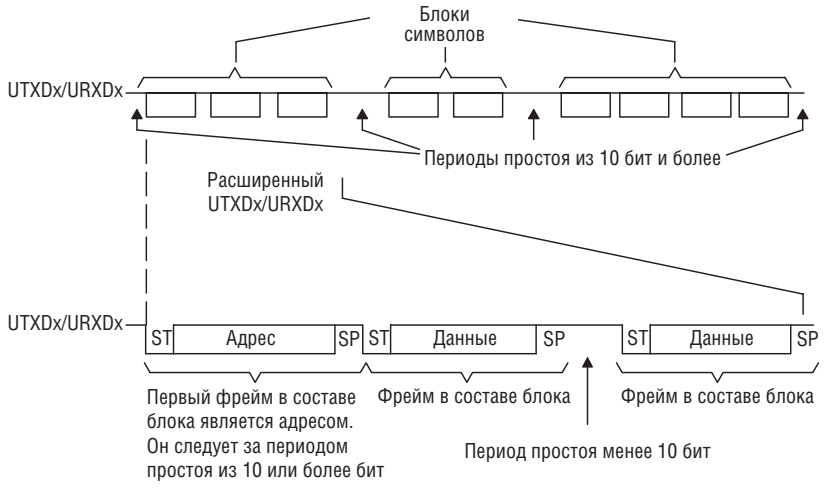


Рис. 13-3. Формат свободной линии

Первый символ, принятый после периода простоя является символом, содержащим адрес. Бит RXWAKE используется как адресный тэг для каждого фрейма. В многопроцессорном формате свободной линии этот бит установлен, когда принятый символ – адрес, помещенный в UxRXBUF.

Бит URXWIE используется для приема управляющих данных в многопроцессорном формате со свободной линией. Когда бит URXWIE установлен, все неадресные символы обрабатываются, но не перемещаются в UxRXBUF и прерывания не генерируются. Когда принят адресный символ, приемник временно активизируется для переноса символа в UxRXBUF и установки флага прерывания URXIFGx. Любой соответствующий флаг ошибки также устанавливается. Теперь пользователь может проверить корректность принятого адреса.

Если адрес принят, программное обеспечение пользователя может проверить правильность адреса и должно сбросить URXWIE для продолжения приема данных. Если URXWIE остается установленным, будут приниматься только адресные символы. Бит URXWIE не изменяется автоматически аппаратным обеспечением USART.

При передаче адреса в многопроцессорном формате со свободной линией точный период простоя для генерации идентификаторов адресного символа на UTXDx может быть сгенерирован модулем USART. Флаг временного пробуждения (WUT) – внутренний флаг с двойной буферизацией битом TXWAKE, доступным пользователю. Когда передатчик загружен из UxTXBUF, WUT также загружается из TXWAKE, сбрасывая бит TXWAKE.

Следующая процедура посылает фрейм простоя для указания, что далее следует символ адреса:

1) Устанавливается TXWAKE, что приводит к записи любого символа в UxTXBUF. UxTXBUF должен быть готов для новых данных (UTXIFGx=1).

Значение TXWAKE сдвигается в WUT и содержимое UxTXBUF сдвигается в сдвиговый регистр передачи, когда он готов для передачи новых данных. Это приводит к установке бита WUT, который препятствует нормальной передаче битов старта, данных и контроля четности, поэтому происходит передача периода простоя длительностью точно 11 бит. Когда для свободной линии используются два стоповых бита, второй стоповый бит считается первым маркерным битом периода простоя. Бит TXWAKE сбрасывается автоматически.

2) Записывается желаемый адресный символ в UxTXBUF. UxTXBUF должен быть готов для новых данных (UTXIFGx=1).

Новый символ, представляющий заданный адрес, сдвигается наружу после периода простоя, идентифицировавшего адрес на UTXDx. Необходима запись первого незначащего символа в UxTXBUF для сдвига бита TXWAKE в WUT и генерации состояния свободной линии. Эти данные отбрасываются и не появляются на UTXDx.

Многопроцессорный формат с адресным битом

Когда MM=1, выбирается многопроцессорный формат с адресным битом. Каждый обрабатываемый символ содержит дополнительный бит, используемый как указатель адреса (см. рис. 13-4). Первый символ в блоке фреймов несет установленный бит адреса, который указывает, что этот символ является адресом. Бит USART RXWAKE устанавливается, когда принятый символ является правильным адресом фрейма, помещенным UxRXBUF.

Бит URXWIE используется для приема управляющих данных в многопроцессорном формате с адресным битом. Когда бит URXWIE установлен, символы данных (бит адреса равен 0) обрабатываются приемником, но не перемещаются в UxRXBUF и прерывания не генерируются. Когда принятый символ содержит установленный адресный бит, приемник временно активизируется для переноса символа в UxRXBUF и установки флага прерывания URXIFGx. Любой соответствующий флаг ошибки также устанавливается.

Если адрес принят, программное обеспечение пользователя может должно сбросить URXWIE для продолжения приема данных. Если URXWIE остается установленным, будут приниматься только адресные символы (адресный бит равен 1). Бит URXWIE не изменяется автоматически аппаратным обеспечением USART.

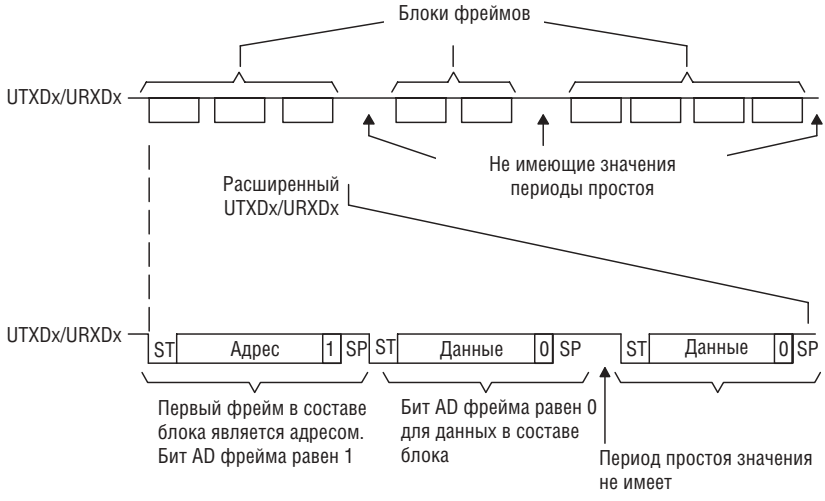


Рис. 13-4. Многопроцессорный формат с адресным битом

При передаче адреса в многопроцессорном режиме с адресным битом, адресный бит символа может изменяться путем записи бита TXWAKE. Значение бита TXWAKE загружается в адресный бит символа, перемещенного из UxTXBUF в сдвиговый регистр передачи, при этом бит TXWAKE автоматически очищается. TXWAKE не должен очищаться программно. Он очищается аппаратными средствами USART после его переноса в WUT или при установке SWRST.

Автоматическое обнаружение ошибок

Подавление импульсных помех предотвращает случайный запуск USART. Любой сигнал низкого уровня на URXDx короче времени tt (около 300 нс) будет проигнорирован. См. руководство по применению конкретного устройства для выяснения точных параметров.

Когда длительность сигнала низкого уровня на URXDx превышает tt, этот сигнал мажоритарно принимается за стартовый бит. Если стартовый бит не будет мажоритарно обнаружен, модуль USART приостанавливает прием символа и ожидает следующего периода низкого уровня на URXDx. Мажоритарный принцип также используется для предотвращения поразрядных ошибок для каждого бита символа.

Модуль USART при приеме символов автоматически обнаруживает ошибки фрейма, четности, переполнения и прерывания (разрыва). Обнаружение ошибки приводит к установке соответствующих битов FE, PE, OE и BRK. При ус-

тановке любого из этих флагов также устанавливается RXERR. Ситуации сбоев описаны в таблице 13-1.

Таблица 13-1. Ошибки приема

Ошибочное состояние	Описание
Ошибка фрейма	Ошибка фрейма (кадровой синхронизации) происходит при обнаружении стопового бита с низким уровнем. Когда используется два стоповых бита, на ошибку фрейма проверяется только первый стоповый бит. При обнаружении ошибки фрейма устанавливается бит FE.
Ошибка четности	Ошибка четности – несоответствие между числом единиц в фрейме и значением бита четности. Когда бит адреса включен в фрейм, он учитывается при определении четности. При обнаружении ошибки четности устанавливается бит PE.
Ошибка переполнения приема	Ошибка переполнения появляется в случае, когда символ загружается в UxRXBUF до прочтения предыдущего символа. Когда происходит перполнение, устанавливается бит OE.
Ошибка прерывания (разрыва)	Состояние разрыва – это период 10 или более нулевых битов, принятых на URXDx после пропущенного стопового бита. Когда обнаруживается состояние разрыва, устанавливается бит BRK. Состояние разрыва также устанавливает флаг прерывания URXIFGx.

Если обнаружена ошибка фрейма, четности или состояние разрыва и URXEIE=0, никакой символ не принимается в UxRXBUF. Когда URXEIE=1, символы принимаются в UxRXBUF и устанавливается любой соответствующий бит ошибки.

Когда любой из битов FE, PE, OE, BRK или RXERR установлен, он остается установленным до сброса программным обеспечением или до чтения UxRXBUF.

13.2.4. Разрешение приема USART

Бит разрешения приема URXEx разрешает или запрещает получение данных на URXDx, как показано на рис. 13-5. Отключение приемника USART приводит к останову операции приема, начиная с символа, следующего за получаемым в настоящий момент символом или немедленно, если прием не выполняется. Буфер принимаемых данных UxRXBUF содержит символ, перемещенный из сдвигового регистра RX после его приема.

Примечание: Повторное разрешение работы приемника (установкой URXEx): режим UART

Если приемник отключен (URXEx=0), его включение (URXEx=1) выполняется асинхронно любому потоку данных, который может присутствовать в этот

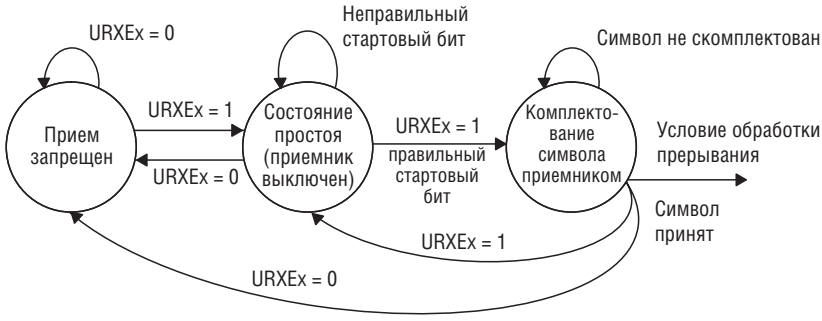


Рис. 13-5. Диаграмма состояний при разрешении приема

момент на URXDx. В этом случае синхронизация может быть выполнена путем проверки свободного состояния линии перед приемом правильного символа (см. URXWIE).

13.2.5. Разрешение передачи USART

Передатчик USART включен, когда установлен бит UTXEx. Передача инициируется путем записи данных в UxTXBUF. При этом данные перемещаются в сдвиговый регистр передачи на следующем после опустошения сдвигового регистра TX импульсе BITCLK и передача начинается. Этот процесс показан на рис. 13-6.

Если бит UTXEx сбрасывается, передача прекращается. Выполнение опе-

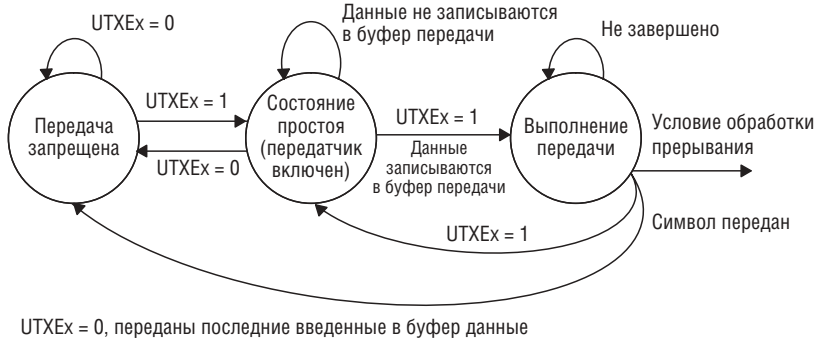


Рис. 13-6. Диаграмма состояний при разрешении передачи

раций перемещения любых данных в UxTXBUF и передачи любых данных в сдвиговый регистр передачи, начатых до очистки бита UTXEx будет продолжено, пока передача не закончится.

Если передатчик включен (UTXEx=1), данные не будет записываться в UxTXBUF, пока его готовность принимать новые данные не будет объявлена установкой UTXIFGx=1. Нарушение этого может привести к ошибочной передаче, т.к. измененные данные будет перемещены в сдвиговый регистр TX.

Рекомендуется отключать передатчик (UTXEx=0) только после завершения любой выполнявшейся передачи. На это указывает установка бита опустошения передатчика (TXEPT=1). Любые данные, записанные в UxTXBUF в тот момент, когда передатчик отключен, будут находится в буфере, но не будут помещены в сдвиговый регистр передачи или переданы. Однократная установка UTXEx вызовет немедленную загрузку в сдвиговый регистр передачи данных из буфера передачи и возобновит передачу символа.

13.2.6. Контроллер скорости передачи UART

Контроллер (генератор) скорости передачи USART может создавать стандартные скорости передачи от источников нестандартных частот. Контроллер скорости передачи использует один прескалер/делитель и модулятор, показанные на рис. 13-7. Эта комбинация позволяет получить дробные коэффициенты деления при генерации скорости передачи в бодах. Максимальная скорость передачи USART составляет одну треть источника таковой частоты USART BRCLK.

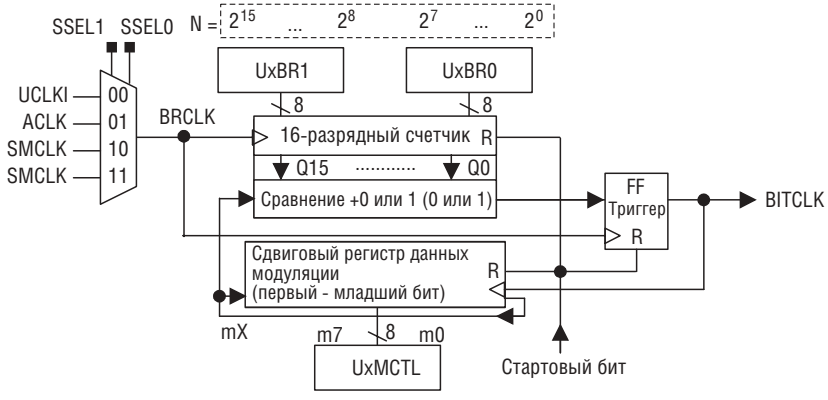


Рис. 13-7. Контроллер генератора передачи MSP430

Синхронизация каждого бита показана на рис. 13-8. Для каждого полученного бита используется мажоритарный принцип определения значения бита. Мажоритарные выборки происходят в N/2-1, N/2 и N/2+1 периоды BRCLK, где N – число импульсов BRCLKs на один импульс BITCLK.

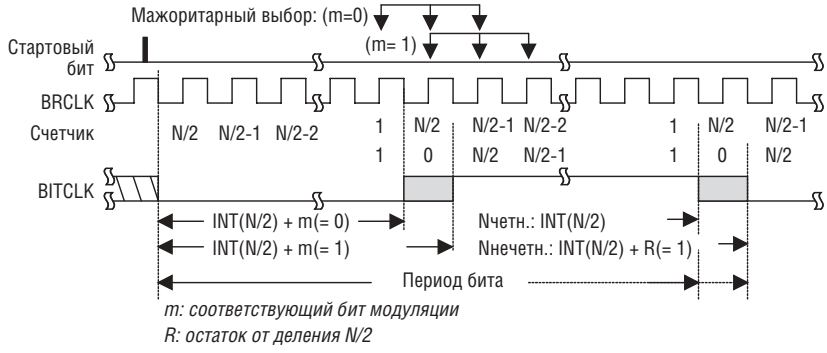


Рис. 13-8. Синхронизация скорости передачи BITCLK

Синхронизация скорости передачи бит

Первая ступень контролера скорости передачи – 16-разрядный счетчик и компаратор. В начале передачи или приема каждого бита счетчик загружается величиной INT(N/2), где N – значение, сохраненное в комбина4ции UxBR0 и UxBR1. Счетчик перезагружает INT(N/2) каждый полупериод периода бита, обеспечивая полный период бита N BRCLK. Для данного источника тактирования BRCLK, скорость передачи определяется требуемым коэффициентом деления N:

N = BRCLK / скорость передачи в бодах;

Коэффициент деления N зачастую является нецелым числом, целочисленная часть которого может быть принята прескалером/делителем. Вторая ступень генератора скорости передачи – модулятор, используемый для максимально точного учета дробной части. Коэффициент деления N в этом случае определяется так:

$$N = UxBR + \frac{1}{n} \cdot \sum_{i=0}^{n-1} m_i,$$

где:

N – получаемый коэффициент деления;

UxBR – 16-разрядное представление регистров UxBR0 и UxBR1;

i – позиция бита в фрейме;

n – общее количество битов в фрейме;

m_i – данные каждого соответствующего модуляционного бита (1 или 0).

BITCLK может подстраиваться от бита к биту с помощью модулятора для удовлетворения потребностей в синхронизации в случае, когда необходим де-

$$Скорость\_передачи = \frac{BRCKL}{N} = \frac{BRCLK}{UxBR + \frac{1}{n} \cdot \sum_{i=0}^{n-1} m_i}$$

литель нецелого числа. Синхронизация каждого бита расширяется одним тактовым циклом BRCLK, если бит модулятора mi установлен. Каждый раз при получении или передаче бита, следующий бит в регистре управления модуляцией определяет синхронизацию этого бита. Установленный модуляционный бит увеличивает коэффициент деления на единицу, в то же время очищенный бит модуляции сохраняет коэффициент деления, заданный UxBR.

Синхронизация стартового бита определяется UxBR плюс m0, следующего бита UxBR плюс m1 и так далее. Модуляционная последовательность начинается с младшего бита. Когда символ содержит более 8 бит, модуляционная последовательность вновь начинается с m0 и продолжается до окончания обработки всех битов.

Определение модуляционного значения

Определение модуляционного значения – интерактивный процесс. Использование формулы ошибки синхронизации, начиная со стартового бита, позволяет рассчитать ошибку для каждого бита с последующей установкой или сбросом соответствующего бита модуляции. Модуляционный бит устанавливается с наименьшей выбранной ошибкой и рассчитанной ошибкой следующего бита. Этот процесс продолжается до минимизации ошибок всех битов. Если фрейм содержит более 8 бит, модуляционные биты повторяются. К примеру, 9-й бит фрейма использует бит модуляции 0.

Синхронизация битов при передаче

Синхронизация каждого символа в совокупности представляет собой сумму синхронизаций отдельных разрядов. При модуляции каждого бита сокращается накапливающая поразрядная погрешность. Индивидуальную разрядную погрешность можно рассчитать так:

где:

$$Ошибка\,[\%] = \left\{ \frac{baudrate}{BRCLK} \times \left[(j+1) \times UxBR + \sum_{i=0}^{n-1} m_i \right] - (j+1) \right\} \times 100\%,$$

baudrate – желаемая скорость передачи в бодах;

BRCLK – входная частота: UCLKI, ACLK или SMCLK;

j – позиция бита – 0 для стартового бита, 1 для бита данных D0 и т.д.;

UxBR – коэффициент деления в регистрах UxBR1 и UxBR0.

Например, ошибки передачи при приведенных ниже условиях рассчитываются так:

baudrate = 2400
BRCLK = 32768 Гц (ACLK)
UxBR = 13, так как идеальный коэффициент деления равен 13.65
UxMCTL = 6Bh: m7=0, m6=1, m5=1, m4=0, m3=1, m2=0, m1=1 и m0=1.
Сначала используется младший бит UxMCTL.

$$Ошибка\_стартового\_бита\,[\%] = \left\{\frac{baudrate}{BRCLK} \times \left[(0+1)\times UxBR + 1\right] - 1\right\} \times 100\% = 2.54\%$$

$$Ошибка\_бита\_данных\_D0\,[\%] = \left\{\frac{baudrate}{BRCLK} \times \left[(1+1)\times UxBR + 2\right] - 2\right\} \times 100\% = 5.08\%$$

$$Ошибка\_бита\_данных\_D1\,[\%] = \left\{\frac{baudrate}{BRCLK} \times \left[(2+1)\times UxBR + 2\right] - 3\right\} \times 100\% = 0.29\%$$

$$Ошибка\_бита\_данных\_D2\,[\%] = \left\{\frac{baudrate}{BRCLK} \times \left[(3+1)\times UxBR + 3\right] - 4\right\} \times 100\% = 2.83\%$$

$$Ошибка\_бита\_данных\_D3\,[\%] = \left\{\frac{baudrate}{BRCLK} \times \left[(4+1)\times UxBR + 3\right] - 5\right\} \times 100\% = -1.95\%$$

$$Ошибка\_бита\_данных\_D4\,[\%] = \left\{\frac{baudrate}{BRCLK} \times \left[(5+1)\times UxBR + 4\right] - 6\right\} \times 100\% = 0.59\%$$

$$Ошибка\_бита\_данных\_D5\,[\%] = \left\{\frac{baudrate}{BRCLK} \times \left[(6+1)\times UxBR + 5\right] - 7\right\} \times 100\% = 3.13\%$$

$$Ошибка\_бита\_данных\_D6\,[\%] = \left\{\frac{baudrate}{BRCLK} \times \left[(7+1)\times UxBR + 5\right] - 8\right\} \times 100\% = -1.66\%$$

$$Ошибка\_бита\_данных\_D7\,[\%] = \left\{\frac{baudrate}{BRCLK} \times \left[(8+1)\times UxBR + 6\right] - 9\right\} \times 100\% = 0.88\%$$

$$Ошибка\_бита\_четности\,[\%] = \left\{\frac{baudrate}{BRCLK} \times \left[(9+1)\times UxBR + 7\right] - 10\right\} \times 100\% = 3.42\%$$

$$Ошибка\_стопового\_бита\_1\,[\%] = \left\{\frac{baudrate}{BRCLK} \times \left[(10+1)\times UxBR + 7\right] - 11\right\} \times 100\% = -1.37\%$$

Результаты показывают, что максимальная поразрядная ошибка была 5,08% за период BITCLK.

Синхронизация битов при приеме

Синхронизация приема состоит из двух источников ошибок. Первый – побитовая ошибка синхронизации. Второй – ошибка между появлением стартового фронта и стартовым фронтом, принятым USART. На рис. 13-9 показаны асинхронные ошибки синхронизации между данными на выводе URXDx и внутренним тактированием скорости передачи.

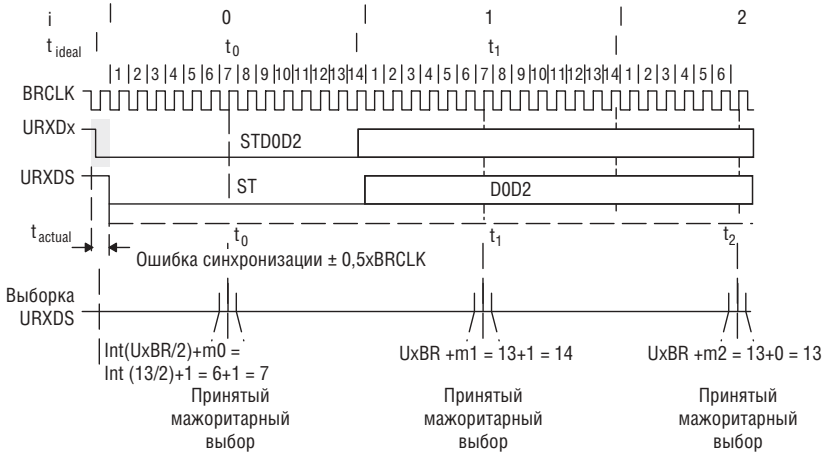


Рис. 13-9. Ошибка приема

Идеальное тактирование стартового бита $t_{ideal(0)}$ есть половина тактирования скорости передачи $t_{baud\ rate}$, поскольку бит проверяется в середине этого периода. Идеальное тактирование скорости передачи $t_{ideal(i)}$ для оставшихся битов символа есть тактирование скорости передачи $t_{baud\ rate}$. Ошибки каждого конкретного бита рассчитываются следующим образом:

$$Ошибка\ [\%] = \left\{ \frac{baudrate}{BRCLK} \times \left\{ 2 \times \left[m0 + int\left(\frac{UxBR}{2} \right) + \left(i \times UxBR + \sum_{i=1}^{n-1} m_i \right) \right] \right\} - 1 - j \right\} \times 100\%,$$

где:

baudrate – желаемая скорость передачи в бодах;

BRCLK – входная частота, которую можно выбрать из UCLK, ACLK или SMCLK;

j – позиция бита – 0 для стартового бита, 1 для бита данных D0 и т.д.;

UxBR – коэффициент деления в регистрах UxBR1 и UxBR0.

Например, ошибки приема при приведенных ниже условиях рассчитываются так:

baudrate = 2400

BRCLK = 32768 Гц (ACLK)

UxBR = 13, так как идеальный коэффициент деления равен 13.65

UxMCTL = 6Bh: m7=0, m6=1, m5=1, m4=0, m3=1, m2=0, m1=1 и m0=1.

Сначала используется младший бит UxMCTL.

$$Ошибка\_стартового\_бита\,[\%] = \left(\frac{baudrate}{BRCLK} \times \left[2 \times (1+6) + (0 \times UxBR + 0)\right] - 1 - 0\right) \times 100\% = 2.54\%$$

$$Ошибка\_бита\_данных\_D0\,[\%] = \left(\frac{baudrate}{BRCLK} \times \left[2 \times (1+6) + (1 \times UxBR + 1)\right] - 1 - 1\right) \times 100\% = 5.08\%$$

$$Ошибка\_бита\_данных\_D1\,[\%] = \left(\frac{baudrate}{BRCLK} \times \left[2 \times (1+6) \times (2 \times UxBR + 1)\right] - 1 - 2\right) \times 100\% = 0.29\%$$

$$Ошибка\_бита\_данных\_D2\,[\%] = \left(\frac{baudrate}{BRCLK} \times \left[2 \times (1+6) \times (3 \times UxBR + 2)\right] - 1 - 3\right) \times 100\% = 2.83\%$$

$$Ошибка\_бита\_данных\_D3\,[\%] = \left(\frac{baudrate}{BRCLK} \times \left[2 \times (1+6) + (4 \times UxBR + 2)\right] - 1 - 4\right) \times 100\% = -1.95\%$$

$$Ошибка\_бита\_данных\_D4\,[\%] = \left(\frac{baudrate}{BRCLK} \times \left[2 \times (1+6) + (5 \times UxBR + 3)\right] - 1 - 5\right) \times 100\% = 0.59\%$$

$$Ошибка\_бита\_данных\_D5\,[\%] = \left(\frac{baudrate}{BRCLK} \times \left[2 \times (1+6) + (6 \times UxBR + 4)\right] - 1 - 6\right) \times 100\% = 3.13\%$$

$$Ошибка\_бита\_данных\_D6\,[\%] = \left(\frac{baudrate}{BRCLK} \times \left[2 \times (1+6) + (7 \times UxBR + 4)\right] - 1 - 7\right) \times 100\% = -1.66\%$$

$$Ошибка\_бита\_данных\_D7\,[\%] = \left(\frac{baudrate}{BRCLK} \times \left[2 \times (1+6) + (8 \times UxBR + 5)\right] - 1 - 8\right) \times 100\% = 0.88\%$$

$$Ошибка\_бита\_четности\,[\%] = \left(\frac{baudrate}{BRCLK} \times \left[2 \times (1+6) + (9 \times UxBR + 6)\right] - 1 - 9\right) \times 100\% = 3.42\%$$

$$Ошибка\_стопового\_бита\_1\,[\%] = \left(\frac{baudrate}{BRCLK} \times \left[2 \times (1+6) + (10 \times UxBR + 6)\right] - 1 - 10\right) \times 100\% = -1.37\%$$

Результаты показывают, что максимальная поразрядная ошибка была 5,08% за период BITCLK.

Типовые скорости передачи и ошибки

Стандартные скорости передачи данных в бодах для UxBR и UxMCTL приведены в таблице 13-2 для часового кристалла (ACLK) на 32768 Гц и для типичного значения SMCLK 1048576 Гц.

Ошибка приема – это накопленное время в сравнении с идеальным временем загрузки сдвигового регистра в середине каждого бита. Ошибка передачи – накопленное время ошибки в сравнении с идеальным временем периода бита.

Таблица 13-2. Наиболее часто используемые величины скорости передачи, скорость передачи данных в бодах и ошибки.

Скорость передачи, бод	Деление на		A: BRCLK = 32768 Гц						B: BRCLK = 1048576 Гц				
	A:	B:	UxBR1	UxBR0	UxMCTL	Макс. ошибка TX, %	Макс. ошибка RX, %	Ошибка синхр. RX, %	UxBR1	UxBR0	UxMCTL	Макс. ошибка TX, %	Макс. ошибка RX, %
1200	27.31	873.81	0	1B	03	-4/3	-4/3	±2	03	69	FF	0/0.3	±2
2400	13.65	436.91	0	0D	6B	-6/3	-6/3	±4	01	B4	FF	0/0.3	±2
4800	6.83	218.45	0	06	6F	-9/11	-9/11	±7	0	DA	55	0/0.4	±2
9600	3.41	109.23	0	03	4A	-21/12	-21/12	±15	0	6D	03	-0.4/1	±2
19200		54.61							0	36	6B	-0.2/2	±2
38400		27.31							0	1B	03	-4/3	±2
76800		13.65							0	0D	6B	-6/3	±4
115200		9.1							0	09	08	-5/7	±7

13.2.7. Прерывания USART

USART имеет один вектор прерывания для передачи и один вектор прерывания для приема.

Функционирование прерывания USART при передаче

Флаг прерывания UTXIFGx устанавливается передатчиком для индикации готовности UxTXBUF к приему другого символа. Запрос прерывания генерируется, если установлены флаги UTXIEx и GIE. UTXIFGx автоматически сбрасывается, если запрос прерывания обслужен или если символ записан в UxTXBUF.

UTXIFGx устанавливается после PUC или когда SWRST=1. UTXIEx сбрасывается после PUC или когда SWRST=1. Это показано на рис. 13-10.

Функционирование прерывания USART при приеме

Флаг прерывания URXIFGx устанавливается каждый раз при приеме символа и его загрузки в UxRXBUF. Запрос прерывания генерируется, если также установлены флаги URXIEx и GIE. URXIFGx и URXIEx сбрасываются сигналом системного сброса PUC или когда SWRST=1. URXIFGx сбрасывается автоматически, если запрос прерывания обработан (когда URXSE=0) или когда прочитан UxRXBUF. Это показано на рис. 13-11.

URXEIE используется для разрешения или запрещения установки URXIFGx от ошибочных символов. В многопроцессорном адресном режиме URXWIE ис-

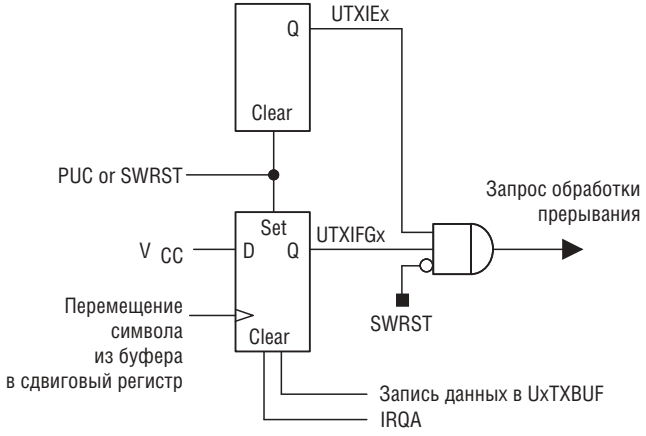


Рис. 13-10. Прерывание при передаче

пользуется для автоматического обнаружения правильных символов адреса и отклонения нежелательных символов данных.

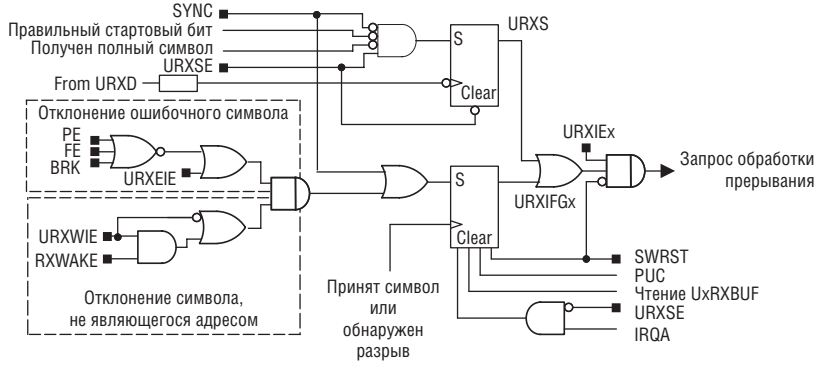


Рис. 13-11. Прерывание при приеме

Два типа символов не устанавливают URXIFGx:

- Ошибочные символы при URXEIE=0
- Символы, не являющиеся адресом при URXWIE=1

Когда URXEIE=1, состояние разрыва установит бит BRK и флаг URXIFGx.

Функционирование механизма обнаружения стартового фронта при приеме

Бит URXSE включает возможность обнаружения стартового фронта при приеме. Рекомендуется использовать возможность обнаружения стартового фронта при приеме, когда источником для BRCLK является DCO, который выключен из-за действующего режима пониженного энергопотребления. Ультрабыстрое включение DCO позволяет выполнить прием символа после обнаружения стартового фронта.

Когда URXSE, URXIEx и GIE установлены и на URXDx появился стартовый фронт, будет установлен внутренний сигнал URXS. После установки URXS будет сгенерирован запрос на прерывание при приеме, но URXIFGx не установится. Программное обеспечение пользователя в процедуре обработки прерывания приема может проверить URXIFGx для определения источника прерывания. Если URXIFGx=0, обнаружен стартовый фронт, а когда URXIFGx=1, был принят правильный символ (или разрыв).

Если процедура обработки прерывания (ISR) обнаружила, что запрос прерывания поступил от стартового фронта, пользовательское программное обеспечение переключает URXSE и должно включить источник BRCLK, вернувшись из ISR в активный режим или в режим пониженного энергопотребления, в котором источник активен. Если возврат из ISR произошел в режим пониженного энергопотребления, в котором источник BRCLK неактивен, символ не будет принят. Переключение URXSE очищает сигнал URXS и вновь активирует возможность обнаружения стартового фронта для последующих символов. См. раздел «Системный сброс, прерывания и режимы работы» для получения информации о входе и выходе из режимов пониженного энергопотребления.

Теперь активный BRCLK позволяет USART принять остаток символа. После приема полного символа и перемещения его в UxRXBUF устанавливается URXIFGx и снова запрашивается обработка прерывания. На входе ISR установка URXIFGx=1 показывает, что символ был получен. Флаг URXIFGx очищается, когда программное обеспечение пользователя читает UxRXBUF.

```
;Обработчик прерывания для условия старта фрейма
;и приема символа. BRCLK=DCO.
U0RX_Int    BIT.B #URXIFG0,&IFG2 ;Проверка URXIFGx для определения
            JNE ST_COND          ;старт или символ ?
            MOV.B &UxRXBUF,dst   ;Чтение буфера
            ...                  ;
            RETI                 ;
ST_COND     BIC.B #URXSE,&U0TCTL ;Очистка сигнала URXS
            BIS.B #URXSE,&U0TCTL ;Повторное разрешение
                                 ;определения фронта
            BIC #SCG0+SCG1,0(SP) ;Включение BRCLK = DCO
            RETI                 ;
```

Примечание: Определение разрыва при остановленном тактировании UART

Когда используется возможность определения стартового фронта при приеме символа, состояние разрыва не может быть выявлено, если источник BRCLK выключен.

Условия определения стартового фронта при приеме

Когда URXSE=1, система подавления импульсных помех предотвращает случайный запуск USART. Любой сигнал низкого уровня на URXDx короче времени tt (около 300 нС) будет проигнорирован USART и запрос прерывания не будет сгенерирован, как показано на рис. 13-12. См. руководство по применению конкретного устройства для выяснения точных параметров.

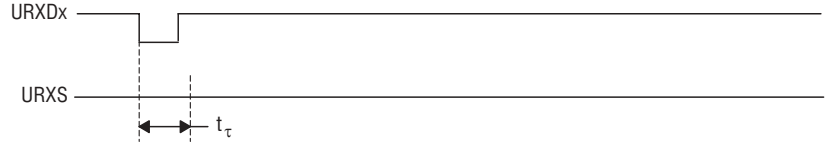


Рис. 13-12. Подавление импульсной помехи – прием в USART не начинается

Когда импульсная помеха дольше tt или на URXDx появился правильный стартовый бит, USART начинает операцию приема по мажоритарному принципу, как показано на рис. 13-13. Если стартовый бит мажоритарно не обнаружен, USART останавливает прием символа.

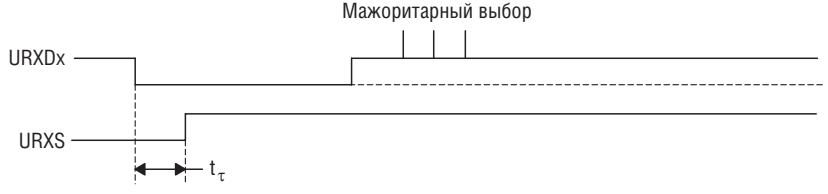


Рис. 13-13. Подавление импульсной помехи, USART активен

Если прием символа остановлен, активность BRCLK не требуется. Период простоя дольше продолжительности приема символа может использоваться программным обеспечением для индикации, что символ не был принят в ожидаемое время и программа может отключить BRCLK.

13.3. Регистры USART: режим USART

В таблице 13-3 приведен перечень регистров для всех устройств с модулем USART. Таблица 13-4 справедлива только для устройств со вторым USART модулем – USART1.

Таблица 13-3. Регистры управления и статуса USART0

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр управления USART	U0CTL	Чтение/запись	070h	001h после PUC
Регистр управления передачей	U0TCTL	Чтение/запись	071h	001h после PUC
Регистр управления приемом	U0RCTL	Чтение/запись	072h	000h после PUC
Регистр управления модуляцией	U0MCTL	Чтение/запись	073h	Не изменяется
Регистр 0 управления скоростью передачи	U0BR0	Чтение/запись	074h	Не изменяется
Регистр 1 управления скоростью передачи	U0BR1	Чтение/запись	075h	Не изменяется
Регистр буфера приема	U0RXBUF	Чтение	076h	Не изменяется
Регистр буфера передачи	U0TXBUF	Чтение/запись	077h	Не изменяется
Регистр 1 включения модуля SFR*	ME1	Чтение/запись	004h	000h после PUC
Регистр 1 разрешения прерывания SFR*	IE1	Чтение/запись	000h	000h после PUC
Регистр 1 флага прерывания SFR*	IFG1	Чтение/запись	002h	082h после PUC

* Не применимо к устройствам `12xx. См. описания регистров для выяснения расположения регистров и бит у этих устройств.

Таблица 13-4. Регистры управления и статуса USART1

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр управления USART	U1CTL	Чтение/запись	078h	001h после PUC
Регистр управления передачей	U1TCTL	Чтение/запись	079h	001h после PUC
Регистр управления приемом	U1RCTL	Чтение/запись	07Ah	000h после PUC
Регистр управления модуляцией	U1MCTL	Чтение/запись	07Bh	Не изменяется

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр 0 управления скоростью передачи	U1BR0	Чтение/запись	07Ch	Не изменяется
Регистр 1 управления скоростью передачи	U1BR1	Чтение/запись	07Dh	Не изменяется
Регистр буфера приема	U1RXBUF	Чтение	07Eh	Не изменяется
Регистр буфера передачи	U1TXBUF	Чтение/запись	07Fh	Не изменяется
Регистр 2 включения модуля SFR	ME2	Чтение/запись	005h	000h после PUC
Регистр 2 разрешения прерывания SFR	IE2	Чтение/запись	001h	000h после PUC
Регистр 2 флага прерывания SFR	IFG2	Чтение/запись	003h	000h после PUC

Примечание: Изменение битов SFR

Чтобы избежать изменения управляющих битов другими модулями, рекомендуется устанавливать или очищать биты IEx и IFGx с помощью команд `BIS.B` *или* `BIC.B` *вместо команд* `MOV.B` *или* `CLR.B`.

UxCTL, регистр управления USART

7	6	5	4	3	2	1	0
PENA	PEV	SPB	CHAR	LISTEN	SYNC	MM	SWRST
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

PENA	**Бит 7**	Включение контроля четности. 0 – Контроль четности отключен 1 – Контроль четности включен. Бит контроля четности сгенерирован (UTXDx) и ожидается (URXDx). В многопроцессорном режиме с адресным битом он учитывается при вычислении четности.
PEV	**Бит 6**	Выбор четности. PEV не используется, когда контроль четности отключен. 00 – Нечетный 01 – Четный
SPB	**Бит 5**	Выбор стопового бита. Количество передаваемых стоповых битов. Приемник всегда проверяет один стоповый бит. 0 – Один стоповый бит 1 – Два стоповых бита

CHAR	**Бит 4**	Длина символа. Можно выбрать 7-ми или 8-ми разрядный символ. 0 – 7-разрядные данные 1 – 8-разрядные данные
LISTEN	**Бит 3**	Включение прослушивания. Бит LISTEN включает режим обратной петли. 0 – Отключен 1 – Включен. UTXDx внутренне подключается назад к приемнику.
SYNC	**Бит 2**	Включение синхронного режима 0 – Режим UART 1 – Режим SPI
MM	**Бит 1**	Выбор многопроцессорного режима 0 – Многопроцессорный протокол со свободной линией 1 – Многопроцессорный протокол с адресным битом
SWRST	**Бит 0**	Разрешение программного сброса 0 – Отключен. Сброс USART не задействован 1 – Разрешен. Логика USART удерживается в состоянии сброса

UxTCTL, регистр управления передачей USART

7	6	5	4	3	2	1	0
Не используется	CKPL	SSELx		URXSE	TXWAKE	Не используется	TXEPT
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

Не используется	**Бит 7**	Не используется.
CKPL	**Бит 6**	Выбор полярности тактового сигнала. 0 – UCLKI = UCLK 1 – UCLKI = инвертированный UCLK
SSELx	**Биты 5-4**	Выбор источника. Эти биты выбирают источник тактирования для BRCLK 00 – UCLKI 01 – ACLK 10 – SMCLK 11 – SMCLK
URXSE	**Бит 3**	UART принимает стартовый фронт. Бит включает возможность приема UART`ом стартового фронта. 0 – Отключено 1 – Включено
TXWAKE	**Бит 2**	«Пробуждение» передатчика 0 – Следующий передаваемый фрейм - данные 1 – Следующий передаваемый фрейм – адрес
Не используется	**Бит 1**	Не используется

TXEPT	**Бит 0**	Флаг опустошения передатчика 0 – UART передает данные и/или данные ожидают в UxTXBUF 1 – Сдвиговый регистр передатчика и UxTXBUF пусты или SWRST=1

UxRCTL, регистр управления приемом USART

7	6	5	4	3	2	1	0
FE	PE	OE	BRK	URXEIE	URXWIE	RXWAKE	RXERR
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

FE	**Бит 7**	Флаг ошибки фрейма 0 – Нет ошибки 1 – Символ принят со стоповым битом низкого уровня
PE	**Бит 6**	Флаг ошибки контроля четности. Когда PENA=0, PE читается как 0. 0 – Нет ошибки 1 – Символ принят с ошибкой четности
OE	**Бит 5**	Флаг ошибки переполнения. Этот бит устанавливается, когда символ перемещен в UxRXBUF до завершения чтения предыдущего символа. 0 – Нет ошибки 1 – Произошла ошибка переполнения
BRK	**Бит 4**	Флаг обнаружения разрыва 0 – Нет состояния разрыва 1 – Появилось состояние разрыва
URXEIE	**Бит 3**	Разрешение прерывания при приеме ошибочного символа 0 – Ошибочный символ отклоняется, а URXIFGx не устанавливается 1 – Принятый ошибочный символ устанавливает URXIFGx
URXWIE	**Бит 2**	Запуск приема с разрешением прерывания. Этот бит разрешает URXIFGx быть установленным, когда принят адресный символ. Если URXEIE=0, символ адреса не будет устанавливать URXIFGx, если он принят с ошибками. 0 – Все принятые символы устанавливают URXIFGx 1 – Только принятые адресные символы устанавливают URXIFGx
RXWAKE	**Бит 1**	Флаг «пробуждения» при приеме 0 – Принятый символ - данные 1 – Принятый символ – адрес
RXERR	**Бит 0**	Флаг ошибки приема. Этот бит показывает, что символ был принят с ошибкой (ошибками). Если RXERR=1, один или более флагов ошибок (FE, PE, OE, BRK) также устанавливаются. RXERR очищается, когда UxRXBUF прочитан. 0 – Ошибки приема не обнаружены 1 – Обнаружена ошибка приема

UxBR0, регистр 0 управления скоростью передачи USART

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
rw	rw	rw	rw	rw	rw	rw	rw

UxBR1, регистр 1 управления скоростью передачи USART

7	6	5	4	3	2	1	0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
rw	rw	rw	rw	rw	rw	rw	rw

UxBRx		Правильный диапазон управления скоростью передачи лежит в пределах 3 ≤ UxBR < 0FFFFh, где UxBR = {UxBR1+UxBR0}. Если UxBR < 3, произойдет непредсказуемая синхронизация приема и передачи.

UxMCTL, регистр управления модуляцией USART

7	6	5	4	3	2	1	0
m7	m6	m5	m4	m3	m2	m1	m0
rw	rw	rw	rw	rw	rw	rw	rw

UxMCTLx	**Биты 7-0**	Биты модуляции. Эти биты выбирают модуляцию для BRCLK.

UxRXBUF, регистр буфера приема USART

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
r	r	r	r	r	r	r	r

UxRXBUFx	**Биты 7-0**	Буфер принятых данных доступен пользователю и содержит последний принятый из сдвигового регистра приема символ. Чтение UxRXBUF сбрасывает биты ошибок приема, бит RXWAKE и URXIFGx. В режиме 7-разрядных данных, UxRXBUF выравнивается по младшему разряду (LSB), а старший разряд (MSB) всегда сбрасывается.

UxTXBUF, регистр буфера передачи USART

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
rw	rw	rw	rw	rw	rw	rw	rw

UxTXBUFx	**Биты 7-0**	Буфер передаваемых данных доступен пользователю и хранит данные, ожидающие перемещения в сдвиговый регистр передачи и отправку на UTXDx. Запись в буфер данных передачи очищает UTXIFGx. Старший разряд UxTXBUF не используется для 7-разрядных данных и поэтому сбрасывается.

ME1, регистр 1 включения модуля

7	6	5	4	3	2	1	0
UTXE0*	URXE0*						
rw-0	rw-0						

UTXE0*	**Бит 7**	Разрешение передачи USART0. Этот бит включает передатчик USART0. 0 – Модуль выключен 1 – Модуль включен
URXE0*	**Бит 6**	Разрешение приема USART0. Этот бит включает приемник USART0. 0 – Модуль выключен 1 – Модуль включен
	Биты 5-0	Эти биты могут быть использованы другими модулями. См. справочные данные конкретного устройства.

* Не используется в устройствах MSP430x12xx. См. ME2 для битов включения модуля USART0 MSP430x12xx.

ME2, регистр 2 включения модуля

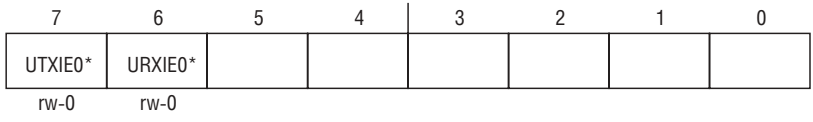
7	6	5	4	3	2	1	0
		UTXE1	URXE1			UTXE0**	URXE0**
		rw-0	rw-0			rw-0	rw-0

	Биты 7-6	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.

UTXE1	**Бит 5**	Включение передачи USART1. Этот бит включает передатчик USART1. 0 – Модуль выключен 1 – Модуль включен
URXE1	**Бит 4**	Включение приема USART1. Этот бит включает приемник USART1. 0 – Модуль выключен 1 – Модуль включен
	Биты 3-2	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
UTXE0**	**Бит 1**	Включение передачи USART0. Этот бит включает передатчик USART0. 0 – Модуль выключен 1 – Модуль включен
URXE0**	**Бит 0**	Включение приема USART0. Этот бит включает приемник USART0. 0 – Модуль выключен 1 – Модуль включен

** Только в устройствах MSP430x12xx

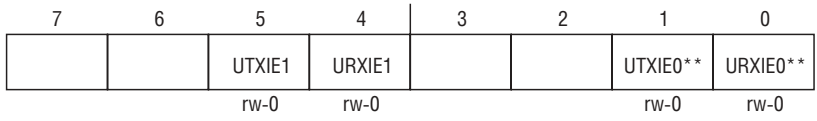
IE1, регистр 1 разрешения прерываний



UTXIE0*	**Бит 7**	Разрешение прерывания при передаче USART0. Этот бит разрешает прерывание UTXIFG0. 0 – Прерывание не разрешено 1 – Прерывание разрешено
URXIE0*	**Бит 6**	Разрешение прерывания при приеме USART0. Этот бит разрешает прерывание URXIFG0. 0 – Прерывание не разрешено 1 – Прерывание разрешено
	Биты 5-0	Эти биты могут быть использованы другими модулями. См. справочные данные конкретного устройства.

* Не используется в устройствах MSP430x12xx. См. IE2 для битов разрешения прерывания USART0 MSP430x12xx.

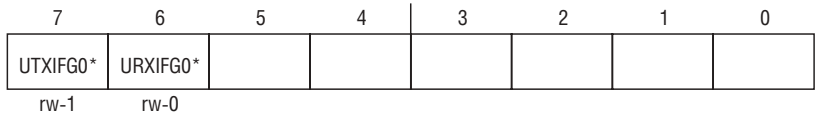
IE2, регистр 2 разрешения прерывания



	Биты 7-6	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
UTXIE1	**Бит 5**	Разрешение прерывания при передаче USART1. Этот бит разрешает прерывание UTXIFG1. 0 – Прерывание не разрешено 1 – Прерывание разрешено
URXIE1	**Бит 4**	Разрешение прерывания при приеме USART1. Этот бит разрешает прерывание URXIFG1. 0 – Прерывание не разрешено 1 – Прерывание разрешено
	Биты 3-2	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
UTXIE0**	**Бит 1**	Разрешение прерывания при передаче USART0. Этот бит разрешает прерывание UTXIFG0. 0 – Прерывание не разрешено 1 – Прерывание разрешено
URXIE0**	**Бит 0**	Разрешение прерывания при приеме USART0. Этот бит разрешает прерывание URXIFG0. 0 – Прерывание не разрешено 1 – Прерывание разрешено

** Только в устройствах MSP430x12xx

IFG1, регистр 1 флагов прерываний



UTXIFG0*	**Бит 7**	Флаг прерывания при передаче USART0. UTXIFG0 устанавливается, когда U0TXBUF пуст. 0 – Прерывание не ожидается 1 – Прерывание ожидается

URXIFG0*	**Бит 6**	Флаг прерывания при приеме USART0. URXIFG0 устанавливается, когда в U0TXBUF принят полный символ. 0 – Прерывание не ожидается 1 – Прерывание ожидается
	Биты 5-0	Эти биты могут быть использованы другими модулями. См. справочные данные конкретного устройства.

* Не используется в устройствах MSP430x12xx. См. IFG2 для битов флагов прерывания USART0 MSP430x12xx.

IFG2, регистр 2 флагов прерываний

7	6	5	4	3	2	1	0
		UTXIFG1	URXIFG1			UTXIFG0**	URXIFG0**
		rw-1	rw-0			rw-1	rw-0

	Биты 7-6	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
UTXIFG1	**Бит 5**	Флаг прерывания при передаче USART1. UTXIFG1 устанавливается, когда U1TXBUF пуст. 0 – Прерывание не ожидается 1 – Прерывание ожидается
URXIFG1	**Бит 4**	Флаг прерывания при приеме USART1. URXIFG1 устанавливается, когда в U1RXBUF принят полный символ. 0 – Прерывание не ожидается 1 – Прерывание ожидается
	Биты 3-2	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
UTXIFG0**	**Бит 1**	Флаг прерывания при передаче USART0. UTXIFG0 устанавливается, когда U0TXBUF пуст. 0 – Прерывание не ожидается 1 – Прерывание ожидается
URXIFG0**	**Бит 0**	Флаг прерывания при приеме USART0. URXIFG0 устанавливается, когда в U0RXBUF принят полный символ. 0 – Прерывание не ожидается 1 – Прерывание ожидается

** Только в устройствах MSP430x12xx

MSP430x1xxFamily

Периферийный интерфейс USART, режим SPI

Раздел XIV.

Texas Instruments

Периферийный интерфейс USART, режим SPI

Универсальный синхронно/асинхронный приемопередающий (USART) периферийный интерфейс поддерживает два последовательных режима в одном аппаратном модуле. Этот раздел описывает работу синхронного периферийного интерфейса или режима SPI. USART0 реализован в устройствах MSP430x12xx, MSP430x13xx и MSP430x15x. В дополнение к USART0, в устройствах MSP430x14x и MSP430x16x реализован второй идентичный USART модуль – USART1.

14.1. Введение в USART: режим SPI

В синхронном режиме USART подключает MSP430 к внешней системе через три или четыре вывода: SIMO, SOMI, UCLK и STE. Режим SPI выбирается, когда бит SYNC установлен, а бит I^2C очищен.

Режим SPI имеет следующие возможности:

- 7-ми или 8-разрядные данные
- Работа SPI с 3-мя или 4-мя выводами
- Режимы ведущий или ведомый
- Независимые сдвиговые регистры передачи и приема
- Раздельные буферные регистры передачи и приема
- Выбираемая полярность UCLK и управление фазой
- Программируемая частота UCLK в режиме ведущего
- Независимая возможность прерывания для приема и передачи

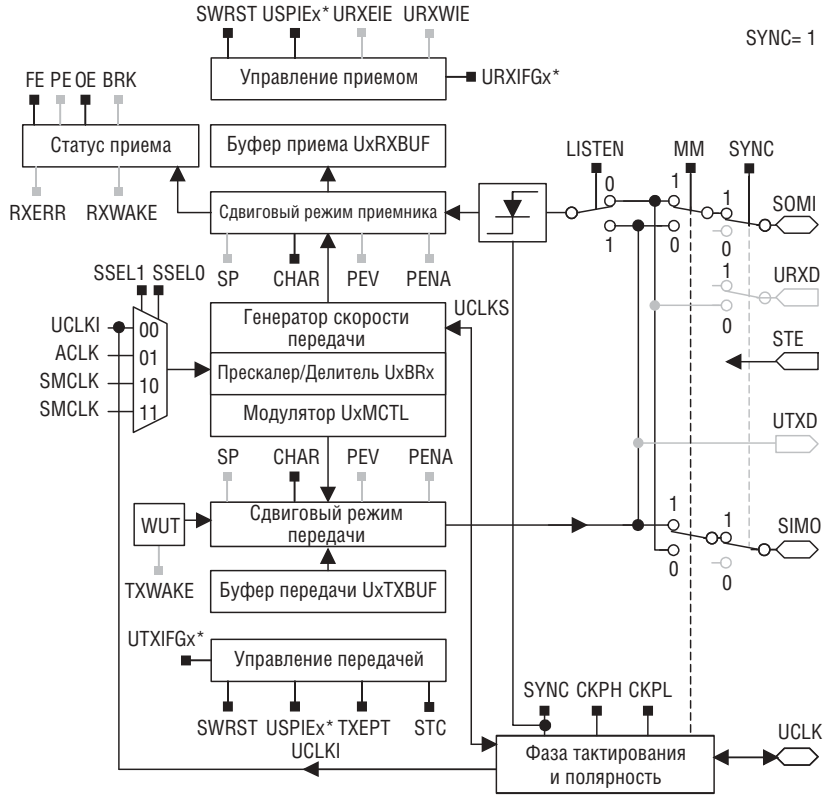
На рис. 14-1 показан USART, сконфигурированный в режиме SPI.

14.2. Функционирование USART: режим SPI

В синхронном режиме последовательные данные передаются и принимаются множеством устройств с использованием общего тактирования, обеспечиваемого ведущим. Дополнительный вывод STE, управляемый ведущим, необходим для разрешения приема и передачи данных устройством.

Три или четыре сигнала используются для обмена данными через SPI:

- SIMO Вход ведомого, выход ведущего
  Режим ведущего: SIMO – линия вывода данных
  Режим ведомого: SIMO – линия ввода данных
- SOMI Выход ведомого, вход ведущего
  Режим ведущего: SOMI – линия ввода данных
  Режим ведомого: SOMI – линия вывода данных



* См. справочное руководство конкретного устройства для определения расположения SPR

Рис. 14-1. Блок-схема USART в режиме SPI

- UCLK Тактирование USART SPI
  Режим ведущего: UCLK – выход
  Режим ведомого: UCLK – вход
- STE Разрешение передачи ведомого. Используется в 4-выводном режиме, когда на одной шине может быть много ведущих. Не применяется в 3-выводном режиме.

4-х выводной режим ведущего:
Когда STE имеет высокий уровень, SIMO и UCLK работают как обычно.
Когда STE имеет низкий уровень, SIMO и UCLK устанавливаются на направление ввода.

4-х выводной режим ведомого:

Когда STE имеет высокий уровень, функционирование RX/TX ведомого отключено и SOMI принудительно устанавливается на направление ввода.

Когда STE имеет низкий уровень, функционирование RX/TX ведомого разрешено и SOMI работает как обычно.

14.2.1. Инициализация USART и сброс

USART сбрасывается сигналом PUC или битом SWRST. После PUC бит SWRST автоматически устанавливается, оставляя USART в состоянии сброса. Когда он установлен, бит SWRST сбрасывает биты URXIEx, UTXIEx, URXIFGx, OE, FE и устанавливает флаг UTXIFGx. Бит USPIEx не изменяется битом SWRST. Для работы USART необходимо очистить SWRST. См. также раздел «Модуль USART, режим I^2C» для USART0, когда он реконфигурируется из режима I^2C в режим SPI.

Примечание: Инициализация и реконфигурирование модуля USART

Для инициализации/реконфигурирования USART необходим следующий процесс:

1) Установить SWRST (BIS.B #SWRST,&UxCTL)

2) Инициализировать все регистры USART установкой SWRST=1 (включая UxCTL)

3) Включить модуль USART через MEx SFRs (USPIEx)

4) Программно очистить SWRST (BIC.B #SWRST,&UxCTL)

5) Разрешить прерывания (если необходимо) через IEx SFRs (URXIEx и/или UTXIEx)

Невыполнение этих действий может привести к непредсказуемому поведению USART.

14.2.2. Режим ведущего

На рис. 14-2 показан USART в качестве мастера в обеих 3-х и 4-х выводных конфигурациях. USART инициализирует передачу данных, когда данные перемещаются в буфер передачи данных UxTXBUF. Данные UxTXBUF перемещаются в сдвиговый регистр TX, когда сдвиговый регистр TX пуст, инициируя передачу данных на SIMO, начиная со старшего разряда. Данные на SOMI сдвигаются в сдвиговый регистр приема по противоположному тактовому фронту, начиная со старшего разряда. Когда символ принят, принятые данные перемещены из сдвигового регистра RX в буфер принятых данных UxRXBUF, флаг прерывания приема URXIFGx установлен, указывая завершение операции RX/TX.

Установка флага прерывания передачи UTXIFGx указывает, что данные перемещены из UxTXBUF в сдвиговый регистр TX и UxTXBUF готов для поступления новых данных. Это не указывает на завершение операции RX/TX.

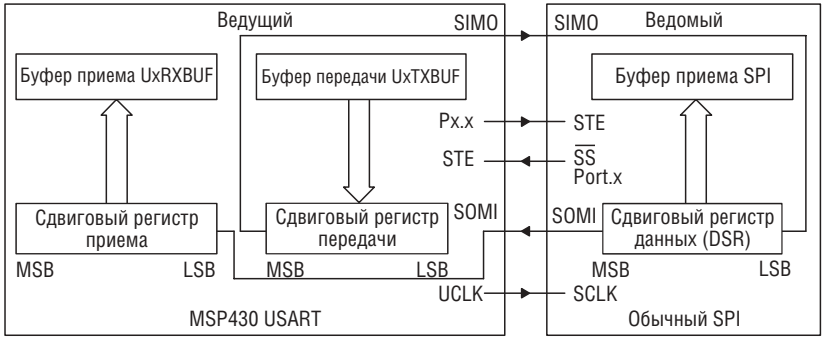


Рис. 14-2. USART – ведущий, внешнее устройство – ведомое

Чтобы принимать данные в USART в режиме ведущего, данные должны быть записаны в UxTXBUF, поскольку операции приема и передачи выполняются одновременно.

4-х выводной режим ведущего SPI

В 4-х выводном режиме ведущего STE используется для предотвращения конфликтов с другим ведущим. Ведущий функционирует нормально, когда STE имеет высокий уровень. Когда у STE низкий уровень:

- SIMO и UCLK установлены на ввод и более не управляют шиной
- Установлен бит ошибки FE, что указывает на нарушение целостности связи, которое будет обработано пользователем

Сигнал STE низкого уровня не сбрасывает модуль USART. Входной сигнал STE не используется в 3-х выводном режиме ведущего.

14.2.3. Режим ведомого

На рис. 14-3 показан USART в качестве ведомого в обеих 3-х и 4-х выводных конфигурациях. UCLK используется как вход для тактирования SPI и должен управляться внешним ведущим. Скорость передачи данных определяется этим тактовым сигналом и не зависит от внутреннего генератора скорости передачи. Данные записываются в UxTXBUF и перемещаются в сдвиговый регистр TX до старта передачи UCLK на SOMI. Данные на SIMO сдвигаются в сдвиговый регистр приема по противоположному фронту UCLK и перемещаются в UxRXBUF, когда принято заданное количество бит. Когда данные перемещаются из сдвигового регистра RX в UxRXBUF, устанавливается флаг прерывания URXIFGx, указывая, что данные были приняты. Бит ошибки переполнения OE устанавливается, когда предыдущие принятые данные не были прочитаны из UxRXBUF до перемещения новых данных в UxRXBUF.

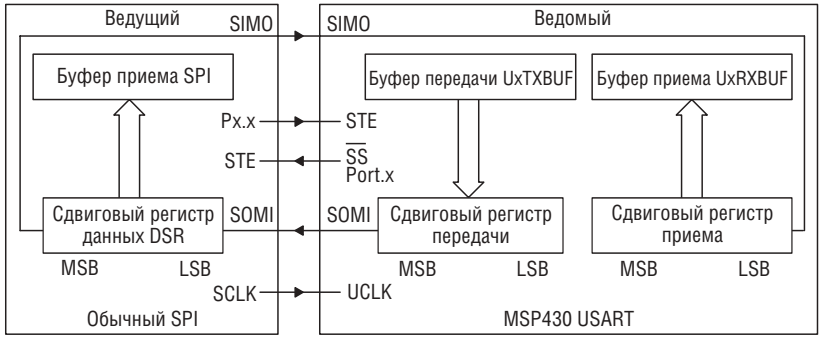


Рис. 14-3. USART – ведомый, внешнее устройство – ведущее

4-х выводной режим ведомого SPI

В 4-х выводном режиме ведущего STE используется ведомым для разрешения операций передачи и приема и управляется ведущим SPI. Когда STE имеет низкий уровень, ведомый работает нормально. Когда у STE высокий уровень:

- Любая выполняющаяся операция приема на SIMO останавливается
- SOMI устанавливается на направление ввода

Высокий уровень сигнала STE не сбрасывает модуль USART. Входной сигнал STE не используется в 3-х выводном режиме ведомого.

14.2.4. Включение SPI

Бит включения USPIEx передачи/приема SPI включает или отключает USART в режиме SPI. Когда USPIEx=0, USART останавливает работу после завершения текущей передачи или немедленно, если действий не выполнялось.

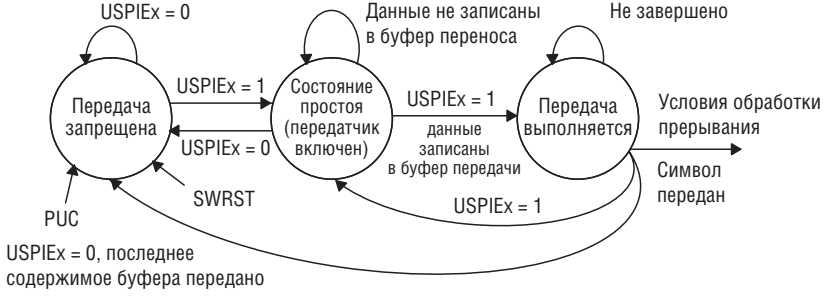


Рис. 14-4. Разрешение передачи в режиме ведущего

Сигнал PUC или установка бита SWRST отключают USART немедленно, при этом любая выполняющаяся передача прерывается.

Разрешение передачи

Когда USPIEx=0, любая последующая запись в UxTXBUF не приводит к передаче. Данные, записанные в UxTXBUF начнут передаваться, когда USPIEx=1 и активен источник BRCLK. На рис. 14-4 и рис. 14-5 показаны диаграммы состояний при разрешении передачи.

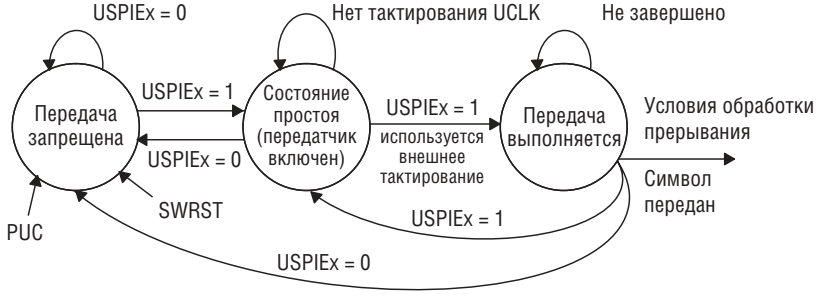


Рис. 14-5. Диаграмма состояний разрешения передачи ведомого

Разрешение приема

Диаграммы состояний разрешения приема SPI показаны на рис. 14-6 и рис. 14-7. Когда USPIEx=0, UCLK не сдвигает данные в сдвиговый регистр RX.

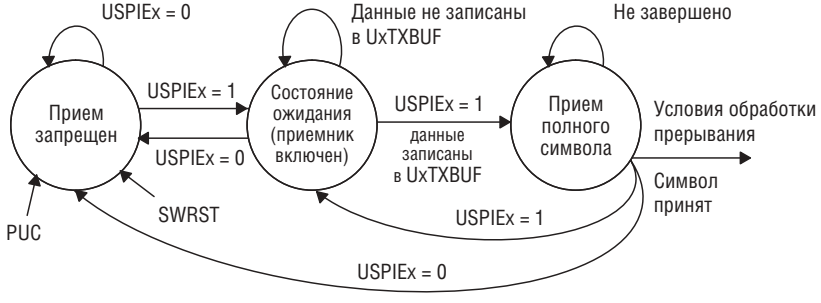


Рис. 14-6. Диаграмма состояний разрешения приема в режиме ведущего SPI

14.2.5. Управление последовательным тактированием

Сигнал UCLK на шине SPI обеспечивается ведущим. Когда MM=1, BITCLK обеспечивается генератором скорости передачи USART на выводе UCLK, как

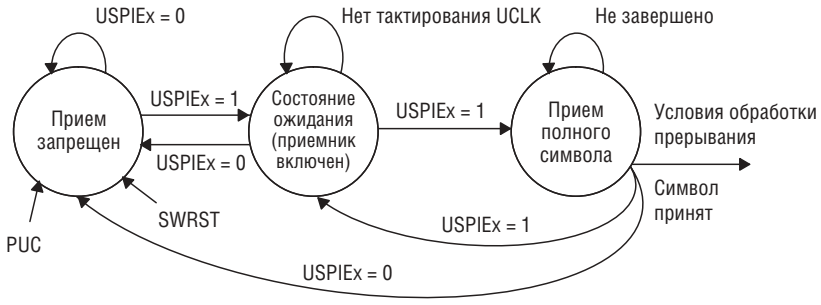


Рис. 14-7. Диаграмма состояний разрешения приема ведомым SPI

показано на рис. 14-8. Когда MM=0, тактирование USART на выводе UCLK обеспечивается ведущим, генератор скорости передачи не используется, а значения битов SSELx не учитываются. Приемник и передатчик SPI работают параллельно и используют одинаковый источник тактирования для передачи данных.

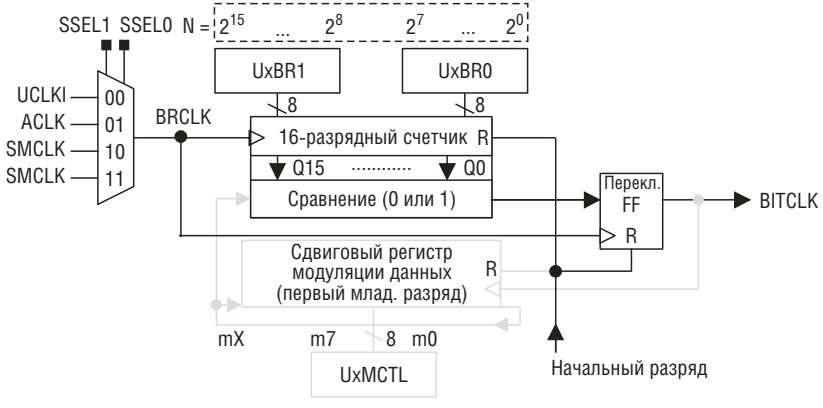


Рис. 14-8. Генератор скорости передачи SPI

16-разрядное значение UxBR0+UxBR1 представляет собой коэффициент деления источника тактирования USART – BRCLK. Максимальная скорость передачи, генерируемая в режиме ведущего равна BRCLK/2. Модулятор в генераторе скорости передачи USART не используется в режиме SPI, рекомендуется устанавливать его значение равным 000h. Частота UCLK определяется так:

Скорость передачи = BRCLK/UxBR, где UxBR = [UxBR1, UxBR0]

Полярность и фаза последовательного тактирования

Полярность и фаза UCLK раздельно конфигурируются через управляющие биты CKPL и CKPH модуля USART. Синхронизация для каждого случая показана на рис. 14-9.

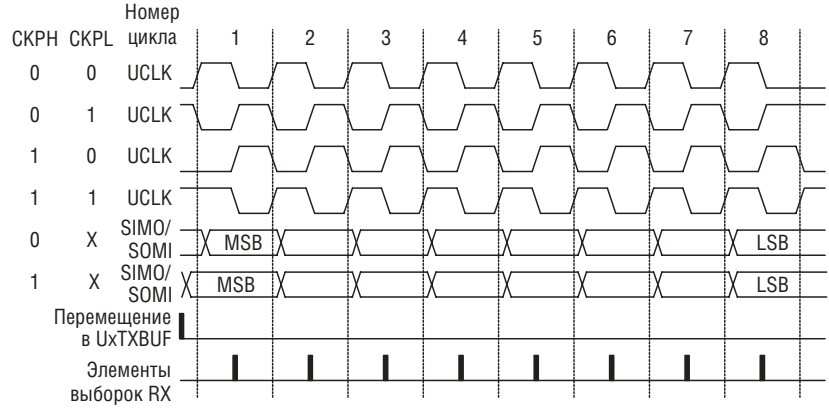


Рис. 14-9. Синхронизация USART SPI

14.2.6. Прерывания SPI

SPI имеет один вектор прерывания для передачи и один вектор прерывания для приема.

Работа прерывания SPI при передаче

Флаг прерывания UTXIFGx устанавливается передатчиком для указания, что UxTXBUF готов к приему другого символа. Запрос прерывания генерируется, если также установлены флаги UTXIEx и GIE. UTXIFGx автоматически сбрасывается, если запрос прерывания обработан или если символ записан в UxTXBUF.

UTXIFGx устанавливается после PUC или когда SWRST=1. UTXIEx сбрасывается после PUC или когда SWRST=1. Это показано на рис. 14-10.

Примечание: запись в UxTXBUF в режиме SPI

Запись данных в UxTXBUF, когда UTXIFGx=0 и USPIEx=1 может привести к ошибочной передаче данных.

Работа прерывания SPI при приеме

Флаг прерывания URXIFGx устанавливается каждый раз, когда символ принят и загружен в UxRXBUF, как показано на рис. 14-11 и 14-12. Запрос пре-

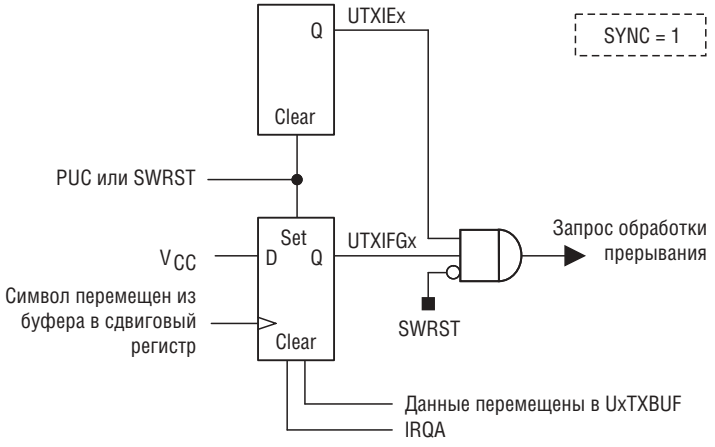


Рис. 14-10. Функционирование прерывания при передаче

рывания генерируется, если также установлены флаги URXIEx и GIE. URXIFGx и URXIEx сбрасываются сигналом системного сброса PUC или когда SWRST=1. URXIFGx сбрасывается автоматически, если ожидаемое прерывание обработано или когда UxRXBUF прочитан. Это показано на рис. 14-11 и рис. 14-12.

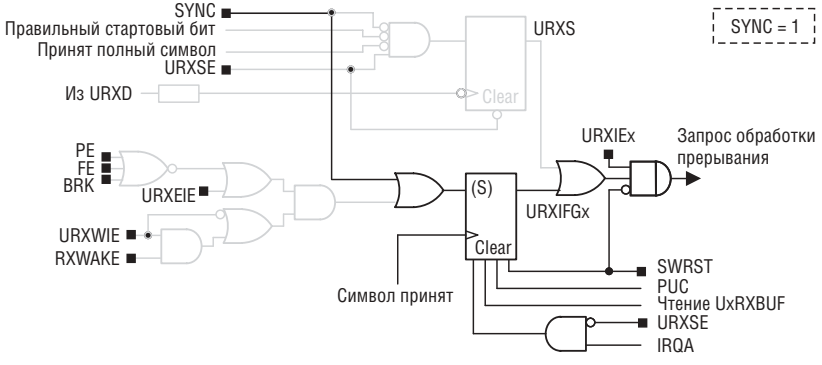


Рис. 14-11. Функционирование прерывания при приеме

14.3. Регистры USART: режим SPI

Регистры USART, показанные в таблице 14-1 и таблице 14-2, структурированы побайтно, поэтому доступ к ним необходимо выполнять с помощью команд работы с байтами.

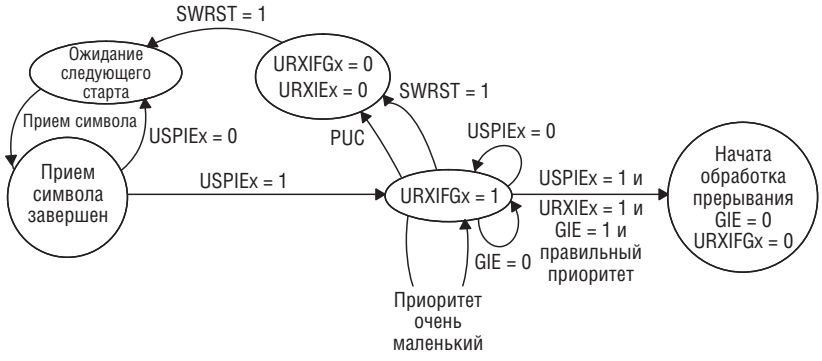


Рис. 14-12. Диаграмма состояний прерывания при приеме

Таблица 14-1. Регистры управления и статуса USART0

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр управления USART	U0CTL	Чтение/запись	070h	001h после PUC
Регистр управления передачей	U0TCTL	Чтение/запись	071h	001h после PUC
Регистр управления приемом	U0RCTL	Чтение/запись	072h	000h после PUC
Регистр управления модуляцией	U0MCTL	Чтение/запись	073h	Не изменяется
Регистр 0 управления скоростью передачи	U0BR0	Чтение/запись	074h	Не изменяется
Регистр 1 управления скоростью передачи	U0BR1	Чтение/запись	075h	Не изменяется
Регистр буфера приема	U0RXBUF	Чтение	076h	Не изменяется
Регистр буфера передачи	U0TXBUF	Чтение/запись	077h	Не изменяется
Регистр 1 включения модуля SFR*	ME1	Чтение/запись	004h	000h после PUC
Регистр 1 разрешения прерывания SFR*	IE1	Чтение/запись	000h	000h после PUC
Регистр 1 флага прерывания SFR*	IFG1	Чтение/запись	002h	082h после PUC

* Не применимо к устройствам MSP430x12xx. См. описания регистров для выяснения расположения регистров и бит в этих устройствах.

Таблица 14-2. Регистры управления и статуса USART1

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр управления USART	U1CTL	Чтение/запись	078h	001h после PUC
Регистр управления передачей	U1TCTL	Чтение/запись	079h	001h после PUC
Регистр управления приемом	U1RCTL	Чтение/запись	07Ah	000h после PUC
Регистр управления модуляцией	U1MCTL	Чтение/запись	07Bh	Не изменяется
Регистр 0 управления скоростью передачи	U1BR0	Чтение/запись	07Ch	Не изменяется
Регистр 1 управления скоростью передачи	U1BR1	Чтение/запись	07Dh	Не изменяется
Регистр буфера приема	U1RXBUF	Чтение	07Eh	Не изменяется
Регистр буфера передачи	U1TXBUF	Чтение/запись	07Fh	Не изменяется
Регистр 2 включения модуля SFR	ME2	Чтение/запись	005h	000h после PUC
Регистр 2 разрешения прерывания SFR	IE2	Чтение/запись	001h	000h после PUC
Регистр 2 флага прерывания SFR	IFG2	Чтение/запись	003h	020h после PUC

Примечание: Изменение битов SFR

Чтобы избежать изменения управляющих битов другими модулями, рекомендуется устанавливать или очищать биты IEx и IFGx с помощью команд `BIS.B` *или* `BIC.B` *вместо команд* `MOV.B` *или* `CLR.B`.

UxCTL, регистр управления USART

7	6	5	4	3	2	1	0
Не используется	Не используется	I^2C*	CHAR	LISTEN	SYNC	MM	SWRST
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

Не используется	**Биты 7-6**	Не используются
I2C*	**Бит 5**	Включение режима I^2C. Этот бит позволяет выбрать режим I^2C или SPI, когда SYNC=1. 0 – Режим SPI 1 – Режим I2C
CHAR	**Бит 4**	Длина символа 0 – 7-разрядные данные 1 – 8-разрядные данные
LISTEN	**Бит 3**	Включение прослушивания. Бит LISTEN включает режим обратной петли. 0 – Отключен 1 – Включен. Сигнал передачи внутренне подключается назад к приемнику.

SYNC	**Бит 2**	Включение синхронного режима 0 – Режим UART 1 – Режим SPI
MM	**Бит 1**	Режим ведущего 0 – USART ведомый 1 – USART ведущий
SWRST	**Бит 0**	Включение программного сброса 0 – Отключен. Сброс USART исключен из работы 1 – Разрешен. Логика USART удерживается в состоянии сброса

*Применимо к USART0 только в устройствах MSP430x15x и MSP430x16x.

UxTCTL, регистр управления передачей USART

7	6	5	4	3	2	1	0
CKPH	CKPL	SSELx		Не используется	Не используется	STC	TXEPT
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

CKPH	**Бит 7**	Выбор фазы тактирования. Управляет фазой UCLK. 0 – Обычная схема тактирования UCLK 1 – Сигнал UCLK отстает на один полупериод
CKPL	**Бит 6**	Выбор полярности тактового сигнала. 0 – Неактивный уровень низкий; вывод данных происходит по нарастающему фронту UCLK; входные данные защелкиваются по спаду UCLK. 1 – Неактивный уровень высокий; вывод данных происходит по спаду UCLK; входные данные защелкиваются по нарастающему фронту UCLK.
SSELx	**Биты 5-4**	Выбор источника. Эти биты выбирают источник тактирования для BRCLK 00 – Внешний UCLK (действует только в режиме ведомого) 01 – ACLK (справедливо только для режима ведущего) 10 – SMCLK (справедливо только для режима ведущего) 11 – SMCLK (справедливо только для режима ведущего)
Не используется	**Бит 3**	Не используется
Не используется	**Бит 2**	Не используется
STC	**Бит 1**	Управление передачей ведомого. 0 – 4-х выводной режим SPI: STE включен 1 – 3-х выводной режим SPI: STE выключен
TXEPT	**Бит 0**	Флаг опустошения передатчика. Флаг TXEPT не используется в режиме ведомого. 0 – Передача активна и/или в UxTXBUF находятся данные 1 – UxTXBUF и сдвиговый регистр TX пусты

UxRCTL, регистр управления приемом USART

7	6	5	4	3	2	1	0
FE	Не используется	OE	Не используется	Не используется	Не используется	Не используется	Не используется
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

FE	**Бит 7**	Флаг ошибки фрэйма. Этот бит указывает на конфликт шины, когда MM=1 и STC=0. FE не используется в режиме ведомого. 0 – Конфликт не обнаружен 1 – На STC появился отрицательный фронт, указывая на конфликт при обращении к шине
Не используется	**Бит 6**	Не используется
OE	**Бит 5**	Флаг ошибки переполнения. Этот бит устанавливается, когда символ перемещен в UxRXBUF до завершения чтения предыдущего символа. OE автоматически сбрасывается, когда UxRXBUF прочитан, когда SWRST=1, а также может быть сброшен программно. 0 – Нет ошибки 1 – Произошла ошибка переполнения
Не используется	**Бит 4**	Не используется
Не используется	**Бит 3**	Не используется
Не используется	**Бит 2**	Не используется
Не используется	**Бит 1**	Не используется
Не используется	**Бит 0**	Не используется

UxBR0, регистр 0 управления скоростью передачи USART

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
rw	rw	rw	rw	rw	rw	rw	rw

UxBR1, регистр 1 управления скоростью передачи USART

7	6	5	4	3	2	1	0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
rw	rw	rw	rw	rw	rw	rw	rw

UxBRx		Генератор скорости передачи использует содержимое {UxBR1+UxBR0} для установки скорости передачи. Возможна некорректная работа SPI в случае установки UxBR < 2.

UxMCTL, регистр управления модуляцией USART

7	6	5	4	3	2	1	0
m7	m6	m5	m4	m3	m2	m1	m0
rw	rw	rw	rw	rw	rw	rw	rw

UxMCTLx	**Биты 7-0**	Регистр управления модуляцией не используется в режиме SPI и должен быть установлен на 000h.

UxRXBUF, регистр буфера приема USART

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
r	r	r	r	r	r	r	r

UxRXBUFx	**Биты 7-0**	Буфер принятых данных доступен пользователю и содержит последний принятый из сдвигового регистра приема символ. Чтение UxRXBUF сбрасывает бит OE и флаг URXIFGx. В режиме 7-разрядных данных, UxRXBUF выравнивается по младшему разряду, а старший разряд всегда сбрасывается.

UxTXBUF, регистр буфера передачи USART

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
rw	rw	rw	rw	rw	rw	rw	rw

UxTXBUFx	**Биты 7-0**	Буфер передаваемых данных доступен пользователю и содержит текущие передаваемые данные. Когда используется длина символа в 7 бит, данные необходимо выровнять по старшему разряду перед перемещением их в UxTXBUF. Данные передаются начиная со старшего разряда. Запись в UxTXBUF очищает UTXIFGx.

ME1, регистр 1 включения модуля

7	6	5	4	3	2	1	0
	USPIE0*						
	rw-0						

	Бит 7	Этот бит может быть использован другими модулями. См. справочные данные конкретного устройства.
USPIE0*	**Бит 6**	Включение USART0 SPI. Этот бит включает режим SPI для USART0. 0 – Модуль выключен 1 – Модуль включен
	Биты 5-0	Эти биты могут быть использованы другими модулями. См. справочные данные конкретного устройства.

* Не используется в устройствах MSP430x12xx. См. ME2 для битов включения модуля USART0 MSP430x12xx.

ME2, регистр 2 включения модуля

7	6	5	4	3	2	1	0
			USPIE1				USPIE0**
			rw-0				rw-0

	Биты 7-5	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
USPIE1	**Бит 4**	Включение USART1 SPI. Этот бит включает режим SPI для USART1. 0 – Модуль выключен 1 – Модуль включен
	Биты 3-1	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
USPIE0**	**Бит 0**	Включение USART0 SPI. Этот бит включает режим SPI для USART0. 0 – Модуль выключен 1 – Модуль включен

** Только устройства MSP430x12xx

IE1, регистр 1 разрешения прерываний

7	6	5	4	3	2	1	0
UTXIE0*	URXIE0*						
rw-0	rw-0						

UTXIE0*	**Бит 7**	Разрешение прерывания при передаче USART0. Этот бит разрешает прерывание UTXIFG0. 0 – Прерывание не разрешено 1 – Прерывание разрешено
URXIE0*	**Бит 6**	Разрешение прерывания при приеме USART0. Этот бит разрешает прерывание URXIFG0. 0 – Прерывание не разрешено 1 – Прерывание разрешено
	Биты 5-0	Эти биты могут быть использованы другими модулями. См. справочные данные конкретного устройства.

* Не используется в устройствах MSP430x12xx. См. IE2 для битов разрешения прерывания USART0 MSP430x12xx.

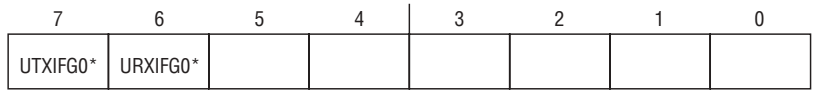
IE2, регистр 2 разрешения прерывания

7	6	5	4	3	2	1	0
		UTXIE1	URXIE1			UTXIE0**	URXIE0**
		rw-0	rw-0			rw-0	rw-0

	Биты 7-6	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
UTXIE1	**Бит 5**	Разрешение прерывания при передаче USART1. Этот бит разрешает прерывание UTXIFG1. 0 – Прерывание не разрешено 1 – Прерывание разрешено
URXIE1	**Бит 4**	Разрешение прерывания при приеме USART1. Этот бит разрешает прерывание URXIFG1. 0 – Прерывание не разрешено 1 – Прерывание разрешено
	Биты 3-2	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
UTXIE0**	**Бит 1**	Разрешение прерывания при передаче USART0. Этот бит разрешает прерывание UTXIFG0. 0 – Прерывание не разрешено 1 – Прерывание разрешено
URXIE0**	**Бит 0**	Разрешение прерывания при приеме USART0. Этот бит разрешает прерывание URXIFG0. 0 – Прерывание не разрешено 1 – Прерывание разрешено

** Только устройства MSP430x12xx

IFG1, регистр 1 флагов прерываний



UTXIFG0*	**Бит 7**	Флаг прерывания при передаче USART0. UTXIFG0 устанавливается, когда U0TXBUF пуст. 0 – Прерывание не ожидается 1 – Прерывание ожидается
URXIFG0*	**Бит 6**	Флаг прерывания при приеме USART0. URXIFG0 устанавливается, когда в U0RXBUF принят полный символ. 0 – Прерывание не ожидается 1 – Прерывание ожидается
	Биты 5-0	Эти биты могут быть использованы другими модулями. См. справочные данные конкретного устройства.

* Не используется в устройствах MSP430x12xx. См. IFG2 для битов флагов прерывания USART0 MSP430x12xx.

IFG2, регистр 2 флагов прерываний

	Биты 7-6	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
UTXIFG1	**Бит 5**	Флаг прерывания при передаче USART1. UTXIFG1 устанавливается, когда U1TXBUF пуст. 0 – Прерывание не ожидается 1 – Прерывание ожидается
URXIFG1	**Бит 4**	Флаг прерывания при приеме USART1. URXIFG1 устанавливается, когда в U1RXBUF принят полный символ. 0 – Прерывание не ожидается 1 – Прерывание ожидается
	Биты 3-2	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
UTXIFG0**	**Бит 1**	Флаг прерывания при передаче USART0. UTXIFG0 устанавливается, когда U0TXBUF пуст. 0 – Прерывание не ожидается 1 – Прерывание ожидается
URXIFG0**	**Бит 0**	Флаг прерывания при приеме USART0. URXIFG0 устанавливается, когда в U0RXBUF принят полный символ. 0 – Прерывание не ожидается 1 – Прерывание ожидается

** Только устройства MSP430x12xx

MSP430x1xxFamily

Периферийный интерфейс USART, режим I²C

Раздел XV.

Texas Instruments

Периферийный интерфейс USART, режим I²C

Универсальный синхронно/асинхронный приемопередающий (USART) периферийный интерфейс поддерживает связь по I²C в модулях USART0. Этот раздел описывает режим I²C. Режим I²C реализован в устройствах MSP430x15x и MSP430x16x.

15.1. Введение в модуль I²C

Модуль управления взаимообменом между интегральными схемами (I²C) обеспечивает интерфейс между MSP430 и I²C-совместимыми устройствами через последовательную двухпроводную шину I²C. Внешние компоненты, присоединенные к шине I²C последовательно передают и/или принимают последовательные данные в/из USART через 2-х проводной I²C-интерфейс.

Модуль I²C имеет следующие возможности:

– Соответствует спецификации I²C v2.1 Philips Semiconductor
• Формат передачи байт/слово
• 7-разрядный и 10-разрядный режимы адресации устройств
• Общий вызов
• Старт/рестарт/стоп
• Режим приемника передатчик/ведущий со многими ведущими
• Режим передатчика приемник/ведомый со многими ведущими
• Комбинированный режим ведущего передача/прием и прием/передача
• Поддержка стандартного режима до 100 кбод и быстрого режима до 400 кбод
– Встроенный FIFO для буферирования чтения и записи
– Программируемая генерация тактовых импульсов
– 16-разрядный доступ к данным для увеличения пропускной способности шины
– Автоматический подсчет количества байт данных
– Разработан для работы при пониженном энергопотреблении
– Определение условия СТАРТ ведомого приемника для автоматического выхода из режимов LPMx
– Расширенные возможности прерываний
– Реализован только в USART0

На рис. 15-1 показана блок-схема модуля I²C.

15.2. Функционирование модуля I²C

Модуль I²C поддерживает любые ведущие или ведомые устройства, совместимые с I²C. На рис. 15-2 показан пример шины I²C. Каждое устройство I²C

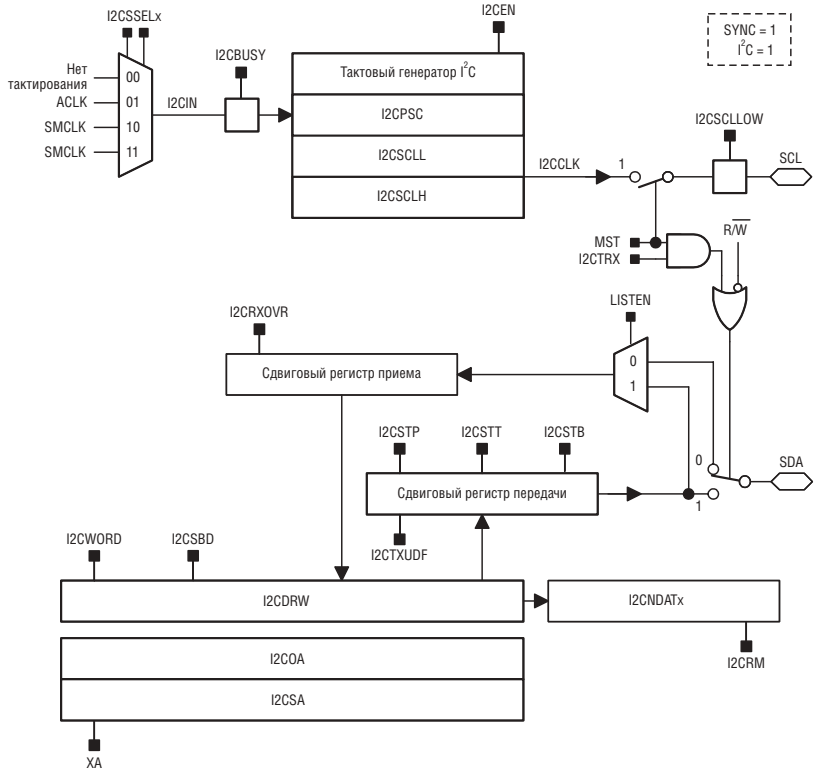


Рис. 15-1. Блок-схема USART: режим I²C

обладает уникальным адресом и может работать и как передатчик и как приемник. Устройство, подключенное к шине I²C, во время передачи данных может рассматриваться как ведущее или ведомое. Ведущий инициирует передачу данных и генерирует тактовый сигнал SCL. Любое устройство, адресованное ведущим, рассматривается как ведомый.

При обмене данными на I²C используется вывод последовательных данных (SDA) и вывод последовательного тактирования (SCK). Оба вывода SDA и SCL являются двунаправленными и должны подключаться к положительному источнику питания с использованием нагрузочного резистора.

Примечание: Уровни SDA и SCL

Уровни напряжения на выводах SDA и SCL не должны быть выше уровня V_{CC} MSP430.

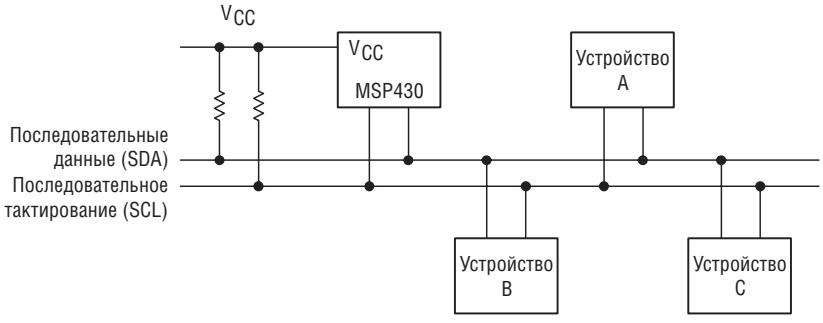


Рис. 15-2. Схема подключений на шине I²C

15.2.1. Инициализация модуля I²C

Модуль I²C является частью периферии USART. Индивидуальный бит устанавливает, когда USART0 используется в режиме I²C, а не в режимах SPI или UART. Значение по умолчанию в регистре U0CTL устанавливает режим UART. Для работы с I²C необходимо установить биты SYNC и I²C. После инициализации модуля, модуль I²C готов для выполнения приема или передачи. Установка I2CEN разрешает работу модуля I²C.

Конфигурирование и переконфигурирование модуля I²C всегда должно выполняться только при I2CEN=0, во избежание непредсказуемого поведения. Установка I2CEN=0 вызывает следующие эффекты:

- связь по I²C останавливается;
- линии SDA и SCL переводятся в высокоимпедансное состояние;
- в регистре I2CTCTL очищаются биты 3-0, а биты 7-4 не изменяются;
- очищаются регистры I2CDCTL и I2CDR;
- сдвиговые регистры приема и передачи очищаются;
- регистры U0CTL, I2CNDAT, I2CPSC, I2CSCLL, I2CSCLH не изменяются;
- регистры I2COA, I2CSA, I2CIE, I2CIFG и I2CIV не изменяются.

Когда USART переконфигурируется из режима I²C в режимы UART или SPI, первыми должны быть очищены биты I²C, SYNC и I2CEN, затем должен быть установлен SWRST и только потом можно начать процедуру инициализации UART или SPI. Нарушение этой последовательности может привести к непредсказуемым результатам.

Примечание: конфигурирование модуля USART для работы в режиме I²C после сброса:

Необходим следующий процесс конфигурации I²C:

1) Выбрать режим I²C установкой SWRST=1	`(BIS.B #I2C + SUNC,&U0CTL)`
2) Выключить модуль I²C	`(BIC.B #I2CEN,&U0CTL)`
3) Сконфигурировать модуль I²C при I2CEN=0;	
4) Программно установить I2CEN	`(BIS.B #I2CEN,&U0CTL)`

Нарушение последовательности выполнения этого процесса может привести к непредсказуемому поведению USART.

Примечание: переконфигурирование модуля USART для работы в режиме UART или SPI:

Когда модуль USART переконфигурируется для работы в режимах UART или SPI из режима I²C, необходимо выполнить следующие действия:

1) Очистить биты I²C, SYNC и I2CEN	`(CLR.B &U0CTL);`
2) Установить SWRST	`(MOV.B #SWRST,&U0CTL);`
3) Продолжить процедуру инициализации UART или SPI;	

Нарушение последовательности выполнения этого процесса может привести к непредсказуемому поведению USART.

15.5.2. Последовательные данные I²C

При передаче каждого бита ведущим устройством генерируется один тактовый импульс. Модуль I²C работает с данными, организованными побайтно. Сначала перемещается старший значащий разряд, как показано на рис. 15-3.

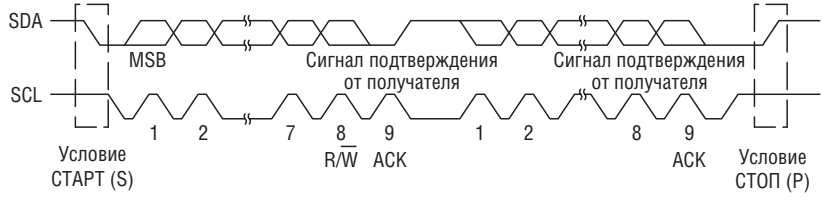


Рис. 15-3. Передача данных модулем I²C

Первый после условия «СТАРТ» байт состоит из 7-разрядного адреса ведомого и бита R/nonW. Когда R/nonW=0, ведущий передает данные ведомому. Когда R/nonW=1, ведущее устройство принимает данные от ведомого. Бит ACK посылается приемником после каждого байта на 9-ом такте SCL.

Условия «СТАРТ» и «СТОП», показанные на рис. 15-3, генерируются ведущим устройством. Под условием «СТАРТ» понимается переход с высокого уровня линии SDA на низкий при высоком уровне на SCL. Под условием «СТОП» понимается переход с низкого уровня на линии SDA на высокий при высоком уровне на SCL. Бит занятости I2CBB устанавливается после условия «СТАРТ» и сбрасывается после условия «СТОП».

Данные на SDA должны быть неизменны в течение периода высокого уровня SCL, как показано на рис. 15-4. Высокий и низкий уровень SDA может изменяться только тогда, когда SCL имеет низкий уровень, в противном случае будет сгенерировано условие «СТАРТ» или «СТОП».

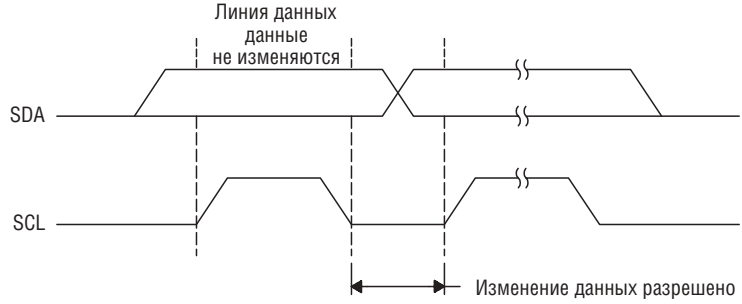


Рис. 15-4. Передача бита на шине I²C

15.2.3. Режимы адресации I²C

Модуль I²C поддерживает 7-разрядный и 10-разрядный режимы адресации.

7-разрядная адресация

В 7-разрядном адресном формате, показанном на рис. 15-5, первый байт – это 7-разрядный адрес ведомого и бит R/nonW. Бит ACK посылается приемником после каждого байта.

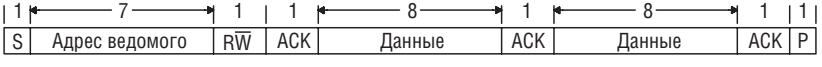


Рис. 15-5. 7-разрядный формат адресации модуля I²C

10-разрядная адресация

В 10-разрядном адресном формате, показанном на рис. 15-6, первый байт содержит 11110b плюс два старших бита 10-разрядного адреса ведомого и бит R/nonW. Бит ACK посылается приемником после каждого байта. Следующий

байт состоит из 8 битов 10-разрядного адреса ведомого, за которым следует бит ACK и 8-разрядные данные.

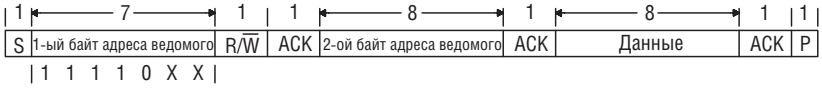


Рис. 15-6. 10-разрядный формат адресации модуля I²C

Повторные условия «СТАРТ»

Направление потока данных на SDA может быть изменено ведущим без первоначального останова переноса, что приведет к повторению условия «СТАРТ». Это вызовет «РЕСТАРТ». После выполнения «РЕСТАРТА» адрес ведомого отправляется снова, но уже с новым направлением данных, заданным битом R/nonW. Условие «РЕСТАРТ» показано на рис. 15-7.

Рис. 15-7. Формат адресации модуля I²C с повторным условием «СТАРТ»

15.2.4. Режимы работы модуля I²C

Модуль I²C работает в режимах ведущий передатчик, ведущий приемник, ведомый передатчик или ведомый приемник.

Режим ведущего

В режиме ведущего выполнение передачи и приема управляется с помощью битов I2CRM, I2CSTT и I2CSTP, как описано в таблице 15-1. Режимы ведущего передатчика и ведущего приемника показаны на рис. 15-8 и рис. 15-9. Когда после приема или передачи байта необходимо вмешательство ЦПУ, линия SCL удерживается в состоянии низкого уровня.

Таблица 15-1. Функционирование ведущего

I2CRM	I2CSTP	I2CSTT	Состояние или активность шины
X	0	0	Модуль I²C находится в режиме ведущего, но свободен. Условия «СТАРТ» и «СТОП» не генерируются.
0	0	1	Активность инициируется установкой I2CSTT. I2CNDAT используется для определения длины передачи. Условие «СТОП» автоматически не генерируется после перемещения байт, количество которых задано в I2CNDAT. Программное обеспечение должно установить I2CSTP для генерации условия «СТОП» в конце передачи. Это используется для условия «РЕСТАРТ».

Таблица 15-1. (Окончание)

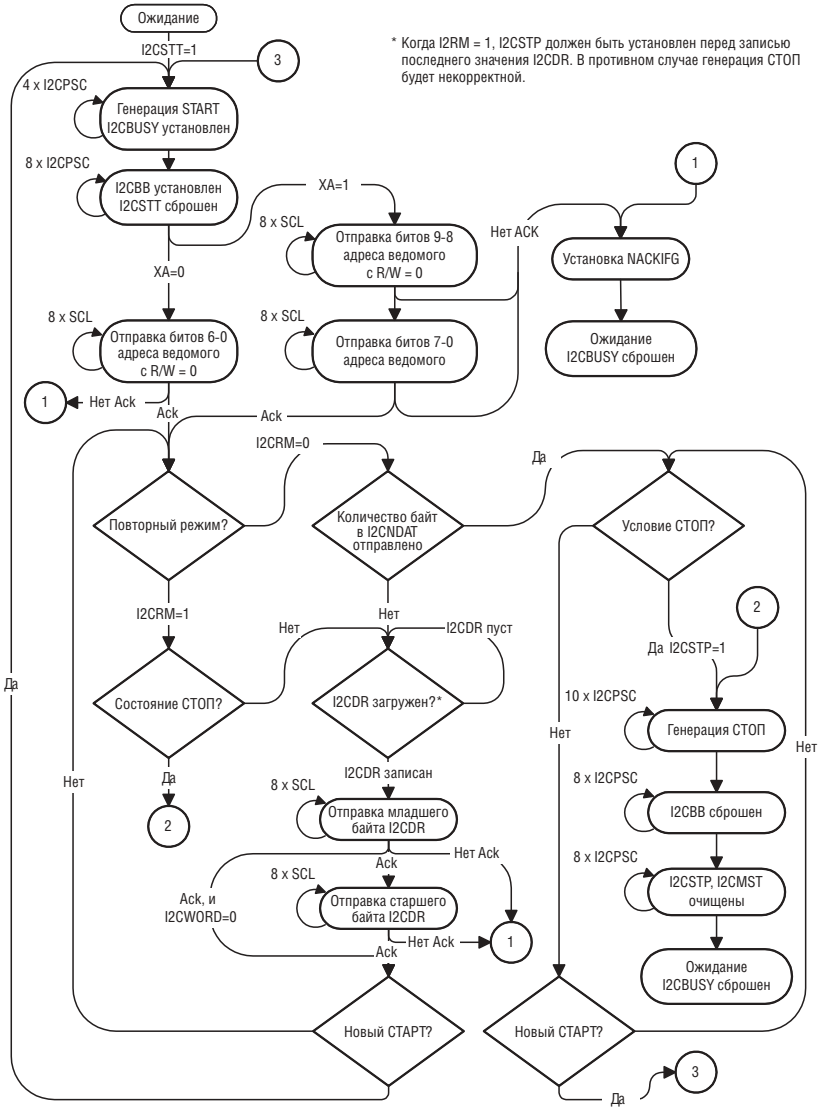
I2CRM	I2CSTP	I2CSTT	Состояние или активность шины
0	1	1	I2CNDAT используется для установки длины передачи. Установка I2CSTT инициирует активность. Условие «СТОП» автоматически генерируется после передачи количества байт, заданного в I2CNDAT.
1	0	1	I2CNDAT не используется для установки длины передачи. Длиной передачи должно управлять программное обеспечение. Установка бита I2CSTT инициирует активность. Для инициирования условия «СТОП» или останова активности программное обеспечение должно установить бит I2CSTP. Этот режим используется, если необходимо передать более 255 байт.
0	1	0	Установка бита I2CSTP генерирует условие «СТОП» на шине после отправки количества байт, заданного в I2CNDAT или немедленно, если уже было передано то количество байт, которое заданное в I2CNDAT.
1	1	0	Установка бита I2CSTP генерирует условие «СТОП» на шине после завершения текущей передачи или немедленно, если текущая передача не выполняется.
1	1	1	Зарезервировано, шина неактивна.

Арбитраж

Если два или более передатчика одновременно начинают передачу на шине, запускается процедура арбитража. На рис. 15-10 поясняется процедура арбитража между двумя устройствами. Процедура арбитража использует данные, представленные на SDA конкурирующими передатчиками. Первый ведущий передатчик, генерирующий логическую единицу, отвергается противостоящим ведущим, генерирующим логический сигнал низкого уровня. Процедура арбитража дает приоритет устройству, которое передает поток последовательных данных с наименьшим двоичным значением. Ведущий передатчик, потерявший арбитраж, переключается в ведомый режим приема и устанавливает флаг потери арбитража ALIFG. Если два или более устройства посылают одинаковые первые байты, арбитраж продолжается на последующих байтах.

Если выполняется процедура арбитража, когда на SDA повторяются условия «СТАРТ» или «СТОП», ведущие передатчики, вовлеченные в арбитраж, должны послать повторные условия «СТАРТ» или «СТОП» в том же самом месте в формате фрэйма. Арбитраж не разрешается между:

- Повторным условием «СТАРТ» и битом данных
- Условием «СТОП» и битом данных
- Повторным условием «СТАРТ» и условием «СТОП»



* Когда I2RM = 1, I2CSTP должен быть установлен перед записью последнего значения I2CDR. В противном случае генерация СТОП будет некорректной.

Рис. 15-8. Режим ведущего передатчика

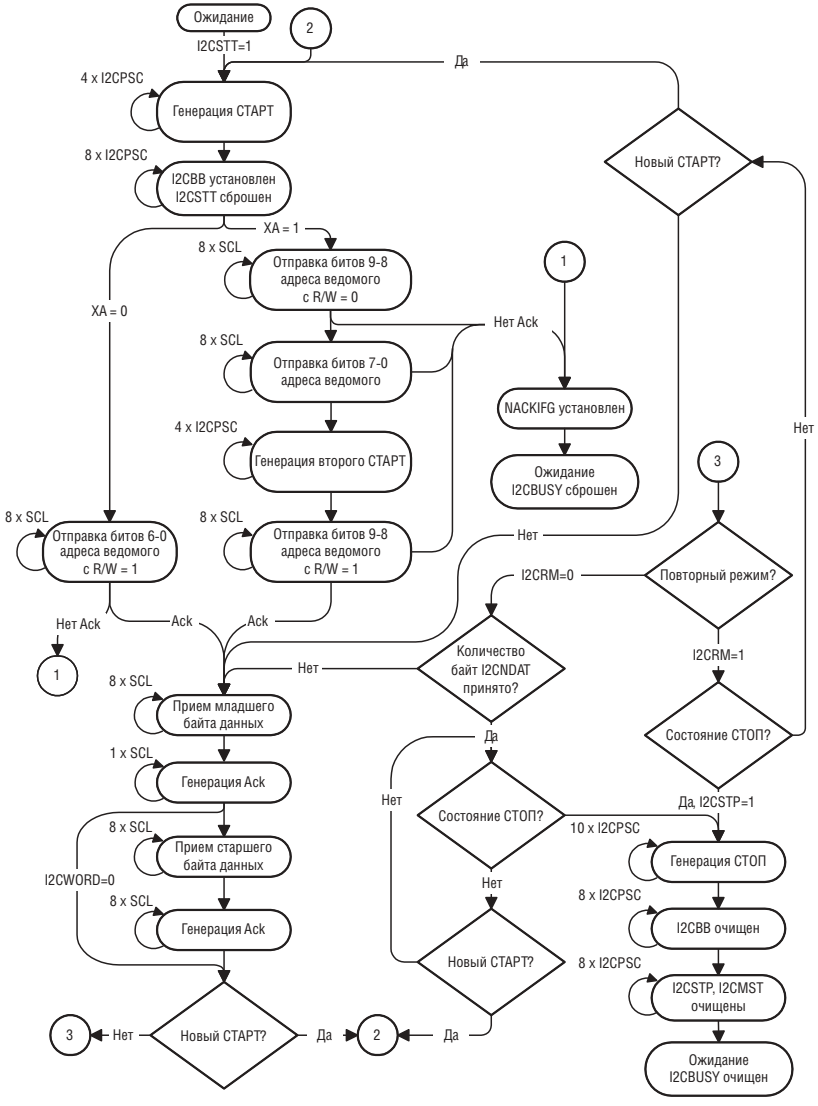


Рис. 15-9. Режим ведущего приемника

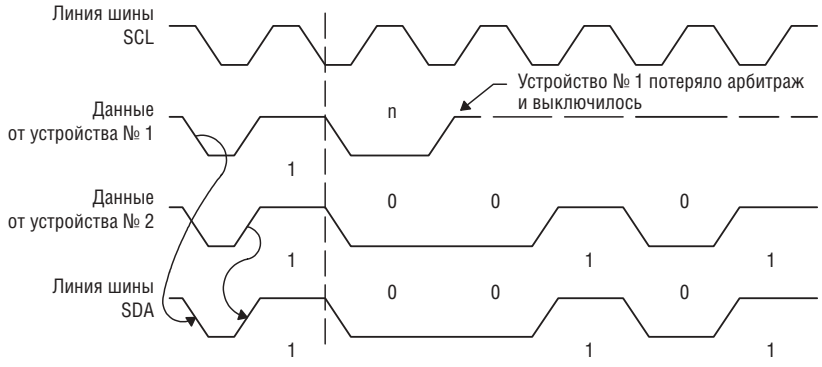


Рис. 15-10. Процедура арбитража между двумя ведущими передатчиками

Автоматический подсчет байтов данных

Автоматический подсчет байтов данных поддерживается в режиме ведущего с помощью регистра I2CNDAT. Когда I2CRM=0, число принятых или отправленных байтов записывается в I2CNDAT. Условие «СТОП» автоматически генерируется после того, как было передано количество байт, содержащееся в I2CNDAT.

Примечание: Регистр I2CNDAT

Не следует изменять содержимое регистра I2CNDAT, когда I2CBB=1 и I2CRM=0. В противном случае возможен непредсказуемый результат.

Режим ведомого

В режиме ведомого операции передачи и приема автоматически управляются модулем I²C. Режимы ведомого передатчика и ведомого приемника показаны на рис. 15-11 и рис. 15-12.

В режиме ведомого приемника биты последовательных данных принимаются на SDA и сдвигаются по тактовым импульсам, генерируемым ведущим устройством. Ведомое устройство не генерирует тактовый сигнал, но может удерживать линию SCL в состоянии низкого уровня, если после приема байта необходимо вмешательство ЦПУ.

Вход в режим ведомого передатчика происходит, когда байт адреса ведомого, переданный ведущим, совпадает с собственным адресом, и был послан установленный бит R/W, указывающий на запрос отправки данных ведущему. Ведомый передатчик сдвигает последовательные данные наружу на SDA по импульсам тактирования, генерируемым ведущим устройством. Ведомое устройство не

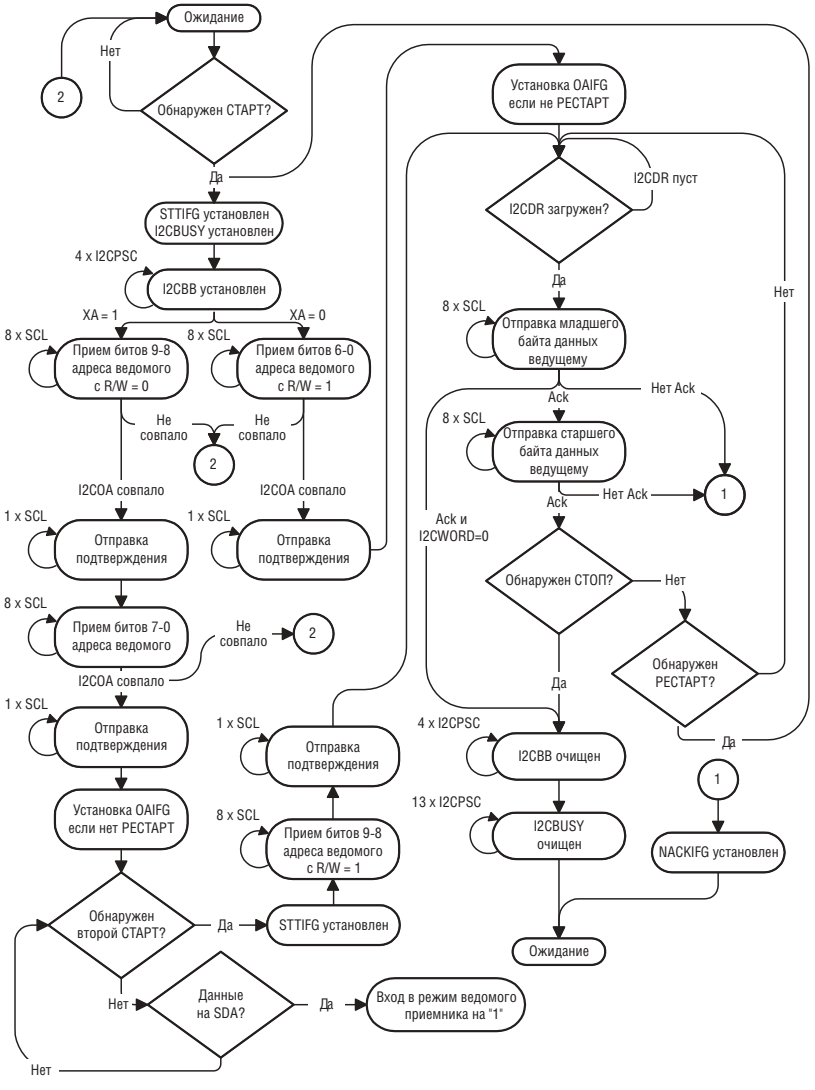


Рис. 15-11. Ведомый передатчик

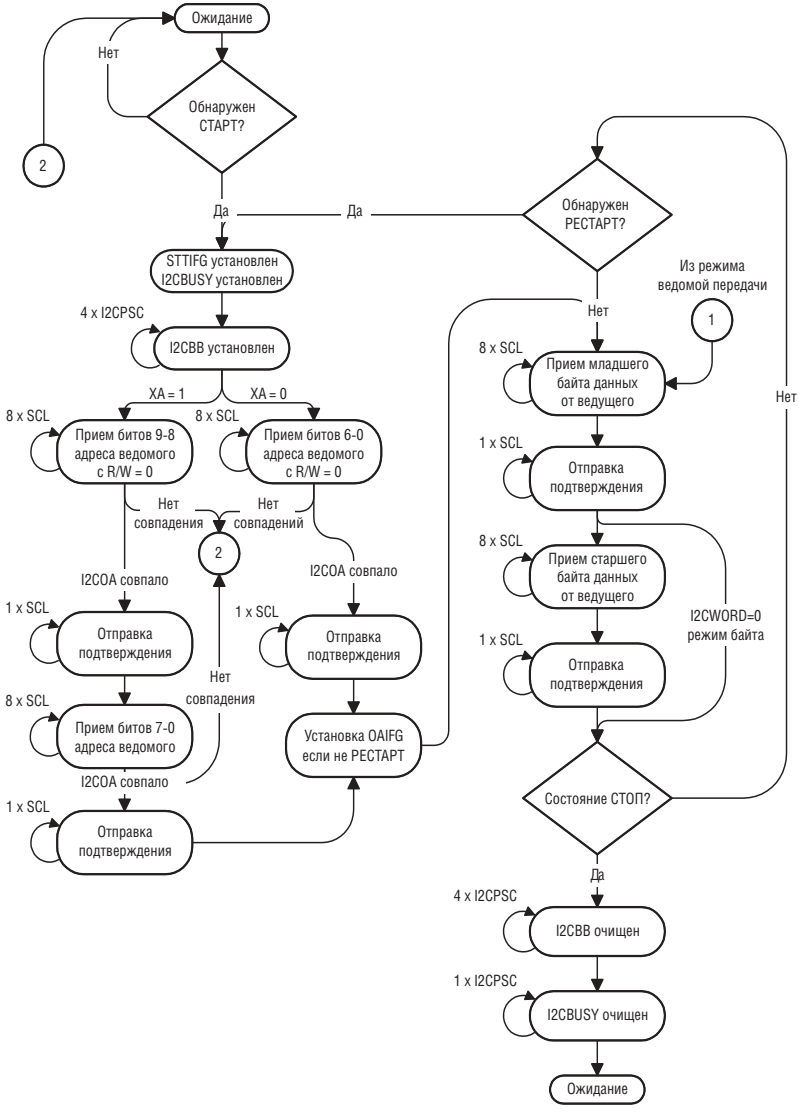


Рис. 15-12. Ведомый приемник

генерирует тактовых сигналов, но может удерживать линию SCL в состоянии низкого уровня, если после передачи байта необходимо вмешательство ЦПУ.

Примечание: Бит I2CTRX в режиме ведомого

Бит I2CTRX должен быть очищен для нормальной работы в режиме ведомого

15.2.5. Регистр данных I2CDR модуля I²C

Регистр I2CDR может быть доступен как 8-ми или 16-разрядный регистр, что определяется битом I2CWORD. Функции регистра I2CDR описаны в таблице 15-2. Когда I2CWORD=1, любая попытка изменить регистр с помощью однобайтной команды будет неуспешной и регистр не будет модифицирован.

Таблица 15-2. Функции регистра I2CDR

I2CWORD	I2CTRX	Функция I2CDR
0	1	Режим передачи байта: Используется только младший байт. Байт дважды буферизируется. Если новый байт записан до передачи предыдущего байта, новый байт ожидает во временном буфере до момента защелкивания в младшем байте регистра I2CDR. Когда I2CDR доступен, устанавливается бит TXRDYIFG.
0	0	Режим приема байта: Используется только младший байт. Байт дважды буферизируется. Если новый байт принят до прочтения предыдущего байта, новый байт ожидает во временном буфере до момента защелкивания в младшем байте регистра I2CDR. Когда I2CDR готов для чтения, устанавливается бит RXRDYIFG.
1	1	Режим передачи слова: Первым передается младший байт слова, затем старший байт. Регистр дважды буферизируется. Если новое слово записано до передачи предыдущего слова, новое слово ожидает во временном буфере до момента защелкивания в регистре I2CDR. Когда I2CDR доступен, устанавливается бит TXRDYIFG.
1	0	Режим приема слова: Первым принимается младший байт слова, затем старший байт. Регистр дважды буферизируется. Если новое слово принято до прочтения предыдущего слова, новое слово ожидает во временном буфере до момента защелкивания в регистре I2CDR. Когда I2CDR готов к доступу, устанавливается бит RXRDYIFG.

Опустошение при передаче

В режиме ведущего опустошение происходит, когда сдвиговый регистр передачи и буфер передачи пусты. В режиме ведомого опустошение происходит, когда сдвиговый регистр передачи и буфер передачи пусты, а внешний ведущий I²C все еще запрашивает данные. Когда происходит опустошение при передаче, устанавливается бит I2CTXUDF. Запись данных в регистр I2CDR или

сброс бита I2CEN сбрасывают I2CTXUDF. I2CTXUDF используется только в режиме передачи.

Переполнение при приеме

Переполнение при приеме происходит, когда сдвиговый регистр приема и буфер приема заполнены. Когда происходит переполнение при приеме, устанавливается бит I2CRXOVR. Потери данных не происходит, поскольку в этом случае линия SCL удерживается в состоянии низкого уровня, которое приостанавливает дальнейшую активность на шине. Чтение регистра I2CDR или сброс бита I2CEN сбрасывают бит I2CRXOVR. Бит I2CRXOVR используется только в режиме приема.

15.2.6. Генерация тактовых сигналов I2C и синхронизация

Модуль I²C работает с источником тактовой частоты, выбираемым битами I2CSSELx. Прескалер I2CPSC и регистры I2CSCLH и I2CSCLL определяют частоту и скважность тактового сигнала SCL для режима ведомого, как показано на рис. 15-13.

Примечание: максимальная частота I2CCLK

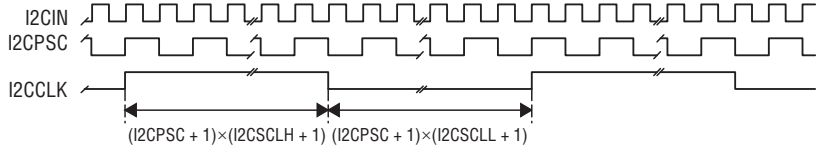


Рис. 15-13. Генерация сигналов на линии SCL модуля I²C

Источник тактовых импульсов I2CIN модуля I²C должен иметь частоту, по крайней мере, в 10 раз больше частоты SCL в обоих режимах ведущего и ведомого. Это условие выполняется автоматически в режиме ведущего регистрами I2CSCLL и I2CSCLH.

Примечание: значение U2CPSC

Когда I2CPSC>4, функционирование может оказаться непредсказуемым. Для установки частоты SCL необходимо использовать регистры I2CSCLL и I2CSCLH.

Во время процедуры арбитража тактовые импульсы от различных ведущих должны быть синхронизированы. Устройство, первым генерирующее период низкого уровня на SCL, берет верх над другими устройствами, вынуждая их запустить их собственные периоды низкого уровня. В этом случае SCL удерживается в состоянии низкого уровня устройством с самым долгим периодом низкого уровня. Другие устройства должны ожидать, пока линия SCL будет освобождена перед стартом своих периодов высокого уровня. Рис. 15-14 ил-

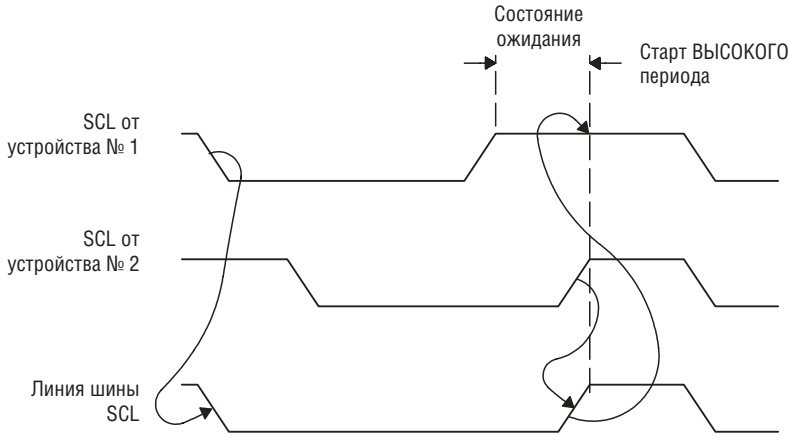


Рис. 15-14. Синхронизация двух тактовых генераторов I²C во время арбитража

люстрирует тактовую синхронизацию. Это позволяет медленному ведомому замедлять быстрого ведущего.

15.2.7. Использование модуля I2C в режимах пониженного энергопотребления

Модуль I²C может использоваться в MSP430 в режимах пониженного энергопотребления. Когда для модуля I²C задействован внутренний источник тактирования, модуль работает нормально независимо от режима работы MSP430. Когда внутренний источник тактирования модуля I²C отсутствует, обеспечивается автоматическая активация тактирования. Когда модуль I²C простаивает, I2CBUSY=0 и источник тактирования I2CIN отключен от модуля I²C, сохраняя энергию источника питания.

Когда источник тактирования I²C неактивен, модуль I²C автоматически активирует выбранный источник тактирования, когда это необходимо, независимо от установок управляющих битов источника тактирования. Источник тактирования остается активным до перехода модуля I²C в состояние ожидания. После возврата модуля I²C в режим ожидания, управление источником тактирования возвращается к установкам его управляющих битов.

Автоматическая активация тактирования I²C происходит в следующих ситуациях:

- В режиме ведущего тактирование активируется, когда I2CSTT=1 и остается активным до завершения переноса, после чего модуль I²C возвращается в состояние ожидания.

- В режиме ведомого тактирование активируется, когда обнаружено условие «СТАРТ» и остается активным до завершения переноса, после чего модуль I²C возвращается в состояние ожидания. После определения условия «СТАРТ» устанавливается флаг STTIFG и модуль удерживает линию SCL в низком состоянии, пока источник тактирования не станет активным. Как только источник запускается, модуль I²C освобождает линию SCL ведущему.

Когда модуль I²C активирует неактивный источник тактирования, источник тактирования становится активным для всего устройства, и любая периферия, сконфигурированная для использования этого источника, окажется задействованной. Например, таймер, использующий SMCLK, будет инкрементироваться, пока модуль I²C будет удерживать SMCLK в активном состоянии.

15.2.8. Прерывания I²C

Модуль I²C имеет один вектор прерывания для восьми флагов прерывания, показанных в таблице 15-3. Каждый флаг прерывания имеет собственный бит разрешения прерывания. Когда прерывание разрешено, и бит GIE установлен, флаг прерывания будет генерировать запрос прерывания.

Таблица 15-3. Прерывания I²C

Флаг прерывания	Условие прерывания
ALIFG	Потеря арбитража. Арбитраж может быть потерян, когда два или более передатчиков начинают передачу одновременно или когда программное обеспечение пытается инициировать I²C перенос при I2CBB=1. Флаг ALIFG устанавливается, когда арбитраж был потерян. Когда ALIFG устанавливается, биты MST и I2CSTP очищаются и контроллер I²C становится ведомым приемником.
NACKIFG	Прерывание при отсутствии подтверждения. Этот флаг устанавливается, когда подтверждение ожидается, но не получено.
OAIFG	Прерывание собственного адреса. Этот флаг устанавливается, когда другой ведущий имеет адрес модуля I²C. OAIFG используется только в режиме ведомого.
ARDYIFG	Прерывание «регистр доступен для чтения». Этот флаг устанавливается, когда ранее запрограммированный перенос завершен, и биты статуса обновлены. Это прерывание используется для уведомления ЦПУ о том, что регистры I²C готовы к доступу.
RXRDYIFG	Прерывание/статус готовности приема. Этот флаг устанавливается, когда модуль I²C принял новые данные. RXRDYIFG автоматически очищается, когда I2CDR прочитан и буфер приема пуст. Переполнение приемника показывается, если бит I2CRXOVR=1. RXRDYIFG используется только в режиме приема.
TXRDYIFG	Прерывание/статус готовности передачи. Этот флаг устанавливается, когда модуль I²C готов для новой передачи данных (режим ведущего передатчика) или когда другой ведущий запрашивает данные (режим ведомого передатчика). TXRDYIFG автоматически очищается, когда I2CDR и буфер передачи полны. Опустошение передачи показывается, если I2CTXUDF=1. Не используется в режиме приема.

Таблица 15-3. (Окончание)

Флаг прерывания	Условие прерывания
GCIFG	Прерывание общего вызова. Этот флаг устанавливается, когда модуль I²C принял адрес общего вызова (00h). GCIFG используется только в режим приема.
STTIFG	Прерывание при обнаружении условия «СТАРТ». Этот флаг устанавливается, когда модуль I²C обнаружил условие «СТАРТ» в режиме ведомого. Это позволяет MSP430 находиться в режиме пониженного энергопотребления с неактивным источником тактирования I²C, пока ведущий не инициирует связь по I²C. STTIFG используется только в режиме ведомого.

Генератор вектора прерывания I2CIV

Флаги прерывания I²C разделены по приоритетам и объединены в источник одного вектора прерывания. Регистр вектора прерывания I2CIV используется для выяснения, какой флаг запросил прерывание. Разрешенное прерывание с наивысшим приоритетом генерирует число в регистре I2CIV, которое может быть оценено или добавлено к программному счетчику для автоматического входа в соответствующую программную процедуру. Запрещенные I²C прерывания не воздействуют на содержимое I2CIV. Когда RXDMAEN=1, RXRDYIFG не будет влиять на значение I2CIV и когда TXDMAEN=1, TXRDYIFG не будет влиять на значение I2CIV, независимо от состояния RXRDYIE или TXRDYIE.

При любом доступе (чтение или запись) к регистру I2CIV автоматически сбрасывается флаг ожидающего прерывания с наивысшим приоритетом. Если устанавливается другой флаг прерывания, после обработки начального прерывания немедленно генерируется другое прерывание.

Пример программного обеспечения, использующего I2CIV

Приведенный далее пример программного обеспечения показывает рекомендуемое использование I2CIV. Значение I2CIV добавляется к PC для автоматического перехода к соответствующей процедуре:

```
I2C_ISR
   ADD &I2CIV, PC     ;Добавление смещения к таблице переходов
   RETI               ;Вектор 0: Нет прерывания
   JMP AL IFG_ISR     ;Вектор 2: ALIFG
   JMP NACKIFG_ISR    ;Вектор 4: NACKIFG
   JMP OAIFG_ISR      ;Вектор 6: OAIFG
   JMP ARDYIFG_ISR    ;Вектор 8: ARDYIFG
   JMP RXRDYIFG_ISR   ;Вектор 10: RXRDYIFG
   JMP TXRDYIFG_ISR   ;Вектор 12: TXRDYIFG
   JMP GCIFG_ISR      ;Вектор 14: GCIFG
STTIFG_ISR            ;Вектор 16
   ...                ;Задача (программный модуль) начинается здесь
```

```
   RETI              ;Возврат
ALIFG_ISR            ;Вектор 2
   ...               ;Задача начинается здесь
   RETI              ;Возврат
NACKIFG_ISR          ;Вектор 4
   ...               ;Задача начинается здесь
   RETI              ;Возврат
OAIFG_ISR            ;Вектор 6
   ...               ;Задача начинается здесь
   RETI              ;Возврат
ARDYIFG_ISR          ;Вектор 8
   ...               ;Задача начинается здесь
   RETI              ;Возврат
RXRDYIFG_ISR         ;Вектор 10
   ...               ;Задача начинается здесь
   RETI              ;Возврат
TXRDYIFG_ISR         ;Вектор 12
   ...               ;Задача начинается здесь
   RETI              ;Возврат
GCIFG_ISR            ;Вектор 14
   ...               ;Задача начинается здесь
   RETI              ;Возврат
```

15.3. Регистры модуля I²C

Регистры модуля I²C приведены в таблице 15-4.

Таблица 15-4. Регистры I²C

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Разрешение прерывания I²C	I2CIE	Чтение/запись	050h	Сброс с PUC
Флаг прерывания I²C	I2CIFG	Чтение/запись	051h	Сброс с PUC
Счет данных I²C	I2CNDAT	Чтение/запись	052h	Сброс с PUC
Управление USART	U0CTL	Чтение/запись	070h	001h с PUC
Управление передачей I²C	I2CTCTL	Чтение/запись	071h	Сброс с PUC
Управление данными I²C	I2CDCTL	Только чтение	072h	Сброс с PUC
Прескалер I²C	I2CPSC	Чтение/запись	073h	Сброс с PUC
«Высокий» SCL I²C	I2CSCLH	Чтение/запись	074h	Сброс с PUC
«Низкий» SCL I²C	I2CSCLL	Чтение/запись	075h	Сброс с PUC
Данные I²C	I2CDRW/I2CDRB	Чтение/запись	076h	Сброс с PUC
Собственный адрес I²C	I2COA	Чтение/запись	0118h	Сброс с PUC
Адрес ведомого I²C	I2CSA	Чтение/запись	011Ah	Сброс с PUC
Вектор прерываний I²C	I2CIV	Только чтение	011Ch	Сброс с PUC

U0CTL, регистр управления USART0 в режиме I²C

7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	I²C	XA	LISTEN	SYNC	MST	I2CEN
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

RXDMAEN	Бит 7	Разрешение DMA при приеме. Этот бит разрешает использовать контроллер DMA для переноса принятых модулем I²C данных. Когда RXDMAEN=1, RXRDYIE игнорируется. 0 – Запрещено 1 – Разрешено
TXDMAEN	Бит 6	Разрешение DMA при передаче. Этот бит разрешает использовать контроллер DMA для переноса данных в модуль I²C для их последующей передачи. Когда TXDMAEN=1, TXRDYIE игнорируется. 0 – Запрещено 1 – Разрешено
I2C	Бит 5	Включение режима I²C. Этот бит позволяет выбрать режим I2C или SPI, когда SYNC=1. 0 – Режим SPI 1 – Режим I²C
XA	Бит 4	Расширенная адресация 0 – 7-разрядная адресация 1 – 10-разрядная адресация
LISTEN	Бит 3	Прослушивание. Этот бит устанавливает режим обратной петли. Бит LISTEN действителен только когда MST=1 и I2CTRX=1 (ведущий передатчик). 0 – Нормальный режим 1 – SDA внутренне подключается назад к приемнику (обратная петля).
SYNC	Бит 2	Включение синхронного режима 0 – Режим UART 1 – Режим SPI или I²C
MST	Бит 1	Ведущий. Этот бит выбирает режим ведомого или ведущего. Бит MST автоматически очищается при потере арбитража или генерации условия СТОП. 0 – Режим ведомого 1 – Режим ведущего
I2CEN	Бит 0	Включение I²C. Бит включает или выключает модуль I²C. В исходном состоянии он установлен, и для UART или SPI функционирует как SWRST. Когда первыми после PUC устанавливаются биты I²C и SYNC, этот бит функционирует как I2CEN и очищается автоматически. 0 – Работа I²C запрещена. 1– Работа I²C разрешена.

I2CTCTL, регистр управления передачей режиме I²C

7	6	5	4	3	2	1	0
I2CWORD	I2CRM	I2CSSELx		I2CTRX	I2CSTB	I2CSTP	I2CSTT
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Модифицируется, только когда I2CEN=0

I2CWORD	Бит 7	Режим слова I²C. Выбирается режим байта или слова для регистра данных I²C. 0 – Режим байта 1 – Режим слова
I2CRM	Бит 6	Режим повтора I²C 0 – I2CNDAT определяет количество переданных байтов. 1 – Количество передаваемых байт определяется программным обеспечением. I2CNDAT не используется.
I2CSSELx	Биты 5-4	Выбор источника тактирования I²C. Когда MST=1 и арбитраж потерян, автоматически используется внешний сигнал SCL. 00 – Нет тактирования – модуль I²C неактивен 01 – ACLK 10 – SMCLK 11 – SMCLK
I2CTRX	Бит 3	Передача I²C. Этот бит выбирает функцию передачи или приема для контроллера I²C когда MST=1. Когда MST=0, бит R/W байта адреса определяет направление данных. I2CTRX должен быть сброшен для нормальной работы в режиме ведомого. 0 – Режим приема. Данные принимаются на выводе SDA. 1 – Режим передачи. Данные передаются на выводе SDA.
I2CSTB	Бит 2	Стартовый байт. Установка бита I2CSTB при MST=1 инициирует стартовый байт когда I2CSTT=1. После инициирования стартового байта, I2CSTB автоматически очищается. 0 – Нет действия 1 – Отправка условия СТАРТ и стартового байта (01h), но не условия СТОП.
I2CSTP	Бит 1	Бит СТОП. Этот бит используется для генерации условия СТОП. После условия СТОП бит I2CSTP автоматически очищается. 0 – Нет действия 1 – Отправка условия СТОП
I2CSTT	Бит 0	Бит СТАРТ. Этот бит используется для генерации условия СТАРТ. После условия СТАРТ бит I2CSTT автоматически очищается. 0 – Нет действия 1 – Отправка условия СТАРТ

I2CDCTL, регистр управления данными I²C

7	6	5	4	3	2	1	0
Не используется	Не используется	I2CBUSY	I2C SCLLOW	I2CSBD	I2CTXUDF	I2CRXOVR	I2CBB
r0	r0	r-0	r-0	r-0	r-0	r-0	r-0

Не используется	Биты 7-6	Не используются. Всегда читаются как 0.
I2CBUSY	Бит 5	Занятость I²C 0 – Модуль I²C свободен (в режиме ожидания – IDLE) 1 – Модуль I²C занят
I2CSCLLOW	Бит 4	«Низкий» SCL I²C. Этот бит показывает, что ведомый удерживает линию SCL в низком состоянии, когда MSP430 ведущий, и не используется в режиме ведомого. 0 – SCL не удерживается в состоянии низкого уровня. 1 – SCL удерживается в состоянии низкого уровня.
I2CSBD	Бит 3	Один байт данных I²C. Этот бит показывает, что содержится в регистре приема – слово или байт. Бит I2CSBD действителен только когда I2CWORD=1. 0 – Слово было принято полностью 1 – В I2CDR достоверен только младший байт
I2CTXUDF	Бит 2	Опустошение при передаче I²C 0 – Опустошения нет 1 – Произошло опустошение при передаче
I2CRXOVR	Бит 1	Переполнение при приеме I²C. 0 – Переполнения при приеме не произошло. 1 – Произошло переполнение при приеме.
I2CBB	Бит 0	Бит занятости I²C. Условие СТАРТ устанавливает I2CBB в 1. I2CBB сбрасывается условием СТОП или при I2CEN=0. 0 – Шина I²C не занята 1 – Шина I²C занята

I2CDRW, I2CDRB регистр данных I²C

15	14	13	12	11	10	9	8
Старший байт I2CDRW							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Младший байт I2CDRW, байт I2CDRB							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

I2CDRW/I2CDRB	Биты 15-0/Биты 7-0	Данные I²C. Когда I2CWORD=1, регистр называется I2CDRW. Когда I2CWORD=0, регистр называется I2CDRB. Когда I2CWORD=1, любая попытка изменить регистр байтной командой будет безуспешной и регистр не будет обновлен.

I2CNDAT, регистр подсчета переданных байтов I²C

7	6	5	4	3	2	1	0
I2CNDATx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

I2CNDATx	Биты 7-0	Количество байтов I²C. Этот регистр обеспечивают автоматический подсчет байтов данных в режиме ведущего. В режиме слова I2CNDATx должен иметь четное значение.

I2CPSC, регистр прескалера тактирования I²C

7	6	5	4	3	2	1	0
I2CPSCx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Модифицируется, только когда I2CEN=0

I2CPSCx	Биты 7-0	Прескалер тактирования I²C. Входная тактовая частота I2CIN модуля I²C делится содержимым I2CPSCx, обеспечивая внутреннюю тактовую частоту I²C. Коэффициент деления равен I2CPSC+1. Значения I2CPSCx > 4 не рекомендуются. Для утановки частоты SCL должны использоваться регистры I2CSCLL и I2CSCLH. 000h – Деление на 1 001h – Деление на 2 . . 0FFh – Деление на 256

I2CSCLH, сдвиговый регистр высокого уровня сигналов тактирования I²C

7	6	5	4	3	2	1	0
I2CSCLHx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Модифицируется, только когда I2CEN=0

I2CSCLHx	Биты 7-0	Сдвиговый регистр высокого уровня тактовых импульсов I²C. Эти биты определяют длительность высокого уровня сигнала на SCL, когда контроллер I²C находится в режиме ведущего. Длительность высокого уровня SCL составляет (I2CSCLH + 2)×(I2CPSC+1). 000h – Длительность высокого уровня SCL = 5×(I2CPSC+1) 001h – Длительность высокого уровня SCL = 5×(I2CPSC+1) 002h – Длительность высокого уровня SCL = 5×(I2CPSC+1) 003h – Длительность высокого уровня SCL = 5×(I2CPSC+1) 004h – Длительность высокого уровня SCL = 6×(I2CPSC+1) . . 0FFh – Длительность высокого уровня SCL = 257×I2CPSC

I2CSCLL, сдвиговый регистр низкого уровня сигналов тактирования I²C

7	6	5	4	3	2	1	0
I2CSCLLx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Модифицируется, только когда I2CEN=0

I2CSCLLx	Биты 7-0	Сдвиговый регистр низкого уровня тактовых импульсов I²C. Эти биты определяют длительность низкого уровня сигнала на SCL, когда контроллер I²C находится в режиме ведущего. Длительность низкого уровня SCL составляет (I2CSCLL + 2)×(I2CPSC+1). 000h – Длительность низкого уровня SCL = 5×(I2CPSC+1) 001h – Длительность низкого уровня SCL = 5×(I2CPSC+1) 002h – Длительность низкого уровня SCL = 5×(I2CPSC+1) 003h – Длительность низкого уровня SCL = 5×(I2CPSC+1) 004h – Длительность низкого уровня SCL = 6×(I2CPSC+1) . . 0FFh – Длительность низкого уровня SCL = 257×I2CPSC

I2COA, регистр собственного адреса I²C в 7-разрядном адресном режиме

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
0	I2COAx						
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Модифицируется, только когда I2CEN=0

I2COAx	Биты 15-0	Собственный адрес I²C. Регистр I2COA содержит локальный адрес контроллера I²C MSP430. Регистр I2COA выровнен по правому краю. Старшим битом (MSB) является 6-ой бит. Биты 15-7 всегда равны 0.

I2COA, регистр собственного адреса I²C в 10-разрядном адресном режиме

15	14	13	12	11	10	9	8
0	0	0	0	0	0	I2COAx	
r0	r0	r0	r0	r0	r0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COAx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

	Модифицируется, только когда I2CEN=0

I2COAx	Биты 15-0	Собственный адрес I²C. Регистр I2COA содержит локальный адрес контроллера I²C MSP430. Регистр I2COA выровнен по правому краю. Старшим битом (MSB) является 9-ый бит. Биты 15-10 всегда равны 0.

I2CSA, регистр адреса ведомого I²C в 7-разрядном адресном режиме

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
r0	r0	r0	r0	r0	r0	r0	r0

7	6	5	4	3	2	1	0
0	I2CSAx						
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

I2CSAx	Биты 15-0	Адрес ведомого I²C. Регистр I2CSA содержит адрес внешнего ведомого устройства, которое адресуется MSP430. Он используется только в режиме ведущего. Регистр I2CSA выровнен по правому краю. Старшим битом (MSB) является 6-ой бит. Биты 15-7 всегда равны 0.

I2CSA, регистр адреса ведомого I²C в 10-разрядном адресном режиме

15	14	13	12	11	10	9	8
0	0	0	0	0	0	I2CSAx	
r0	r0	r0	r0	r0	r0	rw-0	rw-0

7	6	5	4	3	2	1	0
I2CSAx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

I2CSAx	Биты 15-0	Адрес ведомого I²C. Регистр I2CSA содержит адрес внешнего ведомого устройства, которое адресуется MSP430. Он используется только в режиме ведущего. Регистр I2CSA выровнен по правому краю. Старшим битом (MSB) является 9-ый бит. Биты 15-10 всегда равны 0.

I2CIE, регистр разрешения прерываний I²C

7	6	5	4	3	2	1	0
STTIE	GCIE	TXRDYIE	RXRDYIE	ARDYIE	OAIE	NACKIE	ALIE
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

STTIE	Бит 7	Разрешение прерывания при обнаружении старта 0 – Прерывание запрещено 1 – Прерывание разрешено
GCIE	Бит 6	Разрешение прерывания при общем вызове 0 – Прерывание запрещено 1 – Прерывание разрешено
TXRDYIE	Бит 5	Разрешение прерывания при готовности передачи. Когда TXDMAEN=1, TXRDYIE игнорируется и TXRDYIFG не генерирует прерывание. 0 – Прерывание запрещено 1 – Прерывание разрешено
RXRDYIE	Бит 4	Разрешение прерывания при готовности приема. Когда RXDMAEN=1, RXRDYIE игнорируется и RXRDYIFG не генерируется прерывание. 0 – Прерывание запрещено 1 – Прерывание разрешено
ARDYIE	Бит 3	Разрешение прерывания готовности доступа. 0 – Прерывание запрещено 1 – Прерывание разрешено
OAIE	Бит 2	Разрешение прерывания собственного адреса. 0 – Прерывание запрещено 1 – Прерывание разрешено
NACKIE	Бит 1	Разрешение прерывания при отсутствии подтверждения. 0 – Прерывание запрещено 1 – Прерывание разрешено
ALIE	Бит 0	Разрешение прерывания при потере арбитража. 0 – Прерывание запрещено 1 – Прерывание разрешено

I2CIFG, регистр флагов прерываний I²C

7	6	5	4	3	2	1	0
STTIFG	GCIFG	TXRDYIFG	RXRDYIFG	ARDYIFG	OAIFG	NACKIFG	ALIFG
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

STTIFG	Бит 7	Флаг прерывания при обнаружении СТАРТ 0 – Прерывание не ожидается 1 – Прерывание ожидается
GCIFG	Бит 6	Флаг прерывания общего вызова 0 – Прерывание не ожидается 1 – Прерывание ожидается
TXRDYIFG	Бит 5	Флаг прерывания готовности передачи 0 – Прерывание не ожидается 1 – Прерывание ожидается

<table>
<tr><td>RXRDYIFG</td><td>Бит 4</td><td colspan="3">Флаг прерывания готовности приема.
0 – Прерывание не ожидается
1 – Прерывание ожидается</td></tr>
<tr><td rowspan="8">ARDYIFG</td><td rowspan="8">Бит 3</td><td colspan="3">Флаг прерывания готовности доступа. Флаг ARDYIFG устанавливается в следующих случаях:</td></tr>
<tr><td>Режим</td><td>Используемые биты</td><td>Условия установки ARDYIFG</td></tr>
<tr><td>Ведущий передатчик</td><td>I2CRM = 0

I2CRM = 1</td><td>Передано количество байт, содержащееся в I2CNDAT.
После установки I2CSTP послан последний байт данных.</td></tr>
<tr><td>Ведущий приемник</td><td>I2CRM = 0

I2CRM = 1</td><td>Принято количество байт, содержащееся в I2CNDAT и буфер приема пуст.
Принят последний байт данных и буфер приема опустошен после установки I2CSTP.</td></tr>
<tr><td>Ведомый передатчик</td><td>–</td><td>Принято условие СТОП.</td></tr>
<tr><td>Ведомый приемник</td><td>–</td><td>Принято условие СТОП и буфер приема пуст.</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>OAIFG</td><td>Бит 2</td><td colspan="3">Флаг прерывания собственного адреса.
0 – Прерывание не ожидается
1 – Прерывание ожидается</td></tr>
<tr><td>NACKIFG</td><td>Бит 1</td><td colspan="3">Флаг прерывания при отсутствии подтверждения.
0 – Прерывание не ожидается
1 – Прерывание ожидается</td></tr>
<tr><td>ALIFG</td><td>Бит 0</td><td colspan="3">Флаг прерывания потери арбитража.
0 – Прерывание не ожидается
1 – Прерывание ожидается</td></tr>
</table>

I2CIV, регистр вектора прерываний I²C

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
r0	r0	r0	r0	r0	r0	r0	r0

7	6	5	4	3	2	1	0
0	0	0	I2CIVx				0
r0	r0	r0	r-0	r-0	r-0	r-0	r0

		Значение вектора прерываний I²C			
		Содержимое I2CIV	**Источник прерывания**	**Флаг прерывания**	**Приоритет прерывания**
I2CIVx	Биты 15-0	000h	Прерывание не ожидается	–	
		002h	Потеря арбитража	ALIFG	Наивысший
		004h	Отсутствие подтверждения	NACKIFG	
		006h	Собственный адрес	OAIFG	
		008h	Готовность доступа к регистру	ARDYIFG	
		00Ah	Готовность приема данных	RXRDYIFG	
		00Ch	Готовность передачи данных	TXRDYIFG	
		00Eh	Общий вызов	GCIFG	
		010h	Принято условие СТАРТ	STTIFG	Низший

MSP430x1xxFamily

Компаратор A

Раздел XVI.

Texas Instruments

Компаратор A

Компаратор A – это аналоговый компаратор напряжения. В этом разделе описывается компаратор A. Компаратор A реализован в устройствах MSP430x11x1, MSP430x12x, MSP430x13x, MSP430x14x, MSP430x15x и MSP430x16x.

16.1. Введение в компаратор A

Модуль компаратора A поддерживает высокоточные аналого-цифровые преобразования напряжения, контроль напряжения питания и мониторинг внешних аналоговых сигналов. Блок-схема компаратора A показана на рис. 16-1.

Компаратор A имеет следующие возможности:

- Инвертирующий и не инвертирующий выводы входного мультиплексора
- Программно настраиваемый RC-фильтр на выходе компаратора
- Подключение выхода к входу захвата Таймера A

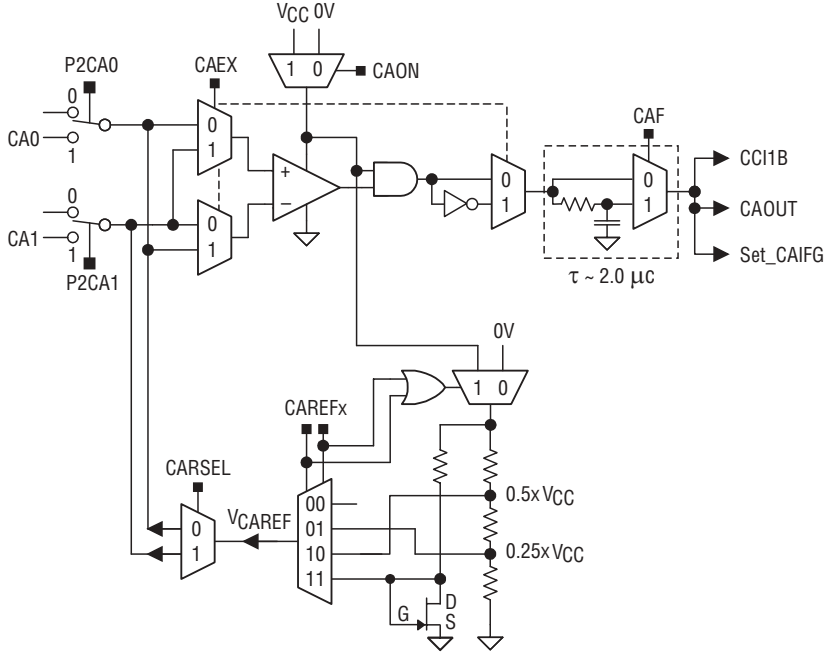


Рис. 16-1. Блок-схема компаратора A

- Программное управление входным буфером порта
- Возможность прерывания
- Настраиваемый генератор опорного напряжения
- Возможность выключения компаратора и опорного генератора

16.2. Функционирование компаратора A

Модуль компаратора A конфигурируется программным обеспечением пользователя. Настройка и работа компаратора A обсуждаются в следующих далее разделах.

16.2.1. Компаратор

Компаратор сравнивает аналоговые напряжения на входах «+» и «-». Если вход «+» более положителен, чем вход «-», на выходе компаратора CAOUT появляется сигнал высокого уровня. Компаратор может быть включен или выключен с помощью управляющего бита CAON. Если компаратор не используется, для уменьшения потребляемого тока его необходимо выключать. Когда компаратор выключен, на выходе CAOUT всегда сигнал низкого уровня.

16.2.2. Входные аналоговые переключатели

Аналоговые входные переключатели подключают или отключают два входа компаратора от соответствующих выводов порта с помощью битов P2CAx. Оба входа компаратора могут управляться индивидуально. Биты P2CAx позволяют:

- Подключать внешние сигналы к входам «+» и «-» компаратора
- Подключать внутреннее опорное напряжения к соответствующему выходному выводу порта

Внутренне входной переключатель выполнен как переключатель T-типа для уменьшения искажений на пути прохождения сигнала.

Примечание: Подключение входа компаратора

Когда компаратор включен, его входы должны быть подключены к источнику сигнала, источнику питания или к общему выводу. В противном случае плавающие уровни напряжения могут вызвать неожиданные прерывания и повышенное потребление тока.

Бит CAEX управляет входным мультиплексором, определяющим, какие входные сигналы будут подключены к входам «+» и «-» компаратора. Кроме того, когда входы компаратора меняются друг с другом, выходной сигнал компаратора инвертируется. Это позволяет пользователю учитывать и компенсировать входное напряжение сдвига компаратора.

16.2.3. Выходной фильтр

Выход компаратора может использоваться как с внутренней фильтрацией, так и без неё. Когда управляющий бит CAF установлен, выход фильтруется с помощью интегрированного RC-фильтра.

Выход любого компаратора беспорядочно генерирует, если разность потенциалов на его входах мала. Внутренние и внешние паразитные эффекты, перекрестная связь при включении и между сигнальными линиями, линиями питания и другими частями системы оказывают влияние на эту генерацию, как показано на рис. 16-2. Колебания на выходе компаратора снижают точность и разрешающую способность результата сравнения. Использование выходного фильтра может уменьшить ошибки, связанные с генерацией компаратора.

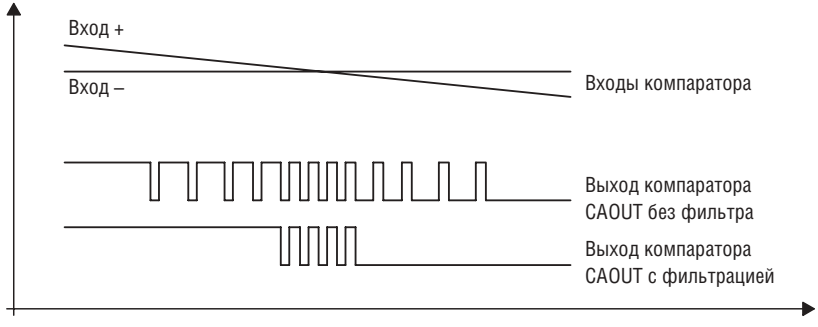


Рис. 16-2. Действие RC-фильтра на выходе компаратора

16.2.4. Генератор опорного напряжения

Генератор опорного напряжения используется для генерации напряжения VCAREF, которое может быть приложено к любому из входов компаратора. Бит CAREFx управляет выходом генератора напряжения. Бит CARSEL определяет вывод компаратора, к которому будет приложено напряжение VCAREF. Если внешние сигналы приложены к обоим входам компаратора, внутренний опорный генератор необходимо выключить, чтобы уменьшить величину потребляемого тока. Генератор опорного напряжения может вырабатывать напряжение, пропорционально уменьшенное относительно напряжения питания устройства VCC или фиксированное пороговое напряжение транзистора около 0,55 В.

16.2.5. Компаратор A, регистр отключения порта CAPD

Функции ввода и вывода компаратора мультиплексированы с соответствующими ножками порта ввода/вывода, которые являются цифровыми КМОП-схемами. Когда аналоговые сигналы подключаются к цифровым КМОП-элементам, может появиться паразитный ток между VCC и общим выводом. Этот паразитный ток появляется в случае, если величина входного напряжения находится около переходного уровня ячейки. Отключение буфера вывода порта устраняет паразитный ток и приводит с снижению общего потребления тока.

Когда биты CAPDx установлены, соответствующий входной буфер P2 отключается, как показано на рис. 16-3. Когда величина уровня потребления тока критична, любой вывод P2, подключенный к аналоговым сигналам, должен быть отключен с помощью соответствующего бита CAPDx.

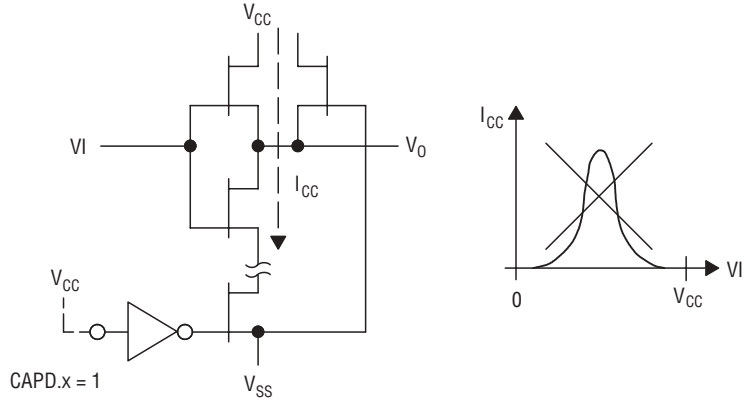


Рис. 16-3. Переходная характеристика и рассеивание мощности в инверторе/буфере КМОП

16.2.6. Прерывания компаратора A

С компаратором A связан один флаг прерывания и один вектор прерывания, как показано на рис. 16-4. Флаг прерывания CAIFG устанавливается по любому фронту (нарастающему или спадающему) сигнала на выходе компаратора, что определяется битом CAIES. Если установлены оба бита CAIE и GIE, флаг CAIFG генерирует запрос прерывания. Флаг CAIFG автоматически сбрасывается, когда обрабатывается запрос прерывания или может быть сброшен программным обеспечением.

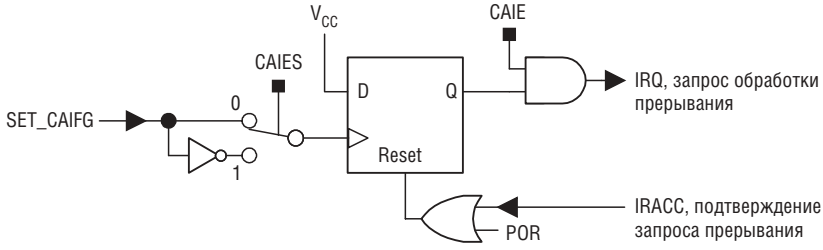


Рис. 16-4. Система прерывания компаратора А

16.2.7. Использование компаратора А для измерения сопротивления элементов

Компаратор А можно оптимизировать для высокоточного измерения резистивных элементов с помощью аналого-цифрового преобразования с одиночным интегрированием. К примеру, температура может быть преобразована в цифровые данные с помощью термистора путем сравнения времен разряда конденсатора, подключаемого сначала к термистору, а затем к опорному резистору, как показано на рис. 16-5. Сопротивление опорного резистора Rref сравнивается с Rmeas.

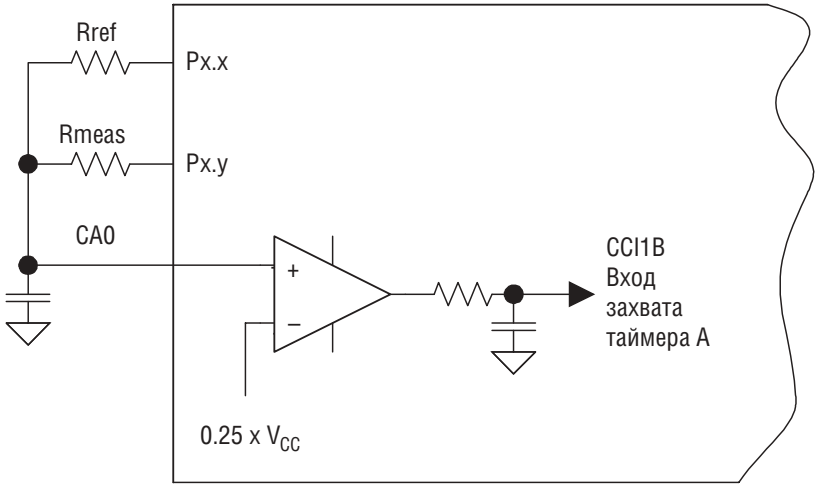


Рис. 16-5. Система измерения температуры

Для вычисления температуры, считанной Rmeas, используются ресурсы MSP430:

- Две цифровых ножки ввода/вывода для заряда и разряда конденсатора.
- Ножка ввода/вывода устанавливается в высокий уровень (VCC) для заряда конденсатора и сбрасывается для разряда.
- Когда ножка ввода/вывода не используется, она переключается на вход в «третье» состояние с установкой CAPDx,.
- Один выход заряжает и разряжает конденсатор через Rref.
- Один выход разряжает конденсатор через Rmeas.
- Вход «+» подключается к положительному выводу конденсатора.
- Вход «-» подключается к опорному уровню, например 0,25 × VCC.
- Для минимизации шумов переключения должен использоваться выходной фильтр.
- Выход CAOUT подключен к CCI1B таймера A, выполняющего захват времени разряда конденсатора.

Может быть измерено более одного резистивного элемента. Дополнительные элементы подключаются к CA0 с помощью доступных ножек ввода/вывода, переключающихся в «третье» состояние в моменты, когда они не участвуют в измерении.

Измерение температуры основано на принципе преобразования соотношения измерений. Отношение двух величин времен разряда конденсатора рассчитывается так, как показано на рис. 16-6.

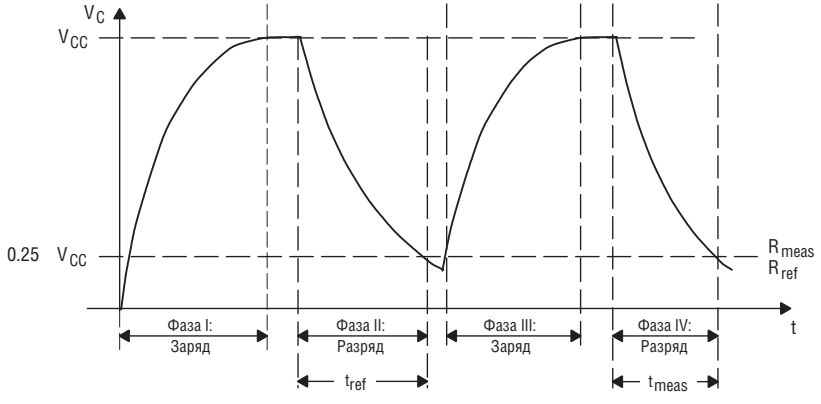


Рис. 16-6. Временная диаграмма систем измерения температуры

$$\frac{N_{meas}}{N_{ref}} = \frac{-R_{meas} \times C \times ln\frac{V_{ref}}{V_{CC}}}{-R_{ref} \times C \times ln\frac{V_{ref}}{V_{CC}}}; \qquad \frac{N_{meas}}{N_{ref}} = \frac{R_{meas}}{R_{ref}};$$

$$R_{meas} = R_{ref} \times \frac{N_{meas}}{N_{ref}}$$

Значения напряжения VCC и емкости конденсатора должны оставаться постоянными во время преобразования, но они не критичны, т.к. исключены из соотношения:

16.3. Регистры компаратора A

Регистры компаратора A приведены в таблице 16-1.

Таблица 16-1. Регистры компаратора A

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Управляющий регистр 1 компаратора A	CACTL1	Чтение/запись	059h	Сброс с POR
Управляющий регистр 2 компаратора A	CACTL2	Чтение/запись	05Ah	Сброс с POR
Отключение порта компаратора A	CAPD	Чтение/запись	05Bh	Сброс с POR

CACTL1, регистр управления 1 компаратора A

7	6	5	4	3	2	1	0
CAEX	CARSEL	CAREFx		CAON	CAIES	CAIE	CAIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

CAEX	**Бит 7**	Обмен в компараторе A. Это бит меняет местами входы компаратора и инвертирует выход компаратора.
CARSEL	**Бит 6**	Выбор опорного источника компаратора A. Этот бит определяет, к какому выводу компаратора прикладывается VCAREF. Когда CAEX=0: 0 – VCAREF прикладывается к выводу «+» 1 – VCAREF прикладывается к выводу «-» Когда CAEX=1: 0 – VCAREF прикладывается к выводу «-» 1 – VCAREF прикладывается к выводу «+»

CAREF	**Биты 5-4**	Опорное напряжение компаратора A. Эти биты выбирают опорное напряжение VCAREF. 00 – Встроенная опора отключена. Может использоваться внешнее опорное напряжение. 01 – 0,25*VCC 10 – 0,50*VCC 11 – Выбирается диодная опора
CAON	**Бит 3**	Включение компаратора A. Этот бит включает компаратор. При выключении компаратора он перестает потреблять ток. Опорная схема включается и выключается независимо от компаратора. 0 – Выключен 1 – Включен
CAIES	**Бит 2**	Выбор фронта прерывания компаратора A 0 – Нарастающий фронт 1 – Спадающий фронт
CAIE	**Бит 2**	Разрешение прерывания от компаратора A 0 – Запрещено 1 – Разрешено
CAIFG	**Бит 0**	Флаг прерывания компаратора A 0 – Прерывание не ожидается 1 – Прерывание ожидается

CACTL2, регистр управления компаратора A

7	6	5	4	3	2	1	0
Не используется				P2CA1	P2CA0	CAF	CAOUT
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)

Не используется	**Бит 7-4**	Не используется
P2CA1	**Бит 3**	Ножка к CA1. Этот бит определяет функцию ножки CA1. 0 – Ножка не подключается к CA1 1 – Ножка подключается к CA1
P2CA0	**Бит 2**	Ножка к CA0. Этот бит определяет функцию ножки CA0. 0 – Ножка не подключается к CA0 1 – Ножка подключается к CA0
CAF	**Бит 1**	Выходной фильтр компаратора A 0 – Выход компаратора A не фильтруется 1 – Выход компаратора A фильтруется

CAOUT	**Бит 0**	Выход компаратора A. Этот бит отражает значение на выходе компаратора. Запись в этот бит не приводит к какому-либо результату.

Регистр CAPD отключения порта компаратора A

7	6	5	4	3	2	1	0
CAPD7	CAPD6	CAPD5	CAPD4	CAPD3	CAPD2	CAPD1	CAPD0
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

CAPDx	**Биты 7-0**	Отключение порта компаратора A. Эти биты индивидуально отключают входной буфер выводов порта, связанных с компаратором A. К примеру, если CAOUT на выводе P2.2, биты CAPDx могут использоваться для индивидуального включения и выключения каждого буфера ножки P2.x. CAPD0 отключает P2.0, CAPD1 отключает P2.1 и т.д. 0 – Входной буфер включен. 1 – Входной буфер отключен.

MSP430x1xxFamily

АЦП12

Раздел XVII.

Texas Instruments

АЦП12

Модуль АЦП12 представляет собой высокоэффективный 12-разрядный аналого-цифровой преобразователь. В этом разделе описывается АЦП12. АЦП12 реализован в устройствах MSP430x13x, MSP430x14x, MSP430x15x и MSP430x16x.

17.1. Введение в АЦП12

Модуль АЦП12 обеспечивает быстрые 12-разрядные аналого-цифровые преобразования. Модуль имеет 12-разрядное ядро SAR, схему выборки, опорный генератор и буфер преобразования и управления объемом 16 слов. Буфер преобразования и управления позволяет получать и сохранять до 16 независимых выборок АЦП без вмешательства ЦПУ.

АЦП12 обладает следующими возможностями:

- Максимальная скорость преобразования свыше 200 ksps
- Монотонный 12-разрядный преобразователь без кодов ошибок
- Выборка и хранение с программируемыми периодами выборки, определяемыми программным обеспечением или таймерами
- Преобразование инициируется программным обеспечением, таймером A или таймером B
- Программно выбираемый интегрированный генератор опорного напряжения (1,5 В или 2,5 В)
- Программно выбираемый внутренний или внешний опорный источник
- Восемь индивидуально конфигурируемых внешних входных каналов
- Каналы преобразования для внутреннего температурного датчика, AVCC и внешних опорных источников
- Независимые опорные источники, задаваемые путем выбора канала, для обоих положительных и отрицательных опорных источников
- Выбираемый источник тактирования преобразований
- Одноканальный, повторный одноканальный, последовательный и повторно-последовательный режимы преобразования
- Ядро АЦП и опорное напряжение могут выключаться раздельно
- Регистр вектора прерываний для быстрого декодирования 18 прерываний АЦП
- 16 регистров хранения результата.

Блок-схема АЦП12 показана на рис. 17-1.

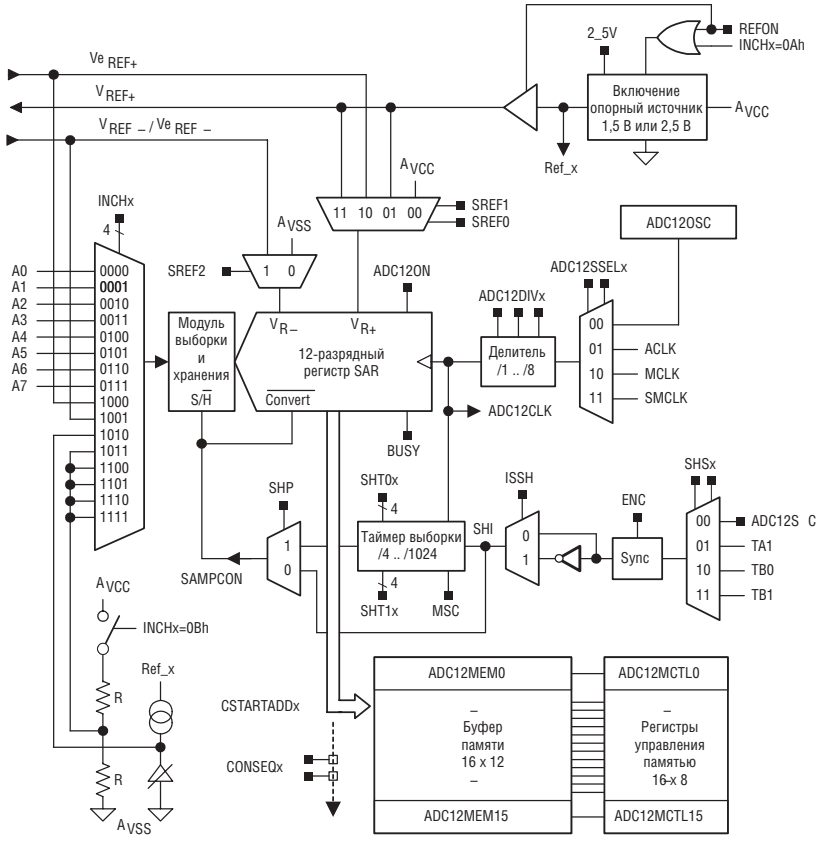


Рис. 17-1. Блок-схема АЦП12

17.2. Функционирование АЦП12

Модуль АЦП12 конфигурируется программным обеспечением пользователя. Настройка и работа АЦП12 рассматриваются в следующих разделах.

17.2.1. 12-разрядное ядро АЦП

Ядро АЦП преобразует аналоговый входной сигнал в 12-разрядное цифровое представление и сохраняет результат в памяти преобразований. Ядро использует два программируемых/выбираемых уровня напряжения (VR+ и VR-) для задания верхнего и нижнего пределов преобразования. На цифровом выходе (NADC) представлена полная шкала (0FFFh), когда входной сигнал равен или

выше VR+, и ноль, когда входной сигнал равен или ниже VR-. Входной канал и опорные уровни напряжения (VR+ и VR-) задаются в памяти управления преобразованиями. Формула преобразования для результата АЦП NADC выглядит следующим образом:

$$N_{ADC} = 4095 \times \frac{V_{in} - V_{R-}}{V_{R-} - V_{R-}}$$

Ядро АЦП12 конфигурируется двумя управляющими регистрами: ADC12CTL0 и ADC12CTL1. Ядро включается битом ADC12ON. Если ADC12 не используется, для сохранения энергии оно может быть выключено. За некоторыми исключениями биты управления АЦП12 могут быть модифицированы, только когда ENC=0. ENC должен быть установлен в 1 перед выполнением любого преобразования.

Выбор тактирования преобразования

ADC12CLK используется как для тактирования преобразования, так и для генерации периода выборки, когда выбран импульсный режим выборки. Для выбора источника тактирования ADC12 используются биты ADC12SSELx, а частота выбранного источника может быть поделена на 1-8 с помощью битов ADC12DIVx. Возможно использование следующих источников ADC12CLK: SMCLK, MCLK, ACLK и внутреннего осциллятора ADC12OSC.

ADC12OSC, генерируемый внутренне, лежит в диапазоне 5 МГц, но варьируется в зависимости от конкретного устройства, напряжения питания и температуры. См. справочное руководство конкретного устройства для уточнения значения ADC12OSC.

Пользователь должен гарантировать, что выбранный источник тактирования для ADC12CLK останется активным до конца преобразования. Если тактовые сигналы будут сняты во время преобразования, операция не будет завершена и любой результат будет неверным.

17.2.2. Входы АЦП12 и мультиплексор

Восемь внешних и четыре внутренних аналоговых сигнала выбираются как канал для преобразования аналоговым входным мультиплексором. Входной мультиплексор имеет тип break-before-make (разрыв перед включением), что уменьшает инжекцию шумов от канала к каналу, возникающую при переключении каналов, как показано на рис. 17-2. Входной мультиплексор также является Т-переключателем, минимизирующим взаимосвязь между каналами. Невыбранные каналы изолированы от АЦП, а промежуточный узел подключен к аналоговой земле (AVSS), поэтому паразитная емкость заземляется, что помогает устранять перекрестные помехи.

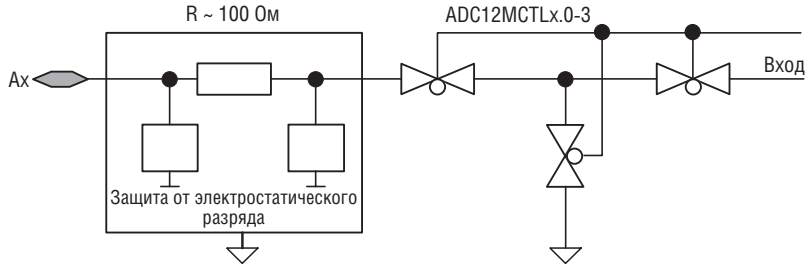


Рис. 17-2. Аналоговый мультиплексор

АЦП12 использует метод перераспределения заряда. Когда входы внутренне переключаются, переключение может привести к переходным процессам на входном сигнале. Эти переходные процессы затухают и устанавливаются до появления ошибочного преобразования.

Выбор аналогового порта

Входы АЦП12 мультиплексированы с ножками порта P6, имеющими цифровые КМОП ячейки. Когда аналоговые сигналы прикладываются к цифровым КМОП-схемам, может течь паразитный ток от VCC к GND. Этот паразитный ток появляется, если величина входного напряжения находится около переходного уровня ячейки. Отключение буфера ножки порта устраняет протекание паразитного тока и вследствие этого уменьшает общий потребляемый ток. Биты P6SELx дают возможность отключать входные и выходные буферы ножки порта.

```
;P6.0 и P6.1 конфигурируются как аналоговые входы
BIS.B #3h,&P6SEL ;P6.1 и P6.0 - функция АЦП12
```

17.2.3. Генератор опорного напряжения

Модуль АЦП12 содержит встроенный генератор опорного напряжения с двумя выбираемыми уровнями напряжения: 1,5 В и 2,5 В. Любое из этих опорных напряжений может быть использовано внутренне или внешне на выводе VREF+.

Установкой REFON=1 включается внутренний опорный источник. Когда REF2_5V=1, внутреннее опорное напряжение равно 2,5 В, при REF2_5V=0 опорное напряжение равно 1,5 В. Если генератор опорного напряжения не используется, он может быть выключен для уменьшения потребления энергии.

Для правильной работы внутреннего генератора опорного напряжения необходимо использовать емкость временного хранения энергии, подключенную между V_{REF+} и AV_{SS}. Рекомендуется в качестве такой емкости использовать комбинацию из включенных параллельно конденсаторов на 10 мкФ и 0,1 мкФ. После включения в течение максимум 17 мС необходимо дать возможность генератору опорного напряжения зарядить конденсаторы хранения энергии. Если внутренний опорный генератор не используется при преобразованиях, конденсаторы не требуются.

Примечание: рекомендация по развязке

*Около 200 мкА необходимы от **любого** опорного источника, используемого АЦП во время определения двух младших бит в течение преобразования. Комбинация из параллельно включенных конденсаторов на 10 мкФ и 0,1 мкФ рекомендуется при использовании **любого** опорного источника, как показано на рис. 17-11.*

Внешние опорные источники могут быть задействованы для VR+ и VR- через выводы Ve_{REF+} и V_{RED-}/Ve_{REF-} соответственно.

17.2.4. Синхронизация выборки и преобразования

Аналого-цифровое преобразование инициируется по нарастающему фронту входного сигнала выборки SHI. Источник для SHI выбирается с помощью битов SHSx и может быть таким:

- Бит ADC12SC
- Модуль вывода 1 таймера A
- Модуль вывода 0 таймера B
- Модуль вывода 1 таймера B

Полярность источника сигнала SHI может быть инвертирована битом ISSH. Сигнал SAMPCON управляет периодом выборки и началом преобразования. Когда SAMPCON имеет высокий уровень, выборка активна. Переход сигнала SAMPCON с высокого уровня на низкий стартует аналого-цифровое преобразование, которому необходимо 13 циклов ADC12CLK. Два различных метода выборки-синхронизации задаются управляющим битом SHP, расширяющим режим выборки и импульсный режим.

Расширенный режим выборки

Расширенный режим выборки выбирается, когда SHP=0. Сигнал SHI напрямую управляет SAMPCON и определяет длительность периода выборки tsample. Когда SAMPCON имеет высокий уровень, выборка активна. Переход сигнала SAMPCON с высокого уровня на низкий стартует преобразование после синхронизации с ADC12CLK. См. рис. 17-3.

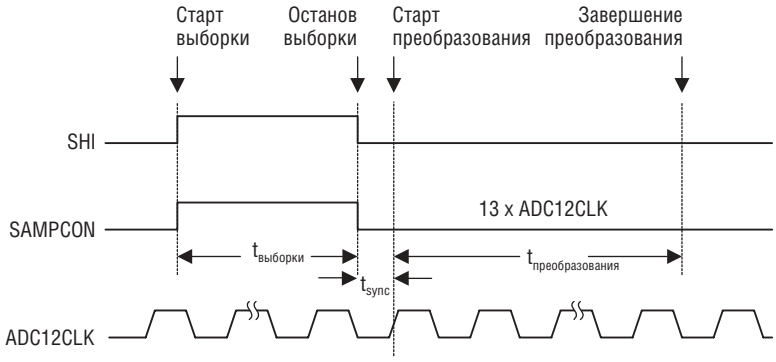


Рис. 17-3. Расширенный режим выборки

Импульсный режим выборки

Импульсный режим выборки выбирается, когда SHP=0. Сигнал SHI используется для запуска таймера выборки. Биты SHT0x и SHT1x в ADC12CTL0 управляют интервалом таймера выборки, который задает период tsampe выборки SAMPCON. Таймер выборки оставляет высокий уровень SAMPCON после синхронизации с ADC12CLK для запрограммированного интервала t_{sampe}. Общее время выборки равно t_{sampe} плюс t_{sync}. См. рис. 17-4.

Биты SHTx устанавливают время выборки в 4 раза больше чем ADC12CLK. SHT0x устанавливает время выборки для ADC12MCTL0-7, а SHT1x устанавливает время выборки для ADC12MCTL8-15.

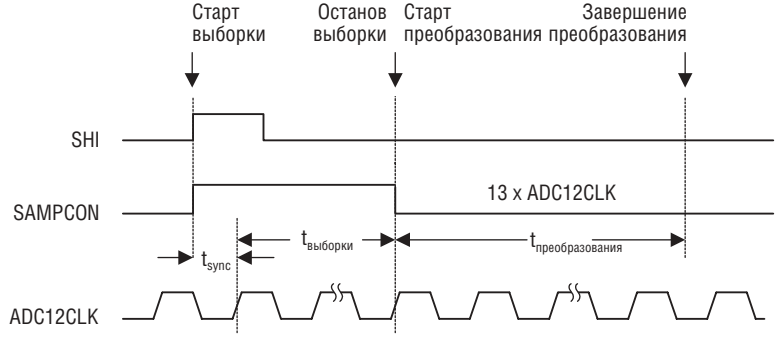


Рис. 17-4. Импульсный режим выборки

Определение длительности выборки

Когда SAMPCON=0, все входы Ax имеют высокое входное сопротивление. Когда SAMPCON=1, выбранный вход Ax можно смоделировать в виде RC-фильтра нижних частот в течение периода квантования t_{sample}, как показано на рис. 17-5. Внутреннее сопротивление R_I (около 2 кОм) мультиплексированного входа последовательно с конденсатором C_I (максимум 40 пФ) представляется источником. Конденсатор C_I должен быть заряжен напряжением V_C в пределах ½ младшего бита источника напряжения V_S для получения точного 12-разрядного преобразования.

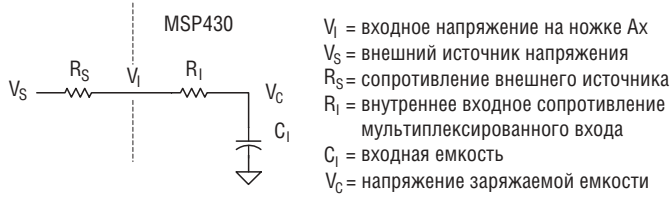


Рис. 17-5. Эквивалентная схема аналогового входа

Сопротивление источника R_S и R_I влияет на t_{sample}. Следующее выражение может быть использовано для вычисления минимального времени выборки tsample при 12-разрядном преобразовании:

$$t_{sample} > (R_S + R_I) \times \ln(2^{13}) \times C_I + 800 \text{ нс}$$

При подстановке значений R_I и C_I, указанных выше, уравнение приобретает следующий вид:

$$t_{sample} > (R_S + 2 \text{ кОм}) \times 9{,}011 \times 40 \text{ пФ} + 800 \text{ нс}$$

К примеру, если R_S равно 10 кОм, t_{sample} должно быть больше 5,13 мкс.

17.2.5. Память преобразований

Результаты преобразований сохраняются в 16-ти регистрах памяти преобразований ADC12MEMx. Каждый регистр ADC12MEMx конфигурируется соответствующим управляющим регистром ADC12MCTLx. Биты SREFx устанавливают опорное напряжение, а биты INCHx задают входной канал. Бит EOS определяет конец последовательности, когда используется последовательный режим преобразования. Следующие друг за другом преобразования последовательно сохраняются в регистрах с ADC12MEM15 по ADC12MEM0, когда бит EOS в ADC12MCTL15 не установлен.

Биты CSTARTADDx определяют первый регистр ADC12MCTLx, используемый для любого преобразования. Если выбраны одноканальный или повторный одноканальный режимы преобразования, CSTARTADDx указывают на единственный ADC12MCTLx, который будет использован.

Если выбран режим преобразования «последовательность каналов» или «повторяющаяся последовательность каналов», CSTARTADDx указывают на расположение ADC12MCTLx, который будет использоваться в последовательности. Программно невидимый указатель автоматически инкрементируется до следующего ADC12MCTLx в последовательности после каждого завершения преобразования. Последовательность продолжается до обработки бита EOS в ADC12MCTLx – это будет обработка последнего управляющего байта.

Когда результат преобразования записывается в выбранный регистр ADC12MEMx, устанавливается соответствующий флаг в регистре ADC12IFGx.

17.2.6. Режимы преобразований АЦП12

АЦП12 имеет четыре режима работы, выбираемые битами CONSEQx так, как описано в таблице 17-1.

Таблица 17-1. Сводный перечень режимов преобразования

CONSEQx	Режим	Операция
00	Одноканальный с одиночным преобразованием	Выполняется одно преобразование в одном канале.
01	Последовательность каналов	Выполняются однократные преобразования последовательности каналов.
10	Повторяющийся одноканальный	Выполняется повторяющееся преобразование в одном канале.
11	Повторяющаяся последовательность каналов	Выполняются повторяющиеся преобразования последовательности каналов.

Одноканальный режим с одиночным преобразованием

В одном канале однократно выполняется выборка и преобразование. Результат АЦП записывается в регистр ADC12MEMx, определенный битами CSTARTADDx. На рис. 17-6 показан процесс одноканального режима с одиночным преобразованием. Если преобразования запускаются ADC12SC, поочередные преобразования могут быть запущены битом ADC12SC. Когда используется другой источник запуска, ENC должен переключаться между каждым преобразованием.

Режим последовательности каналов

В режиме последовательности каналов однократно выполняется выборка и преобразование. Результат АЦП записывается в память преобразований, на-

чиная с ADCMEMx, определенным битами CSTARTADDx. Последовательность останавливается после измерения в канале с установленным битом EOS. На рис. 17-7 показан режим последовательности каналов. Если последователь-

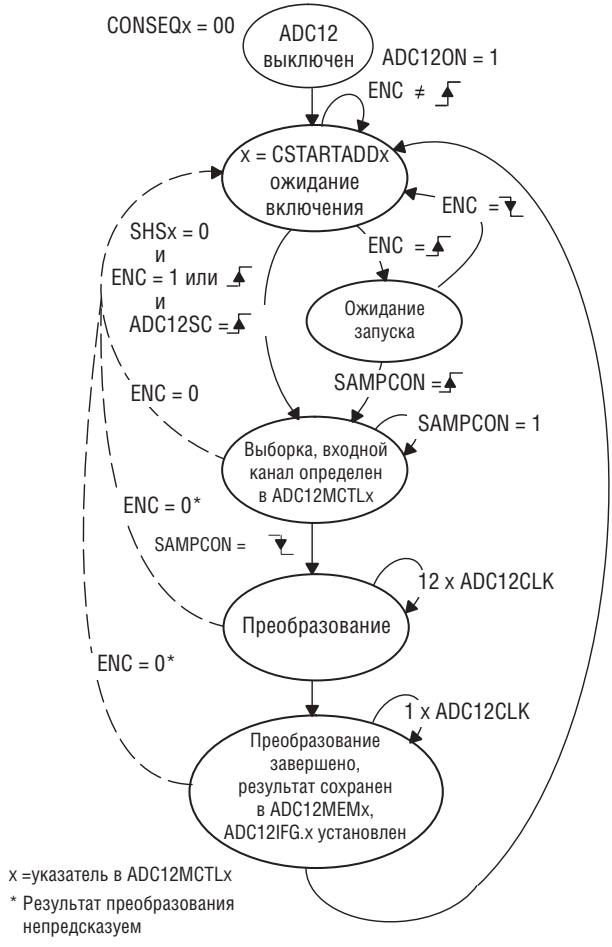


Рис. 17-6. Одноканальный режим одиночного преобразования

ность запускает ADC12SC, поочередные последовательности могут запускаться битом ADC12SC. Когда используется другой источник запуска, ENC должен переключаться между каждой последовательностью.

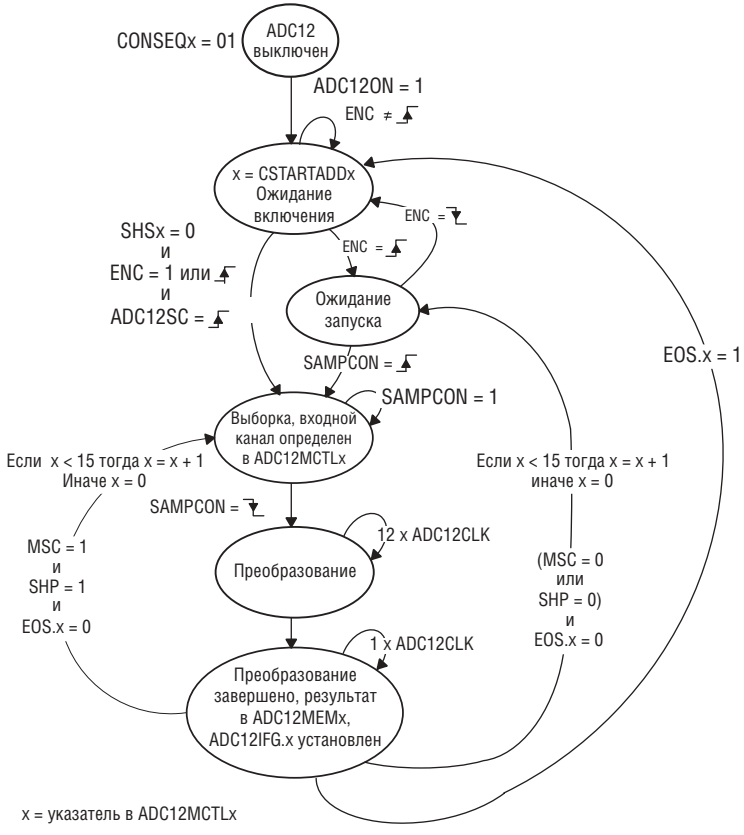


Рис. 17-7. Режим последовательности каналов

Повторяющийся одноканальный режим

В одном канале непрерывно выполняются выборка и преобразование. Результат АЦП записывается в ADC12MEMx, определенный битами CSTARTADDx. Необходимо считывать результат после завершения преобразования, потому что используется только один регистр памяти ADC12MEMx, перезаписываемый с каждым новым преобразованием. На рис. 17-8 показан повторяющийся одноканальный режим.

Режим повторяющейся последовательности каналов

Непрерывно выполняются выборка и преобразование последовательности каналов. Результат АЦП записывается в память преобразований, начиная с

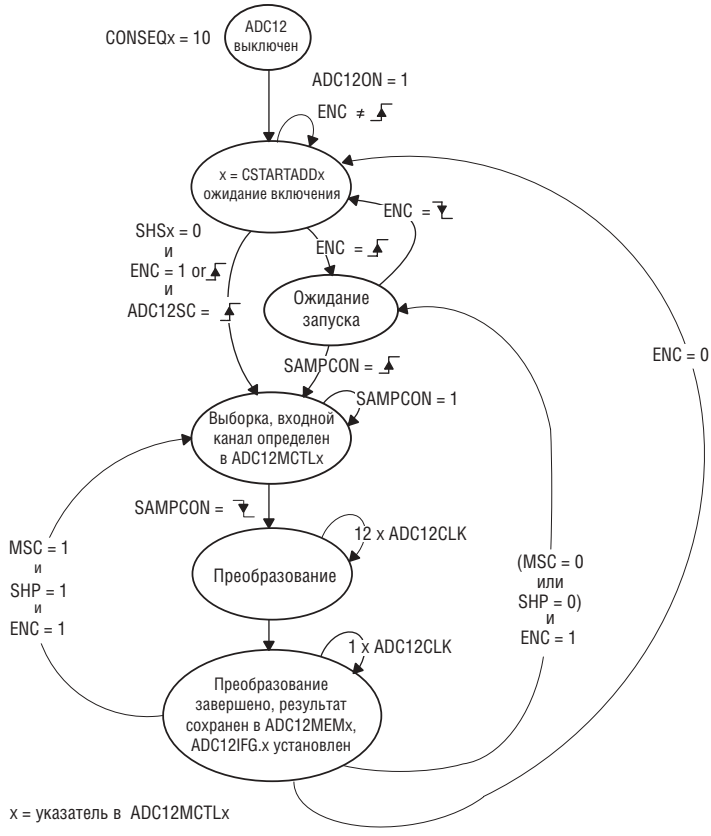


Рис. 17-8. Повторяющийся одноканальный режим

ADC12MEMx, определенного битами CSTARTADDx. Последовательность останавливается после измерения в канале с установленным битом EOS и стартует снова по следующему сигналу запуска. На рис. 17-9 показан режим повторяющейся последовательности каналов.

Использование бита множественных выборок и преобразований (MSC)

Для конфигурирования преобразователя на выполнение автоматических поочередных преобразований с максимальной быстротой можно воспользоваться функцией множественных выборок и преобразований. Если MSC=1, CONSEQx>1 и используется таймер выборок, первый фронт сигнала SHI за-

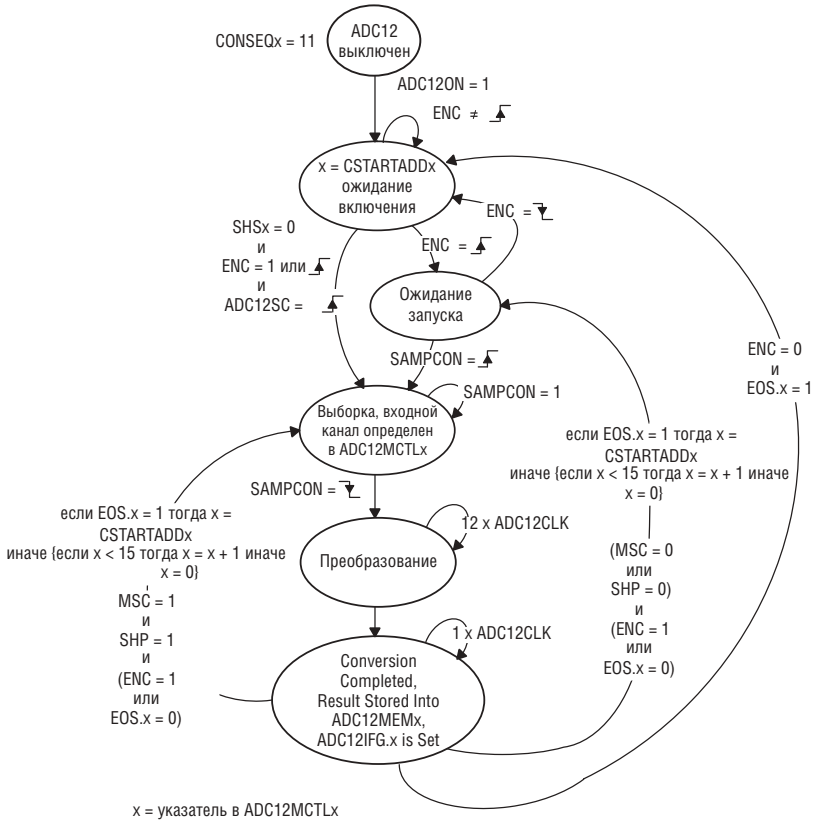


Рис. 17-9. Режим повторяющейся последовательности каналов

пустит первое преобразование. Очередные преобразования запускаются автоматически после завершения предыдущего преобразования. Дополнительные фронты на SHI игнорируются, пока последовательность не закончена или пока бит ENC не переключен в повторяющийся одноканальный режим или повторяющийся режим последовательностей. Функция бита ENC не изменяется, пока используется бит MSC.

Останов преобразований

Прекращение активности АЦП12 зависит от режима работы. Рекомендуются следующие способы останова активного преобразования или последовательности преобразований:

- Сброс ENC в одноканальном режиме одиночного преобразования немедленно останавливает преобразование, при этом результат оказывается непредсказуемым. Для получения правильного результата необходимо опрашивать бит занятости до сброса перед очисткой ENC.
- Сброс ENC во время повторяющегося одноканального преобразования останавливает преобразователь в конце текущего преобразования.
- Сброс ENC во время последовательного или повторно-последовательного режимов останавливает преобразователь в конце последовательности.
- Любой режим преобразования может быть немедленно остановлен установкой CONSEQx=0 и сбросом бита ENC. Данные преобразования будут ненадежны.

Примечание: Отсутствие установленного бита EOS для последовательности

Если установленного бита EOS нет и выбран режим последовательностей, сброс бита ENC не приведет к останову последовательности. Для останова последовательности сначала нужно выбрать одноканальный режим, а затем сбросить ENC.

17.2.7. Использование интегрированного температурного датчика

При использовании имеющегося на кристалле температурного датчика пользователь выбирает аналоговый входной канал INCHx=1010. Любая другая конфигурация рассматривается как выбор внешнего канала, включая выбор опорного источника, выбор памяти преобразований и т.д.

Типичная передаточная функция температурного датчика показана на рис. 17-10. Если используется температурный датчик, период выборки должен быть больше 30 мкС. Ошибка смещения температурного датчика может быть большой и может потребоваться калибровка для большинства приложений. См. справочные данные конкретного устройства для выяснения подробностей.

При выборе температурного датчика автоматически запускается расположенный на кристалле опорный генератор в качестве источника напряжения для температурного датчика. Однако это не включает выход V_{REF+} и не влияет на выбор опорного источника для преобразования. Процедура выбора источника для преобразования информации с температурного датчика подобна процедуре выбора для любого другого канала.

17.2.8. Заземление АЦП12 и рассмотрение влияния помех

Как в любом АЦП с высоким разрешением, для устранения нежелательных паразитных эффектов и шумов, а также предотвращения возникновения

паразитных контуров с замыканием на землю, необходимы особая разводка печатной платы и особые методы заземления.

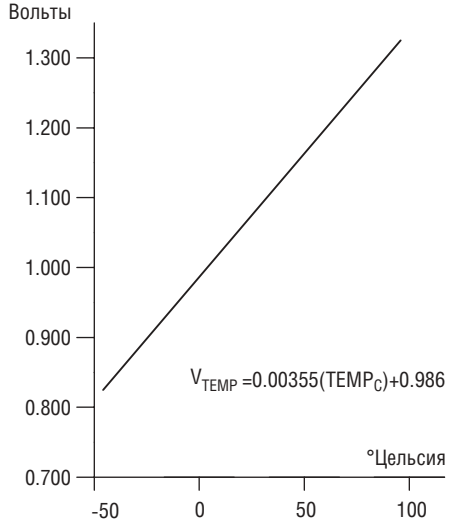


Рис. 17-10. Типичная передаточная функция температурного датчика

Паразитные общие петли формируются, когда ток возврата от АЦП проходит совместно с токами других аналоговых и цифровых схем. Если не принимать специальных мер, этот ток может генерировать нежелательные напряжения смещения, которые могут прибавляться или вычитаться из опорного или входного напряжений аналого-цифрового преобразователя. Способ подключения, показанный на рис. 17-11 позволяет этого избежать.

В дополнение к заземлению, пульсации и шумовые выбросы на линиях источника питания, вызванные переключениями цифровых схем или переключениями в источнике питания могут повредить результат преобразования. Для получения высокой точности рекомендуется создавать разработки, свободные от шумов, что достигается разделением аналоговых и цифровых контуров земли с соединением их в одной точке.

17.2.9. Прерывания АЦП12

АЦП12 имеет 18 источников прерывания:

- ADC12IFG0-ADC12IFG15

- ADC12OV, переполнение AD12MEMx
- ADC12TOV, переполнение времени преобразования АЦП12

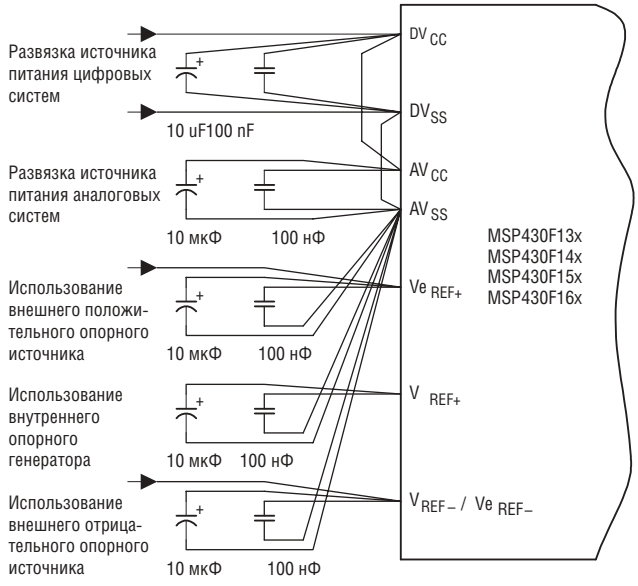


Рис. 17-11. Заземление АЦП12 и устранение помех

Биты ADC12IFGx устанавливаются, когда в их соответствующие регистры памяти ADC12MEMx загружается результат преобразования. Если соответствующий бит ADC12IEx и бит GIE установлены, генерируется запрос прерывания. Состояние ADC12OV появляется, когда результат преобразования записывается в любой регистр ADC12MEMx до прочтения предыдущего результата. Состояние ADC12TOV генерируется, когда до завершения текущего преобразования затребована другая выборка-преобразование.

ADC12IV, генератор вектора прерываний

Все источники прерываний АЦП12 разделены по приоритетам и являются источником одного вектора прерываний. Регистр вектора прерываний ADC12IV используется для определения, какой разрешенный источник прерываний АЦП12 запрашивает прерывание.

Разрешенное прерывание АЦП12 с наивысшим приоритетом генерирует число в регистре ADC12IV (см. описание регистра). Это число может быть оценено или добавлено к программному счетчику для автоматического входа в соответствующую программную процедуру. Запрещенные прерывания АЦП12 не влияют на значение ADC12IV.

При любом типе доступа (чтение или запись), регистр ADC12IV автоматически сбрасывает состояние ADC12OV или состояние ADC12TOV, если любое из них было наивысшим ожидающим прерыванием. Никакое состояние прерывания не имеет доступного флага прерывания. Флаги ADC12IFGx не сбрасываются при доступе к ADC12IV. Биты ADC12IFGx сбрасываются автоматически при доступе к их соответствующим регистрам ADC12MEMx или же могут быть сброшены программно.

Если после обработки текущего прерывания ожидается другое прерывание, генерируется другое прерывание. К примеру, если ожидается обработка прерываний ADC12OV и ADC12IFG3, когда процедура обработки прерывания обращается к регистру ADC12IV, состояние прерывания ADC12OV автоматически сбрасывается. После выполнения команды RETI процедуры обработки прерывания ADC12IFG3 генерирует другое прерывание.

Пример программы-обработчика прерываний АЦП12

Приведенный далее пример программного обеспечения показывает рекомендуемое использование ADC12IV и временные затраты на обработку. Значение ADC12IV добавляется к PC для автоматического перехода к соответствующей процедуре.

Числа в правом поле показывают необходимое для каждой команды количество циклов ЦПУ. Программные затраты для различных источников включают время задержки прерывания и циклы возврата из прерывания, но не обработку собственно задачи. Задержки таковы:

- ADC12IFG0-ADC12IFG14, ADC12TOV и ADC12OV — 16 циклов
- ADC12IFG15 — 14 циклов

Обработчик прерывания для ADC12IFG15 показывает путь к немедленной проверке, если произошло прерывание с наивысшим приоритетом во время обработки ADC12IFG15. Это позволяет сэкономить девять циклов, если ожидается другое прерывание АЦП12.

```
;Обработчик прерывания для АЦП12.
INT_ADC12           ;Вход процедуры обработки прерывания        6
    ADD&ADC12IV,PC  ;Добавление смещения к PC                   3
    RETI            ;Вектор 0: Нет прерывания                   5
    JMPADOV         ;Вектор 2: Переполнение АЦП                 2
    JMPADTOV        ;Вектор 4: Переполнение тактирования АЦП    2
    JMPADM0         ;Вектор 6: ADC12IFG0                        2
    ...             ;Векторы 8-32                               2
    JMPADM14        ;Вектор 34: ADC12IFG14                      2
;
;Обработчик ADC12IFG15 стартует здесь. JMP не требуется.
;
ADM15 MOV &ADC12MEM15, xxx ;Перемещение результата, флаг сброшен
      ...                  ;Другая команда необходима?
      JMP INT_ADC12        ;Проверка другого ожидаемого прерывания
;
;Обработчик ADC12IFG14-ADC12IFG1 запускается здесь
ADM0 MOV &ADC12MEM0, xxx   ;Перемещение результата, флаг сброшен
;
     ...                   ;Другая команда необходима?
     RETI                  ;Возврат                            5
;
ADTOV ...                  ;Обработка переполнения времени
                           ;преобразования
      RETI                 ;Возврат                            5
;
ADOV ...                   ;Обработка переполнения ADCMEMx
     RETI                  ;Возврат                            5
```

17.3. Регистры АЦП12

Регистры АЦП12 приведены в таблице 17-2.

Таблица 17-2. Регистры АЦП12

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Управляющий регистр 0 АЦП12	ADC12CTL0	Чтение/запись	01A0h	Сброс с POR
Управляющий регистр 1 АЦП12	ADC12CTL1	Чтение/запись	01A2h	Сброс с POR
Регистр флагов прерываний АЦП12	ADC12IFG	Чтение/запись	01A4h	Сброс с POR
Регистр разрешения прерываний АЦП12	ADC12IE	Чтение/запись	01A6h	Сброс с POR
Слово вектора прерываний АЦП12	ADC12IV	Чтение	01A8h	Сброс с POR
Регистр памяти 0 АЦП12	ADC12MEM0	Чтение/запись	0140h	Не изменяется
Регистр памяти 1 АЦП12	ADC12MEM1	Чтение/запись	0142h	Не изменяется
Регистр памяти 2 АЦП12	ADC12MEM2	Чтение/запись	0144h	Не изменяется
Регистр памяти 3 АЦП12	ADC12MEM3	Чтение/запись	0146h	Не изменяется
Регистр памяти 4 АЦП12	ADC12MEM4	Чтение/запись	0148h	Не изменяется
Регистр памяти 5 АЦП12	ADC12MEM5	Чтение/запись	014Ah	Не изменяется
Регистр памяти 6 АЦП12	ADC12MEM6	Чтение/запись	014Ch	Не изменяется
Регистр памяти 7 АЦП12	ADC12MEM7	Чтение/запись	014Eh	Не изменяется
Регистр памяти 8 АЦП12	ADC12MEM8	Чтение/запись	0150h	Не изменяется
Регистр памяти 9 АЦП12	ADC12MEM9	Чтение/запись	0152h	Не изменяется
Регистр памяти 10 АЦП12	ADC12MEM10	Чтение/запись	0154h	Не изменяется
Регистр памяти 11 АЦП12	ADC12MEM11	Чтение/запись	0156h	Не изменяется
Регистр памяти 12 АЦП12	ADC12MEM12	Чтение/запись	0158h	Не изменяется
Регистр памяти 13 АЦП12	ADC12MEM13	Чтение/запись	015Ah	Не изменяется
Регистр памяти 14 АЦП12	ADC12MEM14	Чтение/запись	015Ch	Не изменяется
Регистр памяти 15 АЦП12	ADC12MEM15	Чтение/запись	015Eh	Не изменяется
Управление регистром памяти 0 АЦП12	ADC12MCTL0	Чтение/запись	080h	Сброс с POR
Управление регистром памяти 1 АЦП12	ADC12MCTL1	Чтение/запись	081h	Сброс с POR
Управление регистром памяти 2 АЦП12	ADC12MCTL2	Чтение/запись	082h	Сброс с POR
Управление регистром памяти 3 АЦП12	ADC12MCTL3	Чтение/запись	083h	Сброс с POR
Управление регистром памяти 4 АЦП12	ADC12MCTL4	Чтение/запись	084h	Сброс с POR
Управление регистром памяти 5 АЦП12	ADC12MCTL5	Чтение/запись	085h	Сброс с POR
Управление регистром памяти 6 АЦП12	ADC12MCTL6	Чтение/запись	086h	Сброс с POR
Управление регистром памяти 7 АЦП12	ADC12MCTL7	Чтение/запись	087h	Сброс с POR
Управление регистром памяти 8 АЦП12	ADC12MCTL8	Чтение/запись	088h	Сброс с POR
Управление регистром памяти 9 АЦП12	ADC12MCTL9	Чтение/запись	089h	Сброс с POR
Управление регистром памяти 10 АЦП12	ADC12MCTL10	Чтение/запись	08Ah	Сброс с POR
Управление регистром памяти 11 АЦП12	ADC12MCTL11	Чтение/запись	08Bh	Сброс с POR
Управление регистром памяти 12 АЦП12	ADC12MCTL12	Чтение/запись	08Ch	Сброс с POR
Управление регистром памяти 13 АЦП12	ADC12MCTL13	Чтение/запись	08Dh	Сброс с POR
Управление регистром памяти 14 АЦП12	ADC12MCTL14	Чтение/запись	08Eh	Сброс с POR
Управление регистром памяти 15 АЦП12	ADC12MCTL15	Чтение/запись	08Fh	Сброс с POR

ADC12CTL0, управляющий регистр 0 АЦП12

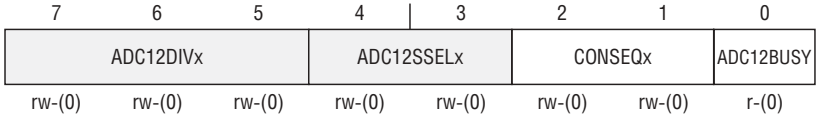
15	14	13	12	11	10	9	8
SHT1x				SHT0x			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
MSC	REF2_5V	REFON	ADC12ON	ADC12OVIE	ADC12TOVIE	ENC	ADC12SC
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Модифицируется, только когда ENC = 0

SHT1x	**Биты 15-12**	Время выборки-хранения. Эти биты определяют число циклов ADC12CLK в периоде выборки для регистров с ADC12MEM8 по ADC12MEM15.	
SHT0x	**Биты 11-8**	Время выборки-хранения. Эти биты определяют число циклов ADC12CLK в периоде выборки для регистров с ADC12MEM0 по ADC12MEM7.	
		Биты SHTx	Циклы ADC12CLK
		0000	4
		0001	8
		0010	16
		0011	32
		0100	64
		0101	96
		0110	128
		0111	192
		1000	256
		1001	384
		1010	512
		1011	768
		1100	1024
		1101	1024
		1110	1024
		1111	1024

MSC	**Бит 7**	Множественная выборка и преобразование. Справедливо только для последовательных или повторных режимов. 0 – Для запуска каждой выборки-преобразования на таймер выборки подается фронт сигнала SHI 1 – Первый фронт сигнала SHI запускает таймер выборки, последующие выборки-преобразования выполняются автоматически, сразу же после завершения предыдущего преобразования
REF2_5V	**Бит 6**	Генератор опорного напряжения. REFON также должен быть установлен. 0 – 1.5 В 1 – 2.5 В
REFON	**Бит 5**	Включение опорного генератора. 0 – Опорный генератор выключен 1 – Опорный генератор включен.
ADC12ON	**Бит 4**	Включение АЦП12 0 – АЦП12 выключен 1 – АЦП12 включен
ADC12OVIE	**Бит 3**	Разрешение прерывания по переполнению ADC12MEMx. Для разрешения прерываний также должен быть установлен бит GIE. 0 – Прерывание по переполнению запрещено 1 – Прерывание по переполнению разрешено
ADC12TOVIE	**Бит 2**	Разрешение прерывания по превышению времени преобразования АЦП12. Для разрешения прерываний также должен быть установлен бит GIE. 0 – Прерывание по превышению времени преобразования запрещено 1 – Прерывание по превышению времени преобразования разрешено
ENC	**Бит 1**	Разрешение преобразования 0 – Преобразование в АЦП12 запрещено 1 – Преобразование в АЦП12 разрешено
ADC12SC	**Бит 0**	Запуск преобразования. Программно управляемый старт выборки-преобразования. ADC12SC и ENC могут быть установлены вместе в одной команде. ADC12SC сбрасывается автоматически. 0 – Нет старта выборки-преобразования 1 – Старт выборки-преобразования

ADC12CTL1, управляющий регистр 1 АЦП12



Модифицируется, только когда ENC = 0

CSTARTADDx	**Биты 15-12**	Стартовый адрес преобразования. Эти биты позволяют выбрать, какой регистр памяти преобразований АЦП12 используется для одиночного преобразования или для первого преобразования в последовательности. Значение в CSTARTADDx может быть от 0 до 0Fh, что соответствует регистрам с ADC12MEM0 по ADC12MEM15.
SHSx	**Биты 11-10**	Выбор источника выборки-хранения. 00 – Бит ADC12SC 01 – Выход 1 Таймера A 10 – Выход 0 Таймера B 11 – Выход 1 Таймера B
SHP	**Бит 9**	Выбор импульсного режима выборки-хранения. Этот бит выбирает источник сигнала выборки (SAMPCON), либо как выход таймера выборки, либо как прямой входной сигнал выборки. 0 – Источником сигнала SAMPCON является входной сигнал выборки. 1 – Источником сигнала SAMPCON является таймер выборки.
ISSH	**Бит 8**	Инвертирование сигнала выборки-хранения 0 – Входной сигнал выборки не инвертирован 1 – Входной сигнал выборки инвертирован
ADC12DIVx	**Биты 7-5**	Тактовый делитель АЦП12 000 – /1 001 – /2 010 – /3 011 – /4 100 – /5 101 – /6 110 – /7 111 – /8
ADC12SSELx	**Биты 4-3**	Выбор источника тактирования АЦП12 00 – ADC12OSC 01 – ACLK 10 – MCLK 11 – SMCLK
CONSEQx	**Биты 2-1**	Выбор режима преобразования 00 – Одноканальный, с одним преобразованием 01 – Последовательность каналов 10 – Повторный одноканальный 11 – Повторяющаяся последовательность каналов

ADC12BUSY	**Бит 0**	Занятость АЦП12. Этот бит показывает активность операции выборки и преобразования. 0 – Действия не выполняются 1 – Выполняется последовательность, выборка или преобразование

ADC12MEMx, регистры памяти преобразований АЦП12

15	14	13	12	11	10	9	8
0	0	0	0	Результаты преобразования			
r0	r0	r0	r0	rw	rw	rw	rw

7	6	5	4	3	2	1	0
Результаты преобразования							
rw	rw	rw	rw	rw	rw	rw	rw

Результаты преобразования	**Биты 15-0**	12-разрядные результаты преобразования выравниваются по правому краю. Бит 11 является старшим битом MSB. Биты 15-12 всегда равны 0. Запись в регистры памяти преобразований повредит результаты.

ADC12MCTLx, управляющие регистры памяти преобразований АЦП12

7	6	5	4	3	2	1	0
EOS	SREFx			INCHx			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Модифицируется, только когда ENC = 0

EOS	**Биты 7**	Конец последовательности. Показывает последнее преобразование в последовательности. 0 – Не конец последовательности 1 – Конец последовательности
SREFx	**Биты 6-4**	Выбор опорного источника 000 – $V_{R+} = AV_{CC}$ и $V_{R-} = AV_{SS}$ 001 – $V_{R+} = V_{REF}+$ и $V_{R-} = AV_{SS}$ 010 – $V_{R+} = Ve_{REF+}$ и $V_{R-} = AV_{SS}$ 011 – $V_{R+} = Ve_{REF+}$ и $V_{R-} = AV_{SS}$ 100 – $V_{R+} = AV_{CC}$ и $V_{R-} = V_{REF-}/Ve_{REF-}$ 101 – $V_{R+} = V_{REF+}$ и $V_{R-} = V_{REF-}/Ve_{REF-}$ 110 – $V_{R+} = Ve_{REF+}$ и $V_{R-} = V_{REF-}/Ve_{REF-}$ 111 – $V_{R+} = Ve_{REF+}$ и $V_{R-} = V_{REF-}/Ve_{REF-}$

INCHx	**Биты 3-0**	Выбор входного канала 0000 – A0 0001 – A1 0010 – A2 0011 – A3 0100 – A4 0101 – A5 0110 – A6 0111 – A7 1000 – Ve_{REF+} 1001 – V_{REF-}/Ve_{REF-} 1010 – Температурный диод 1011 – $(AV_{CC} - AV_{SS})/2$ 1100 – $(AV_{CC} - AV_{SS})/2$ 1101 – $(AV_{CC} - AV_{SS})/2$ 1110 – $(AV_{CC} - AV_{SS})/2$ 1111 – $(AV_{CC} - AV_{SS})/2$

ADC12IE, регистр разрешения прерываний АЦП12

15	14	13	12	11	10	9	8
ADC12IE15	ADC12IE14	ADC12IE13	ADC12IE12	ADC12IE11	ADC12IE10	ADC12IE9	ADC12IE8
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
ADC12IE7	ADC12IE6	ADC12IE5	ADC12IE4	ADC12IE3	ADC12IE2	ADC12IE1	ADC12IE0
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

ADC12IEx	**Биты 15-0**	Разрешение прерывания. Эти биты разрешают или запрещают запрос прерывания для битов ADC12IFGx. 0 – Прерывание запрещено 1 – Прерывание разрешено

ADC12IFG, регистр флагов прерываний АЦП12

15	14	13	12	11	10	9	8
ADC12IFG15	ADC12IFG14	ADC12IFG13	ADC12IFG12	ADC12IFG11	ADC12IFG10	ADC12IFG9	ADC12IFG8
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
ADC12IFG7	ADC12IFG6	ADC12IFG5	ADC12IFG4	ADC12IFG3	ADC12IFG2	ADC12IFG1	ADC12IFG0
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

ADC12IFGx	**Биты 15-0**	Флаг прерывания ADC12MEMx. Эти биты устанавливаются, когда в соответствующий регистр ADC12MEMx загружается результат преобразования. Биты ADC12IFGx сбрасываются, если выполняется доступ к соответствующим регистрам ADC12MEMx или же могут быть сброшены программно. 0 – Прерывание не ожидается 1 – Прерывание ожидается

ADC12IV, регистр вектора прерываний АЦП12

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
r0	r0	r0	r0	r0	r0	r0	r0

7	6	5	4	3	2	1	0
0	0	ADC12IVx					
r0	r0	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r0

ADC12IVx	**Биты 15-0**	Значение вектора прерываний АЦП12			
		Содержимое ADC12IV	Источник прерывания	Флаг прерывания	Приоритет прерывания
		000h	Прерывание не ожидается	–	
		002h	Переполнение ADC12MEMx	–	Наивысший
		004h	Превышение времени преобразования	–	
		006h	Флаг прерывания ADC12MEM0	ADC12IFG0	
		008h	Флаг прерывания ADC12MEM1	ADC12IFG1	
		00Ah	Флаг прерывания ADC12MEM2	ADC12IFG2	
		00Ch	Флаг прерывания ADC12MEM3	ADC12IFG3	
		00Eh	Флаг прерывания ADC12MEM4	ADC12IFG4	
		010h	Флаг прерывания ADC12MEM5	ADC12IFG5	

		012h	Флаг прерывания ADC12MEM6	ADC12IFG6	
		014h	Флаг прерывания ADC12MEM7	ADC12IFG7	
		016h	Флаг прерывания ADC12MEM8	ADC12IFG8	
		018h	Флаг прерывания ADC12MEM9	ADC12IFG9	
		01Ah	Флаг прерывания ADC12MEM10	ADC12IFG10	
		01Ch	Флаг прерывания ADC12MEM11	ADC12IFG11	
		01Eh	Флаг прерывания ADC12MEM12	ADC12IFG12	
		020h	Флаг прерывания ADC12MEM13	ADC12IFG13	
		022h	Флаг прерывания ADC12MEM14	ADC12IFG14	
		024h	Флаг прерывания ADC12MEM15	ADC12IFG15	Низший

MSP430x1xxFamily

АЦП10

Раздел XVIII.

Texas Instruments

АЦП10

Модуль АЦП10 представляет собой высокоэффективный 10-разрядный аналого-цифровой преобразователь. В этом разделе описывается АЦП10. АЦП10 реализован в устройствах MSP430x11x2 и MSP430x12x2.

18.1. Введение в АЦП10

Модуль АЦП10 обеспечивает быстрые 10-разрядные аналого-цифровые преобразования. Модуль имеет 10-разрядное ядро с регистром последовательного приближения SAR, схему выборки, опорный генератор и контроллер переноса данных (DTC).

DTC позволяет выборкам АЦП10 преобразовываться и сохраняться в любом месте памяти без вмешательства ЦПУ. Модуль может конфигурироваться программным обеспечением пользователя для поддержки разнообразных приложений.

АЦП10 обладает следующими возможностями:

- Максимальная скорость преобразования свыше 200 ksps (200000 преобразований в сек.)
- Монотонный 10-разрядный преобразователь без ошибочных кодов
- Выборка и хранение с программируемыми периодами выборки
- Преобразование инициируется программным обеспечением или таймером A
- Программно выбираемый интегрированный генератор опорного напряжения (1,5 В или 2,5 В)
- Программно выбираемый внутренний или внешний опорный источник
- Восемь индивидуально конфигурируемых внешних входных каналов
- Каналы преобразования для внутреннего температурного датчика, AV_{CC} и внешних опорных источников
- Выбираемый источник тактирования преобразований
- Одноканальный, повторный одноканальный, последовательный и повторно-последовательный режимы преобразования
- Ядро АЦП и опорное напряжение могут выключаться раздельно
- Контроллер переноса данных для автоматического сохранения результатов преобразований

Блок-схема АЦП10 показана на рис. 18-1.

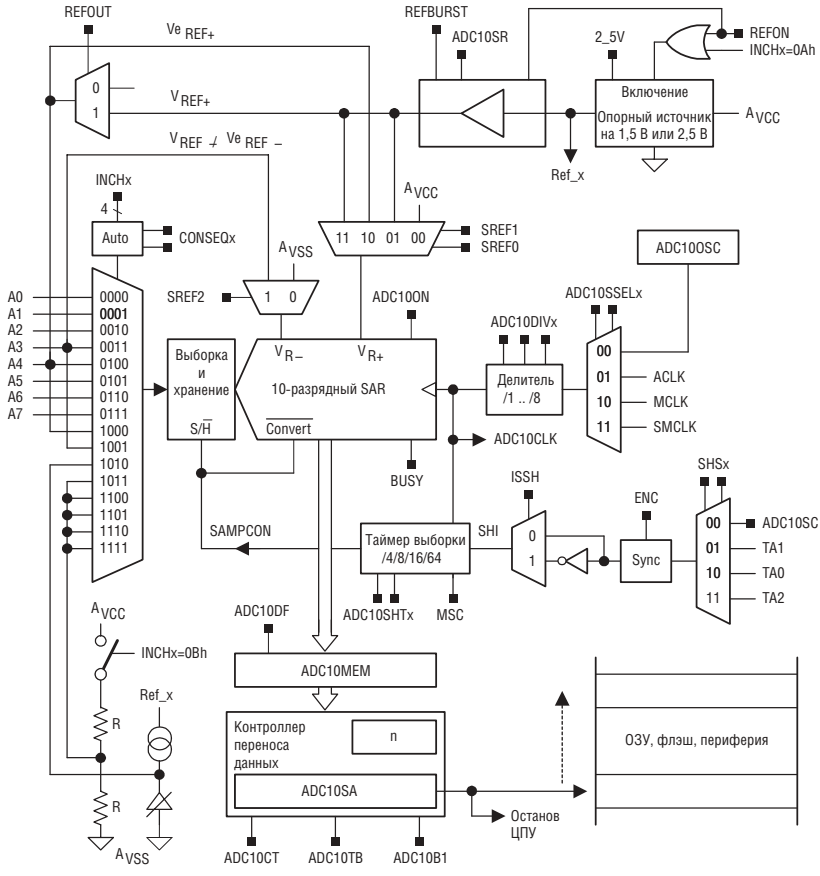


Рис. 18-1. Блок-схема АЦП10

18.2. Функционирование АЦП10

Модуль АЦП10 конфигурируется программным обеспечением пользователя. Настройка и работа АЦП10 рассматриваются в следующих далее разделах.

18.2.1. 10-разрядное ядро АЦП

Ядро АЦП преобразует аналоговый входной сигнал в 10-разрядное цифровое представление и сохраняет результат в регистре ADC10MEM. Ядро использует два программно выбираемых уровня напряжений (V_{R+} и V_{R-}) для задания верхнего и нижнего пределов преобразования. Цифровой выход (N_{ADC})

составляет полную шкалу (03FFh), когда входной сигнал равен или выше V_{R+}, и равен нулю, когда входной сигнал равен или ниже V_{R-}. Входной канал и опорные уровни напряжений (V_{R+} и V_{R-}) задаются в памяти управления преобразованиями. Результаты преобразования могут быть представлены в натуральном двоичном формате или формате с дополнением до двух. Формула преобразования для результата АЦП N_{ADC} с использованием натурального двоичного формата выглядит следующим образом:

$$N_{ADC} = 1023 \times \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}}$$

Ядро АЦП12 конфигурируется двумя управляющими регистрами: ADC10CTL0 и ADC10CTL1. Ядро включается битом ADC10ON. За некоторыми исключениями биты управления АЦП10 могут быть модифицированы только когда ENC=0. ENC должен быть установлен в 1 перед выполнением любого преобразования.

Выбор тактирования преобразования

ADC10CLK используется как для тактирования преобразования, так и для генерации периода выборки. Для выбора источника тактирования АЦП10 используются биты ADC10SSELx, а частота этого источника может быть поделена на 1-8 с помощью битов ADC10DIVx. Возможны следующие источники ADC10CLK: SMCLK, MCLK, ACLK и внутренний осциллятор ADC10OSC.

Внутренне генерируемая частота ADC10OSC лежит в диапазоне 5 МГц, на варьируется в зависимости от конкретного устройства, напряжения питания и температуры. См. справочное руководство конкретного устройства для уточнения значения ADC10OSC.

Пользователь должен гарантировать, что выбранный источник тактирования для ADC10CLK останется активным до конца преобразования. Если тактовые сигналы будут сняты во время преобразования, операция не будет завершена и любой результат будет неверным.

18.2.2. Входы АЦП10 и мультиплексор

Восемь внешних и четыре внутренних аналоговых сигнала выбираются как канал для преобразования входным аналоговым мультиплексором. Входной мультиплексор имеет тип break-before-make (разрыв перед включением), что уменьшает инжекцию шумов от канала к каналу, возникающую при переключении каналов, как показано на рис. 18-2. Входной мультиплексор также является Т-переключателем, минимизирующим взаимосвязь между каналами. Невыбранные каналы изолированы от АЦП, а промежуточный узел подключен к аналоговой земле (AV_{SS}), поэтому паразитная емкость заземляется, что помогает устранить перекрестные помехи.

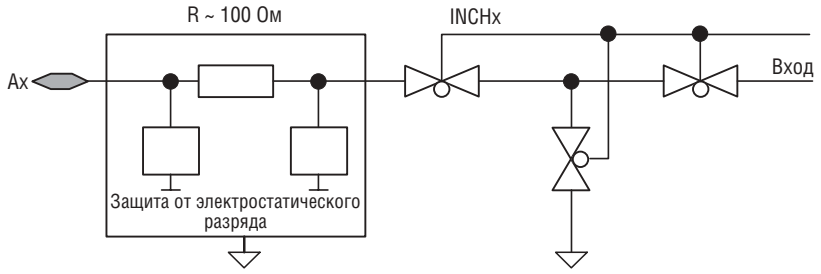


Рис. 18-2. Аналоговый мультиплексор

АЦП10 использует метод перераспределения заряда. Когда входы внутренне переключаются, переключение может привести к переходным процессам на входном сигнале. Эти переходные процессы затухают и устанавливаются до появления ошибочного преобразования.

Выбор аналогового порта

Внешние входы АЦП10 с A0 по A4, Ve_{REF+} и Ve_{REF-} мультиплексированы с ножками порта P2, являющимися цифровые КМОП ячейками. Опциональные входы с A5 по A7 являются у некоторых устройств общими с портом P3 (см. справочные данные конкретного устройства). Когда аналоговые сигналы прикладываются к цифровым КМОП-схемам, может течь паразитный ток от VCC к GND. Этот паразитный ток появляется, когда величина входного напряжения находится около переходного уровня ячейки. Отключение буфера ножки порта устраняет протекание паразитного тока и вследствие этого уменьшает общий потребляемый ток. Биты ADC10AEx дают возможность отключать входные и выходные буфера ножек порта.

```
;P2.3 конфигурируется как аналоговый вход
      BIS.B #4h,&ADC10AE    ;P2.3 включен как функция АЦП10
```

18.2.3. Генератор опорного напряжения

Модуль АЦП10 содержит встроенный генератор опорного напряжения с двумя выбираемыми уровнями напряжения: 1,5 В и 2,5 В. Любое из этих опорных напряжений может быть использовано внутренне или внешне на выводе V_{REF+}.

Внутренний опорный источник включается при установке REFON=1. Когда REF2_5V=1, внутреннее опорное напряжение равно 2,5 В, при REF2_5V=0 опорное напряжение равно 1,5 В.

Внешние опорные источники могут быть задействованы для V_{R+} и V_{R-} через выводы A4 и A3 соответственно.

Маломощные приложения

Внутренний генератор опорного напряжения АЦП10 разработан для маломощных приложений и имеет особые возможности для быстрого запуска. Для правильной работы не требуется внешнего накопительного конденсатора и связанного с ним времени смещения. Общее время включения опорного источника меньше 30 мкС. Для нормальной развязки источника питания требуется только комбинация параллельно включенных конденсаторов на 10 мкФ и 100 нФ.

- Когда V_{CC} и V_{SS} используются как опорные напряжения, внутренний опорный источник должен быть полностью выключен установкой REFON=0.
- Когда используется внешний опорный источник, внутренний источник должен быть полностью выключен. Внешние опорные источники могут быть задействованы для V_{R+} и V_{R-} через выводы A4 и A3 соответственно.
- Когда используется внутренний источник и максимальная скорость преобразования ниже 50 ksps, установка ADC10SR=1 уменьшает потребление тока внутренним опорным буфером примерно на 50%.
- Когда оба бита REFOUT=1 и REFBURST=1, опорный источник представлен снаружи только во время периода выборки и преобразования. Когда REFOUT1=1, а REFBURST=0 очищен, опорное напряжение присутствует внешне постоянно.

18.2.4. Тактирование выборки и преобразования

Аналого-цифровое преобразование инициируется по нарастающему фронту входного сигнала выборки SHI. Источник для SHI выбирается с помощью битов SHSx и может быть таким:

- Бит ADC10SC
- Модуль вывода 1 таймера A
- Модуль вывода 0 таймера A
- Модуль вывода 2 таймера A

Полярность источника сигнала SHI может быть инвертирована битом ISSH. Биты SHTx выбирают период выборки tsample равным 4, 8, 16 или 64 цикла ADC10CLK. Таймер выборки устанавливает SAMPCON в высокий уровень для выбранного периода выборки после синхронизации с ADC10CLK. Общее время выборки составляет t_{sample} плюс t_{sync}. Переход SAMPCON с высокого уровня на низкий стартует аналого-цифровое преобразование, которому необходимо 13 циклов ADC10CLK, как показано на рис. 18-3.

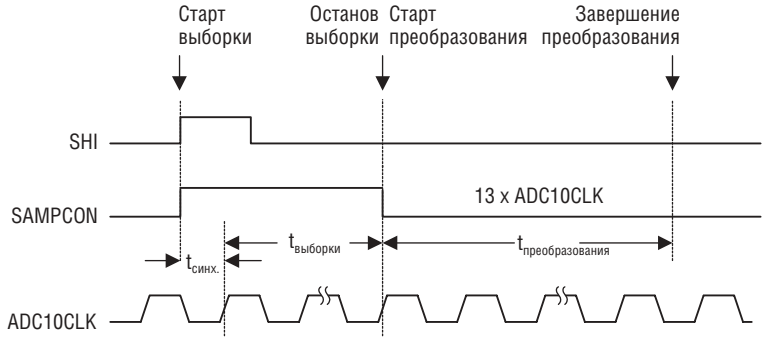


Рис. 18-3. Тактирование выборки

Определение длительности выборки

Когда SAMPCON=0, все входы Ax имеют высокое входное сопротивление. Когда SAMPCON=1, выбранный вход Ax можно смоделировать в виде RC фильтра нижних частот в течение периода выборки t_{sample}, как показано на рис. 18-4. Внутреннее сопротивление R_I (около 2 кОм) мультиплексированного входа последовательно с конденсатором C_I (максимум 40 пФ) представляется источником. Конденсатор C_I должен быть заряжен напряжением V_C в пределах ½ младшего бита источника напряжения V_S для получения точного 10-разрядного преобразования.

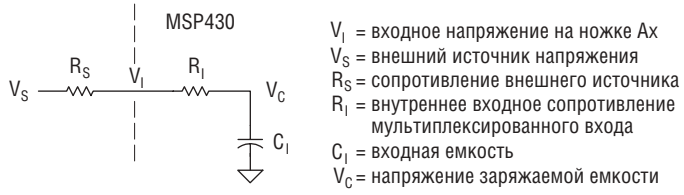


Рис. 18-4. Эквивалентная схема аналогового входа

Сопротивление источника R_S и R_I влияет на t_{sample}. Следующее выражение может быть использовано для вычисления минимального времени выборки t_{sample} при 10-разрядном преобразовании:

Если ADC10SR = 0 $\quad t_{sample} > (R_S + R_I) \times \ln(2^{11}) \times C_I + 800$ нс

Если ADC10SR = 1 $\quad t_{sample} > (R_S + R_I) \times \ln(2^{11}) \times C_I + 2{,}5$ мкс

При подстановке значений R_I и C_I, указанных выше, уравнение приобретает следующий вид:

К примеру, если R_S равно 10 кОм, t_{sample} должно быть больше 2,63 мкс при ADC10SR = 0 или 4,33 мкс при ADC10SR = 1.

$$t_{sample} > (R_S + 2\,кОм) \times 7{,}625 \times 20\,пФ + 800\,нс \quad (ADC10SR = 0)$$

$$t_{sample} > (R_S + 2\,кОм) \times 7{,}625 \times 20\,пФ + 2{,}5\,мкс \quad (ADC10SR = 1)$$

18.2.5. Режимы преобразования

ADC10 имеет четыре режима работы, выбираемые битами CONSEQx так, как описано в таблице 18-1.

Таблица 18-1. Сводный перечень режимов преобразования

CONSEQx	Режим	Операция
00	Одноканальный с одиночным преобразованием	Выполняется одно преобразование в одном канале.
01	Последовательность каналов	Выполняются однократные преобразования последовательности каналов.
10	Повторяющийся одноканальный	Выполняется повторяющееся преобразование в одном канале.
11	Повторяющаяся последовательность каналов	Выполняются повторяющиеся преобразования последовательности каналов.

Одноканальный режим с одиночным преобразованием

В одном канале, выбранном INCHx, однократно выполняется выборка и преобразование. Результат АЦП записывается в регистр ADC10MEM. На рис. 18-5 показан процесс одноканального режима с одиночным преобразованием. Если преобразование запускается ADC10SC, поочередные преобразования могут быть запущены битом ADC10SC. Когда используется другой источник запуска, ENC должен переключаться между каждым преобразованием.

Режим последовательности каналов

В режиме последовательности каналов однократно выполняется выборка и преобразование. Последовательность запускается с выбранного INCHx каналом и декрементируется к каналу A0. Каждый результат АЦП записывается в ADC10MEM. Последовательность останавливается после преобразования в канале A0. На рис. 18-6 показан режим последовательности каналов. Если последовательность запускается ADC10SC, поочередные преобразования могут запускаться битом ADC10SC. Когда используется любой другой источник запуска, ENC должен переключаться между каждой последовательностью.

Повторяющийся одноканальный режим

В одном канале, выбранном INCHx, непрерывно выполняются выборка и преобразование. Каждый результат АЦП записывается в ADC10MEM. На рис. 18- 7 показан повторяющийся одноканальный режим.

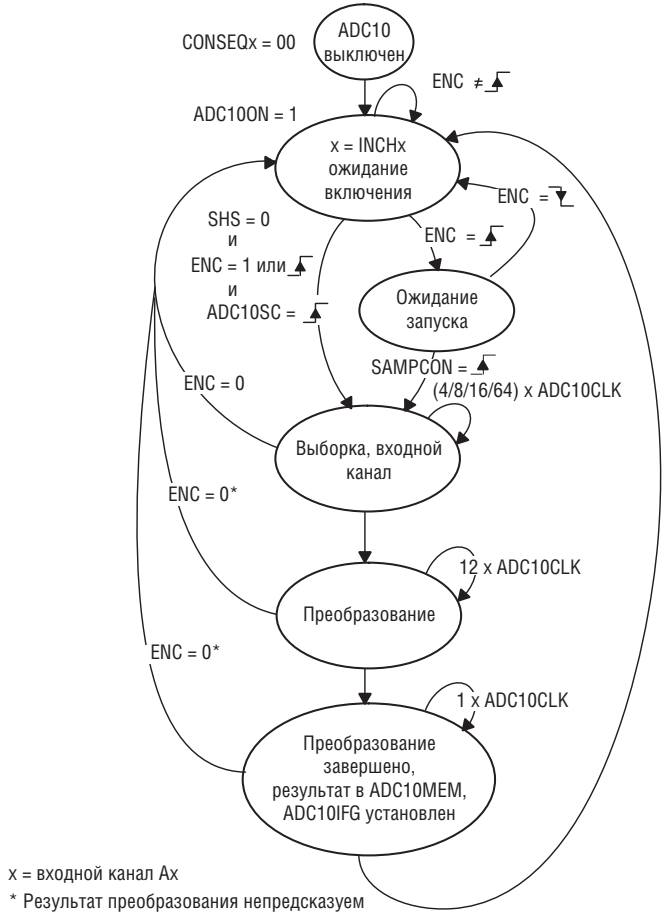


Рис. 18-5. Одноканальный режим одиночного преобразования

Режим повторяющейся последовательности каналов

Непрерывно выполняются выборка и преобразование последовательности каналов. Последовательность начинается с канала, выбранного INCHx и декрементируется к каналу A0. Каждый результат АЦП записывается в ADC10MEM. Последовательность останавливается после преобразования в канале A0, а следующий сигнал запуска стартует последовательность снова. На рис. 18-8 показан режим повторяющейся последовательности каналов.

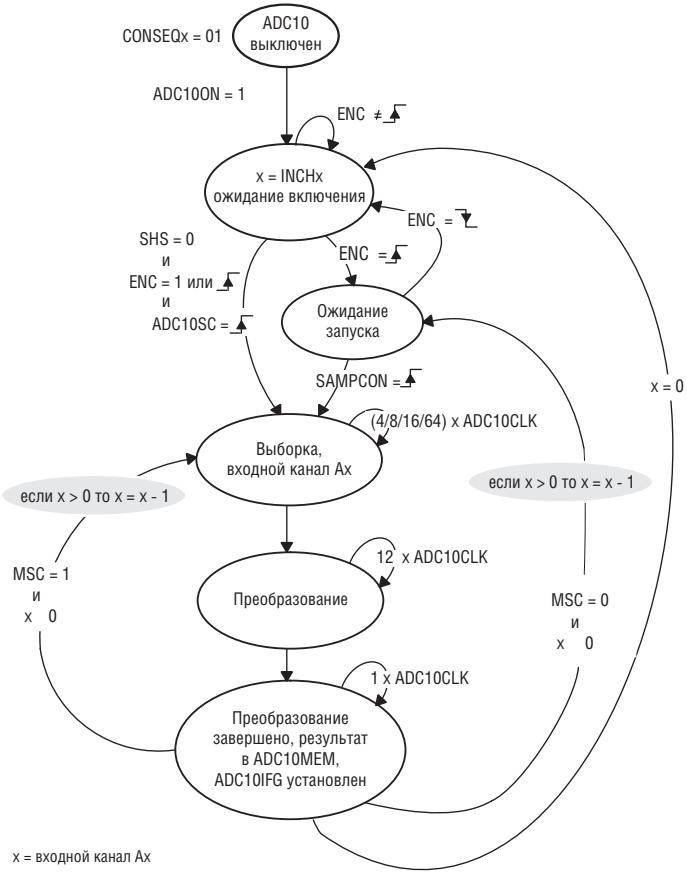


Рис. 18-6. Режим последовательности каналов

Использование бита MSC

Для конфигурирования преобразователя на выполнение автоматических поочередных преобразований с максимальной быстротой можно воспользоваться функцией множественных выборок и преобразований. Если MSC=1 и CONSEQx>1, первый фронт сигнала SHI запустит первое преобразование. Поочередные преобразования запускаются автоматически после завершения предыдущего преобразования. Дополнительные фронты на SHI игнорируются, пока последовательность не закончена в режиме одиночной последовательнос-

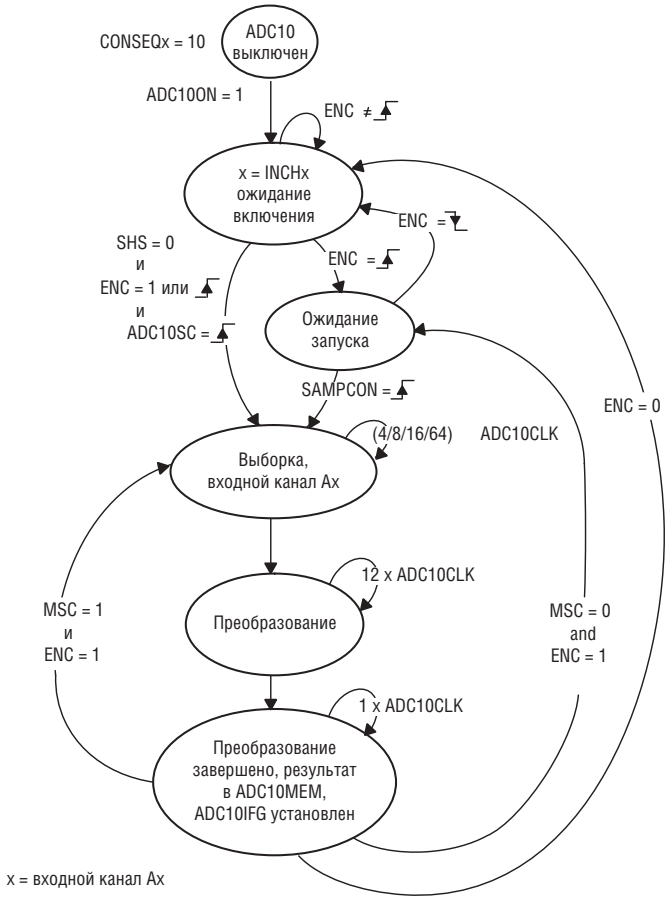


Рис. 18-7. Повторяющийся одноканальный режим

ти или пока бит ENC не будет переключен в повторяющийся одноканальный режим или повторяющийся режим последовательностей. Функция бита ENC не изменяется, пока используется бит MSC.

Останов преобразований

Прекращение активности АЦП10 зависит от режима работы. Рекомендуются следующие способы останова активного преобразования или последовательности преобразований:

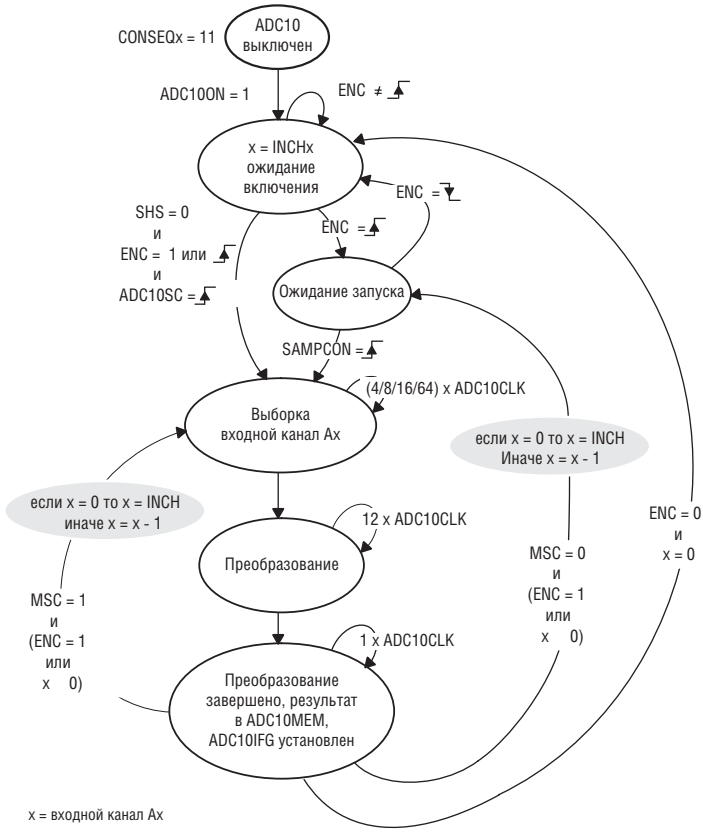


Рис. 18-8. Режим повторяющейся последовательности каналов

- Сброс ENC в одноканальном режиме одиночного преобразования немедленно останавливает преобразование, при этом результат оказывается непредсказуемым. Для получения правильного результата необходимо опрашивать бит занятости ADC10BUSY до сброса перед очисткой ENC.
- Сброс ENC во время повторяющегося одноканального преобразования останавливает преобразователь в конце текущего преобразования.
- Сброс ENC во время последовательного или повторно-последовательного режимов останавливает преобразователь в конце последовательности.

- Любой режим преобразования может быть немедленно остановлен установкой CONSEQx=0 и сбросом бита ENC. Данные преобразования будут ненадежны.

18.2.6. Контроллер переноса данных АЦП10

АЦП10 имеет контроллер переноса данных (DTC) для автоматического переноса результатов преобразования из ADC10MEM в другое место памяти на кристалле. DTC включается установкой регистра ADC10DTC1 в ненулевое значение.

Когда DTC включен, каждый раз по завершении преобразования АЦП10 и загрузки результата в ADC10MEM, запускается перенос данных. Вмешательства программного обеспечения для управления АЦП10 не требуется до тех пор, пока заданное количество данных преобразования не будет перемещено. Для каждого DTC-переноса требуется один такт MCLK ЦПУ. Во избежание любой конфликтной ситуации на шине во время DTC-переноса ЦПУ приостанавливается, если было активно, на один MCLK, необходимый для переноса.

DTC-перенос не должен инициироваться, пока АЦП10 занят. Программное обеспечение должно гарантировать, что никакое активное преобразование или последовательность не выполняется, когда DTC конфигурируется:

```
;Проверка активности АЦП10
          BIC.W #ENC,&ADC10CTL0      ;
busy_test BIT.W #BUSY,&ADC10CTL1     ;
          JNZ busy_test              ;
          MOV.W #xxx,&ADC10SA        ;Безопасность
          MOV.B #xx,&ADC10DTC1       ;
;продолжение настройки
```

Режим одноблочного переноса

Одноблочный режим выбирается, если ADC10TB сброшен. Значение n в ADC10DTC1 определяет общее количество переносов для блока. Стартовый адрес блока задается где-либо в адресном диапазоне MSP430 с помощью 16-разрядного регистра ADC10SA. Блок заканчивается на адресе ADC10SA+2n-2. Режим одноблочного переноса показан на рис. 18-9.

Внутренний указатель адреса первоначально равен ADC10SA, а внутренний счетчик перенос первоначально равен «n». Внутренний указатель и счетчик не видимы для программного обеспечения. DTC переносит значение слова ADC10MEM по адресу, находящемуся в адресном указателе ADC10SA. После каждого DTC-переноса внутренний адресный указатель инкрементируется на два, а внутренний счетчик переносов декрементируется на один.

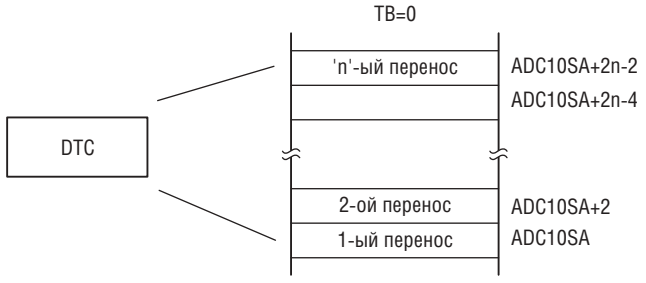


Рис. 18-9. Одноблочный перенос

DTC переносы продолжаются с каждой загрузкой ADC10MEM, пока внутренний счетчик переносов не станет равным нулю. Дополнительные DTC переносы происходить не будут до записи в ADC10SA. Когда DTC используется в одноблочном режиме, флаг ADC10IFG устанавливается только после завершения переноса полного блока. На рис. 18-10 показана диаграмма состояний одноблочного режима.

Режим двухблочного переноса

Двухблочный режим выбирается, если бит ADC10TB установлен. Значение n в ADC10DTC1 определяет количество переносов для одного блока. Адресный диапазон первого блока задается в любом месте диапазона адресов MSP430 с помощью 16-разрядного регистра ADC10SA. Первый блок заканчивается на адресе ADC10SA+2n-2. Адресный диапазон для второго блока задается с SA+2n по SA+4n-2. Режим двухблочного переноса показан на рис. 18-11.

Внутренний указатель адреса первоначально равен ADC10SA, а внутренний счетчик переносов первоначально равен «n». Внутренний указатель и счетчик являются невидимыми для программного обеспечения. DTC переносит значение слова ADC10MEM по адресу, находящемуся в адресном указателе ADC10SA. После каждого DTC-переноса внутренний адресный указатель инкрементируется на два, а внутренний счетчик переносов декрементируется на один.

DTC переносы продолжаются с каждой загрузкой ADC10MEM, пока внутренний счетчик переносов не станет равным нулю. К этому моменту блок один полон и оба бита ADC10IFG и ADC10B1 установлены. Пользователь может проверить бит ADC10B1 для определения, что блок один полон.

DTC продолжает с блока два. Во внутренний счетчик переносов автоматически перезагружается значение «n». При следующей загрузке ADC10MEM контроллер DTC начинает перенос результатов преобразований в блок два. После завершения n переносов блок два полон. Флаг ADC10IFG устанавливается, а бит

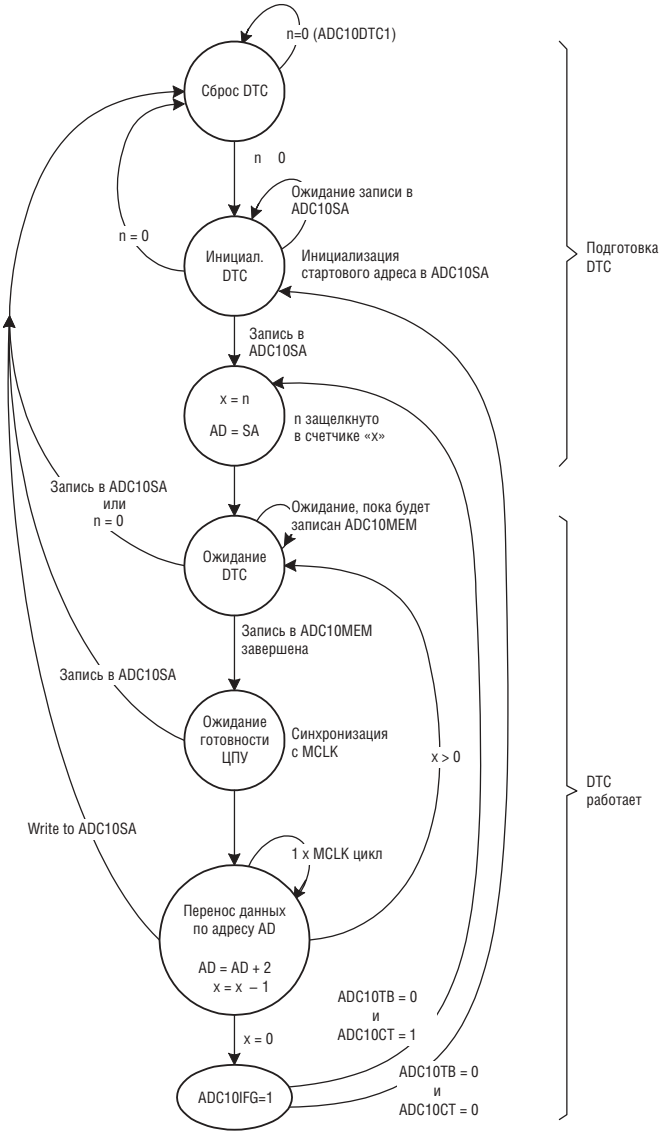


Рис. 18-10. Диаграмма состояний управления переносом данных в одноблочном режиме переноса

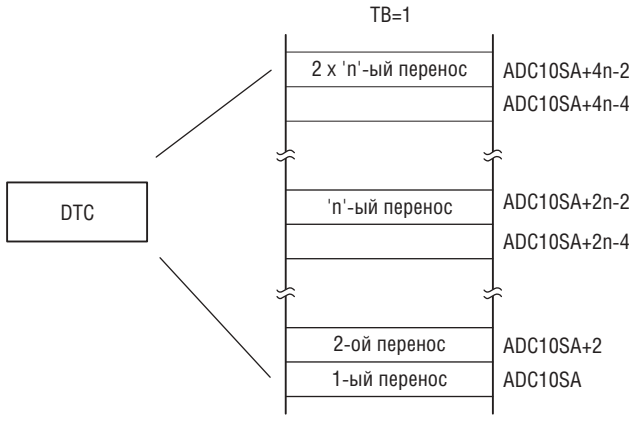


Рис. 18-11. Двухблочный перенос

ADC10B1 очищается. Программное обеспечение пользователя может проверить очистку бита ADC10B1 для определения, что блок два полон. На рис. 18- 12 показана диаграмма состояний двухблочного режима.

Непрерывный перенос

Режим непрерывного переноса выбирается, если установлен бит ADC10CT. DTC не будет останавливаться после завершения переноса блока один (в одноблочном режиме) или блока два (в двухблочном режиме). Внутренний адресный указатель и счетчик переносов устанавливаются на значения ADC10SA и n соответственно. Переносы продолжаются, начиная с блока один. Если бит ADC10CT сброшен, DTC прекращает переносы после завершения текущих переносов в блоке один (в одноблочном режиме) или блоке два (в двухблочном режиме).

Длительность цикла DTC переноса

Для каждого переноса ADC10MEM контроллеру DTC требуется один или два тактовых цикла MCLK для синхронизации, один для выполнения собственно переноса (пока ЦПУ приостановлено) и один цикл времени ожидания. Поскольку DTC использует MCLK, длительность цикла DTC определяется рабочим режимом MSP430 и настройками системы тактирования.

Если источник MCLK активен, но ЦПУ выключено, DTC использует источник MCLK для каждого переноса без включения ЦПУ. Если источник MCLK выключен, DTC временно рестартует MCLK, с использованием в качестве источника тактирования для MCLK частоту DCOCLK, но только во время переноса.

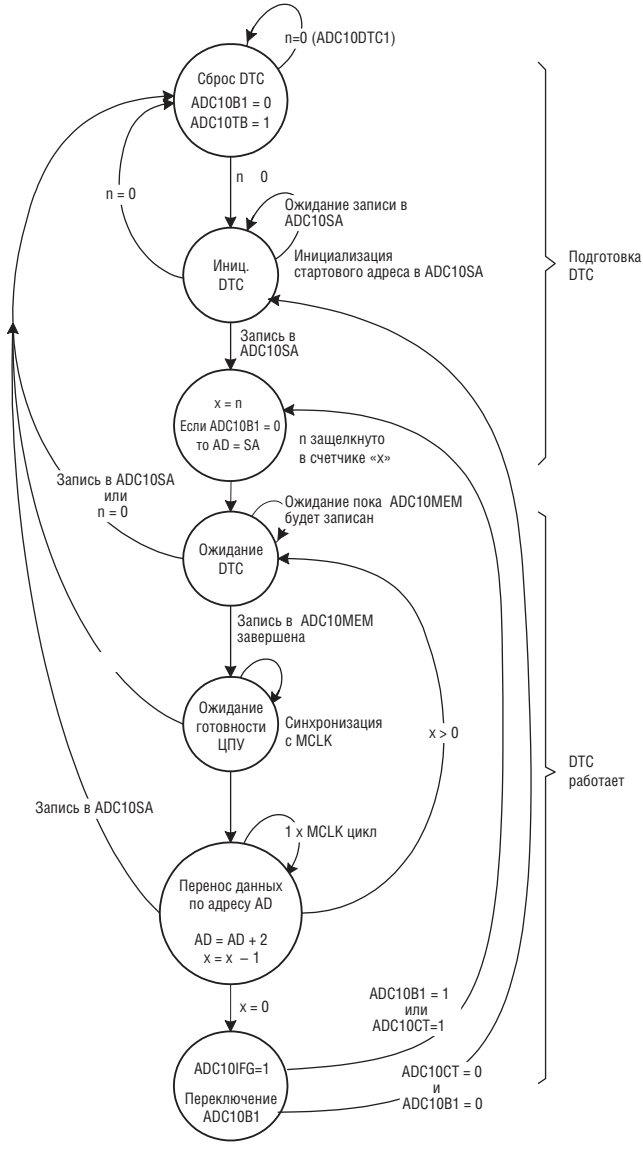


Рис. 18-12. Диаграмма состояний для управления переносом данных в двухблочном режиме переноса

ЦПУ остается выключенным и после DTC переноса, MCLK снова выключается. Максимальная длительность цикла DTC для всех рабочих режимах показана в таблице 18-2.

Таблица 18-2. Максимальная длительность цикла DTC

Режим работы ЦПУ	Источник тактирования	Максимальная длительность цикла DTC
Активный режим	MCLK=DCOCLK	3 цикла MCLK
Активный режим	MCLK=LFXT1CLK	3 цикла MCLK
Режим пониженного энергопотребления LPM0/1	MCLK=DCOCLK	4 цикла MCLK
Режим пониженного энергопотребления LPM3/4	MCLK=DCOCLK	4 цикла MCLK + 6 мкС*
Режим пониженного энергопотребления LPM0/1	MCLK=LFXT1CLK	4 цикла MCLK
Режим пониженного энергопотребления LPM3	MCLK=LFXT1CLK	4 цикла MCLK
Режим пониженного энергопотребления LPM4	MCLK=LFXT1CLK	4 цикла MCLK + 6 мкС*

* Дополнительные 6 мкС необходимы для запуска DCOCLK. Этот параметр в справочном руководстве называется t(LPMx).

18.2.7. Использование интегрированного температурного датчика

При использовании имеющегося на кристалле температурного датчика пользователь выбирает входной аналоговый канал INCHx=1010. Любая другая

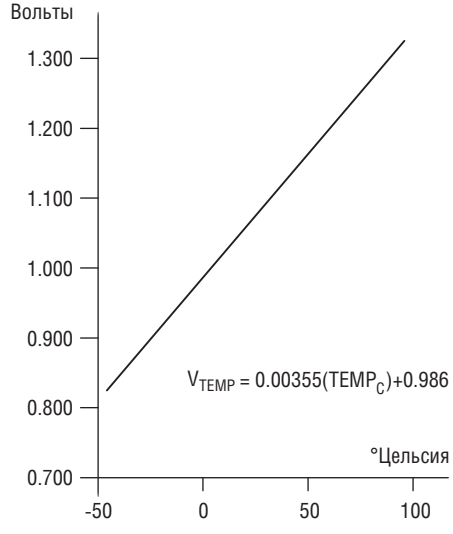


Рис. 18-13. Типичная передаточная функция температурного датчика

конфигурация рассматривается как выбор внешнего канала, включая выбор опорного источника, выбор памяти преобразований и т.д.

Типичная передаточная функция температурного датчика показана на рис. 18-13. При использовании температурного датчика, период выборки должен быть больше 30 мкС. Ошибка смещения температурного датчика может быть большой и для большинства приложений может потребоваться калибровка. См. справочные данные конкретного устройства для выяснения подробностей.

При выборе температурного датчика автоматически запускается расположенный на кристалле опорный генератор в качестве источника напряжения для температурного датчика. Однако это не включает выход V_{REF+} и не влияет на выбор опорного источника для преобразования. Процедура выбора источника для преобразования информации с температурного датчика подобна процедуре выбора любого другого канала.

18.2.8. Заземление АЦП и рассмотрение влияния помех

Как в любом АЦП с высоким разрешением, для устранения нежелательных паразитных эффектов и шумов, а также предотвращения возникновения паразитных контуров с замыканием на землю, необходима особая разводка печатной платы и методы заземления.

Паразитные общие петли формируются, когда ток возврата от АЦП проходит совместно с токами других аналоговых и цифровых схем. Если не принимать специальных мер, этот ток может генерировать небольшие нежелательные напряжения смещения, которые могут прибавляться или вычитаться из опорного или входного напряжений аналого-цифрового преобразователя. Способ подключения, показанный на рис. 18-14 позволяет этого избежать.

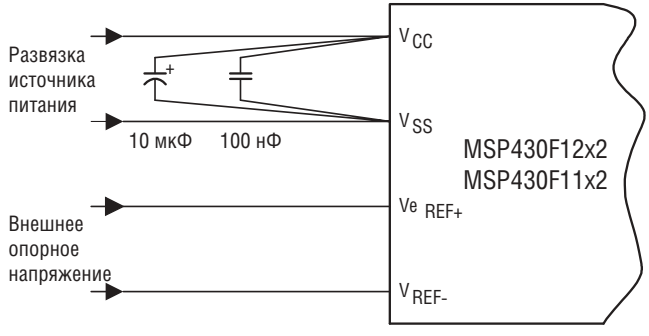


Рис. 18-14. Заземление АЦП10 и устранение помех

В дополнение к заземлению, пульсации и шумовые выбросы на линиях источника питания, вызванные переключениями цифровых схем или переключениями в источнике питания могут повредить результат преобразования. Для получения высокой точности рекомендуется создавать разработки, свободные от шумов.

18.2.9. Прерывания АЦП10

Одно прерывание и один вектор прерываний связаны с АЦП10, как показано на рис. 18-15. Когда DTC не используется (ADC10DTC1=0), флаг ADC10IFG устанавливается, когда результаты преобразования загружаются в ADC10MEM. Когда DTC используется (ADC10DTC1>0), флаг ADC10IFG устанавливается, ког-

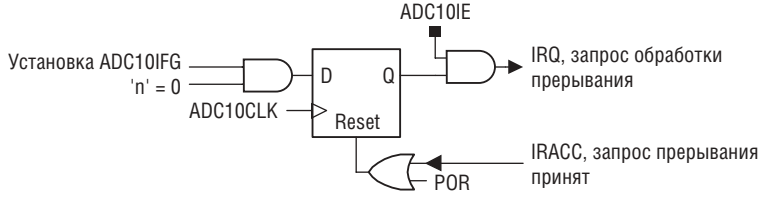


Рис. 18-15. Система прерываний АЦП10

да завершается перенос блока и внутренний счетчик переносов «n»=0. Если оба бита ADC10IE и GIE установлены, флаг ADC10IFG генерирует запрос прерывания. Флаг ADC10IFG автоматически сбрасывается, когда запрос прерывания обработан, кроме того, он может быть сброшен программно.

18.3. Регистры АЦП10

Регистры АЦП10 приведены в таблице 18-3.

Таблица 18-3. Регистры АЦП10

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр включения входа АЦП10	ADC10AE	Чтение/запись	04Ah	Сброс с POR
Управляющий регистр 0 АЦП10	ADC10CTL0	Чтение/запись	01B0h	Сброс с POR
Управляющий регистр 1 АЦП10	ADC10CTL1	Чтение/запись	01B2h	Сброс с POR
Память АЦП10	ADC110MEM	Чтение	01B4h	Не изменяется
Регистр 0 управления переносом данных АЦП10	ADC10DTC0	Чтение/запись	048h	Сброс с POR
Регистр 1 управления переносом данных АЦП10	ADC10DTC1	Чтение/запись	049h	Сброс с POR
Стартовый адрес переноса данных АЦП10	ADC10SA	Чтение/запись	01BCh	0200h с POR

ADC10CTL0, управляющий регистр 0 АЦП10

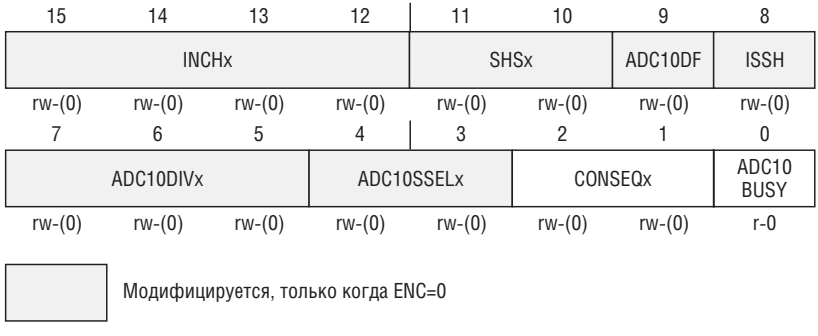
15	14	13	12	11	10	9	8
SREFx			ADC10SHTx		ADC10SR	REFOUT	REFBURST
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
MSC	REF2_5V	REFON	ADC10ON	ADC10IE	ADC10IFG	ENC	ADC10SC
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Модифицируется, только когда ENC=0

SREFx	**Биты 15-13**	Выбор опорного источника 000 – $V_{R+} = AV_{CC}$ и $V_{R-} = AV_{SS}$ 001 – $V_{R+} = V_{REF+}$ и $V_{R-} = AV_{SS}$ 010 – $V_{R+} = Ve_{REF+}$ и $V_{R-} = AV_{SS}$ 011 – $V_{R+} = Ve_{REF+}$ и $V_{R-} = AV_{SS}$ 100 – $V_{R+} = AV_{CC}$ и $V_{R-} = V_{REF-}/Ve_{REF-}$ 101 – $V_{R+} = V_{REF+}$ и $V_{R-} = V_{REF-}/Ve_{REF-}$ 110 – $V_{R+} = Ve_{REF+}$ и $V_{R-} = V_{REF-}/Ve_{REF-}$ 111 – $V_{R+} = Ve_{REF+}$ и $V_{R-} = V_{REF-}/Ve_{REF-}$
ADC10SHTx	**Биты 12-11**	Время выборки-хранения АЦП10 00 – 4 x ADC10CLKs 01 – 8 x ADC10CLKs 10 – 16 x ADC10CLKs 11 – 64 x ADC10CLKs
ADC10SR	**Бит 10**	Скорость выборки АЦП10. С помощью этого бита выбирается ёмкость буфера для максимальной скорости преобразования. Установка ADC10SR снижает потребление буфера. 0 – Выбор размера буфера для скорости преобразования до 200 ksps 1 – Выбор размера буфера для скорости преобразования до 50 ksps
REFOUT	**Бит 9**	Выход опорного источника 0 – Выход опорного источника выключен 1 – Выход опорного источника включен
REFBURST	**Бит 8**	Кратковременное включение опорного источника. REFOUT также должен быть установлен. 0 – Опорное напряжение подается постоянно 1 – Опорное напряжение подается только во время выборки-преобразования
MSC	**Бит 7**	Множественная выборка и преобразование. Справедливо только для последовательного и повторяющегося режимов. 0 – Процедуре выборки необходим фронт сигнала SHI для запуска каждой выборки-преобразования 1 – Первый фронт сигнала SHI запускает таймер выборки, а последующие выборки-преобразования выполняются автоматически по завершении предыдущего преобразования

REF2_5V	**Бит 6**	Генератор опорного напряжения. REFON также должен быть установлен. 0 – 1.5 В 1 – 2.5 В
REFON	**Бит 5**	Включение опорного генератора 0 – Опорный генератор выключен 1 – Опорный генератор включен
ADC10ON	**Бит 4**	Включение АЦП10 0 – АЦП10 выключен 1 – АЦП10 включен
ADC10IE	**Бит 3**	Разрешение прерывания от АЦП10 0 – Прерывание запрещено 1 – Прерывание разрешено
ADC10IFG	**Бит 2**	Флаг прерывания АЦП10. Этот бит устанавливается, если в ADC10MEM загружа-ется результат преобразования. Он автоматически сбрасывается после приема за-проса прерывания или может быть сброшен программно. Когда используется DTC, этот флаг устанавливается, когда завершен перенос блока. 0 – Прерывание не ожидается 1 – Ожидается прерывание
ENC	**Бит 1**	Включение преобразования 0 – АЦП10 отключен 1 – АЦП10 включен
ADC10SC	**Бит 0**	Старт преобразования. Программно-управляемый старт выборки-преобразования. ADC10SC и ENC могут быть установлены вместе в одной команде. ADC10SC сбрасывается автоматически. 0 – Нет старта выборки-преобразования 1 – Старт выборки-преобразования

ADC10CTL1, управляющий регистр 1 АЦП10



INCHx	**Биты 15-12**	Выбор входного канала 0000 – A0 0001 – A1 0010 – A2 0011 – A3 0100 – A4 0101 – A5 0110 – A6 0111 – A7 1000 – Ve_{REF+} 1001 – V_{REF-}/Ve_{REF-} 1010 – Температурный диод 1011 – $(AV_{CC} - AV_{SS})/2$ 1100 – $(AV_{CC} - AV_{SS})/2$ 1101 – $(AV_{CC} - AV_{SS})/2$ 1110 – $(AV_{CC} - AV_{SS})/2$ 1111 – $(AV_{CC} - AV_{SS})/2$
SHSx	**Биты 11-10**	Выбор источника выборки-хранения 00 – Бит ADC10SC 01 – Выход 1 Таймера A 10 – Выход 0 Таймера A 11 – Выход 2 Таймера A
ADC10DF	**Бит 9**	Формат данных АЦП10 0 – Натуральный двоичный 1 – С дополнением до двух
ISSHx	**Бит 8**	Инвертирование сигнала выборки-хранения 0 – Сигнал выборки-хранения не инвертирован 1 – Сигнал выборки-хранения инвертирован
ADC10DIVx	**Биты 7-5**	Тактовый делитель АЦП10 000 – /1 001 – /2 010 – /3 011 – /4 100 – /5 101 – /6 110 – /7 111 – /8
ADC10SSELx	**Биты 4-3**	Выбор источника тактирования АЦП10 00 – ADC10OSC 01 – ACLK 10 – MCLK 11 – SMCLK

CONSEQx	**Биты 2-1**	Выбор режима преобразования 00 – Одноканальный, с одним преобразованием 01 – Последовательность каналов 10 – Повторный одноканальный 11 – Повторяющаяся последовательность каналов
ADC10BUSY	**Бит 0**	Занятость АЦП12. Этот бит показывает активность операций выборки и преобразования. 0 – Действия не выполняются 1 – Выполняется последовательность, выборка или преобразование

ADC10AE, управляющий регистр включения аналогового входа

7	6	5	4	3	2	1	0
ADC10AE7	ADC10AE6	ADC10AE5	ADC10AE4	ADC10AE3	ADC10AE2	ADC10AE1	ADC10AE0
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

ADC10AEx	**Биты 7-0**	Включение аналогового входа 0 – Аналоговый вход выключен 1 – Аналоговый вход включен

ADC10MEM, регистр памяти преобразования, двоичный формат

15	14	13	12	11	10	9	8
0	0	0	0	0	0	Результаты преобразования	
r0	r0	r0	r0	r0	r0	r	r

7	6	5	4	3	2	1	0
Результаты преобразования							
r	r	r	r	r	r	r	r

Результаты преобразования	**Биты 15-0**	10-разрядные результаты преобразования выравниваются по правому краю в натуральном двоичном формате. Бит 9 является старшим битом (MSB). Биты 15-10 всегда равны 0.

ADC10MEM, регистр памяти преобразования, формат с дополнением до двух

15	14	13	12	11	10	9	8
Результаты преобразования							
r	r	r	r	r	r	r	r

7	6	5	4	3	2	1	0
Результаты преобразования		0	0	0	0	0	0
r	r	r0	r0	r0	r0	r0	r0

Результаты преобразования	**Биты 15-0**	10-разрядные результаты преобразования выравниваются по левому краю в формате дополнения до двух. Бит 15 является старшим значащим разрядом (MSB). Биты 5-0 всегда равны 0.

ADC10DTC0, регистр 0 управления переносом данных

7	6	5	4	3	2	1	0
Не используется				ADC10TB	ADC10CT	ADC10B1	ADC10 FETCH
r0	r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)

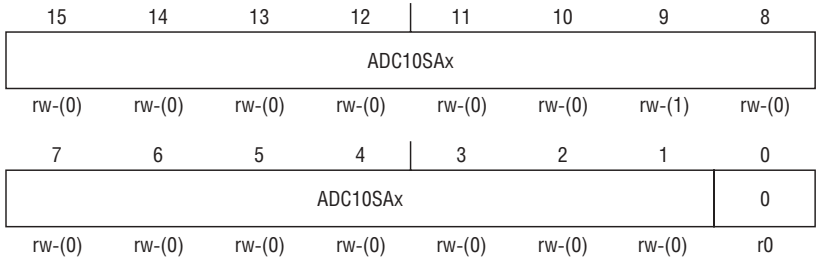
Зарезервировано	**Биты 7-4**	Зарезервированы. Всегда читаются как 0.
ADC10TB	**Бит 3**	Двухблочный режим АЦП10. 0 – Одноблочный режим переноса 1 – Двухблочный режим переноса
ADC10CT	**Бит 2**	Непрерывный перенос АЦП10. 0 – Перенос данных останавливается, когда перенос одного блока (одноблочный режим) или двух блоков (двухблочный режим) завершен. 1 – Перенос данных выполняется непрерывно. Функционирование DTC останавливается только если ADC10CT очищен или произведена запись в ADC10SA.
ADC10B1	**Бит 1**	Блок один АЦП10. Этот бит указывает в двухблочном режиме, какой блок заполнен результатами преобразований АЦП10. Значение ADC10B1 справедливо только после установки ADC10IFG в первый раз во время функционирования DTC. ADC10TB также должен быть установлен. 0 – Заполнен блок 1 1 – Заполнен блок 2
ADC10FETCH	**Бит 0**	Этот бит обычно должен быть сброшен.

ADC10DTC1, регистр 1 управления переносом данных

7	6	5	4	3	2	1	0
Переносы DTC							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Переносы DTC	**Биты 7-4**	Переносы DTC. Эти биты задают количество переносов в каждом блоке. 0 – DTC отключен 01h-0FFh – Количество переносов в блоке

ADC10SA, регистр стартового адреса для переноса данных



ADC10SAx	**Биты 15-1**	Стартовый адрес АЦП10. Эти биты являются стартовым адресом для DTC. Запись в регистр ADC10SA необходима для инициирования DTC переносов.
Не используется	**Бит 0**	Не используется, только чтение. Всегда читается как 0.

MSP430x1xxFamily

ЦАП12

Раздел XIX.

Texas Instruments

ЦАП12

Модуль ЦАП12 представляет собой 12-разрядный цифро-аналоговый преобразователь. В этом разделе описывается ЦАП12. В устройствах MSP430x15x и MSP430x16x реализовано два модуля ЦАП12.

19.1. Введение в ЦАП12

Модуль АЦП12 представляет собой 12-разрядный ЦАП. ЦАП12 может быть сконфигурирован в 8-ми или 12-разрядном режиме и может использоваться совместно с контроллером DMA. Когда в устройстве представлено несколько модулей ЦАП12, они могут быть сгруппированы вместе для синхронного обновления.

ЦАП12 обладает следующими возможностями:

- 12-разрядный монотонный выход
- 8-ми или 12-разрядное разрешение выходного напряжения
- Программируемое время установки в зависимости от потребляемой мощности
- Выбор внутреннего или внешнего опорного источника
- Натуральный двоичный формат данных или формат с дополнением до двух
- Опция самокалибровки для корректировки смещения
- Возможность синхронного обновления при наличии нескольких модулей ЦАП12

Примечание: Множество модулей ЦАП12

Некоторые устройства могут содержать более одного модуля ЦАП12. В случае, когда в устройстве представлено более одного ЦАП12, все модули ЦАП12 работают идентично.

Везде в этом разделе терминология типа DAC12_xDAT или DAC12_xCTL используется для описания имен регистров. При этом x используется для указания, о каком модуле ЦАП12 идет речь. В случае, если операция одинакова для всех модулей, регистр упоминается просто как DAC12_xCTL.

Блок-схема двух модулей ЦАП12 в устройствах MSP430F15x/16x показана на рис. 19-1.

19.2. Функционирование ЦАП12

Модуль ЦАП12 конфигурируется программным обеспечением пользователя. Настройка и функционирование ЦАП12 обсуждаются в нижеследующих разделах.

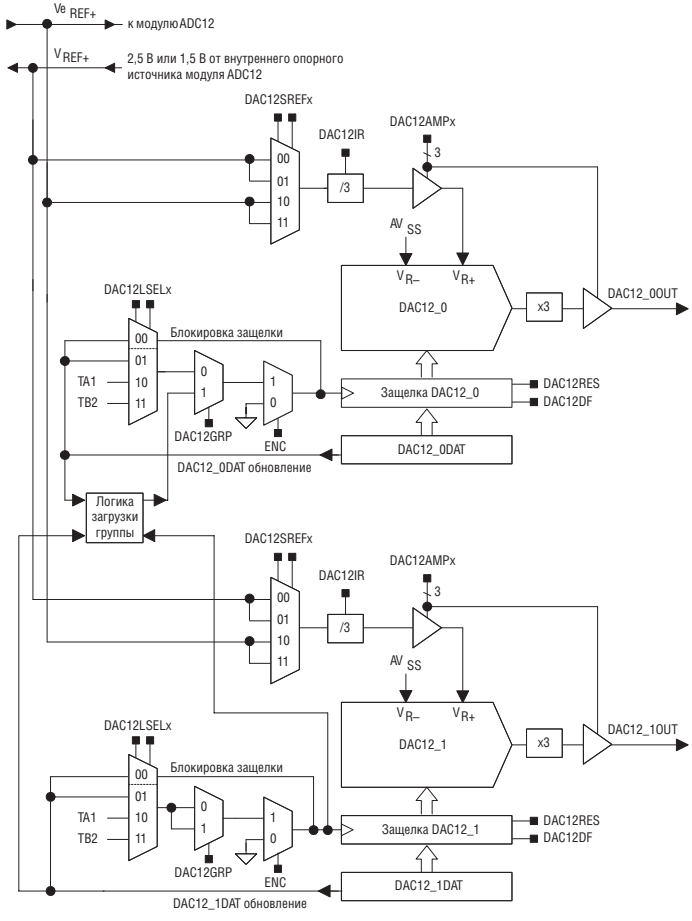


Рис. 19-1. Блок-схема ЦАП12

19.2.1. Ядро ЦАП12

ЦАП12 может быть сконфигурирован на работу в 8-ми или 12-разрядном режиме с помощью бита DASC12RES. Кроме того, полный диапазон вывода программируется через бит DAC12IR и может быть 1x или 3x-кратен выбранному опорному напряжению. Эта возможность позволяет пользователю управлять динамическим диапазоном ЦАП12. Когда используется внутренний опорный источник, полный диапазон вывода всегда равен 1x опорного напряжения. Бит DAC12DF позволяет пользователю выбирать для ЦАП натуральные двоичные

данные или данные с дополнением до двух. Когда используется натуральный двоичный формат данных, справедлива формула для выходного напряжения, представленная в таблице 19-1.

Таблица 19-1. Полный диапазон ЦАП12 (V_{ref} = Ve_{REF+} или V_{REF+})

Разрешение	DAC12RES	DAC12IR	Формула выходного напряжения
12 бит	0	0	$V_{out} = V_{ref} \times 3 \times \frac{DAC12\_xDAT}{4096}$
12 бит	0	1	$V_{out} = V_{ref} \times \frac{DAC12\_xDAT}{4096}$
8 бит	1	0	$V_{out} = V_{ref} \times 3 \times \frac{DAC12\_xDAT}{256}$
8 бит	1	1	$V_{out} = V_{ref} \times \frac{DAC12\_xDAT}{256}$

В 8-разрядном режиме максимальное используемое значение для DAC12_xDAT равно 0FFh, а в 12-разрядном режиме максимальное используемое значение для DAC12_xDAT равно 0FFFh. Значения, превышающие указанные величины могут быть записаны в регистр, но все первые биты будут проигнорированы.

Выбор порта ЦАП12

Выходы ЦАП12 мультиплексированы с ножками порта P6 и аналоговыми входами АЦП12. Когда DAC12AMPx>0, для ножек автоматически выбирается функция ЦАП12, независимо от состояния связанных с ними битов P6SELx и P6DIRx.

19.2.2. Опорный источник ЦАП12

Опорный источник для ЦАП12 конфигурируется для использования либо двух внешних опорных напряжений, либо внутреннего опорного источника 1.5 В/2.5 В от модуля АЦП12 с помощью битов DAC12SREFx. Когда DAC12SREFx={0,1}, как опорный используется сигнал V_{REF+}, а когда DAC12SREFx={2,3}, в качестве опорного используется сигнал Ve_{REF+}.

При использовании внутреннего опорного источника АЦП12, он должен быть включен и сконфигурирован через соответствующие управляющие биты АЦП12 (см. раздел «АЦП12»). Как только опорный источник АЦП12 сконфигурирован, опорное напряжение подается на V_{REF+}.

Буферы входного опорного сигнала и выходного напряжения ЦАП12

Буферы входного опорного сигнала и выходного напряжения ЦАП12 могут быть сконфигурированы для оптимизации времени установки* в зависимости

* Под временем установки понимается время, необходимое ЦАП для установки выходного напряжения, соответствующего его цифровому представлению на входе и при изменении входного значения.

от потребляемой мощности. Восемь возможных комбинаций выбираются с помощью битов DAC12AMPx. При низкой/низкой установке время установки наибольшее, а потребляемый обеими буферами ток наименьший. Средние и высокие настройки позволяют получить быстрое время установки, однако потребляемый ток возрастет. См. справочное руководство конкретного устройства для выяснения подробных параметров.

19.2.3. Обновление выходного напряжения ЦАП12

Регистр DAC12_xDAT может быть напрямую подключен к ядру ЦАП12 или дважды буферизирован. Источник запуска для обновления выходного напряжения ЦАП12 выбирается с помощью битов DAC12LSELx.

Когда DAC12LSELx=0, защелка данных открыта, и регистр DAC12_xDAT напрямую подключен к ядру ЦАП12. Выход ЦАП12 обновляется немедленно, как только новые данные ЦАП12 записаны в регистр DAC12_xDAT, независимо от состояния бита DAC12ENC.

Когда DAC12LSELx=1, данные ЦАП12 защелкнуты и поступают к ядру ЦАП12 после записи новых данных в DAC12_xDAT. Когда DAC12LSELx=2 или 3, данные защелкиваются по фронту сигнала с выхода таймера A CCR1 или с выхода таймера B CCR2 соответственно. DAC12ENC должен быть установлен, чтобы новые данные защелкивались, когда DAC12LSELx>0.

19.2.4. Формат данных DAC12_xDAT

ЦАП12 поддерживает два формата данных: натуральный двоичный и формат с дополнением до двух. Когда используется натуральный формат данных, полный диапазон вывода равен 0FFFh в 12-разрядном режиме (0FFh в 8-разрядном режиме), как показано на рис. 19-2.

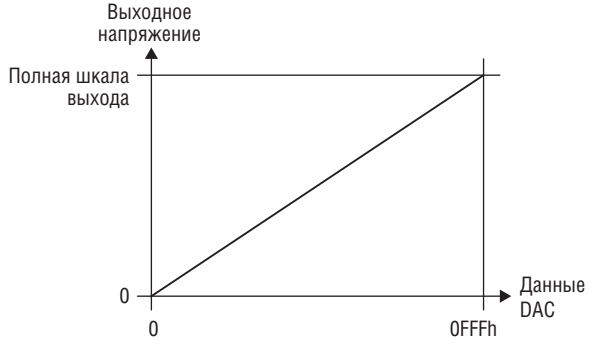


Рис. 19-2. Зависимость выходного напряжения от данных ЦАП12 в 12-разрядном режиме в натуральном двоичном формате

Когда используется формат данных с дополнением до двух, диапазон сдвигается так, что при значении DAC12_xDAT равному 0800h (0080h в 8-разрядном режиме), выходное напряжение будет равно нулю, при 0000h – выходное напряжение составит половину шкалы, а при 07FFh (007Fh для 8-разрядного режима) выходное напряжение достигнет полного диапазона, как показано на рис. 19-3.

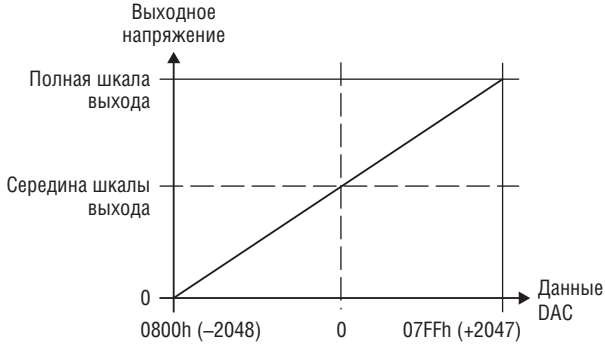


Рис. 19-3. Зависимость выходного напряжения от данных ЦАП12 в 12-разрядном режиме в формате дополнения до двух

19.2.5. Калибровка смещения выходного усилителя ЦАП12

Напряжение смещения выходного усилителя ЦАП12 может быть положительным или отрицательным. Когда смещение отрицательное, выходной усилитель пытается управлять отрицательным напряжением, но не может этого сде-

Рис. 19-4. Отрицательное смещение

лать. Выходное напряжение остается равным нулю, пока цифровой вход ЦАП12 не обеспечит достаточного для преодоления отрицательного напряжения смещения положительного выходного напряжения. Получающаяся передаточная функция показана на рис. 19-4.

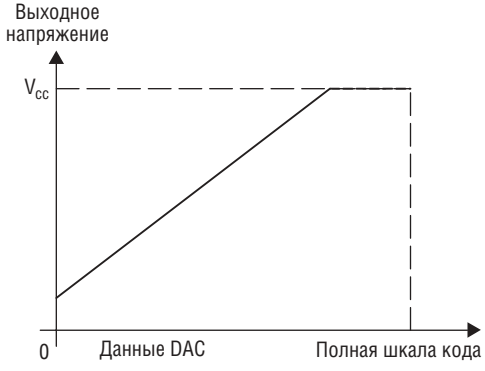


Рис. 19-5. Положительное смещение

Когда выходной усилитель имеет положительное смещение, ноль на цифровом входе не позволяет получить нулевое выходное напряжение. Выходное напряжение ЦАП12 достигает максимального выходного сигнала до того момента, когда данные на входе ЦАП12 достигнут максимального кода. Это показано на рис. 19-5.

ЦАП12 имеет возможность калибровки напряжения смещения выходного усилителя. Установка бита DAC12CALON инициирует калибровку смещения. Калибровка должна быть завершена до использования ЦАП12. Когда калибровка выполнена, бит DAC12CALON автоматически сбрасывается. Биты DAC12AMPx должны быть сконфигурированы до калибровки. Для достижения лучших результатов калибровки, необходимо минимизировать активность ядра и порта в процессе калибровки.

19.2.6. Группировка нескольких модулей ЦАП12

Несколько ЦАП12 могут быть сгруппированы вместе битом DAC12GRP для синхронного обновления каждого выхода ЦАП12. Аппаратно гарантируется, что все модули ЦАП12 в группе обновляются одновременно, независимо от любого прерывания или NMI-события.

В устройствах MSP430x15x и MSP430x16x модули DAC12_0 и DAC12_1 группируются установкой бита DAC12GRP модуля DAC12_0. Бит DAC12GRP модуля DAC12_1 не используется. Когда DAC12_0 и DAC12_1 группируются, необходимо:

- Биты DAC12LSELx модуля DAC12_1 выбирают источник запуска обновления для обоих ЦАПов;
- Биты DAC12LSELx для обоих ЦАП должны быть > 0
- Биты DAC12ENC обоих ЦАП должны быть установлены в 1.

Когда DAS12_0 и DAC12_1 сгруппированы, оба регистра DAC12_xDAT должны быть записаны перед обновлением выходов – даже если данные для одного или обоих ЦАП не изменились. На рис. 19-6 показан пример тактирования защелкивания-обновления для сгруппированных модулей DAC12_0 и DAC12_1.

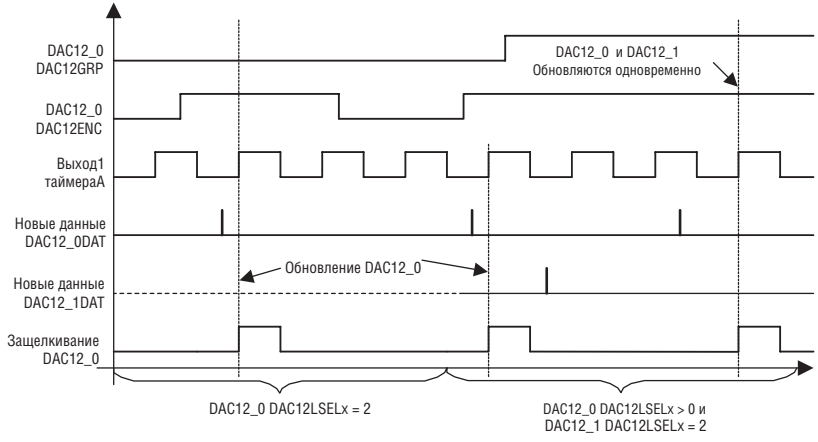


Рис. 19-6. Пример обновления группы ЦАП12, запуск от таймера_A3

Когда бит DAC12GRP=1 модуля DAC12_0 и оба бита DAC12LSELx>0 модулей DAC12_x и любой DAC12ENC=0, никакой ЦАП12 не обновляется.

Примечание: Время установки ЦАП12

Контроллер DMA позволяет переносить данные в ЦАП12 быстрее времени установки их на выходе ЦАП12. Пользователь должен гарантировать, что время установки не будет нарушено при использовании контроллера DMA. См. справочные данные конкретного устройства для выяснения конкретных параметров.

19.2.7. Прерывания ЦАП12

Вектор прерываний ЦАП12 является общим с контроллером DMA. Программное обеспечение должно проверять флаги DAC12IFG и DMAIFG для определения источника прерывания.

Бит DAC12IFG устанавливается, когда DAC12xLSELx>0 и данные ЦАП12 защелкнуты от регистра DAC12_xDAT в защелке данных. Когда DAC12xLSELx=0, флаг DAC12IFG не устанавливается.

Установленный бит DAC12IFG показывает, что ЦАП12 готов для приема новых данных. Если установлены оба бита DAC12IE и GIE, DAC12IFG генерирует запрос прерывания. Флаг DAC12IFG не сбрасывается автоматически. Его должно сбрасывать программное обеспечение.

19.3. Регистры ЦАП12

Регистры ЦАП12 приведены в таблице 19-2.

Таблица 19-2. Регистры ЦАП12

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Управление DAC12_0	DAC12_0CTL	Чтение/запись	01C0h	Сброс с POR
Данные DAC12_0	DAC12_0DAT	Чтение/запись	01C8h	Сброс с POR
Управление DAC12_1	DAC12_1CTL	Чтение/запись	01C2h	Сброс с POR
Данные DAC12_1	DAC12_1DAT	Чтение/запись	01CAh	Сброс с POR

DAC12_xCTL, управляющий регистр ЦАП12

15	14	13	12	11	10	9	8
Зарезервировано	DAC12SREFx		DAC12RES	DAC12LSELx		DAC12 CALON	DAC12IR
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	1
			DAC12DF	DAC12IE	DAC12IFG	DAC12ENC	DAC12 GRP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Модифицируется, только когда DAC12ENC=0

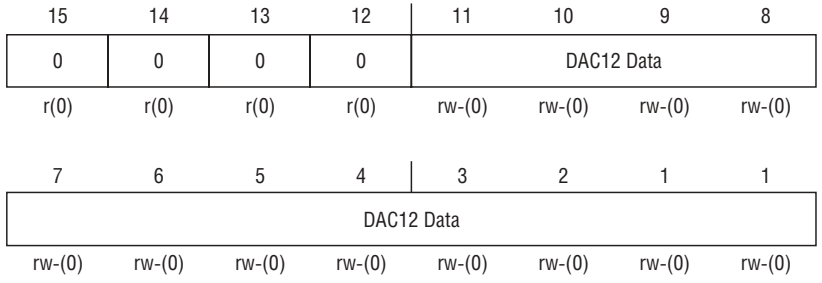
Зарезервирован	**Бит 15**	Зарезервирован
DAC12SREFx	**Биты 14-13**	Выбор опорного напряжения ЦАП12 00 – VREF+ 01 – VREF+ 10 – VeREF+ 11 – VeREF+

Таблица 19-2. (Окончание)

<table>
<tr><td>DAC12RES</td><td>Бит 12</td><td>Выбор разрешения ЦАП12
0 – 12-разрядное разрешение
1 – 8-разрядное разрешение</td></tr>
<tr><td>DAC12LSELx</td><td>Биты11-10</td><td>Выбор загрузки ЦАП12. Выбирается сигнал запуска загрузки защелки ЦАП12. Для обновления ЦАП должен быть установлен DAC12ENC, за исключением случая, когда DAC12LSELx=0.
00 – Загрузка в защелку ЦАП12 выполняется при записи в DAC12_xDAT (DAC12ENC игнорируется)
01 – Загрузка в защелку ЦАП12 выполняется при записи в DAC12_xDAT, или, когда используется группировка, при записи во все регистры DAC12_xDAT группы
10 – Фронт сигнала с Таймера_A3. Выход 1 (TA1)
11 – Фронт сигнала с Таймера_B7. Выход 2 (TB2)</td></tr>
<tr><td>DAC12CALON</td><td>Бит 9</td><td>Включение калибровки ЦАП12. Этот бит инициирует последовательность калибровки смещения ЦАП12 и сбрасывается автоматически после завершения калибровки.
0 – Калибровка не выполняется
1 – Инициирование калибровки / выполняется калибровка</td></tr>
<tr><td>DAC12IR</td><td>Бит 8</td><td>Входной диапазон ЦАП12. Этот бит устанавливает диапазон входного опорного напряжения и выходного напряжения.
0 – Полный диапазон выходного напряжения ЦАП12 равен 3-х кратному опорному напряжению
1 – Полный диапазон выходного напряжения ЦАП12 равен 1-но кратному опорному напряжению</td></tr>
<tr><td>DAC12AMPx</td><td>Биты 7-5</td><td>Настройка усилителя ЦАП12. Эти биты выбирают время установки в зависимости от потребляемого тока для входного и выходного усилителей ЦАП12
<table>
<tr><th>DAC12AMPx</th><th>Входной буфер</th><th>Выходной буфер</th></tr>
<tr><td>000</td><td>Выключен</td><td>ЦАП12 выключен, выход в высокоимпедансном состоянии</td></tr>
<tr><td>001</td><td>Выключен</td><td>ЦАП12 выключен, на выходе 0В</td></tr>
<tr><td>010</td><td>Низкая скорость/ток</td><td>Низкая скорость/ток</td></tr>
<tr><td>011</td><td>Низкая скорость/ток</td><td>Средняя скорость/ток</td></tr>
<tr><td>100</td><td>Низкая скорость/ток</td><td>Высокая скорость/ток</td></tr>
<tr><td>101</td><td>Средняя скорость/ток</td><td>Средняя скорость/ток</td></tr>
<tr><td>110</td><td>Средняя скорость/ток</td><td>Высокая скорость/ток</td></tr>
<tr><td>111</td><td>Высокая скорость/ток</td><td>Высокая скорость/ток</td></tr>
</table></td></tr>
<tr><td>DAC12DF</td><td>Бит 4</td><td>Формат данных ЦАП12
0 – Натуральный двоичный
1 – Формат с дополнением до двух</td></tr>
<tr><td>DAC12IE</td><td>Бит 3</td><td>Разрешение прерывания от ЦАП12
0 – Запрещено
1 – Разрешено</td></tr>
</table>

DAC12IFG	**Бит 2**	Флаг прерывания ЦАП12 0 – Прерывание не ожидается 1 – Прерывание ожидается
DAC12ENC	**Бит 1**	Включение преобразования ЦАП12. Этот бит включает модуль ЦАП12, когда DAC12LSELx>0. Когда DAC12LSELx=0, бит DAC12ENC игнорируется. 0 – ЦАП12 выключен 1 – ЦАП12 включен
DAC12GRP	**Бит 0**	Группировка ЦАП12. Группируется DAC12_x с DAC12_x, имеющий следующий более высокий порядковый номер. Не используется в устройствах MSP430x15x и MSP430x16x. 0 – Нет группировки 1 – ЦАП'ы сгруппированы

DAC12_xDAT, регистр данных ЦАП12



Не используется	**Биты 15-12**	Не используется. Эти биты всегда равны 0 и не влияют на ядро ЦАП12.	
Данные ЦАП12	**Биты 11-0**	Данные ЦАП12	
		Формат данных ЦАП12	**Данные ЦАП12**
		12-разрядный двоичный	Данные ЦАП12 выровнены по правому краю. Бит 11 является старшим значащим битом (MSB).
		12-разрядный с дополнением до двух	Данные ЦАП12 выровнены по правому краю. Бит 11 является старшим значащим битом MSB (знак).
		8-разрядный двоичный	Данные ЦАП12 выровнены по правому краю. Бит 7 является старшим значащим битом (MSB). Биты 11-8 не имеют значения и не влияют на ядро ЦАП12.
		8-разрядный с дополнением до двух	Данные ЦАП12 выровнены по правому краю. Бит 7 является старшим значащим битом MSB (знак). Биты 11-8 не имеют значения и не влияют на ядро ЦАП12.

Семейство микроконтроллеров MSP430x1xx
Руководство пользователя

Руководитель проекта	*Таранков И.В.*
Дизайн обложки	*Георгадзе Е.С.*
Графика	*Писанко В.А.*
Верстка	*Торочков Е.В.*

Подписано в печать 23.09.2004 г. Формат 62×90/16
Печать офсетная. Бумага ролевая. Гарнитура «HeliosCondenced»
Обложка – Бумага мел. импортная. Формат 62×64/8
Усл. печ. л. 23. Тираж 2000 экз. Зак. № 4157

Отпечатано в ГП «Московская типография 13»
Денисовский пер., дом 30
Тел./факс (095) 261-4884
www.printshor13.ru