

# **Семейство микроконтроллеров MSP430x4xx**

Руководство пользователя

**MSP430x4xxFamily**



C12    **Семейство микроконтроллеров MSP430x4xx.**

Руководство пользователя: Пер. с англ. – М.:

Серия «Библиотека Компэл». ЗАО «Компэл», 2005. – 416 с.

**ISBN 5-98730-003-7**

Данное издание представляет собой перевод User's Guide MSP430x4xx Family (slau056d) выпущенного компанией Texas Instruments в апреле 2004 года. В переводе учтены и исправлены все опечатки отмеченные в Errata MSP430x4xx Family User's Guide (slaz008)(июль 2004), а также ряд ошибок, обнаруженных в процессе перевода.

Руководство пользователя MSP430x4xx – рабочий инструмент инженера-разработчика в профессиональной работе по созданию новых электронных устройств на базе популярных микроконтроллеров компании Texas Instruments.

**ISBN 5-98730-003-7**



9 785987 300039

ISBN 5-98730-003-7

© ЗАО «Компэл», 2005

## Содержание

---

# MSP430x4xxFamily



<b>К ЧИТАТЕЛЮ</b>	9
<b>Введение</b>	12
Об этом руководстве .....	12
Дополнительная документация .....	12
Предупреждение FCC .....	12
Принятые обозначения .....	12
<b>Глоссарий</b> .....	13
Соглашения в обозначениях состояния битов регистров .....	14
<b>Раздел I. Введение</b> .....	16
1.1. Архитектура .....	16
1.2. Гибкая система тактирования .....	17
1.3. Встроенная эмуляция .....	18
1.4. Адресное пространство .....	18
1.4.1. Flash-память программ .....	19
1.4.2. ОЗУ .....	19
1.4.3. Периферийные модули .....	19
1.4.4. Регистры специального назначения (SFRs) .....	19
1.4.5. Организация памяти .....	20
<b>Раздел II. Системный сброс, прерывания и рабочие режимы</b> .....	22
2.1. Системный сброс и инициализация .....	22
2.1.1. Сброс при пониженном напряжении питания (BOR) .....	23
2.1.2. Исходное состояние устройства после системного сброса .....	24
2.2. Прерывания .....	24
2.2.1. Немаскируемые прерывания .....	25
2.2.2. Маскируемые прерывания .....	28
2.2.3. Обработка прерывания .....	29
2.2.4. Векторы прерываний .....	31
2.2.5. Специальные функции регистров (SFRs) .....	31
2.3. Режимы работы .....	32
2.3.1. Вход и выход из режимов пониженного энергопотребления .....	34
2.4. Принципы создания приложений с низким энергопотреблением .....	35
2.5. Подключение неиспользуемых выводов .....	36
<b>Раздел III. 16-разрядное RISC CPU</b> .....	38
3.1. Введение в ЦПУ .....	38
3.2. Регистры ЦПУ .....	38
3.2.1. Программный счетчик (PC) .....	38
3.2.2. Указатель стека (SP) .....	40
3.2.3. Регистр статуса (SR) .....	41
3.2.4. Регистры генератора констант CG1 и CG2 .....	43
3.2.5. Регистры общего назначения R4-R15 .....	44
3.3. Режимы адресации .....	44
3.3.1. Регистровый режим .....	45
3.3.2. Индексный режим .....	46
3.3.3. Символьный режим .....	47
3.3.4. Абсолютный режим .....	48
3.3.5. Косвенный регистровый режим .....	49
3.3.6. Косвенный автоинкрементный режим .....	50
3.3.7. Прямой режим .....	51
3.4. Набор команд .....	52
3.4.1. Команды с двойным операндом (Формат I) .....	53
3.4.2. Команды с одним операндом (Формат II) .....	54
3.4.3. Команды перехода .....	54
3.4.4. Командные циклы и длина команд .....	91
3.4.5. Описание набора команд .....	93

<b>Раздел IV. Модуль тактирования FLL+</b>	98
4.1. Введение в Модуль тактирования FLL+ .....	98
4.2. Работа модуля тактирования FLL+ .....	100
4.2.1. Особенности работы модуля тактирования FLL+ в системах с низким энергопотреблением.....	101
4.2.2. Генератор LFXT1 .....	101
4.2.3. Генератор XT2 .....	102
4.2.4. Генератор с цифровым управлением (DCO) .....	103
4.2.5. Автоподстройка частоты (FLL) .....	103
4.2.6. Модулятор генератора DCO .....	104
4.2.7. Запрос устройства стабилизации частоты FLL и модулятора .....	104
4.2.8. Работа модуля FLL в режимах пониженного энергопотребления .....	104
4.2.9. Буферизованный выход тактовой частоты .....	105
4.2.10. Бессбонаяя работа модуля FLL+ .....	105
4.3. Регистры модуля тактирования FLL+ .....	107
<b>Раздел V. Контроллер флэш-памяти</b>	112
5.1. Введение в флэш-память .....	112
5.2. Сегментация флэш-памяти .....	113
5.3. Функционирование флэш-памяти.....	113
5.3.1. Тактовый генератор флэш-памяти .....	114
5.3.2. Стирание флэш-памяти .....	115
5.3.3. Запись в флэш-память .....	118
5.3.4. Доступ к флэш-памяти во время записи или стирания .....	125
5.3.5. Останов цикла записи или стирания.....	125
5.3.6. Конфигурирование и доступ к контроллеру флэш-памяти .....	126
5.3.7. Прерывания контроллера флэш-памяти.....	126
5.3.8. Программирование устройств с флэш-памятью.....	127
5.4. Регистры флэш-памяти.....	128
<b>Раздел VI. Супервизор напряжения питания</b>	134
6.1. Введение в SVS.....	134
6.2. Функционирование SVS .....	134
6.2.1. Конфигурирование SVS .....	134
6.2.2. Функционирование компаратора SVS .....	135
6.2.3. Изменение битов VLxD .....	136
6.2.4. Рабочий диапазон SVS .....	136
6.3. Регистры SVS .....	137
<b>Раздел VII. Аппаратный умножитель</b>	140
7.1. Введение в аппаратный умножитель .....	140
7.2. Функционирование аппаратного умножителя .....	141
7.2.1. Операнд регистров .....	141
7.2.2. Регистры результата.....	141
7.2.3. Примеры программного обеспечения .....	143
7.2.4. Косвенная адресация RESLO .....	143
7.2.5. Использование прерываний .....	144
7.3. Регистры аппаратного умножителя .....	144
<b>Раздел VIII. Контроллер DMA</b>	146
8.1. Введение в контроллер DMA.....	146
8.2. Функционирование DMA .....	146
8.2.1. Режимы адресации DMA.....	146
8.2.2. Режимы переноса DMA .....	149
8.2.3 Инициирование DMA-переносов .....	154
8.2.4. Останов DMA-переносов .....	157
8.2.5. Приоритеты каналов DMA.....	157
8.2.6. Длительность цикла DMA-переноса .....	157
8.2.7. Использование DMA с системными прерываниями .....	158

8.2.8. Прерывания контроллера DMA .....	158
8.2.9. Использование модуля I <sup>2</sup> C с контроллером DMA .....	159
8.2.10. Использование АЦП12 с контроллером DMA .....	159
8.2.11. Использование ЦАП12 с контроллером DMA .....	159
8.3. Регистры DMA .....	160
<b>Раздел IX. Цифровые входы/выходы .....</b>	166
9.1. Введение в цифровые входы/выходы .....	166
9.2. Функционирование цифровых входов/выходов .....	166
9.2.1. Регистры ввода PxIN .....	166
9.2.2. Регистры вывода PxOUT .....	167
9.2.3. Регистры направления PxDIR .....	167
9.2.4. Регистры выбора функции PxSEL .....	167
9.2.5. Прерывания P1 и P2 .....	168
9.2.6. Конфигурирование неиспользуемых выводов порта .....	169
9.3. Регистры цифровых входов/выходов .....	169
<b>Раздел X. Сторожевой таймер .....</b>	172
10.1. Введение в сторожевой таймер .....	172
10.2. Функционирование сторожевого таймера .....	172
10.2.1. Счетчик сторожевого таймера .....	173
10.2.2. Сторожевой режим .....	174
10.2.3. Режим интервального таймера .....	174
10.2.4. Прерывания сторожевого таймера .....	174
10.2.5. Усовершенствованный сторожевой таймер WDT+ .....	175
10.2.6. Работа в режимах пониженного энергопотребления .....	175
10.2.6. Примеры программного обеспечения .....	176
10.3. Регистры сторожевого таймера .....	176
<b>Раздел XI. Базовый таймер .....</b>	180
11.1. Модуль базового таймера Basic Timer 1 – введение .....	180
11.2. Работа модуля базового таймера Basic Timer 1 .....	181
11.2.1. Счётчик №1 модуля базового таймера Basic Timer 1 .....	181
11.2.2. Счётчик №2 модуля базового таймера Basic Timer1 .....	181
11.2.3. Режим 16-битного счётика .....	181
11.2.4. Работа базового таймера Basic Timer1: сигнал f <sub>LOD</sub> .....	182
11.2.5. Прерывания базового таймера Basic Timer1 .....	182
11.3. Регистры базового таймера Basic Timer1 .....	182
<b>Раздел XII. Таймер А .....</b>	186
12.1. Введение в таймер А .....	186
12.2. Функционирование таймера А .....	186
12.2.1. 16-разрядный таймер-счетчик .....	187
12.2.2. Запуск таймера .....	188
12.2.3. Управление режимом таймера .....	188
12.2.4. Блоки захвата/сравнения .....	193
12.2.5. Модуль вывода .....	195
12.2.6. Прерывания Таймера А .....	197
11.3. Регистры Таймера А .....	201
<b>Раздел XIII. Таймер В .....</b>	208
13.1. Введение в таймер В .....	208
13.1.1. Сходства и различия с таймером А .....	208
13.2. Функционирование таймера В .....	209
13.2.1. 16-разрядный счетчик таймера .....	210
13.2.2. Старт таймера .....	210
13.2.3. Управление режимом таймера .....	211
13.2.4. Блоки захвата/сравнения .....	215
13.2.5. Модуль вывода .....	218

13.2.6. Прерывания Таймера В .....	221
13.3. Регистры таймера В .....	224
<b>Раздел XIV. Периферийный интерфейс USART, режим UART</b> .....	230
14.1. Введение в USART: режим UART .....	230
14.2. Функционирование USART: режим UART .....	230
14.2.1. Инициализация и сброс USART .....	230
14.2.2. Формат символа .....	232
14.2.3. Асинхронные коммуникационные форматы .....	232
14.2.4. Разрешение приема USART .....	236
14.2.5. Разрешение передачи USART .....	237
14.2.6. Контроллер скорости передачи UART .....	238
14.2.7. Прерывания USART .....	244
14.3. Регистры USART: режим USART .....	248
<b>Раздел XV. Периферийный интерфейс USART, режим SPI</b> .....	256
15.1. Введение в USART: режим SPI .....	256
15.2. Функционирование USART: режим SPI .....	256
15.2.1. Инициализация USART и сброс .....	258
15.2.2. Режим ведущего .....	258
15.2.3. Режим ведомого .....	259
15.2.4. Включение SPI .....	260
15.2.5. Управление последовательным тактированием .....	261
15.2.6. Прерывания SPI .....	263
15.3. Регистры USART: режим SPI .....	264
<b>Раздел XVI. Модуль операционного усилителя OA</b> .....	274
16.1. Модуль операционного усилителя OA – введение .....	274
16.2. Работа модуля операционного усилителя OA .....	274
16.2.1. Усилитель OA .....	274
16.2.2. Вход OA .....	276
16.2.3. Выход OA .....	276
16.2.4. Конфигурация OA .....	276
16.3. Регистры модуля операционного усилителя OA .....	281
<b>Раздел XVII. Компаратор А</b> .....	286
17.1. Введение в компаратор А .....	286
17.2. Функционирование компаратора А .....	287
17.2.1. Компаратор .....	287
17.2.2. Входные аналоговые переключатели .....	287
17.2.3. Выходной фильтр .....	288
17.2.4. Генератор опорного напряжения .....	288
17.2.5. Компаратор А, регистр отключения порта CAPD .....	289
17.2.6. Прерывания компаратора А .....	289
17.2.7. Использование компаратора А для измерения сопротивления элементов .....	290
17.3. Регистры компаратора А .....	292
<b>Раздел XVIII. Контроллер ЖКИ</b> .....	296
18.1. Модуль контроллера ЖКИ – введение .....	296
18.2. Работа модуля контроллера ЖКИ .....	297
18.2.1. Дисплейная память .....	297
18.2.2. Мигание ЖКИ .....	298
18.2.3. Генерация тактовых сигналов .....	299
18.2.4. Генерация напряжений ЖКИ .....	299
18.2.5. Выходы модуля контроллера ЖКИ .....	299
18.2.6. Статический режим .....	300
18.2.7. Режим двойного мультиплексирования (2-Mux) .....	303
18.2.8. Режим тройного мультиплексирования (3-Mux) .....	305
18.2.9. Режим четырёхкратного мультиплексирования (4-Mux) .....	309
18.3. Регистры модуля контроллера ЖКИ .....	312

<b>Раздел XIX. АЦП12</b>	316
19.1. Введение в АЦП12	316
19.2. Функционирование АЦП12	316
19.2.1. 12-разрядное ядро АЦП	317
19.2.2. Входы АЦП12 и мультиплексор	318
19.2.3. Генератор опорного напряжения	319
19.2.4. Синхронизация выборки и преобразования	320
19.2.5. Память преобразований	322
19.2.6. Использование интегрированного температурного датчика	328
19.2.8. Заземление АЦП12 и рассмотрение влияния помех	328
19.2.9. Прерывания АЦП12	329
19.3. Регистры АЦП12	333
<b>Раздел XX. Модуль АЦП SD16</b>	342
20.1. Модуль АЦП SD16 – введение	342
20.2. Работа модуля АЦП SD16	342
20.2.1. Ядро аналого-цифрового преобразователя	342
20.2.2. Цифровой фильтр	343
20.2.3. Диапазон аналогового сигнала и усилитель с программируемым коэффициентом усиления (PGA)	345
20.2.4. Источник опорного напряжения	346
20.2.5. Регистры памяти преобразований: SD16MEMx	346
20.2.6. Выбор канала	348
20.2.7. Режимы преобразования	349
20.2.8. Режим преобразования с предварительной загрузкой	352
20.2.9. Использование встроенного датчика температуры	353
20.2.10. Обработка прерываний	355
20.3. Регистры модуля АЦП SD16	357
<b>Раздел XXI. ЦАП12</b>	364
21.1. Введение в ЦАП12	364
21.2. Функционирование ЦАП12	364
21.2.1. Ядро ЦАП12	365
21.2.2. Опорный источник ЦАП12	366
21.2.3. Обновление выходного напряжения ЦАП12	367
21.2.4. Формат данных DAC12_xDAT	367
21.2.5. Калибровка смещения выходного усилителя ЦАП12	368
21.2.6. Группировка нескольких модулей ЦАП12	370
21.2.7. Прерывания ЦАП12	371
21.3. Регистры ЦАП12	371
<b>Раздел XXII. Модуль детекторов SCAN IF</b>	376
22.1. Модуль детекторов SCAN IF – введение	376
22.2. Работа модуля АЦП SD16	376
22.2.1. Блок аналоговых формирователей	376
22.2.3. Блок автомата состояний модуля Scan IF	389
22.2.4. Регистр отладки модуля Scan IF	394
22.2.5. Прерывания модуля Scan IF	394
22.2.6. Использование модуля Scan IF с датчиками LC-типа	395
22.2.7. Использование модуля Scan IF с резистивными датчиками	397
22.2.8. Квадратурное декодирование	399
22.3. Регистры модуля Scan IF	402

## К ЧИТАТЕЛЮ



Вы держите в руках уже третье издание, посвященное семейству MSP430, которое выпущено компанией «КОМПЭЛ». Почему мы обратились к данной тематике и с неослабевающим упорством переводим и издаем техническую документацию компании Texas Instruments? Еще в предисловии к первому изданию мы отмечали, что микроконтроллеры семейства MSP430 не имеют себе равных по соотношению функциональных возможностей и цене. Этот фактор и стал причиной их невероятной мировой популярности. Однако в нашей стране эта популярность ощущается не столь явно. Что же вызвало этот дисбаланс? С одной стороны – грамотная целенаправленная работа дистрибуторов компаний Atmel и Microchip по продвижению микроконтроллеров этих производителей на отечественном рынке, с другой стороны – катастрофический недостаток русскоязычной технической литературе по продукции Texas Instruments.

Чтобы устраниТЬ этот дисбаланс и восстановить «справедливость», мы и предприняли эту серию публикаций. На сайте компании Texas Instruments <http://www.ti.com> представлено огромное количество технической информации, однако языковый барьер иногда мешает отечественным разработчикам в полном объеме воспользоваться существующими возможностями. Более того, Texas Instruments регулярно обновляет и пополняет эту информацию, поэтому изданием данного руководства, наша работа по популяризации продукции этого производителя не завершается. Во второй половине 2005 года мы планируем выпустить вторую часть рекомендаций по применению MSP430, куда войдут переводы статей по применению, вышедшие в настоящем году и руководство пользователя по новейшей серии MSP430x2xx.

Ваши замечания и предложения по материалам данной книги просим присыпать по адресу:  
**E-mail: TI@compel.ru.**

Бренд-менеджер  
по продукции Texas Instruments  
компании «КОМПЭЛ» – Илья Фурман.



## Предисловие

---

# MSP430x4xxFamily



## Введение

### Об этом руководстве

В настоящем руководстве рассматриваются модули и периферийные устройства семейства микроконтроллеров MSP430x4xx. 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, ЦДП)	
BOR	Brown-Out Reset (Сброс при пониженном питании напряжении)	«Системы сброса, прерываний и режимы работы»
BSL	Bootstrap Loader (начальный загрузчик программной памяти или ОЗУ)	<a href="http://www.ti.com/msp430">www.ti.com/msp430</a>
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»

<b>Сокращение</b>	<b>Значение</b>	<b>Раздел с подробным описанием</b>
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*

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

# MSP430x4xxFamily

## Введение

---

*Раздел I.*



## 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 (МАВ) памяти и шину памяти данных (MDB). Объединяя современное CPU с отображаемыми в памяти аналого-выми и цифровыми периферийными устройствами, семейство MSP430 предлагает решения для приложений со смешанными сигналами.

**Семейство MSP430 обладает следующими ключевыми особенностями:**

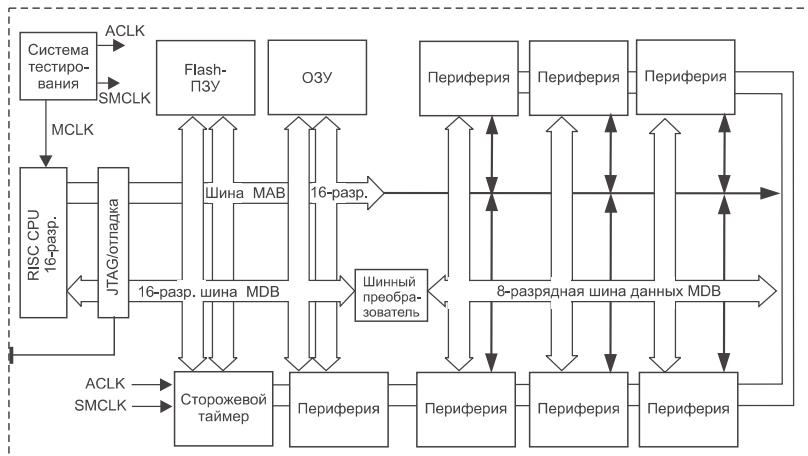
- Архитектура с ультранизким потреблением, увеличивающая время работы при питании от батарей:
  - для сохранности содержимого ОЗУ необходим ток не более 0,1 мкА;
  - модуль тактирования реального времени потребляет 0,8 мкА;
  - ток потребления при максимальной производительности составляет 250 мкА;
- Высококачественная аналоговая периферия для выполнения точных измерений:
  - встроенные модули 12-разрядного или 10-разрядного АЦП скоростью 200 ksp/s;
  - 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 высокουровневого 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.



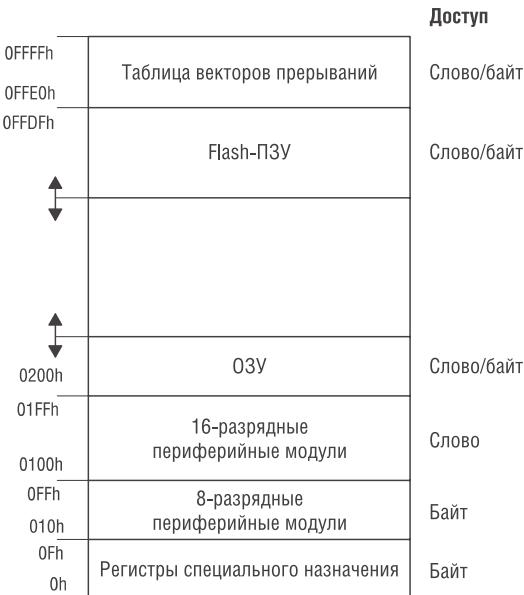
**Рис. 1-1. Архитектура MSP430**

## 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. Кarta 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 единственным адресным пространством для регистров специального назначения (SFR), перифе-

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 (0FFEh).

#### 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 OFFh 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.

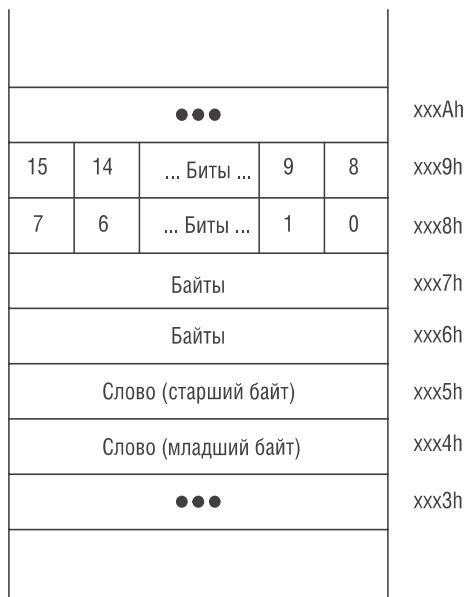
#### 1.4.4. Регистры специального назначения (SFRs)

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

# **Системный сброс, прерывания и рабочие режимы**

---

*Раздел II.*

**MSP430x4xxFamily**

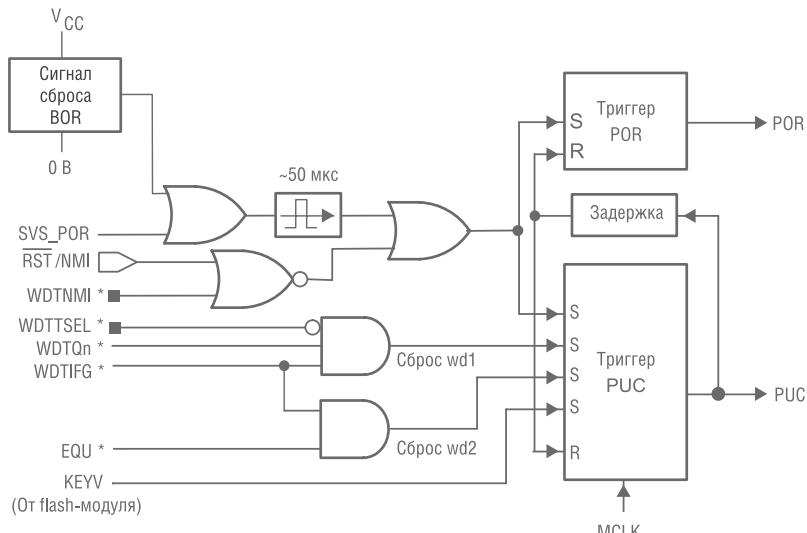


## Системный сброс, прерывания и рабочие режимы

Этот раздел описывает системный сброс, прерывания и рабочие режимы семейства MSP430x4xx.

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



\* Со сторожевого таймера

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

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.

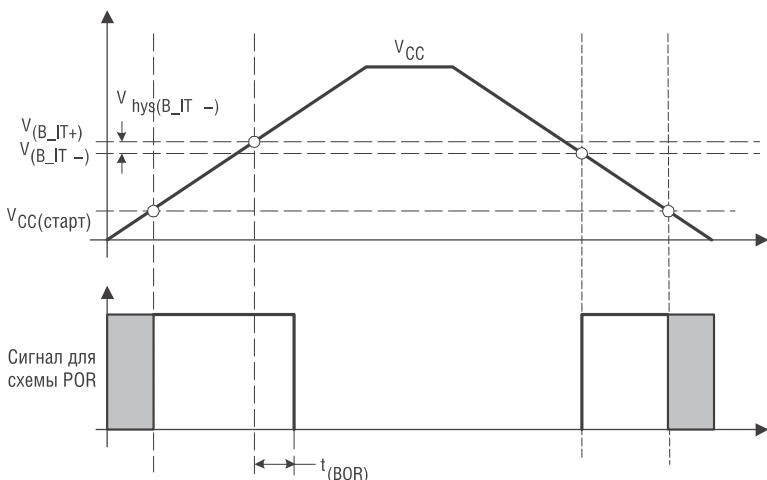
Сигнал PUC генерируется всегда при появлении сигнала POR, сигнал POR не генерируется сигналом PUC. Следующие события приводят к появлению сигнала PUC:

- 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 пониженном напряжении питания (BOR)

Все устройства серии MSP430x4xx имеют схему сброса при пониженном напряжении питания. Схема сброса при пониженном напряжении детектирует понижение питающего напряжения, например, напряжение, подаваемое или снимаемое с вывода V<sub>cc</sub>. Схема сброса при пониженном напряжении сбрасывает устройство, вызывая появление POR-сигнала, когда напряжение прикладывается или снимается. Рабочие уровни показаны на рис. 2-2.

Сигнал POR становится активным, когда напряжение V<sub>cc</sub> достигает уровня V<sub>CC(start)</sub>. И остается активным до тех пор, пока V<sub>cc</sub> не пересечет порог V(B\_IT+) и не закончится задержка t(BOR). Адаптивная задержка t(BOR) бывает больше при медленном изменении V<sub>cc</sub>. Гистерезис V<sub>hys(B\_IT-)</sub> введен, чтобы гарантировать, что питающее напряжение должно снизиться ниже уровня V(B\_IT-), прежде чем схемой сброса будет сгенерирован другой сигнал POR.



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

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.

### 2.1.2. 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 векторе 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. 2-3. Приоритеты определяются порядком расположения модулей в соединяющей их цепи. Чем ближе модуль к ЦПУ/NMIRS, тем выше его приоритет.

**Прерывания делятся на три типа:**

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

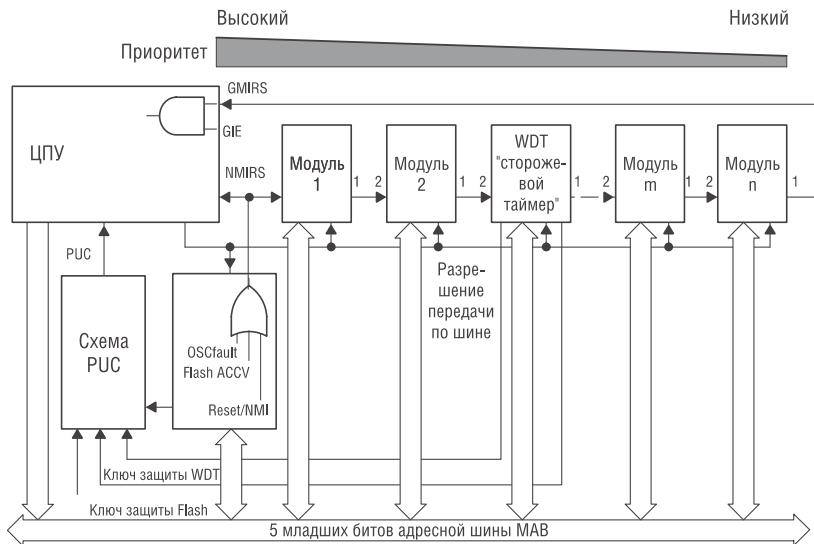


Рис. 2-3. Приоритеты прерываний

## 2.2.1. Немаскируемые прерывания

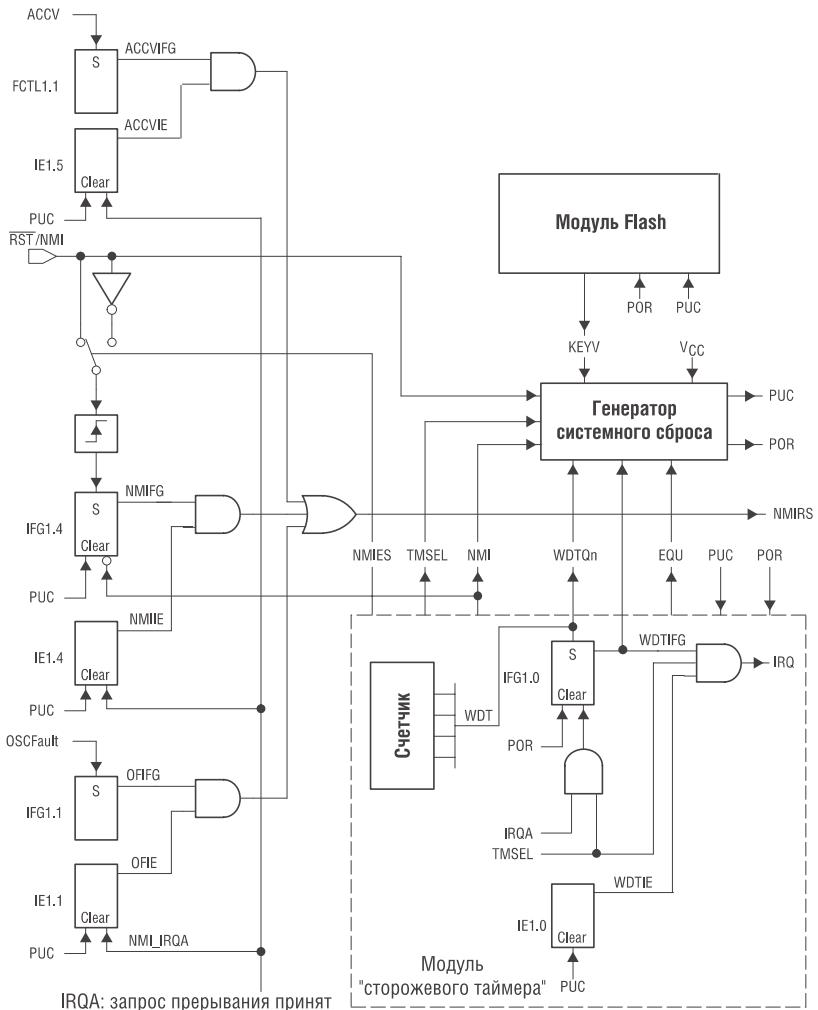
Немаскируемые прерывания NMI не маскируются общим битом разрешения прерываний (GIE), но могут управляться индивидуальными битами включения прерывания (ACCVIE, NMIIIE, OFIE). Когда происходит немаскируемое прерывание NMI, все биты разрешения NMI-прерываний автоматически сбрасываются. Выполнение программы 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. 2-4.

**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



**Рис. 2-4.** Блок-схема источников немаскируемого прерывания

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 векторе сброса (0FFFh).

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

**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.

## 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.

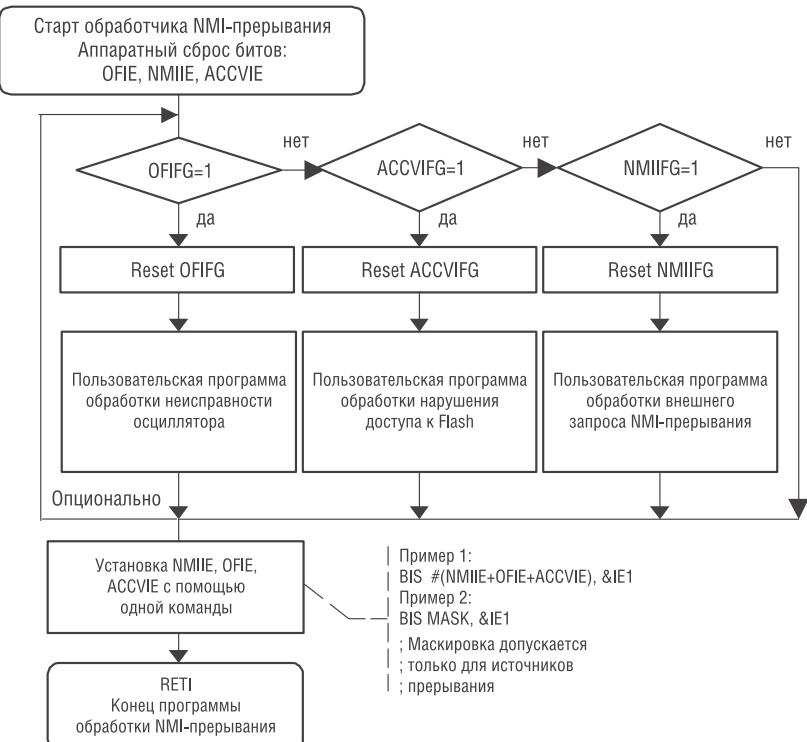
Сигнал неисправности осциллятора может быть вызван PUC-сигналом, поскольку он переводит осциллятор LFXT1 из режима HF в режим LF. Сигнал PUC также отключает осциллятор XT2.

## 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.

## Пример программы обработки немаскируемого прерывания NMI

NMI-прерывание имеет много возможных источников. NMI-прерывание автоматически сбрасывает биты разрешения прерываний NMIIIE, 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. 2-5.



**Рис. 2-5.** Алгоритм процедуры обработки NMI-прерывания

### 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 сразу с использованием общего бита разрешения всех прерываний (GIE) в регистре статуса (SR).

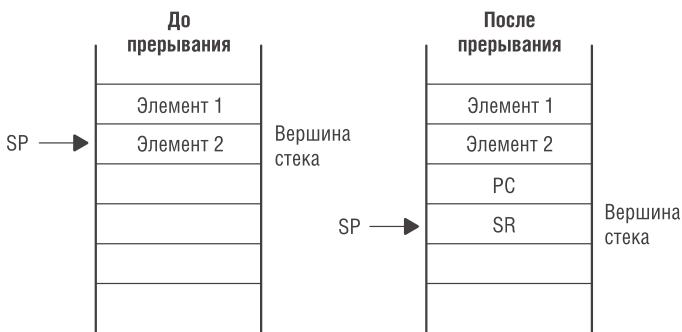
Каждое конкретное периферийное прерывание будет рассмотрено в этом руководстве в разделе описания соответствующего периферийного устройства.

### 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. 2-6. Логика обработки запроса прерывания имеет следующую последовательность:



*Рис. 2-6.* Процесс прерывания

- 1) Любая текущая команда выполняется до конца;
- 2) Содержимое программного счетчика PC, указывающего на следующую команду, помещается в стек;
- 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. Процессор переходит из режима пониженного потребления в активный режим;

7) Содержимое вектора прерывания загружается в РС 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. 2-7.



**Рис. 2-7.** Возврат из прерывания

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 РС и начинается выполнение программы с того места, где она была прервана.

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 OFFFFh по OFFE0h, 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	OFFFEh	15, наивысший
NMI-прерывание, неисправность осциллятора, нарушение доступа к Flash-памяти	NMIIFG OFIFG ACCVIFG	Немаскируемое Немаскируемое Немаскируемое	OFFFCCh	14
Определяется устройством			OFFFAh	13
Определяется устройством			OFFF8h	12
Определяется устройством			OFFF6h	11
Сторожевой таймер	WDTIFG	Маскируемое	OFFF4h	10
Определяется устройством			OFFF2h	9
Определяется устройством			OFFF0h	8
Определяется устройством			OFFEEh	7
Определяется устройством			OFFECh	6
Определяется устройством			OFFEAh	5
Определяется устройством			OFFE8h	4
Определяется устройством			OFFE6h	3
Определяется устройством			OFFE4h	2
Определяется устройством			OFFE2h	1
Определяется устройством			OFFE0h	0, низший

## 2.2.5. Специальные функции регистров (SFRs)

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.

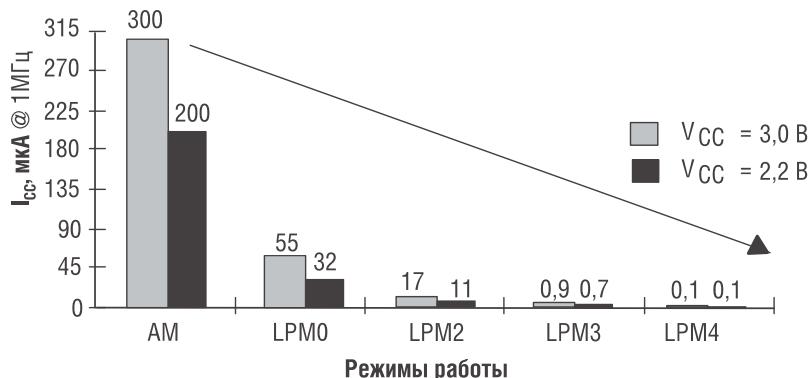
## 2.3. Режимы работы

Семейство MSP430 разработано для приложений с ультранизким потреблением мощности и имеет различные режимы работы, показанные на рис. 2-9.

**Режимы работы учитывают три различные потребности:**

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

Типичное потребление тока микроконтроллерами семейства MSP430 показано на рис. 2-8.



**Рис. 2-8.** Типичное потребление тока устройствами 41x в зависимости от режима работы

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 работы, если сохраненное содержимое регистра SR не было изменено процедурой обработки прерывания. Выполнение программы может продолжаться 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.

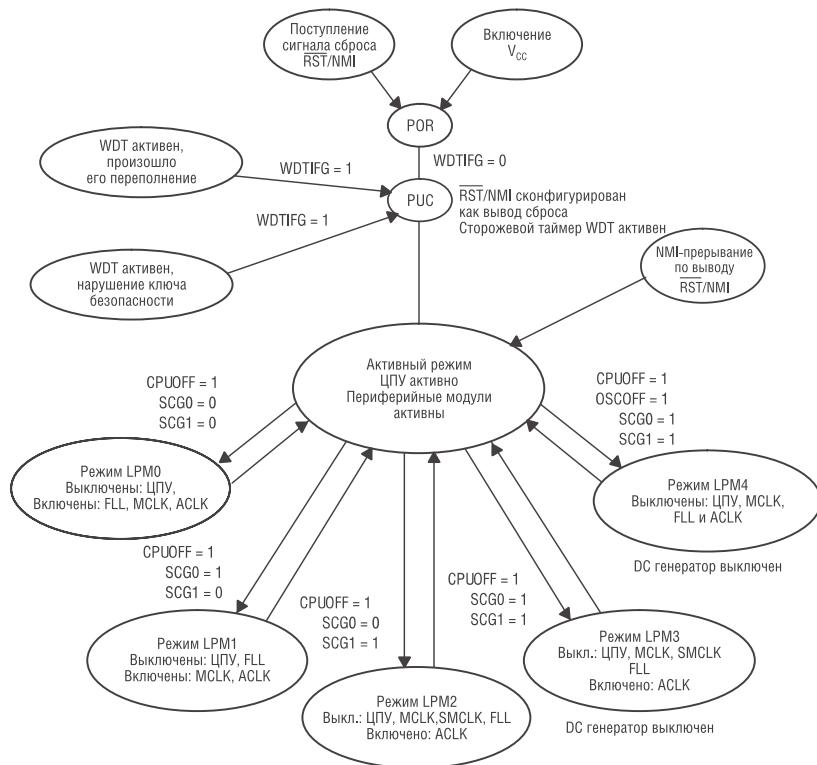


Рис. 2-9. Режимы работы основной системы тактирования MSP430x4x

SCG1	SCGO	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 отключен; FLL+, MCLK и ACLK активны
0	1	0	1	LPM1	ЦПУ и FLL+ отключены; SMCLK и ACLK активны

<b>SCG1</b>	<b>SCGO</b>	<b>OSCOFF</b>	<b>CPUOFF</b>	<b>Режим</b>	<b>Состояние ЦПУ и систем тактирования</b>
1	0	0	1	LPM2	ЦПУ, FLL+, MCLK отключены; DC генератор остается включенным; ACLK активно
1	1	0	1	LPM3	ЦПУ, FLL+, MCLK 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 LPM0
;...
;Останов
;программы
;

;Выход из режима LPM0 по процедуре обработки прерывания
BIC #CPUOFF, 0(SP) ;Выход из LPM0
;по RETI
RETI

;Пример входа в режим LPM3
BIS #GIE+CPUOFF+SCG1+SCGO,SR ;Вход в LPM3
;...
;Останов
;программы
;

;Выход из режима LPM3 по процедуре обработки прерывания
BIC #CPUOFF+SCG1+SCGO,0(SP) ;Выход из LPM3
;по RETI
RETI
```

## Нахождение в режимах пониженного энергопотребления в течение длительного времени

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.B #FN_8+FN_4+FN_3+FN_2,&SCFI0      ;Низший
                                                ;уровень
MOV.B #010h,&SCFI1                          ;Выбор
                                                ;Tap 2
BIS    #GIE+CPUOFF+OSCOFF+SCG1+SCG0,SR ;Вход
                                                ;в LPM4
; ...
                                                ;Останов
                                                ;программы

;Процедура обработки прерывания
BIC    #CPUOFF+OSCOFF+SCG1+SCG0,0 (SR) ;Выход
                                                ;из LPM4
                                                ;по 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 мА при функционирующей схеме тактирования реального времени и активированной системе прерываний. Модуль ACLK использует часовой 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 А и таймер В 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
AVCC	DVCC	
AVSS	DVSS	
VREF+	Свободный	
VeREF+	DVSS	
VREF-/VeREF-	DVSS	
XIN	DVCC	
XOUT	Свободный	
XT2IN	DVSS	Устройства 43х и 44х
XT2OUT	Свободный	Устройства 43х и 44х
с Px.0 по Px.7	Свободный	Переключены к функции порта, направленного на вывод
RST/NMI	DVCC или VCC	«Подтягивающий» резистор 47 кОм с понижающей емкостью 10 нФ
R03	DVSS	
COM0	Свободный	
TDO	Свободный	
TDI	Свободный	
TMS	Свободный	
TCK	Свободный	
Sxx	Свободный	

# MSP430x4xxFamily

## 16-разрядное RISC CPU

*Раздел III.*



# 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

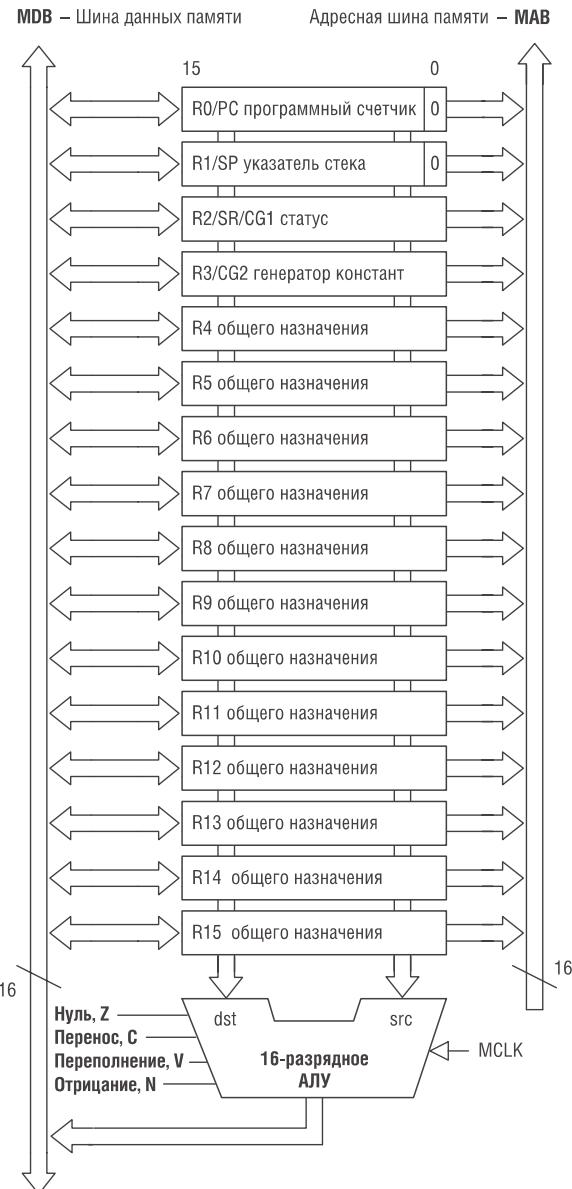


Рис. 3-1. Блок-схема ЦПУ

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



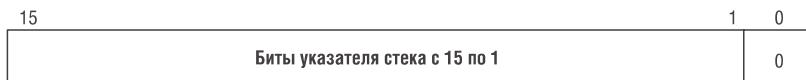
**Рис. 3-2.** Программный счетчик

Программный счетчик РС 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 переменной LABEL
MOV    @R14, PC        ;Косвенный переход по косвенному содержимому 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) содержимого 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.

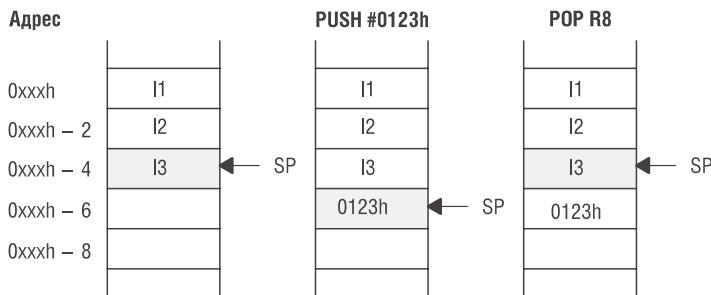


Рис. 3-4. Использование стека



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

Указатель стека изменяется после выполнения команды PUSH SP.

Указатель стека не изменяется после выполнения 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.

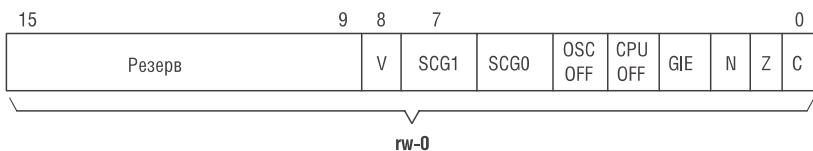


Рис. 3-6. Биты регистра статуса

В таблице 3.1 приведено описание битов регистра статуса.

**Таблица 3.1. Описание битов регистра статуса**

Бит	Описание	
<b>V</b>	Бит переполнения. Этот бит устанавливается, если результат арифметической операции имеет переполнение в области знаковых переменных.	
	<i>ADD (.B)</i> , <i>ADDC (.B)</i>	Устанавливается, когда: Положительный + Положительный = Отрицательный Отрицательный + Отрицательный = Положительный в противном случае сбрасывается
	<i>SUB (.B)</i> , <i>SUBC (.B)</i> , <i>CMP (.B)</i>	Устанавливается, когда: Положительный – Отрицательный = Отрицательный Отрицательный – Положительный = Позитивный в противном случае сбрасывается
<b>SCG1</b>	Системный тактовый генератор 1. Когда этот бит установлен, генератор DCO выключен, если DCOCLK не используется для MCLK или SMCLK.	
<b>SCG0</b>	Системный тактовый генератор 0. Когда этот бит установлен, Цепь управления FLL+ выключена.	
<b>OSCOFF</b>	Выключение осциллятора. Когда этот бит установлен, осциллятор LFXT1, использующий кристалл, выключен, если LFXT1CLK не используется для MCLK или SMCLK.	
<b>CPUOFF</b>	Выключение ЦПУ. Когда этот бит установлен, ЦПУ выключено.	
<b>GIE</b>	Общий бит разрешения прерываний. Когда этот бит установлен, маскируемые прерывания разрешены. Когда сброшен, все маскируемые прерывания запрещены.	
<b>N</b>	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	
<b>Z</b>	Бит нуля. Этот бит устанавливается, когда результат операции с байтом или словом равен «0» и очищается, если результат не равен «0».	
<b>C</b>	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
$R5=0A28Fh$ $R6=0203h$ $Mem(0203h)=012h$ $ADD.B \quad R5, 0(R6)$ $\quad \quad \quad 08Fh$ $\quad \quad \quad +012h$ $\hline$ $\quad \quad \quad 0A1h$ $Mem(0203h)=0A1h$ $C=0, Z=0, N=1$ $(младший байт регистра) + (адресуемый байт)$ $\hline$ $\rightarrow (адресуемый байт)$	$R5=01202Fh$ $R6=0223h$ $Mem(0223h)=05Fh$ $ADD.B \quad @R6, R5$ $\quad \quad \quad 05Fh$ $\quad \quad \quad +002h$ $\hline$ $\quad \quad \quad 00061h$ $R5=00061h$ $C=0, Z=0, N=0$ $(адресуемый байт) + (младший байт регистра)$ $\hline$ $\rightarrow (младший байт регистра, ноль в старшем байте)$

### 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 operand
11 / -	Косвенный автоинкремент	@Rn+	Содержимое Rn использовано как указатель на operand. 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, TONI, TOM и LEO

Везде в документации по семейству MSP430 используются универсальные метки EDE, TONI, TOM и LEO. 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 регистрационного 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

До		После	
R10	0A023h	R10	0A023h
R11	0FA15h	R11	0A023h
PC	PC <sub>old</sub>	PC	PC <sub>old</sub> +2

### **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
<b>MOV 2 (R5) , 6 (R6)</b>	<b>MOV X(R5) , Y(R6) X=2 Y=6</b>
<b>Длина:</b> Два или три слова	
<b>Операция:</b> 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.	
<b>Комментарий:</b> Действительно для источника и получателя	
<b>Пример:</b> MOV 2 (R5) , 6 (R6) :	

До:	Адресное пространство	Регистр	После:	Адресное пространство	Регистр
0FF16h	00006h	R5 01080h	0FF16h	0xxxxh	PC
0FF14h	00002h	R6 0108Ch	0FF14h	00006h	R5 01080h
0FF12h	04596h	PC	0FF12h	00002h	R6 0108Ch
				04596h	
01094h	0xxxxh	0108Ch +0006h 01092h	01094h	0xxxxh	
01092h	05555h		01092h	01234h	
01090h	0xxxxh		01090h	0xxxxh	
01084h	0xxxxh	01080h +0002h 01082h	01084h	0xxxxh	
01082h	01234h		01082h	01234h	
01080h	0xxxxh		01080h	0xxxxh	

### 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-PC Y=TONI-PC
<b>Длина:</b> Два или три слова	
<b>Операция:</b> 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.	
<b>Комментарий:</b> действително для источника и получателя	
<b>Пример:</b> MOV EDE, TONI ;Адрес источника EDE=0F016h ;Адрес получателя TONI=01114h	

До:	Адресное пространство	Регистр	После:	Адресное пространство	Регистр
0FF16h	011FEh		0FF16h	011FEh	PC
0FF14h	0F102h		0FF14h	0F102h	
0FF12h	04090h	PC	0FF12h	04090h	
0F018h	0xxxxh		0F018h	0xxxxh	
0F016h	0A123h		0F016h	0A123h	
0F014h	0xxxxh		0F014h	0xxxxh	
01116h	0xxxxh		01116h	0xxxxh	
01114h	05555h		01114h	0A123h	
01112h	0xxxxh		01112h	0xxxxh	

### 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=EDE Y=TONI
<b>Длина:</b> Два или три слова	
<b>Операция:</b> 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.	
<b>Комментарий:</b> Действительно для источника и получателя	
<b>Пример:</b> MOV &EDE, &TONI ;Адрес источника EDE=0F016h ;Адрес получателя TONI=01114h	

До:	Адресное пространство	Регистр	До:	Адресное пространство	Регистр
0FF16h	01114h	PC	0FF16h	0xxxxh	PC
	0F016h			01114h	
	04292h			0F016h	
				04292h	
0F018h	0xxxxh		0F018h	0xxxxh	
	0A123h			0A123h	
	0xxxxh			0F014h	
01116h	0xxxxh		01116h	0xxxxh	
	01234h			0A123h	
	0xxxxh			01112h	

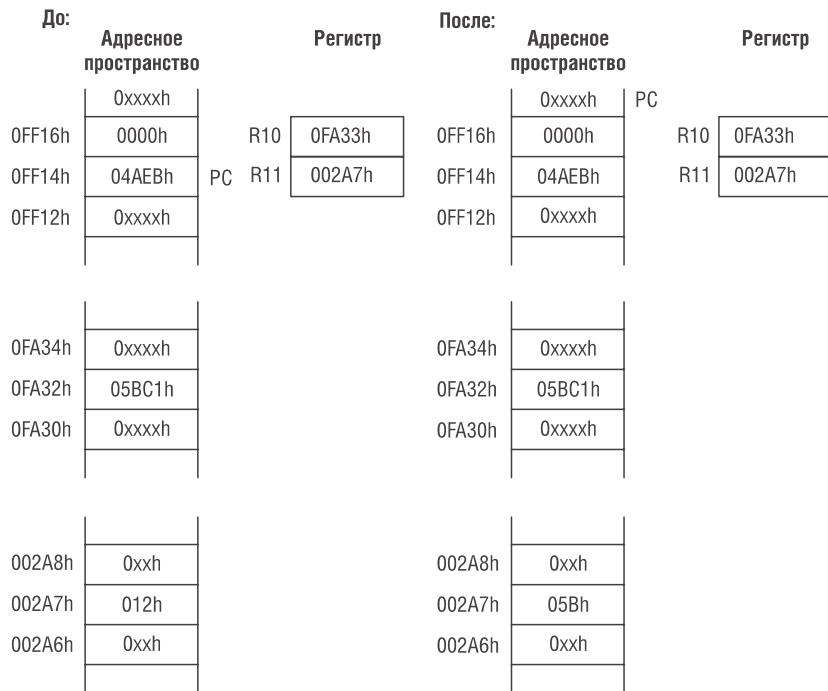
Этот режим адресации предназначен главным образом для аппаратных периферийных модулей, расположенных по абсолютным, фиксированным адресам. Они адресуются в абсолютном режиме, что гарантирует переносимость программы (например, при написании позиционно-независимого, переносимого кода).

### 3.3.5. Косвенный регистровый режим

Косвенный регистровый режим описан в таблице 3.8.

**Таблица 3.8. Описание косвенного режима**

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

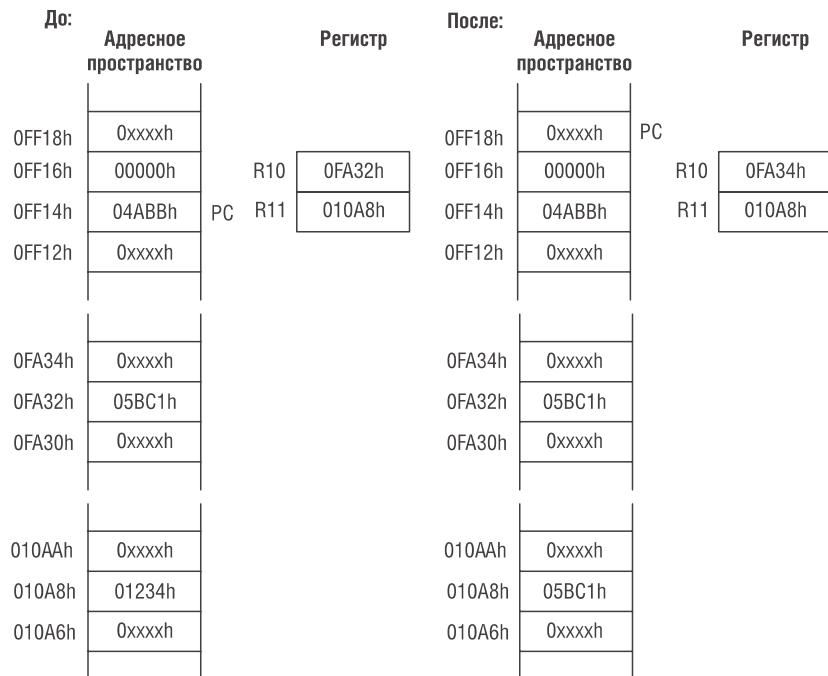


### 3.3.6. Косвенный автоинкрементный режим

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)
<b>Длина:</b> Одно или два слова	
<b>Операция:</b> 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.	
<b>Комментарий:</b> Действительно только для операнда источника. В качестве операнда получателя подставляется 0(Rd) плюс вторая команда INCD Rd.	
<b>Пример:</b> MOV.B @R10+, 0 (R11)	



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

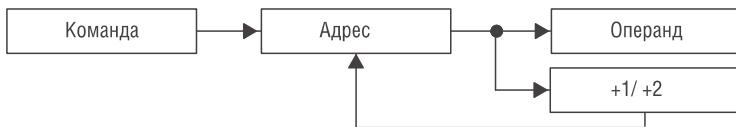


Рис. 3-8. Операция выборки операнда

### 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) 45h=TONI-PC
<b>Длина:</b> Два или три слова. На одно слово меньше, если может использоваться константа генераторов CG1 или CG2.	
<b>Операция:</b> 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.	
<b>Комментарий:</b> Действительно только для операнда источника	
<b>Пример:</b> MOV #45h, TONI	

До:

Адресное пространство		Регистр PC	Адресное пространство		Регистр PC
0FF16h	01192h		0FF18h	0xxxxh	
0FF14h	00045h		0FF16h	01192h	
0FF12h	040B0h		0FF14h	00045h	
			0FF12h	040B0h	
010AAh	0xxxxh	OFF16h	010AAh	0xxxxh	
010A8h	01234h	+01192h	010A8h	00045h	
010A6h	0xxxxh	010A8h	010A6h	0xxxxh	

## 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.

**3.4.1. Команды с двойным операндом (Формат I)**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код операции		Регистр-источник S-Reg		Ad	B/W	As	Регистр-получатель D-Reg								

**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
<b>MOV (.B)</b>	src,dst	$src \rightarrow dst$	-	-	-	-
<b>ADD (.B)</b>	src,dst	$src + dst \rightarrow dst$	*	*	*	*
<b>ADDC (.B)</b>	src,dst	$src + dst + C \rightarrow dst$	*	*	*	*
<b>SUB (.B)</b>	src,dst	$dst + .not.src + 1 \rightarrow dst$	*	*	*	*
<b>SUBC (.B)</b>	src,dst	$dst + .not.src + C \rightarrow dst$	*	*	*	*
<b>CMP (.B)</b>	src,dst	$dst - src$	*	*	*	*
<b>DADD (.B)</b>	src,dst	$src + dst + C \rightarrow dst$ (десятичное)	*	*	*	*
<b>BIT (.B)</b>	src,dst	$src .and. dst$	0	*	*	*
<b>BIC (.B)</b>	src,dst	$.not.src .and. dst \rightarrow dst$	-	-	-	-
<b>BIS (.B)</b>	src,dst	$src .or. dst \rightarrow dst$	-	-	-	-
<b>XOR (.B)</b>	src,dst	$src .xor. dst \rightarrow dst$	*	*	*	*
<b>AND (.B)</b>	src,dst	$src .and. dst \rightarrow dst$	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.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код операции								B/W	Ad	Регистр-получатель D-Reg					

**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
<b>RRC (.B)</b>	dst	C → MSB → ... LSB → C	*	*	*	*
<b>RRA (.A)</b>	dst	MSB → MSB → ... LSB → C	0	*	*	*
<b>PUSH (.B)</b>	src	SP-2 → SP, src → @SP	-	-	-	-
<b>SWPB</b>	dst	Обмен байтами	-	-	-	-
<b>CALL</b>	dst	SP-2 → SP, PC+2 → @SP dst → PC	-	-	-	-
<b>RETI</b>		TOS → SR, SP+2 → SP TOS → PC, SP+2 → SP	*	*	*	*
<b>SXT</b>	dst	Бит7 → Бит8 ... ... Бит15	0	*	*	*
*	Влияет на бит статуса					
-	Не влияет на бит статуса					
<b>0</b>	Бит статуса очищается					
<b>1</b>	Бит статуса устанавливается					

Для команды CALL возможны все способы адресации. Если используется символьический 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.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код операции				С				10-разрядное смещение PC							

**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
<b>JEQ/JZ</b>	Метка	Переход к метке, если бит нуля (Z) установлен
<b>JNE/JNZ</b>	Метка	Переход к метке, если бит нуля (Z) сброшен
<b>JC</b>	Метка	Переход к метке, если бит переноса (C) установлен
<b>JNC</b>	Метка	Переход к метке, если бит переноса (C) сброшен
<b>JN</b>	Метка	Переход к метке, если бит отрицательного результата (N) установлен
<b>JGE</b>	Метка	Переход к метке, если (N.XOR.V)=0
<b>JL</b>	Метка	Переход к метке, если (N.XOR.V)=1
<b>JMP</b>	Метка	Безусловный переход к метке

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

$$PC_{new} = PC_{old} + 2 + PC_{offset} \times 2$$

где:  $PC_{new}$  – новое содержимое программного счетчика;

$PC_{old}$  – исходное содержимое программного счетчика;

$PC_{offset}$  – 10-разрядная величина смещения программного счетчика.

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

<b>Биты статуса</b>	C:	Устанавливается, если содержимое получателя dst инкрементируется от OFFFh к 0000, в противном случае сбрасывается;
	V:	Устанавливается, если произошло арифметическое переполнение, в противном случае сбрасывается
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	
<b>Пример</b>	<p>Сложение содержимого 16-разрядного счетчика, указанного в R13, с 32-разрядным счетчиком, указанным в R12:</p> <pre>ADD @R13, 0 (R12)      ; сложение LSD ADC 2 (R12)            ; сложение переноса с MSD</pre>	
<b>Пример</b>	<p>Сложение содержимого 8-разрядного счетчика, указанного в R13, с 16-разрядным счетчиком, указанным в R12:</p> <pre>ADD.B @R13, 0 (R12)    ; сложение LSD ADC.B 1 (R12)          ; сложение переноса с MSD</pre>	
<b>ADD [ .W ]</b>	<b>Сложение содержимого источника с содержимым получателя</b>	
<b>ADD .B</b>	<b>Сложение содержимого источника с содержимым получателя</b>	
<b>Синтаксис</b>	<pre>ADD src,dst или ADD.W src,dst ADD.B src,dst</pre>	
<b>Операция</b>	$src + dst \rightarrow dst$	
<b>Описание</b>	Операнд источника складывается с операндом получателя. Операнд источника не изменяется. Предыдущее содержимое получателя теряется	
<b>Биты статуса</b>	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
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	
<b>Пример</b>	<p>Прибавление 10 к содержимому R5. Выполняется переход к метке TONI, если произошел перенос (установлен бит C):</p> <pre>ADD #10, R5 JC TONI      ; произошел перенос ...           ; переноса нет</pre>	
<b>Пример</b>	<p>Прибавление 10 к содержимому R5. Выполняется переход к метке TONI, если произошел перенос (установлен бит C):</p> <pre>ADD.B #10, R5 ; прибавление 10 к младшему байту R5 JC TONI       ; перенос произошел, если (R5) ≥ 246                 ; [0Ah+0F6h] ...           ; переноса нет</pre>	

<b>ADDC [ .W ]</b>	<b>Сложение содержимого источника и переноса с содержимым получателя</b>				
<b>ADDC.B</b>	<b>Сложение содержимого источника и переноса с содержимым получателя</b>				
<b>Синтаксис</b>	ADDC src,dst или ADDC.W src,dst ADDC.B src,dst				
<b>Операция</b>	$src + dst + C \rightarrow dst$				
<b>Описание</b>	Операнд источника и бит переноса (C) складываются с операндом получателя. Операнд источника не изменяется. Предыдущее содержимое получателя теряется				
<b>Биты статуса</b>	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			
<b>Биты режима</b>	Биты OSOFF, CPUOFF и GIE не изменяются				
<b>Пример</b>	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				
<b>Пример</b>	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				

<b>AND [ .W ]</b>	<b>Логическое «И» источника и получателя</b>		
<b>AND.B</b>	<b>Логическое «И» источника и получателя</b>		
<b>Синтаксис</b>	AND src,dst или AND.W src,dst AND.B src,dst		
<b>Операция</b>	$src .AND. dst \rightarrow dst$		
<b>Описание</b>	Над операндом источника и операндом получателя выполняется операция логического «И» (логическое умножение). Результат остается в получателе.		

<b>Биты статуса</b>	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
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	
<b>Пример</b>	<p>Установка битов в R5 с использованием маски (#0AA55h) для слова, адресованного меткой TOM. Если результат «0», выполняется переход к метке TONI:</p> <pre>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</pre>	
<b>Пример</b>	<p>Логическое перемножение битов маски #0A5h с младшим байтом TOM. Если результат «0», выполняется переход к метке TONI:</p> <pre>AND.B #0A5, TOM      ;маскирование младшего байта маской                       ;#0A5h JZ   TONI            ; ...                  ;результат не «0»</pre>	

<b>BIC[.W]</b>	<b>Очистка битов получателя</b>
<b>BIC.B</b>	<b>Очистка битов получателя</b>
<b>Синтаксис</b>	BIC src,dst или BIC.W src,dst BIC.B src,dst
<b>Операция</b>	.NOT.src .AND. dst → dst
<b>Описание</b>	Инвертированный операнд источника и операнд получателя логически перемножаются. Результат помещается в получатель. Операнд источника не изменяется.
<b>Биты статуса</b>	Биты статуса не изменяются
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются
<b>Пример</b>	Очистка шести битов MSB слова LEO в ОЗУ: BIC #0FC00h, LEO ;очистка 6-ти битов MSB в слове ;памяти (LEO)
<b>Пример</b>	Очистка пяти битов MSB байта LEO в ОЗУ: BIC.B #0F8h, LEO ;очистка 5-ти битов MSB в ОЗУ ;в байте LEO

<b>BIS [.W]</b>	<b>Установка битов получателя</b>
<b>BIS.B</b>	<b>Установка битов получателя</b>
<b>Синтаксис</b>	BIS src,dst или BIS.W src,dst BISC.B src,dst
<b>Операция</b>	src .OR. dst → dst
<b>Описание</b>	Над операндом источника и операндом получателя выполняется операция логического «ИЛИ» (логическое сложение). Результат помещается в получатель. Операнд источника не изменяется.
<b>Биты статуса</b>	Биты статуса не изменяются
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются
<b>Пример</b>	Установка шести LSB-битов слова ТОМ в ОЗУ: BIS #003Fh, ТОМ ;установка 6-ти LSB-битов слова ТОМ ;в ОЗУ
<b>Пример</b>	Установка трех MSB-битов байта LEO в ОЗУ: BIS.B #0E0h, ТОМ ;установка 3-х MSB-битов в байте ;ТОМ в ОЗУ

<b>BIT [.W]</b>	<b>Проверка битов получателя</b>	
<b>BIT.B</b>	<b>Проверка битов получателя</b>	
<b>Синтаксис</b>	BIT src,dst или BIT.W src,dst	
<b>Операция</b>	src .AND. dst	
<b>Описание</b>	Над операндом источника и операндом получателя выполняется операция логического «И» (логическое умножение). Результат влияет только на биты статуса. Операнды источника и получателя не изменяются.	
<b>Биты статуса</b>	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
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	
<b>Пример</b>	Если бит 9 регистра R8 установлен, выполняется переход к метке ТОМ: BIT #0200h, R8 ;бит 9 регистра R8 установлен? JNZ ТОМ ;да, переход к метке ТОМ ... ;Нет, продолжение программы	
<b>Пример</b>	Если бит 3 регистра R8 установлен, выполняется переход к метке ТОМ: BIT #8, R8JC ТОМ	

<b>Пример</b>	<p>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:</p> <pre> ; ;Последовательная передача данных, начиная с младшего ; бита (LSB), сдвиг которого происходит в первую оче- ; редь: ; ;xxxx xxxx xxxx xxxx BIT.B #RCV, RCCTL ;Информационный бит в бите переноса RRC RECBUF ;Бит переноса → в MSB регистра ;RECBUF ; ;xxxx 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 xxxx ... ;повтор двух предыдущих команд ... ;8 раз ; ;cccc cccc ; ;LSB ; ;MSB </pre>
---------------	--

<b>*BR, BRANCH</b>	<b>Переход к ... месту назначения</b>
<b>Синтаксис</b>	BR dst
<b>Операция</b>	dst → PC
<b>Эмуляция</b>	MOV dst, PC
<b>Описание</b>	Безусловный переход выполняется в любое место 64 кБайт адресного пространства. Могут использоваться все способы адресации. Команда перехода – это команда-слово.
<b>Биты статуса</b>	Биты статуса не изменяются
<b>Пример</b>	<p>Примеры для всех режимов адресации:</p> <pre> 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 </pre>

<b>Пример</b>	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 в регистре 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 следующий 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

<b>CALL</b>	<b>Вызов подпрограммы</b>
<b>Синтаксис</b>	CALL dst
<b>Операция</b>	dst → tmp dst оценивается и сохраняется
	SP - 2 → SP
	PC → @SP PC сохраняется на вершине стека (TOS)
	tmp → PC dst записывается в PC
<b>Описание</b>	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.
<b>Биты статуса</b>	Биты статуса не изменяются
<b>Пример</b>	Примеры для всех режимов адресации: CALL #EXEC ;Вызов с меткой EXEC или прямая адресация ; (например, #0A4h) ;SP-2 → SP, PC+2 → @SP, @PC+ → PC

<b>Пример</b>	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, ;указанный 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

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

<b>*CLRC</b>	<b>Очистка бита переноса</b>
<b>Синтаксис</b>	CLRC
<b>Операция</b>	$0 \rightarrow C$
<b>Эмуляция</b>	BIC #1, SR
<b>Описание</b>	Бит переноса (C) очищается. Команда очистки переноса – это команда-слово.
<b>Биты статуса</b>	N: Не изменяется
	Z: Не изменяется
	C: Очищается
	V: Не изменяется
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются
<b>Пример</b>	<p>Прибавление содержимого 16-разрядного десятичного счетчика, указанного в R13 к содержимому 32-разрядного счетчика, указанного в R12:</p> <p>CLRC ;C=0: определение исходного состояния</p> <p>DADD @R13, 0(R12) ;сложение 16-разрядного счетчика с младшим словом 32-разрядного счетчика</p> <p>DADC 2(R12) ;прибавление переноса к старшему слову 32-разрядного счетчика</p>

<b>*CLRN</b>	<b>Очистка бита отрицания</b>
<b>Синтаксис</b>	CLRN
<b>Операция</b>	$0 \rightarrow N$ или (.NOT.src .AND. dst $\rightarrow$ dst)
<b>Эмуляция</b>	BIC #4, SR
<b>Описание</b>	Константа 04h инвертируется (0FFF8h) и логически умножается (AND) с операндом получателя. Результат помещается в получатель. Команда очистки бита отрицания – это команда-слово.
<b>Биты статуса</b>	N: Сбрасывается в «0»
	Z: Не изменяется
	C: Не изменяется
	V: Не изменяется
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются
<b>Пример</b>	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

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

<b>CMP [ .W ]</b>	<b>Сравнение источника и получателя</b>				
<b>CMP .B</b>	<b>Сравнение источника и получателя</b>				
<b>Синтаксис</b>	CMP src,dst или CMP.W src,dst CMP.B src, dst				
<b>Операция</b>	dst + .NOT.src + 1 или (dst - src)				
<b>Описание</b>	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.				
<b>Биты статуса</b>	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный (src $\geq$ 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			
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются				
<b>Пример</b>	Сравнивается содержимое R5 и R6. Если оно одинаково, выполнение программы продолжается с меткой EQUAL. CMP R5,R6 ;R5=R6? JEQ EQUAL ;да, переход к метке EQUAL				
<b>Пример</b>	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				

<b>Пример</b>	L\$1 MOV #BLOCK2, R7 ; начальный адрес BLOCK2 CMP @R6+, 0 (R7) ; в регистр R7 JNZ ERROR ; сравнение содержимого ячеек и инкремент R6 ; если не равны, переход к метке ERROR INCD R7 ; если равны инкремент R7 DEC R5 ; декрементировать R5 JNZ L\$1 ; если сравнение не завершено – продолжить
<b>Пример</b>	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
<b>*DADC [ .W ]</b>	<b>Десятичное сложение переноса с получателем</b>
<b>*DADC .B</b>	<b>Десятичное сложение переноса с получателем</b>
<b>Синтаксис</b>	DADC dst или DADC.W src,dst DADC.B dst
<b>Операция</b>	dst + C → dst (десятичное)
<b>Эмуляция</b>	DADD #0, dst DADD.B #0, dst
<b>Описание</b>	Бит переноса (C) десятично прибавляется к получателю
<b>Биты статуса</b>	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
<b>Биты режима</b>	Биты OSOFF, CPUOFF и GIE не изменяются
<b>Пример</b>	Десятичное число из четырех цифр, содержащееся в регистре R5 прибавляется к десятичному числу из восьми цифр, указанному в регистре R8. CLRC ;брос переноса ;стартовое условие для следующих ;команд задано DADD R5, 0 (R8) ;сложение LCDs и переноса DADC 2 (R8) ;прибавление переноса к MSD
<b>Пример</b>	Десятичное число из двух цифр, содержащееся в регистре R5 прибавляется к десятичному числу из четырех цифр, указанному в регистре R8. CLRC ;брос переноса ;стартовое условие для следующих ;команд задано DADD.B R5, 0 (R8) ;сложение LCDs и переноса DADC 1 (R8) ;прибавление переноса к MSD

<b>DADD [ .W ]</b>	<b>Десятичное сложение источника, переноса и получателя</b>	
<b>DADD .B</b>	<b>Десятичное сложение источника, переноса и получателя</b>	
<b>Синтаксис</b>	DADD src,dst или DADD.W src,dst DADD.B src, dst	
<b>Операция</b>	src + dst + C → dst (десятичное)	
<b>Описание</b>	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.	
<b>Биты статуса</b>	N:	Устанавливается, если MSB равен «1»; в противном случае сбрасывается
	Z:	Устанавливается, если результат равен «0»; в противном случае сбрасывается
	C:	Устанавливается, если результат превышает 9999 Устанавливается, если результат превышает 99
	V:	Не определено
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	
<b>Пример</b>	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	
<b>Пример</b>	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	

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

<b>Описание</b>	Операнд получателя уменьшается (декрементируется) на единицу. Исходное содержимое теряется.
<b>Биты статуса</b>	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
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются
<b>Пример</b>	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.	
<pre> MOV    #EDE, R6 MOV    #255, R10 L\$1  MOV.B  @R6+, TONI-EDE-1 (R6) DEC    R10 JNZ    L\$1 </pre>	

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

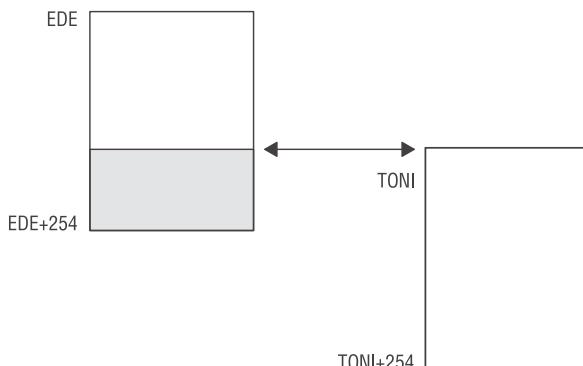


Рис. 3-12. Перекрытие (наложение) при декременте

<b>*DECD [ .W ]</b>	Двойной декремент получателя
<b>*DECD .B</b>	Двойной декремент получателя
<b>Синтаксис</b>	DECD dst или DECD.W dst DECD.B dst
<b>Операция</b>	dst - 2 → dst
<b>Эмуляция</b>	SUB #2, dst SUB.B #2, dst
<b>Описание</b>	Операнд получателя уменьшается (декрементируется) на два. Исходное содержимое теряется.
<b>Биты статуса</b>	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
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются
<b>Пример</b>	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.	
L\$1	MOV #EDE, R6 MOV #510, R10 MOV @R6+, TONI-EDE-2 (R6) DEC D JNZ L\$1
<b>Пример</b>	Содержимое ячейки памяти LEO декрементируется на два. DECD.B LEO ; декремент MEM (LEO) Декремент байта статуса STATUS на два. DECD.B STATUS
<b>*DINT</b>	<b>Запрещение (общее) прерываний</b>
<b>Синтаксис</b>	DINT
<b>Операция</b>	0 → GIE или (0FFF7h .AND. SR → SR/.NOT.src .AND. dst → dst)
<b>Эмуляция</b>	BIC #8, SR

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

### 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.

<b>*EINT</b>	<b>Разрешение (общее) прерываний</b>
<b>Синтаксис</b>	EINT
<b>Операция</b>	$1 \rightarrow \text{GIE}$ или $(0008h .OR. \text{SR} \rightarrow \text{SR} / .src .OR. \text{dst} \rightarrow \text{dst})$
<b>Эмуляция</b>	BIS #8, SR
<b>Описание</b>	Все прерывания разрешаются. Константа #08h и регистр статуса SR логически складываются (OR). Результат помещается в регистр статуса SR.
<b>Биты статуса</b>	Биты статуса не изменяются
<b>Биты режима</b>	GIE устанавливается. Биты OSCOFF и CPUOFF не изменяются.
<b>Пример</b>	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 OFFh, в противном случае сбрасывается
	C:	Устанавливается, если dst содержал 0FFFFh, в противном случае сбрасывается; Устанавливается, если dst содержал OFFh, 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	

<b>*INCD [ .W ]</b>	<b>Двойной инкремент получателя</b>				
<b>*INCD .B</b>	<b>Двойной инкремент получателя</b>				
<b>Синтаксис</b>	INCD dst или INCD.W dst INCD.B dst				
<b>Операция</b>	dst + 2 → dst				
<b>Эмуляция</b>	ADD #2, dst				
<b>Эмуляция</b>	ADD.B #2, dst				
<b>Описание</b>	Операнд получателя инкрементируется на два. Исходное содержимое теряется.				
<b>Биты статуса</b>	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный			
	Z:	Устанавливается, если dst содержал 0FFFFh, в противном случае сбрасывается; Устанавливается, если dst содержал 0FEh, в противном случае сбрасывается			
	C:	Устанавливается, если dst содержал 0FFEh или 0FFFFh, в противном случае сбрасывается; Устанавливается, если dst содержал 0FEh или OFFh, в противном случае сбрасывается			
	V:	Устанавливается, если dst содержал 07FFEh или 07FFh, в противном случае сбрасывается; Устанавливается, если dst содержал 07Eh или 07Fh, в противном случае сбрасывается			
	<b>Биты режима</b>				
<b>Биты OSOFF, CPUOFF и GIE не изменяются</b>					
<b>Пример</b>	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.		
<b>Пример</b>	Байт на вершине стека инкрементируется на два. INCD.B 0 (SP) ;байт на вершине стека (TOS) ;инкрементируется на два				

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

<b>Описание</b>	Операнд получателя инвертируется. Исходное содержимое теряется.	
<b>Биты статуса</b>	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный
	Z:	Устанавливается, если dst содержал OFFFFh, в противном случае сбрасывается; Устанавливается, если dst содержал OFFh, 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
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	
<b>Пример</b>	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	
<b>Пример</b>	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	
<b>JC</b>	Переход, если перенос установлен	
<b>JHS</b>	Переход, если наивысший* или равный	
<b>Синтаксис</b>	JC label JHS label	
<b>Операция</b>	Если C=1: PC+2 × смещение → PC Если C=0: выполняется следующая команда	
<b>Описание</b>	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 С сброшен, выполняется команда, следующая за инструкцией jc напр. Команда JC (переход, если перенос / наивысший или равный) используется для сравнения чисел без знака (от 0 до 65536).	
<b>Биты статуса</b>	Биты статуса не изменяются	
<b>Пример</b>	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	

<b>Пример</b> <pre>CMP #15, R5 JHS LABEL ;Если R5≥15, происходит переход ... ;Продолжение с этого места, если R5&lt;15</pre>	<p>Содержимое R5 сравнивается с числом 15. Если содержимое наивысшее или такое же, происходит переход к метке LABEL.</p> <p>* 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 <i>больше</i> или <i>равно</i>), с помощью которой можно сравнивать числа со знаком.</p>
---	---

<b>JEQ, JZ</b> <b>Синтаксис</b> <b>Операция</b> <b>Описание</b> <b>Биты статуса</b> <b>Пример</b>	<p><b>Переход, если равно; переход, если ноль</b></p> <p>JEQ label, JZ label</p> <p>Если Z=1: PC+2 × смещение → PC Если Z=0: выполняется следующая команда</p> <p>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 <i>jmp</i>.</p> <p>Биты статуса не изменяются</p> <p>Переход по адресу, содержащемуся в TONI, если R7 содержит ноль. TST R7 ;Проверка содержимого регистра R7 JZ TONI ;Переход, если «ноль»</p> <p>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</p> <p>Переход к метке LABEL, если содержимое R5 равно нулю. TST R5 JZ LABEL ...</p>
--	--

<b>JGE</b> <b>Синтаксис</b> <b>Операция</b> <b>Описание</b> <b>Биты статуса</b>	<p><b>Переход, если больше или равно</b></p> <p>JGE label</p> <p>Если (N .XOR. V)=0, то переход к метке: PC+2 × смещение → PC Если (N .XOR. V)=1, то выполняется следующая команда</p> <p>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 <i>jmp</i>. Это позволяет сравнивать числа со знаком.</p> <p>Биты статуса не изменяются</p>
---	---

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

<b>JL</b>	<b>Переход, если меньше</b>
<b>Синтаксис</b>	<b>JL label</b>
<b>Операция</b>	<p>Если (N .XOR. V)=1, то переход к метке: PC+2 × смещение → PC</p> <p>Если (N .XOR. V)=0, то выполняется следующая команда</p>
<b>Описание</b>	<p>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 <code>jmp</code>. Это позволяет сравнивать числа со знаком.</p>
<b>Биты статуса</b>	<p>Биты статуса не изменяются</p>
<b>Пример</b>	<p>Если содержимое регистра R6 меньше содержимого памяти по адресу, указанному в R7, выполнение программы продолжается с меткой EDE.</p> <pre>CMP @R7, R6      ;R6&lt;(R7) ?, сравнение чисел со знаком JL EDE          ;Да, R6&lt;(R7); переход к метке EDE ... ... ;Нет, продолжение программы</pre>

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

<b>JN</b>	<b>Переход, если отрицание</b>
<b>Синтаксис</b>	<b>JN label</b>
<b>Операция</b>	<p>Если N=1: PC+2 × смещение → PC</p> <p>Если N=0: выполняется следующая команда</p>
<b>Описание</b>	<p>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 <code>jmp</code>.</p>

<b>Биты статуса</b>	Биты статуса не изменяются
<b>Пример</b>	<p>Результат вычисления в R5 вычитается из COUNT. Если получается отрицательная величина, COUNT очищается и выполнение программы продолжается по другому пути.</p> <pre> 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 ... </pre>
<b>JNC</b>	<b>Переход, если перенос не установлен</b>
<b>JLO</b>	<b>Переход, если низший</b>
<b>Синтаксис</b>	JNC label JLO label
<b>Операция</b>	<p>Если C=0: PC+2 × смещение → PC</p> <p>Если C=1: выполняется следующая команда</p>
<b>Описание</b>	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 <code>jmp</code> . Команда JNC (переход, если нет переноса / низший) используется для сравнения чисел без знака (от 0 до 65536).
<b>Биты статуса</b>	Биты статуса не изменяются
<b>Пример</b>	<p>Результат в R6 прибавляется к BUFFER. Если происходит переполнение, выполняется процедура обработки ошибки по адресу ERROR.</p> <pre> ADD R6,BUFFER ;BUFFER + R6 → BUFFER JNC CONT      ;Переход к CONT, если переноса нет ERROR ... ... </pre> <p>Начало процедуры обработки ошибки</p> <pre> COUNT ...      ;Продолжение нормального хода                 ;программы ...</pre>
<b>Пример</b>	<p>Переход к STL2, если байт STATUS содержит 1 или 0.</p> <pre> CMP.B #2,STATUS JLO STL2       ;STATUS&lt;2 ...                 ;STATUS≥2, продолжение здесь </pre>

<b>JNE</b>	Переход, если не равно
<b>JNZ</b>	Переход, если не ноль
<b>Синтаксис</b>	JNE label JNZ label
<b>Операция</b>	Если Z=0: PC+2 × смещение → PC Если Z=1: выполняется следующая команда
<b>Описание</b>	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 <code>jump</code> .
<b>Биты статуса</b>	Биты статуса не изменяются
<b>Пример</b>	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

<b>MOV [ .W ]</b>	Пересылка содержимого источника в получатель
<b>MOV .B</b>	Пересылка содержимого источника в получатель
<b>Синтаксис</b>	MOV src, dst или MOV.W src, dst MOV.B src, dst
<b>Операция</b>	src → dst
<b>Описание</b>	Операнд источника посыпается в получатель. Операнд источника не изменяется. Предыдущее содержимое получателя теряется.
<b>Биты статуса</b>	Биты статуса не изменяются
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются
<b>Пример</b>	Содержимое таблицы EDE (данные в виде слов) копируется в таблицу ТОМ. 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 ... ...
<b>Пример</b>	Содержимое таблицы EDE (данные в виде байт) копируется в таблицу ТОМ. 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	<pre>DEC R9 JNZ Loop ... ... ...</pre>	; в регистре R10 для ; обеих таблиц ; Декремент счетчика ; Содержимое ; счетчика ≠ 0, ; копирование ; продолжается ; Копирование закончено
--------	--	---

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

<b>*POP [ .W ]</b>	<b>Снятие со стека слова в получатель</b>
<b>*POP.B</b>	<b>Снятие со стека слова в получатель</b>
<b>Синтаксис</b>	POP dst POP.B dst
<b>Операция</b>	$\begin{aligned} @SP \rightarrow temp \\ SP + 2 \rightarrow SP \\ temp \rightarrow dst \end{aligned}$

<b>Эмуляция</b>	MOV @SP+, dst или MOV.W @SP+, dst MOV.B @SP+, dst
<b>Описание</b>	Содержимое стека, на которое указывает указатель стека (TOS) помещается в получатель. Затем указатель стека инкрементируется на два.
<b>Биты статуса</b>	Биты статуса не изменяются
<b>Пример</b>	Восстановление из стека содержимого регистра R7 и регистра статуса. POP R7 ; Восстановление R7 POP SR ; Восстановление регистра статуса
<b>Пример</b>	Восстановление из стека содержимого байта ОЗУ LEO. POP.B LEO ; Младший байт помещается из стека в LEO
<b>Пример</b>	Восстановление из стека содержимого регистра R7. POP.B R7 ; Младший байт помещается из стека в R7, ; старший байт регистра R7 равен 00h
<b>Пример</b>	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.

<b>PUSH [ .W ]</b>	Помещение слова в стек
<b>PUSH.B</b>	Помещение байта в стек
<b>Синтаксис</b>	PUSH src или PUSH.W src PUSH.B src
<b>Операция</b>	SP - 2 → SP src → @SP
<b>Описание</b>	Указатель стека декрементируется на два, затем операнд источника помещается в слово ОЗУ, адрес которого содержит указатель стека (TOS).
<b>Биты статуса</b>	Биты статуса не изменяются
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются
<b>Пример</b>	Содержимое регистра статуса и регистра R8 сохраняются в стеке. PUSH SR ; сохранение регистра статуса PUSH R8 ; сохранение регистра R8
<b>Пример</b>	Сохранение содержимого периферии TCDAT в стеке. PUSH.B &TCDAT ; сохранение в стеке данных ;из 8-разрядного периферийного ;модуля, адресованного TCDAT

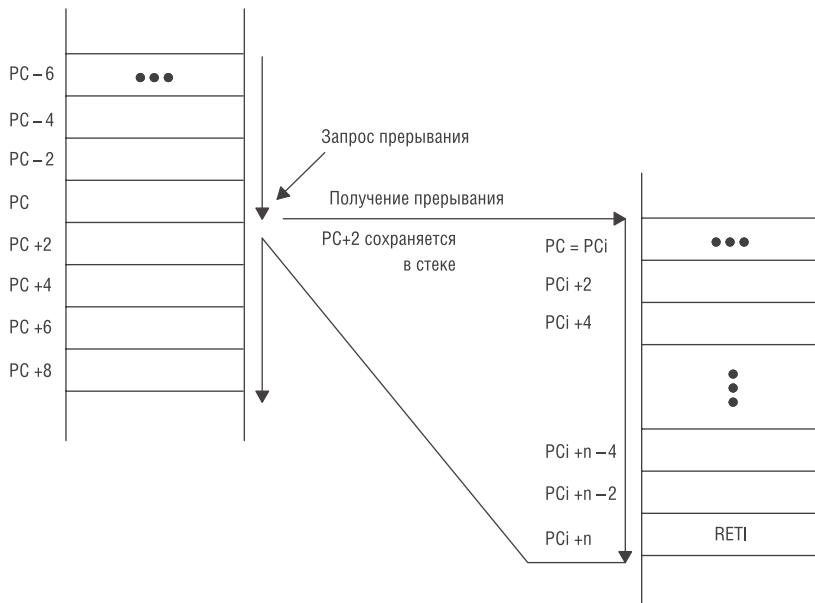
### Примечание: указатель системного стека

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

<b>*RET</b>	<b>Возврат из подпрограммы</b>
<b>Синтаксис</b>	RET
<b>Операция</b>	$\text{@SP} \rightarrow \text{PC}$ $\text{SP} + 2 \rightarrow \text{SP}$
<b>Эмуляция</b>	MOV @SP+, PC
<b>Описание</b>	Адрес возврата, помещенный в стек командой CALL посыпается в счетчик команд. Программа продолжается с адреса кода, следующего за адресом команды вызова подпрограммы.
<b>Биты статуса</b>	Биты статуса не изменяются

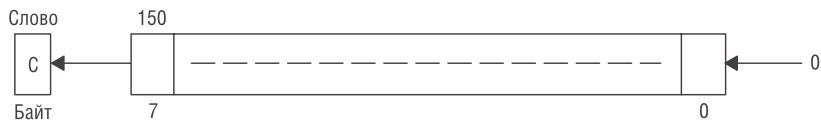
<b>RETI</b>	<b>Возврат из прерывания</b>
<b>Синтаксис</b>	RETI
<b>Операция</b>	$\text{TOS} \rightarrow \text{SR}$ , $\text{SP} + 2 \rightarrow \text{SP}$ , $\text{TOS} \rightarrow \text{PC}$ , $\text{SP} + 2 \rightarrow \text{SP}$
<b>Описание</b>	Регистр состояния восстанавливает свое исходное значение, 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.
<b>Биты статуса</b>	N: восстанавливается из системного стека Z: восстанавливается из системного стека C: восстанавливается из системного стека V: восстанавливается из системного стека
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE восстанавливаются из системного стека

<b>*RLA[ .W]</b>	<b>Арифметическая ротация влево</b>
<b>*RLA.B</b>	<b>Арифметическая ротация влево</b>
<b>Синтаксис</b>	RLA dst или RLA.W dst RLA.B dst
<b>Операция</b>	$C \leftarrow \text{MSB} \leftarrow \text{MSB}-1 \dots \text{LSB}+1 \leftarrow \text{LSB} \leftarrow 0$
<b>Эмуляция</b>	ADD dst, dst ADD.B dst, dst
<b>Описание</b>	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 $\text{dst} \geq 04000\text{h}$ и $\text{dst} < 0C000\text{h}$ перед выполнением операции: результат меняет знак.



**Рис. 3-13.** Прерывание главной программы

<b>Биты статуса</b>	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 $04000h \leq dst < 0C000h$ ; в противном случае сбрасывается
	V:	Устанавливается, если произошло арифметическое переполнение: исходное значение $040h \leq dst < 0C0h$ ; в противном случае сбрасывается
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	



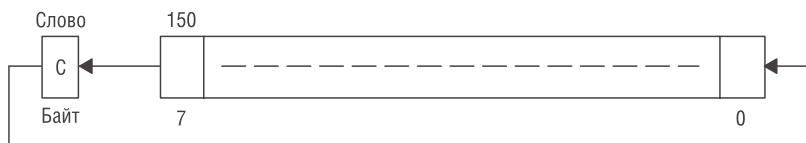
**Рис. 3-14.** Операнд получателя – арифметический сдвиг влево

<b>Пример</b>	Содержимое регистра R7 умножается на 2. RLA R7 ;Сдвиг влево R7 (умножение на 2)
<b>Пример</b>	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)*

<b>*RLC[ .W]</b>	<b>Арифметическая ротация влево через перенос</b>
<b>*RLC.B</b>	<b>Арифметическая ротация влево через перенос</b>
<b>Синтаксис</b>	RLC dst или RLC.W dst RLC.B dst
<b>Операция</b>	C $\leftarrow$ MSB $\leftarrow$ MSB-1 ... LSB+1 $\leftarrow$ LSB $\leftarrow$ C
<b>Эмуляция</b>	ADDC dst, dst
<b>Описание</b>	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).

<b>Биты статуса</b>	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 $04000h \leq dst < 0C000h$ ; в противном случае сбрасывается
		Устанавливается, если произошло арифметическое переполнение исходное значение $040h \leq dst < 0C0h$ ; в противном случае сбрасывается

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

<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются
<b>Пример</b>	Содержимое регистра R5 сдвигается влево на одну позицию. $RLC\ R5\ ;\ (R5 \times 2) + C \rightarrow R5$
<b>Пример</b>	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
<b>Пример</b>	Содержимое MEM (LEO) сдвигается влево на одну позицию. $RLC.B\ LEO\ ;$ Mem (LEO) $\times 2 + C \rightarrow$ Mem (LEO)

### Примечание: эмуляция RLC и RLC.B

Ассемблер не распознает команду:

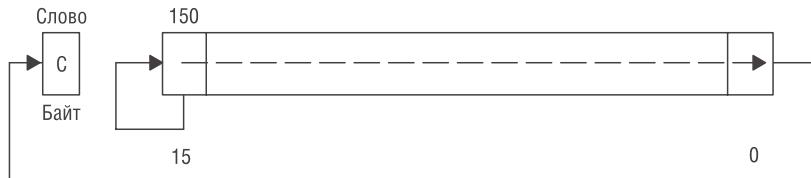
$RLC\ @R5+$

Вместо неё должна использоваться следующая команда:

$ADDC\ @R5+, -2\ (R5)$

<b>RRA [ .W ]</b>	<b>Арифметическая ротация вправо</b>
<b>RRA .B</b>	<b>Арифметическая ротация вправо</b>
<b>Синтаксис</b>	RRA dst или RRA.W dst RRA.B dst
<b>Операция</b>	MSB → MSB, MSB → MSB-1, ... LSB+1 → LSB, LSB → C
<b>Описание</b>	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.

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



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

<pre> ; 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 ... </pre>	
<b>Пример</b> <pre> 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 ... </pre>	

<b>RRC.[W]</b>	Ротация вправо через перенос
<b>RRC.B</b>	Ротация вправо через перенос
<b>Синтаксис</b>	RRC dst или RRC.W dst RRC.B dst
<b>Операция</b>	C → MSB → MSB-1 ... LSB+1 → LSB → C
<b>Описание</b>	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).

<b>Биты статуса</b>	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
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	

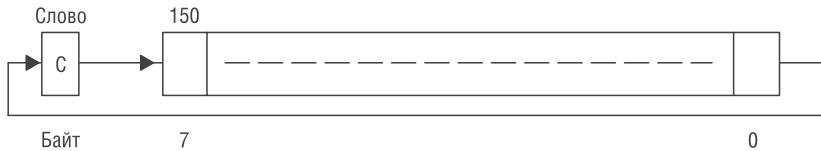


Рис. 3-17. Операнд получателя – сдвиг вправо через перенос

<b>Пример</b>	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
<b>Пример</b>	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

<b>*SBC [ .W ]</b>	<b>Вычитание заема/.NOT. переноса из получателя</b>				
<b>*SBC .B</b>	<b>Вычитание заема/.NOT. переноса из получателя</b>				
<b>Синтаксис</b>	SBC dst или SBC.W dst SBC.B dst				
<b>Операция</b>	dst + 0FFFFh + C → dst dst + 0FFh + C → dst				
<b>Эмуляция</b>	SUBC #0, dst SUBC.B #0, dst				
<b>Описание</b>	Бит переноса (C) прибавляется к операнду получателя минус один. Предыдущее содержимое получателя теряется.				
<b>Биты статуса</b>	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			
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются				
<b>Пример</b>	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				
<b>Пример</b>	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

<b>*SETC</b>	<b>Установка бита переноса</b>	
<b>Синтаксис</b>	SETC	
<b>Операция</b>	$1 \rightarrow C$	
<b>Эмуляция</b>	BIS #1, SR	
<b>Описание</b>	Устанавливается бит переноса (C).	
<b>Биты статуса</b>	N:	Не изменяется
	Z:	Не изменяется
	C:	Устанавливается
	V:	Не изменяется
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	
<b>Пример</b>	<p>Эмуляция десятичного вычитания: Десятичное вычитание R5 из R6. Принимается, что R5=3987 и R6=4137:</p> <pre>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</pre>	

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

<b>*SETZ</b>	<b>Установка бита нуля</b>	
<b>Синтаксис</b>	SETZ	
<b>Операция</b>	$1 \rightarrow Z$	
<b>Эмуляция</b>	BIS #2, SR	
<b>Описание</b>	Устанавливается бит нуля (Z).	

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

<b>SUB [ .W ]</b>	<b>Вычитание источника из получателя</b>	
<b>SUB .B</b>	<b>Вычитание источника из получателя</b>	
<b>Синтаксис</b>	SUB src, dst или SUB.W src,dst SUB.B src,dst	
<b>Операция</b>	dst + .NOT.src + 1 → dst или [(dst - src → dst)]	
<b>Описание</b>	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.	
<b>Биты статуса</b>	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
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	
<b>Пример</b>	См. пример для команды SBC	
<b>Пример</b>	См. пример для команды SBC .B	

### Примечание: обработка заема как .NOT.

Заем обрабатывается как операция .NOT. переноса:

Заем	Да	Нет
Бит переноса	0	1

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

<b>Описание</b>	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.	
<b>Биты статуса</b>	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
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	
<b>Пример</b>	<p>Вычитываются две мантиссы (24-разрядные) с плавающей точкой. Младшие байты LSB находятся в R13 и R10, старшие байты MSB находятся в R12 и R9.</p> <pre>SUB.W R13,R10 ;16-разрядная часть, LSB SUBC.B R12,R9 ;8-разрядная часть, MSB</pre>	
<b>Пример</b>	<p>Содержимое 16-разрядного счетчика, указанного в R13, вычитается из 16-разрядного счетчика, находящегося в регистрах R10 и R11(MSD).</p> <pre>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</pre>	

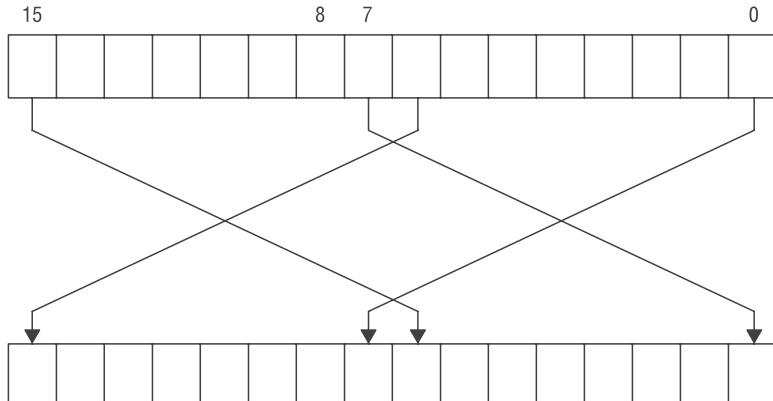
### Примечание: реализация заема

Заем обрабатывается как операция *NOT*. переноса:

Заем	Да	Нет
Бит переноса	0	1

<b>SWPB</b>	<b>Обмен байтами</b>	
<b>Синтаксис</b>	SWPB dst	
<b>Операция</b>	Биты с 15 по 8 « биты с 7 по 0	
<b>Описание</b>	Старший и младший байты операнда получателя меняются местами, как показано на рис. 3.18	
<b>Биты статуса</b>	Биты статуса не изменяются	
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	
<b>Пример</b>	<pre>MOV    #040BFh,R7          ;0100000010111111 → R7 SWPB  R7                  ;1011111010000000 в R7</pre>	

<b>Пример</b>	<p>Содержимое R5 умножается на 255. Результат сохраняется в R5, R4.</p> <pre> SWPB R5      ; MOV R5, R4    ;копирование значения после обмена               ;в R4 BIC #0FF00h, R5 ;корректировка результата BIC #00FFh, R4 ;корректировка результата </pre>
---------------	---



*Рис. 3-18. Обмен байтов в операнде получателя*

<b>SXT</b>	<b>Распространение знака</b>	
<b>Синтаксис</b>	SXT dst	
<b>Операция</b>	Бит 7 → в биты с 8 по 15	
<b>Описание</b>	Знак младшего байта распространяется в старшем байте, как показано на рис. 3.19.	
<b>Биты статуса</b>	Биты статуса не изменяются	
<b>Биты статуса</b>	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
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	
<b>Пример</b>	В R7 загружается значение P1IN. Команда распространения знака выполняет операцию развертывания значения бита 7 в биты с 8 по 15. <pre> MOV.B &amp;P1IN,R7 ;P1IN = 080h: .... .... 1000 0000 SXT R7          ;R7 = OFF80h: 1111 1111 1000 0000 </pre>	

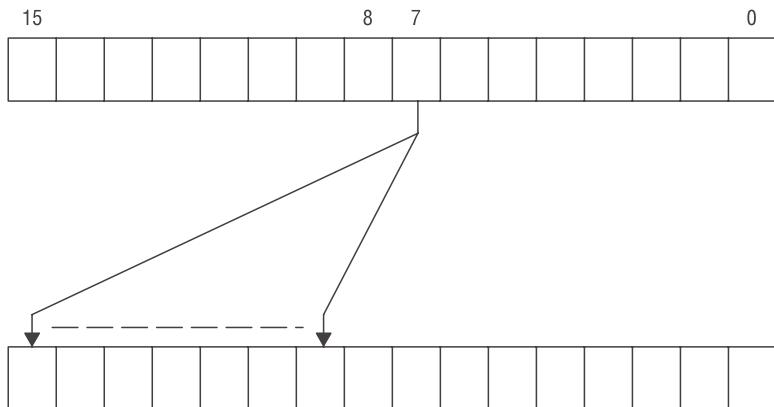


Рис. 3-19. Распространение знака операнда получателя

<b>*TST[.W]</b>	Проверка получателя				
<b>*TST.B</b>	Проверка получателя				
<b>Синтаксис</b>	TST dst или TST.W dst TST.B dst				
<b>Операция</b>	dst + 0FFFFh + 1 dst + 0FFh + 1				
<b>Эмуляция</b>	CMP #0, dst CMP.B #0, dst				
<b>Описание</b>	Операнд получателя сравнивается с нулем. Биты статуса устанавливаются в соответствии с результатом сравнения. Получатель не изменяется.				
<b>Биты статуса</b>	N:	Устанавливается, если результат отрицательный; сбрасывается, если положительный.			
	Z:	Устанавливается, если результат содержит «0», в противном случае сбрасывается.			
	C:	Устанавливается			
	V:	Сбрасывается			
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются				
<b>Пример</b>	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				

<b>Пример</b> <pre> 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 </pre>	Проверяется младший байт регистра R7. Если его содержимое отрицательно, программа продолжается с меткой R7NEG; если положительно, но не равно нулю, выполняется переход к метке R7POS.
---	--

<b>XOR [ .W ]</b>	<b>Исключающее «ИЛИ» источника и получателя</b>	
<b>XOR .B</b>	<b>Исключающее «ИЛИ» источника и получателя</b>	
<b>Синтаксис</b>	XOR src, dst или XOR.W src, dst XOR.B src, dst	
<b>Операция</b>	src .XOR. dst → dst	
<b>Описание</b>	Над operandами источника и получателя выполняется операция логического «ИЛИ» (OR). Результат помещается в получатель. Operand источника не изменяется.	
<b>Биты статуса</b>	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
<b>Биты режима</b>	Биты OSCOFF, CPUOFF и GIE не изменяются	
<b>Пример</b>	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	
<b>Пример</b>	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	
<b>Пример</b>	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																
1Cxх																
20xx							JNE/JNZ									
24xx							JEQ/JZ									
28xx							JNC									
2Cxx							JC									
30xx							JN									
34xx							JGE									
38xx							JL									
3Cxх							JMP									
4xxx						MOV, MOV.B										
5xxx						ADD, ADD.B										
6xxx						ADDC, ADC.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
<b>ADC (.B) *</b>	dst	Сложение бита C с получателем	dst + C → dst			*	*	*	*
<b>ADD (.B)</b>	src, dst	Сложение источника с получателем	src + dst → dst			*	*	*	*
<b>ADDC (.B)</b>	src, dst	Сложение источника и бита C с получателем	src + dst + C → dst			*	*	*	*
<b>AND (.B)</b>	src, dst	Операция «И» источника и получателя	src .and. dst → dst		0	*	*	*	*
<b>BIC (.B)</b>	src, dst	Очистка битов в получателе	.not.src .and. dst → dst		-	-	-	-	-
<b>BIS (.B)</b>	src, dst	Установка битов в получателе	src .or. dst → dst		-	-	-	-	-
<b>BIT (.B)</b>	src, dst	Проверка битов в получателе	src .and. dst		0	*	*	*	*
<b>BR*</b>	dst	Переход по назначению	dst → PC		-	-	-	-	-
<b>CALL</b>	dst	Вызов получателя	PC + 2 → stack, dst → PC		-	-	-	-	-
<b>CLR (.B) *</b>	dst	Очистка получателя	0 → dst		-	-	-	-	-
<b>CLRC*</b>		Очистка бита C	0 → C		-	-	-	0	
<b>CLRN*</b>		Очистка бита N	0 → N		-	0	-	-	
<b>CLRZ*</b>		Очистка бита Z	0 → Z		-	-	0	-	
<b>CMP (.B)</b>	src, dst	Сравнение источника и получателя	dst - src		*	*	*	*	*
<b>DADC (.B) *</b>	dst	Десятичное сложение бита C с получателем	dst + c → dst (десятичное)		*	*	*	*	*
<b>DADD (.B)</b>	src, dst	Десятичное сложение источника и бита C с получателем	src + dst + C → dst (десятичное)		*	*	*	*	*
<b>DEC (.B) *</b>	dst	Декремент получателя	dst - 1 → dst		*	*	*	*	*

Таблица 3.17. (Продолжение)

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

**Таблица 3.17. (Окончание)**

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

\*Эмулированные команды

**MSP430x4xxFamily**

## **Модуль тактирования FLL+**

*Раздел IV.*



## Модуль тактирования FLL+

Модуль тактирования FLL+ формирует тактовую частоту для микроконтроллеров серии MSP430x4xx. В этой главе описана работа модуля FLL+. Модуль тактирования FLL+ присутствует во всех микроконтроллерах семейства MSP430x4xx.

### 4.1. Введение в Модуль тактирования FLL+

Модуль тактирования с автоподстройкой частоты (FLL+) разработан с учётом требований сверхнизкого энергопотребления и снижения стоимости системы в целом. Имея выбор из трёх внутренних источников тактирования, пользователь может найти оптимальный баланс между производительностью и низким потреблением энергии. Основой модуля FLL+ является устройство цифровой автоподстройки частоты (FLL). Функционируя совместно с цифровым модулятором, FLL стабилизирует частоту встроенного генератора с цифровым управлением (DCO). Установившееся значение частоты генератора DCO программируется кратным значению частоты низкочастотного кварцевого генератора LFXT1. Модуль тактирования FLL+ может быть сконфигурирован для работы как без внешних элементов, так и с одним либо с двумя внешними кварцевыми или керамическими резонаторами. Все настройки и конфигурации осуществляются программно. Модуль FLL+ может тактироваться двумя или тремя источниками сигнала:

- LFXT1CLK: Низко- либо высокочастотный генератор может использоваться как с низкочастотным кварцевым генератором 32.768 кГц, так и со стандартным резонатором в диапазоне от 450 кГц до 8 МГц
- XT2CLK: Дополнительный источник тактирования, по желанию может быть использован со стандартными кварцевыми или керамическими резонаторами или внешними источниками сигнала частотой от 450 кГц до 8 МГц.
- DCOCLK: встроенный генератор с цифровым управлением (DCO) с характеристиками RC-типа, стабилизируемый модулем FLL.

Модуль тактирования FLL+ может являться источником четырёх тактовых сигналов:

- ACLK: Вспомогательная частота. Частота ACLK задаётся генератором LFXT1CLK и может быть выбрана в качестве источника тактирования для различных встроенных периферийных модулей.
- ACLK/n: Буферизованный выход частоты ACLK. Частота ACLK/n представляет собой частоту ACLK, делённую на 1,2,4 или 8 и может использоваться только внешними устройствами.

- MCLK: Основной источник тактирования. В качестве источника для MCLK могут служить генераторы LFXT1CLK, XT2CLK (если присутствует), или DCOCLK. Частота MCLK может быть разделена на 1, 2, 4, или 8 внутри модуля FLL. MCLK используется ядром CPU и системными устройствами.
- SMCLK: Дополнительная тактовая частота. В качестве источника для SMCLK программно можно выбрать XT2CLK (если присутствует), или DCOCLK. SMCLK может служить источником тактирования различных встроенных периферийных модулей.

Блок схема модуля тактирования FLL+ показана на рис. 4-1 для устройств семейства MSP430x44x и MSP430x43x.

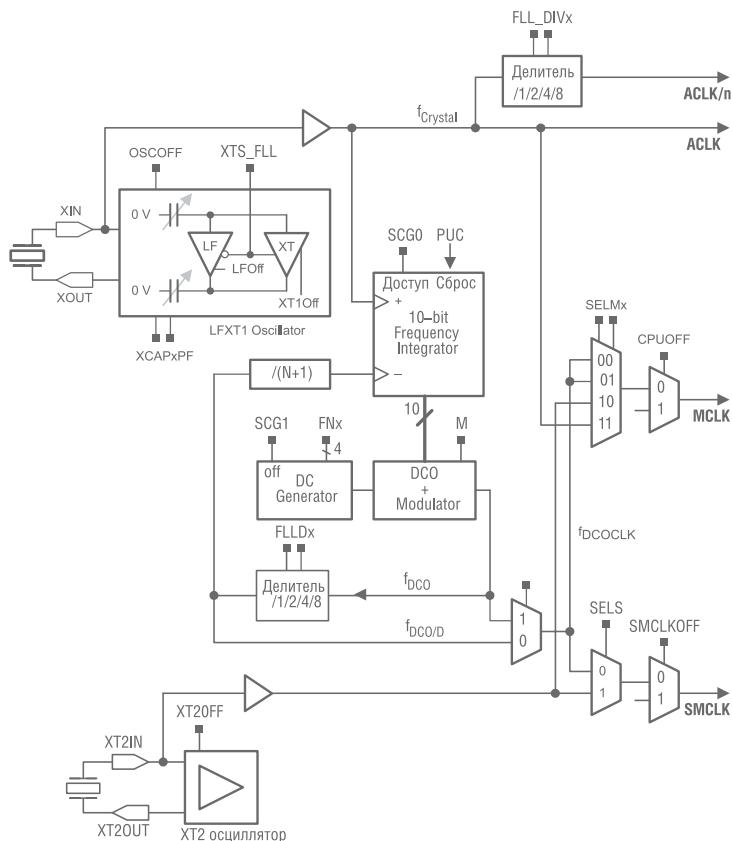
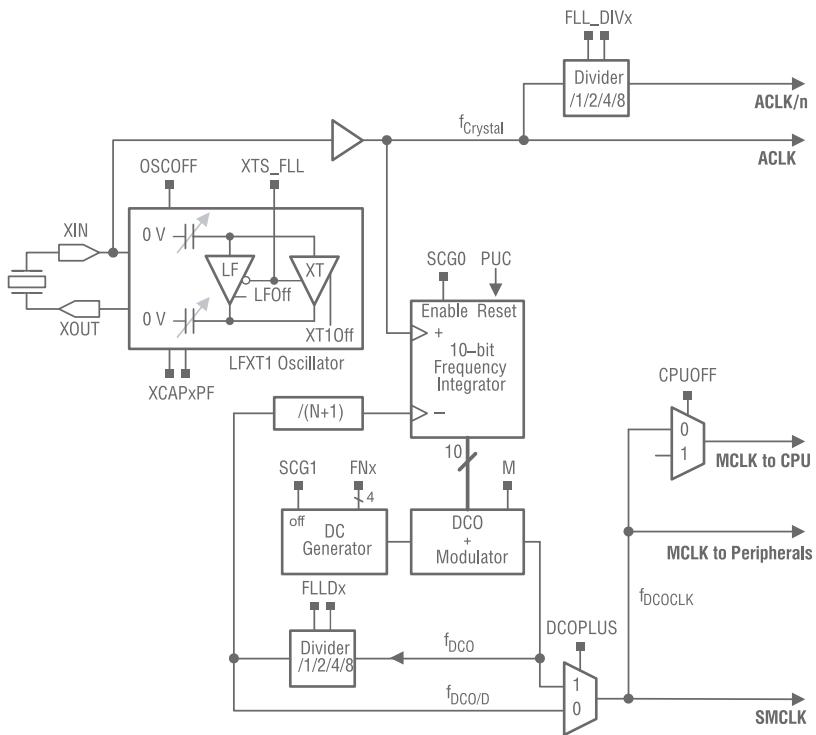


Рис. 4-1. Блок схема модуля тактирования FLL+ устройств семейства MSP430x44x и MSP430x43x

Для устройств семейства MSP430x42x и MSP430x41x блок-схема модуля тактирования FLL+ показана на рис. 4-2.



**Рис. 4-2.** Блок схема модуля тактирования FLL+ устройств семейства MSP430x42x и MSP430x41x

## 4.2. Работа модуля тактирования FLL+

После сброса по включению питания PUC, для частот MCLK и SMCLK источником является генератор DCOCLK с частотой, равной 32\*частоту генератора ACLK. При использовании в качестве тактирующего для ACLK кварца 32768 Гц, MCLK и SMCLK будут тактироваться стабильной частотой 1.048576 МГц.

Биты контроля SCG0, SCG1, OSCOFF, и CPUOFF в статусном регистре SR конфигурируют режимы работы микроконтроллера MSP430 а также разрешают либо запрещают работу отдельных блоков модуля тактирования FLL+. См. раздел «Системный сброс, прерывания и рабочие режимы» (System Resets,

Interrupts and Operating Modes). Регистры SCFQCTL, SCFI0, SCFI1, FLL\_CTL0, и FLL\_CTL1 конфигурируют модуль тактирования FLL+. Конфигурация либо ре-конфигурация модуля может быть осуществлена программно в любой момент в процессе выполнения программы.

### Пример конфигурации: MCLK = 64 x ACLK = 2097152

<i>BIC</i>	<i>#GIE,SR</i>	<i>;Запрет прерываний</i>
<i>MOV.B</i>	<i>(64-1),&amp;SCFQTL</i>	<i>;MCLK = 64 * ACLK,</i>
		<i>;DCOPLUS=0</i>
<i>MOV.B</i>	<i>#FN_2,&amp;SCFIO</i>	<i>; Выбор диапазона DCO</i>
<i>BIS</i>	<i>#GIE,SR</i>	<i>;Разрешение прерываний</i>

#### 4.2.1. Особенности работы модуля тактирования FLL+ в системах с низким энергопотреблением

В устройствах на базе MSP430x4xx, использующих батарейное питание к системе тактирования зачастую предъявляются противоречивые требования:

- Низкая тактовая частота для сбережения энергии и часовых функций
- Высокая тактовая частота для быстрой реакции на события и возможности быстрой обработки результатов
- Стабильность тактовой частоты во всём температурном диапазоне и диапазоне питающих напряжений

Модуль тактирования FLL+ удовлетворяет всем вышеуказанным требованиям, предоставляя пользователю выбор из трёх источников тактирования: ACLK, MCLK, и SMCLK. Для оптимальной работы в условиях низкого энергопотребления ACLK обычно конфигурируется для работы с низкопотребляющим кварцевым резонатором 32786 Гц, обеспечивающим стабильную частоту и одновременно низкое потребление в ждущем режиме. В качестве источника частоты MCLK может быть использован встроенный генератор DCO, стабилизируемый модулем FLL и активизируемый по необходимости при возникновении прерываний.

Цифровая петля стабилизации частоты обеспечивает уменьшенные времена запуска и пауз для стабилизации частоты по сравнению с аналоговыми системами фазовой автоподстройки частоты. Устройства с ФАПЧ требуют от сотен до тысяч тактовых импульсов для старта и стабилизации. Модуль FLL запускается немедленно на заранее установленной частоте.

#### 4.2.2. Генератор LFXT1

Генератор LFXT1 обеспечивает сверхнизкое энергопотребление при использовании «часового» кварцевого резонатора частотой 32768 Гц и режима LF (XTS\_FLL = 0). Часовой кварц подключается к выводам XIN и XOUT без дополнительных элементов.

Также генератор LFXT1 может быть использован совместно с высокочастотными кварцевыми или керамическими резонаторами в режиме HF ( $XTS\_FLL = 1$ ). Высокочастотные резонаторы подключаются к выводам XIN XOUT.

Кроме этого, для работы LFXT1 может использоваться внешний сигнал тактирования, который подаётся на вывод XIN (при этом необходима установка  $XTS\_FLL = 1$ ). Диапазон входных частот – ~1 Гц – 8 МГц. Когда входная частота меньше 450 кГц, бит XT1OF может быть установлен, запрещая ядру (CPU) тактирование от внешнего источника.

Программно конфигурируемые биты XCAPxPF управляют встроеннымми нагрузочными конденсаторами для кварцевого резонатора LFXT1. Выбираемые ёмкости могут иметь значение 1, 6, 8, или 10 пФ. При необходимости могут быть добавлены внешние конденсаторы.

Генератор LFXT1 может быть программно отключен установкой бита OSCOFF, если сигнал с генератора не используется в качестве источника для MCLK ( $SELM <> 3$  или  $CPUOFF = 1$ ).

### Примечание: характеристики генератора LFXT1

*Низкочастотные кварцевые резонаторы, как правило, требуют сотен миллисекунд для старта, это время зависит от типа кварца. Рекомендуется оставлять генератор LFXT1 включенным в режиме LF.*

*Генераторы со сверхнизким потреблением, такие, как LFXT1 в режиме LF должны быть защищены от внешних помех. Сам кварцевый резонатор должен быть расположен как можно ближе к MSP430, его корпус заземлён, а дорожки, идущие от кварца к МК защищены «земляными» цепями.*

*Значение по умолчанию для XCAPxPF это 0, что соответствует нагрузочному конденсатору с ёмкостью ~1 пФ. Кварцевый генератор может не работать с таким значением ёмкости до тех пор, пока соответствующим выбором бит XCAPxPF или внешних конденсаторов не будет выбрана необходимая нагрузочная ёмкость в соответствии с типом резонатора.*

### 4.2.3. Генератор XT2

В некоторых устройствах семейства присутствует второй кварцевый генератор, XT2. XT2 является источником частоты для XT2CLK, его характеристики идентичны характеристикам генератора LFXT1 в режиме HF, за исключением того, что генератор XT2 не имеет встроенных нагрузочных конденсаторов. Требуемые для работы высокочастотного кварцевого либо керамического резонатора нагрузочные конденсаторы должны быть установлены извне.

Бит XT2OFF запрещает работу генератора XT2 если частота XT2CLK не используется в качестве источника для MCLK ( $SELM <> 2$  или  $CPUOFF = 1$ ) и SMCLK ( $SELs = 0$  или  $SMCLKOFF = 1$ ).

Генератор XT2 может использовать внешнюю тактовую частоту, подаваемую на вход XT2IN. Диапазон внешних частот должен удовлетворять требованиям руководства пользователя в части параметров для XT2.

#### 4.2.4. Генератор с цифровым управлением (DCO)

Генератор с цифровым управлением DCO – встроенное устройство с характеристиками RC-типа. Частота генерации DCO стабилизируется модулем FLL, значение частоты при этом равно значению частоты ACLK, умноженной на коэффициент N, который определяется 7-ю младшими битами регистра SCFQCTL.

Бит DCOPLUS определяет частоту  $f_{DCOCLK}$ , равную  $f_{DCO}$  или  $f_{DCO/D}$ . Биты FLLDx определяют значение делителя D, который может принимать значения 1, 2, 4 или 8. По умолчанию установлено DCOPLUS = 0 и D = 2, что соответствует тактовой частоте  $f_{DCO/2}$  на выходе  $f_{DCOCLK}$ .

Коэффициент умножения (N+1) и делитель D определяют частоту DCOCLK.

$$\begin{aligned} \text{DCOPLUS} = 0: f_{DCOCLK} &= (N + 1) \times f_{ACLK} \\ \text{DCOPLUS} = 1: f_{DCOCLK} &= D \times (N + 1) \times f_{ACLK} \end{aligned}$$

#### Диапазон частот генератора DCO

Частотный диапазон  $f_{DCO}$  определяется битами FNx, как указано в Таблице 4-1. Контроль диапазона позволяет генератору DCO работать на частоте, близкой к центру диапазона перестройки для желаемой частоты DCOCLK. При этом пользователь должен следить за тем, чтобы частота MCLK не превышала допустимых значений, указанных в руководстве пользователя на конкретный микроконтроллер.

**Таблица 4-1. Биты контроля диапазона DCO**

FN_8	FN_4	FN_3	FN_2	Типовой диапазон частот $f_{DCO}$
0	0	0	0	0,65 – 6,1
0	0	0	1	1,3 – 12,1
0	0	1	x	2 – 17,9
0	1	x	x	2,8 – 26,6
1	x	x	x	4,2 – 46

#### 4.2.5. Автоподстройка частоты (FLL)

Сигнал с выхода системы автоподстройки частоты непрерывно инкрементирует либо декрементирует 10-битный интегратор частоты. Состояние выхода интегратора, управляющего генератором DCO, может быть прочитано в регистрах SCFI1 и SCFI0. За каждый период частоты ACLK значение может измениться на плюс или минус единицу.

Пять бит регистра управления интегратором, от SCFI13 до 7, устанавливают рабочую частоту генератора DCO. Для генератора DCO существуют 29 различных значений частоты (значения 28, 29, 30, и 31 эквивалентны), каждый из которых примерно на 10% больше предыдущего. Модулятор использует две соседние частоты генератора DCO для получения промежуточных частот. Биты 2-0 регистра SCFI1 и биты 1-0 регистра SCFI0 bits 1-0 определяют состояние модулятора.

Генератор DCO после сброса по питанию PUC или после очистки регистров SCFI0 и SCFI1 запускается на нижнем значении из диапазона перестройки. Для установления стабильного режима работы генератору DCO требуется определённое время. Т.к. интервал перестройки частоты генератора DCO составляет 32 такта частоты ACLK, процесс стабилизации частоты в худшем случае может занимать  $32 \times 28$  такта частоты ACLK.

#### **4.2.6. Модулятор генератора DCO**

Модулятор использует две соседние частоты генератора DCO для получения эффективных значений промежуточных частот и расширения спектра частот генератора, что уменьшает электромагнитное взаимодействие (EMI). Модулятор смешивает две соседние частоты на протяжении 32 тактов DCOCLK.

Ошибка результирующей за 32 такта частоты DCOCLK равна нулю и не накапливается. Контроль над значением модулятора и контрольными регистрами генератора DCO осуществляются модулем FLL автоматически. На рис. 4-3 проиллюстрирована работа модулятора.

#### **4.2.7. Запрет устройства стабилизации частоты FLL и модулятора**

Устройство стабилизации частоты FLL находится в выключенном состоянии, когда бит статусного регистра SCG0 = 1. При этом, генератор DCO работает на последней из установленных частот и частота DCOCLK не стабилизируется автоматически.

Модулятор частоты генератора DCO находится в выключенном состоянии, когда SCFQ\_M = 1. В этом случае, частота DCOCLK стабилизируется на ближайшей к требуемому значению частоте генератора DCO.

#### **4.2.8. Работа модуля FLL в режимах пониженного энергопотребления**

При возникновении запроса прерывания биты SCG1, CPUOFF и OSCOFF автоматически обнуляются, но бит SCG0 остаётся неизменным. Это означает, что при запуске обработчика прерывания из режимов LPM1, 2, 3 или 4 модуль FLL останется выключенным, и генератор DCO будет работать на частоте, определяемой предыдущими настройками регистров SCFI0 и SCFI1. Если в данном случае требуется работа модуля FLL, бит SCG0 должен быть принудительно обнулён в пользовательской программе.

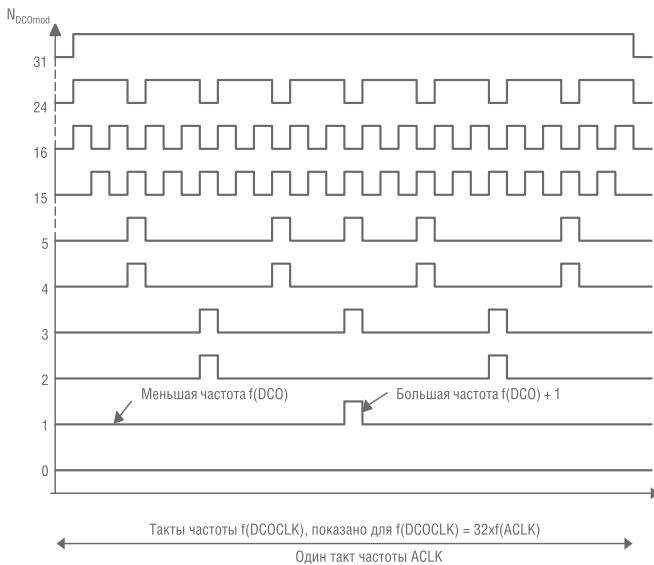


Рис. 4-3. Работа модулятора

#### 4.2.9. Буферизованный выход тактовой частоты

Частота ACLK может быть поделена на 1, 2, 4, или 8, буферизована и выведена наружу через вывод P1.5. Коэффициент деления задаётся битами FLL\_DIV.

Выход ACLK может быть мультиплексирован с другими функциями данного вывода. В этом случае его следует сконфигурировать как выход ACLK.

<i>BIS.B #P1SEL_5,&amp;P1SEL</i>	<i>;Выдать частоту ACLK/n</i>
<i>;на вывод P1.5</i>	
<i>BIS.B #P1DIR_5,&amp;P1DIR</i>	<i>;Направление вывода</i>
	<i>;P1.5 – выход</i>

#### 4.2.10. Бессбойная работа модуля FLL+

Модуль FLL+ обладает особыми функциями для обеспечения бессбойной работы. Эти функции обеспечиваются детектором сбоя генерации для LFXT1, DCO и XT2 как показано на рис. 4-4.

Возможные источники сигнала сбоя:

- Сбой низкочастотного генератора (LFOF) для LFXT1 в режиме LF
- Сбой высокочастотного генератора (XT1OF) для LFXT1 в режиме HF
- Сбой высокочастотного генератора (XT2OF) для XT2
- Сбой генератора DCO (DCOF) для DCO

Биты сбоя генерации LFOF, XT1OF и XT2OF устанавливаются в том случае, если соответствующий генератор включен, но не работает в нормальном режиме. Эти биты остаются установленными до тех пор, пока присутствует условие сбоя, и автоматически сбрасываются при переходе генератора в нормальный режим. При сбое генератора LFXT1 частота ACLK не вырабатывается и модуль FLL+ продолжает счёт до нуля при попытке уравнять частоты ACLK и MCLK/(Dx[N+1]). Рабочая частота генератора DCO при этом смещается на минимально возможное значение (биты от SCFI1.7 до SCFI1.3 обнуляются) и устанавливается бит DCOF. Этот бит также устанавливается в том случае, если значение умножителя N слишком велико для выбранного диапазона частот генератора DCO, и частота генератора приняла максимально возможное значение в диапазоне (биты SCFI1.7...SCFI1.3 установлены). Флаг DCOF автоматически очищается в том случае, если частота генератора DCO не равна максимальной или минимальной в диапазоне.

Флаг сбоя генерации OFIFG устанавливается при сбросе POR и при обнаружении сбоя любого из генераторов (установлен LFOF, XT1OF, XT2OF, или DCOF). После того, как флаг OFIFG установлен, источником для MCLK является генератор DCO. При этом, если установлен бит OFIE, флаг OFIFG вызывает запрос немаскируемого прерывания NMI. После обработки прерывания бит OFIE автоматически сбрасывается. Флаг OFIFG должен очищаться в пользовательской программе. Определить источник сбоя можно анализом соответствующих бит.

Если обнаружен сбой генерации в кварцевом генераторе, задействованном для MCLK, источник тактирования MCLK автоматически переключается на генератор DCO. При этом биты SELMx не изменяются. Эта ситуация должна корректно обрабатываться пользовательской программой.

#### Примечание: Активность генератора DCO при сбое генерации

*DCOCLK является активным даже при минимальной частоте генератора DCO в диапазоне. Тактирование ЦПУ не прекращается, что позволяет ему продолжить выполнение программы и обслуживание прерывания NMI при возникновении сбоя генерации.*

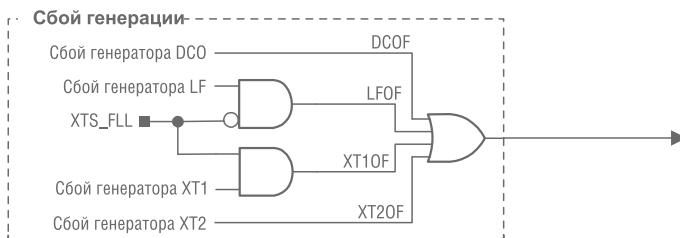


Рис. 4-4. Логика работы детектора сбоя генерации

## 4.3. Регистры модуля тактирования FLL+

Регистры модуля тактирования FLL+ перечислены в таблице 4-2.

**Таблица 4-2. Регистры модуля тактирования FLL+**

Регистр	Краткое название	Тип	Адрес	Начальное состояние
Контроль тактирования системы	SCFQCTL	Чтение/запись	052h	01Fh по сбросу PUC
Интегратор системной частоты 0	SCFI0	Чтение/запись	050h	040h по сбросу PUC
Интегратор системной частоты 1	SCFI1	Чтение/запись	051h	Обнулён по сбросу PUC
Регистр управления модулем FLL+ 0	FLL_CTL0	Чтение/запись	053h	003h по сбросу PUC
Регистр управления модулем FLL+ 1	FLL_CTL1	Чтение/запись	054h	Обнулён по сбросу PUC
Регистр разрешения прерываний 1	IE1	Чтение/запись	000h	Обнулён по сбросу PUC
Регистр флагов прерываний 1	IFG1	Чтение/запись	002h	Обнулён по сбросу PUC

### SCFQCTL, регистр контроля тактирования системы

7	6	5	4	3	2	1	0	
<b>SCFQ_M</b>		<b>N</b>						
rw-0	rw-0	rw-0	rw-1	rw-1	rw-1	rw-1	rw-1	
<b>SCFQ_M</b>		Бит 7						
		Модуляция. Этот бит разрешает либо запрещает модуляцию 0 – модуляция запрещена 1 – модуляция разрешена						
<b>N</b>		Биты 6-0						
		Коэффициент умножения. Эти биты устанавливают коэффициент умножения для DCO. Если DCOPLUS=0: $f_{DCOCLK} = (N + 1) \times f_{\text{кварца}}$ Если DCOPLUS=1: $f_{DCOCLK} = D \times (N + 1) \times f_{\text{кварца}}$						

### SCFI0, Интегратор системной частоты 0

7	6	5	4	3	2	1	0	
<b>FLLDx</b>		<b>FN_x</b>				<b>MODx (M3P)</b>		
rw-0	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	
<b>FLLDx</b>		Биты 7-6						
		Делитель цепи обратной связи модуля FLL. Этими битами определяется коэффициент деления $f_{DCOCLK}$ в цепи ОС модуля FLL+. Результат такого деления эквивалентен дополнительному множителю для бит умножения. См. также Биты умножения 00/1 01/2 10/4 11/8						

<b>FN_x</b>	Биты 5-2	Контроль диапазона перестройки генератора DCO. Эти биты определяют диапазон перестройки частоты $f_{DCO}$ . 0000 0.65 – 6.1 МГц 0001 1.3 – 12.1 МГц 001x 2 – 17.9 МГц 01xx 2.8 – 26.6 МГц 1xxx 4.2 – 46 МГц
<b>MODx</b>	Биты 1-0	Младшие значащие биты модулятора. Бит 0 – МЗР модулятора. Эти биты влияют на поведение модулятора. Все биты MODx автоматически управляются модулем FLL+.

### SCFI1, Интегратор системной частоты 1

7	6	5	4	3	2	1	0
DCOx				MODx (C3P)			
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
<b>DCOx</b>	Биты 7-3		Эти биты определяют диапазон частоты, они автоматически управляются модулем FLL+.				
<b>MODx</b>	Биты 2-0		Старшие значащие биты модулятора. Бит 2 – С3Р модулятора. Эти биты влияют на поведение модулятора. Все биты MODx автоматически управляются модулем FLL+..				

### FLL\_CTL0, Регистр управления модулем FLL+0

7	6	5	4	3	2	1	0
DCOPLUS	XTS_FLL	XCAPxPF	XT20F*	XT10F	LFOF	DCOF	
rw-0	rw-0	rw-0	rw-0	r-0	r-0	r-(1)	r-1

\* В устройствах MSP430x41x, MSP430x42x отсутствует

<b>DCOPLUS</b>	Бит 7	Предделитель выхода DCO. Этот бит определяет будет ли поделена частота DCO перед тем, как она будет подана на MCLK или SMCLK. Коэффициент деления задаётся битами FLL_DIV 0 – выход DCO поделен 1 – выход DCO не делится
<b>XTS_FLL</b>	Бит 6	Выбор режима генератора LFXT1 0 – низкочастотный режим 1 – высокочастотный режим
<b>XCAPxPF</b>	Биты 5-4	Выбор конденсаторов генератора. Эти биты определяют значение ёмкости для кварцевого или керамического резонатора генератора LFXT1. 00 ~1 пФ 01 ~6 пФ 10 ~8 пФ 11 ~10 пФ

<b>XT2OF</b>	Бит 3	Сбой генератора XT2. В устройствах MSP430x41x, MSP430x42x отсутствует 0 – нет сбоя 1 – присутствует сбой
<b>XT10F</b>	Бит 2	Сбой генератора LFXT1 в высокочастотном режиме. 0 – нет сбоя 1 – присутствует сбой
<b>LFOF</b>	Бит 1	Сбой генератора LFXT1 в низкочастотном режиме. 0 – нет сбоя 1 – присутствует сбой
<b>DCOF</b>	Бит 0	Сбой генератора DCO. 0 – нет сбоя 1 – присутствует сбой

**FLL\_CTL0, Регистр управления модулем FLL+1**

	7	6	5	4	3	2	1	0
<b>Не используется</b>	<b>SMCLKOFF*</b>	<b>XT2OFF*</b>	<b>SELM*</b>	<b>SELS*</b>	<b>FLL_DIVx</b>			
r-0	r-1	rw-(1)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

\* В устройствах MSP430x41x, MSP430x42x отсутствует

<b>Не используется</b>	Бит 7	
<b>SMCLKOFF</b>	Бит 6	Отключение SMCLK. Этот бит отключает SMCLK. В устройствах MSP430x41x, MSP430x42x отсутствует 0 – SMCLK включена 1 – SMCLK выключена
<b>XT2OFF</b>	Бит 5	Отключение XT2. Этот бит отключает генератор XT2. В устройствах MSP430x41x, MSP430x42x отсутствует 0 – XT2 включен 1 – XT2 выключен, если он не задействован для MCLK или SMCLK.
<b>SELM</b>	Биты 4-3	Выбор MCLK. Эти биты выбирают источник тактирования MCLK. В устройствах MSP430x41x, MSP430x42x отсутствует 00 – DCOCLK 01 – DCOCLK 10 – XT2CLK 11 – LFXT1CLK
<b>SELS</b>	Бит 2	Выбор SMCLK. Эти биты выбирают источник тактирования SMCLK. В устройствах MSP430x41x, MSP430x42x отсутствует 0 – DCOCLK 1 – XT2CLK

<b>FLL_DIVx</b>	Биты 1-0	Делитель ACLK 00/1 01/2 10/4 11/8
-----------------	----------	---

### IE1, Регистр разрешения прерываний 1

7	6	5	4	3	2	1	0
						<b>OFIE</b>	

rw-0

	Биты 7-2	Эти биты могут быть использованы другими модулями. См. документацию на конкретный МК
<b>OFIE</b>	Бит 1	Разрешение прерывания по сбою генерации. Этот бит разрешает прерывание по флагу OFIFG. Так как другие биты в регистре IE1 могут использоваться другими модулями, рекомендуется устанавливать и сбрасывать данный бит используя инструкции BIS.B или BIC.B, а не MOV.B или CLR.B. 0 – прерывание запрещено 1 – прерывание разрешено
	Бит 0	Этот бит может быть использован другими модулями. См. документацию на конкретный МК

### IFG1, Регистр флагов прерываний 1

7	6	5	4	3	2	1	0
						<b>OFIFG</b>	

rw-0

	Биты 7-2	Эти биты могут быть использованы другими модулями. См. документацию на конкретный МК
<b>OFIFG</b>	Бит 1	Флаг прерывания по сбою генерации. Так как другие биты в регистре IE1 могут использоваться другими модулями, рекомендуется устанавливать и сбрасывать данный бит используя инструкции BIS.B или BIC.B, а не MOV.B или CLR.B. 0 – прерывание запрещено 1 – прерывание разрешено
	Бит 0	Этот бит может быть использован другими модулями. См. документацию на конкретный МК

# MSP430x4xxFamily

## Контроллер флэш-памяти

*Раздел V.*



## 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.

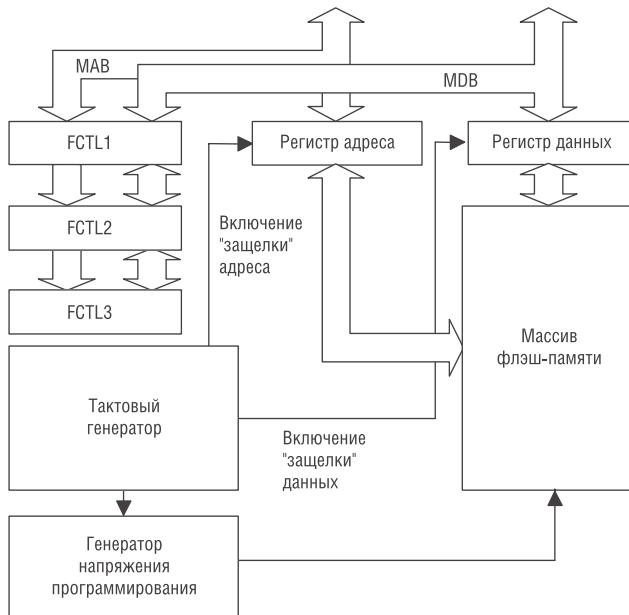


Рис. 5-1. Блок-схема модуля флаш-памяти

**Примечание: Минимальное напряжение  $V_{cc}$  во время записи или стирания флэш-памяти**

Минимальное значение напряжения  $V_{cc}$  во время записи или стирания флэш-памяти должно составлять 2,7 В. Если  $V_{cc}$  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.

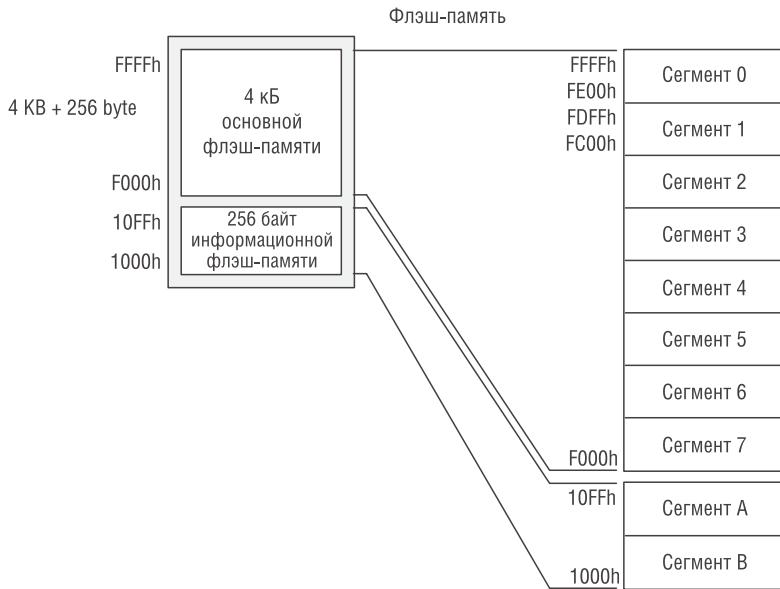
На рис. 5.2 показана сегментация памяти на основе примера 4 кБ флэш-памяти, имеющей восемь основных сегментов и оба информационных сегмента.

## 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)

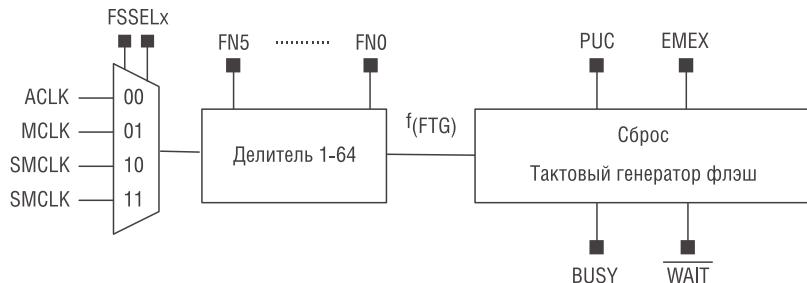


**Рис. 5-2.** Сегменты флэш-памяти, пример для 4кБ

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  $\sim 257$  кГц до  $\sim 476$  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, гарантировавших надежную работу.

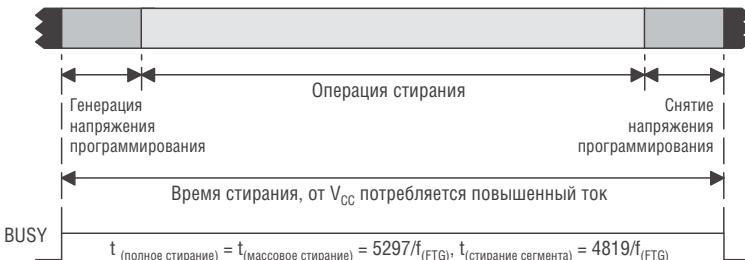
### 5.3.2. Стирание флэш-памяти

После стирания бит флэш-памяти читается как «1». Можно программирувать 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<sup>1</sup> 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

<sup>1</sup> 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 из 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.

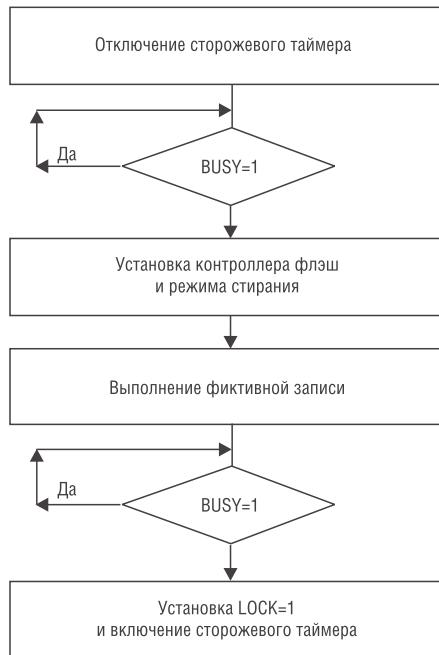
```
;Стирание сегмента из флэш. 514 кГц < SMCLK < 952 кГц
;Принимается ACCVIE = NMIIE = OFIE = 0.
MOV #WDTPW+WDTHOLD,&WDTCTL ;Отключение WDT
MOV #FWKEY+FSSEL1+FNO,&FCTL2 ;SMCLK/2
MOV #FWKEY,&FCTL3 ;Очистка LOCK
MOV #FWKEY+ERASE,&FCTL1 ;Разрешение стирания
                           ;сегмента
CLR &0FC10h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

### Иницирование 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
L1 BIT #BUSY,&FCTL3          ;Проверка BUSY
JNZ L1                         ;Ожидание, пока занято
MOV #FWKEY+FSSEL1+FNO,&FCTL2 ;SMCLK/2
MOV #FWKEY,&FCTL3             ;Очистка LOCK
MOV #FWKEY+ERASE,&FCTL1       ;Разрешение стирания
CLR &0FC10h                     ;Фиктивная запись,
                               ;стирание S1
```



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

```

L2 BIT #BUSY,&FCTL3           ;Проверка BUSY
      JNZ L2                   ;Ожидание, пока занято
      MOV #FWKEY+LOCK,&FCTL3   ;Завершено, установка
                                ;LOCK
      . . .
                                ;Повторное включение
                                ;WDT?
  
```

### 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.

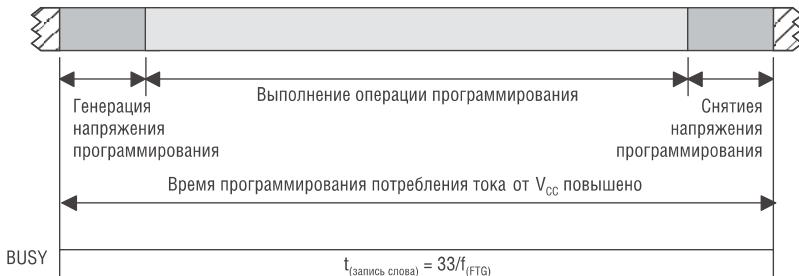


Рис. 5-7. Временная диаграмма операции записи байта/слова

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.

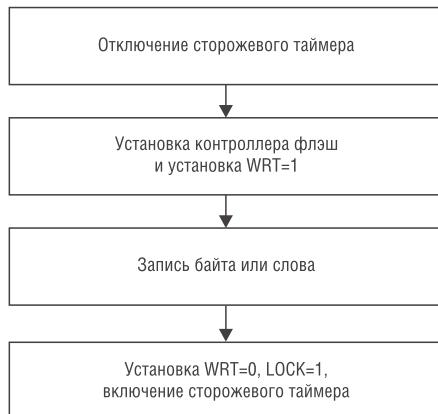


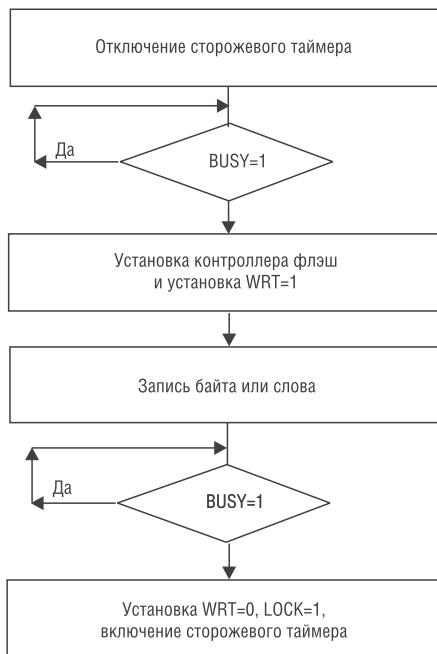
Рис. 5-8. Инициирование записи байта/слова из флэш-памяти

;Запись байта/слова из флэш. 514 кГц < SMCLK < 952 кГц  
;Принимается, что OFF1Eh уже стерто  
;Принимается ACCVIE = NMIIE = OFIE = 0.  
MOV #WDTPW+WDTHOLD, &WDTCTL ;Отключение сторожевого  
;таймера  
MOV #FWKEY+FSSEL1+FNO, &FCTL2 ;SMCLK/2  
MOV #FWKEY, &FCTL3 ;Очистка LOCK  
MOV #FWKEY+WRT, &FCTL1 ;Разрешение записи  
MOV #0123h, &OFF1Eh ;0123h -> OFF1Eh  
MOV #FWKEY, &FCTL1 ;Выполнено. Очистка WRT  
MOV #FWKEY+LOCK, &FCTL3 ;Установка LOCK  
... ;Повторный запуск  
;сторожевого таймера ?

### Инициирование записи байта/слова программой из ОЗУ

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

;Запись байта/слова из ОЗУ. 514 кГц < SMCLK < 952 кГц  
;Принимается, что OFF1Eh уже стерто  
;Принимается ACCVIE = NMIIE = OFIE = 0.  
MOV #WDTPW+WDTHOLD, &WDTCTL ;Отключение сторожевого  
;таймера



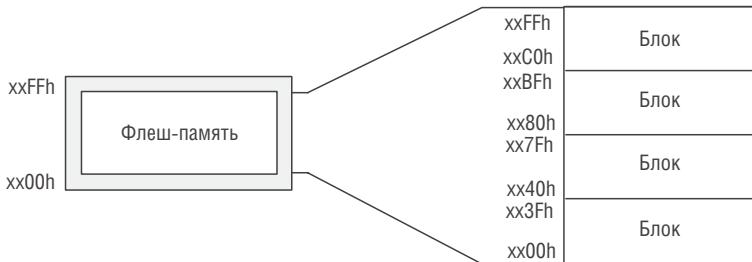
**Рис. 5-9.** Инициирование записи байта/слова из ОЗУ

```

L1 BIT #BUSY,&FCTL3 ;Проверка BUSY
JNZ L1 ;Ожидание в цикле,
;пока занято
MOV #FWKEY+FSSEL1+FNO,&FCTL2 ;SMCLK/2
MOV #FWKEY,&FCTL3 ;Очистка LOCK
MOV #FWKEY+WRT,&FCTL1 ;Разрешение записи
MOV #0123h,&0FF1Eh ;0123h -> OFF1Eh
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 ?
...
  
```

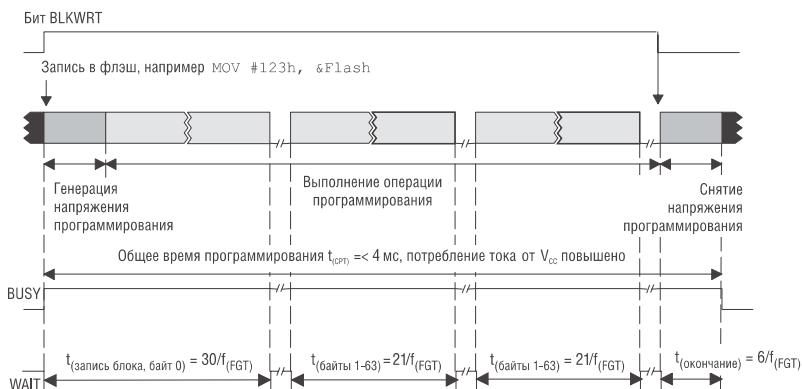
### Запись блока

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 за-



**Рис. 5-11.** Временная диаграмма цикла блочной записи

пиши 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.

### 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
L1   BIT #BUSY,&FCTL3      ;Проверка BUSY
    JNZ L1                ;Ожидание в цикле,
                        ;пока занято
    MOV #FWKEY+FSSEL1+FNO,&FCTL2 ;SMCLK/2
    MOV #FWKEY,&FCTL3          ;Очистка LOCK
    MOV #FWKEY+BLKWRT+WRT,&FCTL1 ;Разрешение записи
                                ;блока
L2   MOV Write_Value,0(R6)  ;Месторасположение
                            ;записи
    BIT #WAIT,&FCTL3        ;Проверка WAIT
    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,BLKWR
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
```

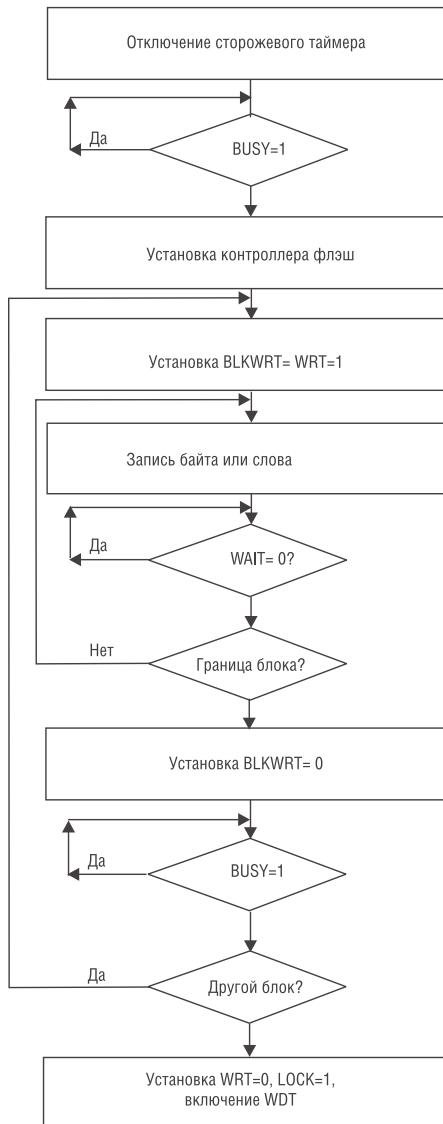


Рис. 5-12. Программный поток блочного записи

### 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.
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/сlova 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<sup>1</sup>
- 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<sup>2</sup>*» на сайте [www.ti.com/sc/msp430](http://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<sup>3</sup>. Доступ к флэш-памяти MSP430 через BSL защищен 256-разрядным паролем, определяемым пользователем. Подробности см. в приложении «*Features of the MSP430 Bootstrap Loader<sup>4</sup>*» на сайте [www.ti.com/sc/msp430](http://www.ti.com/sc/msp430).

#### Программирование флэш-памяти через пользовательское решение

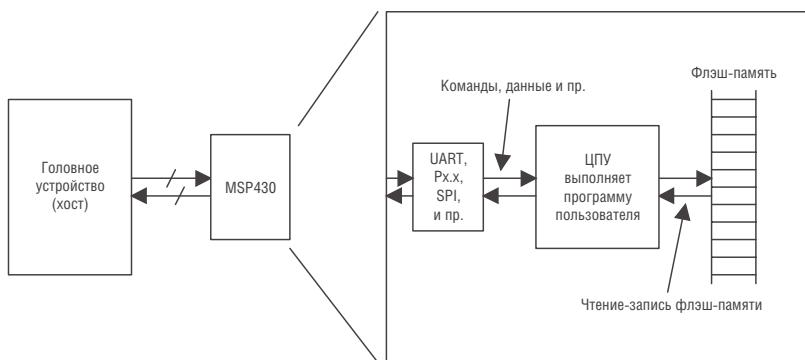
Способность ЦПУ в MSP430 записывать собственную флэш-память позволяет реализовать внутрисистемное программирование внешними пользовательскими решениями, как показано на рис. 5.13. Пользователь может выбрать, каким образом данные будут поступать в MSP430 с использованием любого имеющегося доступного способа (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.

<sup>1</sup> JTAG (Joint Test Automation Group) interface – интерфейс «объединенной рабочей группы по автоматизации тестирования»

<sup>2</sup> «Программирование MSP430 с флэш-памятью через JTAG-интерфейс»

<sup>3</sup> UART (Universal Asynchronous Receiver / Transmitter) - универсальный асинхронный приемопередатчик

<sup>4</sup> «Возможности самозагрузчика MSP430»



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

## 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
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

15	14	13	12	11	10	9	8
----	----	----	----	----	----	---	---

FRKEY, читается как 096h FWKEY, должен записываться как 0A5h
---

rw-0

	7	6	5	4	3	2	1	0
BLKWRT	WRT	Резерв	Резерв	Резерв	MERAS	ERASE	Резерв	Резерв
rw-0	rw-0	r0	r0	r0	rw-0	rw-0	r0	
<b>FRKEY/FWKEY</b>		Биты 15-8		Пароль FCTLx. Всегда читается как 096h. Должен записываться как 0A5h, в противном случае будет генерироваться сигнал PUC.				
<b>BLKWRT</b>		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				
<b>WRT</b>		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				
<b>Зарезервировано</b>		Биты 5-3		Зарезервировано. Всегда читается как 0.				
<b>MERAS</b>		Бит 2		Массовое стирание и обычное стирание. Эти биты используются совместно для выбора режима стирания. Биты MERAS и ERASE автоматически сбрасываются, когда устанавливается EMEX.				
<b>ERASE</b>		Бит 1						
<b>MERAS</b>	<b>ERASE</b>			<b>ЦИКЛ СТИРАНИЯ</b>				
0	0			Нет стирания				
0	1			Стирание только конкретного сегмента				
1	0			Стирание всех сегментов основной памяти				
0	1			Стирание всех сегментов основной и информационной памяти				
<b>Зарезервировано</b>	Бит 0			Зарезервировано. Всегда читается как 0.				

**FCTL2, регистр управления флэш-памятью**

15	14	13	12	11	10	9	8
<b>FWKEY<sub>x</sub>, читается как 096h должен записываться как 0A5h</b>							

	7	6	5	4	3	2	1	0	
	<b>FSSELx</b>		<b>FNx</b>						
rw-0	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1	rw-0	
<b>FWKEYx</b>	<b>Биты 15-8</b>		Пароль FCTLx. Всегда читается как 096h. Должен записываться как 0A5h, в противном случае будет генерироваться сигнал PUC.						
<b>FSSELx</b>	<b>Биты 7-6</b>		Выбор источника тактирования контроллера флэш 00 – ACLK 01 – MCLK 10 – SMCLK 11 – SMCLK						
<b>FNx</b>	<b>Биты 5-0</b>		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

	15	14	13	12	11	10	9	8	
<b>FWKEYx, читается как 096h должен записываться как 0A5h</b>									
<b>Резерв</b>	7	6	5	4	3	2	1	0	
r0	r0	rw-0	rw-1	r-1	rw-0	rw-(0)	r(w)-0		
<b>FWKEYx</b>	<b>Биты 15-8</b>		Пароль FCTLx. Всегда читается как 096h. Должен записываться как 0A5h, в противном случае будет генерироваться сигнал PUC.						
<b>Зарезервировано</b>	<b>Биты 7-6</b>		Зарезервировано. Всегда читается как 0.						
<b>EMEX</b>	<b>Бит 5</b>		Аварийный выход 0 – Нет аварийного выхода 1 – Аварийный выход						

<b>LOCK</b>	<b>Бит 4</b>	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
<b>WAIT</b>	<b>Бит 3</b>	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.
<b>ACCVIFG</b>	<b>Бит 2</b>	Флаг прерывания при нарушении прав доступа 0 – Прерывание не ожидается 1 – Ожидание прерывание
<b>KEYV</b>	<b>Бит 1</b>	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
<b>BUSY</b>	<b>Бит 0</b>	Занято. Этот бит показывает состояние тактового генератора флэш. 0 – Не занят 1 – Занят

**IE1, регистр 1 разрешения прерывания**

7	6	5	4	3	2	1	0
		<b>ACCVIE</b>					

rw-0

	<b>Биты 7-6, 4-0</b>	Эти биты могут быть использованы для других модулей. См. справочное руководство конкретного устройства.
--	--------------------------	---

<b>ACCVIE</b>	<b>Бит 5</b>	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
---------------	--------------	---

# MSP430x4xxFamily

## Супервизор напряжения питания

*Раздел VI.*



## Супервизор напряжения питания

В этом разделе описывается работа супервизора напряжения питания (SVS<sup>1</sup>). Модуль SVS реализован во всех устройствах MSP430x4xx.

### 6.1. Введение в SVS

Супервизор напряжения питания (SVS) используется для мониторинга напряжения питания  $AV_{cc}$  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  $AV_{cc}$ ;
- 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.

**Примечание: Детектор уровня напряжения в MSP430x412/MSP430x413**

В микроконтроллерах MSP430x412 и MSP430x413 реализована возможность установки уровня срабатывания на фиксированное значение. При установке значения  $VLDx=0$  модуль SVS отключается. Установка любого другого значения  $VLDx$ , отличного от 0, активизирует модуль SVS, устанавливая уровень срабатывания 1.9 В.

### 6.2. Функционирование SVS

SSVS определяет снижение напряжения  $AV_{CC}$  ниже заданного уровня. Модуль SVS можно сконфигурировать на выработку сигнала POR или установку флага при снижении напряжения. Модуль SVS отключается по сигналу сброса при пониженном напряжении питания (brownout reset), 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  $AV_{cc}$ . SVS выключен, когда  $VLDx=0$  и включен, когда  $VLDx>0$ . Бит SVSON не включает SVS. Он по-

<sup>1</sup> SVS – Supply Voltage Supervisor.

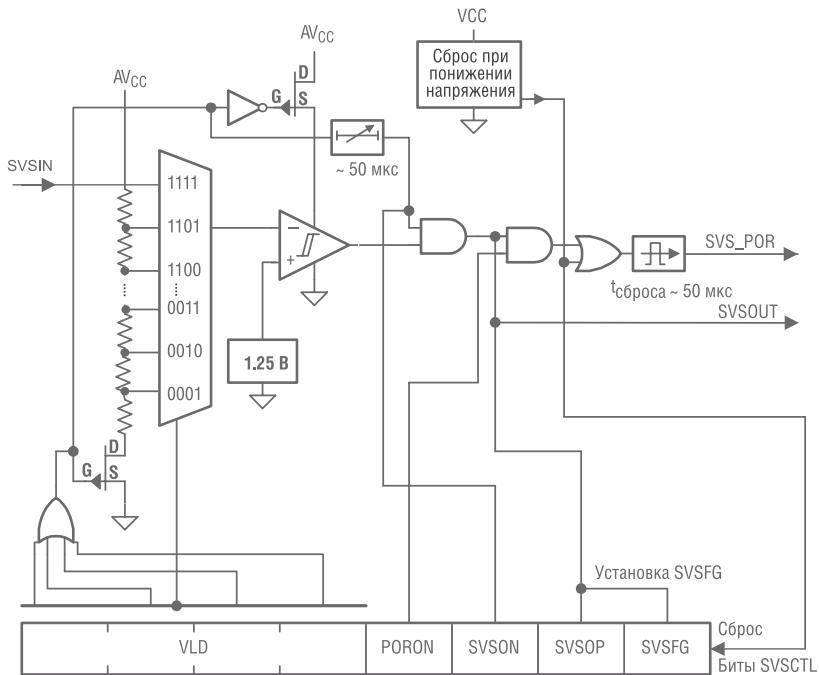


Рис. 6-1. Блок схема модуля SVS

казывает включенное/выключенное состояние модуля SVS и может использоваться для определения, включен ли SVS.

При  $VLD_x=1111$  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  $AV_{CC}$  понижается меньше выбранного порога или когда внешнее напряжение снижается ниже 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 дол-

жен сбрасываться программным обеспечением пользователя. Если состояние пониженного напряжения 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  $t_d(SVSon)$  и  $t_{settle}$  показаны на рис. 6.2. Задержка  $t_d(SVSon)$  действует, когда VLDx изменяются от нуля к любому отличному от нуля значению, и составляет примерно 50 мкС. Задержка  $t_{settle}$  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.

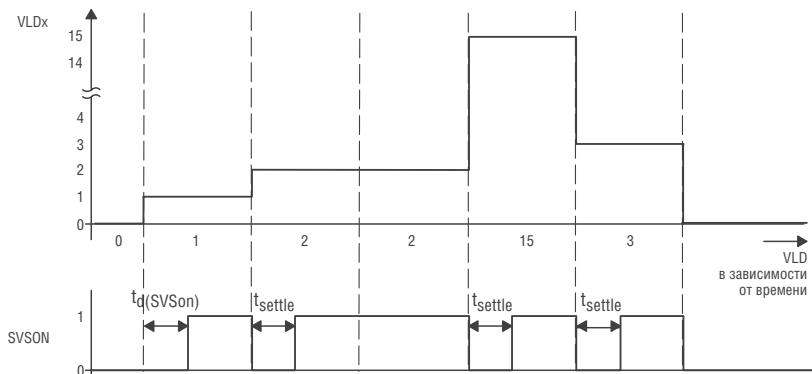


Рис. 6-2. Состояние бита SVSON при изменении VLDx

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  $AV_{CC}$  близка к

установленному порогу. Работа SVS и SVS/Brownout<sup>1</sup> взаимодействие показано на рис. 6.3.

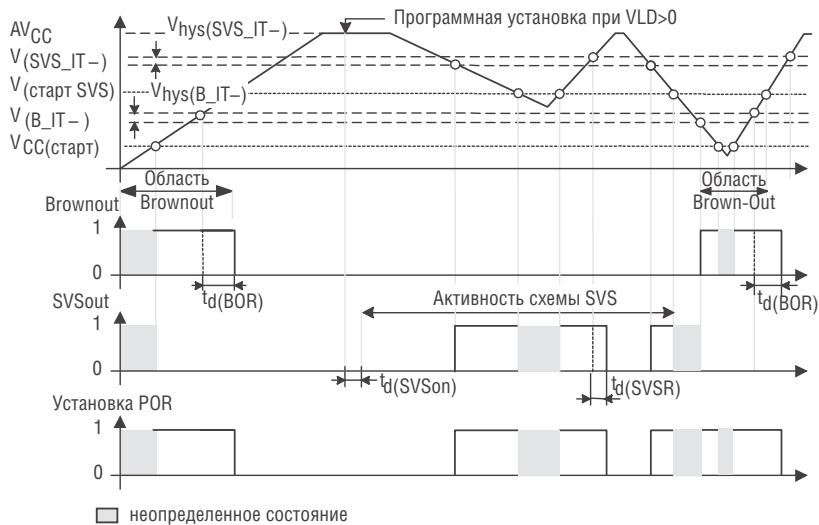


Рис. 6-3. Рабочие уровни для SVS и схемы Brownout/сброс

### 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	056h	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

<sup>1</sup> Brownout - понижение напряжения.

		Детектируемый уровень напряжения. Эти биты включают SVS и позволяют выбрать номинальный пороговый уровень напряжения SVS. Точные параметры см. в руководстве на конкретное устройство.
VLDx	Биты 7-4	0000 – SVS выключен 0001 – 1.9 В 0010 – 2.1 В 0011 – 2.2 В 0100 – 2.3 В 0101 – 2.4 В 0110 – 2.5 В 0111 – 2.65 В 1000 – 2.8 В 1001 – 2.9 В 1010 – 3.05 В 1011 – 3.2 В 1100 – 3.35 В 1101 – 3.5 В 1110 – 3.7 В 1111 – Сравнение внешнего напряжения на входе SVSin со значением 1.2 В
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.

# MSP430x4xxFamily

## Аппаратный умножитель

*Раздел VII.*



## Аппаратный умножитель

В этом разделе описывается аппаратный умножитель. Аппаратный умножитель реализован в устройствах MSP430x44x.

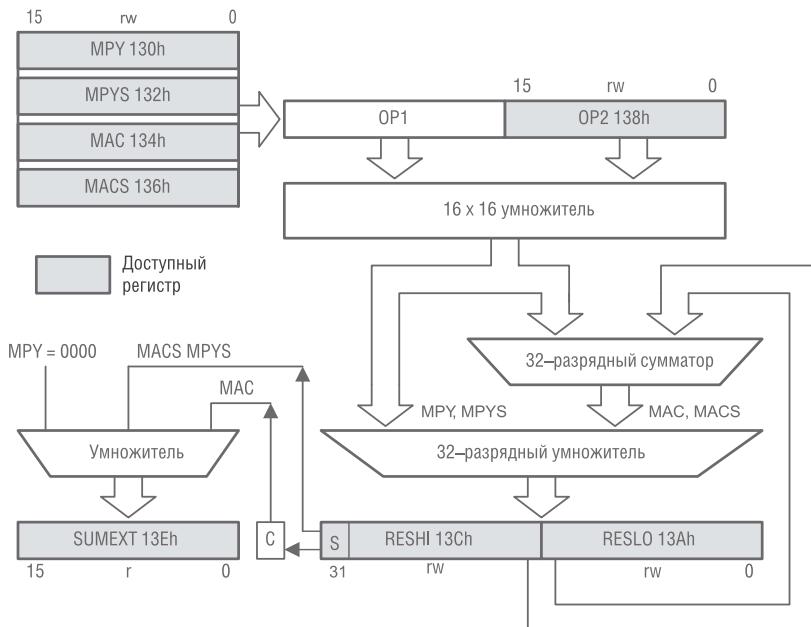
### 7.1. Введение в аппаратный умножитель

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 operand.

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. Operand 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
<b>MPY</b>	Старшие 16 разрядов результата
<b>MPYS</b>	В старшем бите MSB регистра находится знак результата. Оставшиеся биты содержат старшие 15 разрядов результата. Используется представление результата с дополнением до двух.
<b>MAC</b>	Старшие 16 разрядов результата
<b>MACS</b>	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
<b>MPY</b>	SUMEXT всегда содержит 0000h
<b>MPYS</b>	SUMEXT содержит расширенный знак результата 00000h результат был положительный 0FFFFh результат был отрицательный
<b>MAC</b>	SUMEXT содержит перенос результата 0000h результат не содержит переноса 0001h результат имеет перенос
<b>MACS</b>	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 FFFFh – 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 .В 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

# MSP430x4xxFamily

## Контроллер DMA

*Раздел VIII.*



## Контроллер DMA

Модуль контроллера DMA переносит данные из одного адреса в другой без участия ЦПУ. Этот раздел описывает работу контроллера DMA.

В устройствах MSP430xFG43x контроллер DMA реализован так, что предоставляет только один канал DMA.

### 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.

Устройства MSP430FG43x имеют только один канал DMA. Поэтому некоторые особенности, описанные в этой главе, к ним не применимы.

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:**

- Один канал 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.

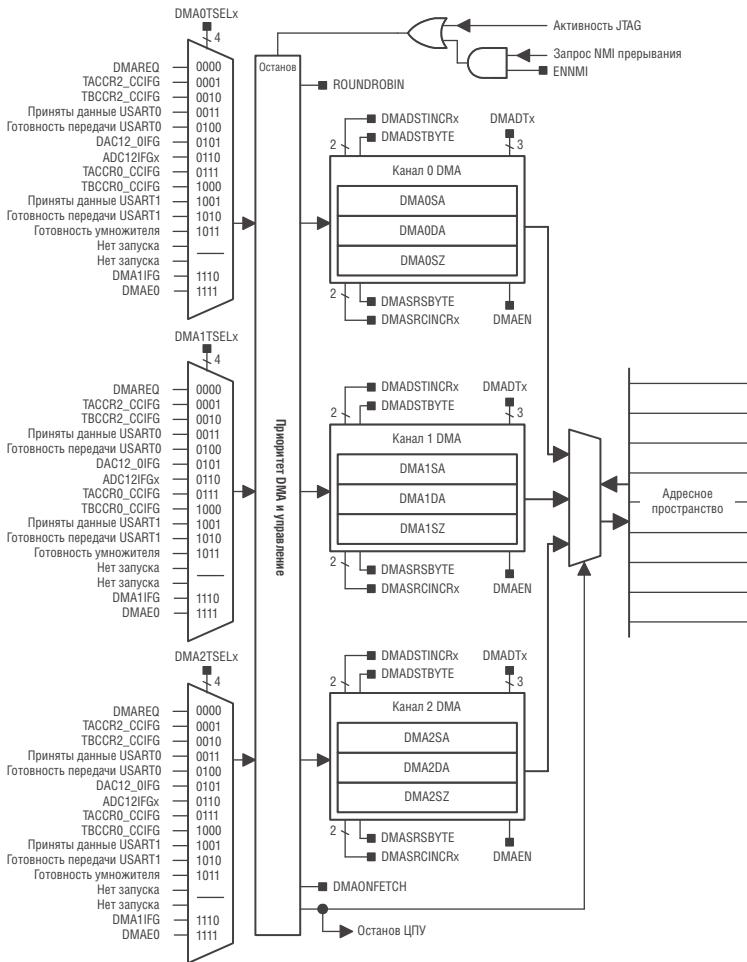
Блок-схема контроллера DMA показана на рис. 8.1.

### 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

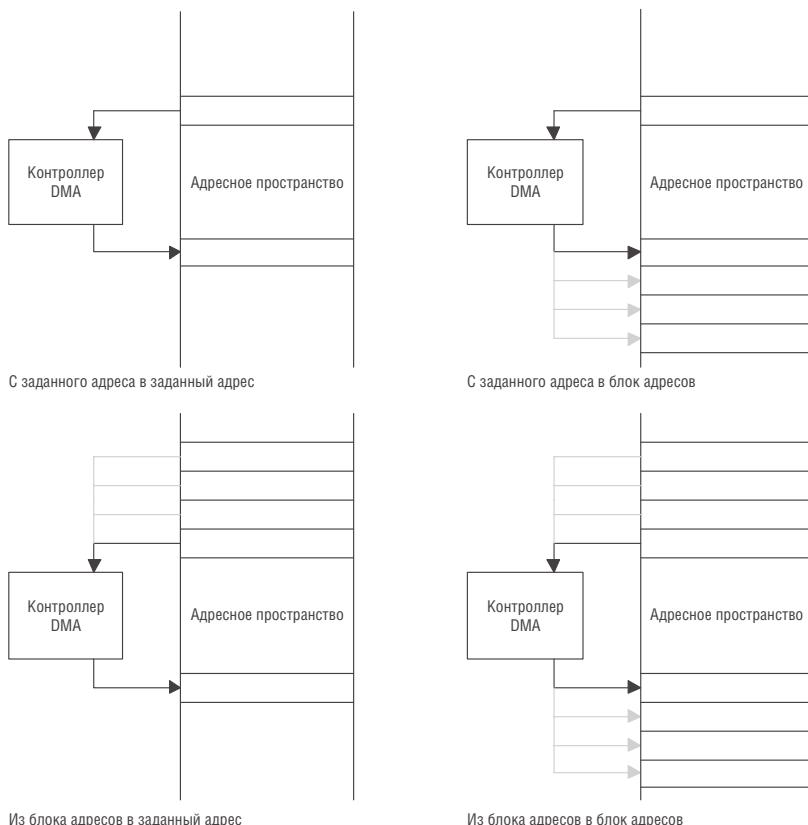
Контроллер DMA имеет четыре режима адресации. Режимы адресации каждого канала DMA конфигурируются независимо друг от друга. Например,



**Рис. 8-1.** Блок-схема контроллера DMA

канал 0 может выполнять перенос между двумя фиксированными адресами, в то время как в канале 1 выполняются переносы между двумя блоками адресов. Режимы адресации показаны на рис. 8.2. Существуют следующие режимы адресации:

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



**Рис. 8-2.** Режимы адресации DMA

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

Режимы адресации конфигурируются с помощью управляющих битов DMASRCINCR<sub>x</sub> и DMADSTINCR<sub>x</sub>. Биты DMASRCINCR<sub>x</sub> выбираются, если адрес источника инкрементируется, декрементируется или не изменяется после каждого переноса. Биты DMADSTINCR<sub>x</sub> 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

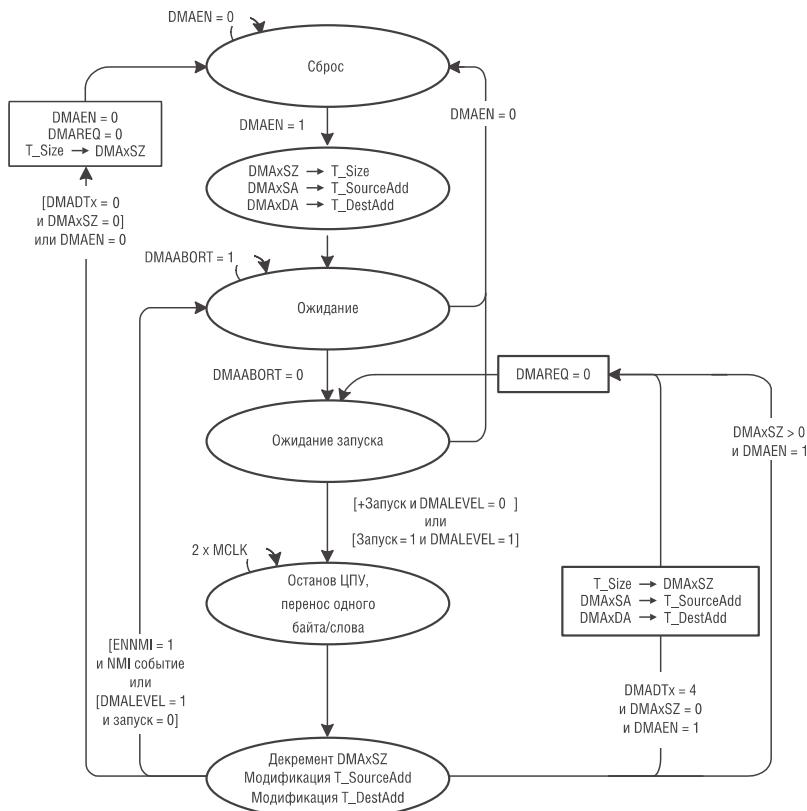
Контроллер DMA имеет шесть режимов переноса, определяемых битами DMADTx, в соответствии с таблицей 8-1. В каждом канале режим переноса конфигурируется индивидуально. К примеру, канал 0 может быть сконфигурирован в одиночном режиме переноса, канал 1 в режиме пакетно-блочного переноса, а канал 2 работать 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	Одиночный перенос	Каждый перенос нуждается в запуске. DMAEN автоматически очищается после выполнения количества переносов, определенного в DMAxSZ.
001	Блочный перенос	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.

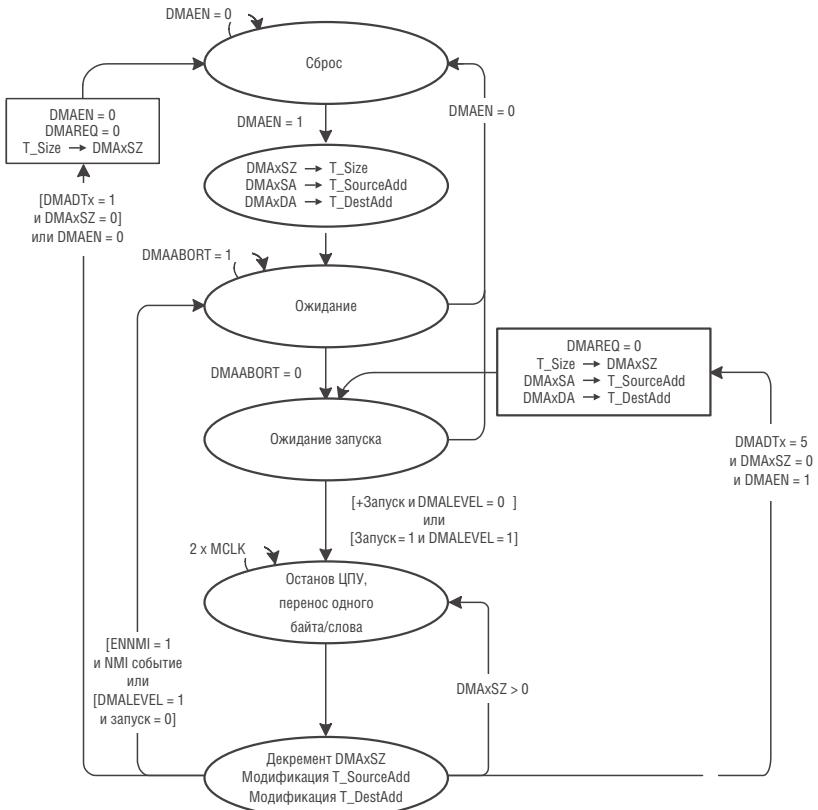
Регистры DMAxSA, DMAxDA и DMAxSZ копируются во временные регистры. Временные значения DMAxSA и DMAxDA инкрементируются или декрементируются после каждого переноса в блоке. Регистр DMAxSZ декрементируется после каждого переноса блока и содержит количество оставшихся в блоке переносов. Когда регистр DMAxSZ 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  $2 \times MCLK \times DMA \times SZ$  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 каждого 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 установлен



**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-

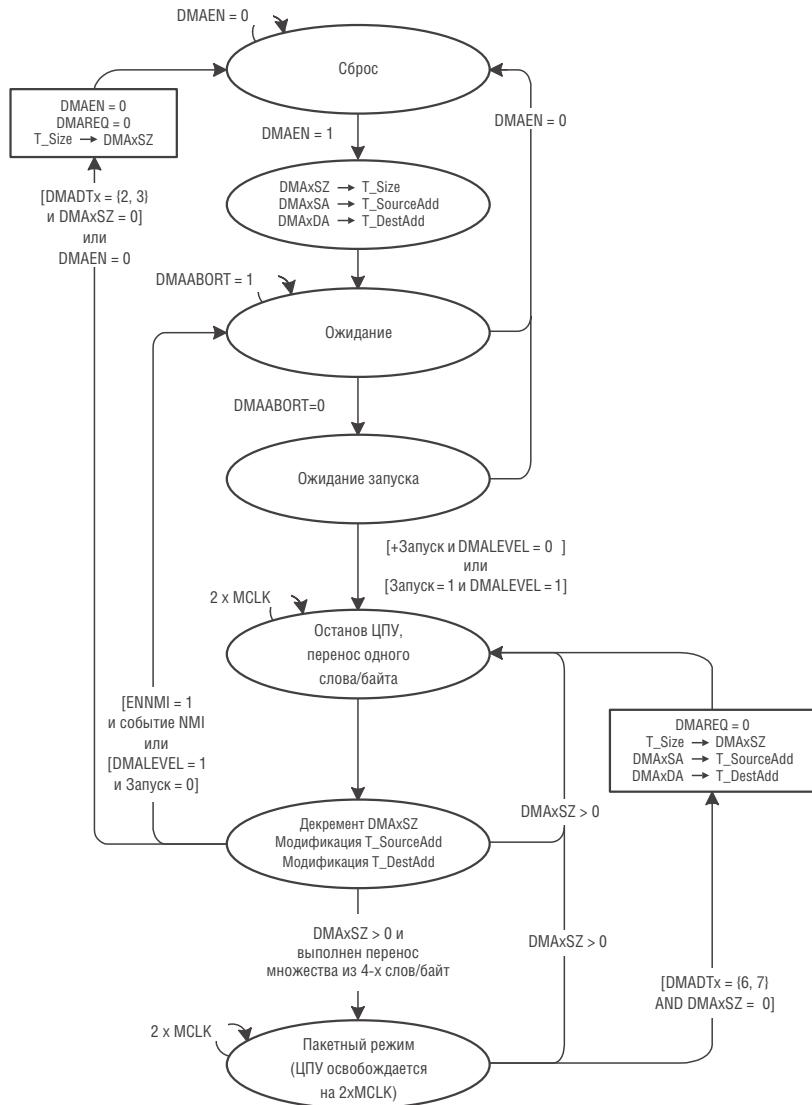


Рис. 8-5. Диаграмма состояний пакетно-блочного переноса DMA

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 останавливается 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)**

<b>DMAxTSELx</b>	<b>Действие</b>
<b>0100</b>	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.
<b>0101</b>	Перенос запускается, когда устанавливается флаг DAC12_OCTL DAC12IFG. Флаг DAC12_OCTL DAC12IFG автоматически очищается, когда начинается перенос. Если DAC12_OCTL DAC12IE установлен, флаг DAC12_OCTL DAC12IFG не запускает перенос.
<b>0110</b>	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 сбрасываются автоматически, когда контроллер DMA обращается к соответствующему регистру ADC12MEMx.
<b>0111</b>	Перенос запускается, когда устанавливается флаг TACCRO CCIFG. Флаг TACCRO CCIFG автоматически сбрасывается, когда перенос стартует. Если бит TACCRO CCIE установлен, флаг TACCRO CCIFG не запускает перенос.
<b>1000</b>	Перенос запускается, когда устанавливается флаг TBCCRO CCIFG. Флаг TBCCRO CCIFG автоматически сбрасывается, когда перенос стартует. Если бит TBCCRO CCIE установлен, флаг TBCCRO CCIFG не запускает перенос.
<b>1001</b>	Перенос запускается, когда устанавливается флаг URXIFG1. URXIFG1 автоматически сбрасывается, когда перенос стартует. Если URXIE1 установлен, флаг URXIFG1 не запускает перенос.
<b>1010</b>	Перенос запускается, когда устанавливается флаг UTXIFG1. UTXIFG1 автоматически сбрасывается, когда перенос стартует. Если UTXIE1 установлен, флаг UTXIFG1 не запускает перенос.
<b>1011</b>	Перенос запускается, когда аппаратный умножитель готов для нового операнда.
<b>1100</b>	Перенос не запускается.
<b>1101</b>	Перенос не запускается.
<b>1110</b>	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.
<b>1111</b>	Перенос запускается внешним сигналом от DMAEO.

### 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 возвращаются к приоритетам по умолчанию.

Понятие приоритета каналов DMA не применимо к устройствам MSP430FG43x.

### 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 I<sup>2</sup>C с контроллером DMA

Модуль I<sup>2</sup>C может стать источником двух условий запуска для контроллера DMA. Модуль I<sup>2</sup>C может запустить перенос, когда приняты новые данные I<sup>2</sup>C и когда появилась необходимость в передаче данных.

Биты TXDMAEN и RXDMAEN разрешают или запрещают использование контроллера DMA с модулем I<sup>2</sup>C. Когда RXDMAEN=1, контроллер DMA может быть использован для переноса данных из модуля I<sup>2</sup>C после приема данных модулем I<sup>2</sup>C. Когда RXDMAEN=1, RXRDYIE игнорируется и RXRDYIFG не генерирует прерывание.

Когда TXDMAEN=1, контроллер DMA может быть использован для переноса данных в модуль I<sup>2</sup>C 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	01EC <sub>h</sub>	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
<b>Зарезервировано</b>				<b>DMA2TSELX</b>			
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)	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 принятые USART0 (режим I <sup>2</sup> C) 0100 – UTXIFG0 (режим UART/SPI), готовность передачи USART0 (режим I <sup>2</sup> C) 0101 – Бит DAC12IFG DAC12_OCTL 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

<b>ROUNDRBIN</b>	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
<b>ENNMI</b>	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
<b>Зарезервировано</b>	<b>DMADTx</b>			<b>DMADSTINCRx</b>		<b>DMASRCINCRx</b>	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
<b>DMA DSTBYTE</b>	<b>DMA SRCBYTE</b>	<b>DMA LEVEL</b>	<b>DMAEN</b>	<b>DMAIFG</b>	<b>DMAIE</b>	<b>DMA ABORT</b>	<b>DMAREQ</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

<b>Зарезервировано</b>	Бит 15	Зарезервирован
<b>DMADTx</b>	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
<b>DMADSTINCRx</b>	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

<b>DMASRCINCRx</b>	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
<b>DMA DST BYTE</b>	Бит 7	Байт DMA получателя. Этот бит определяет формат получателя: байт или слово. 0 – Слово. 1 – Байт.
<b>DMASRCBYTE</b>	Бит 6	Байт DMA источника. Этот бит определяет формат источника: байт или слово. 0 – Слово. 1 – Байт.
<b>DMALEVEL</b>	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)
<b>DMAEN</b>	Бит 4	Разрешение DMA 0 – Запрещено 1 – Разрешено
<b>DMAIFG</b>	Бит 3	Флаг DMA прерывания 0 – Прерывание не ожидается 1 – Ожидается прерывание
<b>DMAIE</b>	Бит 2	Разрешение DMA прерывания 0 – Запрещено 1 – Разрешено
<b>DMAABORT</b>	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
<b>DMAREQ</b>	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
<b>DMAxSAx</b>							
rw	rw	rw	rw	rw	rw	rw	rw

7	6	5	4		3	2	1	0
<b>DMAxSAx</b>								
rw	rw	rw	rw		rw	rw	rw	rw

<b>DMAxSAx</b>	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
<b>DMAxDAx</b>								
rw	rw	rw	rw		rw	rw	rw	rw
<b>DMAxDAx</b>								
rw	rw	rw	rw		rw	rw	rw	rw

<b>DMAxDAx</b>	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
<b>DMAxSZx</b>								
rw	rw	rw	rw		rw	rw	rw	rw
<b>DMAxSZx</b>								
rw	rw	rw	rw		rw	rw	rw	rw

<b>DMAxSZx</b>	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 . .OffFFFh – Перенос 65535 байт или слов
----------------	-----------	--

# MSP430x4xxFamily

## Цифровые входы/выходы

*Раздел IX.*



## Цифровые входы/выходы

В этом разделе описывается работа портов цифровых входов/выходов. Порты P1-P6 имеются во всех устройствах MSP430x4xx.

### 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 – источником 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»**

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 значение

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 Rx или устанавливается после команды RETI выполняемой процедуры обработки прерывания Rx, 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	-
	Выход	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 )**

<b>Порт</b>	<b>Регистр</b>	<b>Краткое обозначение</b>	<b>Адрес</b>	<b>Тип регистра</b>	<b>Исходное состояние</b>
P2	Ввод	P2IN	028h	Только чтение	–
	Выход	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	–
	Выход	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	–
	Выход	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	–
	Выход	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	–
	Выход	P6OUT	035h	Чтение/запись	Не изменяется
	Направление	P6DIR	036h	Чтение/запись	Сброс с PUC
	Выбор порта	P6SEL	037h	Чтение/запись	Сброс с PUC

# MSP430x4xxFamily

## Сторожевой таймер WDT и WDT+

*Раздел X.*



## Сторожевой таймер

Сторожевой таймер – это 16-разрядный 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 MSP430x4xx. А устройства MSP430X42X и MSP430FE42X имеют расширенный WDT называемый WDT+.

### 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
- Высокая надежность WDT +

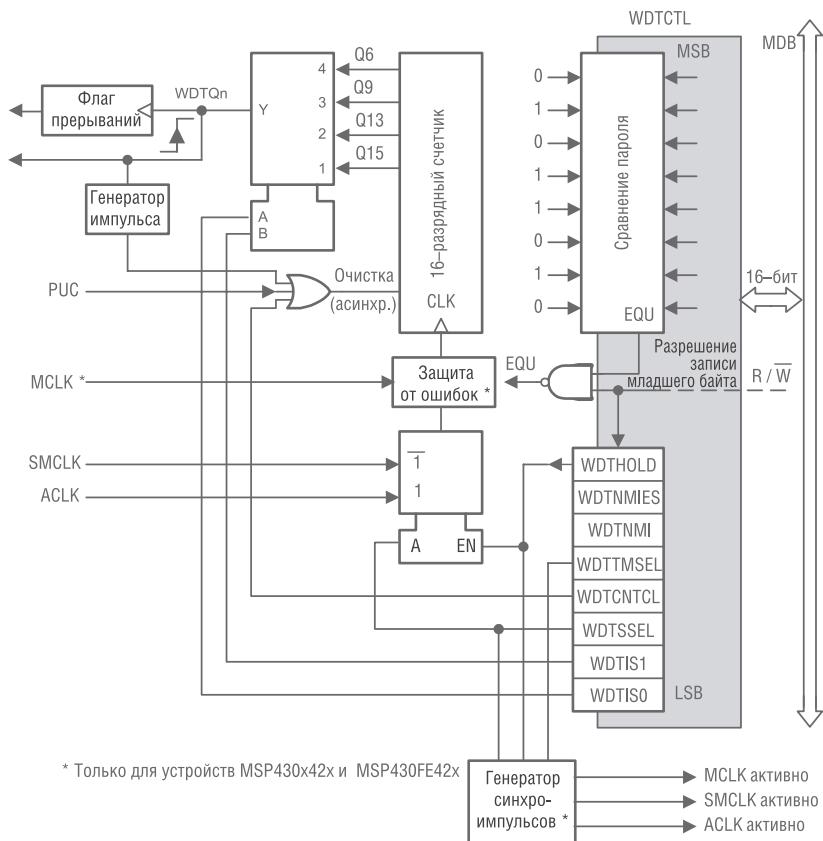
Блок схема модуля WDT показана на рис. 10-1.

**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 DCOCCLK. 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 долж-



**Рис. 10-1.** Блок-схема сторожевого таймера

на содержать пароль записи 05Ah в старшем байте. Любая запись в WDTCNTL любого значения, отличного от 05Ah в старшем байте приведет к нарушению ключа безопасности и запуску системного сброса PUC независимо от режима таймера. При любом чтении WDTCNTL 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 WDTCNTL.

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

### 10.2.2. Сторожевой режим

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

рой 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. Усовершенствованный сторожевой таймер WDT+**

Модуль WDT+ имеет ряд функциональных преимуществ в по сравнению с WDT. В данном модуле реализована функция защиты от сбоев тактирования, предотвращающая опасность отключения сигнала тактирования WDT+ во время работы его в режиме сторожевого таймера. Это означает, что режимы пониженного энергопотребления зависят находятся в зависимости от источника тактирования модуля WDT+. Например, если для модуля WDT+ используется сигнал тактирования ACLK, переход в режим LPM4 невозможен, т.к. WDT+ не разрешит отключение сигнала ACLK. Кроме того, если произойдет сбой сигналов ACLK или SMCLK, которые в данный момент используются для тактирования модуля WDT+, будет выполнено автоматическое переключение на сигнал MCLK. В этом случае, если произойдет сбой в кварцевом генераторе, который используется в качестве источника MCLK, функцией защиты от сбоев FLL+ будет выполнена активация DCO для тактирования модуля MCLK. Функция защиты от сбоев не работает в том случае, если модуль WDT+ используется в режиме таймера.

#### **10.2.6. 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 очистка 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+WDTISO,&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 PUC <sup>1)</sup>

1) WDTIFG сбрасывается с POR

### WDTCTL, регистр сторожевого таймера

15	14	13	12		11	10	9	8
Читается как 069h WDTPW, должен записываться как 05Ah								

7	6	5	4	3	2	1	0
WDTHOLD	WDTNMIES	WDTNMI	WDTTMSEL	WDTCNTCL	WDTSSSEL	WDTISX	
rw-0	rw-0	rw-0	rw-0	r0(w)	rw-0	rw-0	rw-0
<b>WDTPWx</b>	<b>Биты 15-8</b>	Пароль сторожевого таймера. Всегда читается как 069h. Должен записываться как 05Ah, в противном случае будет сгенерирован PUC.					
<b>WDTHOLD</b>	<b>Бит 7</b>	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					
<b>WDTNMIES</b>	<b>Бит 6</b>	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					
<b>WDTNMI</b>	<b>Бит 5</b>	Выбор NMI сторожевого таймера. Этот бит позволяет установить режим функционирования вывода nonRST/NMI. 0 – Функция сброса 1 – Функция NMI					
<b>WDTTMSEL</b>	<b>Бит 4</b>	Выбор режима сторожевого таймера 0 – Сторожевой режим 1 – Режим интервального таймера					
<b>WDTCNTCL</b>	<b>Бит 3</b>	Очистка счетчика сторожевого таймера. Установкой WDTCNTCL=1 производится очистка счетчика до значения 0000h. Бит WDTCNTCL автоматически сбрасывается. 0 – Действие не производится 1 – WDTCNT = 0000h					
<b>WDTSSSEL</b>	<b>Бит 2</b>	Выбор источника тактирования сторожевого таймера 0 – SMCLK 1 – ACLK					
<b>WDTISx</b>	<b>Биты 1-0</b>	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	NMIIE	3	2	1	0
					rw-0			rw-(0)

	<b>Биты 7-1</b>	Эти биты могут быть использованы другими модулями. См. справочное руководство конкретного устройства.
<b>NMIIE</b>	<b>Бит 4</b>	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
<b>WDTIE</b>	<b>Бит 0</b>	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

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

	<b>Биты 7-1</b>	Эти биты могут быть использованы другими модулями. См. справочное руководство конкретного устройства.
<b>NMIIFG</b>	<b>Бит 4</b>	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
<b>WDTIE</b>	<b>Бит 0</b>	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

# MSP430x4xxFamily

## Базовый таймер

*Раздел XI.*



## Базовый таймер

Модуль базового таймера состоит из двух независимых, каскадируемых 8-битных таймеров. В этой главе описана работа модуля базового таймера Basic Timer 1. Данный модуль присутствует во всех микроконтроллерах семейства MSP430x4xx.

### 11.1. Модуль базового таймера Basic Timer 1 – введение

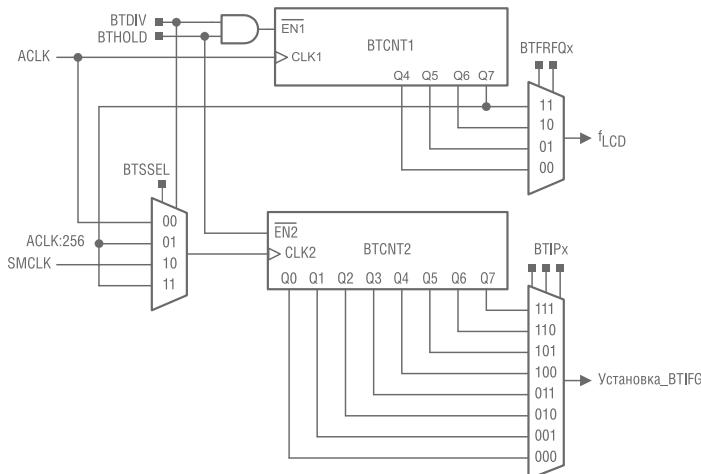
Модуль базового таймера обеспечивает тактирование ЖКИ и генерацию низкочастотных временных интервалов. Модуль базового таймера состоит из двух независимых 8-битных таймеров, которые могут быть каскадированы в один 16-битный таймер. Базовый таймер может быть использован в качестве:

- Часов реального времени;
- Программного таймера.

**Модуль базового таймера обладает следующими особенностями:**

- Выбор различных источников тактирования;
- Два независимых, каскадируемых 8-битных таймера;
- Возможность генерации прерываний;
- Генерация сигналов тактирования ЖКИ.

Блок схема модуля тактирования модуля базового таймера Basic Timer 1 показана на рис. 11-1.



**Рис. 11-1.** Блок схема модуля базового таймера Basic Timer 1

**Примечание: Инициализация базового таймера**

Регистры модуля базового таймера не имеют значений по умолчанию. Перед использованием обязательна программная инициализация.

## 11.2. Работа модуля базового таймера Basic Timer 1

Модуль базового таймера может быть сконфигурирован как два независимых 8-битных таймера, либо один 16-битный таймер при помощи 8-битного доступного для чтения и записи управляющего регистра BTCTL. Операции чтения/записи в данный регистр должны иметь байтовый формат. Модуль базового таймера управляет частотой ЖКИ при помощи счётчика BTCNT1.

### 11.2.1. Счётчик №1 модуля базового таймера Basic Timer 1

Счётчик №1 BTCNT1 модуля базового таймера Basic Timer1 является 8-битным счётчиком/таймером с непосредственным программным доступом. Тактовой частотой для него является вспомогательная частота тактирования ACLK, а на выходе формируется кадровая частота ЖКИ. Останов счётчика BTCNT1 осуществляется установкой в «1» бит BT HOLD и BT DIV.

### 11.2.2. Счётчик №2 модуля базового таймера Basic Timer1

Счётчик №2 BTCNT2 модуля базового таймера Basic Timer1 является 8-битным счётчиком/таймером с непосредственным программным доступом. Тактовой частотой для него является вспомогательная частота тактирования ACLK, дополнительная тактовая частота SMCLK или частота ACLK/256 в случае каскадирования со счётчиком BTCNT1. Выбор источника тактирования осуществляется битами BTSEL и BT DIV. Для снижения энергопотребления счётчик BTCNT2 может быть остановлен установкой в «1» бита HOLD. Счётчик BTCNT2 может служить источником прерывания базового таймера BTIFG. Интервал прерываний задаётся битами BTIPx.

**Примечание: Чтение и запись регистров BTCNT1 и BTCNT2**

В случае, когда тактовая частота процессора и частота тактирования счётчика не синхронны, чтение из регистров BTCNT1 и BTCNT2 может иметь неопределённый результат. Запись значения в регистры BTCNT1 и BTCNT2 имеет незамедлительное действие.

### 11.2.3. Режим 16-битного счётчика

16-битный режим счётчика выбирается установкой управляющего бита BT DIV в «1». В таком режиме счётчики BTCNT1 и BTCNT2 каскадируются. Источником тактирования для BTCNT является частота ACLK, а для BTCNT2 – соответственно ACLK/256.

### 11.2.4. Работа базового таймера Basic Timer1: сигнал $f_{LCD}$

Контроллер ЖКИ использует сигнал  $f_{LCD}$  от модуля BTCNT1 для формирования управляющих сигналов для строк и сегментов ЖКИ. Источником тактирования для BTCNT1 является частота ACLK, которую считаем равной 32768 Гц, из неё формируется частота  $f_{LCD}$ . Значение частоты  $f_{LCD}$  определяется битами BTFRFQx и может принимать значение ACLK/256, ACLK/128, ACLK/64, или ACLK/32. Требуемая частота  $f_{LCD}$  зависит от частоты обновления ЖКИ  $f_{Frame}$  и его степени мультилиплицирования и вычисляется как:

$$f_{LCD} = 2 \times \text{mux} \times f_{Frame}$$

Например, вычислим частоту  $f_{LCD}$  для ЖКИ с коэффициентом мультилиплицирования 3 (3-тих) с частотой обновления индикации  $f_{Frame}$  30 – 100 Гц:

$$f_{Frame} \text{ (из документации на ЖКИ)} = 30 - 100 \text{ Гц}$$

$$f_{LCD} = 2 \times 3 \times f_{Frame}$$

$$f_{LCD}(\min) = 180 \text{ Гц}$$

$$f_{LCD}(\max) = 600 \text{ Гц}$$

$$\text{выбираем } f_{LCD} = 32768/128 = 256 \text{ Гц или } 32768/64 = 512 \text{ Гц}$$

См. также раздел Контроллер ЖКИ (LCD Controller), где приведена дополнительная информация по контроллеру ЖКИ.

### 11.2.5. Прерывания базового таймера Basic Timer1

Модуль базового таймера использует два бита в регистрах специального назначения SFR для управления прерываниями.

- Флаг прерывания модуля базового таймера Basic Timer1, BTIFG, расположенный в IFG 2.7
- Бит разрешения прерывания модуля базового таймера Basic Timer1, BTIE, расположенный в IE 2.7

Флаг прерывания BTIFG устанавливается по истечении заданного интервала времени и вызывает прерывание базового таймера Basic Timer1 в том случае, если это разрешено битами BTIE и GIE. Флаг BTIFG сбрасывается автоматически после обработки прерывания либо может быть очищен программно.

## 11.3. Регистры базового таймера Basic Timer1

Регистры базового таймера Basic Timer1 перечислены в таблице 11-1.

Регистр	Краткое название	Тип	Адрес	Начальное состояние
Контроль базового таймера Basic Timer1	BTCTL	Чтение/запись	040h	Не изменяется

Регистр	Краткое название	Тип	Адрес	Начальное состояние
Счётчик №1 модуля базового таймера Basic Timer1	BTCNT1	Чтение/запись	046h	046h
Счётчик №2 модуля базового таймера Basic Timer1	BTCNT2	Чтение/запись	047h	Не изменяется
Регистр флагов прерываний 1	IFG2	Чтение/запись	IFG2	Обнулён по сбросу PUC
Регистр разрешения прерываний 1	IE2	Чтение/запись	003h	Обнулён по сбросу PUC

Примечание: Регистры модуля базового таймера BTCTL, BTCNT1, BTCNT2 не имеют значений по умолчанию. Перед использованием обязательна программная инициализация.

### BTCTL, Контроль базового таймера Basic Timer1

7	6	5	4	3	2	1	0							
<b>BTSSEL</b>	<b>BTHold</b>	<b>BTDIV</b>	<b>BTFRFQx</b>			<b>BTIPx</b>								
rw	rw	rw	rw	rw	rw	rw	rw							
<b>BTSSEL</b>		Бит 7	Выбор источника тактирования счётчика BTCNT2. Совместно с битом BTDIV определяет тактовую частоту счётчика BTCNT2. См. определение BTDIV.											
<b>BTHold</b>		Бит 6	Останов базового таймера. 0 – BTCNT1 и BTCNT2 функционируют 1 – BTCNT1 остановлен, в случае, если BTDIV=1 BTCNT2 остановлен											
<b>BTDIV</b>	Бит 5	<b>BTSSEL</b>	<b>BTDIV</b>	<b>Источник тактирования BTCNT2</b>										
		0	0	ACLK										
		0	1	ACLK/256										
		1	0	SMCLK										
		1	1	ACLK/256										
<b>BTFRFQx</b>		Биты 4-3	Выбор частоты $f_{LCD}$ . Эти биты определяют частоту обновления индикации ЖКИ. 00 – $f_{ACLK}/32$ 01 – $f_{ACLK}/64$ 10 – $f_{ACLK}/128$ 11 – $f_{ACLK}/256$											
<b>BTIPx</b>		Биты 2-0	Интервал прерываний базового таймера Basic Timer1. 000 – $f_{CLK2}/2$ 001 – $f_{CLK2}/4$ 010 – $f_{CLK2}/8$ 011 – $f_{CLK2}/16$ 100 – $f_{CLK2}/32$ 101 – $f_{CLK2}/64$ 110 – $f_{CLK2}/128$ 111 – $f_{CLK2}/256$											

### **BTCNT1, Счётчик №1 модуля базового таймера Basic Timer1**

7	6	5	4		3	2	1	0
<b>BTCNT1x</b>								

<b>BTCNT1x</b>	Биты 7-0	Регистр BTCNT1. Значение в регистре BTCNT1 представляет собой выход счётчика BTCNT1.
----------------	----------	--

### **BTCNT2, Счётчик №2 модуля базового таймера Basic Timer1**

7	6	5	4		3	2	1	0
<b>BTCNT2x</b>								

<b>BTCNT2x</b>	Биты 7-0	Регистр BTCNT2. Значение в регистре BTCNT2 представляет собой выход счётчика BTCNT2.
----------------	----------	--

### **IE2, Регистр разрешения прерываний 2**

7	6	5	4		3	2	1	0
<b>BTIE</b>								

<b>BTIE</b>	Бит 7	Разрешение прерывания базового таймера. Этот бит разрешает прерывание по флагу BTIFG. Так как другие биты в регистре IE2 могут использоваться другими модулями, рекомендуется устанавливать и сбрасывать данный бит используя инструкции BIS.B или BIC.B, а не MOV.B или CLR.B. 0 – прерывание запрещено 1 – прерывание разрешено
	Биты 6-0	Эти биты могут быть использованы другими модулями. См. документацию на конкретный МК

### **IFG2, Регистр флагов прерываний 2**

7	6	5	4		3	2	1	0
<b>BTIFG</b>								

<b>BTIFG</b>	Бит 7	Флаг прерывания базового таймера. Так как другие биты в регистре IE1 могут использоваться другими модулями, рекомендуется устанавливать и сбрасывать данный бит используя инструкции BIS.B или BIC.B, а не MOV.B или CLR.B. 0 – нет запроса прерывания 1 – есть запрос прерывания
	Биты 6-0	Эти биты могут быть использованы другими модулями. См. документацию на конкретный МК

# MSP430x4xxFamily

**Таймер А**

---

*Раздел XII.*



## Таймер А

Таймер А – это 16-разрядный таймер/счетчик с несколькими регистрами захвата/сравнения. В этом разделе описывается Таймер А3 (три регистра захвата/сравнения) реализованный во всех устройствах MSP430x4xx и Таймер1 А5 (пять регистров захвата/сравнения) так же реализованы в устройствах MSP430x415, MSP430x417 и MSP430xW42x.

### 12.1. Введение в таймер А

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

**Таймер А обладает следующими возможностями:**

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

Блок-схема таймера А 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. Если в счетчик напрямую записывается конкретное значение, соответствующее действие не происходит.

**Примечание: Второй Таймер А**

В приборах MSP430x415, MSP430x417, and MSP430xW42x реализован второй Таймер А с пятью регистрами захвата/сравнения. В этих приборах оба модуля Таймера А функционируют аналогично, за исключением дополнительных регистров захвата/сравнения.

### 12.2. Функционирование таймера А

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

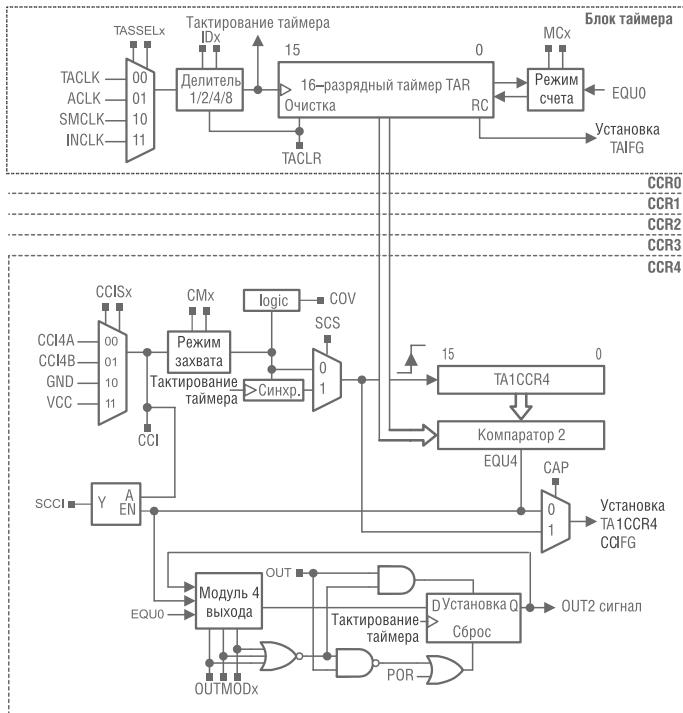


Рис. 12-1. Блок-схема таймера А

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

#### Примечание: Изменение регистров таймера А

Рекомендуется останавливать таймер перед изменением режима его работы (за исключением операций с флагом прерывания и разрешения прерывания, и TA CLR), чтобы предотвратить его некорректную работу. Когда сигнал TACLK асинхронен с сигналом тактирования ЦПУ, любое чтение из TAR должно проходить, 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.

### 12.2.2. 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.

### 12.2.3. Управление режимом таймера

Таймер имеет четыре режима работы, описанных в таблице 12-1: «стоп», «вверх», «непрерывный» и «вверх/вниз». Режимы работы выбираются 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 TACCR0
10	Непрерывный	Таймер многократно считает от нуля до значения OFFFFh
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 OFFFFh. Таймер многократно считает вверх до значения, содержащегося в регистре сравнения TACCR0, который задает период, как показано на рис.12-2. 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.

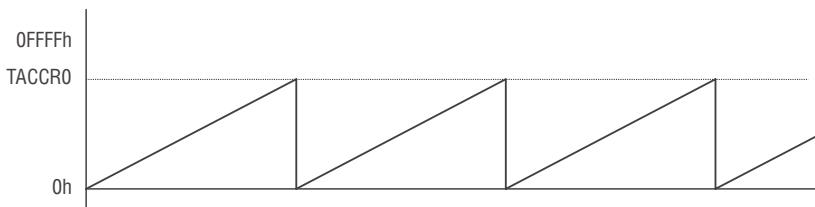


Рис. 12-2. Режим «Вверх»

Флаг прерывания TACCR0 CCIFG устанавливается, когда значение таймера равно содержимому TACCR0. Флаг прерывания TAIFG устанавливается, когда таймер считает от TACCR0 к нулю. На рис. 12-3 показан цикл установки флагов.

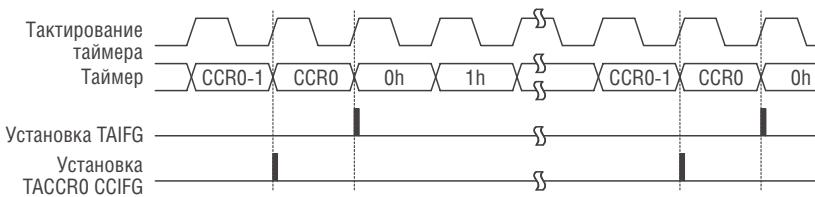


Рис. 12-3. Установка флагов в режиме «Вверх»

### 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 OFFFFh и перезапускается от нуля, как показано на рис. 12-4. Регистр захвата/сравнения TACCR0 работает подобно другим регистрам захвата/сравнения.

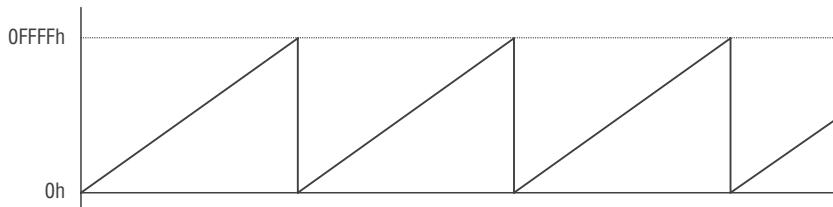
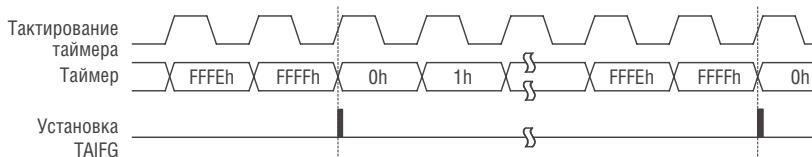


Рис. 12-4. Непрерывный режим

Флаг прерывания TAIFG устанавливается, когда таймер считает от 0FFFFh к нулю. На рис. 12-5 показан цикл установки флага.

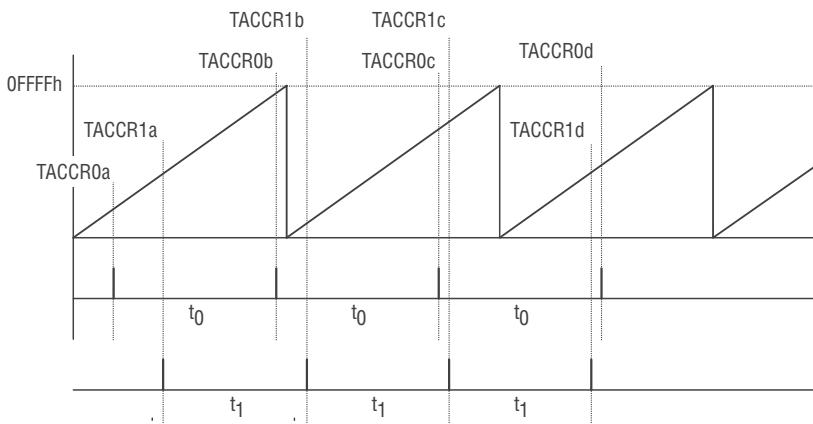


**Рис. 12-5.** Установка флага в непрерывном режиме

### 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. 12-6 показаны два раздельных временных интервала  $t_0$  и  $t_1$ , добавляемые к регистрам захвата/сравнения. При таком использовании временные интервалы управляются аппаратно, без программного обеспечения, без влияния времени задержки прерывания. При использовании всех трех (Таймер A3) или пяти (Таймер A5) регистров захвата/сравнения можно генерировать независимые временные интервалы или выходные частоты.

Временные интервалы можно реализовать также в других режимах, где TACCR0 используется как регистр периода. Их обработка более комплексна,

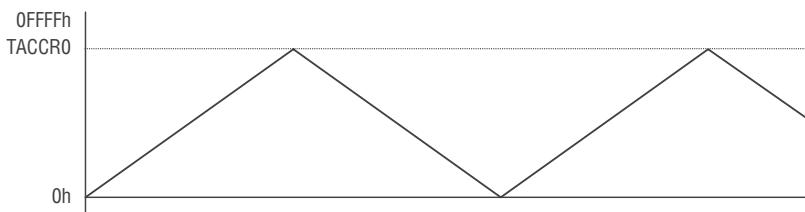


**Рис. 12-6.** Временные интервалы непрерывного режима

поскольку сумма старого значения TACCRx и нового периода может быть выше значения TACCR0. Когда предыдущее значение TACCRx плюс tx больше величины TACCR0, значение TACCR0 должно быть вычтено для получения правильно-го временного интервала.

### Режим «вверх/вниз»

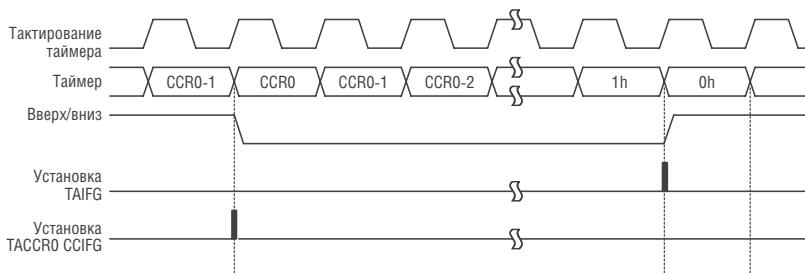
Режим «вверх/вниз» используется, если период таймера должен отличаться от значения OFFFFh и необходима генерация симметричных импульсов. Таймер непрерывно считает вверх до значения, находящегося в регистре срав-нения TACCR0 и назад к нулю, как показано на рис. 12-7. Период составляет удвоенную величину значения в TACCR0.



**Рис. 12-7.** 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 TACLK. Бит TACLK 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 счи-тает от TACCR0-1 к TACCR0, а флаг TAIFG устанавливается, когда таймер завершает счет вниз от 0001h к 0000h. На рис. 12-8 показан цикл установки флагов.



**Рис. 12-8.** 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, управляемые H-мостом никогда не должны одновременно иметь высокий уровень. В примере, показанном на рис. 12-9 величина времени простоя  $t_{dead}$  составляет:

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

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

$t_{timer}$  – время цикла тактирования таймера

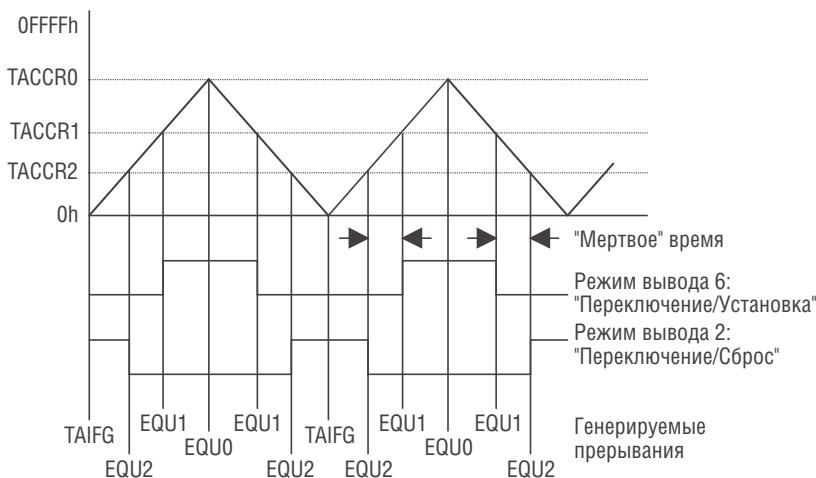


Рис. 12-9. Модуль вывода в режиме «вверх/вниз»

TACCRx – содержимое регистра захвата/сравнения х

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

#### 12.2.4. Блоки захвата/сравнения

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

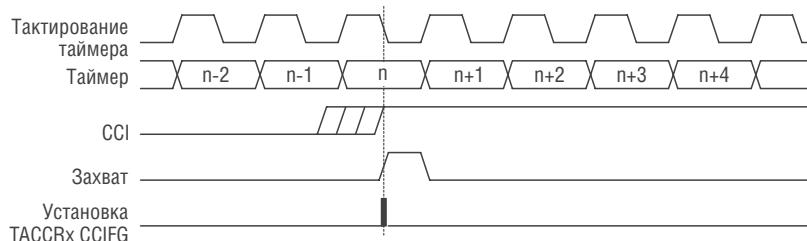
#### Режим захвата

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 происходить 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 MSP430x4xx 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. 12-10.



**Рис. 12-10.** Сигнал захвата (SCS=1)

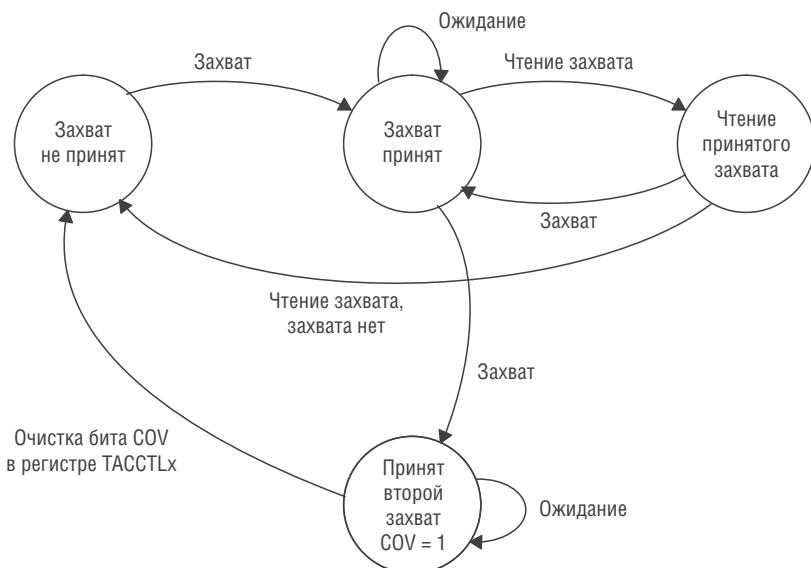


Рис. 12-11. Цикл захвата

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. 12-11. Бит COV должен сбрасываться программно.

### Иницирование захвата программным обеспечением

Захваты могут быть иницированы 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  $V_{cc}$  и GND, инициируя захват каждый раз, когда CCISO изменяет состояние:

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

### Примечание: Разногласие с официальной документацией

Здесь в описании что-то не то:

Перевод. Раздел «Таймер A», подраздел

«Иницирование захвата программным обеспечением».

Читаем:

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  $V_{cc}$  и GND, инициируя захват каждый раз, когда CCISO изменяет состояние...

Но бит CCI *read only!* Значит, софт его поставить не может. (Да и правильно, иначе есть опасность столкновения сигналов, см. Блок-схему).

Но нужно не CCI ставить, а CCIS1.

Об этом говорит и рядом расположенный ассемблерный пример, и аналогичный фрагмент в описании Таймера B. Там не CCI ставится, а CCIS1.

### Режим сравнения

Режим сравнения выбирается, когда CAP=0. Режим сравнения используеться 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 действует 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 12-2. 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 12-2. Режимы вывода.**

OUTMODx	Режим	Описание
000	Вывод	Выходной сигнал OUTx определяется битом OUTx. Сигнал OUTx изменяется немедленно при изменении OUTx.

**Таблица 12-2. (Окончание)**

<b>OUTMODx</b>	<b>Режим</b>	<b>Описание</b>
<b>001</b>	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.
<b>010</b>	Переключение/Сброс	Выход переключается, когда таймер досчитывает до значения TACCRx. Он сбрасывается, когда таймер досчитывает до значения TACCR0.
<b>011</b>	Установка/Сброс	Выход устанавливается, когда таймер досчитывает до значения TACCRx. Он сбрасывается, когда таймер досчитывает до значения TACCR0.
<b>100</b>	Переключение	Выход переключается, когда таймер досчитывает до значения TACCRx. Период выходного сигнала равен удвоенному периоду таймера.
<b>101</b>	Сброс	Выход сбрасывается, когда таймер досчитывает до значения TACCRx. Это остается сброшенным до выбора другого режима вывода и воздействия на выход.
<b>110</b>	Переключение/Установка	Выход переключается, когда таймер досчитывает до значения TACCRx. Он устанавливается, когда таймер досчитывает до значения TACCR0.
<b>111</b>	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. 12-12.

#### 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. 12-13.

#### 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. 12-14.

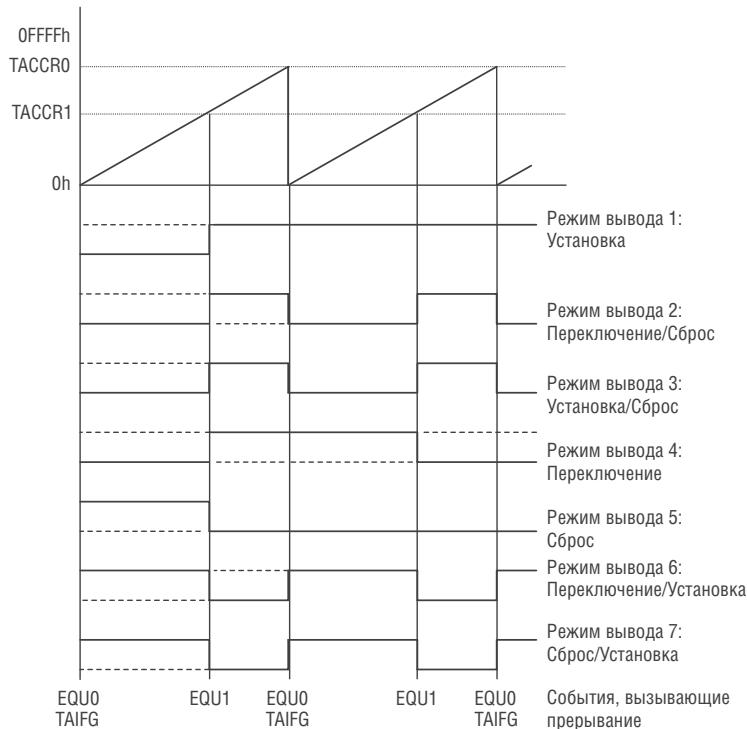


Рис. 12-12. Пример вывода – таймер в режиме «вверх»

#### Примечание: Переключение между режимами вывода

При переключении между режимами вывода один из битов OUTMODx должен оставаться установленным во время перехода, кроме перехода 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, &TACCTLx      ; Установка режима вывода 7
BIC #OUTMODx, &TACCTLx       ; Очистка лишних битов
```

#### 11.2.6. Прерывания Таймера А

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

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

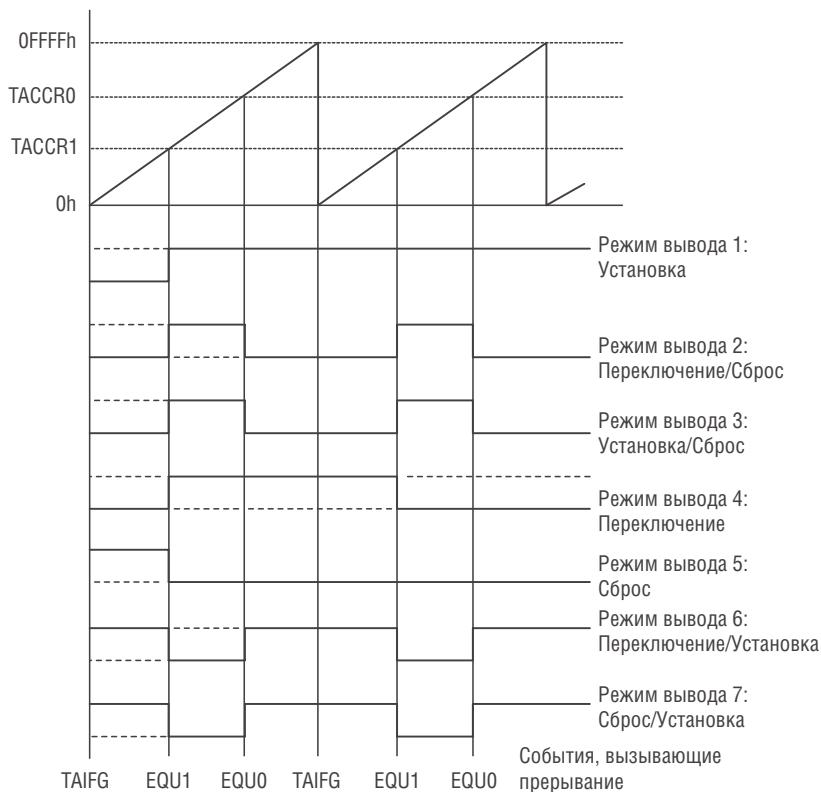


Рис. 12-13. Пример вывода – таймер в непрерывном режиме

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 соответствующего 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 А и имеет специализированный вектор прерывания, как показано на рис. 12-15. Флаг TACCR0 CCIFG автоматически сбрасывается, когда обслуживается запрос на прерывание TACCR0.

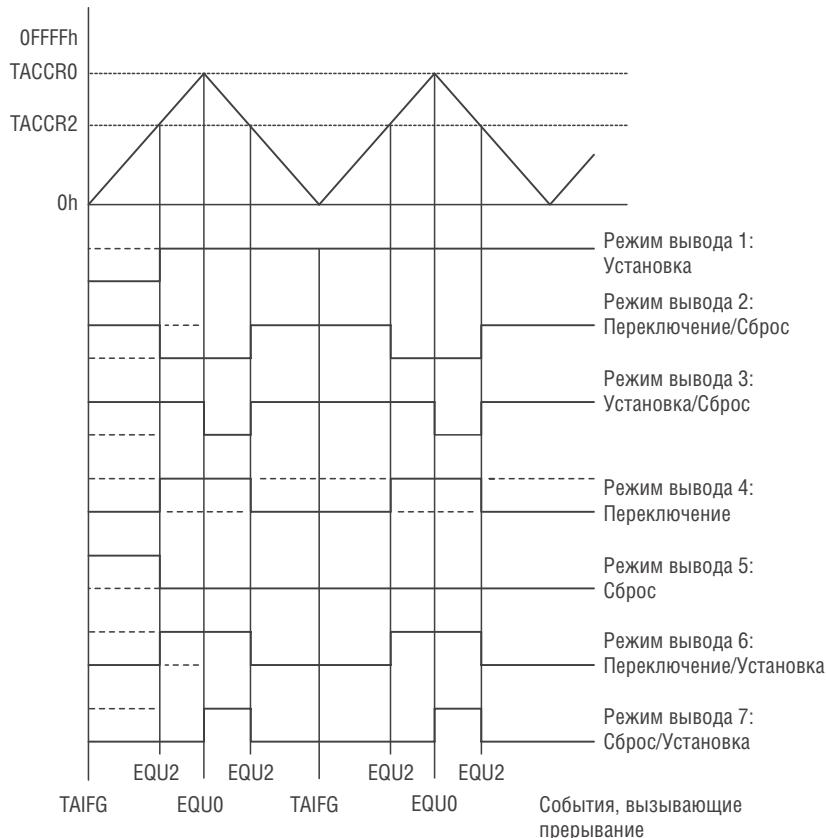


Рис. 12-14. Пример вывода – таймер в режиме «вверх/вниз»

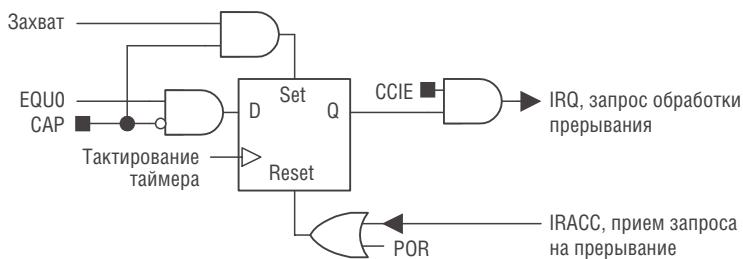


Рис. 12-15. Флаг прерывания захвата/сравнения TACCR0

## 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 А 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 РС 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. CCIFG_0_HND	Циклы
; ... ; 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 А

Перечень регистров Таймера А приведен в таблицах 12.3 и 12.4.

Таблица 12-3. Регистры Таймера А3

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Управление Таймером А	TACTL/	Чтение/запись	0160h	Сброс с POR
Управление Таймером А3	TAOCTL			
Счетчик Таймера А	TAR/	Чтение/запись	0170h	Сброс с POR
Счетчик Таймера А3	TAOR			
Регистр 0 управления захватом/сравнением Таймера А	TACCTL0/	Чтение/запись	0162h	Сброс с POR
Регистр 0 управления захватом/сравнением Таймера А3	TAOCCTL			
Регистр 0 захвата/сравнения Таймера А	TACCR0/	Чтение/запись	0172h	Сброс с POR
Регистр 0 захвата/сравнения Таймера А3	TAOCCR0			
Регистр 1 управления захватом/сравнением Таймера А	TACCTL1/	Чтение/запись	0164h	Сброс с POR
Регистр 1 управления захватом/сравнением Таймера А3	TAOCCTL1			
Регистр 1 захвата/сравнения Таймера А	TACCR1/	Чтение/запись	0174h	Сброс с POR
Регистр 1 захвата/сравнения Таймера А3	TAOCCR1			

**Таблица 12-3. (Окончание)**

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр 2 управления захватом/сравнением Таймера А Регистр 2 управления захватом/сравнением Таймера0 А3	TACCTL2/ TAOCCTL2	Чтение/запись	0166h	Сброс с POR
Регистр 2 захвата/сравнения Таймера А Регистр 2 захвата/сравнения Таймера0 А3	TACCR2/ TAOCCR2	Чтение/запись	0176h	Сброс с POR
Вектор прерывания Таймера А Вектор прерывания Таймера0 А3	TAIV/ TA0IV	Только чтение	012Eh	Сброс с POR

**Таблица 12-4. Регистры Таймера1 A5**

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Управление Таймером1 A5	TA1CTL	Чтение/запись	0180h	Сброс с POR
Счетчик Таймера1 A5	TA1R	Чтение/запись	0190h	Сброс с POR
Регистр 0 управления захватом/сравнением Таймера1 A5	TA1CCTL0	Чтение/запись	0182h	Сброс с POR
Регистр 0 захвата/сравнения Таймера1 A5	TA1CCR0	Чтение/запись	0192h	Сброс с POR
Регистр 1 управления захватом/сравнением Таймера1 A5	TA1CCTL1	Чтение/запись	0184h	Сброс с POR
Регистр 1 захвата/сравнения Таймера1 A5	TA1CCR1	Чтение/запись	0194h	Сброс с POR
Регистр 2 управления захватом/сравнением Таймера1 A5	TA1CCTL2	Чтение/запись	0186h	Сброс с POR
Регистр 2 захвата/сравнения Таймера1 A5	TA1CCR2	Чтение/запись	0196h	Сброс с POR
Регистр 3 управления захватом/сравнением Таймера1 A5	TA1CCTL3	Чтение/запись	0188h	Сброс с POR
Регистр 3 захвата/сравнения Таймера1 A5	TA1CCR3	Чтение/запись	0198h	Сброс с POR
Регистр 4 управления захватом/сравнением Таймера1 A5	TA1CCTL4	Чтение/запись	018Ah	Сброс с POR
Регистр 4 захвата/сравнения Таймера1 A5	TA1CCR4	Чтение/запись	019Ah	Сброс с POR
Вектор прерывания Таймера1 A5	TA1IV	Только чтение	011Eh	Сброс с POR

**TACTL, регистр управления Таймером А**

15	14	13	12		11	10	9	8
<b>Не используется</b>							<b>TASSELx</b>	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4		3	2	1	0
<b>IDx</b>		<b>MCx</b>			<b>Не используется</b>	<b>TACLR</b>	<b>TAIE</b>	<b>TAIFG</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	w-(0)	rw-(0)	rw-(0)	rw-(0)

<b>Не используются</b>	<b>Биты 15-10</b>	Не используются
<b>TASSELx</b>	<b>Биты 9-8</b>	Выбор источника тактирования Таймера А 00 – TACLK 01 – ACLK 10 – SMCLK 11 – инвертированное значение TACLK
<b>IDx</b>	<b>Биты 7-6</b>	Входной делитель. Эти биты позволяют выбрать коэффициент деления для входной тактовой частоты. 00 – /1 01 – /2 10 – /4 11 – /8
<b>MCx</b>	<b>Биты 5-4</b>	Выбор режима. Установка MCx=00h, когда Таймер А не используется, позволяет уменьшить потребляемую мощность. 00 – Режим остановки: 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
<b>Не используется</b>	<b>Бит 3</b>	Не используется
<b>TACLR</b>	<b>Бит 2</b>	Очистка Таймера А. Установка этого бита сбрасывает TAR, IDx и выбранное направление счета. Бит TACLR автоматически сбрасывается и всегда читается как нуль.
<b>TAIE</b>	<b>Бит 1</b>	Разрешение прерывания от Таймера А. Этот бит разрешает запрос прерывания TAIFG. 0 – Запрещение прерывания 1 – Разрешение прерывания
<b>TAIFG</b>	<b>Бит 0</b>	Флаг прерывания Таймера А 0 – Прерывание не ожидается 1 – Ожидается прерывание

## TAR, регистр Таймера А

15	14	13	12		11	10	9	8
<b>TARx</b>								
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4		3	2	1	0
<b>TARx</b>								
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
<b>TARx</b>	<b>Биты 15-0</b>	Регистр Таймера А. Регистр TAR является счетчиком Таймера А.						

## TACCTLx, регистр управления захватом/сравнением

15	14	13	12		11	10	9	8
<b>CMx</b>	<b>CCISx</b>			<b>SCS</b>	<b>SCCI</b>	<b>Не использу- ется</b>	<b>CAP</b>	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4		3	2	1	0
<b>OUTMODx</b>			<b>CCIE</b>	<b>CCI</b>	<b>OUT</b>	<b>COV</b>	<b>CCIFG</b>	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)	rw-(0)
<b>CMx</b>	<b>Биты 15-14</b>	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						
<b>CCISx</b>	<b>Биты 13-12</b>	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						
<b>SCS</b>	<b>Бит 1</b>	Синхронизация источника захвата. Этот бит используется для синхронизации входного сигнала захвата с тактовым сигналом таймера. 0 – Асинхронный захват 1 – Синхронный захват						
<b>SCCI</b>	<b>Бит 10</b>	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 режим 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 А**

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 А	
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	Захват/сравнение 3*	-	
08h	Захват/сравнение 4*	-	
0Ah	Переполнение таймера	TAIFG	
0Ch	Зарезервировано	-	
0Eh	Зарезервировано	-	Низший

\* Только у Таймера1 A5

# MSP430x4xxFamily

Таймер В

*Раздел XIII.*



## Таймер В

Таймер В – это 16-разрядный таймер/счетчик с несколькими регистрами захвата/сравнения. В этом разделе описывается работа таймера В. Таймер В3 (с тремя регистрами захвата/сравнения) реализован в устройствах MSP430x43x. Таймер В7 (с семью регистрами захвата/сравнения) реализован в устройствах MSP430x44x.

### 13.1. Введение в таймер В

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

**Таймер В 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 В

Блок-схема таймера В показана на рис. 13-1.

**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, соответствующее действие не происходит.

#### 13.1.1. Сходства и различия с таймером A

**Таймер В идентичен таймеру A, но со следующими исключениями:**

- Длина таймера В программируется и может составлять 8, 10, 12 или 16 бит
- Регистры TBCCR<sub>x</sub> таймера В имеют двойную буферизацию и могут группироваться

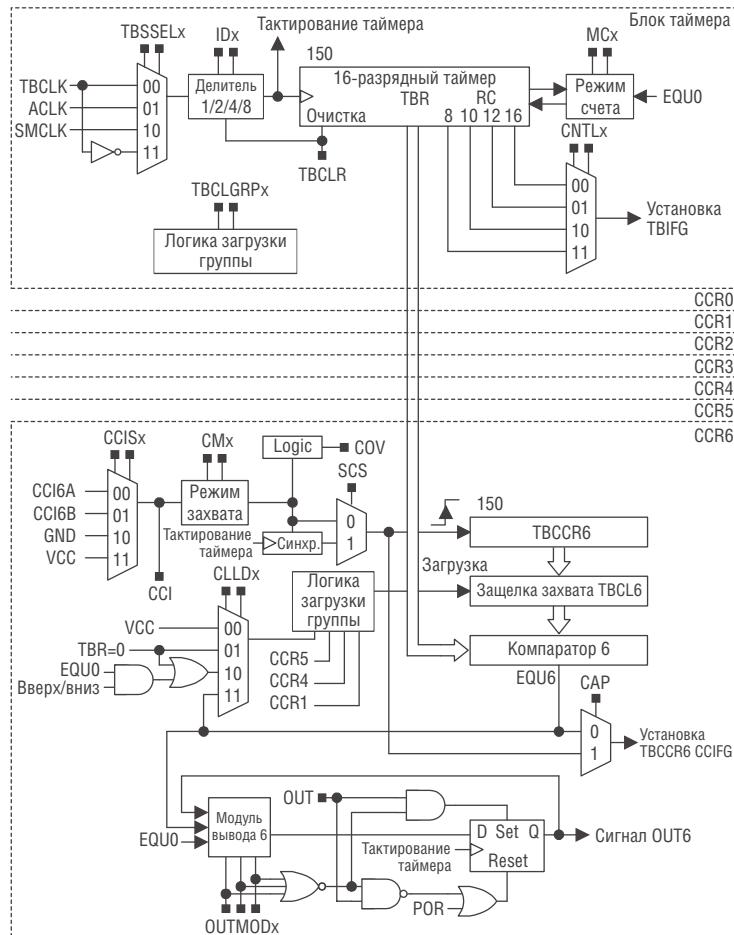


Рис. 13-1. Блок-схема таймера В

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

## 13.2. Функционирование таймера В

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

### 13.2.1. 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 В

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 В 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 OFFh, 03FFh, OFFFh и OFFFFh. 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.

### 13.2.2. 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.

### 13.2.3. Управление режимом таймера

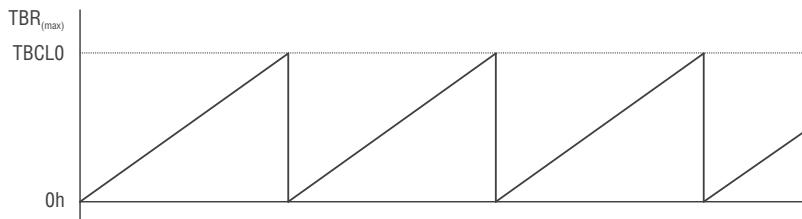
Таймер имеет четыре режима работы, описанных в таблице 13-1: «стоп», «вверх», «непрерывный» и «вверх/вниз». Рабочий режим выбирается с помощью битов MCx.

**Таблица 13-1. 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. 13-2. 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.

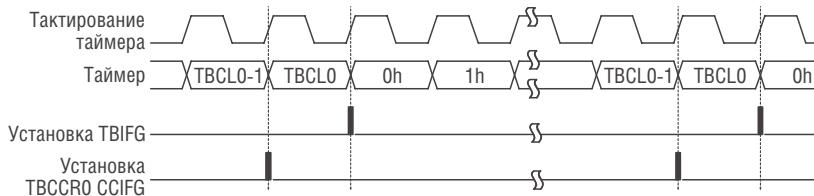


**Рис. 13-2. Режим «вверх»**

Флаг прерывания TBCCR0 CCIFG устанавливается, когда значение таймера равно значению TBCL0. Флаг прерывания TBIFG устанавливается, когда таймер пересчитывает от TBCL0 к нулю. На рис. 13-3 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

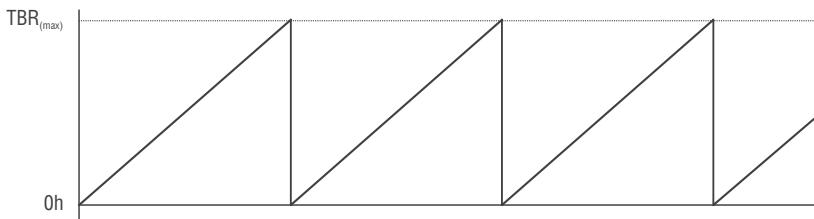


**Рис. 13-3.** Установка флагов в режиме «вверх»

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.

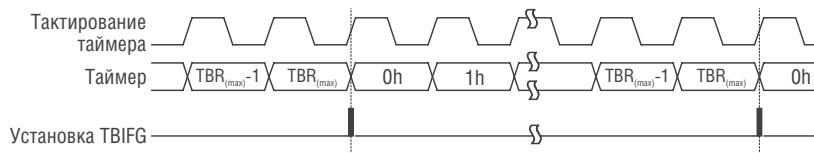
### Непрерывный режим

В непрерывном режиме таймер многократно считает вверх до значения  $TBR_{(max)}$  и перезапускается от нуля, как показано на рис. 13-4. Защелка сравнения TBCLO работает подобно другим регистрам захвата/сравнения.



**Рис. 13-4.** Непрерывный режим

Флаг прерывания TBIFG устанавливается, когда таймер считает от  $TBR_{(max)}$  к нулю. На рис. 13-5 показан цикл установки флагов.

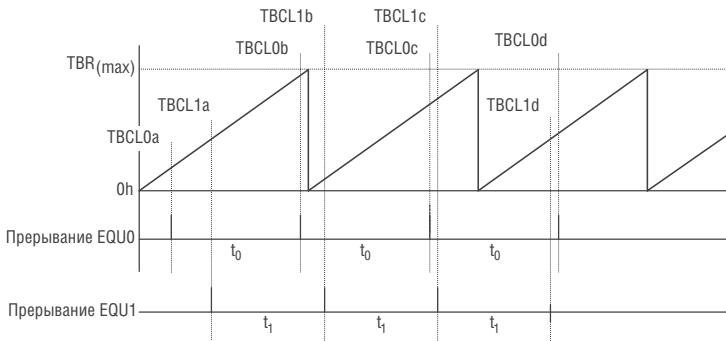


**Рис. 13-5.** Установка флагов в непрерывном режиме

### 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 TBCl<sub>x</sub> в процедуре обработки прерывания. На рис. 13-6 показаны два раздельных временных интервала t<sub>0</sub> и t<sub>1</sub>, 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. 13-6.** 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 вычесть 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. 13-7. Период в этом случае равен удвоенному значению TBCL0.

**Примечание:** TBCL0 > TBR(max)

Если  $TBCLO > TBR(max)$ , 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 остановки.

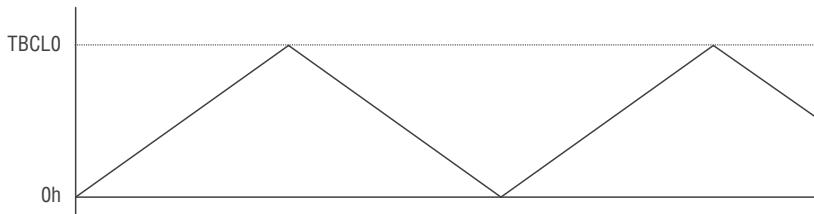


Рис. 13-7. Режим «вверх/вниз»

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. 13-8 показан цикл установки флагов.

#### Изменение значение регистра периода TBCL0

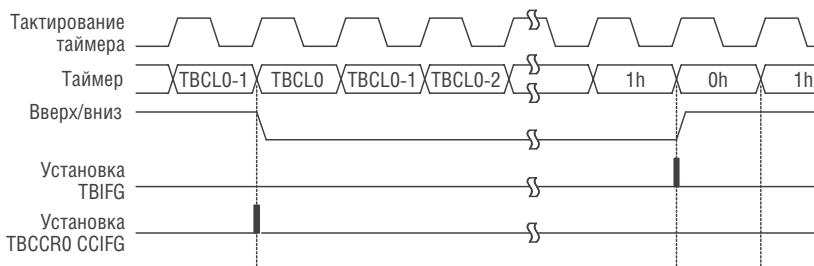


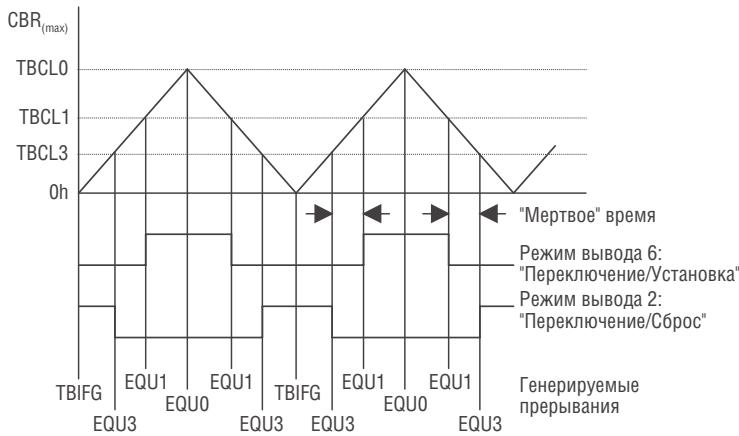
Рис. 13-8. Установка флагов в режиме «вверх/вниз»

Когда изменяется TBCL0 во время работы таймера при выбранном направлении счета вниз и установленном непосредственном режиме загрузке TBCL0, таймер продолжает отсчет вниз до нуля. Новый период будет активизирован после завершения 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 В). К примеру, чтобы избежать режима перегрузки, два выхода, управляемые Н-мостом никогда не должны иметь высокий уровень одновременно. В примере, показанном на рис. 13-9 «мертвое» время  $t_{dead}$  задается так:



**Рис. 13-9.** Модуль вывода в режиме «вверх/вниз»

$$t_{dead} = t_{timer} \times (TBCL1 - TBCL3), \text{ где:}$$

$t_{dead}$  — 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.

#### 13.2.4. Блоки захвата/сравнения

Три или семь идентичных блоков захвата/сравнения TBCCRx представлены в таймере В. Любой из блоков может использоваться для захвата данных таймера или для генерации временных интервалов.

#### Режим захвата

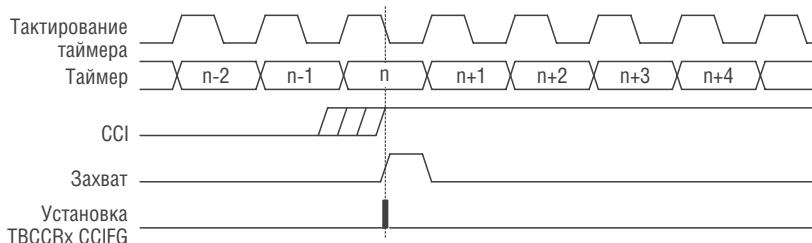
Режим захвата выбирается, когда CAP=1. Режим захвата используется для регистрации временных событий. Он может быть использован для выполнения быстрых вычислений или измерений времени. Входы захвата CCIAx и CCIBx под-

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 TBCCR<sub>x</sub>
- 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. 13-10.



**Рис. 13-10.** 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. 13-11. 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 CCISO 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 #CCISO, &TBCCTLx ; TBCCTLx = TBR
```

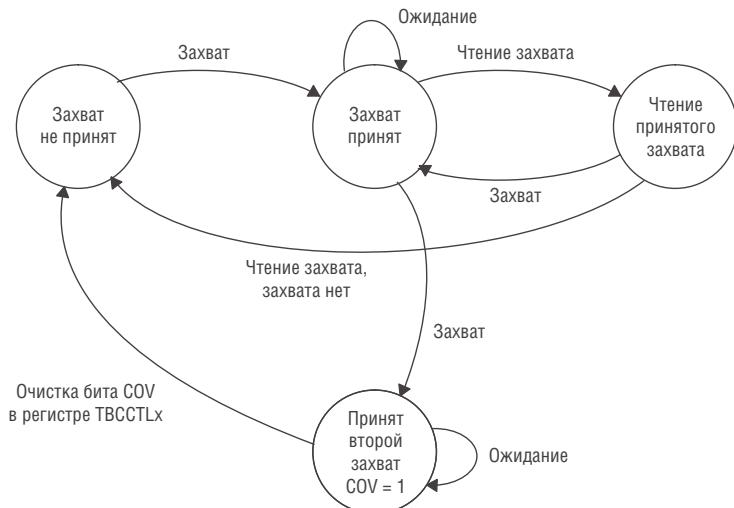


Рис. 12-11. Цикл захвата

### 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 13-2.

Таблица 13-2. 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.

<b>CLLDx</b>	<b>Описание</b>
<b>10</b>	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».
<b>11</b>	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 13-3. 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 13-3. Рабочие режимы защелок сравнения**

<b>TBCLGRPx</b>	<b>Группировка</b>	<b>Управление обновлением</b>
<b>00</b>	Нет	Индивидуальное
<b>01</b>	TBCL1 + TBCL2 TBCL3 + TBCL4 TBCL5 + TBCL6	TBCCR1 TBCCR3 TBCCR5
<b>10</b>	TBCL1 + TBCL2 + TBCL3 TBCL4 + TBCL5 + TBCL6	TBCCR1 TBCCR4
<b>11</b>	TBCL0 + TBCL1 + TBCL2 + TBCL3 + TBCL4 + TBCL5 + TBCL6	TBCCR1

### 13.2.5. 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 13-4. 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 13-4. Режимы вывода.**

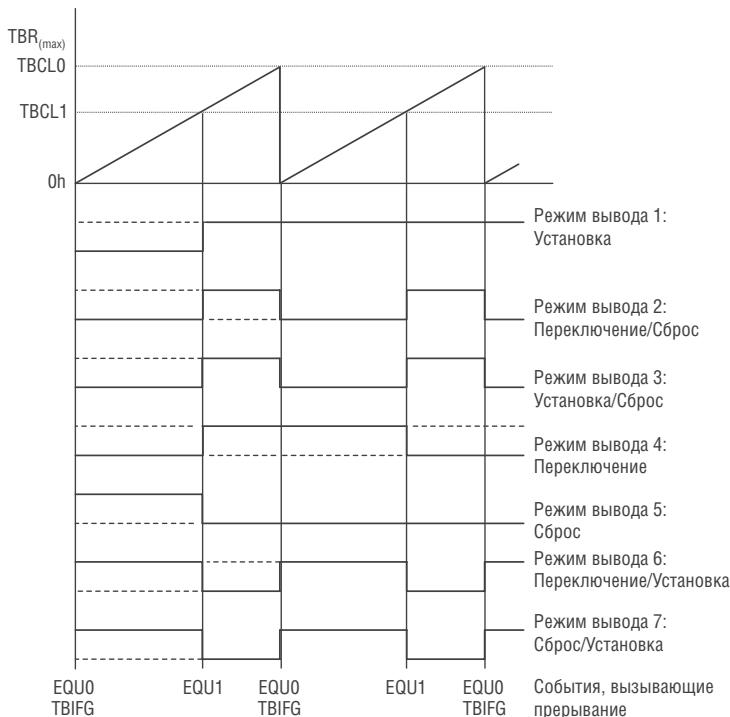
OUTMODx	Режим	Описание
000	Вывод	Выходной сигнал OUTx определяется битом OUTx. Сигнал OUTx изменяется немедленно при изменении OUTx.
001	Установка	Выход устанавливается, когда таймер досчитывает до значения TBCLx. Он остается установленным до сброса таймера или до выбора другого режима вывода и воздействия 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. 13-12.

### 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. 13-13.



**Рис. 13-12.** Пример вывода – таймер в режиме «вверх»

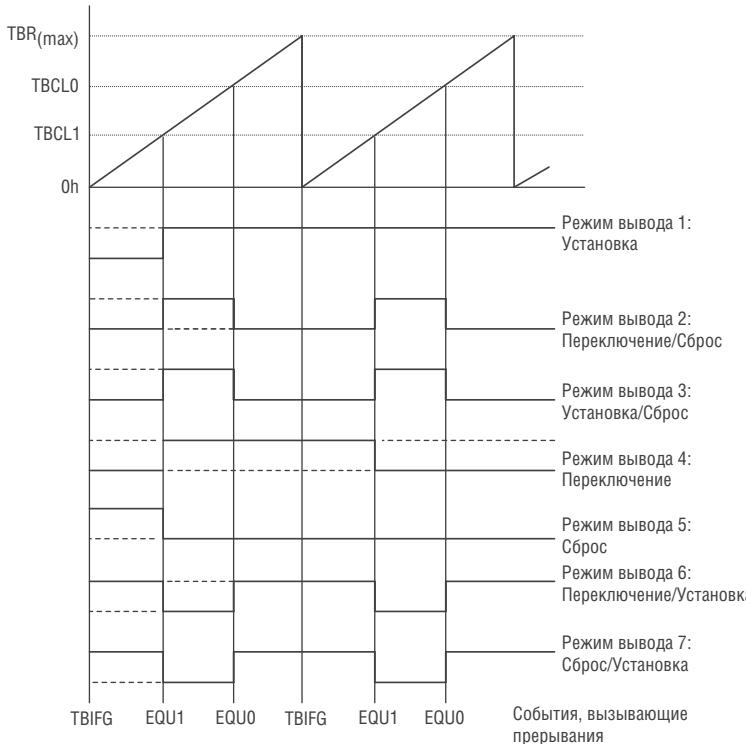
### Пример вывода – таймер в режиме «вверх/вниз»

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. 13-14.

#### 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. 13-13. Пример вывода – таймер в «непрерывном» режиме*

### 13.2.6. Прерывания Таймера В

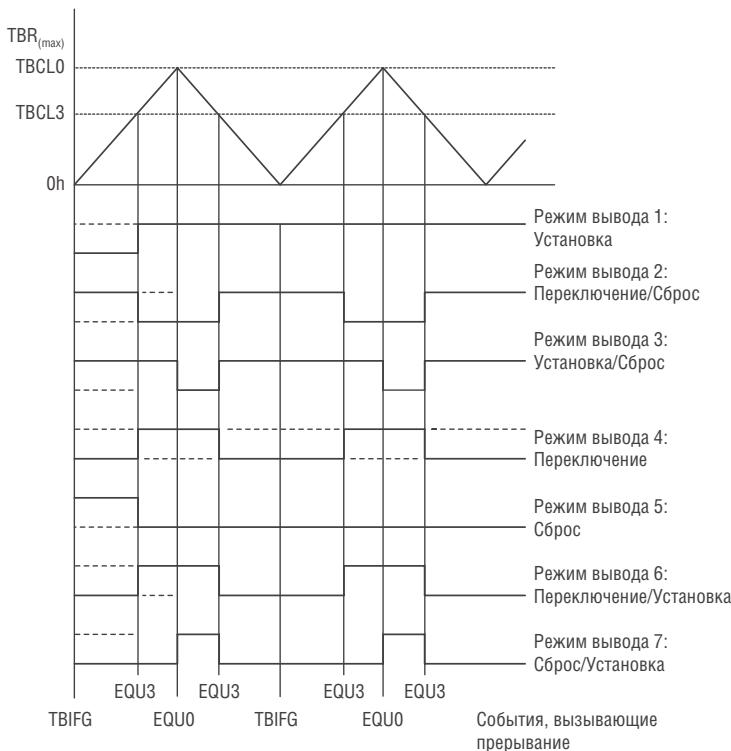
**С 16-разрядным модулем таймера В связаны два вектора прерываний:**

- Вектор прерывания TBCCRO для TBCCRO 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 TBCCR<sub>x</sub>. В режиме сравнения устанавливается любой флаг CCIFG, если TBR досчитал до соответствующего значения TBCL<sub>x</sub>. Программное обеспечение может также устанавливать или очищать любой флаг CCIFG. Все флаги CCIFG запрашивают прерывания, когда установлены их соответствующие биты CCIE и бит GIE.

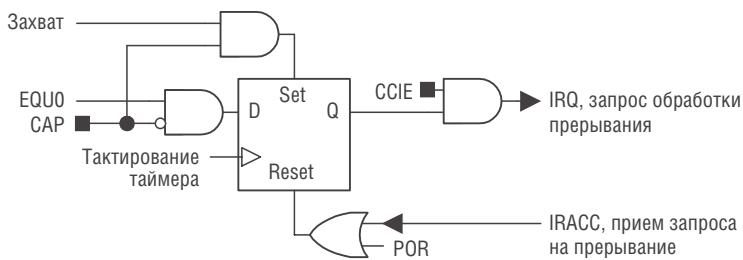
#### Вектор прерывания TBCCRO

Флаг TBCCRO CCIFG обладает наивысшим приоритетом прерывания Таймера В и имеет специализированный вектор прерывания, как показано на



**Рис. 13-14.** Пример вывода – таймер в режиме «вверх/вниз»

рис. 13-15. Флаг TBCCR0 CCIFG автоматически сбрасывается, когда обрабатывается запрос на прерывание TBCCR0.



**Рис. 13-15.** Флаг прерывания TBCCR0 захвата/сравнения

## Генератор вектора прерывания TBIV

Флаг TBIFG и флаги TBCCR<sub>x</sub> 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 В не действуют 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 В3:

; 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	
	; Модуль 1 обработчика позволяет узнать,	
	; что ожидается любое другое прерывание:	
	; на это тратится 5 циклов, но 9 циклов можно	
	; съэкономить, если ожидается другое прерывание	
CCIFG_1_HND	; Вектор 6: Модуль 3	
...	; Задача стартует здесь	
JMP TB_HND	; Просмотр ожидающих прерываний	2

### 13.3. Регистры таймера B

Перечень регистров таймера B приведен в таблице 13-5.

**Таблица 13-5. 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 В	TBCCTL4	Чтение/запись	018Ah	Сброс с POR
Регистр 4 захвата/сравнения таймера В	TBCCR4	Чтение/запись	019Ah	Сброс с POR
Регистр 5 управления захватом/сравнением таймера В	TBCCTL5	Чтение/запись	018Ch	Сброс с POR
Регистр 5 захвата/сравнения таймера В	TBCCR5	Чтение/запись	019Ch	Сброс с POR
Регистр 6 управления захватом/сравнением таймера В	TBCCTL6	Чтение/запись	018Eh	Сброс с POR
Регистр 6 захвата/сравнения таймера В	TBCCR6	Чтение/запись	019Eh	Сброс с POR
Вектор прерывания Таймера В	TBIV	Только чтение	011Eh	Сброс с POR

**TBCTL, регистр управления таймером В**

15	14	13	12	11	10	9	8
Не используется	TBCLGRPx		CNTLx		Не используется	TBSSELx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
Idx		MCx		Не используется	TBCLR	TBIE	TBIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	w-(0)	rw-(0)	rw-(0)

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)

<b>CNTLx</b>	<b>Биты 12-11</b>	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
<b>Не используется</b>	<b>Бит 10</b>	Не используется
<b>TBSSELx</b>	<b>Биты 9-8</b>	Выбор источника тактирования таймера В 00 – TBCLK 01 – ACLK 10 – SMCLK 11 – инвертированный INCLK
<b>IDx</b>	<b>Биты 7-6</b>	Входной делитель. Эти биты позволяют выбрать коэффициент деления для входной тактовой частоты. 00 – /1 01 – /2 10 – /4 11 – /8
<b>MCx</b>	<b>Биты 5-4</b>	Выбор режима. Установка MCx=00h, когда таймер В не используется, позволяет уменьшить потребляемую мощность. 00 – Режим «останов»: таймер остановлен 01 – Режим «вверх»: таймер считает вверх к TBCLO 10 – Непрерывный режим: таймер считает вверх к значению, установленному TBCNTLx 11 – Режим «вверх/вниз»: таймер считает вверх к TBCLO, затем вниз к 0000h
<b>Не используется</b>	<b>Бит 3</b>	Не используется
<b>TBCLR</b>	<b>Бит 2</b>	Очистка таймера В. Установка этого бита сбрасывает TBR, IDx и выбранное направление счета. Бит TBCLR автоматически сбрасывается и всегда читается как нуль.
<b>TBIE</b>	<b>Бит 1</b>	Разрешение прерывания от таймера В. Этот бит разрешает запрос прерывания TBIFG. 0 – Прерывание запрещено 1 – Прерывание разрешено
<b>TBIFG</b>	<b>Бит 0</b>	Флаг прерывания таймера В 0 – Прерывание не ожидается 1 – Ожидается прерывание

### TBR, регистр таймера В

15	14	13	12	11	10	9	8
<b>TBRx</b>							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7        6        5        4        3        2        1        0							
<b>TBRx</b>							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
<b>TBRx</b>	<b>Биты 15-0</b>	Регистр таймера В. Регистр TBR является счетчиком таймера В.					

**TBCCTLx, регистр управления захватом/сравнением**

15	14	13	12	11	10	9	8
<b>CMx</b>		<b>CCISx</b>		<b>SCS</b>	<b>CLLDx</b>		<b>CAP</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
<b>OUTMODx</b>		<b>CCIE</b>	<b>CCI</b>	<b>OUT</b>	<b>COV</b>	<b>CCIFG</b>	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)
<b>CMx</b>	<b>Биты 15-14</b>	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					
		Выбор входа захвата/сравнения. Этими битами выбирается входной сигнал TBCCR <sub>x</sub> . См. справочное руководство конкретного устройства для выяснения подробностей подключения сигналов. 00 – CC <sub>lx</sub> A 01 – CC <sub>lx</sub> B 10 – GND 11 – VCC					
<b>SCS</b>	<b>Бит 11</b>	Синхронизация источника захвата. Этот бит используется для синхронизации входного сигнала захвата с тактовым сигналом. 0 – Асинхронный захват 1 – Синхронный захват					
<b>CLLDx</b>	<b>Бит 10-9</b>	Загрузка защелки сравнения. Эти биты определяют, какое событие вызовет загрузку защелки сравнения. 00 – TBCL <sub>x</sub> загружается при записи в TBCCR <sub>x</sub> 01 – TBCL <sub>x</sub> загружается, когда TBR досчитывает к нулю 10 – TBCL <sub>x</sub> загружается, когда TBR досчитывает к нулю (непрерывный режим или режим «вверх») 11 – TBCL <sub>x</sub> загружается, когда TBR досчитывает к TBCL <sub>x</sub>					
<b>CAP</b>	<b>Бит 8</b>	Режим захвата. 0 – Режим сравнения 1 – Режим захвата					
<b>OUTMODx</b>	<b>Биты 7-5</b>	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					

<b>CCIE</b>	<b>Бит 4</b>	Разрешение прерывания по захвату/сравнению. Этот бит разрешает запрос прерывания от соответствующего флага CCIFG. 0 – Прерывание запрещено 1 – Прерывание разрешено
<b>CCI</b>	<b>Бит 3</b>	Вход захвата/сравнения. Выбранный входной сигнал может быть прочитан этим битом.
<b>OUT</b>	<b>Бит 2</b>	Выход. Этот бит указывает состояние вывода. Если выбран режим вывода 0, этот бит напрямую управляет состоянием выхода. 0 – Низкий уровень выхода 1 – Высокий уровень выхода
<b>COV</b>	<b>Бит 1</b>	Переполнение захвата. Этот бит указывает, что произошло переполнение захвата. Бит COV должен быть сброшен программно. 0 – Переполнения захвата не произошло 1 – Произошло переполнение захвата
<b>CCIFG</b>	<b>Бит 0</b>	Флаг прерывания захвата/сравнения 0 – Прерывание не ожидается 1 – Ожидается прерывание

### TBIV, регистр вектора прерывания таймера В

15	14	13	12		11	10	9	8
<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>		<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
r0	r0	r0	r0		r0	r0	r0	r0
7	6	5	4		3	2	1	0
<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>		<b>TBIVx</b>			<b>0</b>
r0	r0	r0	r0		r-(0)	r-(0)	r-(0)	r0

TBIVx	Биты 15-0	Значение вектора прерывания таймера В
<b>Содержимое TBIV</b>	<b>Источник прерывания</b>	<b>Флаг прерывания</b>
00h	Прерывание не ожидается	–
02h	Захват/сравнение 1	TBCCR1 CCIFG
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
		<b>Приоритет прерывания</b>
		Высший
		Низший

\* Только в устройствах MSP430x4xx.

# MSP430x4xxFamily

## Периферийный интерфейс USART, режим UART

*Раздел XIV.*



## 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 USART. USART0 реализован в устройствах MSP430x42x и MSP430x43x. В дополнение к USART0 в устройствах MSP430x44x реализован второй идентичный USART модуль – USART1.

### 14.1. 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 USART 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

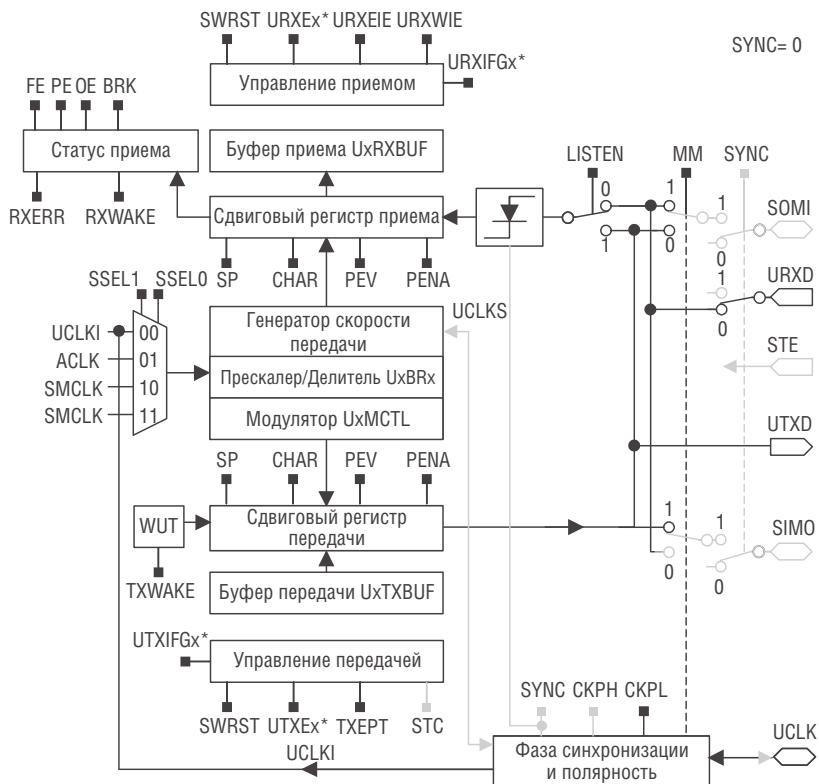
На рис. 14-1 показан USART, сконфигурированный в режиме UART.

### 14.2. Функционирование 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.

#### 14.2.1. 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. 14-1.** Блок-схема USART в режиме UART

USART в состоянии сброса. Когда бит SWRST установлен, сброшены биты URXIE<sub>x</sub>, UTXIE<sub>x</sub>, URXIFG<sub>x</sub>, RXWAKE, TXWAKE, RXERR, BRK, PE, OE, FE и установлены биты UTXIFG<sub>x</sub> и TXEPT. Флаги разрешения приема и передачи URXIE<sub>x</sub> и UTXIE<sub>x</sub> не изменяются битом SWRST. Очистка SWRST позволяет модулю USART функционировать. См. также раздел «Модуль USART, режим I<sup>2</sup>C» при реконфигурировании USART0 из режима I<sup>2</sup>C в режим USART.

#### Примечание: Инициализация и реконфигурирование модуля USART

Процесс инициализации/реконфигурирования USART необходимо выполнить 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*.

#### 14.2.2. Формат символа

Формат символа *USART*, показанный на рис. 14-2, 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.

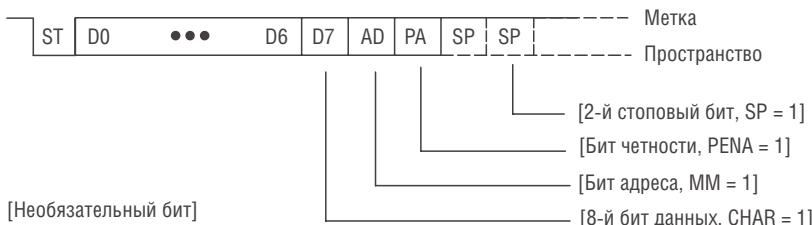


Рис. 14-2. Формат символа

#### 14.2.3. 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. 14-3. 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.

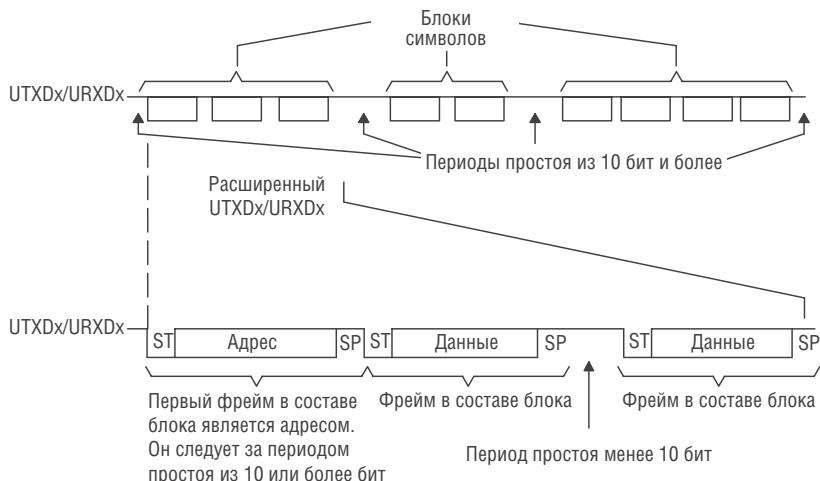


Рис. 14-3. Формат свободной линии

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 посыпает 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).

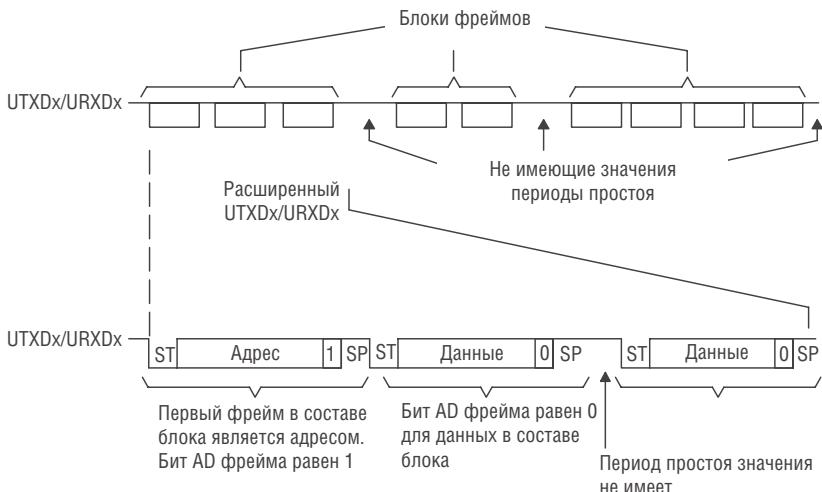
Новый символ, представляющий заданный адрес, сдвигается наружу после периода простоя, идентифицировавшего адрес на UTXDx. Необходима запись первого незначащего символа в UxTXBUF для сдвига бита TXWAKE в WUT и генерации состояния свободной линии. Эти данные отбрасываются и не появляются на UTXDx.

## 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. 14-4). 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 в 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. 14-4.** 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  $t_t$  (около 300 нс) будет проигнорирован. См. руководство по применению конкретного устройства для выяснения точных параметров.

Когда длительность сигнала низкого уровня на URXDx превышает  $t_t$ , 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 14-1.

**Таблица 14-1. Ошибки приема**

Ошибкачное состояние	Описание
<b>Ошибка фрейма</b>	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.
<b>Ошибка четности</b>	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.
<b>Ошибка переполнения приема</b>	Ошибка переполнения появляется в случае, когда символ загружается в UxRXBUF до прочтения предыдущего символа. Когда происходит переполнение, устанавливается бит OE.
<b>Ошибка прерывания (разрыва)</b>	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.

#### **14.2.4. Разрешение приема USART**

Бит разрешения приема URXEx разрешает или запрещает получение данных на URXDx, как показано на рис. 14-5. 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*

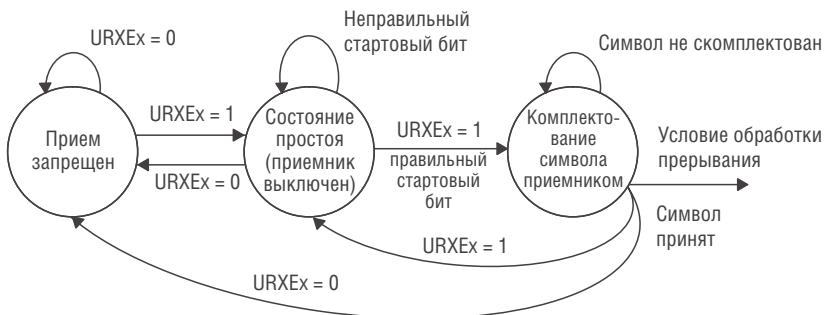


Рис. 14-5. Диаграмма состояний при разрешении приема

момент на  $URXDx$ . В этом случае синхронизация может быть выполнена путем проверки свободного состояния линии перед приемом правильного символа (см.  $URXWIE$ ).

#### 14.2.5. 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. 14-6.

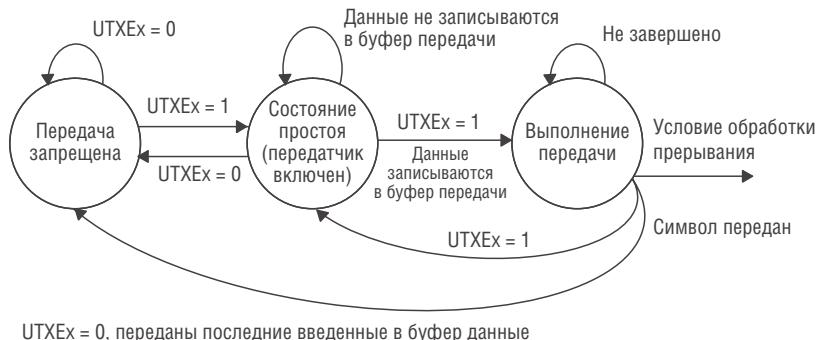


Рис. 14-6. Диаграмма состояний при разрешении передачи

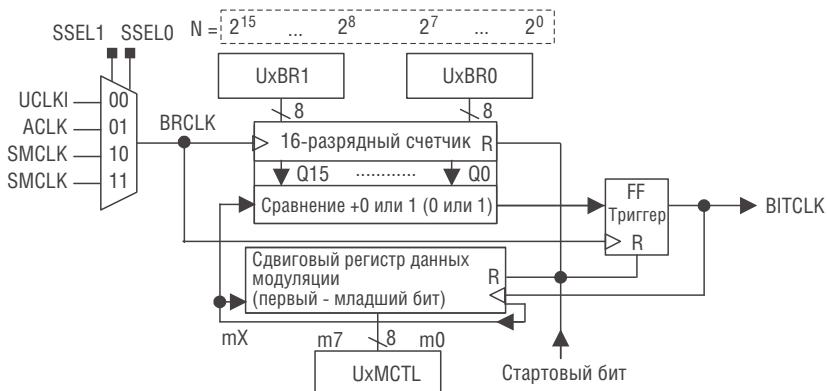
Если бит  $UTXEx$  сбрасывается, передача прекращается. Выполнение операций перемещения любых данных в  $UxTXBUF$  и передачи любых данных в сдвиговый регистр передачи, начатых до очистки бита  $UTXEx$  будет продолжено, пока передача не закончится.

Если передатчик включен ( $UTXEx=1$ ), данные не будут 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 будут перемещены в сдвиговый регистр TX.

Рекомендуется отключать передатчик ( $UTXEx=0$ ) только после завершения любой выполняющейся передачи. На это указывает установка бита опустошения передатчика ( $TXEPT=1$ ). Любые данные, записанные в  $UxTXBUF$  в тот момент, когда передатчик отключен, будут находиться 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.

#### 14.2.6. 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. 14-7. 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. 14-7.** Контроллер генератора передачи MSP430

Синхронизация каждого бита показана на рис. 14-8. 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.

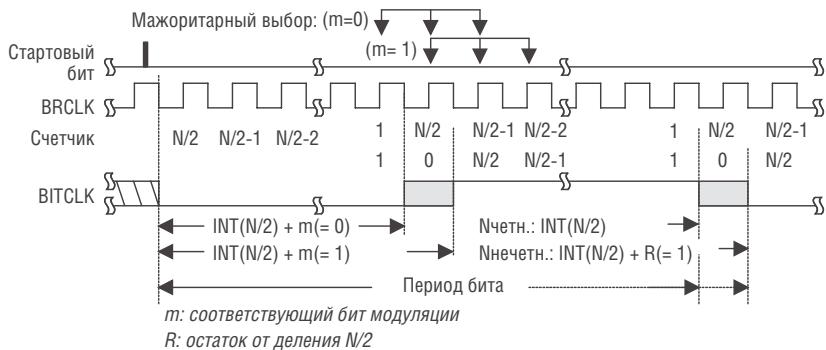


Рис. 14-8. Синхронизация скорости передачи BITCLK

### 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 комбинации  $UxBR0$  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  $N$ :

$$N = BRCLK / \text{скорость передачи в бодах};$$

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-

$$\text{Скорость\_передачи} = \frac{BRCKL}{N} = \frac{BRCLK}{UxBR + \frac{1}{n} \sum_{i=0}^{n-1} m_i}$$

литер нецелого числа. Синхронизация каждого бита расширяется одним тактовым циклом BRCLK, если бит модулятора  $m_i$  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  $m_0$ , следующего бита UxBR плюс  $m_1$  и так далее. Модуляционная последовательность начинается с младшего бита. Когда символ содержит более 8 бит, модуляционная последовательность вновь начинается с  $m_0$  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:

$$\text{Ошибка [%]} = \left\{ \frac{\text{baudrate}}{BRCLK} \times \left[ (j+1) \times UxBR + \sum_{i=0}^{n-1} m_i \right] - (j+1) \right\} \times 100\%,$$

*baudrate* – желаемая скорость передачи в бодах;

*BRCLK* – входная частота: UCLK1, 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.

$$\text{Ошибка\_стартового\_бита [\%]} = \left\{ \frac{\text{baudrate}}{\text{BRCLK}} \times [(0+1) \times UxBR + 1] - 1 \right\} \times 100\% = 2.54\%$$

$$\text{Ошибка\_бита\_данных\_D0 [\%]} = \left\{ \frac{\text{baudrate}}{\text{BRCLK}} \times [(1+1) \times UxBR + 2] - 2 \right\} \times 100\% = 5.08\%$$

$$\text{Ошибка\_бита\_данных\_D1 [\%]} = \left\{ \frac{\text{baudrate}}{\text{BRCLK}} \times [(2+1) \times UxBR + 2] - 3 \right\} \times 100\% = 0.29\%$$

$$\text{Ошибка\_бита\_данных\_D2 [\%]} = \left\{ \frac{\text{baudrate}}{\text{BRCLK}} \times [(3+1) \times UxBR + 3] - 4 \right\} \times 100\% = 2.83\%$$

$$\text{Ошибка\_бита\_данных\_D3 [\%]} = \left\{ \frac{\text{baudrate}}{\text{BRCLK}} \times [(4+1) \times UxBR + 3] - 5 \right\} \times 100\% = -1.95\%$$

$$\text{Ошибка\_бита\_данных\_D4 [\%]} = \left\{ \frac{\text{baudrate}}{\text{BRCLK}} \times [(5+1) \times UxBR + 4] - 6 \right\} \times 100\% = 0.59\%$$

$$\text{Ошибка\_бита\_данных\_D5 [\%]} = \left\{ \frac{\text{baudrate}}{\text{BRCLK}} \times [(6+1) \times UxBR + 5] - 7 \right\} \times 100\% = 3.13\%$$

$$\text{Ошибка\_бита\_данных\_D6 [\%]} = \left\{ \frac{\text{baudrate}}{\text{BRCLK}} \times [(7+1) \times UxBR + 5] - 8 \right\} \times 100\% = -1.66\%$$

$$\text{Ошибка\_бита\_данных\_D7 [\%]} = \left\{ \frac{\text{baudrate}}{\text{BRCLK}} \times [(8+1) \times UxBR + 6] - 9 \right\} \times 100\% = 0.88\%$$

$$\text{Ошибка\_бита\_четности [\%]} = \left\{ \frac{\text{baudrate}}{\text{BRCLK}} \times [(9+1) \times UxBR + 7] - 10 \right\} \times 100\% = 3.42\%$$

$$\text{Ошибка\_стопового\_бита\_1 [\%]} = \left\{ \frac{\text{baudrate}}{\text{BRCLK}} \times [(10+1) \times UxBR + 7] - 11 \right\} \times 100\% = -1.37\%$$

Результаты показывают, что максимальная поразрядная ошибка была 5,08% за период BITCLK.

### Синхронизация битов при приеме

Синхронизация приема состоит из двух источников ошибок. Первый – по-битовая ошибка синхронизации. Второй – ошибка между появлением старто-вого фронта и стартовым фронтом, принятым USART. На рис. 14-9 показаны асинхронные ошибки синхронизации между данными на выводе URXDx и внутренним тактированием скорости передачи.

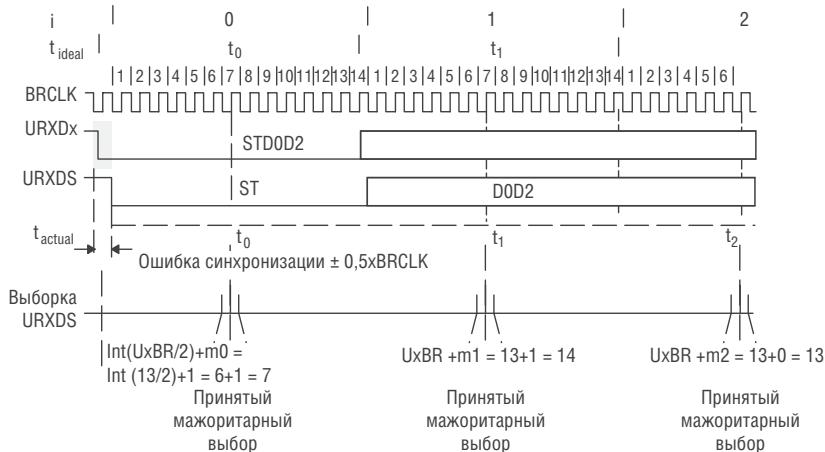


Рис. 14-9. Ошибка приема

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:

$$\text{Ошибка [%]} = \left\{ \frac{baudrate}{BRCLK} \times \left\{ 2 \times \left[ m0 + \text{int}\left(\frac{UxBR}{2}\right) + \left( i \times UxBR + \sum_{i=1}^{n-1} m_i \right) \right] \right\} - 1 - j \right\} \times 100\%,$$

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 (ACKL)

*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*.

$$\text{Ошибка\_стартового\_бита [\%]} = \left( \frac{\text{baudrate}}{\text{BRCLK}} \times [2 \times (1+6) + (0 \times \text{UxBR} + 0)] - 1 - 0 \right) \times 100\% = 2.54\%$$

$$\text{Ошибка\_бита\_данных\_D0 [\%]} = \left( \frac{\text{baudrate}}{\text{BRCLK}} \times [2 \times (1+6) + (1 \times \text{UxBR} + 1)] - 1 - 1 \right) \times 100\% = 5.08\%$$

$$\text{Ошибка\_бита\_данных\_D1 [\%]} = \left( \frac{\text{baudrate}}{\text{BRCLK}} \times [2 \times (1+6) \times (2 \times \text{UxBR} + 1)] - 1 - 2 \right) \times 100\% = 0.29\%$$

$$\text{Ошибка\_бита\_данных\_D2 [\%]} = \left( \frac{\text{baudrate}}{\text{BRCLK}} \times [2 \times (1+6) \times (3 \times \text{UxBR} + 2)] - 1 - 3 \right) \times 100\% = 2.83\%$$

$$\text{Ошибка\_бита\_данных\_D3 [\%]} = \left( \frac{\text{baudrate}}{\text{BRCLK}} \times [2 \times (1+6) + (4 \times \text{UxBR} + 2)] - 1 - 4 \right) \times 100\% = -1.95\%$$

$$\text{Ошибка\_бита\_данных\_D4 [\%]} = \left( \frac{\text{baudrate}}{\text{BRCLK}} \times [2 \times (1+6) + (5 \times \text{UxBR} + 3)] - 1 - 5 \right) \times 100\% = 0.59\%$$

$$\text{Ошибка\_бита\_данных\_D5 [\%]} = \left( \frac{\text{baudrate}}{\text{BRCLK}} \times [2 \times (1+6) + (6 \times \text{UxBR} + 4)] - 1 - 6 \right) \times 100\% = 3.13\%$$

$$\text{Ошибка\_бита\_данных\_D6 [\%]} = \left( \frac{\text{baudrate}}{\text{BRCLK}} \times [2 \times (1+6) + (7 \times \text{UxBR} + 4)] - 1 - 7 \right) \times 100\% = -1.66\%$$

$$\text{Ошибка\_бита\_данных\_D7 [\%]} = \left( \frac{\text{baudrate}}{\text{BRCLK}} \times [2 \times (1+6) + (8 \times \text{UxBR} + 5)] - 1 - 8 \right) \times 100\% = 0.88\%$$

$$\text{Ошибка\_бита\_четности [\%]} = \left( \frac{\text{baudrate}}{\text{BRCLK}} \times [2 \times (1+6) + (9 \times \text{UxBR} + 6)] - 1 - 9 \right) \times 100\% = 3.42\%$$

$$\text{Ошибка\_столового\_бита\_1 [\%]} = \left( \frac{\text{baudrate}}{\text{BRCLK}} \times [2 \times (1+6) + (10 \times \text{UxBR} + 6)] - 1 - 10 \right) \times 100\% = -1.37\%$$

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 14-2 для часовогого 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 14-2. 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	

#### 14.2.7. 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 UTXIE и GIE. UTXIFGx автоматически сбрасывается, если запрос прерывания об служен или если символ записан в UxTXBUF.

UTXIFGx устанавливается после PUC или когда SWRST=1. UTXIE сбрасывается после PUC или когда SWRST=1. Это показано на рис. 14-10.

##### 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 URXIE и GIE. URXIFGx и URXIE 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. 14-11.

URXEIE используется для разрешения или запрещения установки URXIFGx от ошибочных символов. В многопроцессорном адресном режиме URXWE ис-

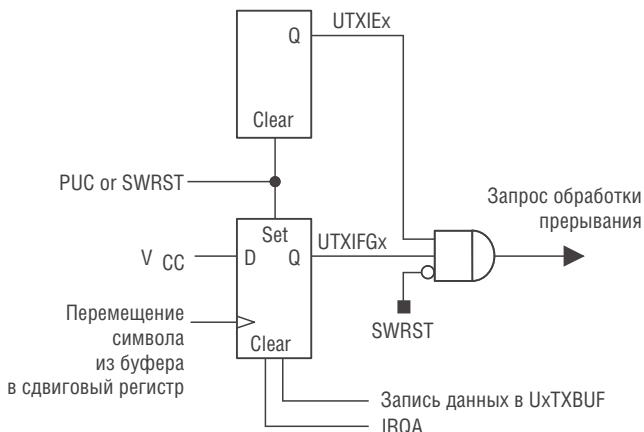


Рис. 14-10. Прерывание при передаче

пользуется для автоматического обнаружения правильных символов адреса и отклонения нежелательных символов данных.

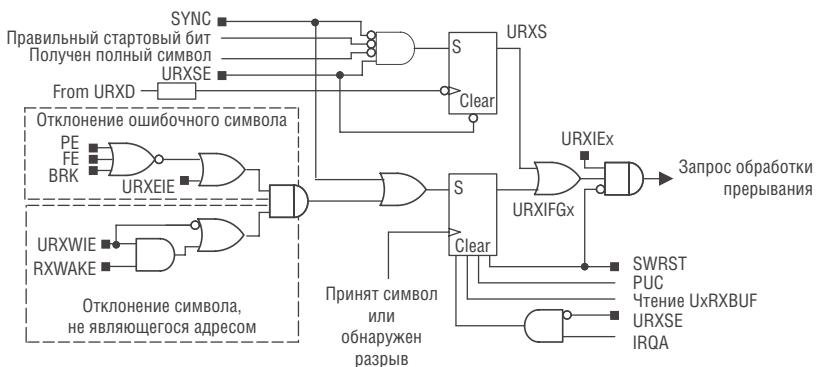


Рис. 14-11. Прерывание при приеме

### 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 програмное обеспечение пользователя читает UxRXBUF.

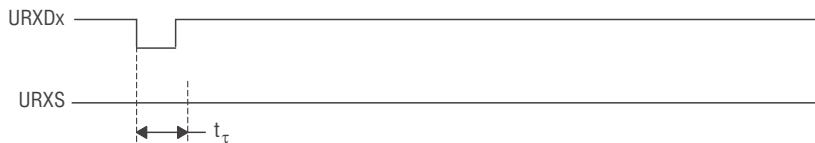
```
; Обработчик прерывания для условия старта фрейма
; и приема символа. BRCLK=DCO.
U0RX_Int    BIT.B #URXIFG0,&IFG2 ; Проверка URXIFGx для определения
                ;старт или символ ?
JNE ST_COND      ;Чтение буфера
MOV.B &UxRXBUF,dst
...
RETI             ;
ST_COND        BIC.B #URXSE,&U0TCTL ; 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 USART**

Когда используется возможность определения стартового фронта при приеме символа, состояние разрыва не может быть выявлено, если источник BRCLK выключен.

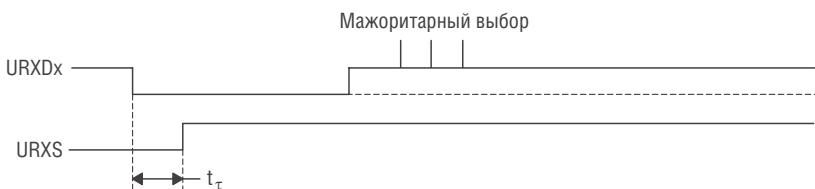
#### Условия определения стартового фронта при приеме

Когда URXSE=1, система подавления импульсных помех предотвращает случайный запуск USART. Любой сигнал низкого уровня на URXDx короче времени  $t_t$  (около 300 нС) будет проигнорирован USART и запрос прерывания не будет сгенерирован, как показано на рис. 14-12. См. руководство по применению конкретного устройства для выяснения точных параметров.



**Рис. 14-12.** Подавление импульсной помехи – прием в USART не начинается

Когда импульсная помеха дольше  $t_t$  или на URXDx появился правильный стартовый бит, USART начинает операцию приема по мажоритарному принципу, как показано на рис. 14-13. Если стартовый бит мажоритарно не обнаружен, USART останавливает прием символа.



**Рис. 14-13.** 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.

## 14.3. Регистры USART: режим USART

В таблице 14-3 приведен перечень регистров для всех устройств с модулем USART. Таблица 14-4 справедлива только для устройств со вторым USART модулем – USART1.

**Таблица 14-3. 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

**Таблица 14-4. 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 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	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
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.* *V* или *BIC.* *V* вместо команд *MOV.* *V* или *CLR.* *V*.

**UxCTL, регистр управления USART**

7	6	5	4	3	2	1	0	
<b>PENA</b>	<b>PEV</b>	<b>SPB</b>	<b>CHAR</b>	<b>LISTEN</b>	<b>SYNC</b>	<b>MM</b>	<b>SWRST</b>	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1	
<b>PENA</b>		<b>Бит 7</b> Включение контроля четности. 0 – Контроль четности отключен 1 – Контроль четности включен. Бит контроля четности сгенерирован ( <i>UTXDx</i> ) и ожидается ( <i>URXDx</i> ). В многопроцессорном режиме с адресным битом он учитывается при вычислении четности.						
<b>PEV</b>		<b>Бит 6</b> Выбор четности. PEV не используется, когда контроль четности отключен. 00 – Нечетный 01 – Четный						
<b>SPB</b>		<b>Бит 5</b> Выбор стопового бита. Количество передаваемых стоповых битов. Приемник всегда проверяет один стоповый бит. 0 – Один стоповый бит 1 – Два стоповых бита						
<b>CHAR</b>		<b>Бит 4</b> Длина символа. Можно выбрать 7-ми или 8-ми разрядный символ. 0 – 7-разрядные данные 1 – 8-разрядные данные						
<b>LISTEN</b>		<b>Бит 3</b> Включение прослушивания. Бит LISTEN включает режим обратной петли. 0 – Отключен 1 – Включен. <i>UTXDx</i> внутренне подключается назад к приемнику.						
<b>SYNC</b>		<b>Бит 2</b> Включение синхронного режима 0 – Режим USART 1 – Режим SPI						
<b>MM</b>		<b>Бит 1</b> Выбор многопроцессорного режима 0 – Многопроцессорный протокол со свободной линией 1 – Многопроцессорный протокол с адресным битом						
<b>SWRST</b>		<b>Бит 0</b> Разрешение программного сброса 0 – Отключен. Сброс USART не задействован 1 – Разрешен. Логика USART удерживается в состоянии сброса						

### UxTCTL, регистр управления передачей USART

15	14	13	12	11	10	9	8
<b>Не используется</b>	<b>CKPL</b>	<b>SSELx</b>		<b>TBCLGRPx</b>	<b>CNTLx</b>	<b>Не используется</b>	<b>TBSSELx</b>
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1
<b>Не используется</b>	<b>Бит 7</b>	Не используется.					
<b>CKPL</b>	<b>Бит 6</b>	Выбор полярности тактового сигнала. 0 – UCLKI = UCLK 1 – UCLKI = инвертированный UCLK					
<b>SSELx</b>	<b>Биты 5-4</b>	Выбор источника. Эти биты выбирают источник тактирования для BRCLK 00 – UCLKI 01 – ACLK 10 – SMCLK 11 – SMCLK					
<b>URXSE</b>	<b>Бит 3</b>	UART принимает стартовый фронт. Бит включает возможность приема UART'ом стартового фронта. 0 – Отключено 1 – Включено					
<b>TXWAKE</b>	<b>Бит 2</b>	«Пробуждение» передатчика 0 – Следующий передаваемый фрейм - данные 1 – Следующий передаваемый фрейм – адрес					
<b>Не используется</b>	<b>Бит 1</b>	Не используется					
<b>TXEPT</b>	<b>Бит 0</b>	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
<b>FE</b>	<b>PE</b>	<b>OE</b>	<b>BRK</b>	<b>URXEIE</b>	<b>URXWIE</b>	<b>RXWAKE</b>	<b>RXERR</b>
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
<b>FE</b>	<b>Бит 7</b>	Флаг ошибки фрейма 0 – Нет ошибки 1 – Символ принят со стоповым битом низкого уровня					
<b>PE</b>	<b>Бит 6</b>	Флаг ошибки контроля четности. Когда PENA=0, PE читается как 0. 0 – Нет ошибки 1 – Символ принят с ошибкой четности					

<b>OE</b>	<b>Бит 5</b>	Флаг ошибки переполнения. Этот бит устанавливается, когда символ перемещен в UxRXBUF до завершения чтения предыдущего символа. 0 – Нет ошибки 1 – Произошла ошибка переполнения
<b>BRK</b>	<b>Бит 4</b>	Флаг обнаружения разрыва 0 – Нет состояния разрыва 1 – Появилось состояние разрыва
<b>URXEIE</b>	<b>Бит 3</b>	Разрешение прерывания при приеме ошибочного символа 0 – Ошибочный символ отклоняется, а URXIFGx не устанавливается 1 – Принятый ошибочный символ устанавливает URXIFGx
<b>URXWIE</b>	<b>Бит 2</b>	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
<b>RXWAKE</b>	<b>Бит 1</b>	Флаг «пробуждения» при приеме 0 – Принятый символ – данные 1 – Принятый символ – адрес
<b>RXERR</b>	<b>Бит 0</b>	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

<b>UxBRx</b>		Правильный диапазон управления скоростью передачи лежит в пределах $3 \leq \text{UxBR} < \text{FFFFh}$ , где $\text{UxBR} = \{\text{UxBR1+UxBR0}\}$ . Если $\text{UxBR} < 3$ , произойдет непредсказуемая синхронизация приема и передачи.
--------------	--	--

### UxMCTL, регистр управления модуляцией USART

7	6	5	4		3	2	1	0
<b>m7</b>	<b>m6</b>	<b>m5</b>	<b>m4</b>		<b>m3</b>	<b>m2</b>	<b>m1</b>	<b>m0</b>
rw	rw	rw	rw		rw	rw	rw	rw

<b>UxMCTLx</b>	<b>Биты 7-0</b>	Биты модуляции. Эти биты выбирают модуляцию для BRCLK.
----------------	-----------------	--

### UxRXBUF, регистр буфера приема USART

7	6	5	4		3	2	1	0
<b>2<sup>7</sup></b>	<b>2<sup>6</sup></b>	<b>2<sup>5</sup></b>	<b>2<sup>4</sup></b>		<b>2<sup>3</sup></b>	<b>2<sup>2</sup></b>	<b>2<sup>1</sup></b>	<b>2<sup>0</sup></b>
r	r	r	r		r	r	r	r

<b>UxRXBUFx</b>	<b>Биты 7-0</b>	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
<b>2<sup>7</sup></b>	<b>2<sup>6</sup></b>	<b>2<sup>5</sup></b>	<b>2<sup>4</sup></b>		<b>2<sup>3</sup></b>	<b>2<sup>2</sup></b>	<b>2<sup>1</sup></b>	<b>2<sup>0</sup></b>
rw	rw	rw	rw		rw	rw	rw	rw

<b>UxTXBUFx</b>	<b>Биты 7-0</b>	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
<b>UTXEO</b>	<b>URXEO</b>							
rw-0	rw-0							

<b>UTXEO</b>	<b>Бит 7</b>	Разрешение передачи USART0. Этот бит включает передатчик USART0. 0 – Модуль выключен 1 – Модуль включен
<b>URXEO</b>	<b>Бит 6</b>	Разрешение приема USART0. Этот бит включает приемник USART0. 0 – Модуль выключен 1 – Модуль включен

	<b>Биты 5-0</b>	Эти биты могут быть использованы другими модулями. См. справочные данные конкретного устройства.
--	-----------------	--

**ME2, регистр 2 включения модуля**

7	6	5	4	3	2	1	0
		<b>UTXE1</b>	<b>URXE1</b>				

rw-0      rw-0

	<b>Биты 7-6</b>	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
<b>UTXE1</b>	<b>Бит 5</b>	Включение передачи USART1. Этот бит включает передатчик USART1. 0 – Модуль выключен 1 – Модуль включен
<b>URXE1</b>	<b>Бит 4</b>	Включение приема USART1. Этот бит включает приемник USART1. 0 – Модуль выключен 1 – Модуль включен
	<b>Биты 3-0</b>	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.

**IE1, регистр 1 разрешения прерываний**

7	6	5	4	3	2	1	0
<b>UTXIE0</b>	<b>URXIE0</b>						

rw-0      rw-0

<b>UTXIE0</b>	<b>Бит 7</b>	Разрешение прерывания при передаче USART0. Этот бит разрешает прерывание UTXIFG0. 0 – Прерывание не разрешено 1 – Прерывание разрешено
<b>URXIE0</b>	<b>Бит 6</b>	Разрешение прерывания при приеме USART0. Этот бит разрешает прерывание URXIFG0. 0 – Прерывание не разрешено 1 – Прерывание разрешено
	<b>Биты 5-0</b>	Эти биты могут быть использованы другими модулями. См. справочные данные конкретного устройства.

**IE2, регистр 2 разрешения прерывания**

7	6	5	4	3	2	1	0
		<b>UTXIE1</b>	<b>URXIE1</b>				

rw-0      rw-0

	<b>Биты 7-6</b>	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
--	-----------------	---

<b>UTXIE1</b>	<b>Бит 5</b>	Разрешение прерывания при передаче USART1. Этот бит разрешает прерывание UTXIFG1. 0 – Прерывание не разрешено 1 – Прерывание разрешено
<b>URXIE1</b>	<b>Бит 4</b>	Разрешение прерывания при приеме USART1. Этот бит разрешает прерывание URXIFG1. 0 – Прерывание не разрешено 1 – Прерывание разрешено
	<b>Биты 3-0</b>	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.

### IFG1, регистр 1 флагов прерываний

7	6	5	4		3	2	1	0
<b>UTXIFG0</b>	<b>URXIFG0</b>							
rw-1 rw-0								

<b>UTXIFG0</b>	<b>Бит 7</b>	Флаг прерывания при передаче USART0. UTXIFG0 устанавливается, когда UOTXBUF пуст. 0 – Прерывание не ожидается 1 – Прерывание ожидается
<b>URXIFG0</b>	<b>Бит 6</b>	Флаг прерывания при приеме USART0. URXIFG0 устанавливается, когда в UOTXBUF принят полный символ. 0 – Прерывание не ожидается 1 – Прерывание ожидается
	<b>Биты 5-0</b>	Эти биты могут быть использованы другими модулями. См. справочные данные конкретного устройства.

### IFG2, регистр 2 флагов прерываний

7	6	5	4		3	2	1	0
		<b>UTXIFG1</b>	<b>URXIFG1</b>					
rw-1 rw-0								

	<b>Биты 7-6</b>	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
<b>UTXIFG1</b>	<b>Бит 5</b>	Флаг прерывания при передаче USART1. UTXIFG1 устанавливается, когда U1TXBUF пуст. 0 – Прерывание не ожидается 1 – Прерывание ожидается
<b>URXIFG1</b>	<b>Бит 4</b>	Флаг прерывания при приеме USART1. URXIFG1 устанавливается, когда в U1RXBUF принят полный символ. 0 – Прерывание не ожидается 1 – Прерывание ожидается
	<b>Биты 3-0</b>	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.

**MSP430x4xxFamily**

## **Периферийный интерфейс USART, режим SPI**

---

*Раздел XV.*



## 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 MSP430x42x и MSP430x43x. В дополнение к USART0, в устройствах MSP430x44x реализован второй идентичный USART модуль – USART1.

### 15.1. 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<sup>2</sup>C 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

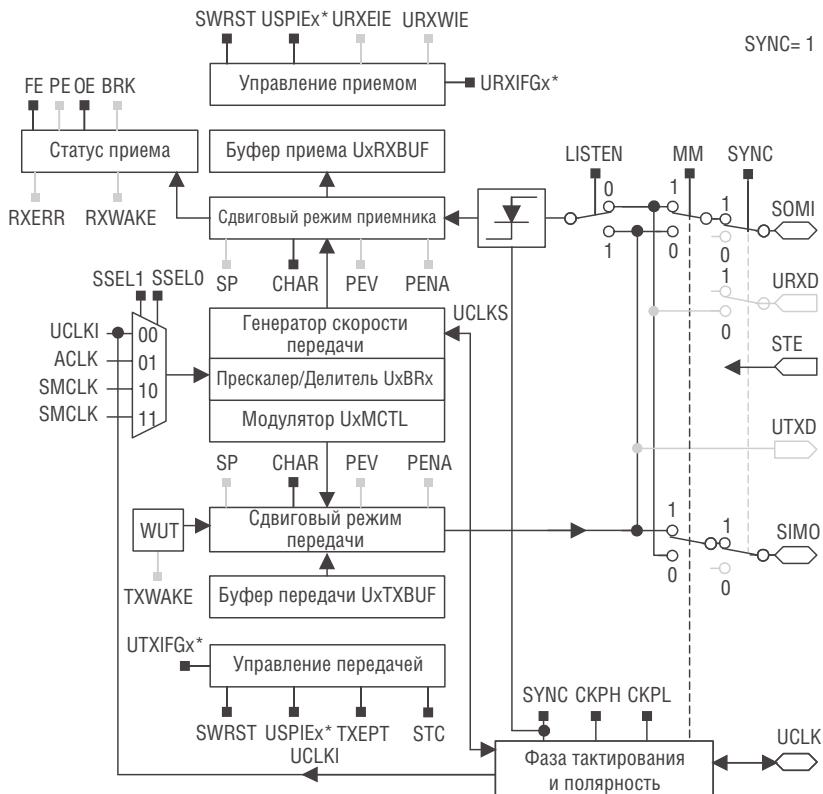
На рис. 15-1 показан USART, сконфигурированный в режиме SPI.

### 15.2. 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
- UCLK Тактирование USART SPI



\* См. справочное руководство конкретного устройства для определения расположения SFR

**Рис. 15-1.** Блок-схема USART в режиме SPI

Режим ведущего: UCLK – выход

Режим ведомого: UCLK – вход

- STE Разрешение передачи ведомого. Используется в 4-выводном режиме, когда на однойшине 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.

### 15.2.1. 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<sup>2</sup>C» для USART0, когда он реконфигурируется из режима I<sup>2</sup>C 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.

### 15.2.2. Режим ведущего

На рис. 15-2 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.

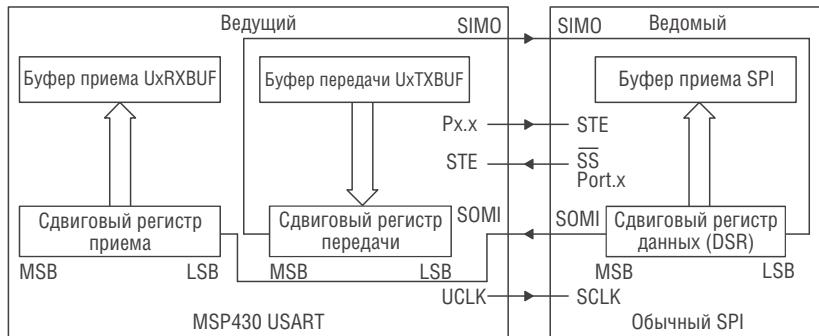


Рис. 15-2. USART – ведущий, внешнее устройство – ведомое

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

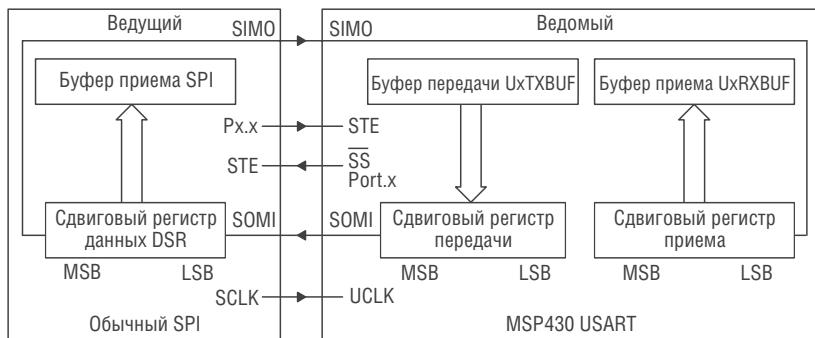
В 4-х выводном режиме ведущего STE используется для предотвращения конфликтов с другим ведущим. Ведущий функционирует normally, когда STE имеет высокий уровень. Когда у STE низкий уровень:

- 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.

#### 15.2.3. Режим ведомого

На рис. 15-3 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. 15-3.** USART – ведомый, внешнее устройство – ведущее

#### 4-х выводной режим ведомого SPI

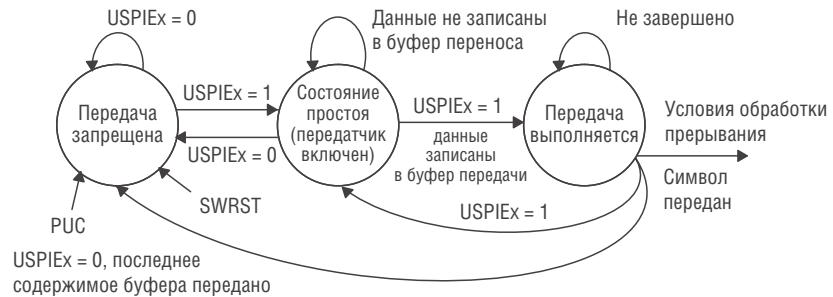
В 4-х выводном режиме ведущего STE используется ведомым для разрешения операций передачи и приема и управляется ведущим SPI. Когда STE имеет низкий уровень, ведомый работает normally. Когда у STE высокий уровень:

- 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.

#### 15.2.4. 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.

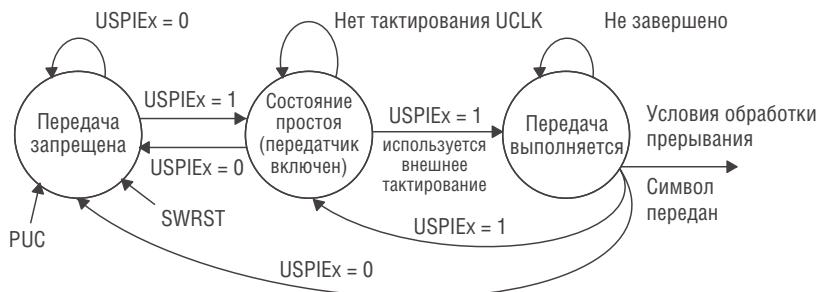


**Рис. 15-4.** Разрешение передачи в режиме ведущего

Сигнал PUC или установка бита SWRST отключают USART немедленно, при этом любая выполняющаяся передача прерывается.

### Разрешение передачи

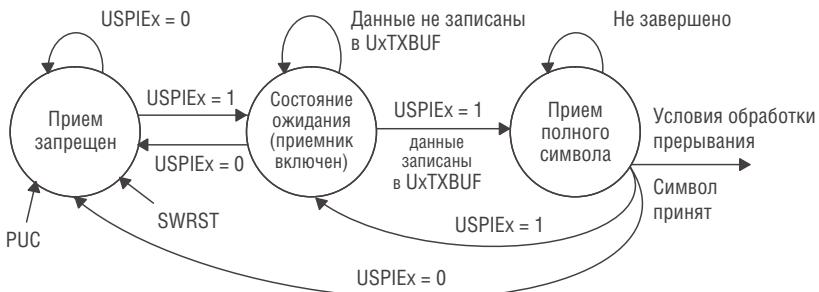
Когда USPIEx=0, любая последующая запись в UxTXBUF не приводит к передаче. Данные, записанные в UxTXBUF начнут передаваться, когда USPIEx=1 и активен источник BRCLK. На рис. 15-4 и рис. 15-5 показаны диаграммы состояний при разрешении передачи.



**Рис. 15-5.** Диаграмма состояний разрешения передачи ведомого

### Разрешение приема

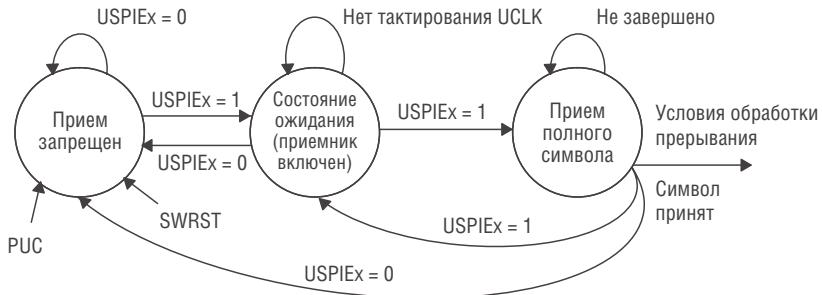
Диаграммы состояний разрешения приема SPI показаны на рис. 15-6 и рис. 15-7. Когда USPIEx=0, UCLK не сдвигает данные в сдвиговый регистр RX.



**Рис. 15-6.** Диаграмма состояний разрешения приема в режиме ведущего SPI

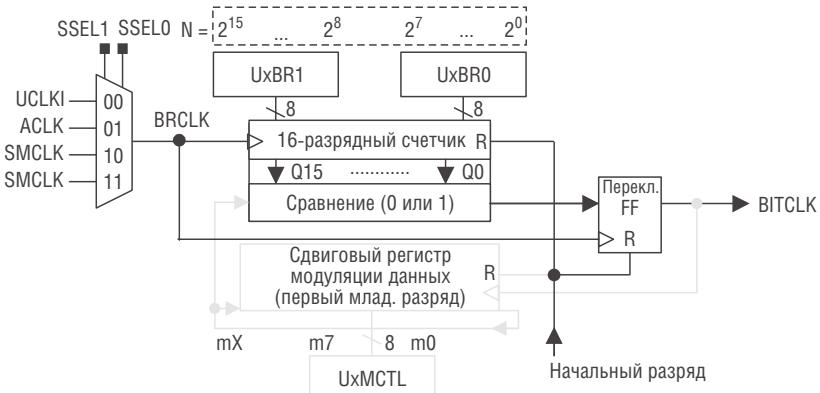
### 15.2.5. Управление последовательным тактированием

Сигнал UCLK на шине SPI обеспечивается ведущим. Когда MM=1, BITCLK обеспечивается генератором скорости передачи USART на выводе UCLK, как



**Рис. 15-7.** Диаграмма состояний разрешения приема ведомым SPI

показано на рис. 15-8. 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.



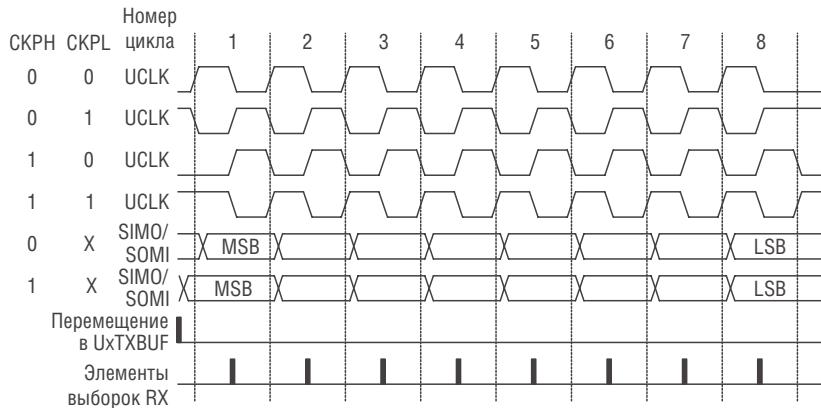
**Рис. 15-8.** Генератор скорости передачи SPI

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:

$$\text{Скорость передачи} = \text{BRCLK}/\text{UxBR}, \text{ где } \text{UxBR} = [\text{UxBR1}, \text{UxBR0}]$$

## 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. 15-9.



**Рис. 15-9.** Синхронизация USART SPI

### 15.2.6. Прерывания SPI

SPI имеет один вектор прерывания для передачи и один вектор прерывания для приема.

#### Работа прерывания SPI при передаче

Флаг прерывания UTXIFG $x$  устанавливается передатчиком для указания, что UxTXBUF готов к приему другого символа. Запрос прерывания генерируется, если также установлены флаги UTXIE $x$  и GIE. UTXIFG $x$  автоматически сбрасывается, если запрос прерывания обработан или если символ записан в UxTXBUF.

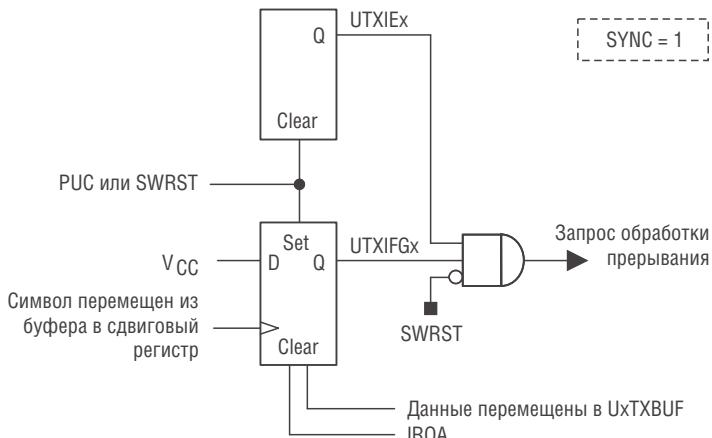
UTXIFG $x$  устанавливается после PUC или когда SWRST=1. UTXIE $x$  сбрасывается после PUC или когда SWRST=1. Это показано на рис. 15-10.

#### Примечание: запись в UxTXBUF в режиме SPI

Запись данных в UxTXBUF, когда UTXIFG $x$ =0 и USPIE $x$ =1 может привести к ошибочной передаче данных.

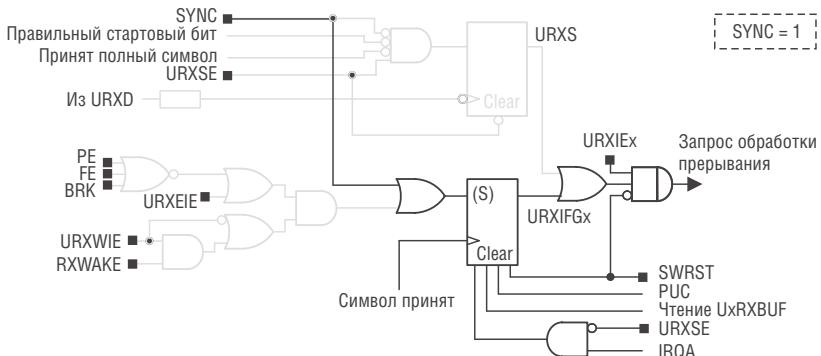
#### Работа прерывания SPI при приеме

Флаг прерывания URXIFG $x$  устанавливается каждый раз, когда символ принят и загружен в UxRXBUF, как показано на рис. 15-11 и 15-12. Запрос пре-



**Рис. 15-10.** 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. 15-11 и рис. 15-12.



**Рис. 15-11.** Функционирование прерывания при приеме

### 15.3. Регистры USART: режим SPI

Регистры USART, показанные в таблице 15-1 и таблице 15-2, структурированы побайтно, поэтому доступ к ним необходимо выполнять с помощью команд работы с байтами.

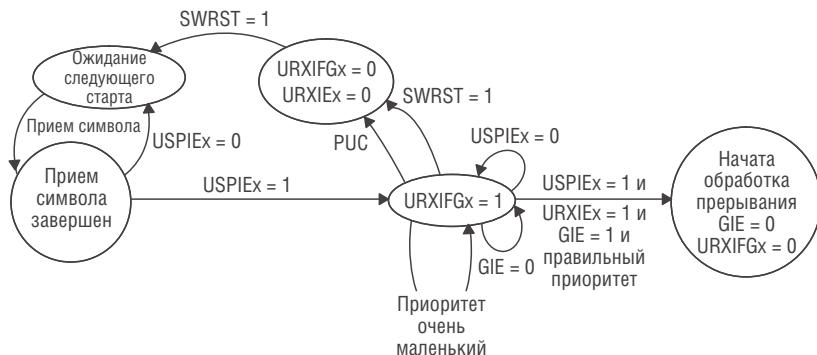


Рис. 15-12. Диаграмма состояний прерывания при приеме

Таблица 15-1. Регистры управления и статуса USART

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Регистр управления USART	UOCTL	Чтение/запись	070h	001h после PUC
Регистр управления передачей	UOTCTL	Чтение/запись	071h	001h после PUC
Регистр управления приемом	UORCTL	Чтение/запись	072h	000h после PUC
Регистр управления модуляцией	UOMCTL	Чтение/запись	073h	Не изменяется
Регистр 0 управления скоростью передачи	UOBRO	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

**Таблица 15-2. 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*. В или *BIC*. Вместо команд *MOV*. В или *CLR*. В.

#### UxCTL, регистр управления USART

7	6	5	4	3	2	1	0
Не используется	Не используется	<b>I<sup>2</sup>C*</b>	<b>CHAR</b>	<b>LISTEN</b>	<b>SYNC</b>	<b>MM</b>	<b>SWRST</b>
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1
<b>Не используется</b>	<b>Биты 7-6</b>	Не используются					
<b>I<sup>2</sup>C*</b>	<b>Бит 5</b>	Включение режима I <sup>2</sup> C. Этот бит позволяет выбрать режим I <sup>2</sup> C или SPI, когда SYNC=1. 0 – Режим SPI 1 – Режим I <sup>2</sup> C					
<b>CHAR</b>	<b>Бит 4</b>	Длина символа 0 – 7-разрядные данные 1 – 8-разрядные данные					
<b>LISTEN</b>	<b>Бит 3</b>	Включение прослушивания. Бит LISTEN включает режим обратной петли. 0 – Отключен 1 – Включен. Сигнал передачи внутренне подключается назад к приемнику.					

<b>SYNC</b>	<b>Бит 2</b>	Включение синхронного режима 0 – Режим UART 1 – Режим SPI
<b>MM</b>	<b>Бит 1</b>	Режим ведущего 0 – USART ведомый 1 – USART ведущий
<b>SWRST</b>	<b>Бит 0</b>	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
<b>CKPH</b>	<b>CKPL</b>		<b>SSELx</b>	<b>Не используется</b>	<b>Не используется</b>	<b>STC</b>	<b>TXEPT</b>
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1
<b>CKPH</b>	<b>Бит 7</b>	Выбор фазы тактирования. Управляет фазой UCLK. 0 – Обычная схема тактирования UCLK 1 – Сигнал UCLK отстает на один полупериод					
<b>CKPL</b>	<b>Бит 6</b>	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.					
<b>SSELx</b>	<b>Биты 5-4</b>	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)					
<b>Не используется</b>	<b>Бит 3</b>	Не используется					
<b>Не используется</b>	<b>Бит 2</b>	Не используется					
<b>STC</b>	<b>Бит 1</b>	Управление передачей ведомого. 0 – 4-х выводной режим SPI: STE включен 1 – 3-х выводной режим SPI: STE выключен					
<b>TXEPT</b>	<b>Бит 0</b>	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	
<b>FE</b>	Не используется	<b>OE</b>	Не используется					
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	
<b>FE</b>	<b>Бит 7</b>	Флаг ошибки фрейма. 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	<b>Бит 6</b>	Не используется						
<b>OE</b>	<b>Бит 5</b>	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	<b>Бит 4</b>	Не используется						
Не используется	<b>Бит 3</b>	Не используется						
Не используется	<b>Бит 2</b>	Не используется						
Не используется	<b>Бит 1</b>	Не используется						
Не используется	<b>Бит 0</b>	Не используется						

### UxBR0, регистр 0 управления скоростью передачи USART

7	6	5	4	3	2	1	0	
<b>2<sup>7</sup></b>	<b>2<sup>6</sup></b>	<b>2<sup>5</sup></b>	<b>2<sup>4</sup></b>	<b>2<sup>3</sup></b>	<b>2<sup>2</sup></b>	<b>2<sup>1</sup></b>	<b>2<sup>0</sup></b>	
rw								

### UxBR1, регистр 1 управления скоростью передачи USART

7	6	5	4	3	2	1	0	
<b>2<sup>15</sup></b>	<b>2<sup>14</sup></b>	<b>2<sup>13</sup></b>	<b>2<sup>12</sup></b>	<b>2<sup>11</sup></b>	<b>2<sup>10</sup></b>	<b>2<sup>9</sup></b>	<b>2<sup>8</sup></b>	
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
<b>m7</b>	<b>m6</b>	<b>m5</b>	<b>m4</b>	<b>m3</b>	<b>m2</b>	<b>m1</b>	<b>m0</b>

rw      rw      rw      rw      rw      rw      rw      rw

UxMCTLx	<b>Биты 7-0</b>	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
<b>2<sup>7</sup></b>	<b>2<sup>6</sup></b>	<b>2<sup>5</sup></b>	<b>2<sup>4</sup></b>	<b>2<sup>3</sup></b>	<b>2<sup>2</sup></b>	<b>2<sup>1</sup></b>	<b>2<sup>0</sup></b>

r      r      r      r      r      r      r      r

UxRXBUFx	<b>Биты 7-0</b>	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
<b>2<sup>7</sup></b>	<b>2<sup>6</sup></b>	<b>2<sup>5</sup></b>	<b>2<sup>4</sup></b>	<b>2<sup>3</sup></b>	<b>2<sup>2</sup></b>	<b>2<sup>1</sup></b>	<b>2<sup>0</sup></b>

rw      rw      rw      rw      rw      rw      rw      rw

UxTXBUFx	<b>Биты 7-0</b>	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
	<b>USPIE0</b>							

rw-0

	<b>Бит 7</b>	Этот бит может быть использован другими модулями. См. справочные данные конкретного устройства.
<b>USPIE0</b>	<b>Бит 6</b>	Включение USART0 SPI. Этот бит включает режим SPI для USART0. 0 – Модуль выключен 1 – Модуль включен
	<b>Биты 5-0</b>	Эти биты могут быть использованы другими модулями. См. справочные данные конкретного устройства.

### ME2, регистр 2 включения модуля

7	6	5	4		3	2	1	0
				<b>USPIE1</b>				

rw-0

	<b>Биты 7-5</b>	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
<b>USPIE1</b>	<b>Бит 4</b>	Включение USART1 SPI. Этот бит включает режим SPI для USART1. 0 – Модуль выключен 1 – Модуль включен
	<b>Биты 3-0</b>	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.

### IE1, регистр 1 разрешения прерываний

7	6	5	4		3	2	1	0
<b>UTXIE0</b>	<b>URXIE0</b>							

rw-0      rw-0

<b>UTXIE0</b>	<b>Бит 7</b>	Разрешение прерывания при передаче USART0. Этот бит разрешает прерывание UTXIFG0. 0 – Прерывание не разрешено 1 – Прерывание разрешено
<b>URXIE0</b>	<b>Бит 6</b>	Разрешение прерывания при приеме USART0. Этот бит разрешает прерывание URXIFG0. 0 – Прерывание не разрешено 1 – Прерывание разрешено

	<b>Биты 5-0</b>	Эти биты могут быть использованы другими модулями. См. справочные данные конкретного устройства.
--	-----------------	--

**IE2, регистр 2 разрешения прерывания**

7	6	5	4	3	2	1	0
		<b>UTXIE1</b>	<b>URXIE1</b>				
rw-0				rw-0			

	<b>Биты 7-6</b>	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
<b>UTXIE1</b>	<b>Бит 5</b>	Разрешение прерывания при передаче USART1. Этот бит разрешает прерывание UTXIFG1. 0 – Прерывание не разрешено 1 – Прерывание разрешено
<b>URXIE1</b>	<b>Бит 4</b>	Разрешение прерывания при приеме USART1. Этот бит разрешает прерывание URXIFG1. 0 – Прерывание не разрешено 1 – Прерывание разрешено
	<b>Биты 3-0</b>	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.

**IFG1, регистр 1 флагов прерываний**

7	6	5	4	3	2	1	0
<b>UTXIFGO</b>	<b>URXIFGO</b>						
rw-1				rw-0			

<b>UTXIFGO</b>	<b>Бит 7</b>	Флаг прерывания при передаче USART0. UTXIFGO устанавливается, когда UOTXBUF пуст. 0 – Прерывание не ожидается 1 – Прерывание ожидается
<b>URXIFGO</b>	<b>Бит 6</b>	Флаг прерывания при приеме USART0. URXIFGO устанавливается, когда в UORXBUF принят полный символ. 0 – Прерывание не ожидается 1 – Прерывание ожидается
	<b>Биты 5-0</b>	Эти биты могут быть использованы другими модулями. См. справочные данные конкретного устройства.

## IFG2, регистр 2 флагов прерываний

7	6	5	4		3	2	1	0
			<b>UTXIFG1</b>	<b>URXIFG1</b>				
rw-1				rw-0				

	<b>Биты 7-6</b>	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.
<b>UTXIFG1</b>	<b>Бит 5</b>	Флаг прерывания при передаче USART1. UTXIFG1 устанавливается, когда U1TXBUF пуст. 0 – Прерывание не ожидается 1 – Прерывание ожидается
<b>URXIFG1</b>	<b>Бит 4</b>	Флаг прерывания при приеме USART1. URXIFG1 устанавливается, когда в U1RXBUF принят полный символ. 0 – Прерывание не ожидается 1 – Прерывание ожидается
	<b>Биты 3-0</b>	Эти биты могут использоваться другими модулями. См. справочные данные конкретного устройства.

**MSP430x4xxFamily**

# **Модуль операционного усилителя ОА**

***Раздел XVI.***



## Модуль операционного усилителя ОА

Модуль ОА представляет собой операционный усилитель общего назначения. В этой главе описана работа модуля ОА. Модуль операционного усилителя ОА присутствует в микроконтроллерах MSP430FG43x.

### 16.1. Модуль операционного усилителя ОА – введение

Модуль операционного усилителя ОА предназначен для обработки аналоговых сигналов для последующего аналогово-цифрового преобразования. Модуль обладает следующими свойствами:

- Питание от одного источника, низкое энергопотребление
- Размах выходного сигнала от положительного до отрицательного напряжения питания (Rail-To-Rail Output).
- Программно устанавливаемый размах входного сигнала от положительного до отрицательного напряжения питания (Rail-To-Rail Input).
- Программно выбираемое соотношение между временем установления и энергопотреблением.
- Программно выбираемая конфигурация.
- Программно выбираемый резистор обратной связи позволяет создавать усилители с программируемым коэффициентом усиления (PGA).

#### Примечание: несколько модулей ОА

Некоторые микроконтроллеры могут содержать более одного модуля ОА. В таких случаях все модули ОА имеют одинаковые характеристики. В данной главе описания модулей и регистров даются в виде  $OAxCTL0$ , где  $x$  обозначает порядковый номер модуля. Если порядковый номер приведен, то описание относится к конкретному модулю. В случае, когда модули идентичны, названия регистров будут иметь вид  $OAxCTL0$ .

Блок схема модуля операционного усилителя ОА показана на рис. 16-1.

### 16.2. Работа модуля операционного усилителя ОА

Модуль ОА конфигурируется при помощи пользовательского программного обеспечения. Настройка и функционирование модуля будут подробно рассмотрены ниже.

#### 16.2.1. Усилитель ОА

Модуль ОА представляет собой конфигурируемый, низкопотребляющий усилитель с широким диапазоном входных и выходных сигналов. Он может быть сконфигурирован как инвертирующий или неинвертирующий усилитель, либо, в комбинации с другими модулями ОА может составлять дифференци-

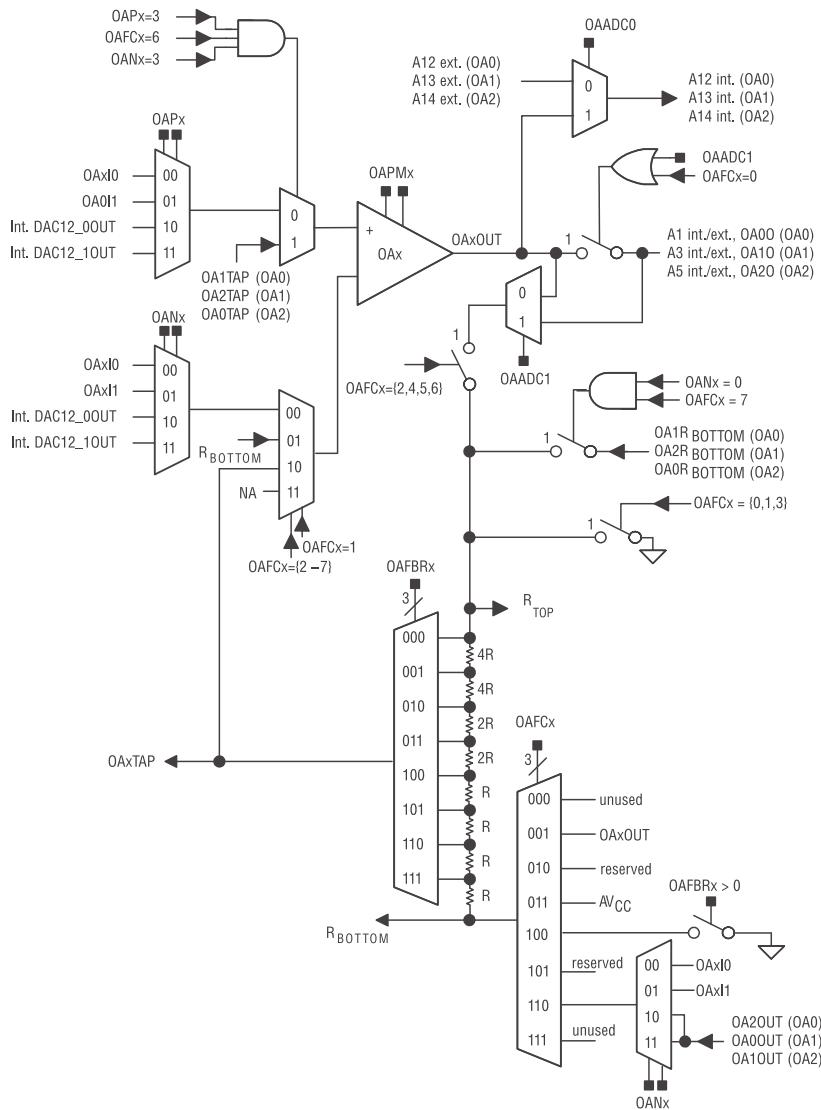


Рис. 16-1. Блок схема модуля операционного усилителя ОА

альный усилитель. Скорость нарастания выходного сигнала регулируется программно с помощью бит OAPMx с целью минимизации потребляемого тока при

требуемом времени установления. Когда OAPM<sub>x</sub> = 0, операционный усилитель выключен, а его выход находится в высокомпедансном состоянии. Когда OAPM<sub>x</sub> > 0, операционный усилитель включен. См. параметры модуля в документации на конкретный МК.

### 16.2.2. Вход ОА

Модуль ОА имеет конфигурируемые входы. Сигналы, подаваемые на инвертирующий (« - ») и неинвертирующий (« + ») входы индивидуально выбираются битами OAN<sub>x</sub> и OAP<sub>x</sub> соответственно. Они могут быть как внешними, так и внутренними – сигналами с одного из модулей ЦП DAC12. Один из неинвертирующих входов объединён у всех модулей ОА. Диапазон входного сигнала модуля ОА программно выбирается битом OARRIP. Когда OARRIP = 0, диапазон входного сигнала может лежать во всём диапазоне от «+» до «-» питания (rail-to-rail input mode), потребление усилителя при этом увеличивается. См. параметры модуля в документации на конкретный МК.

### 16.2.3. Выход ОА

Модуль ОА имеет конфигурируемый выход. Сигналы с выхода ОА могут подаваться на входы A12 (OA0), A13 (OA1) и A14 (OA2) модуля АЦП ADC12 с помощью бита OAADC0. Когда OAADC = 1, выход ОА подключен ко входу АЦП непосредственно внутри микроконтроллера, внешний вход АЦП при этом отключен. Сигналы с выхода ОА могут подаваться на входы A1 (OA0), A3 (OA1) и A5 (OA2) модуля АЦП если OAFC<sub>x</sub>=0 или OAADC1 = 1. В этом случае, выход ОА подключен как ко входу АЦП, так и к внешнему выводу. Выход ОА также подключается к цепочке резисторов при помощи бита OAFC<sub>x</sub>. Для обеспечения требуемого усиления битами OAFBR<sub>x</sub> выбирается требуемый резистор из цепочки.

### 16.2.4. Конфигурация ОА

Модуль ОА может быть сконфигурирован для различных режимов работы с помощью бит OAFC<sub>x</sub> в соответствии с таблицей 16-1.

Таблица 16-1. Выбор режима ОА

OAFC <sub>x</sub>	Режим ОА
000	ОУ общего назначения
001	Буфер с единичным усилением
010	Зарезервирован
011	Компаратор
100	Неинвертирующий усилитель с программируемым коэффициентом усиления (PGA)
101	Зарезервирован

OAFCx	Режим ОА
110	Инвертирующий усилитель с программируемым коэффициентом усиления (PGA)
111	Дифференциальный усилитель

### Режим операционного усилителя общего назначения

В этом режиме цепочка резисторов обратной связи отключена от OAх, а биты регистра OAхCTL0 определяют подключение сигналов. Входы модулей OAх выбираются битами OAPx и OANx . Выход модуля OAх подключен ко входу, выбираемому битами регистра OAхCTL0, АЦП ADC12 внутри микроконтроллера.

### Режим буфера с единичным усилением

В этом режиме выход операционного усилителя подключен к верхнему резистору RTOP из цепочки обратной связи и к инвертирующему входу ОУ, формируя буфер с единичным усилением. Подключение неинвертирующего входа определяется битом OAPx. Внешнее подключение инвертирующего входа отключено, бит OANx игнорируется. Выход модуля OAх подключен ко входу, выбираемому битами регистра OAхCTL0, АЦП ADC12 внутри микроконтроллера.

### Режим компаратора

В этом режиме цепочка резисторов обратной связи отключена от OAх. Верхний резистор из цепочки обратной связи RTOP подключен к AVSS, а нижний RBOTTOM – к AVCC. Сигнал OaxTAP подключен к инвертирующему входу ОА, формируя, таким образом, компаратор с программируемым порогом. Порог срабатывания определяется битами OAFBRx, а подключение неинвертирующего входа определяется битом OAPx. Дополнительно, можно сформировать гистерезис при помощи внешнего резистора положительной обратной связи. Внешнее подключение инвертирующего входа отключено, бит OANx игнорируется. Выход модуля OAх подключен ко входу, выбираемому битами регистра OAхCTL0, АЦП ADC12 внутри микроконтроллера.

### Режим неинвертирующего усилителя с программируемым коэффициентом усиления (PGA)

В этом режиме выход ОА подключен к верхнему резистору цепочки обратной связи RTOP, а нижний резистор цепочки обратной связи RBOTTOM подключен к AVSS. Сигнал OaxTAP подключен к неинвертирующему входу ОА, формируя, таким образом, неинвертирующий усилитель с программируемым усилением, равным  $[1 + \text{отношение } OAxTAP]$ . Отношение OAxTAP определяется битами OAFBRx. Если OAFBRx = 0, усиление равно единице. Подключение неинвертирующего входа определяется битом OAPx. Внешнее подключение

инвертирующего входа отключено, бит OANx игнорируется. Выход модуля OAх подключен ко входу, выбиравшему битами регистра OAхCTL0, АЦП ADC12 внутри микроконтроллера.

### Режим инвертирующего усилителя с программируемым коэффициентом усиления (PGA)

В этом режиме выход OA подключен к верхнему резистору цепочки обратной связи RTOP, а нижний резистор цепочки обратной связи RBOTTOM подключен к аналоговому мультиплексору, который подключает этот вывод к IN0N, IN1N или выходу остальных модулей OA. Управление осуществляется битами OANx. Сигнал OAхTAP подключен к инвертирующему входу OA, формируя, таким образом, инвертирующий усилитель с программируемым усиленiem, равным [минус отношение OAхTAP]. Отношение OAхTAP определяется битами OAFBRx. Подключение неинвертирующего входа определяется битом OAPx. Выход модуля OAх подключен ко входу, выбиравшему битами регистра OAхCTL0, АЦП ADC12 внутри микроконтроллера.

### Режим дифференциального усилителя

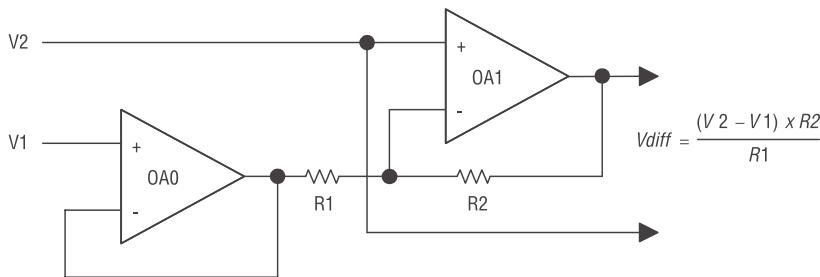
Этот режим позволяет соединять внутри микроконтроллера сигналы двух или трёх модулей OA. На рис. 16-2 изображено подключение двух модулей OA – OA0 и OA1. Выход модуля OAх подключен к верхнему резистору цепочки обратной связи RTOP и второму модулю OAх в режиме инвертирующего усилителя с программируемым коэффициентом усиления. Нижний резистор цепочки обратной связи RBOTTOM остаётся неподключенным, при этом образуется буфер с единичным усилением. Данный буфер в комбинации с одним или двумя оставшимися модулями OAх формируют дифференциальный усилитель. Выход модуля OAх подключен ко входу, выбиравшему битами регистра OAхCTL0, АЦП ADC12 внутри микроконтроллера. На рис. 16-2 показан пример формирования дифференциального усилителя с использованием двух модулей OA – OA0 и OA1. Параметры управляющих регистров даны в таблице 16-2. Усиление определяется битами OAFBRx для OA1 и приведено в таблице 16-3. Внутренние подключения между модулями OA показаны на рис. 16-3.

**Таблица 16-2. Настройки регистров управления для дифференциального усилителя на базе двух ОУ**

Регистр	Настройки (в двоичном виде)
OA0CTL0	00 xx xx 0 0
OA0CTL1	000 111 0 x
OA1CTL0	10 xx xx x x
OA1CTL1	xxx 110 0 x

**Таблица 16-3. Параметры усиления для дифференциального усилителя на базе двух ОУ.**

OA1 OAFBRx	Усиление
000	0
001	1/3
010	1
011	1 2/3
100	3
101	4 1/3
110	7
111	15

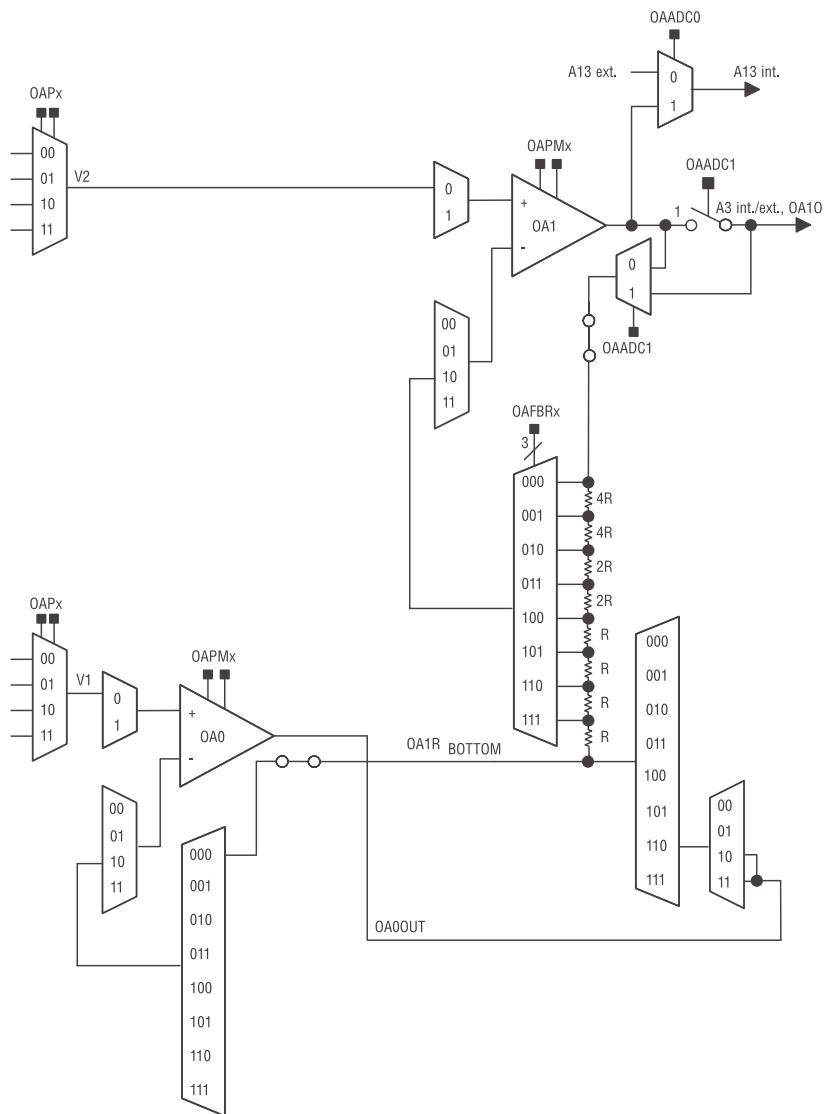


**Рис. 16-2. Дифференциальный усилитель на базе двух ОУ**

На рис. 16-4 показан пример формирования дифференциального усилителя с использованием трёх модулей ОА – OA0, OA1 и OA2. Параметры управляемых регистров даны в таблице 16-4. Усиление определяется битами OAFBRx для OA0 и OA2. Настройки бит OAFBRx должны быть идентичными для обоих модулей OA0 и OA2. Параметры усиления приведены в таблице 16-5. Внутренние подключения между модулями ОА показаны на рис. 16-5.

**Таблица 16-4. Настройки регистров управления для дифференциального усилителя на базе трёх ОУ**

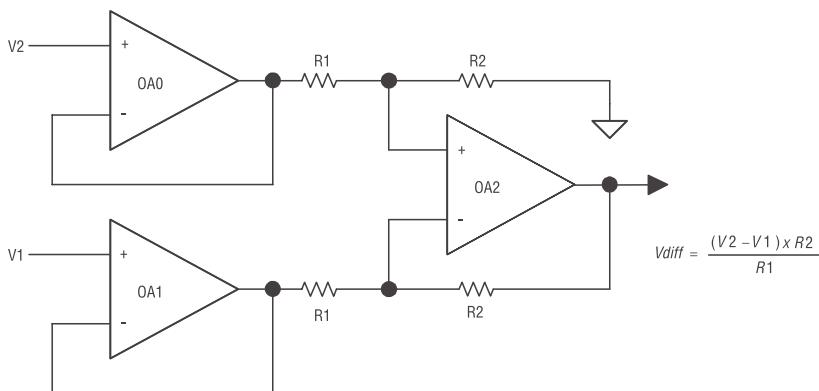
Регистр	Настройки (в двоичном виде)
OA0CTL0	00 xx xx 0 0
OA0CTL1	xxx 001 0 x
OA1CTL0	00 xx xx 0 0
OA1CTL1	000 111 0 x
OA2CTL0	11 11 xx x x
OA2CTL1	xxx 110 0 x



**Рис. 16-3.** Внутренние соединения дифференциального усилителя на базе двух ОУ

**Таблица 16-5. Параметры усиления для дифференциального усилителя на базе трёх ОУ**

ОА0/ОА2 ОАФБРх	Усиление
000	0
001	1/3
010	1
011	1 2/3
100	3
101	4 1/3
110	7
111	15



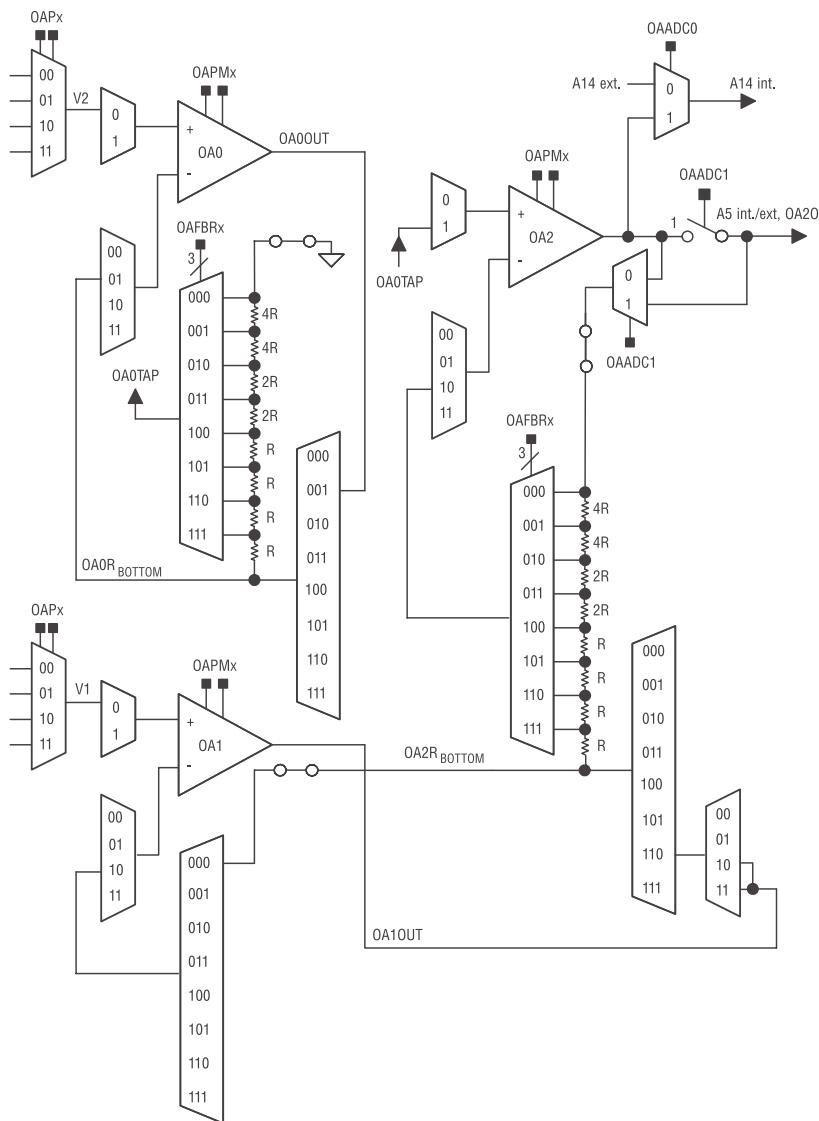
**Рис. 16-4.** Дифференциальный усилитель на базе трёх ОУ

### 16.3. Регистры модуля операционного усилителя ОА

Регистры модуля операционного усилителя ОА перечислены в таблице 16-6.

**Таблица 16-6.**

Регистр	Краткое название	Тип	Адрес	Начальное состояние
Модуль ОА регистр управления 0	ОАОСТЛ0	Чтение/запись	0C0h	Обнулён по сбросу РУС
Модуль ОА регистр управления 1	ОАОСТЛ1	Чтение/запись	0C1h	Обнулён по сбросу РУС



**Рис. 16-5.** Внутренние соединения дифференциального усилителя на базе трёх ОУ

Таблица 16-6. (Окончание)

Регистр	Краткое название	Тип	Адрес	Начальное состояние
Модуль OA1 регистр управления 0	OA1CTL0	Чтение/запись	0C2h	Обнулён по сбросу PUC
Модуль OA1 регистр управления 1	OA1CTL1	Чтение/запись	0C3h	Обнулён по сбросу PUC
Модуль OA2 регистр управления 0	OA2CTL0	Чтение/запись	0C4h	Обнулён по сбросу PUC
Модуль OA2 регистр управления 1	OA2CTL1	Чтение/запись	0C5h	Обнулён по сбросу PUC

**OAxCTL0, регистр управления операционным усилителем 0**

7	6	5	4	3	2	1	0
OANx	OAPx	OAPMx	OAADC1	OAADCO			
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
<b>OANx</b>	Биты 7-6	Выбор подключения инвертирующего входа. Эти биты определяют входной сигнал для инвертирующего входа ОУ. 00 – OAxI0 01 – OAxI1 10 – DAC0 встроенный 11 – DAC1 встроенный					
<b>OAPx</b>	Биты 5-4	Выбор подключения неинвертирующего входа. Эти биты определяют входной сигнал для неинвертирующего входа ОУ. 00 – OAxI0 01 – OAxI1 10 – DAC0 встроенный 11 – DAC1 встроенный					
<b>OAPMx</b>	Биты 3-2	Выбор скорости нарастания выходного сигнала. Этими битами определяется соотношение между скоростью нарастания выходного сигнала и потребляемым током ОУ. 00 – Выключен, выход в высокоимпедансном состоянии 01 – Медленная 10 – Средняя 11 – Быстрая					
<b>OAADC1</b>	Бит 1	Выбор подключения выхода ОУ. Этим битом управляется подключение выхода модуля OAx ко входу Ax модуля АЦП ADC12 при OAFCx > 0. 0 – выход модуля OAx не подключен к A1 (OA0), A3 (OA1) или A5 (OA2) 1 – выход модуля OAx подключен к A1 (OA0), A3 (OA1) или A5 (OA2)					
<b>OAADCO</b>	Бит 0	Выбор подключения выхода ОУ. Этим битом управляется подключение выхода модуля OAx ко входу Ax модуля АЦП ADC12. 0 – выход модуля OAx не подключен к A12 (OA0), A13 (OA1) или A14 (OA2) 1 – выход модуля OAx подключен к A12 (OA0), A13 (OA1) или A14 (OA2)					

## OAxCTL1, регистр управления операционным усилителем 1

7	6	5	4		3	2	1	0
OAFBRx				OAFCx			резервный	OARRIP
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
<b>OAFBRx</b>	Биты 7-5	Выбор резистора обратной связи ОУ 000 – Номинал 0 001 – Номинал 1 010 – Номинал 2 011 – Номинал 3 100 – Номинал 4 101 – Номинал 5 110 – Номинал 6 111 – Номинал 7						
<b>OAFCx</b>	OAFCx	Выбор функции ОУ. Эти биты определяют режим функционирования ОУ. 000 – Общего назначения 001 – Буфер с единичным усилением 010 – Резервный 011 – Компаратор 100 – Неинвертирующий PGA 101 – Резервный 110 – Инвертирующий PGA 111 – Дифференциальный усилитель						
<b>резервный</b>	Бит 1	резервный						
<b>OARRIP</b>	Бит 0	Отключение разрешения размаха входного сигнала от положительного до отрицательного напряжения питания (Rail-To-Rail Input). 0 – режим rail-to-rail включен 1 – размах входного сигнала ограничен. См. параметры в документации на конкретный МК.						

# MSP430x4xxFamily

## Компаратор А

*Раздел XVII.*



## Компаратор А

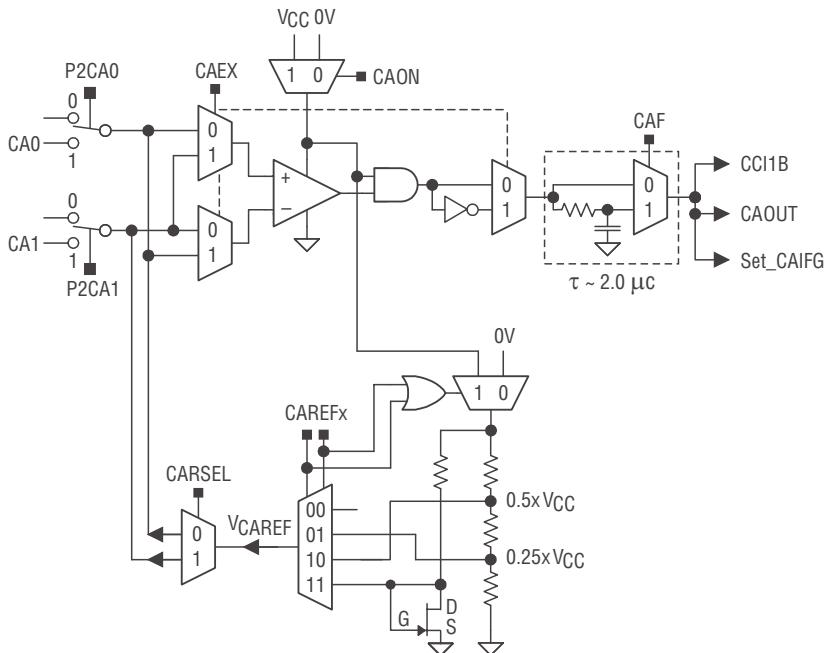
Компаратор А – это аналоговый компаратор напряжения. В этом разделе описывается компаратор А. Компаратор А реализован во всех устройствах MSP430x4xx.

### 17.1. Введение в компаратор А

Модуль компаратора А поддерживает высокоточные аналого-цифровые преобразования напряжения, контроль напряжения питания и мониторинг внешних аналоговых сигналов. Блок-схема компаратора А показана на рис. 17-1.

**Компаратор А имеет следующие возможности:**

- Инвертирующий и не инвертирующий выводы входного мультиплексора
- Программно настраиваемый RC-фильтр на выходе компаратора
- Подключение выхода к входу захвата Таймера А
- Программное управление входным буфером порта



*Рис. 17-1. Блок-схема компаратора А*

- Возможность прерывания
- Настраиваемый генератор опорного напряжения
- Возможность выключения компаратора и опорного генератора

## 17.2. Функционирование компаратора А

Модуль компаратора А конфигурируется программным обеспечением пользователя. Настройка и работа компаратора А обсуждаются в следующих далее разделах.

### 17.2.1. 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.

### 17.2.2. 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 напряжение к соответствующему выходному выводу порта

Внутренне входной переключатель выполнен как переключатель Т-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.

### 17.2.3. 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. 17-2. Колебания на выходе компаратора снижают точность и разрешающую способность результата сравнения. Использование выходного фильтра может уменьшить ошибки, связанные с генерацией компаратора.

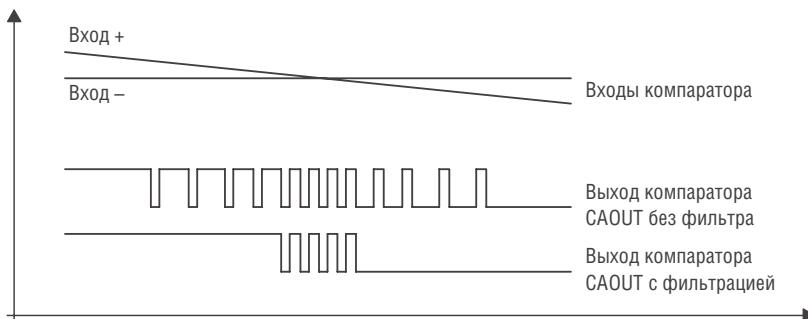


Рис. 17-2. Действие RC-фильтра на выходе компаратора

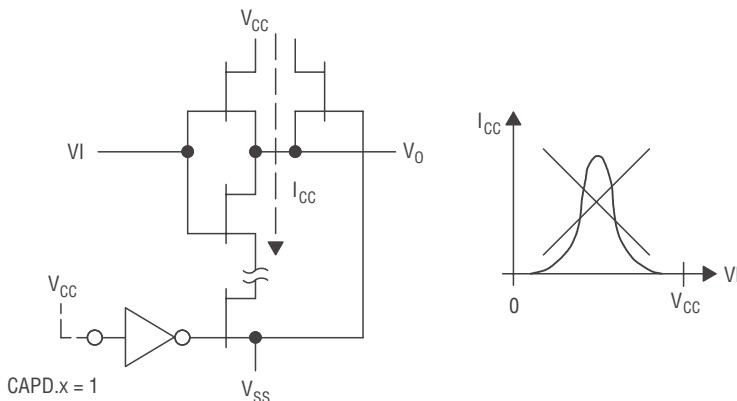
### 17.2.4. Генератор опорного напряжения

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  $V_{cc}$  или фиксированное пороговое напряжение транзистора около 0,55 В.

### 17.2.5. 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  $V_{CC}$  и общим выводом. Этот паразитный ток появляется в случае, если величина входного напряжения находится около переходного уровня ячейки. Отключение буфера вывода порта устраниет паразитный ток и приведет к снижению общего потребления тока.

Когда биты CAPDx установлены, соответствующий входной буфер P2 отключается, как показано на рис. 17-3. Когда величина уровня потребления тока критична, любой вывод P2, подключенный к аналоговым сигналам, должен быть отключен с помощью соответствующего бита CAPDx.



**Рис. 17-3.** Переходная характеристика и рассеивание мощности в инверторе/буфере КМОП

### 17.2.6. Прерывания компаратора A

С компаратором A связан один флаг прерывания и один вектор прерывания, как показано на рис. 17-4. 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.

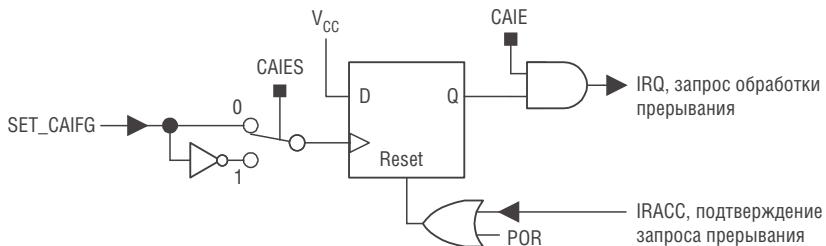


Рис. 17-4. Система прерывания компаратора А

### 17.2.7. 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. 17-5. Сопротивление опорного резистора Rref сравнивается с Rmeas.

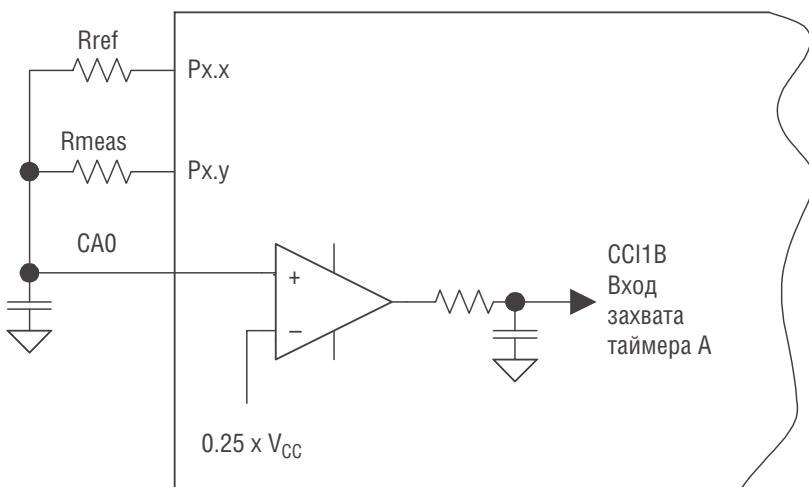


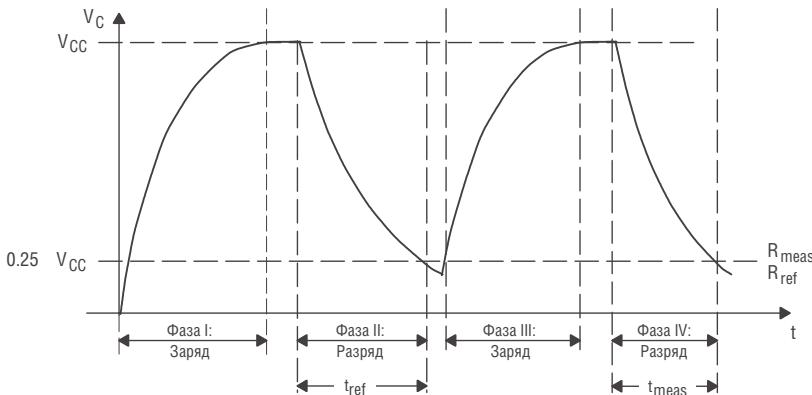
Рис. 17-5. Система измерения температуры

Для вычисления температуры, считанной R<sub>meas</sub>, используются ресурсы MSP430:

- Две цифровых ножки ввода/вывода для заряда и разряда конденсатора.
- Ножка ввода/вывода устанавливается в высокий уровень ( $V_{CC}$ ) 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 R<sub>ref</sub>.
- Один выход разряжает конденсатор через R<sub>meas</sub>.
- Вход «+» подключается к положительному выводу конденсатора.
- Вход «-» подключается к опорному уровню, например  $0,25 \times V_{CC}$ .
- Для минимизации шумов переключения должен использоваться выходной фильтр.
- Выход CAOUT подключен к CCI1B таймера А, выполняющего захват времени разряда конденсатора.

Может быть измерено более одного резистивного элемента. Дополнительные элементы подключаются к САО 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. 17-6.



**Рис. 17-6.** Временная диаграмма систем измерения температуры

$$\frac{N_{meas}}{N_{ref}} = \frac{-R_{meas} \times C \times \ln \frac{V_{ref}}{V_{CC}}}{-R_{ref} \times C \times \ln \frac{V_{ref}}{V_{CC}}}; \quad \frac{N_{meas}}{N_{ref}} = \frac{R_{meas}}{R_{ref}};$$

$$R_{meas} = R_{ref} \times \frac{N_{meas}}{N_{ref}}$$

Значения напряжения  $V_{CC}$  и ёмкости конденсатора должны оставаться постоянными во время преобразования, но они не критичны, т.к. исключены из соотношения:

### 17.3. Регистры компаратора А

Регистры компаратора А приведены в таблице 17-1.

*Таблица 17-1. Регистры компаратора А*

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
Управляющий регистр 1 компаратора А	CACTL1	Чтение/запись	059h	Сброс с POR
Управляющий регистр 2 компаратора А	CACTL2	Чтение/запись	05Ah	Сброс с POR
Отключение порта компаратора А	CAPD	Чтение/запись	05Bh	Сброс с POR

#### CACTL1, регистр управления 1 компаратора А

7	6	5	4	3	2	1	0
CAEX	CARSEL	CAREFx		CAON	CAIES	CAIE	CAIFG
rw-(0)							

CAEX	Бит 7	Обмен в компараторе А. Это бит меняет местами входы компаратора и инвертирует выход компаратора.
CARSEL	Бит 6	Выбор опорного источника компаратора А. 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 «+»

<b>CAREF</b>	Биты 5-4	Опорное напряжение компаратора А. Эти биты выбирают опорное напряжение VCAREF. 00 – Встроенная опора отключена. Может использоваться внешнее опорное напряжение. 01 – $0,25^*V_{cc}$ 10 – $0,50^*V_{cc}$ 11 – Выбирается диодная опора
<b>CAON</b>	Бит 3	Включение компаратора А. Этот бит включает компаратор. При выключении компаратора он перестает потреблять ток. Опорная схема включается и выключается независимо от компаратора. 0 – Выключен 1 – Включен
<b>CAIES</b>	Бит 2	Выбор фронта прерывания компаратора А 0 – Нарастающий фронт 1 – Спадающий фронт
<b>CAIE</b>	Бит 2	Разрешение прерывания от компаратора А 0 – Запрещено 1 – Разрешено
<b>CAIFG</b>	Бит 0	Флаг прерывания компаратора А 0 – Прерывание не ожидается 1 – Прерывание ожидается

**CACTL2, регистр управления компаратора А**

7	6	5	4	3	2	1	0
<b>Не используется</b>				<b>P2CA1</b>	<b>P2CA0</b>	<b>CAF</b>	<b>CAOUT</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)

<b>Не используется</b>	Бит 7-4	Не используется
<b>P2CA1</b>	Бит 3	Ножка к СА1. Этот бит определяет функцию ножки СА1. 0 – Ножка не подключается к СА1 1 – Ножка подключается к СА1
<b>P2CA0</b>	Бит 2	Ножка к СА0. Этот бит определяет функцию ножки СА0. 0 – Ножка не подключается к СА0 1 – Ножка подключается к СА0
<b>CAF</b>	Бит 1	Выходной фильтр компаратора А 0 – Выход компаратора А не фильтруется 1 – Выход компаратора А фильтруется
<b>CAOUT</b>	Бит 0	Выход компаратора А. Этот бит отражает значение на выходе компаратора. Запись в этот бит не приводит к какому-либо результату.

## Регистр CAPD отключения порта компаратора А

7	6	5	4	3	2	1	0
<b>CAPD7</b>	<b>CAPD6</b>	<b>CAPD5</b>	<b>CAPD4</b>	<b>CAPD3</b>	<b>CAPD2</b>	<b>CAPD1</b>	<b>CAPD0</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
<b>CAPDx</b>	Биты 7-0	Отключение порта компаратора А. Эти биты индивидуально отключают входной буфер выводов порта, связанных с компаратором А. 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.					

# MSP430x4xxFamily

## Контроллер ЖКИ

*Раздел XVIII.*



## Контроллер ЖКИ

Модуль контроллера ЖКИ поддерживает статические ЖКИ и ЖКИ с коэффициентом мультиплексирования 2, 3 и 4 (2-тих, 3-тих, 4-тих). Модуль контроллера ЖКИ присутствует во всех микроконтроллерах семейства MSP430x4xx.

### 18.1. Модуль контроллера ЖКИ – введение

Модуль контроллера ЖКИ непосредственно управляет жидкокристаллическими индикаторами, автоматически генерируя требуемые для них переменные напряжения на выводах строк и сегментов. Модуль контроллера ЖКИ, встроенный в микроконтроллеры MSP430, поддерживает статические ЖКИ и ЖКИ с коэффициентом мультиплексирования 2,3 и 4 (2-тих, 3-тих, 4-тих). Модуль контроллера ЖКИ обладает следующими свойствами:

- Дисплейная память
- Автоматическая генерация сигналов управления
- Программно устанавливаемая частота обновления индикации
- Возможность мигания
- Поддержка четырёх типов ЖКИ:
  - Статических
  - С коэффициентом мультиплексирования 2 и смещением 1/2 (2-tix, 1/2 bias)
  - С коэффициентом мультиплексирования 3 и смещением 1/3 (2-tix, 1/3 bias)
  - С коэффициентом мультиплексирования 4 и смещением 1/3 (2-tix, 1/3 bias)

Блок схема модуля контроллера ЖКИ показана на рис. 18-1.

#### Примечание: максимальное количество сегментов ЖКИ

Различных микроконтроллеры семейства имеют разное максимально допустимое число сегментов ЖКИ:

Микроконтроллеры '41x: от S0 до S23

Микроконтроллеры '42x: от S0 до S31

Микроконтроллеры '43x: от S0 до S31 (в 80-выводном корпусе) или от S0 до S39 (в 100-выводном корпусе)

Микроконтроллеры '44x: от S0 до S39

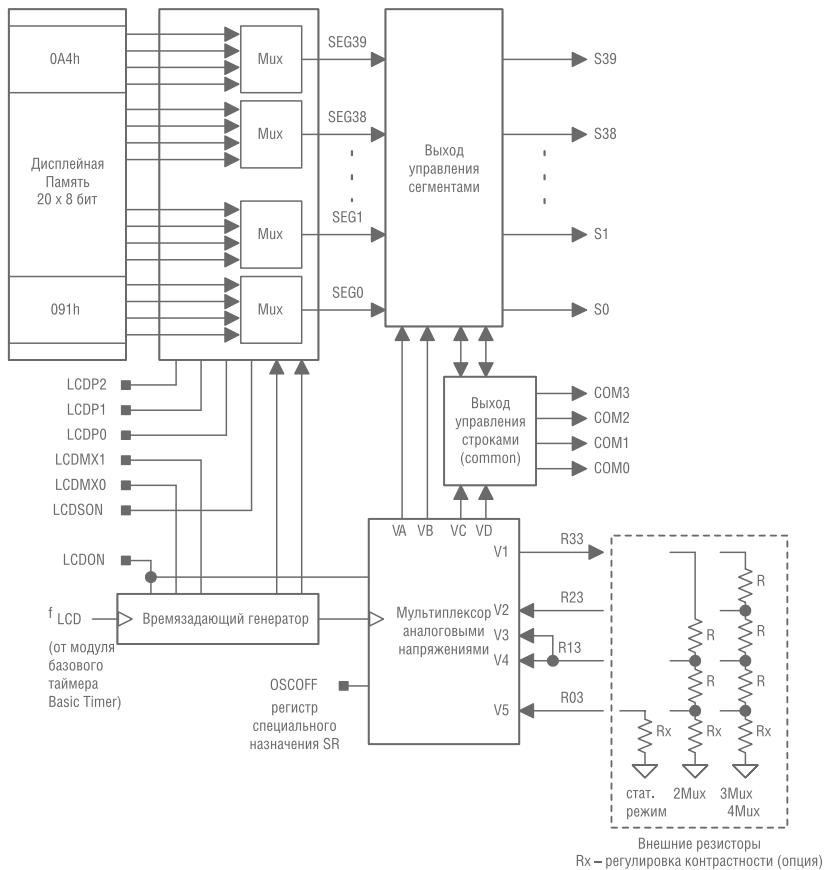


Рис. 18-1. Блок схема модуля контроллера ЖКИ

## 18.2. Работа модуля контроллера ЖКИ

Модуль контроллера ЖКИ конфигурируется при помощи пользовательского программного обеспечения. Настройка и функционирование модуля будут подробно рассмотрены ниже.

### 18.2.1. Дисплейная память

Карта памяти модуля контроллера ЖКИ показана на рис. 18-2. Каждый бит дисплейной памяти либо соответствует одному сегменту ЖКИ, либо не ис-

пользуется, в зависимости от установленного режима работы. Для включения сегмента ЖКИ следует установить в «1» соответствующий бит.

Соответствующие выводы строк (common)	3	2	1	0	3	2	1	0	Соответствующие выводы сегментов
Адрес	7				0	n			
0A4h	--	--	--	--	--	--	--	--	38 39, 38
0A3h	--	--	--	--	--	--	--	--	36 37, 36
0A2h	--	--	--	--	--	--	--	--	34 35, 34
0A1h	--	--	--	--	--	--	--	--	32 33, 32
0A0h	--	--	--	--	--	--	--	--	30 31, 30
09Fh	--	--	--	--	--	--	--	--	28 29, 28
09Eh	--	--	--	--	--	--	--	--	26 27, 26
09Dh	--	--	--	--	--	--	--	--	24 25, 24
09Ch	--	--	--	--	--	--	--	--	22 23, 22
09Bh	--	--	--	--	--	--	--	--	20 21, 20
09Ah	--	--	--	--	--	--	--	--	18 19, 18
099h	--	--	--	--	--	--	--	--	16 17, 16
098h	--	--	--	--	--	--	--	--	14 15, 14
097h	--	--	--	--	--	--	--	--	12 13, 12
096h	--	--	--	--	--	--	--	--	10 11, 10
095h	--	--	--	--	--	--	--	--	8 9, 8
094h	--	--	--	--	--	--	--	--	6 7, 6
093h	--	--	--	--	--	--	--	--	4 5, 4
092h	--	--	--	--	--	--	--	--	2 3, 2
091h	--	--	--	--	--	--	--	--	0 1, 0

**Рис. 18-2. Дисплейная память модуля контроллера ЖКИ**

### 18.2.2. Мигание ЖКИ

Модуль контроллера ЖКИ поддерживает мигание. Над каждым битом в дисплейной памяти производится операция «логическое И» с битом LCDSON. Когда LCDSON =1, каждый сегмент либо включен, либо выключен, в зависимости от значения соответствующего бита в дисплейной памяти. Когда LCDSON =0, все сегменты выключены.

### 18.2.3. Генерация тактовых сигналов

Для генерации тактовых сигналов, управляющих строками и сегментами, модуль контроллера ЖКИ использует сигнал  $f_{LCD}$  от модуля базового таймера Basic Timer1. Необходимая частота сигнала  $f_{LCD}$  зависит от требований по частоте обновления индикации ЖКИ и от степени мультиплексирования. См. главу Базовый таймер Basic Timer1 по части дополнительной информации о конфигурировании частоты  $f_{LCD}$ .

### 18.2.4. Генерация напряжений ЖКИ

Требуемые для ЖКИ уровни напряжений формируются извне и подаются на выводы R33, R23, R13 и R03. Аналоговые напряжения на этих выводах формируются резистивными делителями с соответствующими весовыми коэффициентами, как показано в таблице 18-1. Типовое значение для  $R$  – 680 кОм. В зависимости от типа ЖКИ могут применяться резисторы от 100 кОм до 1МОм. Вывод R33 представляет собой коммутируемый выход питания  $V_{CC}$ . Это даёт возможность отключать резистивный делитель, снижая энергопотребление в те моменты, когда ЖКИ не используется.

**Таблица 18-1. Внешние аналоговые напряжения для модуля контроллера ЖКИ**

OSCOFF	LCDMXx	LCDON	VA	VB	VC	VD	R33
x	xx	0	0	0	0	0	Выключен
1	xx	x	0	0	0	0	Выключен
0	00	1	V5/V1	V1/V5	V5/V1	V1/V5	Включен
0	01	1	V5/V1	V1/V5	V3/V3	V1/V5	Включен
0	1x	1	V5/V1	V2/V4	V4/V2	V1/V5	Включен

### Контроль контрастности ЖКИ

Контроль контрастности ЖКИ может осуществляться извне регулировкой напряжения на выводе R03. Как правило, такая регулировка осуществляется подключением дополнительного резистора RX на землю. Увеличение напряжения на выводе R03 уменьшает эквивалентное напряжение на сегментах, что, в свою очередь, приводит к снижению контрастности.

### 18.2.5. Выходы модуля контроллера ЖКИ

Некоторые из выходов сегментов, строк и RX модуля контроллера ЖКИ мультиплексированы со входами/выходами общего назначения процессора. Такие выводы могут быть использованы как в качестве цифровых входов/выходов, так и в качестве выходов модуля контроллера ЖКИ. Функция выводов

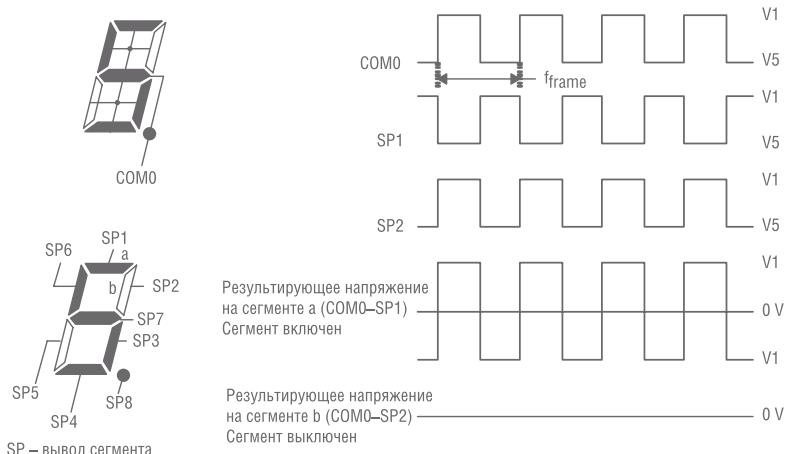
COMx и Rxх, если они мультиплексированы с цифровыми входами/выходами, определяются состоянием бит PxSELx, как описано в главе Цифровые входы/выходы. Функция выводов сегментов ЖКИ, если они мультиплексированы с цифровыми входами/выходами, определяются состоянием бит LCDPx. Биты LCDPx определяют состояние выводов по группам. Когда  $LCDPx = 0$ , ни один из выводов не является выходом модуля контроллера ЖКИ. Когда  $LCDPx = 1$ , в качестве выходов модуля контроллера ЖКИ функционируют выводы S0-S15. Когда  $LCDPx > 1$ , выводы сегментов подключаются к Модулю контроллера ЖКИ группами по четыре. Например, если  $LCDPx = 2$ , в качестве выходов модуля контроллера ЖКИ функционируют выводы S0-S19.

**Примечание:** Биты LCDPx не влияют на обособленные сегментные выводы контроллера ЖКИ

Биты LCDPx влияют только на те выводы, которые имеют функции, переключаемые между выходами контроллера ЖКИ и цифровыми входами/выходами. На обособленные сегментные выходы контроллера ЖКИ биты LCDPx влияния не оказывают.

### 18.2.6. Статический режим

В статическом режиме каждый из сегментных выходов микроконтроллера MSP430 управляет одним сегментом ЖКИ, при этом используется только одна строка, COM0. На рис. 18-3 показаны примеры осцилограмм сигналов в статическом режиме.



**Рис. 18-3.** Примеры осцилограмм сигналов в статическом режиме

На рис. 18-4 показан пример статического ЖКИ, его подключение к MSP430 и карта сегментов. Обратите внимание, что это всего лишь пример, реальная карта сегментов в каждом конкретном случае будет зависеть от расположения выводов ЖКИ и его подключения к MSP430.

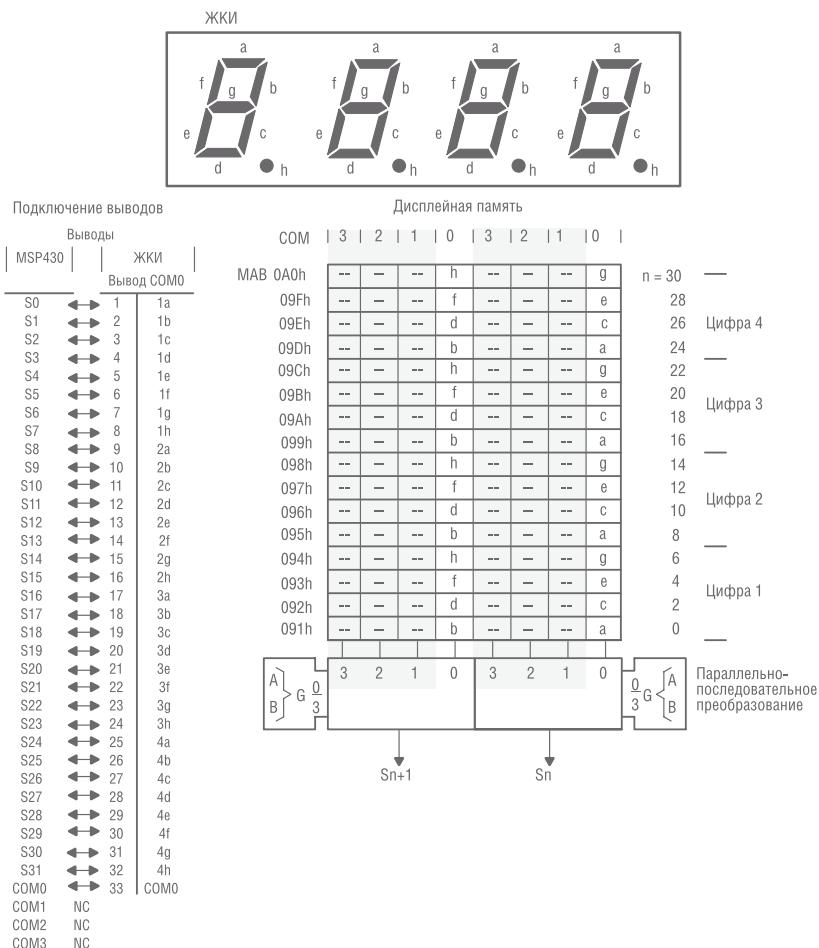


Рис. 18-4. Пример статического ЖКИ

### Пример программы для статического режима

; Все восемь сегментов цифры как правило расположены  
; в четырёх байтах дисплейной памяти при использовании  
; статического режима.

;

a	EQU	001h
b	EQU	010h
c	EQU	002h
d	EQU	020h
e	EQU	004h
f	EQU	040h
g	EQU	008h
h	EQU	080h

; Будет отображаться содержимое регистра Rx.

; Таблица представляет включенные сегменты

; в соответствии с содержимым Rx

```

MOV.B Table (Rx),RY ;Загрузить расположение
                      ;сегментов в служебную
                      ;область памяти.
                      ;(Ry) = 0000 0000 hfdb geca
MOV.B Ry,&LCDn      ;Примечание:
                      ;Записываются все биты
                      ;байта памяти ЖКИ
                      ;(Ry) = 0000 0000 0hfd bgec
MOV.B Ry,&LCDn+1   ;Примечание:
                      ;Записываются все биты
                      ;байта памяти ЖКИ
                      ;(Ry) = 0000 0000 00hf dbge
MOV.B Ry,&LCDn+2   ;Примечание:
                      ;Записываются все биты
                      ;байта памяти ЖКИ
                      ;(Ry) = 0000 0000 000h fdbg
MOV.B Ry,&LCDn+3   ;Примечание:
                      ;Записываются все биты
                      ;байта памяти ЖКИ
.....
.....
;
```

Table	DB	$a+b+c+d+e+f$ ; отображает «0»
	DB	$b+c$ ; отображает «1»
.....		
.....		
	DB	
.....		

### 18.2.7. Режим двойного мультиплексирования (2-Mux)

В режиме двойного мультиплексирования каждый сегментный выход микроконтроллера MSP430 управляет двумя сегментами ЖКИ, при этом используется две линии строк, COM0 и COM1. На рис. 18-5 показаны примеры осциллографм сигналов в режиме двойного мультиплексирования.

На рис. 18-6 показан пример ЖКИ в режиме двойного мультиплексирования, его подключение к MSP430 и карта сегментов. Обратите внимание, что это всего лишь пример, реальная карта сегментов в каждом конкретном случае будет зависеть от расположения выводов ЖКИ и его подключения к MSP430.

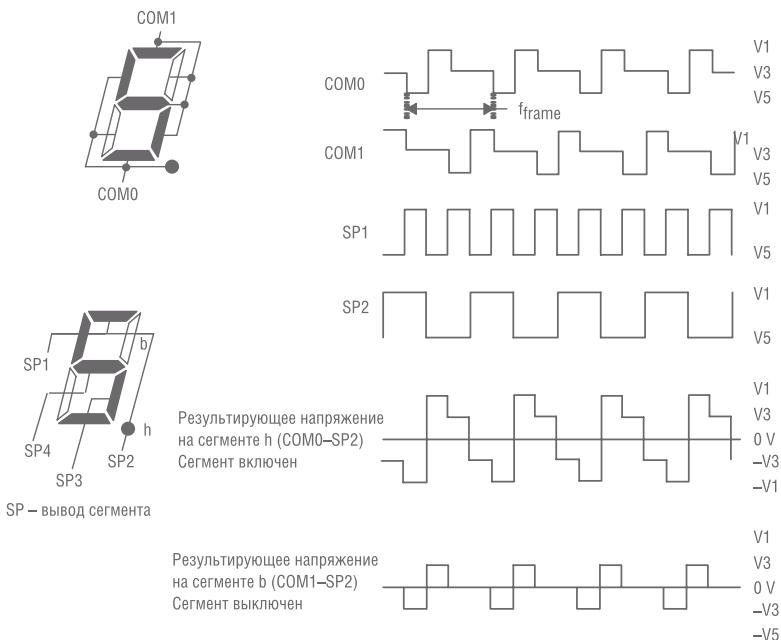
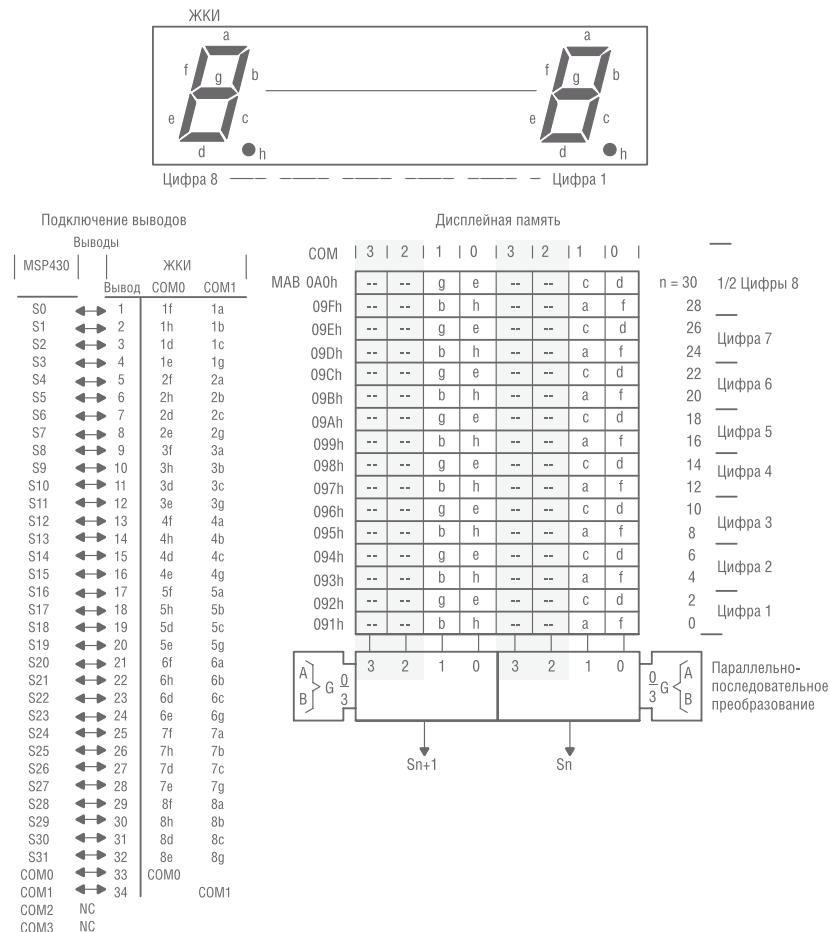


Рис. 18-5. Примеры осциллографм сигналов в режиме двойного мультиплексирования



**Рис. 18-6.** Пример ЖКИ в режиме двойного мультиплексирования

### Пример программы для режима двойного мультиплексирования

- ; Все восемь сегментов цифры как правило расположены
- ; в двух байтах дисплейной памяти при использовании
- ; режима двойного мультиплексирования.

a EQU 002h

<i>b</i>	<i>EQU</i>	020h
<i>c</i>	<i>EQU</i>	008h
<i>d</i>	<i>EQU</i>	004h
<i>e</i>	<i>EQU</i>	040h
<i>f</i>	<i>EQU</i>	001h
<i>g</i>	<i>EQU</i>	080h
<i>h</i>	<i>EQU</i>	010h

;Будет отображаться содержимое регистра Rx.

;Таблица представляет включенные сегменты

;в соответствии с содержимым Rx

```
MOV.B Table (Rx),RY ;Загрузить расположение
;сегментов в служебную
;область памяти.
;(Ry) = 0000 0000 gebh cda
```

```
MOV.B Ry,&LCDn ;Примечание:
;Записываются все биты
;байта памяти ЖКИ
```

```
RRA Ry ;(Ry) = 0000 0000 0geb hcda
RRA Ry ;(Ry) = 0000 0000 00ge bhcd
```

```
MOV.B Ry,&LCDn+1 ;Примечание:
;Записываются все биты
;байта памяти ЖКИ
```

.....

.....

.....

```
Table DB a+b+c+d+e+f ;отображает «0»
```

.....

```
DB a+b+c+d+e+f+g+ ;отображает «8»
```

.....

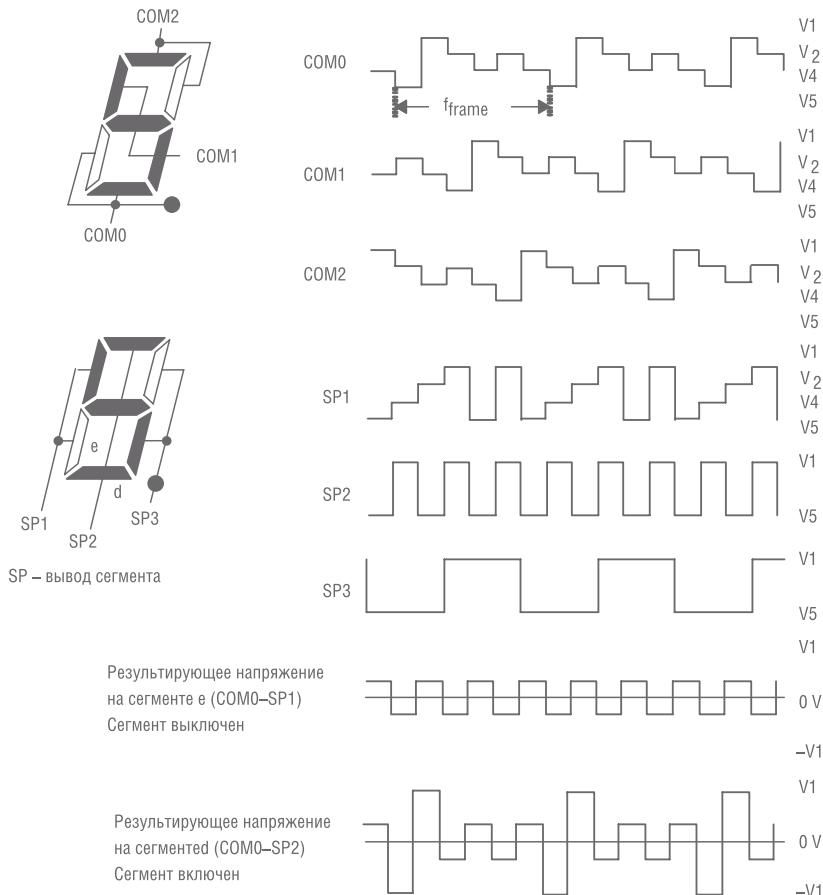
.....

DB

.....

### 18.2.8. Режим тройного мультиплексирования (3-Mux)

В режиме двойного мультиплексирования каждый сегментный выход микроконтроллера MSP430 управляет тремя сегментами ЖКИ, при этом используется три линии строк, COM0, COM1 и COM2. На рис. 18-7 показаны примеры осциллограмм сигналов в режиме двойного мультиплексирования.



**Рис. 18-7.** Примеры осциллографм сигналов в режиме тройного мультиплексирования

На рис. 18-8 показан пример ЖКИ в режиме тройного мультиплексирования, его подключение к MSP430 и карта сегментов. Обратите внимание, что это всего лишь пример, реальная карта сегментов в каждом конкретном случае будет зависеть от расположения выводов ЖКИ и его подключения к MSP430.

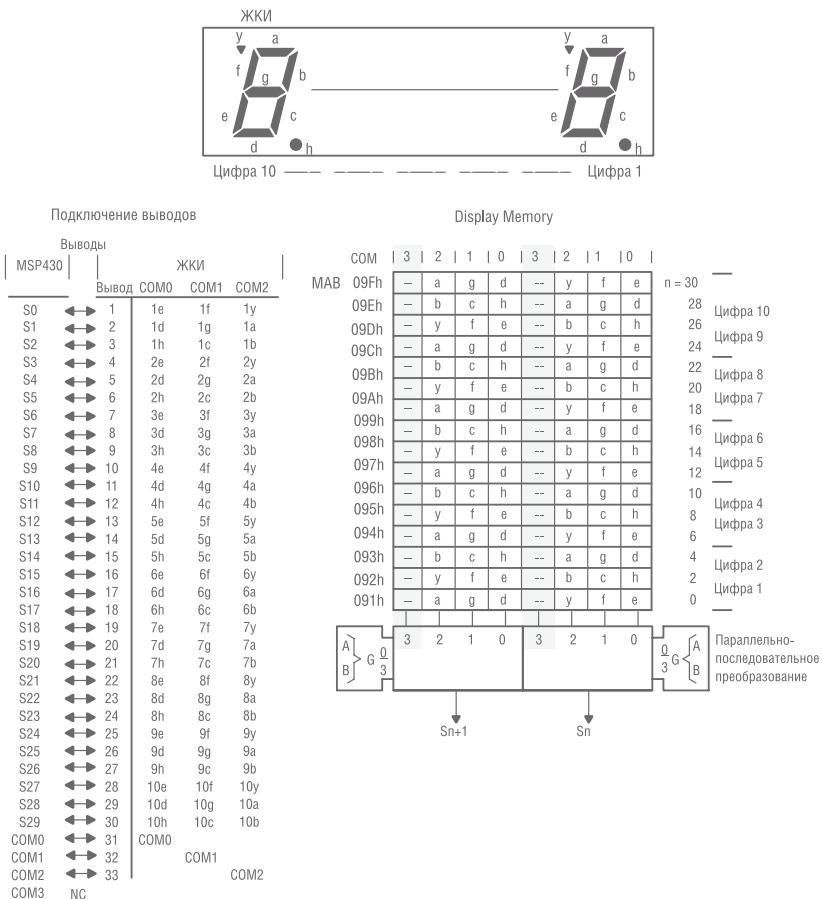


Рис. 18-8. Пример ЖКИ в режиме тройного мультиплексирования

**Пример программы для режима тройного мультиплексирования**

- При тройном мультиплексировании поддерживается 9 сегментов для каждой цифры. Они расположены в 1 1/2 байтах дисплейной памяти.

```

a    EQU      0040h
b    EQU      0400h
c    EQU      0200h
d    EQU      0010h

```

```

e    EQU    0001h
f    EQU    0002h
g    EQU    0020h
h    EQU    0100h
y    EQU    0004h

; Будет отображаться младшая цифра из регистра Rx.
; Таблица представляет включенные сегменты
; в соответствии с младшей цифрой регистра Rx.
; Регистр Ry используется в служебных целях

ODDDIG RLA      Rx    ;ЖКИ в режиме тройного
                        ;мультиплексирования имеет 9
                        ;сегментов на цифру
                        ;для отображаемых символов
                        ;требуется 16-битная таблица.

MOV    Table(Rx),Ry   ;Загрузка информации
                        ;о сегментах в служебную
                        ;память.

; (Ry) = 0000 0bch 0agd 0yfe
MOV.B Ry,&LCDn        ;записывает сегменты
                        ;'a, g, d, y, f, e'
                        ;Цифры № n (младший байт)

SWPB  Ry            ;(Ry) = 0agd 0yfe 0000 0bch
BIC.B #07h,&LCDn+1  ;записывает сегменты
                        ;'b, c, h'
                        ;Цифры № n (старший байт)

BIS.B Ry,&LCDn+1

.....
EVNDIG RLA      Rx    ;ЖКИ в режиме тройного
                        ;мультиплексирования имеет 9
                        ;сегментов на цифру для
                        ;отображаемых символов
                        ;требуется 16-битная таблица.

MOV    Table(Rx),Ry   ;Загрузка информации
                        ;о сегментах в служебную
                        ;память.

; (Ry) = 0000 0bch 0agd 0yfe
RLA   Ry            ;(Ry) = 0000 bch0 agd0 yfe0
RLA   Ry            ;(Ry) = 000b ch0a gd0y fe00

```

```

RLA    Ry          ; (Ry) = 00bc h0ag d0yf e000
RLA    Ry          ; (Ry) = 0bch 0agd 0yfe 0000
BIC.B #070h,&LCDn+1
BIS.B Ry,&LCDn+1      ; записывает сегменты
                        ; 'у, f, e'
                        ; Цифры № n+1 (младший байт)
SWPB  Ry          ; (Ry) = 0yfe 0000 0bch 0agd
MOV.B Ry,&LCDn+2      ; записывает сегменты
                        ; 'b, c, h, a, g, d'
                        ; Цифры № n+1 (старший байт)

```

.....

Table	DW	$a+b+c+d+e+f$	; отображает «0»
	DW	$b+c$	; отображает «1»

.....

DW	$a+e+f+g$	; отображает «F»
----	-----------	------------------

### 18.2.9. Режим четырёхкратного мультиплексирования (4-Mux)

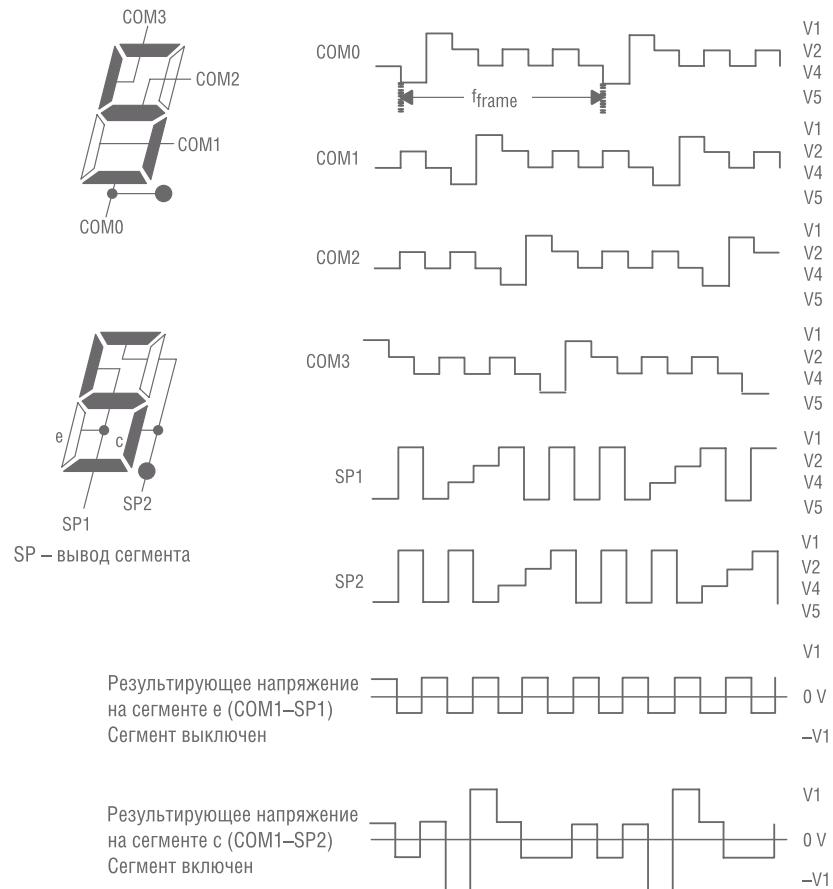
В режиме четырёхкратного мультиплексирования каждый сегментный выход микроконтроллера MSP430 управляет четырьмя сегментами ЖКИ, при этом используются все четыре линии строк, COM0, COM1, COM2 и COM3. На Рис. 18-9 показаны примеры осцилограмм сигналов в режиме четырёхкратного мультиплексирования.

На рис. 18-10 показан пример ЖКИ в режиме четырёхкратного мультиплексирования, его подключение к MSP430 и карта сегментов. Обратите внимание, что это всего лишь пример, реальная карта сегментов в каждом конкретном случае будет зависеть от расположения выводов ЖКИ и его подключения к MSP430.

#### Пример программы для режима четырёхкратного мультиплексирования

; В режиме четырёхкратного мультиплексирования  
; поддерживается восемь сегментов для каждой цифры.  
; Все восемь сегментов для каждой цифры,  
; как правило, расположены в одном байте дисплейной  
; памяти

a	EQU	080h
b	EQU	040h
c	EQU	020h
d	EQU	001h
e	EQU	002h
f	EQU	008h



**Рис. 18-9.** Примеры осциллографм сигналов в режиме четырёхкратного мультиплексирования

```

g      EQU      004h
h      EQU      010h
;
;Будет отображаться младшая цифра из регистра Rx.
;Таблица представляет включенные сегменты
;в соответствии с младшей цифрой регистра Rx.
;

```

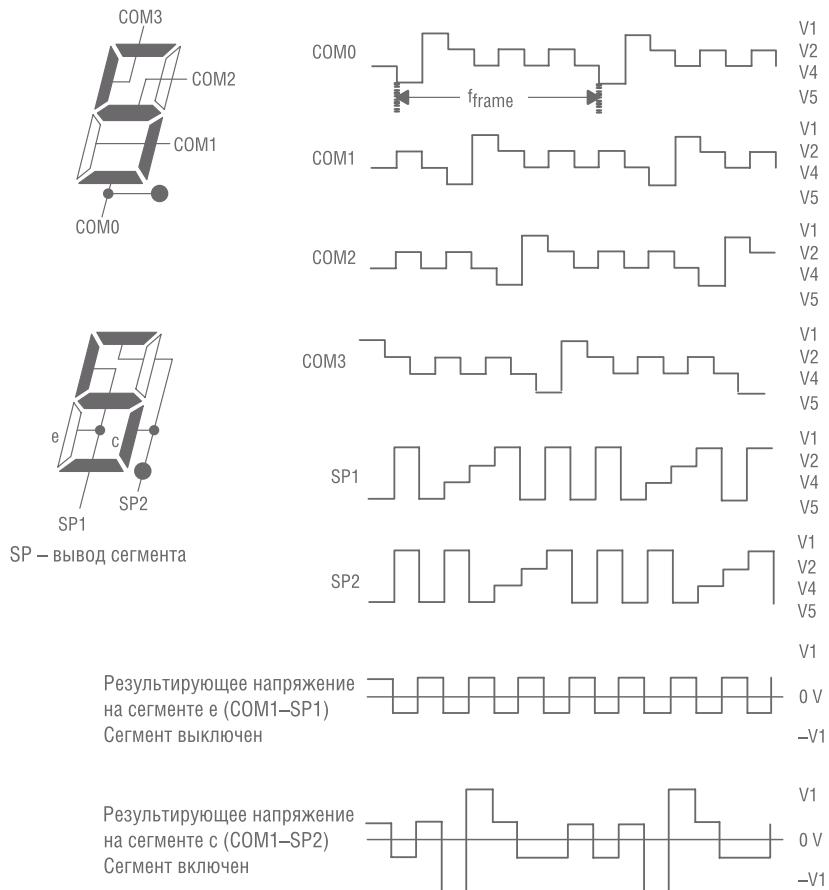


Рис. 18-10. Пример ЖКИ в режиме четырёхкратного мультиплексирования

```
MOV.B Table(Rx), &LCDn      ; n = 1 ..... 15
                                ; все восемь сегментов
                                ; записываются
                                ; в дисплейную память
```

.....  
.....

Table DB a+b+c+d+e+f ; отображает «0»

<i>DB</i>	<i>b+c</i>	<i>; отображает «1»</i>
.....		
<i>DB</i>	<i>b+c+d+e+g</i>	<i>; отображает «d»</i>
<i>DB</i>	<i>a+d+e+f+g</i>	<i>; отображает «E»</i>
<i>DB</i>	<i>a+e+f+g</i>	<i>; отображает «F»</i>

## 18.3. Регистры модуля контроллера ЖКИ

Регистры модуля контроллера ЖКИ перечислены в таблице 18-1.

**Таблица 18-1**

Регистр	Краткое название	Тип	Адрес	Начальное состояние
Регистр управления контроллером модуля ЖКИ	LCDCTL1	Чтение/запись	090h	Обнулён по сбросу РУС
Регистр дисплейной памяти 1	LCDM1	Чтение/запись	091h	Не изменяется
Регистр дисплейной памяти 2	LCDM2	Чтение/запись	092h	Не изменяется
Регистр дисплейной памяти 3	LCDM3	Чтение/запись	093h	Не изменяется
Регистр дисплейной памяти 4	LCDM4	Чтение/запись	094h	Не изменяется
Регистр дисплейной памяти 5	LCDM5	Чтение/запись	095h	Не изменяется
Регистр дисплейной памяти 6	LCDM6	Чтение/запись	096h	Не изменяется
Регистр дисплейной памяти 7	LCDM7	Чтение/запись	097h	Не изменяется
Регистр дисплейной памяти 8	LCDM8	Чтение/запись	098h	Не изменяется
Регистр дисплейной памяти 9	LCDM9	Чтение/запись	09Ah	Не изменяется
Регистр дисплейной памяти 10	LCDM10	Чтение/запись	09Bh	Не изменяется
Регистр дисплейной памяти 11	LCDM11	Чтение/запись	09Ch	Не изменяется
Регистр дисплейной памяти 12	LCDM12	Чтение/запись	09Dh	Не изменяется

Таблица 18-1 (Окончание)

Регистр	Краткое название	Тип	Адрес	Начальное состояние
Регистр дисплейной памяти 13	LCDM13	Чтение/запись	09Eh	Не изменяется
Регистр дисплейной памяти 14	LCDM14	Чтение/запись	09Fh	Не изменяется
Регистр дисплейной памяти 15	LCDM15	Чтение/запись	0A0h	Не изменяется
Регистр дисплейной памяти 16	LCDM16	Чтение/запись	0A1h	Не изменяется
Регистр дисплейной памяти 17	LCDM17	Чтение/запись	0A2h	Не изменяется
Регистр дисплейной памяти 18	LCDM18	Чтение/запись	0A3h	Не изменяется
Регистр дисплейной памяти 19	LCDM19	Чтение/запись	0A4h	Не изменяется
Регистр дисплейной памяти 20	LCDM20	Чтение/запись	0A5h	Не изменяется

**LCDCTL1, Регистр управления контроллером модуля ЖКИ**

7	6	5	4	3	2	1	0
						Не используется	LCDON
<b>LCDPx</b>			<b>LCDMXx</b>		<b>LCDSON</b>		
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

<b>LCDPx</b>	Биты 7-5	Выбор портов модуля контроллера ЖКИ. Эти биты определяют функцию вывода – выходной сигнал контроллера ЖКИ илипорт ввода/вывода. Биты влияют ТОЛЬКО на выводы с мультиплексируемыми функциями. Обособленные выводы контроллера ЖКИ всегда сохраняют свою функцию. 000 – Нет мультиплексируемых выводов с функцией выхода контроллера ЖКИ 001 – S0-S15 – выходы контроллера ЖКИ 010 S0-S19 – выходы контроллера ЖКИ 011 – S0-S23 – выходы контроллера ЖКИ 100 – S0-S27 – выходы контроллера ЖКИ 101 – S0-S31 – выходы контроллера ЖКИ 110 – S0-S35 – выходы контроллера ЖКИ 111 – S0-S39 – выходы контроллера ЖКИ

<b>LCDMXx</b>	Биты 4-3	Выбор режима мультиплексии. Этими битами определяется режим работы ЖКИ. 00 – Статический 01 – Двойная мультиплексия (2-mux) 10 – Тройная мультиплексия (3-mux) 11 – Четырёхкратная мультиплексия (4-mux)
<b>LCDSON</b>	Бит 2	Включение сегментов ЖКИ. Этим битом реализуется режим мигания. При этом тактовый генератор ЖКИ и R33 остаются включенными, а все сегменты выключаются. 0 – Все сегменты ЖКИ выключены 1 – Сегменты ЖКИ включены либо выключены в зависимости от настроек дисплейной памяти
<b>Не используется</b>	Бит 1	Не используется
<b>LCDON</b>	Бит 0	Включение контроллера ЖКИ. Этим битом управляются тактовый генератор и R33. 0 – тактовый генератор и R33 выключены 1 – тактовый генератор и R33 включены

# MSP430x4xxFamily

АЦП12

---

*Раздел XIX.*



## АЦП12

Модуль АЦП12 представляет собой высокоеффективный 12-разрядный аналого-цифровой преобразователь. В этом разделе описывается АЦП12. АЦП12 реализован в устройствах MSP430x43x и MSP430x44x.

### 19.1. Введение в АЦП12

Модуль АЦП12 представляет собой высокоеффективный 12-разрядный аналого-цифровой преобразователь. В этом разделе описывается АЦП12. АЦП12 реализован в устройствах MSP430x43x и MSP430x44x.

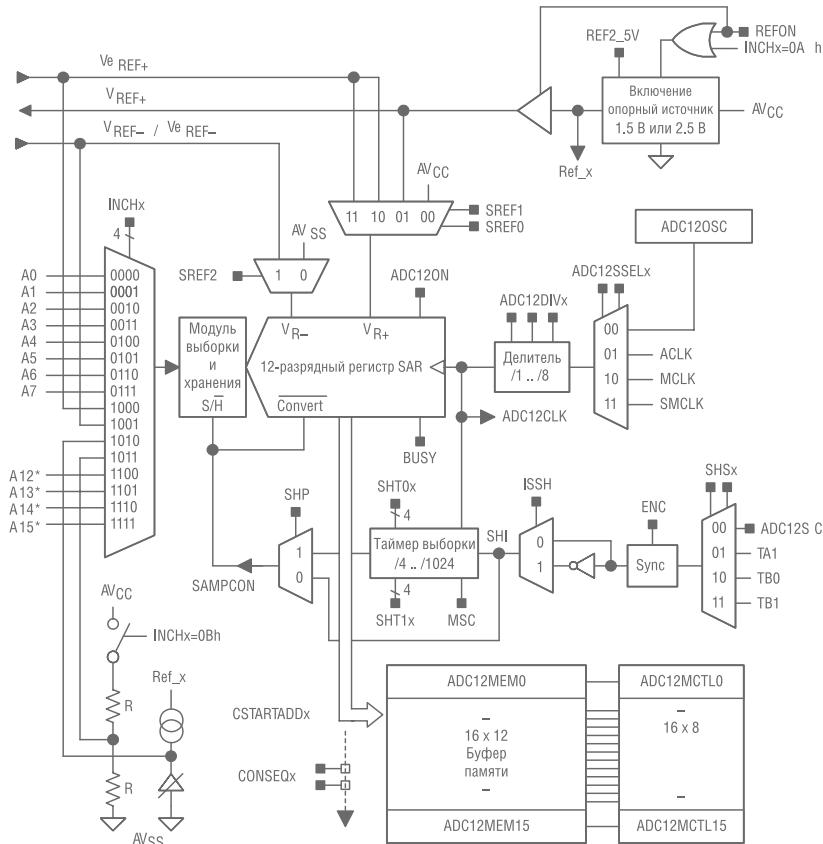
#### АЦП12 обладает следующими возможностями:

- 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 (и 12 в устройствах MSP430FG43x)
- Каналы преобразования для внутреннего температурного датчика,  $AV_{cc}$  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. 19-1.

### 19.2. Функционирование АЦП12

Модуль АЦП12 конфигурируется программным обеспечением пользователя. Настройка и работа АЦП12 рассматриваются в следующих разделах.



\* Только в устройствах MSP430FG43x

Рис. 19-1. Блок-схема АЦП12

### 19.2.1. 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 пре-

образованиями. Формула преобразования для результата АЦП NADC выглядит следующим образом:

Ядро АЦП12 конфигурируется двумя управляющими регистрами: ADC12CTL0 и ADC12CTL1. Ядро включается битом ADC12ON. Если ADC12 не

$$N_{ADC} = 4095 \times \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}}$$

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.

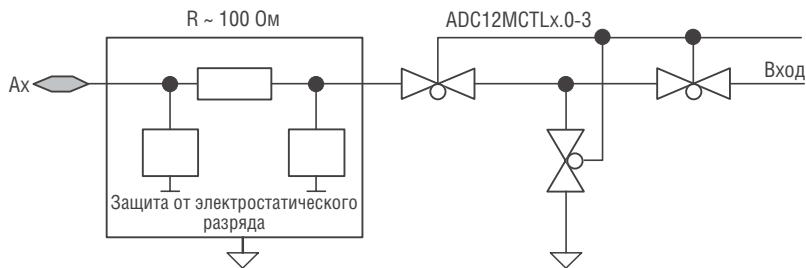
ADC12OSC, генерируемый внутренне, лежит в диапазоне 5 МГц, но варьируется в зависимости от конкретного устройства, напряжения питания и температуры. См. справочное руководство конкретного устройства для уточнения значения ADC12OSC.

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.

### 19.2.2. 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. 19-2. 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.

АЦП12 использует метод перераспределения заряда. Когда входы внутренне переключаются, переключение может привести к переходным процессам



**Рис. 19-2.** Аналоговый мультиплексор

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  $V_{cc}$  к GND. Этот паразитный ток появляется, если величина входного напряжения находится около переходного уровня ячейки. Отключение буфера ножки порта устраниет 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
```

### 19.2.3. 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  $A_{VSS}$ . 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 мА 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. 19-11.

Внешние опорные источники могут быть задействованы для VR+ и VR- через выводы  $V_{e_{REF+}}$  и  $V_{e_{REF-}}$ , соответственно.

#### 19.2.4. 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. 19-3.

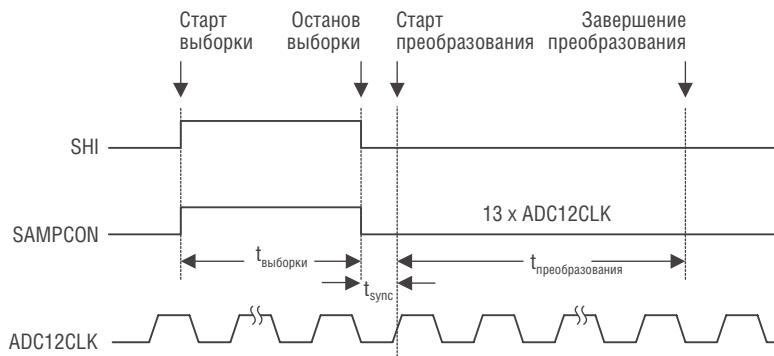


Рис. 19-3. Расширенный режим выборки

### 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  $t_{sampe}$  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. 19-4.

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.

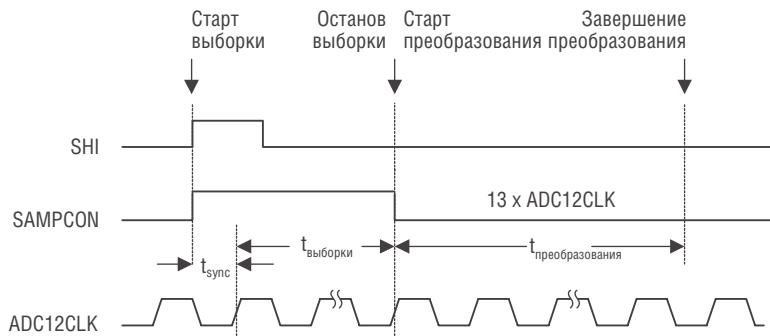
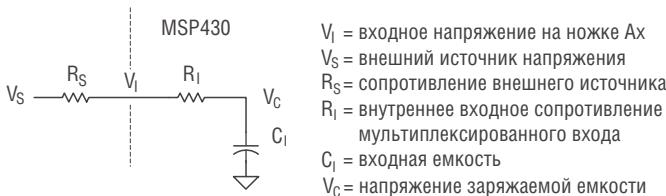


Рис. 19-4. Импульсный режим выборки

## 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 низких частот в течение периода квантования  $t_{\text{sample}}$ , как показано на рис. 19-5. 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  $\frac{1}{2}$  младшего бита источника напряжения  $V_s$  для получения точного 12-разрядного преобразования.



**Рис. 19-5.** Эквивалентная схема аналогового входа

Сопротивление источника  $R_s$  и  $R_i$  влияет на  $t_{\text{sample}}$ . Следующее выражение может быть использовано для вычисления минимального времени выборки  $t_{\text{sample}}$  при 12-разрядном преобразовании:

$$t_{\text{sample}} > (R_s + R_i) \times \ln(2^{13}) \times C_i + 800 \text{ нс}$$

При подстановке значений  $R_i$  и  $C_i$ , указанных выше, уравнение приобретает следующий вид:

$$t_{\text{sample}} > (R_s + 2 \text{ кОм}) \times 9,011 \times 40 \text{ пФ} + 800 \text{ нс}$$

К примеру, если  $R_s$  равно 10 кОм,  $t_{\text{sample}}$  должно быть больше 5,13 мкс.

### 19.2.5. 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.

### 19.2.6. Режимы преобразований АЦП12

АЦП12 имеет четыре режима работы, выбираемые битами CONSEQx так, как описано в таблице 19-1.

**Таблица 19-1. 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.

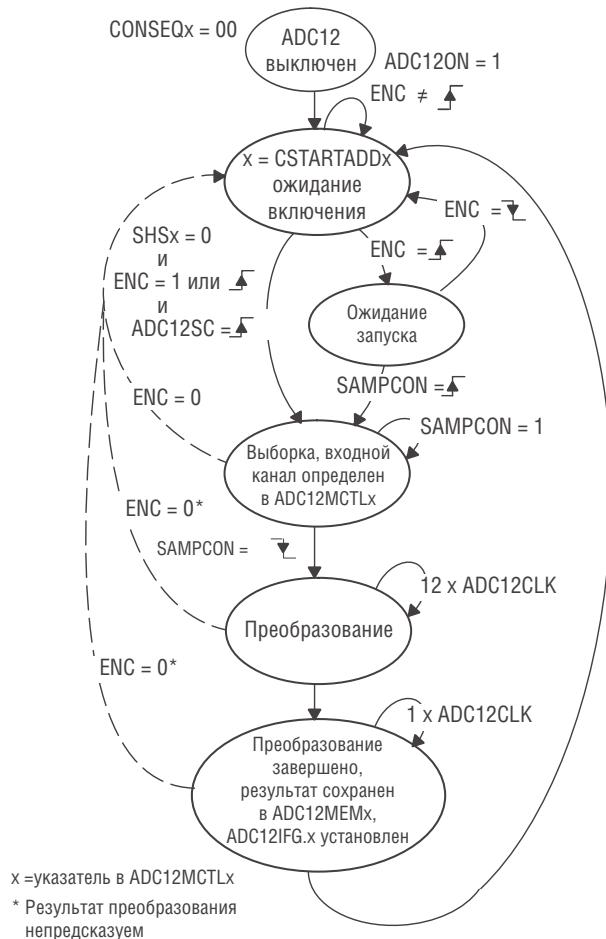
#### Одноканальный режим с одиночным преобразованием

В одном канале однократно выполняется выборка и преобразование. Результат АЦП записывается в регистр ADC12MEMx, определенный битами CSTARTADDx. На рис. 19-6 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-

чина с ADCMEMx, определенным битами CSTARTADDx. Последовательность останавливается после измерения в канале с установленным битом EOS. На рис. 19-7 показан режим последовательности каналов. Если последователь-



**Рис. 19-6.** Одноканальный режим одиночного преобразования

ность запускает ADC12SC, поочередные последовательности могут запускаться битом ADC12SC. Когда используется другой источник запуска, ENC должен переключаться между каждой последовательностью.

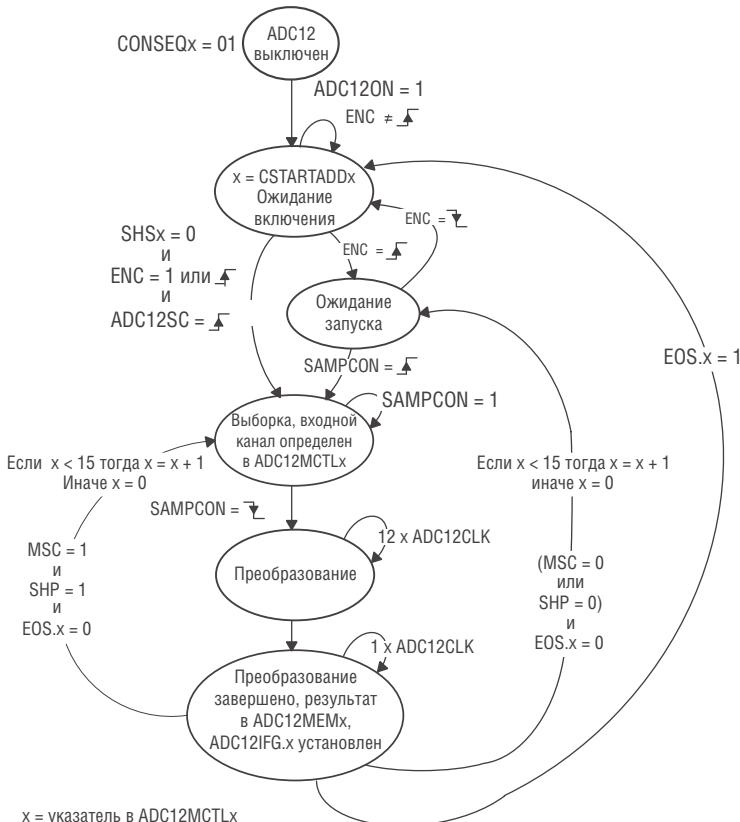


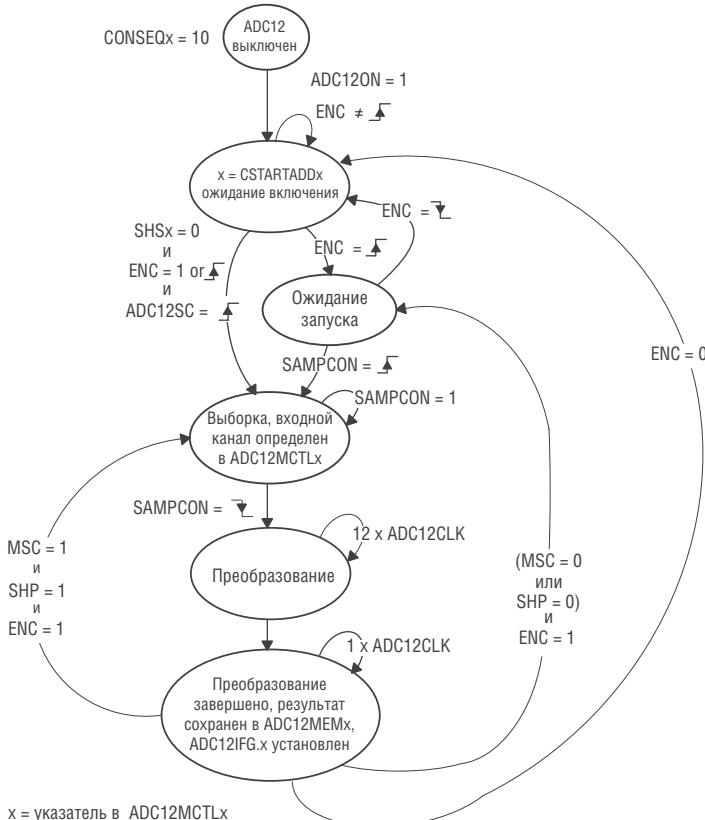
Рис. 19-7. Режим последовательности каналов

### 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. 19-8 показан повторяющийся одноканальный режим.

### Режим повторяющейся последовательности каналов

Непрерывно выполняются выборка и преобразование последовательности каналов. Результат АЦП записывается в память преобразований, начиная с



**Рис. 19-8.** Повторяющийся одноканальный режим

ADC12MEM $x$ , определенного битами CSTARTADD $x$ . Последовательность останавливается после измерения в канале с установленным битом EOS и стартует снова по следующему сигналу запуска. На рис. 19-9 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,  $\text{CONSEQ}_x > 1$  и используется таймер выборок, первый фронт сигнала SHI за-

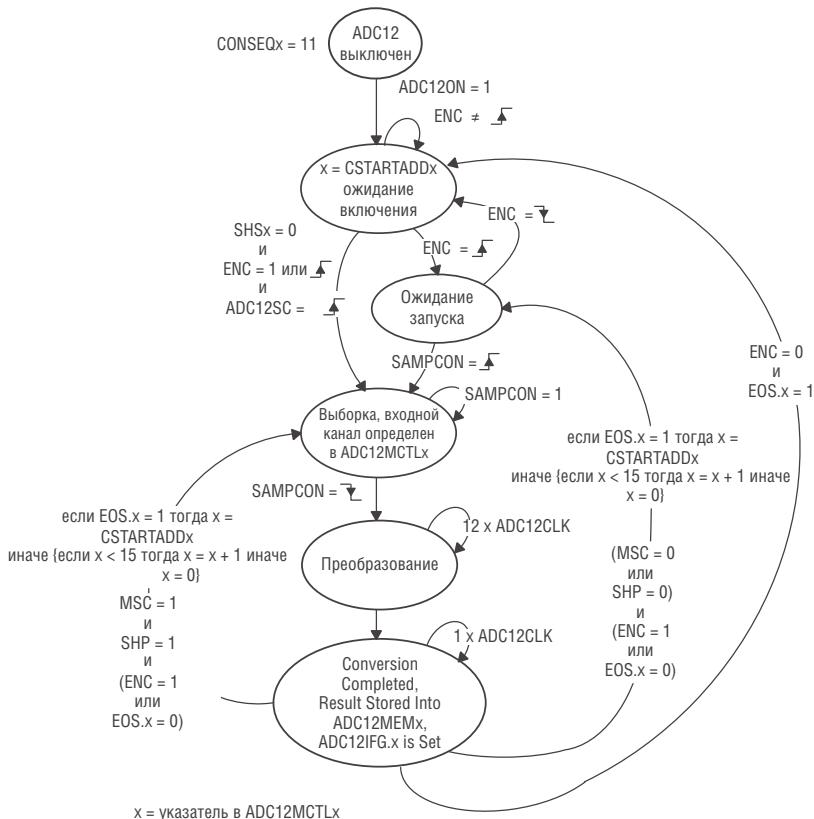


Рис. 19-9. Режим повторяющейся последовательности каналов

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. Рекомендуется 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 CONSEQ<sub>x</sub>=0 и сбросом бита ENC. Данные преобразования будут ненадежны.

#### **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.

#### **19.2.7. Использование интегрированного температурного датчика**

При использовании имеющегося на кристалле температурного датчика пользователь выбирает аналоговый входной канал INCH<sub>x</sub>=1010. Любая другая конфигурация рассматривается как выбор внешнего канала, включая выбор опорного источника, выбор памяти преобразований и т.д.

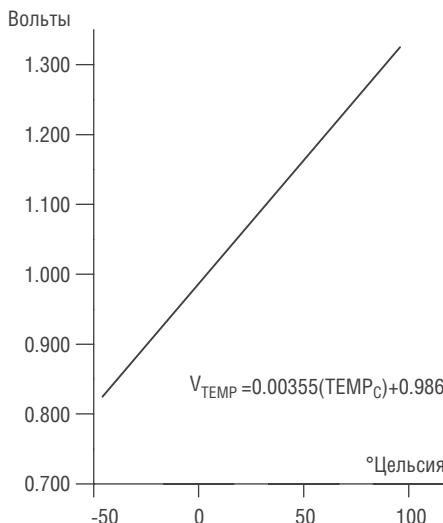
Типичная передаточная функция температурного датчика показана на рис. 19-10. Если используется температурный датчик, период выборки должен быть больше 30 мкС. Ошибка смещения температурного датчика может быть большой и может потребоваться калибровка для большинства приложений. См. справочные данные конкретного 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.

#### **19.2.8. 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. 19-10.** 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. 19-11 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.

### 19.2.9. Прерывания АЦП12

АЦП12 имеет 18 источников прерывания:

- ADC12IFG0-ADC12IFG15

- ADC12OV, переполнение AD12MEMx
- ADC12TOV, переполнение времени преобразования АЦП12

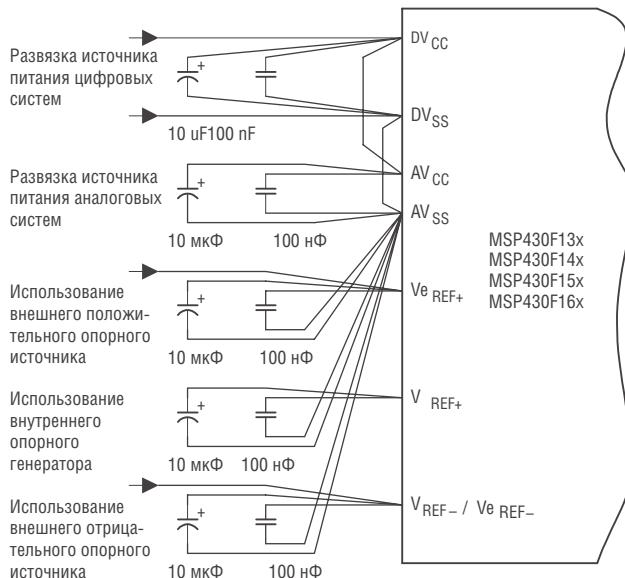


Рис. 19-11. Заземление АЦП12 и устранение помех

Биты ADC12IFGx устанавливаются, когда в их соответствующие регистры памяти ADC12MEMx загружается результат преобразования. Если соответствующий бит ADC12IE<sub>x</sub> 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 РС 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 РС          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

```

## 19.3. Регистры АЦП12

Регистры АЦП12 приведены в таблице 19-2.

**Таблица 19-2. 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
<b>SHT1x</b>				<b>SHT0x</b>				
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4		3	2	1	0
<b>MSC</b>	<b>REF2_5V</b>	<b>REFON</b>	<b>ADC12ON</b>	<b>ADC12 OVIE</b>	<b>ADC12 TOVIE</b>	<b>ENC</b>	<b>ADC12SC</b>	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Модифицируется, только когда ENC=0

<b>SHT1x</b>	<b>Биты 15-12</b>	Время выборки-хранения. Эти биты определяют число циклов ADC12CLK в периоде выборки для регистров с ADC12MEM8 по ADC12MEM15.	
		Время выборки-хранения. Эти биты определяют число циклов ADC12CLK в периоде выборки для регистров с ADC12MEM0 по ADC12MEM7.	
<b>SHT0x</b>	<b>Биты 11-8</b>	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

<b>MSC</b>	<b>Бит 7</b>	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
<b>REF2_5V</b>	<b>Бит 6</b>	Генератор опорного напряжения. REFON также должен быть установлен. 0 – 1.5 В 1 – 2.5 В
<b>REFON</b>	<b>Бит 5</b>	Включение опорного генератора. 0 – Опорный генератор выключен 1 – Опорный генератор включен.
<b>ADC12ON</b>	<b>Бит 4</b>	Включение АЦП12 0 – АЦП12 выключен 1 – АЦП12 включен
<b>ADC12OVIE</b>	<b>Бит 3</b>	Разрешение прерывания по переполнению ADC12MEMx. Для разрешения прерываний также должен быть установлен бит GIE. 0 – Прерывание по переполнению запрещено 1 – Прерывание по переполнению разрешено
<b>ADC12TOVIE</b>	<b>Бит 2</b>	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
<b>ENC</b>	<b>Бит 1</b>	Разрешение преобразования 0 – Преобразование в АЦП12 запрещено 1 – Преобразование в АЦП12 разрешено
<b>ADC12SC</b>	<b>Бит 0</b>	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**

15	14	13	12	11	10	9	8
<b>CSTARTADDx</b>				<b>SHSx</b>		<b>SHP</b>	<b>ISSH</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4		3	2	1	0
ADC12DIVx				ADC12SSELx	CONSEQx			ADC12 BUSY
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)

	Модифицируется, только когда ENC=0
--	------------------------------------

<b>CSTARTADDx</b>	<b>Биты 15-12</b>	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.
<b>SHSx</b>	<b>Биты 11-10</b>	Выбор источника выборки-хранения. 00 – Бит ADC12SC 01 – Выход 1 Таймера A 10 – Выход 0 Таймера B 11 – Выход 1 Таймера B
<b>SHP</b>	<b>Бит 9</b>	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.
<b>ISSH</b>	<b>Бит 8</b>	Инвертирование сигнала выборки-хранения 0 – Входной сигнал выборки не инвертирован 1 – Входной сигнал выборки инвертирован
<b>ADC12DIVx</b>	<b>Биты 7-5</b>	Тактовый делитель АЦП12 000 – /1 001 – /2 010 – /3 011 – /4 100 – /5 101 – /6 110 – /7 111 – /8
<b>ADC12SSELx</b>	<b>Биты 4-3</b>	Выбор источника тактирования АЦП12 00 – ADC12OSC 01 – ACLK 10 – MCLK 11 – SMCLK
<b>CONSEQx</b>	<b>Биты 2-1</b>	Выбор режима преобразования 00 – Одноканальный, с одним преобразованием 01 – Последовательность каналов 10 – Повторный одноканальный 11 – Повторяющаяся последовательность каналов

<b>ADC12BUSY</b>	<b>Бит 0</b>	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
<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>		<b>Результаты преобразования</b>			
r0	r0	r0	r0		rw	rw	rw	rw
7	6	5	4		3	2	1	0
<b>Результаты преобразования</b>					rw	rw	rw	rw

<b>Результаты преобразования</b>	<b>Биты 15-0</b>	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
<b>EOS</b>		<b>SREFx</b>			<b>INCHx</b>			
rw-(0)	rw-(0)	rw-(0)	rw-(0)		rw-(0)	rw-(0)	rw-(0)	rw-(0)
Модифицируется, только когда ENC=0								

<b>EOS</b>	<b>Биты 7</b>	Конец последовательности. Показывает последнее преобразование в последовательности. 0 – Не конец последовательности 1 – Конец последовательности
<b>SREFx</b>	<b>Биты 6-4</b>	Выбор опорного источника 000 – $V_{R+} = AV_{CC}$ и $V_{R-} = AV_{SS}$ 001 – $V_{R+} = V_{REF+}$ и $V_{R-} = AV_{SS}$ 010 – $V_{R+} = V_{REF+}$ и $V_{R-} = AV_{SS}$ 011 – $V_{R+} = V_{REF+}$ и $V_{R-} = AV_{SS}$ 100 – $V_{R+} = AV_{CC}$ и $V_{R-} = V_{REF-}/V_{REF+}$ 101 – $V_{R+} = V_{REF+}$ и $V_{R-} = V_{REF-}/V_{REF+}$ 110 – $V_{R+} = V_{REF+}$ и $V_{R-} = V_{REF-}/V_{REF+}$ 111 – $V_{R+} = V_{REF+}$ и $V_{R-} = V_{REF-}/V_{REF+}$

<b>INCHx</b>	<b>Биты 3-0</b>	Выбор входного канала 0000 – A0 0001 – A1 0010 – A2 0011 – A3 0100 – A4 0101 – A5 0110 – A6 0111 – A7 1000 – $V_{e_{REF+}}$ 1001 – $V_{e_{REF-}}/V_{e_{REF}}$ 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
<b>ADC12IE15</b>	<b>ADC12IE14</b>	<b>ADC12IE13</b>	<b>ADC12IE12</b>	<b>ADC12IE11</b>	<b>ADC12IE10</b>	<b>ADC12IE9</b>	<b>ADC12IE8</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
<b>ADC12IE7</b>	<b>ADC12IE6</b>	<b>ADC12IE5</b>	<b>ADC12IE4</b>	<b>ADC12IE3</b>	<b>ADC12IE2</b>	<b>ADC12IE1</b>	<b>ADC12IE0</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

<b>ADC12IEx</b>	<b>Биты 15-0</b>	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
<b>ADC12 IFG15</b>	<b>ADC12 IFG14</b>	<b>ADC12 IFG13</b>	<b>ADC12 IFG12</b>	<b>ADC12 IFG11</b>	<b>ADC12 IFG10</b>	<b>ADC12 IE9</b>	<b>ADC12 IFG8</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
<b>ADC12 IFG7</b>	<b>ADC12 IFG6</b>	<b>ADC12 IFG5</b>	<b>ADC12 IFG4</b>	<b>ADC12 IFG3</b>	<b>ADC12 IFG2</b>	<b>ADC12 IFG1</b>	<b>ADC12 IFG0</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

<b>ADC12IFGx</b>	<b>Биты 15-0</b>	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
<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
<b>0</b>	<b>0</b>			<b>ADC12IVx</b>			<b>0</b>
r0	r0	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r0

<b>ADC12IVx</b>	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	

<b>ADC12IVx</b>	<b>Биты 15-0</b>	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

# MSP430x4xxFamily

## Модуль АЦП SD16

*Раздел XX.*



## Модуль АЦП SD16

Модуль АЦП SD16 представляет собой многоканальный 16-битный сигма-дельта аналогово-цифровой преобразователь. В это главе описана работа модуля АЦП SD16. Модуль SD16 присутствует в микроконтроллерах MSP430FE42x и MSP430F42x.

### 20.1. Модуль АЦП SD16 – введение

Модуль АЦП SD16 содержит до трёх независимых сигма-дельта аналого-цифровых преобразователей и встроенный источник опорного напряжения. Каждый из АЦП имеет до восьми полностью дифференциальных переключаемых каналов, в том числе встроенный датчик температуры. АЦП созданы на базе дельта-сигма модуляторов второго порядка с передискретизацией и цифровых децимирующих фильтров. Для прореживания (децимации) используются фильтры comb-типа с программируемым коэффициентом прореживания до 256. Дополнительная фильтрация может быть осуществлена программно.

**Модуль АЦП SD16 обладает следующими свойствами:**

- 16-битная сигма-дельта архитектура;
- До 3-х независимо функционирующих каналов АЦП;
- До 8 переключаемых дифференциальных аналоговых входов на канал;
- Программно включаемый встроенный источник опорного напряжения (1,2 В);
- Программный выбор встроенного или внешнего источника опорного напряжения;
- Встроенный датчик температуры, доступный для всех каналов;
- Входная частота модулятора до 1,048576 МГц (частота выборки  $f_{\text{SAMPLE}}$  до 4096 Гц при передискретизации 256x);
- Программно выбираемый режим преобразования с пониженным энергопотреблением.

Блок-схема АЦП12 показана на рис. 20-1.

### 20.2. Работа модуля АЦП SD16

Модуль АЦП SD16 конфигурируется при помощи пользовательского программного обеспечения. Настройка и функционирование модуля будут подробно рассмотрены ниже.

#### 20.2.1. Ядро аналого-цифрового преобразователя

Аналого-цифровое преобразование осуществляется однобитным сигма-дельта модулятором второго порядка. Однобитный компаратор в составе

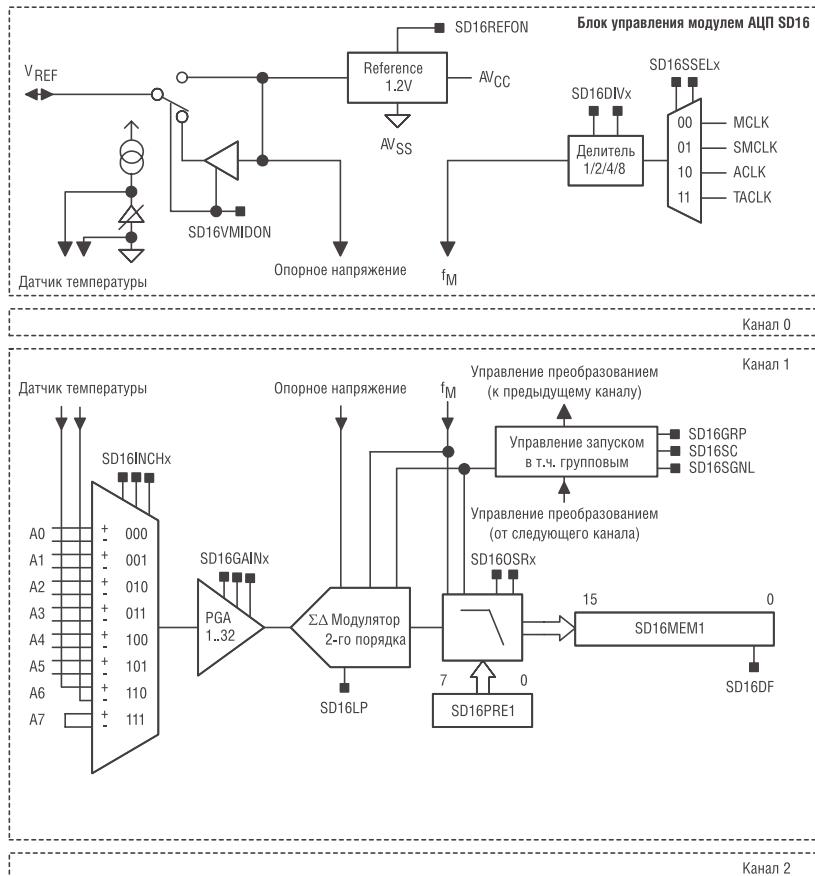


Рис. 20-1. Блок схема модуля АЦП SD16

модулятора осуществляет квантование сигнала с частотой модулятора  $f_M$ . Получающийся в результате однобитовый поток усредняется цифровым фильтром, формируя результат преобразования.

### 20.2.2. Цифровой фильтр

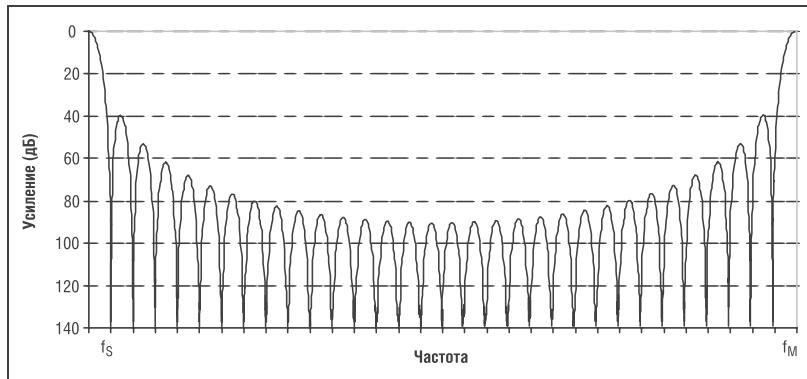
Цифровой фильтр обрабатывает 1-битный поток данных, поступающий с модулятора при помощи SINC<sup>3</sup> фильтра comb-типа. Передаточная характеристика такого фильтра может быть представлена в Z-плоскости как:

$$H(z) = \left( \frac{1}{OSR} \times \frac{1 - z^{-OSR}}{1 - z^{-1}} \right)^3$$

а в частотной области как:

$$H(f) = \left[ \frac{\text{sinc}\left(\text{OSR}\pi \frac{f}{f_M}\right)}{\text{sinc}\left(\pi \frac{f}{f_M}\right)} \right]^3 = \left( \frac{1}{OSR} \times \frac{\sin\left(\text{OSR} \times \pi \times \frac{f}{f_M}\right)}{\sin\left(\pi \times \frac{f}{f_M}\right)} \right)^3$$

где степень передискретизации, OSR, это отношение частоты модулятора  $f_M$  к частоте выборки  $f_s$ . На рис. 20-2 показана частотная характеристика фильтра при OSR равном 32. Первый срез фильтра приходится на частоту  $f_s = f_M / OSR$ . Частота среза может быть подстроена изменением частоты модулятора,  $f_M$ , при помощи бит SD16SELx и SD16DIVx или степени передискретизации при помощи бит SD16OSRx. Цифровой фильтр для каждого из разрешённых каналов завершает прореживание (декимацию) цифрового потока данных и выдаёт результат преобразования в соответствующий регистр SD16MEMx с частотой выборки  $f_s$ .



**Рис. 20-2.** Частотная характеристика фильтра при OSR равном 32

На рис. 20-3 показаны пошаговый отклик фильтра и точки преобразования. Для корректного пошагового преобразования после старта преобразования, следует выждать определенное время, называемое временем установ-

ления (settling time) до того момента, пока результат преобразования не станет корректным. Битами SD16INTDLYx можно задать время установления достаточное, для единовременного изменения сигнала на входе АЦП в пределах полной шкалы измерения. Если шаг преобразований синхронизирован с децимацией цифрового фильтра, корректный результат на выходе появится после третьего преобразования. Асинхронность этих сигналов потребует дополнительного цикла преобразования.

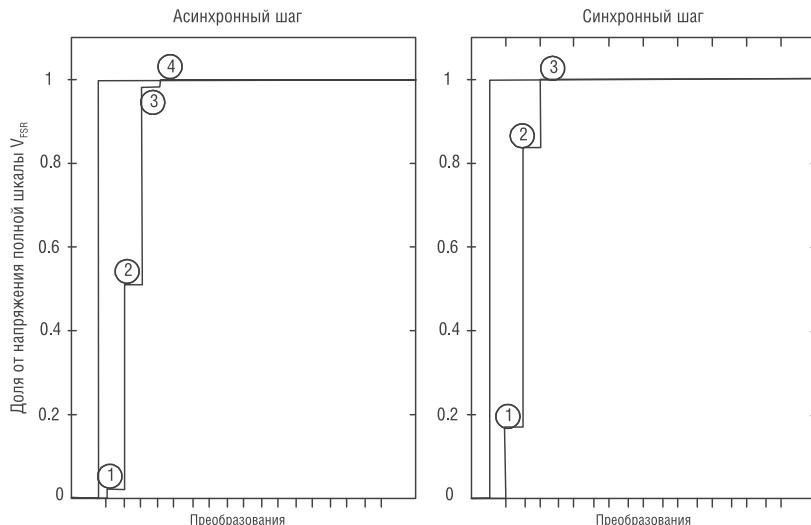


Рис. 20-3. Пошаговый отклик цифрового фильтра и точки преобразования

### 20.2.3. Диапазон аналогового сигнала и усилитель с программируемым коэффициентом усиления (PGA)

Полный диапазон входного аналогового сигнала для каждой пары входов зависит от установленного коэффициента усилителя по каждому каналу. Максимально допустимый диапазон –  $\pm V_{FSR}$ , где  $V_{FSR}$  определяется как:

$$V_{FSR} = \frac{V_{REF}/2}{GAIN_{PGA}}$$

При использовании опорного напряжения величиной 1.2 В, максимальный диапазон входного сигнала при усилении 1 равен:

$$\pm V_{FSR} = \frac{1.2V/2}{1} = \pm 0.6V$$

Конкретные значения допустимых значений входных сигналов см. в документации на конкретный тип МК.

#### **20.2.4. Источник опорного напряжения**

Модуль АЦП SD16 содержит встроенный источник опорного напряжения величиной 1.2 В, который может быть использован для всех каналов модуля SD16 и включается битом SD16REFON. Для снижения помех при использовании встроенного источника опорного напряжения рекомендуется использование внешнего конденсатора ёмкостью в 100 нФ, подключенного между выводами  $V_{REF}$  и  $AV_{SS}$ . Опорное напряжение также может использоваться вне микроконтроллера, если это разрешено битом SD16VMIDON. Выход ИОН буферизован, нагрузочный ток буфера до 1mA. При использовании опорного напряжения вне микроконтроллера, требуется внешний конденсатор ёмкостью в 470 нФ, подключенный между выводами  $V_{REF}$  и  $AV_{SS}$ . Уточнённые параметры см. в документации на конкретный тип МК.

В том случае, если биты SD16REFON и SD16VMIDON обнулены, АЦП может использовать внешний ИОН, подключенный к выводу  $V_{REF}$ .

#### **20.2.5. Регистры памяти преобразований: SD16MEMx**

Каждому каналу модуля SD16 сопоставлен регистр памяти SD16MEMx. На каждом шаге децимации цифрового фильтра результат преобразования записываются в соответствующий регистр SD16MEMx. При перезаписи нового значения в регистр SD16MEMx устанавливается бит SD16IFG. Бит SD16IFG обнуляется автоматически при чтении из регистра SD16MEMx либо может быть обнулён программно.

#### **Формат выходных данных**

Выходные данные могут иметь вид двоичного кода со смещением либо кода с дополнением до 2-х, как показано в таблице 20-1. Формат данных определяется битом SD16DF.

**Таблица 20-1. Формат данных**

SD16DF	Формат	Напряжение на входе	SD16MEMx*	Выход цифрового фильтра (OSR = 256)
0	Однополярный: двоичный код со смещением	Максимум (+FSR) Ноль Минимум (-FSR)	FFFF 8000 0000	FFFFFF 800000 000000

SD16DF	Формат	Напряжение на входе	SD16MEMx*	Выход цифрового фильтра (OSR = 256)
1	Двухполярный: код с дополнением до 2-х	Максимум (+FSR) Ноль Минимум (-FSR)	7FFF 0000 8000	7FFFFF 800000 000000

\* не зависит от настроек SD16OSRx; SD16LSBACC = 0.

На рис. 20-4 показана связь между входным напряжением во всём диапазоне от минимума до максимума ( $-V_{FSR}$  ...  $+V_{FSR}$ ) и результатом преобразования. Проиллюстрирован результат для обоих типов формата данных.

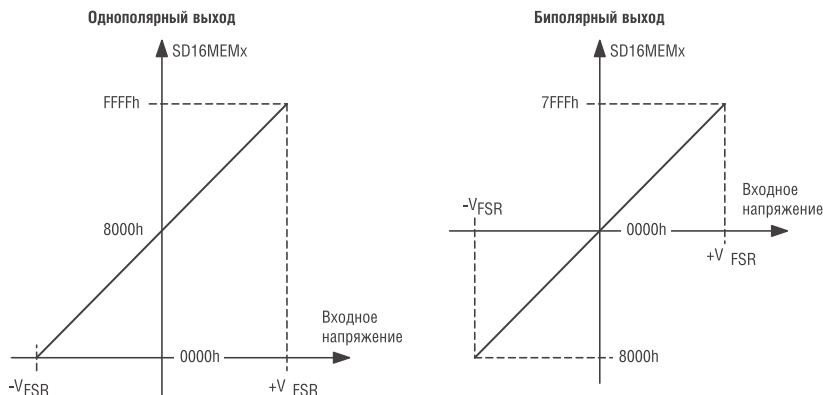


Рис. 20-4. Зависимость результата преобразования от входного напряжения

### Выход цифрового фильтра

Число бит на выходе каждого из цифровых фильтров зависит от степени передискретизации и изменяется от 16 до 24. На рис. 20-5 показан выход цифрового фильтра и его связь с содержимым регистров SD16MEMx для всех настроек бит SD16OSRx.

Биты SD16LSBACC и SD16LSBTOG позволяют иметь доступ к младшим значащим разрядам (M3P) выхода цифрового фильтра. Когда SD16LSBACC=1, 16 младших значащих разрядов выхода цифрового фильтра могут быть прочитаны через регистры SD16MEMx с помощью 16-битных инструкций. Доступ к регистрам SD16MEMx также может быть осуществлён при помощи байтовых инструкций (.B), возвращающих 8 M3P выхода цифрового фильтра.

Когда SD16LSBTOG=1, бит SD16LSBACC автоматически инвертируется каждый раз при чтении соответствующего канала регистра SD16MEMx. Это позволяет прочитать результат преобразования на выходе цифрового фильтра

**SD16OSRx = 256**

Выход цифрового фильтра

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB												LSB											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

**SD16MEMx**

(SD16LSBACC=0)

**SD16MEMx**

(SD16LSBACC=1)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB												LSB											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB												LSB											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB												LSB											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

**SD16OSRx = 128**

Выход цифрового фильтра

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB												LSB											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

**SD16OSRx = 64**

Выход цифрового фильтра

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB												LSB											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

**SD16OSRx = 32**

Выход цифрового фильтра

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB												LSB											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB												LSB											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB												LSB											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

**Рис. 20-5.** Используемые биты на выходе цифрового фильтра

полностью с помощью двух операций чтения регистра SD16MEMx. Установка или очистка бита SD16LSTOG не влияет на состояние SD16LSBACC вплоть до следующего доступа к регистру SD16MEMx.

Диапазон положительных и отрицательных значений на выходе цифрового фильтра зависит от выбранного коэффициента передискретизации.

## 20.2.6. Выбор канала

Каждый канал модуля АЦП SD16 может осуществлять преобразование по восьми дифференциальным входам, мультиплексируемым на вход усилителя с программируемым коэффициентом усиления (PGA). До шести пар входов (A0-A5) имеют внешние выводы. По вопросу подключения аналоговых входов см. документацию на конкретный МК. Встроенный датчик температуры доступен в каждом канале АЦП по входу A6 мультиплексора. Вход A7 представляет собой перемычку между положительным и отрицательным входом пары и может быть использован для калибровки смещения в каждом входном каскаде модуля АЦП SD16.

## Конфигурация аналогового входа

Конфигурация аналогового входа для всех каналов АЦП осуществляется при помощи регистров SD16INCTLx. Настройки, хранимые в этих регистрах независимы для каждого из каналов.

Битами SD16INCHx выбирается одна из восьми входных пар сигналов при помощи аналогового мультиплексора. Усиление для каждого PGA устанавливается битами SD16GAINx. Всего доступно 6 вариантов установки усиления.

Любые изменения бит SD16INCHx и SD16GAINx во время преобразования вступают в силу только на следующем шаге децимации цифрового фильтра. После изменения этих бит, последующие три результата преобразования могут быть некорректными из-за конечного времени установления цифрового фильтра. Эта проблема может быть разрешена аппаратно с помощью бит SD16INTDLYx. Если SD16INTDLY = 00h, запрос прерывания по завершению преобразования возникнет только после четвёртого по счёту преобразования после запуска.

### 20.2.7. Режимы преобразования

Модуль АЦП SD16 может работать в четырёх различных режимах, перечисленных в таблице 20-2. Выбор режима преобразования для каждого канала устанавливается битами SD16SNGL и SD16GRP.

**Таблица 20-2. Режимы преобразования**

SD16SNGL	SD16GRP*	Режим	Описание
1	0	Одиночное преобразование по одному каналу	Значение по одному из каналов оцифровывается один раз
0	0	Повторяющиеся преобразования по одному каналу	Значение по одному из каналов оцифровывается постоянно
1	1	Однократное групповое преобразование	Значения по группе каналов оцифровываются один раз
0	1	Повторяющиеся групповые преобразования	Значения по группе каналов оцифровываются постоянно

\* Бит означает, что канал входит в группу и что канал является ведущим в группе в случае, если SD16GRP = 0 и если установлен бит SD16GRP предыдущего канала.

#### Одиночное преобразование по одному каналу

Установка в «1» бита SD16SC запускает одиночное преобразование для соответствующего канала в случае, если SD16SNGL=1 и канал не сгруппирован с другими. Бит SD16SC очищается автоматически по завершению преобразования. Установка в «0» бита SD16SC в процессе преобразования немедленно

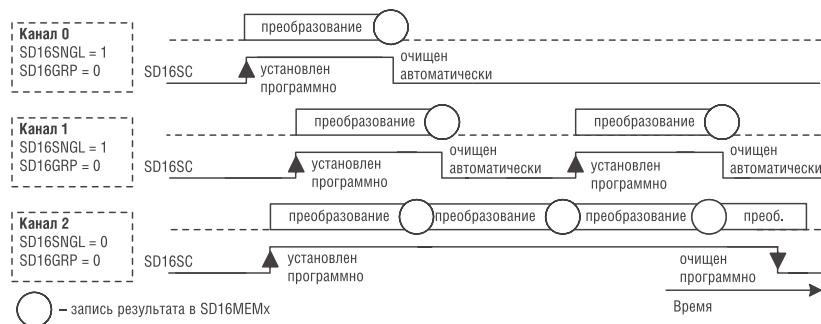
прекращает процесс, питание канала и соответствующий цифровой фильтр отключаются. При этом значение в регистре SD16MEMx после очистки бита SD16SC может измениться. Ввиду этого, рекомендуется считывать результат преобразования из регистра SD16MEMx до очистки бита SD16SC в целях предотвращения считывания ложного результата.

### Повторяющиеся преобразования по одному каналу

Выбор данного режима осуществляется установкой SD16SNGL=0. Преобразование по выбранному каналу запускается установкой в «1» бита SD16SC продолжается до тех пор, пока SD16SC не будет программно очищен в том случае, канал не сгруппирован с другими.

Установка в «0» бита SD16SC в процессе преобразования немедленно прекращает процесс, питание канала и соответствующий цифровой фильтр отключаются. При этом значение в регистре SD16MEMx после очистки бита SD16SC может измениться. Ввиду этого, рекомендуется считывать результат преобразования из регистра SD16MEMx до очистки бита SD16SC в целях предотвращения считывания ложного результата.

На рис. 20-6 показан процесс преобразования по одиночному каналу в режимах одиночного и повторяющихся преобразований.



**Рис. 20-6.** Процесс преобразования по одиночному каналу

### Однократное групповое преобразование

Последовательные каналы модуля АЦП SD16 могут быть сгруппированы с помощью бита SD16GRP для синхронизации преобразований. Установка бита SD16GRP для определённого канала группирует данный канал с последующим в модуле. Например, установка бита SD16GRP для 0-го канала группирует его с 1-ым каналом. В этом случае 1-ый канал является ведущим, его битом SD16SC разрешаются и запрещаются преобразования для всех остальных каналов группы.

пы. Бит SD16GRP ведущего канала всегда равен нулю. Бит SD16GRP последнего канала модуля АЦП SD16 является незначащим и также всегда равен нулю.

Если для канала в составе группы установлено SD16SNGL=1, для этого канала будет выбран режим однократного преобразования. Одиночное преобразование по этому каналу будет осуществляться синхронно с установкой в «1» бита SD16SC ведущего канала. Бит SD16SC для всех каналов в группе автоматически управляет битом SD16SC ведущего канала. Программная очистка бита SD16SC может осуществляться для каждого канала независимо.

Установка в «0» бита SD16SC ведущего канала в процессе преобразования немедленно прекращает процесс преобразования по всем каналам в группе, питание каналов и соответствующие цифровые фильтры отключаются. При этом значения в регистрах SD16MEMx после очистки бита SD16SC может изменяться. Ввиду этого, рекомендуется считывать результат преобразования из регистров SD16MEMx до очистки бита SD16SC в целях предотвращения считывания ложного результата.

### Повторяющиеся групповые преобразования

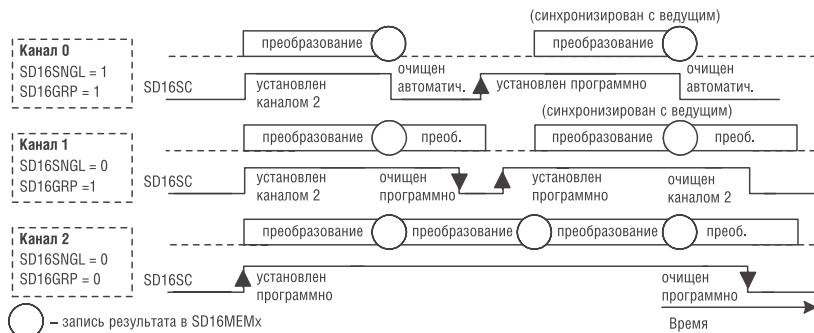
Когда SD16SNGL=0 для канала в составе группы, для данного канала выбран режим повторяющихся преобразований. Повторяющееся преобразование по этому каналу будет осуществляться синхронно с установкой в «1» бита SD16SC ведущего канала. Бит SD16SC для всех каналов в группе автоматически управляет битом SD16SC ведущего канала. Программная очистка бита SD16SC может осуществляться для каждого канала независимо.

Когда бит SD16SC для канала в составе группы программно установлен в «1» независимо от ведущего канала, преобразование по данному каналу будет автоматически синхронизировано с преобразованием по ведущему каналу. Таким образом, обеспечивается гарантированная синхронность преобразования сгруппированных каналов с преобразованием по ведущему каналу.

Установка в «0» бита SD16SC ведущего канала в процессе преобразования немедленно прекращает процесс преобразования по всем каналам в группе, питание каналов и соответствующие цифровые фильтры отключаются. При этом значения в регистрах SD16MEMx после очистки бита SD16SC может изменяться. Ввиду этого, рекомендуется считывать результат преобразования из регистров SD16MEMx до очистки бита SD16SC в целях предотвращения считывания ложного результата.

На рис. 20-7 показан процесс преобразования для трёх сгруппированных каналов модуля АЦП SD16. Канал 0 предназначен для работы в режиме однократных преобразований, (SD16SNGL=1), каналы 1 и 2 предназначены для работы в режиме последовательных преобразований (SD16SNGL=0). Канал 2 является главным каналом.

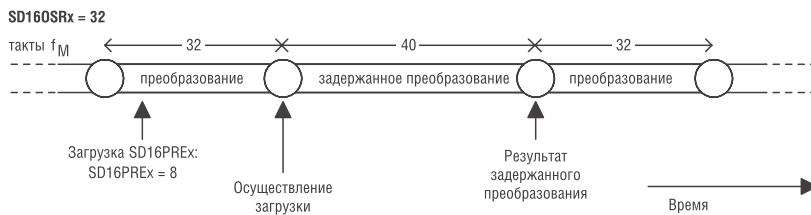
Обратите внимание, что преобразования по всем каналам осуществляются синхронно с ведущим каналом вне зависимости от программной установки индивидуальных бит SD16SC для ведомых каналов.



**Рис. 20-7.** Процесс преобразования для трёх сгруппированных каналов

### 20.2.8. Режим преобразования с предварительной загрузкой

Когда несколько каналов сгруппированы, регистры SD16PREx могут быть использованы за задания задержки преобразования для каждого канала. При использовании SD16PREx, время децимации цифрового фильтра возрастает на определённое число тактов частоты fM , находящееся в диапазоне от 0 до 255. На рис. 20-8 показан пример использования регистров SD16PREx.



**Рис. 20-8.** Преобразование с предварительной загрузкой

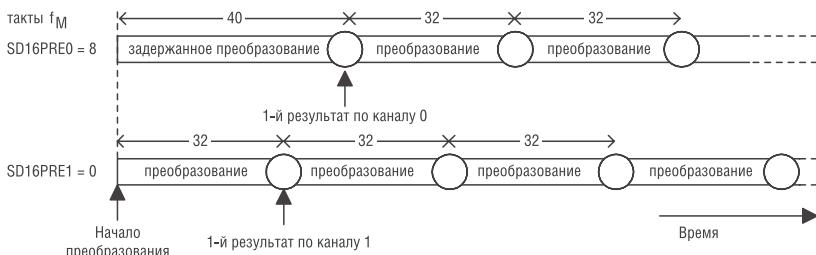
Задержка преобразования, записанная в SD16PREx, вступает в силу с начала следующего цикла преобразования после того, в котором она была записана. Задержка имеет действие только во время первого преобразования после установки бита SD16SC, следующего за записью в регистр SD16PREx. Последующие преобразования осуществляются в обычном режиме. После изменения содержимого регистров SD16PREx, следующая запись в эти регистры должна про-

изводиться не ранее, чем по завершению следующего цикла преобразования, в противном случае результаты преобразования могут быть некорректными.

Точность результата задержанного преобразования зависит от длины задержки, записанной в SD16PREx и от частоты оцифровываемого аналогового сигнала. Например, при измерении постоянного сигнала, задержка SD16PREx не влияет на результат преобразования вне зависимости от её длительности. Целесообразность использования режима задержанного преобразования определяется пользователем.

На рис. 20-9 показана работа сгруппированных каналов 0 и 1. Регистр предварительной загрузки канала 1 обнулён, что соответствует немедленному запуску преобразования, преобразование по каналу 0 задержано установкой SD16PRE0=8. Первое преобразование по каналу 0 с задержкой SD16PREx=8 приводит к сдвигу всех последующих преобразований на время, равное 8 тактов частоты  $f_M$ .

$SD160SRx = 32$

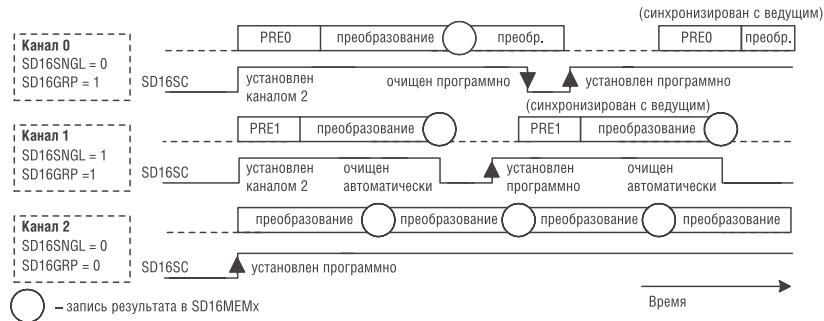


**Рис. 20-9.** Запуск преобразования с предварительной загрузкой

Следует соблюдать осторожность в том случае, если один канал или группа каналов функционируют в режиме однократного преобразования или программно запрещены в то время, как ведущий канал остаётся активным. Каждый раз после того, как канал в группе будет разрешён по новой и засинхронизирован с ведущим, значение задержки для этого канала будет использовано снова. На рис. 20-10 показан процесс синхронизации и применения задержек к каналам в группе. Рекомендуется задавать для ведущего канала  $SD16PREx = 0$  для сохранения значений задержек между ведущим и ведомыми каналами при запрещении и последующем разрешении последних.

### 20.2.9. Использование встроенного датчика температуры

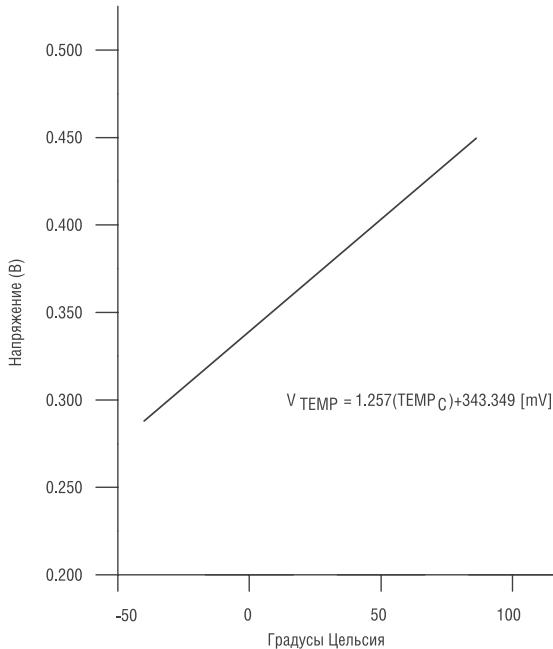
Для использования встроенного датчика температуры пользователь должен выбрать входной аналоговый канал  $SD16INCHx = 110$ . Все остальные на-



**Рис. 20-10.** Предварительная загрузка и синхронизация каналов

стройки, включая настройки регистров SD16INTDLYx и SD16GAINx, конфигурируются аналогично, как и для случая использования внешних входов.

Типовая характеристика температурной зависимости датчика приведена на рис. 20-11.



**Рис. 20-11.** Типовая характеристика датчика температуры

После переключения входа АЦП SD16 на температурный датчик следует выждать определённое время, используя для этого регистр SD16INTDLYx, для установления цифрового фильтра. В противном случае результаты могут быть некорректными. Характеристика температурного датчика может иметь значительное смещение, поэтому для многих применений может потребоваться калибровка. Уточнённые параметры датчика температуры см. в документации на конкретный тип МК.

### 20.2.10. Обработка прерываний

Модуль АЦП имеет 2 источника прерываний для каждого из каналов:

- SD16IFG
- SD16OVIFG

Биты SD16IFG устанавливаются, когда в соответствующий регистр памяти SD16MEMx записывается результат преобразования. В случае, если установлен соответствующий бит SD16IE и бит GIE в такой ситуации генерируется запрос прерывания. Флаг переполнения SD16OVIFG устанавливается, если результат преобразования записывается в регистр памяти SD16MEMx ранее, чем был прочитан результат предыдущего преобразования.

### Генерация векторов прерывания модуля АЦП SD16IV

Все источники прерываний модуля АЦП SD16 имеют собственный приоритет, но скомбинированы в один вектор прерываний. Регистр SD16IV используется для определения конкретного источника, вызвавшего прерывание модуля SD16. Разрешённый источник прерывания с максимальным приоритетом формирует число в регистре SD16IV (см. описание регистра). Это число может программно обработано либо просто добавлено к программному счётчику для автоматического перехода на соответствующую подпрограмму обработки. Запрет прерываний от модуля SD16 не влияет на значение в регистре SD16IV.

Любой доступ к регистру SD16IV, будь то чтение либо запись, не влияет на флаги SD16OVIFG и SD16IFG. Флаги SD16IFG сбрасываются при чтении соответствующего регистра памяти SD16MEMx либо могут быть очищены программно. Флаги SD16OVIFG могут быть очищены только программно.

Если во время обработки прерывания возникает следующее прерывание, для него также будет сгенерирован запрос прерывания. Например, если возникает прерывание по флагу SD16OVIFG и одному (либо более) флагу SD16IFG в то время как обработчик прерывания обращается к регистру SD16IV, прерывание по флагу SD16OVIFG будет обработано первым, а соответствующий флаг (либо флаги) должен быть программно очищен. После выполнения инструкции

RETI обработчика, флаг SD16IFG с наибольшим прерыванием вызовет следующий запрос прерывания.

### Задержка прерываний

Биты SD16INTDLYx управляют временными задержками генерации запроса прерываний для соответствующего канала. Эта опция позволяет задержать запрос прерывания после завершения преобразования на время до четырёх циклов преобразования для формирования времени установления цифрового фильтра. Эта задержка применяется каждый раз после установки бита SD16SC либо после изменения бит SD16GAINx или SD16INCHx для соответствующего канала. Кроме этого, SD16INTDLYx задерживает генерацию прерывания по переполнению для данного канала на выбранное значение циклов задержки. Запросы прерывания в течение этих циклов для задержанных преобразований не генерируются.

### Пример программы обработчика прерываний модуля АЦП SD16

Приведенный ниже пример программы рекомендуется для обслуживания регистра SD16IV ввиду наименьших программных затрат на обработку. Содержимое регистра SD16IV добавляется к программному счётчику PC, автоматически генерируя переход на соответствующий обработчик.

Число в правом поле показывает количество тактов ЦПУ для каждой инструкции. Показанные программные затраты включают вызов прерывания и возврат из него и не включают собственно обслуживание задачи. Общие программные затраты в тактах:

- SD16OVIFG, CH0 SD16IFG, CH1 SD16IFG 16 тактов
- CH2 SD16IFG 14 тактов

Обработка прерывания канала 2 SD16IFG даёт пример непосредственной проверки источника с наибольшим приоритетом в обработчике прерывания. Такой подход позволяет сэкономить 9 тактов в том случае, если потребуется обработка ещё одного прерывания модуля SD16.

```
; Обработчик прерываний модуля АЦП SD16.  
INT_SD16          ;Вход в обработчик      6  
    ADD  &SD16IV,PC ;Прибавить смещение к PC 3  
    RETI            ;Вектор 0: нет источника 5  
    JMP   ADOV      ;Вектор 2: переполнение  
                  ;АЦП                      2  
    JMP   ADM0      ;Вектор 4: CH_0 SD16IFG  2  
    JMP   ADM1      ;Вектор 6: CH_1 SD16IFG  2  
;  
; Обработка CH_2 SD16IFG. Оператор JMP не нужен.
```

```

;
ADM2      MOV    &SD16MEM2,xxx      ;Чтение результата,
                                         ;очистка флага
                                         ...
JMP       INT_SD16                  ;Остальные инструкции,
                                         ;если требуется
                                         ;Проверка следующего
                                         ;запроса прерывания

;
; Обработка остальных источников

;
ADM1      MOV    &SD16MEM1,xxx      ;Чтение результата,
                                         ;очистка флага
                                         ...
RETI          ;Остальные инструкции,
                                         ;если требуется
                                         ;Возврат
                                         ;из прерывания      5

;
ADM0      MOV    &SD16MEM0,xxx      ;Чтение результата,
                                         ;очистка флага
RETI          ;Возврат
                                         ;из прерывания      5

;
ADOV      ...                      ; Обработка
                                         ;переполнения SD16MEMx
RETI          ;Возврат
                                         ;из прерывания      5

```

## 20.3. Регистры модуля АЦП SD16

Регистры модуля контроллера ЖКИ перечислены в таблице 20-3.

**Таблица 20-3.**

Регистр	Краткое название	Тип	Адрес	Начальное состояние
Регистр управления модулем АЦП SD16	SD16CTL	Чтение/запись	0100h	Обнулён по сбросу PUC
Вектор прерываний модуля АЦП SD16	SD16IV	Чтение/запись	0110h	Обнулён по сбросу PUC
Регистр управления канала 0 модуля АЦП SD16	SD16CCTL0	Чтение/запись	0102h	Обнулён по сбросу PUC
Регистр памяти преобразования канала 0 модуля АЦП SD16	SD16MEM0	Чтение/запись	0112h	Обнулён по сбросу PUC

**Таблица 20-3. (Окончание)**

Регистр	Краткое название	Тип	Адрес	Начальное состояние
Регистр управления входами канала 0 модуля АЦП SD16	SD16INCTL0	Чтение/запись	0B0h	Обнулён по сбросу PUC
Регистр предварительной загрузки канала 0 модуля АЦП SD16	SD16PRE0	Чтение/запись	0B8h	Обнулён по сбросу PUC
Регистр управления канала 1 модуля АЦП SD16	SD16CCTL1	Чтение/запись	0104h	Обнулён по сбросу PUC
Регистр памяти преобразования канала 1 модуля АЦП SD16	SD16MEM1	Чтение/запись	0114h	Обнулён по сбросу PUC
Регистр управления входами канала 1 модуля АЦП SD16	SD16INCTL1	Чтение/запись	0B1h	Обнулён по сбросу PUC
Регистр предварительной загрузки канала 1 модуля АЦП SD16	SD16PRE1	Чтение/запись	0B9h	Обнулён по сбросу PUC
Регистр управления канала 2 модуля АЦП SD16	SD16CCTL2	Чтение/запись	0106h	Обнулён по сбросу PUC
Регистр памяти преобразования канала 2 модуля АЦП SD16	SD16MEM2	Чтение/запись	0116h	Обнулён по сбросу PUC
Регистр управления входами канала 2 модуля АЦП SD16	SD16INCTL2	Чтение/запись	09Eh	Обнулён по сбросу PUC
Регистр предварительной загрузки канала 2 модуля АЦП SD16	SD16PRE2	Чтение/запись	0BAh	Обнулён по сбросу PUC

### SD16CTL, Регистр управления модулем АЦП SD16

15	14	13	12	Резервные				SD16LP
r0	r0	r0	r0	r0	r0	r0	r0	rw-0
7	6	5	4	3	2	1	0	
<b>SD16DIVx</b>	<b>SD16SSELx</b>	<b>SD16VMIDON</b>	<b>SD16REFON</b>	<b>SD16OVIE</b>	<b>Резервный</b>			
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r0

<b>Резервные</b>	Биты 15-9	Резервные
<b>SD16LP</b>	Бит 8	Режим пониженного потребления. Этим битом выбирается режим работы АЦП SD16 с пониженной скоростью и пониженным энергопотреблением. 0 – режим пониженного потребления выключен 1 – режим пониженного потребления включен, максимальная тактовая частота модуля SD16 снижена.
<b>SD16DIVx</b>	Биты 7-6	Делитель тактовой частоты модуля SD16 00/1 01/2 10/4 11/8
<b>SD16SSELx</b>	Биты 5-4	Выбор источника тактирования модуля SD16 00 – MCLK 01 – SMCLK 10 – ACLK 11 – Внешняя TACLK
<b>SD16VMIDON</b>	Бит 3	Включение буфера VMID 0 – выключен 1 – включен
<b>SD16REFON</b>	Бит 2	Включение источника опорного напряжения 0 – выключен 1 – включен
<b>SD16OVIE</b>	Бит 1	Разрешение прерывания по переполнению модуля SD16. Для разрешения также следует установить бит GIE. 0 – Прерывание по переполнению запрещено 1 – Прерывание по переполнению разрешено
<b>Резервный</b>	Бит 0	Резервный

**SD16CCTLx, Регистр управления канала x модуля АЦП SD16**

15	14	13	12	11	10	9	8
Резервные				SD16SNGL	SD16OSRx		
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
<b>SD16LS BTOG</b>	<b>SD16LS BACC</b>	<b>SD16 OVIFG</b>	<b>SD16DF</b>	<b>SD16IE</b>	<b>SD16IFG</b>	<b>SD16SC</b>	<b>SD16GRP</b>
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
<b>Резервные</b>	Биты 15-11	Резервные					
<b>SD16SNGL</b>	Бит 10	Выбор режима одиночного преобразования 0 – режим повторяющихся преобразований 1 – режим одиночного преобразования					

<b>SD16OSRx</b>	Биты 9-8	Степень передискредитации 00 – 256 01 – 128 10 – 64 11 – 32
<b>SD16LSBT0G</b>	Бит 7	Инверсия МЗР. Когда этот бит установлен, SD16LSBACC инвертируется каждый раз при чтении регистра SD16MEMx. 0 – SD16LSBACC не инвертируется при чтении регистра SD16MEMx 1 – SD16LSBACC инвертируется при чтении регистра SD16MEMx
<b>SD16LSBACC</b>	Бит 6	Доступ к МЗР. Этот бит определяет доступ к старшим либо младшим 16 битам результата преобразования модуля SD16. 0 – SD16MEMx содержит СЗР преобразования. 1 – SD16MEMx содержит МЗР преобразования.
<b>SD160VIFG</b>	Бит 5	Флаг прерывания по переполнению модуля SD16. 0 – нет прерывания по переполнению 1 – есть прерывание по переполнению
<b>SD16DF</b>	Бит 4	Формат кода данных модуля SD16 0 – двоичный со смещением 1 – дополнительный до двух
<b>SD16IE</b>	Бит 3	Разрешение прерываний модуля SD16 0 – запрещены 1 – разрешены
<b>SD16IFG</b>	Бит 2	Флаг прерывания модуля SD16. SD16IFG выставляется, когда готовы результаты преобразования. Флаг SD16IFG сбрасывается автоматически при чтении из соответствующего регистра SD16MEMx либо может быть очищен программно. 0 – прерывания нет 1 – прерывание есть
<b>SD16SC</b>	Бит 1	Запуск преобразования модуля SD16 0 – нет запуска преобразования 1 – есть запуск преобразования
<b>SD16GRP</b>	Бит 0	Группировка каналов модуля SD16. Группирует канал с последующим. В последнем канале не используется. 0 – не сгруппирован 1 – сгруппирован

### SD16INCTLx, Регистр управления входами канала x модуля АЦП SD16

7	6	5	4	3	2	1	0
<b>SD16INTDLYx</b>	<b>SD16GAINx</b>			<b>SD16INCHx</b>			
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

<b>SD16INTDLYx</b>	Биты 7-6	Задержка генерации прерывания после запуска преобразования. Этими битами выбирается величина задержки в циклах преобразования. 00 – прерывание вызывается после четвёртого преобразования 01 – прерывание вызывается после третьего преобразования 10 – прерывание вызывается после второго преобразования 11 – прерывание вызывается после первого преобразования
<b>SD16GAINx</b>	Биты 5-3	Выбор коэффициента усиления предварительного усилителя модуля SD16 000 – x1 001 – x2 010 – x4 011 – x8 100 – x16 101 – x32 110 – зарезервировано 111 – зарезервировано
<b>SD16INCHx</b>	Биты 2-0	Выбор входной дифференциальной пары модуля SD16 000 – Ax.0 001 – Ax.1 010 – Ax.2 011 – Ax.3 100 – Ax.4 101 – Ax.5 110 – Ax.6 – датчик температуры 111 – Ax.7 – перемычка для калибровки PGA

**SD16MEMx, Регистр памяти преобразования канала x модуля АЦП SD16**

15	14	13	12	11	10	9	8
<b>Результат преобразования</b>							
r	r	r	r	r	r	r	r
7	6	5	4	3	2	1	0
<b>Результат преобразования</b>							
r	r	r	r	r	r	r	r
<b>Результат преобразования</b>	Биты 15-0	Результат преобразования. В регистре SD16MEMx хранятся данные старших или младших 16 бит с выхода цифрового фильтра в зависимости от состояния бита SD16LSBACC.					

**SD16PREx, Регистр предварительной загрузки канала x модуля АЦП SD16**

7	6	5	4	3	2	1	0
<b>Значение загрузки</b>							
r	r	r	r	r	r	r	r

<b>Значение загрузки</b>	Биты 7-0	Значение предварительной загрузки для цифрового фильтра
--------------------------	----------	---

### SD16IV, Регистр векторов прерываний модуля АЦП SD16

15	14	13	12	11	10	9	8
<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
<b>0</b>	<b>0</b>	<b>0</b>		<b>SD16IVx</b>		<b>0</b>	
r0	r0	r0	r-0	r-0	r-0	r-0	r0

SD16IVx	Биты 15-0	Вектор прерываний модуля SD16			
		Содержимое SD16IV	Источник прерывания	Флаг прерывания	Приоритет
		000h	Нет прерываний		
		002h	Переполнение SD16MEMx	SD16CCTLx SD16OVIFG*	Наивысший
		004h	Прерывание от SD16_0	SD16CCTL0 SD16IFG	
		006h	Прерывание от SD16_1	SD16CCTL1 SD16IFG	
		008h	Прерывание от SD16_2	SD16CCTL2 SD16IFG	
		00Ah	Зарезервировано	–	
		00Ch	Зарезервировано	–	
		00Eh	Зарезервировано	–	
		010h	Зарезервировано	–	Низший
* При возникновении переполнения SD16, пользователь должен проверить все флаги SD16CCTLx и SD16OVIFG для определения источника переполнения.					

# MSP430x4xxFamily

ЦАП12

---

*Раздел XXI.*



## ЦАП12

Модуль ЦАП12 представляет собой 12-разрядный цифро-аналоговый преобразователь. В этом разделе описывается ЦАП12. В устройствах MSP430FG43x реализовано два модуля ЦАП12.

### 21.1. 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 MSP430FG43x показана на рис. 21-1.

### 21.2. Функционирование ЦАП12

Модуль ЦАП12 конфигурируется программным обеспечением пользователя. Настройка и функционирование ЦАП12 обсуждаются в нижеследующих разделах.

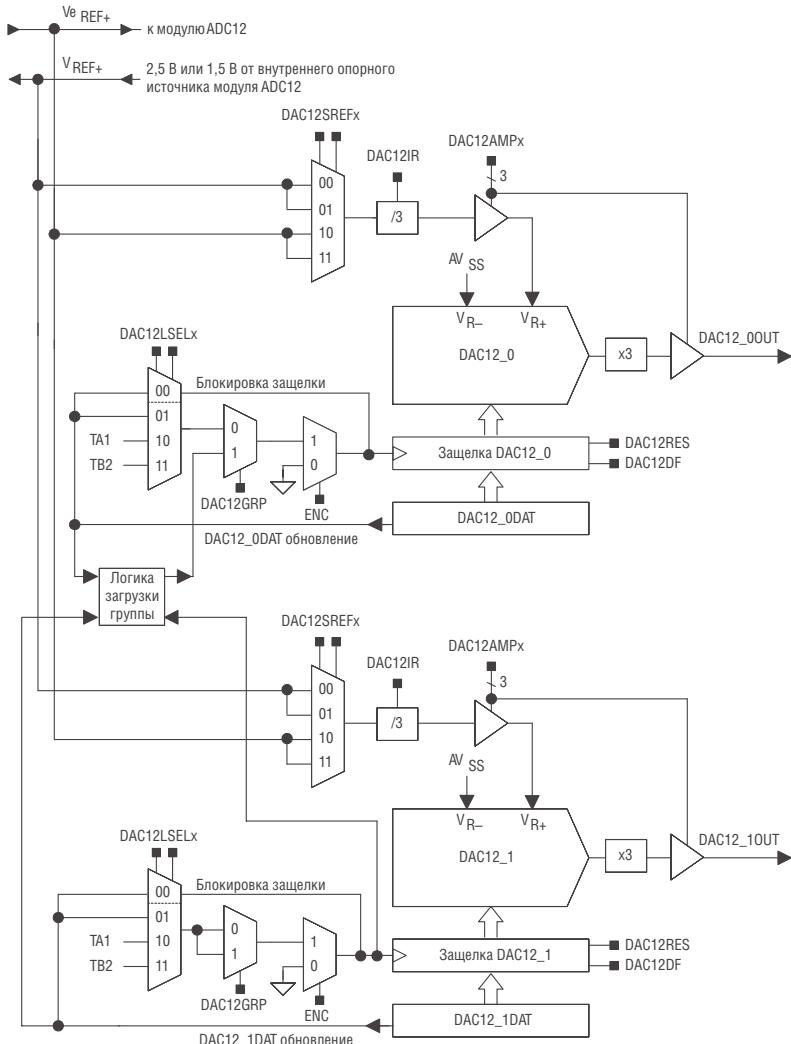


Рис. 21-1. Блок-схема ЦАП12

### 21.2.1. 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 выбранно-

му 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 21-1.

**Таблица 21-1. Полный диапазон ЦАП12 ( $V_{ref} = V_{REF+}$  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 OFFh, а в 12-разрядном режиме максимальное используемое значение для DAC12\_xDAT равно OFFFh. 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.

#### 21.2.2. 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  $V_{REF-}$ .

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 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.

#### 21.2.3. 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 данныхкрыта, 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.

#### Примечание: Время установки

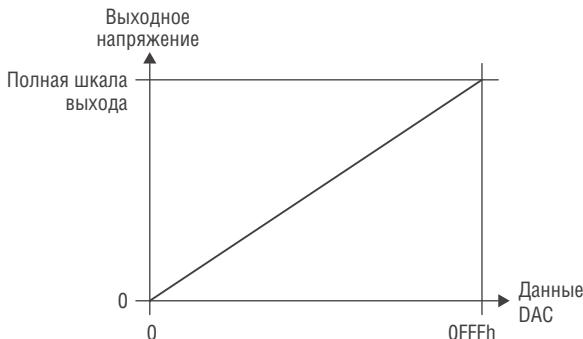
Под временем установки понимается время, необходимое ЦАП для установки выходного напряжения, соответствующего его цифровому представлению на входе и при изменении входного значения.

#### 21.2.4. Формат данных DAC12\_xDAT

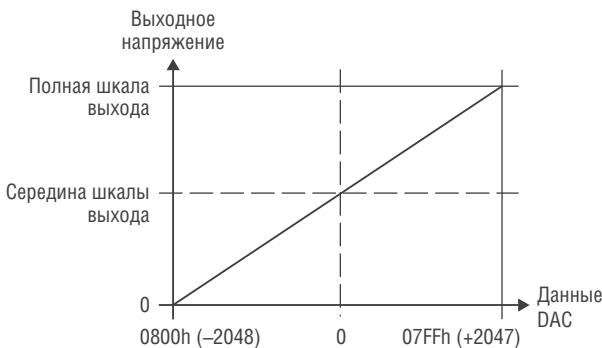
ЦАП12 поддерживает два формата данных: натуральный двоичный и формат с дополнением до двух. Когда используется натуральный формат данных, полный диапазон вывода равен 0FFFh в 12-разрядном режиме (0FFh в 8-разрядном режиме), как показано на рис. 21-2.

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 максимальное.

жение составит половину шкалы, а при 07FFh (007Fh для 8-разрядного режима) выходное напряжение достигнет полного диапазона, как показано на рис. 21-3.



**Рис. 21-2.** Зависимость выходного напряжения от данных ЦАП12 в 12-разрядном режиме в натуральном двоичном формате



**Рис. 21-3.** Зависимость выходного напряжения от данных ЦАП12 в 12-разрядном режиме в формате дополнения до двух

### 21.2.5. 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 сделать. Выходное напряжение остается равным нулю, пока цифровой вход ЦАП12 не обеспечит достаточного для преодоления отрицательного напряжения сме-

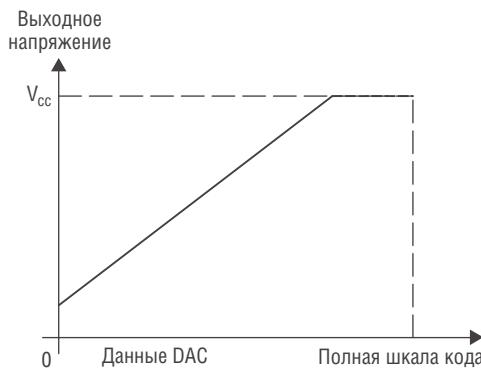
щения положительного выходного напряжения. Получающаяся передаточная функция показана на рис. 21-4.



*Рис. 21-4.* Отрицательное 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. 21-5.

ЦАП12 имеет возможность калибровки напряжения смещения выходного усилителя. Установка бита DAC12CALON инициирует калибровку смещения. Ка-



*Рис. 21-5.* Положительное смещение

либровка 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.

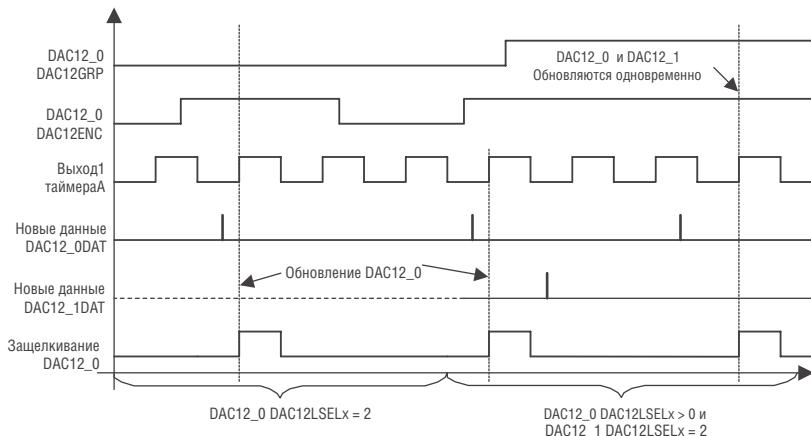
### **21.2.6. Группировка нескольких модулей ЦАП12**

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 MSP430FG43x 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. 21-6 показан пример тактиро-



**Рис. 21-6.** Пример обновления группы ЦАП12, запуск от таймера\_A3

вания защелкивания-обновления для сгруппированных модулей DAC12\_0 и DAC12\_1.

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.

#### **21.2.7. 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.

### **21.3. Регистры ЦАП12**

Регистры ЦАП12 приведены в таблице 21-2.

**Таблица 21-2. Регистры ЦАП12**

Регистр	Краткое обозначение	Тип регистра	Адрес	Исходное состояние
<b>Управление DAC12_0</b>	DAC12_OCTL	Чтение/запись	01C0h	Сброс с POR
<b>Данные DAC12_0</b>	DAC12_0DAT	Чтение/запись	01C8h	Сброс с POR
<b>Управление DAC12_1</b>	DAC12_1CTL	Чтение/запись	01C2h	Сброс с POR
<b>Данные DAC12_1</b>	DAC12_1DAT	Чтение/запись	01CAh	Сброс с POR

## DAC12\_xCTL, управляющий регистр ЦАП12

15	14	13	12		11	10	9	8
<b>DAC12OPS</b>	<b>DAC12SREFx</b>	<b>DAC12RES</b>		<b>DAC12LSELx</b>	<b>DAC12CALON</b>	<b>DAC12IR</b>		
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4		3	2	1	0
	<b>DAC12AMPx</b>		<b>DAC12DF</b>	<b>DAC12IE</b>	<b>DAC12IFG</b>	<b>DAC12ENC</b>	<b>DAC12GRP</b>	
rw-(0)	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
<b>DAC12SREFx</b>	<b>Биты 14-13</b>	Выбор опорного напряжения ЦАП12 00 – VREF+ 01 – VREF+ 10 – VeREF+ 11 – VeREF+
<b>DAC12RES</b>	<b>Бит 12</b>	Выбор разрешения ЦАП12 0 – 12-разрядное разрешение 1 – 8-разрядное разрешение
<b>DAC12LSELx</b>	<b>Биты 11-10</b>	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)
<b>DAC12CALON</b>	<b>Бит 9</b>	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
<b>DAC12IR</b>	<b>Бит 8</b>	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-му кратному опорному напряжению

<b>DAC12AMPx</b>	<b>Биты 7-5</b>	Настройка усилителя ЦАП12. Эти биты выбирают время установки в зависимости от потребляемого тока для входного и выходного усилителей ЦАП12		
		<b>DAC12AMPx</b>	<b>Входной буфер</b>	<b>Выходной буфер</b>
		000	Выключен	ЦАП12 выключен, выход в высокоимпедансном состоянии
		001	Выключен	ЦАП12 выключен, на выходе 0В
		010	Низкая скорость/ток	Низкая скорость/ток
		011	Низкая скорость/ток	Средняя скорость/ток
		100	Низкая скорость/ток	Высокая скорость/ток
		101	Средняя скорость/ток	Средняя скорость/ток
		110	Средняя скорость/ток	Высокая скорость/tok
		111	Высокая скорость/tok	Высокая скорость/tok
<b>DAC12DF</b>	<b>Бит 4</b>	Формат данных ЦАП12 0 – Натуральный двоичный 1 – Формат с дополнением до двух		
<b>DAC12IE</b>	<b>Бит 3</b>	Разрешение прерывания от ЦАП12 0 – Запрещено 1 – Разрешено		
<b>DAC12IFG</b>	<b>Бит 2</b>	Флаг прерывания ЦАП12 0 – Прерывание не ожидается 1 – Прерывание ожидается		
<b>DAC12ENC</b>	<b>Бит 1</b>	Включение преобразования ЦАП12. Этот бит включает модуль ЦАП12, когда DAC12LSELx>0. Когда DAC12LSELx=0, бит DAC12ENC игнорируется. 0 – ЦАП12 выключен 1 – ЦАП12 включен		
<b>DAC12GRP</b>	<b>Бит 0</b>	Группировка ЦАП12. Группируются 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**

15	14	13	12		11	10	9	8
0	0	0	0		<b>Данные ЦАП12</b>			
r(0)	r(0)	r(0)	r(0)		rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4		3	2	1	0				
<b>Данные ЦАП12</b>												
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)				
<b>Данные ЦАП12</b>	<b>Биты 15-12</b>	Не используется. Эти биты всегда равны 0 и не влияют на ядро ЦАП12.										
		Данные ЦАП12			<b>Данные ЦАП12</b>							
		<b>Формат данных ЦАП12</b>		<b>Данные ЦАП12</b>								
		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.								

# MSP430x4xxFamily

## Модуль детекторов SCAN IF

*Раздел XXII.*



## Модуль детекторов SCAN IF

Модуль детекторов SCAN IF автоматически сканирует внешние датчики, определяющие линейное или вращательное движение. В этой главе описана работа интерфейса SCAN. Модуль SCAN IF присутствует в микроконтроллерах MSP430FW42x.

### 22.1. Модуль детекторов SCAN IF – введение

Модуль детекторов SCAN IF используется для автоматического измерения линейного либо вращательного перемещения с минимально возможным энергопотреблением. Модуль SCAN IF состоит из трёх блоков: блока аналоговых формирователей (AFE), блока автомата состояний (PSM) и блока автомата формирования импульсов (TSM). Блок аналоговых формирователей формирует сигналы возбуждения датчиков, детектирует сигнал с них и преобразует его в цифровой вид. Далее сигналы поступают на автомат состояний, который предназначен для определения и измерения вращения либо перемещения. Блок автомата формирования импульсов формирует управляющие сигналы для модулей аналоговых преобразователей и автомата состояний.

**Модуль SCAN IF обладает следующими свойствами:**

- Поддержка различных датчиков LC-типа
- Обработка огибающей сигнала
- Измерение амплитуды колебаний детектора
- Поддержка резистивных датчиков, таких, как датчик Холла и магниторезистор (GMR)
- Непосредственный аналоговый вход для аналого-цифрового преобразования
- Непосредственный цифровой вход для цифровых датчиков, таких, как оптический энкодер
- Поддержка квадратурного декодирования

Блок схема модуля SCAN IF показана на рис. 22-1.

### 22.2. Работа модуля АЦП SD16

Модуль SCAN IF конфигурируется при помощи пользовательского программного обеспечения. Настройка и функционирование модуля будут подробно рассмотрены ниже.

#### 22.2.1. Блок аналоговых формирователей

Блок аналоговых формирователей модуля SCAN IF обеспечивает необходимые сигналы возбуждения для датчиков и измерение сигналов. Блок анало-

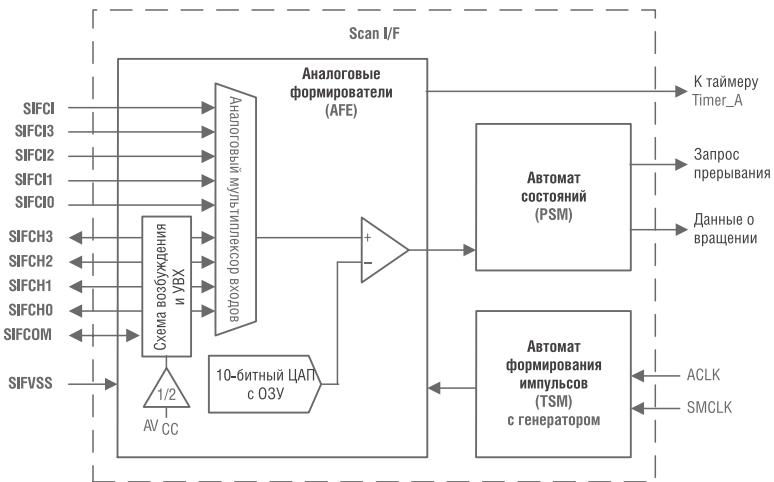


Рис. 22-1. Блок схема модуля SCAN IF

говых формирователей автоматически управляетяется автоматом формирования импульсов в соответствии с информацией в его таблице. Блок схема блока аналоговых формирователей показана на рис. 22-2.

#### Примечание: Сигналы автомата формирования импульсов

Далее в этом разделе сигналы, поступающие от автомата формирования импульсов (TSM) будут обозначаться индексом (*tsm*). Например, сигнал *SIFEX(tsm)* поступает от автомата формирования импульсов.

#### Схема возбуждения

Схема возбуждения предназначена для питания резистивных делителей либо для возбуждения датчиков LC-типа. Блок-схема схемы возбуждения с одним подключенным датчиком LC-типа показана на рис. 22-3. Когда бит SIFTEN установлен, а бит SIFSH сброшен, схема возбуждения включена, а устройство выборки-хранения (УВХ) выключено.

Во время высокого уровня сигнала *SIFEX(tsm)* от автомата формирования импульсов вход *SIFCHx* выбранного канала подключен к *SIVSS*, а вход *SIFCOM* подключен к буферизованной средней точке для возбуждения датчика. Сигнал *SIFLCEN(tsm)* для функционирования схемы возбуждения должен иметь низкий уровень. Во время подачи сигнала возбуждения на один из каналов и измерения по этому каналу все остальные каналы автоматически отключаются. Запитывается и измеряется только один выбранный канал.

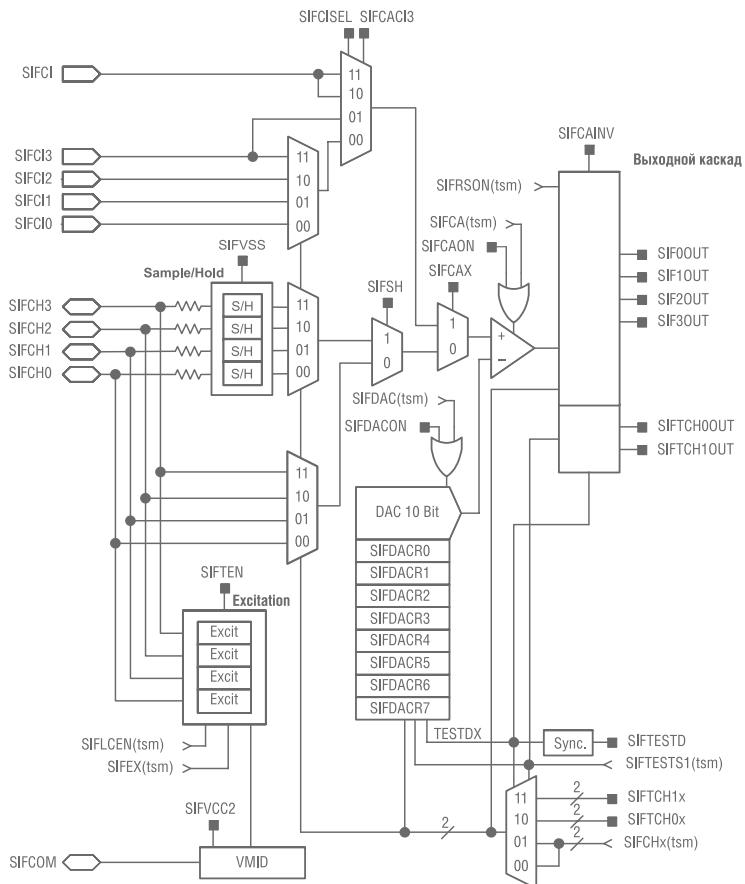


Рис. 22-2. Блок схема блока аналоговых формирователей

Период возбуждения должен быть достаточно длительным для исключения сильных перегрузок LC-датчика. После завершения процесса возбуждения вход SIFCHx отключается от «земли» во время низкого уровня сигнала SIFEX(tsm), LC-датчик при этом переходит в режим свободных колебаний. Амплитуда колебаний может превышать уровень напряжения питания, при этом она будет «обрезана» защитным диодом до уровня «напряжение питания + падение на диоде». Таким образом, обеспечивается постоянство максимума амплитуды. По завершению измерения датчик должен быть закорочен уста-

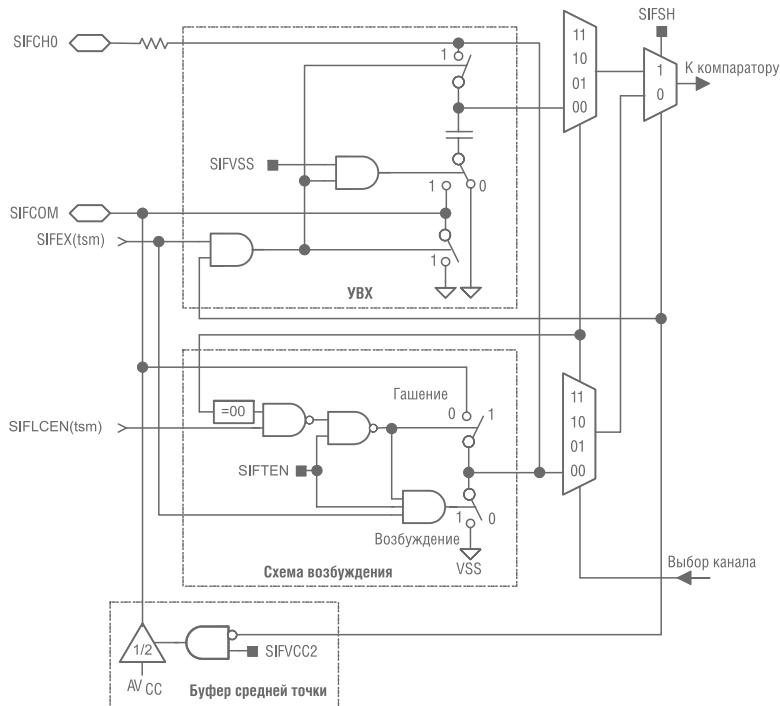


Рис. 22-3. Схема возбуждения и УВХ

новкой SIFLCEN(tsm) = 0 для погашения остаточной энергии перед следующим измерением.

### Буфер средней точки

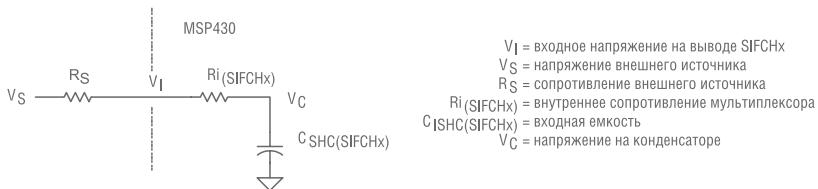
Буфер средней точки включается установкой SIFVCC2=1 и позволяет LC-датчику находиться в режиме свободных колебаний. Для установления буферу средней точки требуется до 6 мс, кроме этого для него необходим включенный и работающий на частоте 32768 Гц генератор частоты ACLK.

### Устройство выборки-хранения (УВХ)

Устройство выборки-хранения (УВХ) используется для запоминания измеряемого напряжения с датчика. Блок схема УВХ показана на рис. 22-3. Когда SIFSH=1 а SIFTEN=0 УВХ включено а схема возбуждения и буфер средней точки выключены. УВХ используется для резистивных делителей либо для других аналоговых сигналов, которым требуется оцифровка.

К выводам SIFCH<sub>x</sub> и SIFCOM может быть подключено до четырёх резистивных делителей. AV<sub>cc</sub> и SIFCOM являются общими положительным и отрицательным потенциалами соответственно для всех подключаемых резистивных делителей. Когда SIFEX<sub>(tsm)</sub>=1, SIFCOM подключен к SIV<sub>ss</sub>, а через резистивные делители протекает ток. При этом конденсаторы каждого УВХ заряжаются до напряжений точек делителя. Значения по всем каналам резистивных делителей запоминаются одновременно. Когда SIFEX<sub>(tsm)</sub>=0, конденсатор УВХ отключен от резистивного делителя, а SIFCOM отключен от SIV<sub>ss</sub>. После запоминания все каналы могут быть оцифрованы последовательно при помощи устройства выбора каналов, компаратора и цифро-аналогового преобразователя (ЦАП).

Активный вход SIFCH<sub>x</sub> во время осуществления выборки t<sub>sample</sub> может быть представлен в виде RC-фильтра высоких частот, как показано на рис. 22-4. Паразитное сопротивление мультиплексора во включенном состоянии R<sub>i(SIFCH<sub>x</sub>)</sub> (3 кОм макс.) последовательно с конденсатором C<sub>SHC(SIFCH<sub>x</sub>)</sub> (7 пФ макс.) подключено к резистивному делителю. Для точного 10-битного преобразования напряжение на конденсаторе V<sub>C</sub> должно отличаться от напряжения на резистивном делителе на величину, не превышающую 1/2 МЗР. См. уточнённые параметры в документации на конкретный МК.



**Рис. 22-4.** Эквивалентная схема аналогового входа

Сопротивление источника R<sub>s</sub> и сопротивление мультиплексора R<sub>i(SIFCHx)</sub> влияют на время t<sub>sample</sub>. Для вычисления минимального времени выборки t<sub>sample</sub> при 10-битном преобразовании может быть использована следующая формула:

$$t_{\text{SAMPLE}} > (R_s + R_{i(SIFCHx)}) \times \ln(2^{11}) \times C_{\text{SHC}(SIFCHx)}$$

Подставляя приведенные выше значения для R<sub>i</sub> и C<sub>i</sub>, получаем:

$$t_{\text{SAMPLE}} > (R_s + 3k) \times 7.625 \times 7\text{pF}$$

Например, если R<sub>s</sub> равен 10 кОм, t<sub>sample</sub> должно быть более 684 нс.

## Непосредственные аналоговые и цифровые входы

Установкой бита SIFCAX внешние аналоговые либо цифровые сигналы могут быть подключены непосредственно к компаратору через входы SIFClx. Эта опция позволяет обрабатывать сигналы с оптических энкодеров и других датчиков.

### Выбор входа компаратора и выходных бит

Битами SIFCAX и SIFSH в качестве входных сигналов компаратора выбираются каналы SIFClx или SIFCHx, как показано в таблице 22-1.

**Таблица 22-1. Выбор каналов SIFClx и SIFCHx**

SIFCAX	SIFSH	Описание
0	0	Активны SIFCHx и схема возбуждения
0	1	Активны SIFCHx и УВХ
1	x	Активны входы SIFClx

Сигналы TESTDX и SIFTSTS1(tsm) определяют используемые выходные биты SIFxOUT и SIFTCHxOUT для выхода компаратора как показано в таблице 22-2. Сигнал TESTDX управляетяется битом SIFTTESTD.

**Таблица 22-2. Активные выходные биты**

TESTDX	SIFCH(tsm)	SIFTSTS1(tsm)	Активный выходной бит
0	00	X	SIFOOUT
0	01	X	SIF1OUT
0	10	X	SIF2OUT
0	11	X	SIF3OUT
1	x	0	SIFTCH0OUT
1	x	x	SIFTCH1OUT

Когда TESTDX=0, сигналами SIFCHx(tsm) определяется какой из каналов SIFClx или SIFCHx запитан от схемы возбуждения и подключен к компаратору. Сигналы SIFCHx(tsm) также определяют соответствующий выходной бит для результата компаратора.

Когда TESTDX=1, выбор канала определяется сигналом SIFTSTS1<sub>(tsm)</sub>. В том случае, если TESTDX=1 и SIFTSTS1<sub>(tsm)</sub>=0, выбор входного сигнала определяется битами SIFTCH0x, а активным выходным битом является SIFTCH0OUT. Когда TESTDX=1 и SIFTSTS1<sub>(tsm)</sub>=1, выбор входного сигнала определяется битами SIFTCH1x, а активным выходным битом является SIFTCH1OUT.

При SIFCAX=1, биты SIFCSEL и SIFCI3 определяют переключение между каналами SIFClx и входом SIFCI, позволяя запоминать состояние выхода компаратора для одного из входных сигналов в одном из четырёх выходных

бит SIF0OUT – SIF3OUT. Эта функция может быть использована для получения сигнала огибающей с датчика. Выходной каскад включается сигналом SIFRSON(tsm). Если выход компаратора равен единице и SIFRSON=1, внутренний триггер устанавливается. В противном случае триггер сбрасывается. Состояние выхода триггера записывается в выбранный выходной бит по фронту сигнала SIFSTOP(tsm) как показано на рис. 22-5.

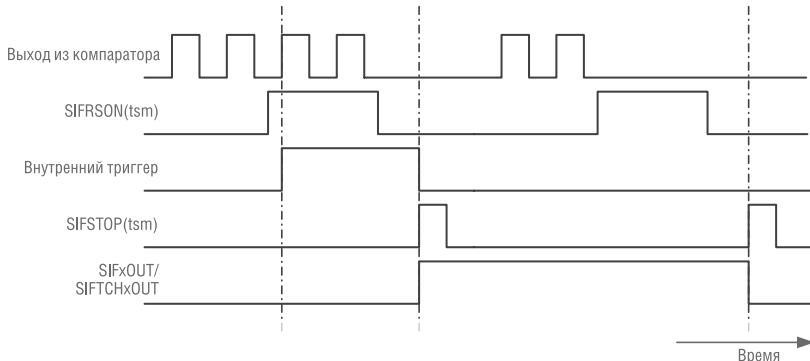


Рис. 22-5. Диаграмма выходных сигналов модуля аналоговых формирователей

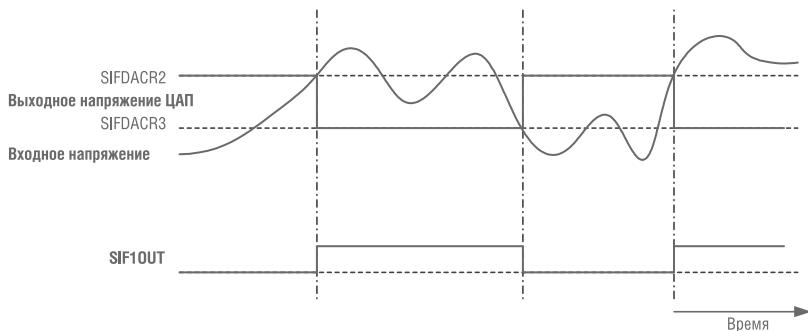
## Компаратор и ЦАП

Входные аналоговые сигналы преобразуются в цифровой вид при помощи компаратора и программируемого 10-битного ЦАП. Компаратор сравнивает измеряемый аналоговый сигнал с опорным напряжением, которое генерирует ЦАП. Если напряжение выше опорного, на выходе компаратора высокий уровень, если ниже – низкий. Выход компаратора может быть проинвертирован установкой бита SIFCAINV. Состояние выхода компаратора сохраняется в выбранном выходном бите и обрабатывается автоматом состояний для определения наличия перемещения и его направления.

Включение и выключение компаратора и ЦАП осуществляется сигналами SIFCA<sub>(tsm)</sub> и SIFDAC<sub>(tsm)</sub> соответственно. Управление осуществляется автоматом состояний. Также есть возможность оставить их включенными постоянно, для этого следует установить биты SIFCAON и SIFDACON. Таким способом можно повысить точность прецизионных измерений.

Для каждого из входов имеется 2 регистра ЦАП для установки опорного напряжения, как указано в таблице 22-3. Совместно с последним сохраненным состоянием выхода компаратора, SIFxOUT, эти два уровня могут быть использованы для формирования аналогового гистерезиса, как показано на рис. 22-6.

Индивидуальные настройки для каждого из входов позволяют компенсировать неидентичность датчиков.



**Рис. 22-6.** Формирование аналогового гистерезиса регистрами ЦАП

**Таблица 22-3. Активные регистры ЦАП**

Активный выходной бит, SIFxOUT	Последнее значение бита SIFxOUT	Активный регистр ЦАП
SIFOOUT	0	SIFDACR0
	1	SIFDACR1
SIF1OUT	0	SIFDACR2
	1	SIFDACR3
SIF2OUT	0	SIFDACR4
	1	SIFDACR5
SIF3OUT	0	SIFDACR6
	1	SIFDACR7

Когда TESTDX=1, регистры SIFDACR6 и SIFDACR7 используются для формирования опорного напряжения компаратора как показано в таблице 22-4.

**Таблица 22-4. Активные регистры ЦАП при TESTDX=1**

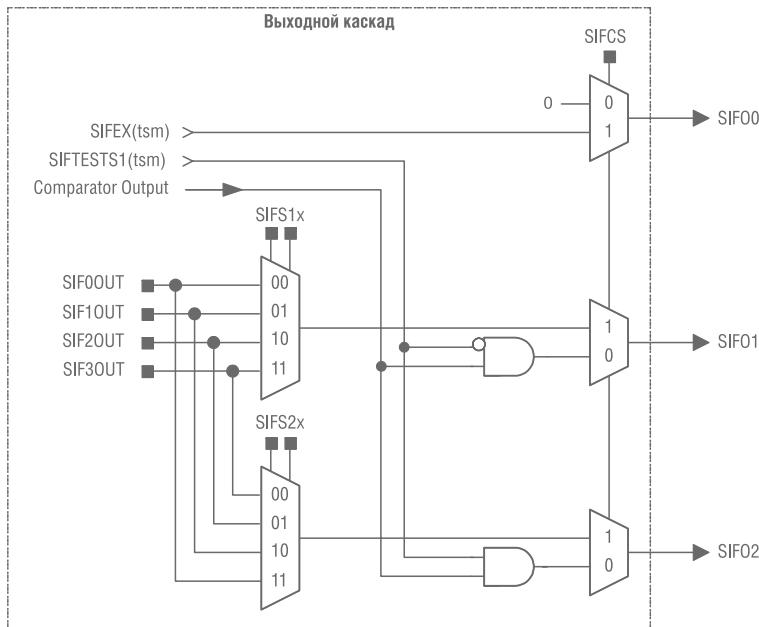
SIFTTESTS1(tsm)	Активный регистр ЦАП
1	SIFDACR6
0	SIFDACR7

## Внутренние подключения к модулю таймера Timer1\_A5

Выходы блока аналоговых формирователей подключены к 3-м различным регистрам захвата-сравнения модуля таймера Timer1\_A5. Выходной каскад блока аналоговых формирователей, показанный на рис. 22-7 может работать в двух различных режимах, определяемых битом SIFCS и передающих сигналы SIFO<sub>x</sub> в модуль таймера Timer1\_A5. Подробную информацию о подключении этих сигналов см. в документации на конкретный МК.

Когда SIFCS=1, сигнал SIFEX(tsm) и сигнал с выхода компаратора могут быть выбраны в качестве входов для различных регистров захвата-сравнения модуля таймера Timer1\_A5. Эта опция может быть использована для измерения времени между возбуждением датчика и последним периодом колебаний, поступившим на вход компаратора для осуществления аналого-цифрового преобразования slope-типа.

В случае, если SIFCS=0, выходные биты SIFxOUT могут быть выбраны в качестве входов таймера Timer1\_A5 совместно с битами SIFS1x и SIFS2x. Эта опция позволяет измерять скважность сигнала SIFxOUT.



**Рис. 22-7.** Выходной каскад блока аналоговых формирователей, подключаемый к таймеру Timer1\_A5

## 22.2.2. Автомат формирования импульсов модуля SCAN IF

Блок автомата формирования импульсов (TSM) представляет собой автомат последовательных конечных состояний, определяемых регистрами SIFTSMx. Данный блок автоматически управляет блоком аналоговых формирователей, в том числе схемой возбуждения датчиков без использования ресурсов ЦПУ. Состояния определяются матрицей  $24 \times 16$ -бит, расположенной в регистрах SIFTSM0...SIFTSM23. Битом SIFEN включается блок TSM. Когда SIFEN=0, входной делитель частоты ACLK, триггер запуска TSM, и выходы TSM сброшены, а встроенный генератор остановлен. Блок-схема автомата формирования импульсов показана на рис. 22-8.

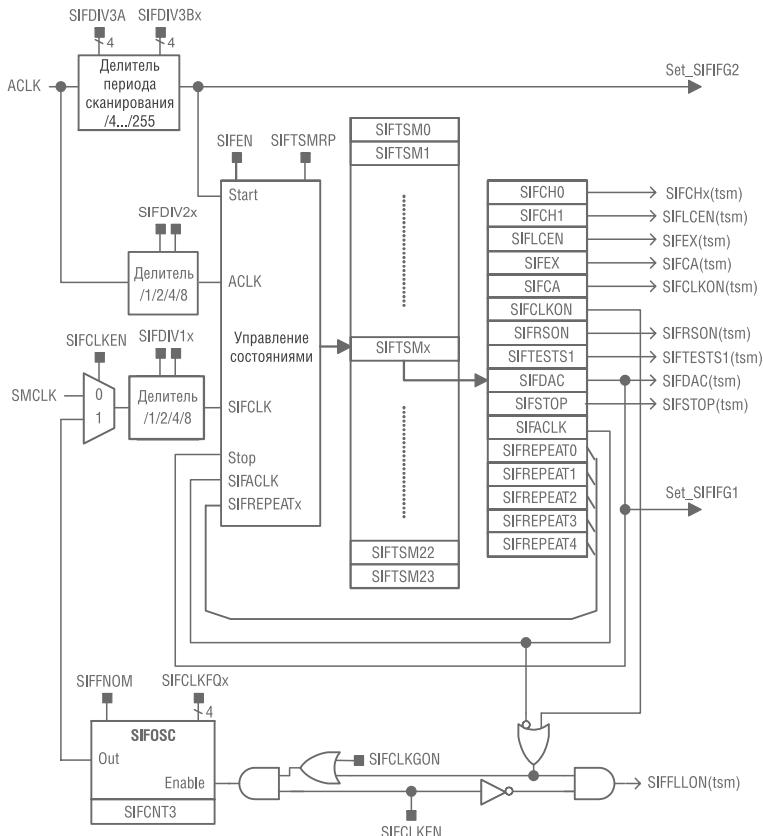


Рис. 22-8. Блок-схема автомата формирования импульсов

Состояние автомата начинается с SIFTSM0, конечное состояние – SIFTSMx с установленным битом SIFTSTOP. Когда достигнуто состояние с установленным битом SIFSTOP, счётчик состояний сбрасывается в 0 и переключение состояний останавливается. Обработка состояний перезапускается с SIFTSM0 при возникновении условия запуска, если SIFTSMRP=0, либо незамедлительно, если SIFTSMRP=1.

После генерации импульса SIFSTOP<sub>(tsm)</sub>, в автомат формирования импульсов будет загружено состояние, хранящееся в SIFTSM0. В этом состоянии SIFLCEN<sub>(tsm)</sub> должен быть сброшен для гарантированного замыкания всех LC-датчиков.

## Работа блока TSM

Автомат состояний TSM перезапускается автоматически с периодом, определяемым делителем частоты ACLK. Коэффициент деления для сигнала запуска при SIFTSMRP = 0 задаётся битами SIFDIV3Ax и SIFDIV3Bx. Например, если SIFDIV3A и SIFDIV3B определяют величину в 270 тактов частоты ACLK, автомат TSM автоматически запускается через каждые 270 тактов частоты ACLK. Если SIFTSMRP=1, автомат TSM перезапускается незамедлительно с состояния SIFTSM0 на следующем такте частоты ACLK после достижения состояния с установленным битом SIFTSTOP. По запуску автомата TSM выставляется флаг прерывания SIFIFG2.

Биты SIFDIV3Ax и SIFDIV3Bx могут быть модифицированы в любое время в процессе работы. После их изменения текущий цикл смены состояний будет завершён со старыми настройками, следующий цикл будет начат с новыми настройками.

## Управление аналоговыми формирователями автоматом TSM

SIFLCEN, SIFEX, SIFCA, SIFRSON, SIFTESTS1, SIFDAC, и SIFTSTOP. При установке любого из этих бит в «1» соответствующие сигналы, SIFCHx<sub>(tsm)</sub>, SIFLCEN<sub>(tsm)</sub>, SIFEX<sub>(tsm)</sub>, SIFCA<sub>(tsm)</sub>, SIFRSON<sub>(tsm)</sub>, SIFTESTS1<sub>(tsm)</sub>, SIFDAC<sub>(tsm)</sub>, и SIFTSTOP<sub>(tsm)</sub> имеют высокий уровень в пределах текущего состояния. В противном случае сигналы имеют низкий уровень.

## Длительность состояний автомата TSM

Длительность каждого состояния автомата программируется индивидуально битами SIFREPEATx. Длительность равна SIFREPEATx+1 тактов выбранной частоты тактирования. Например, если для состояния определено SIFREPEATx=3 и SIFACLK=1, длительность такого состояния будет равна четырём тактам частоты ACLK. Ввиду синхронизации сигналов, на длительность состояния также будет влиять выбор источника тактирования для предыдущего состояния, как показано в таблице 22-5.

Таблица 22-5. Длительность состояний автомата TSM

SIFACLK		Длительность состояния, Т
Для предыдущего состояния	Для текущего состояния	
0	0	$T = (\text{SIFREPEAT}x+1) \times 1/f_{\text{SIFCLK}}$
0	1	$(\text{SIFREPEAT}x) \times 1/f_{\text{ACLK}} < T \leq (\text{SIFREPEAT}x+1) \times 1/f_{\text{ACLK}}$
1	0	$(\text{SIFREPEAT}x+1) \times 1/f_{\text{SIFCLK}} \leq T < (\text{SIFREPEAT}x+2) \times 1/f_{\text{SIFCLK}}$
1	1	$T = (\text{SIFREPEAT}x+1) \times 1/f_{\text{ACLK}}$

### Выбор источника тактирования состояния автомата TSM

Источник тактирования выбирается независимо для каждого состояния автомата TSM. Такими источниками могут служить либо частота ACLK, либо высокочастотный источник тактирования, что определяется битом SIFACLK. Когда SIFACLK=1, для тактирования текущего состояния используется ACLK, когда SIFACLK=0, используется высокочастотный генератор. В качестве высокочастотного источника тактирования может быть выбран источник SMCLK или встроенный генератор модуля TSM, что определяется битом SIFCLKEN. Частота ACLK может быть поделена на 1,2,4, или 8 в зависимости от состояния бит SIFDIV2x, частота высокочастотного генератора также может быть поделена на 1,2,4 или 8 в зависимости от состояния бит SIFDIV1x.

Установкой в «1» бита SIFCLKON можно принудительно включить высокочастотный генератор на длительность текущего состояния, если этот генератор в данном состоянии не используется для тактирования. Если в качестве высокочастотного генератора выбран DCO, он остаётся постоянно включенным, независимо от настроек низкопотребляющих режимов MSP430.

Номинальная частота внутреннего генератора модуля TSM переключается между 1 и 4 МГц битом SIFFNOM и может подстраиваться в пределах от -40% до +35% с номинальным шагом 5% при помощи бит SIFCLKFQx. Точное значение частоты и шага подстройки имеет разброс, см. уточнённые параметры в документации на конкретный МК.

При помощи частоты ACLK можно измерить точное значение частоты внутреннего генератора модуля TSM. Когда SIFCLKEN=1 и SIFCLKGON=1 регистр SIFCNT3 сброшен, а по следующему фронту сигнала ACLK SIFCNT3 является счётчиком тактов внутреннего генератора. Если SIFNOM=0 в регистре SIFCNT3 сохраняется число тактов внутреннего генератора за один период частоты ACLK, а при SIFNOM=1 за четыре периода частоты ACLK. При попытке доступа к SIFCNT3 во время счёта будет возвращено значение 01h.

## Условие останова автомата TSM

Последнее из состояний автомата TSM помечается битом SIFSTOP=1. Длительность этого состояния всегда равна одному такту частоты SIFCLK вне зависимости от настроек SIFACLK и SIFREPEATx. Когда автомат переходит в состояние с установленным битом SIFSTOP, генерируется флаг прерывания SIFIFG1.

## Тестовые состояния автомата TSM

Между состояниями автомата TSM могут быть вставлены тестовые состояния для калибровки, определения ухода параметров датчика или измерения внешних сигналов. Это осуществляется установкой бита SIFTTESTD. При этом время между состояниями автомата TSM не изменяется, как показано на рис. 22-9. По завершении тестового состояния бит SIFTTESTD очищается автоматически. Во время тестового состояния сигнал TESTDX является активным, что позволяет управлять входными и выходными каналами. Активный уровень сигнала TESTDX появляется после установки бита SIFTTESTD и завершения текущего состояния автомата TSM.

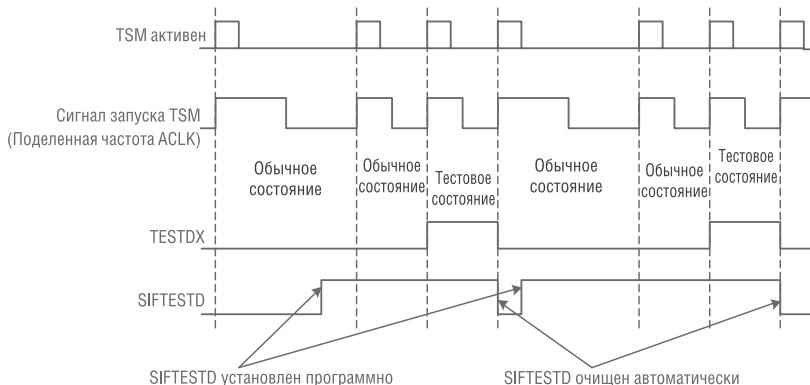


Рис. 22-9. Введение тестовых состояний

## Пример работы автомата TSM

На рис. 22-10 показан пример смены состояний автомата TSM. Состояние регистров TSMx для этого примера показано в таблице 22-6. Сигналы ACLK и SIFCLK не соответствуют масштабу. Начальное состояние автомата TSM – SIFTSM0, конечное – SIFTSM9 с установленным битом SIFSTOP. Показаны только состояния от SIFTSM5 до SIFTSM9.

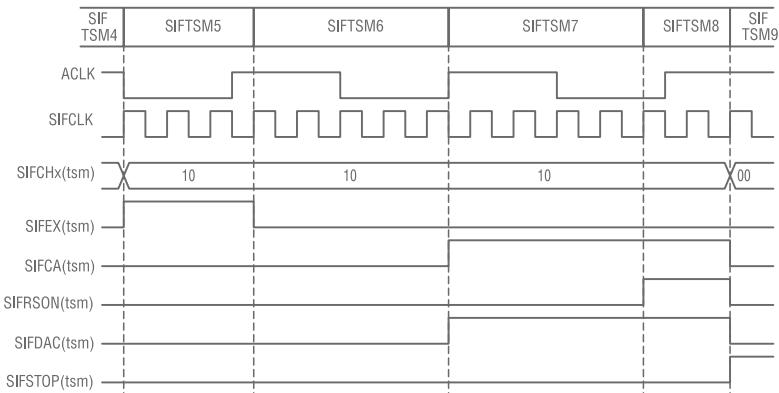


Рис. 22-10. Пример смены состояний автомата TSM

Таблица 22-6. Значения регистров

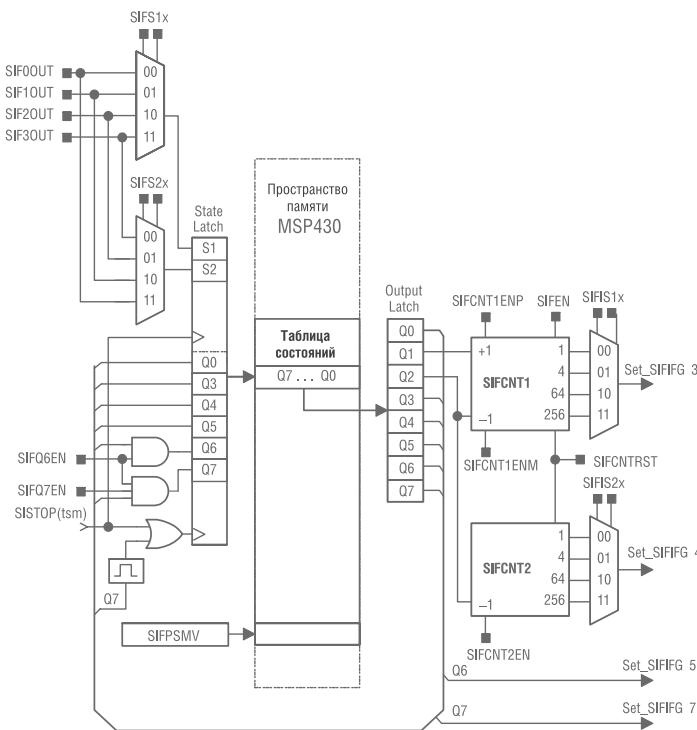
Регистры TSMx	Содержимое регистров TSMx
SIFTSM5	0100Ah
SIFTSM6	00402h
SIFTSM7	01812h
SIFTSM8	00952h
SIFTSM9	00200h

В примере также показано влияние синхронизации сигналов при переключении между SIFCLK и ACLK. В состоянии SIFTSM6, бит SIFACLK установлен, в то время как в предыдущем состоянии он очищен. На диаграмме видно, что длительность состояния SIFTSM6 меньше, чем один период частоты ACLK, а длительность состояния SIFTSM7 почти на один период частоты SIFCLK дольше, чем установлено битами SIFREPEATx.

### 22.2.3. Блок автомата состояний модуля Scan IF

Блок PSM представляет собой программируемый автомат конечных состояний, используемый для определения наличия вращения и его направления. Таблица состояний автомата хранится в памяти микроконтроллера MSP430 (флэш, ПЗУ либо ОЗУ). Автомат состояний измеряет скорость вращения и контролирует генерацию прерываний, анализируя состояние входов от блока аналоговых формирователей и автомата генерации импульсов. Вектор SIFPSMV должен быть инициализирован в качестве указателя на таблицу состояний автомата PSM. При необходимости может быть использовано несколько таблиц

состояний, переход между ними осуществляется переинициализацией вектора SIFPSMV. Блок схема автомата состояний PSM показана на рис 22-11.



**Рис. 22-11.** Блок схема автомата состояний PSM

### Работа автомата состояний PSM

По спаду сигнала SISTOP(tsm) автомат состояний PSM помещает байт текущего состояния из таблицы состояний в выходной регистр-зашёлку. Для автомата состояний PSM выделен обособленный канал прямого доступа к памяти (DMA), таким образом, все обращения к таблице состояний автомата PSM осуществляются автоматически, без участия ЦПУ.

При отключении модуля Scan IF память текущего и последующего состояний будет сброшена. После включения модуля Scan IF в память будет загружен один из байт, расположенных по адресам от SIFPSMV до SIFPSMV+3. Конкретный адрес байта определяется сигналами S1 и S2.

Сигналы S1 и S2 формируют 2-битное смещение к начальному адресу SIFPSMV, определяя который из байт будет загружен в выходной регистр-зашёлку первым. Например, когда S2=1 и S1=0, первым будет загружен байт, расположенный по адресу SIFPSMV+2. Расположение последующих байт определяется вычислением следующего состояния, которое осуществляется автоматом PSM в зависимости от состояния таблицы и значений сигналов S1 и S2.

#### **Примечание: Частота сигнала SIFSTOP<sub>(tsm)</sub>**

Частота сигнала SIFSTOP<sub>(tsm)</sub> должна быть как минимум в 32 раза меньше частоты MCLK. В противном случае может наблюдаться непредсказуемое поведение модуля.

#### **Вычисление следующего состояния**

Биты 0 и 3 – 5 (Q0, Q3, Q4, Q5), и, если это разрешено битами SIFQ6EN или SIFQ7EN, биты 6 и 7 (Q6, Q7) используются совместно с сигналами S1 и S2 для вычисления следующего состояния. Когда SIFQ6EN=1, для вычисления следующего состояния используется Q6, а когда SIFQ6EN=1 и SIFQ7EN=1, для вычисления следующего состояния используется Q7. Следующее состояние определяется как:

Q7	Q6	Q5	Q4	Q3	Q0	S2	S1
----	----	----	----	----	----	----	----

Когда Q7=0, состояние автомата PSM обновляется по спаду сигнала SIFSTOP<sub>(tsm)</sub> в конце последовательности автомата TSM. После обновления текущего состояния автомат PSM записывает соответствующее значение из таблицы состояний в выходной регистр-зашёлку. Когда Q7=1, следующее состояние вычисляется незамедлительно, без ожидания спада сигнала SIFSTOP<sub>(tsm)</sub> и вне зависимости от состояния SIFQ6EN и SIFQ7EN. При этом состояние будет обновлено при переходе на следующую инструкцию. Таким образом, в худшем случае время между переходами будет составлять 6 тактов частоты MCLK.

#### **Счётчики автомата состояний PSM**

Блок PSM содержит два 8-битных счётчика – SIFCNT1 и SIFCNT2. Входными сигналами для SIFCNT1 являются Q1 и Q2, а входным сигналом для SIFCNT2 является Q2. Состояние счётчиков может быть прочитано через регистр SIFCNT. Если установлен бит SIFCNTRST, каждая операция чтения будет обнулять счётчики, в противном случае их содержимое останется неизменным. Если в процессе чтения счётчика на него поступит счётный импульс, обновление содержащего счётчика будет задержано до завершения чтения. Однако, если в процессе чтения на счётчик поступит несколько счётных импульсов, учтён будет лишь один из них. Когда SIFEN=0, оба счётчика находятся в сброшенном состоянии.

Счётчик SIFCNT1 может инкрементироваться либо декрементироваться в зависимости от Q1 и Q2. Если SIFCNT1ENM=1, счётчик SIFCNT1 декрементируется при переходе к состоянию с установленным битом Q2. Если SIFCNT1ENP=1, счётчик SIFCNT1 инкрементируется при переходе к состоянию с установленным битом Q1. В том случае, если установлен как бит SIFCNT1ENM, так и бит SIFCNT1ENP, и оба бита Q1 и Q2 установлены при переходе к состоянию, содержимое счётчика SIFCNT1 не изменяется.

Счётчик SIFCNT2 декрементируется в зависимости от Q2. Когда SIFCNT2EN=1, SIFCNT2 декрементируется при переходе к состоянию с установленным битом Q2. По первому импульсу после сброса состояние SIFCNT2 изменится с нуля на 255 (OFFh).

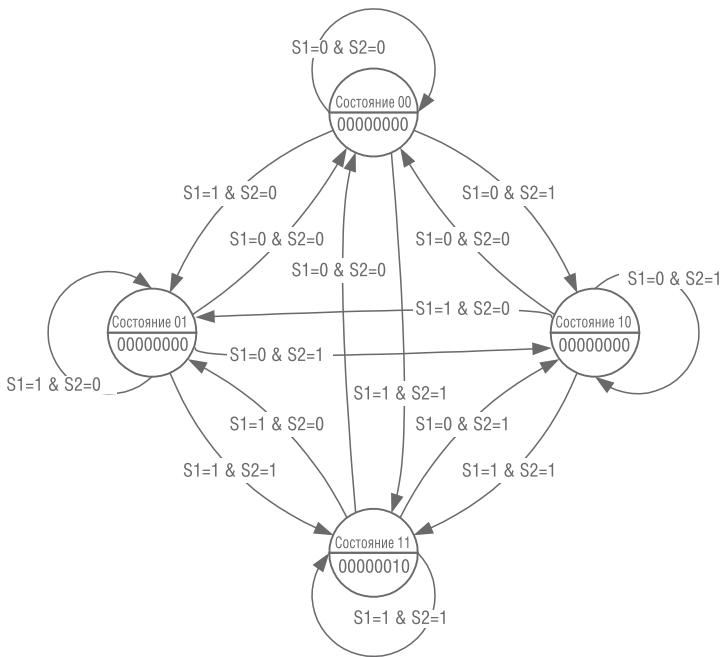
Если в качестве последующего состояния вычислено текущее, счётчики SIFCNT1 и SIFCNT2 также будут инкрементироваться либо декрементироваться в зависимости от состояния бит Q1 и Q2 в текущем состоянии. Например, если текущее состояние 05h, т.е. установлен Q2 и в качестве последующего состояния вычислено текущее, переход от 05h к 05h декрементирует счётчик SIFCNT2 если SIFCNT2EN=1.

### Пример реализации простейшего аппарата конечных состояний

На рис. 22-12 показан пример реализации простейшего аппарата конечных состояний на базе блока PSM. В примере программы показано формирование таблицы состояний и инициализация PSM.

```
; Пример реализации простейшего аппарата конечных
; состояний
SIMPLEST_PSM db 000h      ; состояние 00
                ; (индекс в таблице
                ; состояний 0)
db 000h      ; состояние 01
                ; (индекс в таблице
                ; состояний 1)
db 000h      ; состояние 10
                ; (индекс в таблице
                ; состояний 2)
db 002h      ; состояние 11
                ; (индекс в таблице
                ; состояний 3)

PSM_INIT
    MOV #SIMPLEST_PSM, &SIFPSMV
                ; инициализация
                ; вектора PSM
```



**Рис. 22-12.** Пример реализации простейшего аппарата конечных состояний на базе блока PSM

```

MOV #SIFS20,&SIFCTL3 ; источник S1/S2
MOV #SIFCNT1ENP+SIFCNT1ENM,&SIFCTL4
; в вычислении
; следующего состояния
; Q7 и Q6 не участвуют
; инкремент
; и декремент счёта
; SIFCNT1 разрешены
  
```

Когда автомат PSM находится в состоянии 01, PSM загружает соответствующий байт с индексом 01h из таблицы состояний:

Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
0	0	0	0	0	0	0	0

В этом примере в следующем состоянии автомата TSM S1 и S2 установлены. Для вычисления следующего состояния биты Q5 – Q3 и Q0 состояния 01 таблицы комбинируются с сигналами S1 и S2:

<b>Q7</b>	<b>Q6</b>	<b>Q5</b>	<b>Q4</b>	<b>Q3</b>	<b>Q2</b>	<b>S2</b>	<b>S1</b>
<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>

При переходе к последующему состоянию загружается байт из таблицы для состояния 11:

<b>Q7</b>	<b>Q6</b>	<b>Q5</b>	<b>Q4</b>	<b>Q3</b>	<b>Q2</b>	<b>Q1</b>	<b>Q0</b>
<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>

В этом состоянии Q1 установлен, поэтому счётчик SIFCNT1 инкрементируется. Более сложные автоматы состояний могут быть построены из комбинации простых автоматов.

#### **22.2.4. Регистр отладки модуля Scan IF**

Для целей разработки и отладки модуль Scan IF содержит регистр SIFDEBUG. Для записи в данный регистр доступны только два младших бита, операцию записи можно осуществлять только инструкцией MOV. После записи двух младших бит, в регистре SIFDEBUG будет содержаться различная отладочная информация. После записи значения 00h в регистр SIFDEBUG, чтение из него вернёт последний адрес чтения автомата PSM. После записи значения 01h в регистр SIFDEBUG, чтение из него вернёт текущее состояние автомата TSM и биты Q7 – Q0 автомата PSM. После записи значения 02h в регистр SIFDEBUG, чтение из него вернёт состояние выхода автомата TSM. После записи значения 03h в регистр SIFDEBUG, чтение из него покажет активный регистр ЦАП и его состояние.

#### **22.2.5. Прерывания модуля Scan IF**

Модуль Scan IF имеет один вектор прерываний для семи флагов прерываний, как показано в таблице 22-7. Каждому из флагов соответствует свой бит разрешения прерываний. Когда прерывание разрешено, и бит GIE установлен, флаг прерывания будет вызывать прерывание. Автоматическая очистка флагов не производится, они должны сбрасываться программно.

**Таблица 22-7. Прерывания модуля Scan IF**

<b>Флаг прерывания</b>	<b>Условие возникновения</b>
SIFIFG0	SIFIFG0 устанавливается одним из выходов блока аналоговых формирователей SIFxOUT, выбранным битами SIFIFGSETx.
SIFIFG1	SIFIFG1 устанавливается по фронту сигнала SIFSTOP(tsm).

Таблица 22-7. (Окончание)

Флаг прерывания	Условие возникновения
SIFIFG2	SIFIFG2 устанавливается в начале последовательности автомата TSM.
SIFIFG3	SIFIFG3 выставляется по завершению интервалов счёта счётчика SIFCNT1, задаваемых битами SIFIS1x.
SIFIFG4	SIFIFG4 выставляется по завершению интервалов счёта счётчика SIFCNT2, задаваемых битами SIFIS2x
SIFIFG5	SIFIFG5 устанавливается при переходе автомата в состояния с установленным битом Q6.
SIFIFG6	SIFIFG6 устанавливается при переходе автомата в состояния с установленным битом Q7.

Флаги прерываний SIFIFG3 и SIFIFG4 имеют гистерезис, т.е. при изменении состояния счётчика вблизи уровня прерывания флаг прерывания будет установлен только единожды, как показано на рис. 22-13

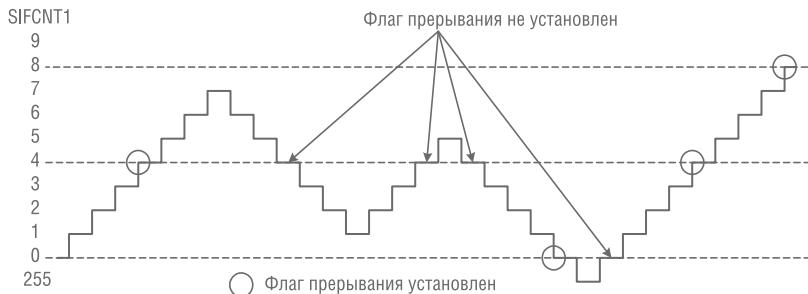


Рис. 22-13. Входной гистерезис при генерации флагов прерываний SIFIFG3 и SIFIFG4

### 22.2.6. Использование модуля Scan IF с датчиками LC-типа

В системах с датчиками LC-типа используются диски, частично покрытые поглощающим материалом для измерения скорости вращения. Скорость вращения измеряется методом возбуждения датчиков и наблюдения результирующих колебаний. Генерация при вращении диска может ослабляться либо не ослабляться. Колебания всегда имеют затухающий характер из-за потерь энергии, но скорость затухания всегда выше в случае, если поглощающий материал диска находится в поле LC-датчика, как показано на рис. 22-14. Измерение генерации LC-датчика осуществляется методом измерения огибающей либо непосредственным измерением колебаний.

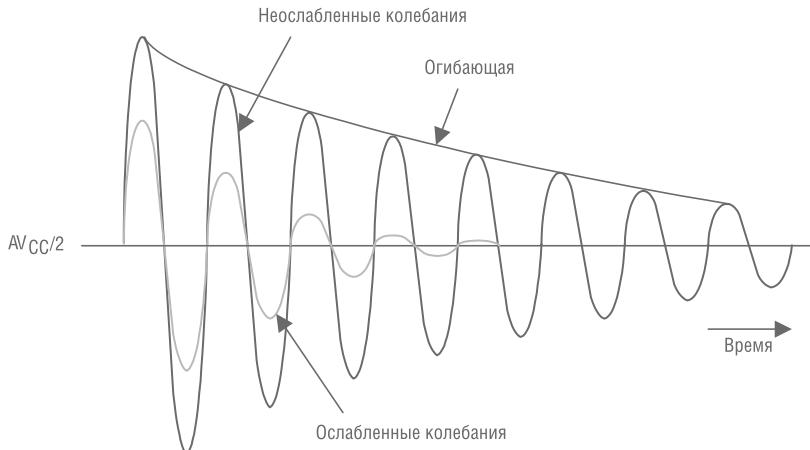


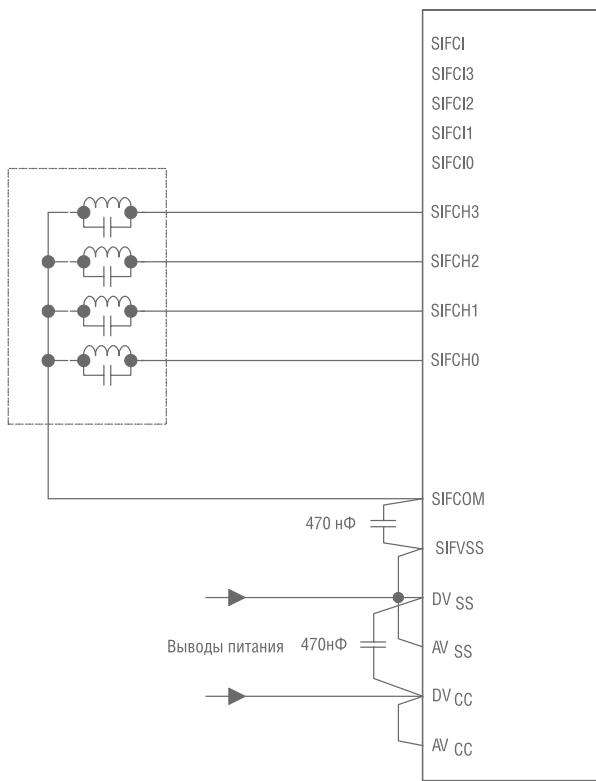
Рис. 22-14. Колебания в LC-датчике

### 22.2.6.1. Непосредственное измерение колебаний

В этом режиме производится сравнение – находится ли амплитуда колебаний после возбуждения датчика выше опорного уровня. Опорное напряжение для компаратора формируется ЦАПом, компаратор детектирует превышение амплитудой колебаний опорного уровня. Если амплитуда колебаний превышает опорный уровень, на выходе компаратора появится серия импульсов в соответствии с колебаниями, а активный выход модуля аналоговых формирователей AFE будет выставлен в «1». Временные параметры измерения и опорный уровень зависят от датчиков, система должна конструироваться таким образом, чтобы разность амплитуд ослабленного и неослабленного сигнала была максимальной. На рис. 22-15 показан пример подключения при непосредственном измерении колебаний.

### 22.2.6.2. Режим измерения огибающей

В этом режиме производится измерение времени затухания после возбуждения счётчика. Огибающая формируется диодами и RC-фильтрами. Опорное напряжение для компаратора формируется ЦАПом, компаратор детектирует превышение амплитудой огибающей опорного уровня. Выходы компаратора и модуля аналоговых формирователей AFE подключены к таймеру Timer1\_A5. Регистры захвата-сравнения таймера Timer1\_A5 используются для измерения времени затухания огибающей. Блок автомата состояний PSM в этом режиме не используется.



**Рис. 22-15.** Пример подключения при непосредственном измерении колебаний

Когда датчики подключены к индивидуальным входам SIFCI<sub>x</sub>, как показано на рис. 22-16, опорный уровень компаратора может быть настроен для каждого датчика индивидуально. Когда все датчики подключены ко входу SIFCI, как показано на рис. 22-17, для всех датчиков используется единый опорный уровень компаратора.

### 22.2.7. Использование модуля Scan IF с резистивными датчиками

Системы, построенные на базе магниторезисторов (GMR) используют для измерения вращения магниты на крыльчатке. Поглощающий материал и магниты изменяют электрическое поведение датчика, при этом можно определить скорость и направление вращения. Скорость вращения измеряется резистивными датчиками путём подключения резистивных делителей на ко-

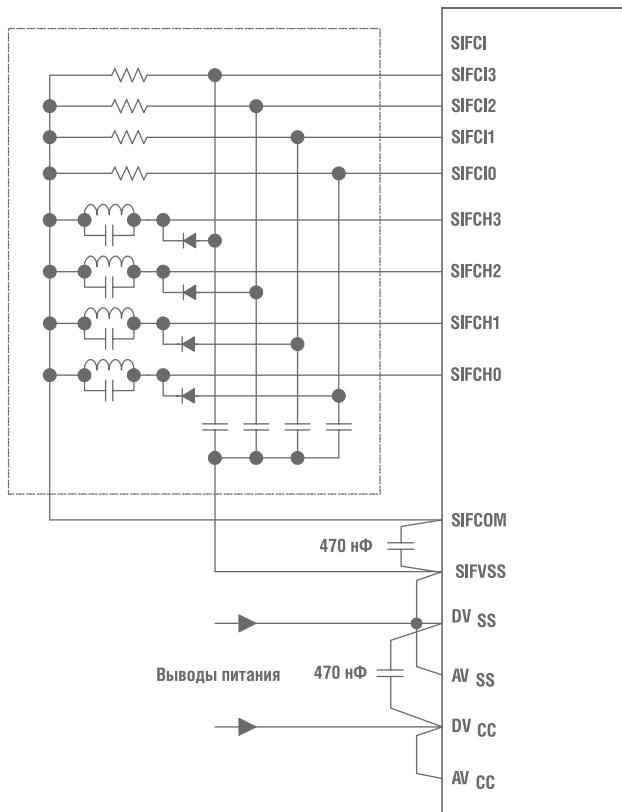
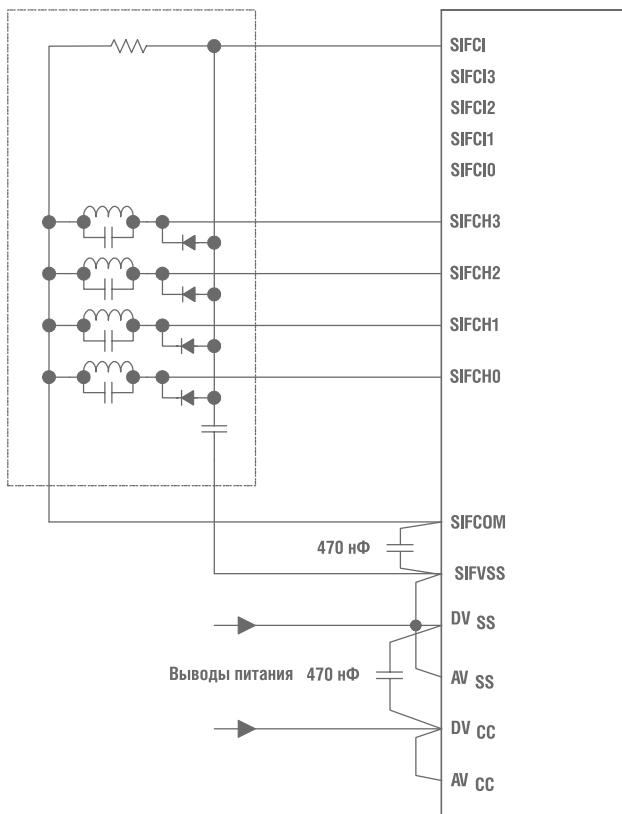


Рис. 22-16. Пример подключения в режиме измерения огибающей

роткое время к «земле». При этом через делитель начинает течь ток. На резисторы воздействует врачающийся диск, создавая различные напряжения в точках делителя. Эти напряжения запоминаются при помощи УВХ. После стабилизации сигналов делитель может быть отключен для снижения энергопотребления. Опорное напряжение для компаратора формируется ЦАПом, компаратор детектирует превышение амплитудой сигнала опорного уровня. Если измеряемое напряжение выше опорного, на выходе компаратора будет высокий уровень. На рис. 22-18 показан пример подключения резистивных датчиков.



**Рис. 22-17.** Пример подключения в режиме измерения огибающей

### 22.2.8. Квадратурное декодирование

Модуль Scan IF может использоваться для декодирования квадратурно-кодированных сигналов. Квадратурными называют сигналы, смещённые друг относительно друга на фазу  $90^\circ$ . Для создания таких сигналов используется специальное расположение датчиков, зависящее от прорезей либо покрытий на диске энкодера. На рис. 22-19 показаны два примера положений датчиков и диаграммы квадратурно-кодированных сигналов.

Для квадратурного декодирования требуется знать предыдущее значение пары сигналов  $S1$  и  $S2$ , а также текущее их значение. Сравнение этих значений

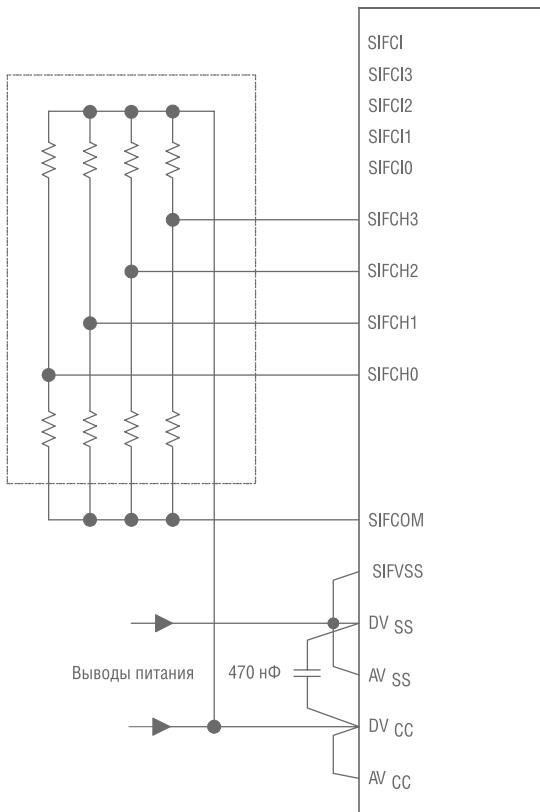


Рис. 22-18. Пример подключения резистивных датчиков

позволяет вычислить направление вращения. Например, если текущее значение 00, то оно может измениться на 01 или 10, в зависимости от направления вращения. Другая смена сигнала будет ошибочной, как показано на рис. 22-20.

Для преобразования состояний в счётные импульсы, прежде всего следует определить, какой угол поворота будет соответствовать счётному импульсу и какие переходы состояний соответствуют этому углу. В данном примере счётному импульсу соответствует полный оборот и переходы 00 → 01 и 00 → 10 при использовании диска с половинным затемнением и датчиков, расположенных под углом 90°. Все возможные варианты переходов помещены в таблицу,

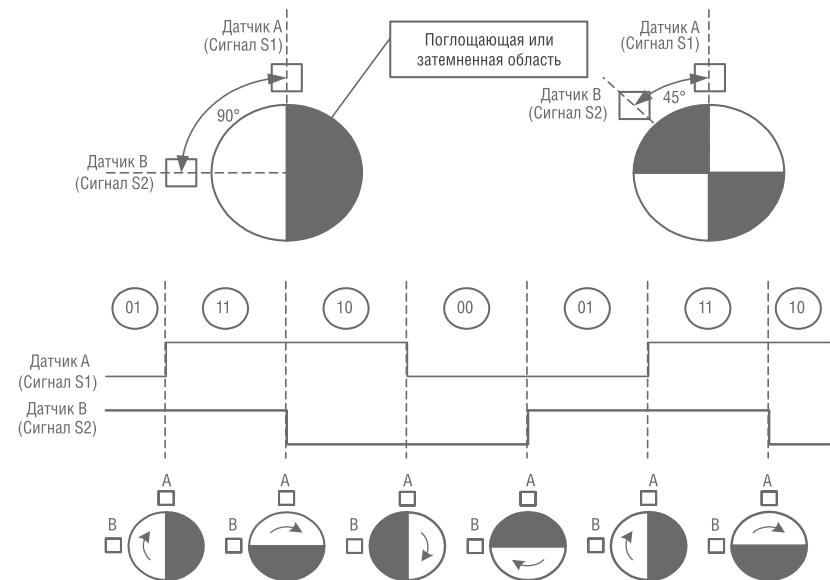


Рис. 22-19. Примеры положений датчиков и диаграммы квадратурно-кодированных сигналов

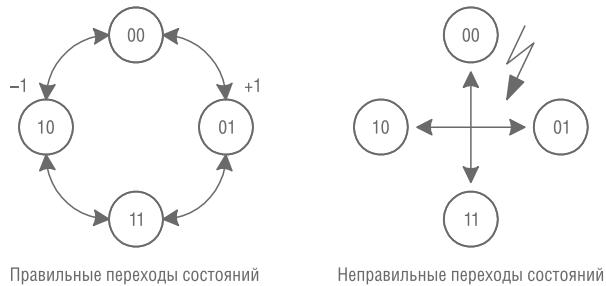


Рис. 22-20. Диаграммы квадратурного декодирования

которая в дальнейшем может быть преобразована в соответствующую таблицу состояний для автомата PSM, как показано в таблице 22-8.

**Таблица 22-8.**

Предыдущее значение сигналов	Предыдущее значение сигналов	Направление движения	Таблица состояний PSM					
			Q6	Q2	Q2	Q3	Q0	Код
Ошибка	-1	+1	Текущая пара квадрантных сигналов					
00	00	Нет движения	0	0	0	0	0	000h
00	01	Поворот вправо на +1	0	0	1	0	1	003h
00	10	Поворот влево на -1	0	1	0	1	0	00Ch
00	11	Ошибка	1	0	0	1	1	049h
01	00	Поворот влево	0	0	0	0	0	000h
01	01	Нет движения	0	0	0	0	1	001h
01	10	Ошибка	1	0	0	1	0	048h
01	11	Поворот вправо	0	0	0	1	1	009h
10	00	Поворот вправо	0	0	0	0	0	000h
10	01	Ошибка	1	0	0	0	1	041h
10	10	Нет движения	0	0	0	1	0	048h
10	11	Поворот влево	0	0	0	1	1	009h
11	00	Ошибка	1	0	0	0	0	040h
11	01	Поворот влево	0	0	0	0	1	001h
11	10	Поворот вправо	0	0	0	1	0	008h
11	11	Нет движения	0	0	0	1	1	009h

## 22.3. Регистры модуля Scan IF

Регистры модуля Scan IF перечислены в таблице 22-9.

**Таблица 22-9.**

Регистр	Краткое название	Тип	Адрес	Начальное состояние
Отладочный регистр модуля Scan IF	Отладочный регистр модуля Scan IF	Чтение/запись	01B0h	Не изменяется
Счётчики модуля Scan IF 1 и 2	SIFCNT	Чтение/запись	01B2h	Обнулён по сбросу POR
Вектор блока PSM модуля Scan IF	SIFPSMV	Чтение/запись	01B4h	Не изменяется

Таблица 22-9. (Окончание)

Регистр	Краткое название	Тип	Адрес	Начальное состояние
Регистр управления модулем Scan IF №1	SIFCTL1	Чтение/запись	01B6h	Обнулён по сбросу POR
Регистр управления модулем Scan IF №2	SIFCTL2	Чтение/запись	01B8h	Обнулён по сбросу POR
Регистр управления модулем Scan IF №3	SIFCTL3	Чтение/запись	01BAh	Обнулён по сбросу POR
Регистр управления модулем Scan IF №4	SIFCTL4	Чтение/запись	01BCh	Обнулён по сбросу POR
Регистр управления модулем Scan IF №5	SIFCTL5	Чтение/запись	01BEh	Обнулён по сбросу POR
ЦАП DAC0 модуля Scan IF	SIFDACR0	Чтение/запись	01C0h	Не изменяется
ЦАП DAC1 модуля Scan IF	SIFDACR1	Чтение/запись	01C2h	Не изменяется
ЦАП DAC2 модуля Scan IF	SIFDACR2	Чтение/запись	01C4h	Не изменяется
ЦАП DAC3 модуля Scan IF	SIFDACR3	Чтение/запись	01C6h	Не изменяется
ЦАП DAC4 модуля Scan IF	SIFDACR4	Чтение/запись	01C8h	Не изменяется
ЦАП DAC5 модуля Scan IF	SIFDACR5	Чтение/запись	01CAh	Не изменяется
ЦАП DAC6 модуля Scan IF	SIFDACR6	Чтение/запись	01CEh	Не изменяется
ЦАП DAC7 модуля Scan IF	SIFDACR7	Чтение/запись	01CCh	Не изменяется
Регистр TSM0 модуля Scan IF	SIFTSM0	Чтение/запись	01D0h	Не изменяется
Регистр TSM1 модуля Scan IF	SIFTSM1	Чтение/запись	01D2h	Не изменяется
Регистр TSM2 модуля Scan IF	SIFTSM2	Чтение/запись	01D4h	Не изменяется
Регистр TSM3 модуля Scan IF	SIFTSM3	Чтение/запись	01D6h	Не изменяется
Регистр TSM4 модуля Scan IF	SIFTSM4	Чтение/запись	01D8h	Не изменяется
Регистр TSM5 модуля Scan IF	SIFTSM5	Чтение/запись	01DAh	Не изменяется
Регистр TSM6 модуля Scan IF	SIFTSM6	Чтение/запись	01DCh	Не изменяется
Регистр TSM7 модуля Scan IF	SIFTSM7	Чтение/запись	01DEh	Не изменяется

**Таблица 22-9. (Окончание)**

<b>Регистр</b>	<b>Краткое название</b>	<b>Тип</b>	<b>Адрес</b>	<b>Начальное состояние</b>
Регистр TSM8 модуля Scan IF	SIFTSM8	Чтение/запись	01E0h	Не изменяется
Регистр TSM9 модуля Scan IF	SIFTSM9	Чтение/запись	01E2h	Не изменяется
Регистр TSM10 модуля Scan IF	SIFTSM10	Чтение/запись	01E4h	Не изменяется
Регистр TSM11 модуля Scan IF	SIFTSM11	Чтение/запись	01E6h	Не изменяется
Регистр TSM12 модуля Scan IF	SIFTSM12	Чтение/запись	01E8h	Не изменяется
Регистр TSM13 модуля Scan IF	SIFTSM13	Чтение/запись	01EAh	Не изменяется
Регистр TSM14 модуля Scan IF	SIFTSM14	Чтение/запись	01EC <sub>h</sub>	Не изменяется
Регистр TSM15 модуля Scan IF	SIFTSM15	Чтение/запись	01EEh	Не изменяется
Регистр TSM16 модуля Scan IF	SIFTSM16	Чтение/запись	01F0h	Не изменяется
Регистр TSM17 модуля Scan IF	SIFTSM17	Чтение/запись	01F2h	Не изменяется
Регистр TSM18 модуля Scan IF	SIFTSM18	Чтение/запись	01F4h	Не изменяется
Регистр TSM19 модуля Scan IF	SIFTSM19	Чтение/запись	01F6h	Не изменяется
Регистр TSM20 модуля Scan IF	SIFTSM20	Чтение/запись	01F8h	Не изменяется
Регистр TSM21 модуля Scan IF	SIFTSM21	Чтение/запись	01FAh	Не изменяется
Регистр TSM22 модуля Scan IF	SIFTSM22	Чтение/запись	01FC <sub>h</sub>	Не изменяется
Регистр TSM23 модуля Scan IF	SIFTSM23	Чтение/запись	01FEh	Не изменяется

**SIFDEBUG, Отладочный регистр модуля Scan IF. Режим записи**

15	14	13	12		11	10	9	8
<b>Резервные</b>								
W	W	W	W		W	W	W	W
7	6	5	4		3	2	1	0
<b>Резервные</b>						<b>SIFDEBUGx</b>		
W	W	W	W		W	W	W	W

Резервные	Биты 15-2	Резервные
SIFDEBUGx	SIFDEBUGx	<p>Режим регистра SIFDEBUG. Запись этих бит позволяет выбрать возвращаемое значение при чтении регистра 00 – чтение регистра SIFDEBUG вернёт последний адрес чтения автомата PSM.</p> <p>01 – чтение регистра SIFDEBUG вернёт текущее состояние автомата TSM и биты Q7 – Q0 автомата PSM.</p> <p>10 – чтение регистра SIFDEBUG вернёт состояние текущего регистра SIFTSMx.</p> <p>11 – чтение регистра SIFDEBUG вернёт активный регистр ЦАП и его состояние.</p>

### SIFDEBUG, Отладочный регистр модуля Scan IF. Режим чтения после записи значения 00h

15	14	13	12		11	10	9	8
Последний адрес чтения автомата PSM								
r	r	r	r	r	r	r	r	r
7	6	5	4		3	2	1	0
Последний адрес чтения автомата PSM								
r	r	r	r	r	r	r	r	r
<b>Последний адрес PSM</b>		<b>Биты 15-0</b>		После записи значения 00h в регистр SIFDEBUG, чтение из него вернёт последний адрес чтения автомата PSM.				

### SIFDEBUG, Отладочный регистр модуля Scan IF. Режим чтения после записи значения 01h

15	14	13	12		11	10	9	8
<b>0</b>	<b>0</b>	<b>0</b>		<b>Состояние автомата TSM</b>				
r	r	r	r	r	r	r	r	r
7	6	5	4		3	2	1	0
<b>биты Q7 – Q0 автомата PSM</b>								
r	r	r	r	r	r	r	r	r
<b>Не используется</b>		<b>Биты 15-13</b>		После записи значения 01h в регистр SIFDEBUG, чтение этих бит вернёт 0				
<b>Состояние TSM</b>		<b>Биты 12-8</b>		После записи значения 01h в регистр SIFDEBUG, чтение этих бит вернёт состояние регистра указателя TSM				
<b>Биты PSM</b>		<b>Биты 7-0</b>		После записи значения 01h в регистр SIFDEBUG, чтение этих бит вернёт биты Q7 – Q0 автомата PSM				

### SIFDEBUG, Отладочный регистр модуля Scan IF. Режим чтения после записи значения 02h

15	14	13	12		11	10	9	8
<b>Состояние текущего регистра SIFTSMx</b>								
r	r	r	r		r	r	r	r
7	6	5	4		3	2	1	0
<b>Состояние текущего регистра SIFTSMx</b>								
r	r	r	r		r	r	r	r
<b>SIFTSMx</b>	Биты 15-0	После записи значения 02h в регистр SIFDEBUG, чтение этих бит вернёт значение регистра TSM						

### SIFDEBUG, Отладочный регистр модуля Scan IF. Режим чтения после записи значения 03h

15	14	13	12		11	10	9	8
<b>0</b>	<b>Активный регистр ЦАП</b>			<b>0</b>	<b>0</b>	<b>0</b>	<b>Данные ЦАП</b>	
r	r	r	r		r	r	r	r
7	6	5	4		3	2	1	0
<b>Данные ЦАП</b>								
r	r	r	r	r	r	r	r	r
<b>Не используется</b>	Не используется	После записи значения 03h в регистр SIFDEBUG, чтение этого бита вернёт 0						
<b>Активный регистр ЦАП</b>	Биты 14-12	После записи значения 03h в регистр SIFDEBUG, чтение этих бит покажет, какой из регистров управляет ЦАПом						
<b>Не используется</b>	Биты 11-10	После записи значения 03h в регистр SIFDEBUG, чтение этих бит вернёт 0						
<b>Данные ЦАП</b>	Биты 9-0	После записи значения 03h в регистр SIFDEBUG, чтение этих бит вернёт состояние активного регистра ЦАП						

### SIFCNT, Счётчики модуля Scan IF

15	14	13	12		11	10	9	8
<b>SIFCNT2x</b>								
r-(0)	r-(0)	r-(0)	r-(0)		r-(0)	r-(0)	r-(0)	r-(0)
7	6	5	4		3	2	1	0
<b>SIFCNT1x</b>								
r-(0)	r-(0)	r-(0)	r-(0)		r-(0)	r-(0)	r-(0)	r-(0)
<b>SIFCNT2x</b>	Биты 15-8	Эти биты представляют значение счётчика SIFCNT2. Счётчик SIFCNT2 сбрасывается при SIFEN = 0 или при чтении из него при SIFCNTRST = 1.						

<b>SIFCNT1x</b>	Биты 7-0	Эти биты представляют значение счётчика SIFCNT1. Счётчик SIFCNT1 сбрасывается при SIFEN = 0 или при чтении из него при SIFCNTRST = 1.
-----------------	----------	---

**SIFSPMV, Вектор блока PSM модуля Scan IF**

15	14	13	12	11	10	9	8
<b>SIFSPMVx</b>							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
<b>SIFSPMVx</b>							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
<b>SIFSPMVx</b>	Биты 15-0	Вектор SIFPSM. Эти биты определяют адрес первого состояния в таблице автомата PSM.					

**SIFCTL1, Регистр управления модулем Scan IF №1**

15	14	13	12	11	10	9	8
<b>SIFIE6</b>	<b>SIFIE5</b>	<b>SIFIE4</b>	<b>SIFIE3</b>	<b>SIFIE3</b>	<b>SIFIE1</b>	<b>SIFIE0</b>	<b>SIFIFG6</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
<b>SIFIFG5</b>	<b>SIFIFG4</b>	<b>SIFIFG3</b>	<b>SIFIFG2</b>	<b>SIFIFG1</b>	<b>SIFIFG0</b>	<b>SIFTSETD</b>	<b>SIFEN</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
<b>SIFIEx</b>	Биты 15-9	Разрешение прерываний. Эти биты разрешают либо запрещают прерывания по флагам SIFIFGx. 0 – прерывание запрещено 1 – прерывание разрешено					
<b>SIFIFG6</b>	Бит 8	Флаг прерывания 6 модуля SIF. Этот бит устанавливается при переходе PSM в состояние с установленным битом Q7. Флаг SIFIFG6 должен очищаться программно. 0 – нет прерывания 1 – есть прерывание					
<b>SIFIFG5</b>	Бит 7	Флаг прерывания 5 модуля SIF. Этот бит устанавливается при переходе PSM в состояние с установленным битом Q6. Флаг SIFIFG5 должен очищаться программно. 0 – нет прерывания 1 – есть прерывание					
<b>SIFIFG4</b>	Бит 6	Флаг прерывания 4 модуля SIF. Этот бит устанавливается при достижении счётчиком SIFCNT2 условия, определяемого битами SIFIS2x. Флаг SIFIFG4 должен очищаться программно. 0 – нет прерывания 1 – есть прерывание					

<b>SIFIFG3</b>	Бит 5	Флаг прерывания 3 модуля SIF. Этот бит устанавливается при достижении счётчиком SIFCNT1 условия, определяемого битами SIFIS1x. Флаг SIFIFG3 должен очищаться программно. 0 – нет прерывания 1 – есть прерывание
<b>SIFIFG2</b>	Бит 4	Флаг прерывания 2 модуля SIF. Этот бит устанавливается в начале последовательности модуля TSM. Флаг SIFIFG2 должен очищаться программно. 0 – нет прерывания 1 – есть прерывание
<b>SIFIFG1</b>	Бит 3	Флаг прерывания 1 модуля SIF. Этот бит устанавливается по фронту сигнала SIFSTOP <sub>(tsm)</sub> . Флаг SIFIFG1 должен очищаться программно. 0 – нет прерывания 1 – есть прерывание
<b>SIFIFG0</b>	Бит 2	Флаг прерывания 1 модуля SIF. Этот бит устанавливается при условии на выходе SIFxOUT, определяемом битами SIFIFGSETx. Флаг SIFIFG1 должен очищаться программно. 0 – нет прерывания 1 – есть прерывание
<b>SIFTSETD</b>	Бит 1	Введение тестовых состояний. Установка этого бита вводит тестовое состояние между тактами автомата TSM. SIFTTESTD сбрасывается автоматически по завершению тестового состояния. 0 – тестовых состояний нет 1 – есть тестовое состояние между тактами автомата TSM
<b>SIFEN</b>	Бит 0	Разрешение модуля. Установкой этого бита включается модуль Scan IF. 0 – модуль Scan IF выключен 1 – модуль Scan IF включен

### SIFCTL2, Регистр управления модулем Scan IF №2

15	14	13	12	11	10	9	8
<b>SIFDACON</b>	<b>SIFCAON</b>	<b>SIFCAINV</b>	<b>SIFCAX</b>	<b>SIFCISEL</b>	<b>SIFCACI3</b>	<b>SIVSS</b>	<b>SIVCC2</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
<b>SIFSH</b>	<b>SIFTEN</b>	<b>SIFTCH1x</b>		<b>SIFTCHOx</b>		<b>SIFTCH1OUT</b>	<b>SIFTCH0OUT</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
<b>SIFDACON</b>	Бит 15	Разрешение модуля ЦАП. Установкой этого бита включается модуль ЦАП независимо от управляющих сигналов TSM. 0 – модуль ЦАП управляемся блоком TSM 1 – модуль ЦАП включен					

<b>SIFCAON</b>	Бит 14	Разрешение компаратора. Установкой этого бита включается компаратор независимо от управляющих сигналов TSM. 0 – компаратор управляемся блоком TSM 1 – компаратор включен
<b>SIFCAINV</b>	Бит 13	Инверсия выхода компаратора. 0 – выход не инвертирован 1 – выход инвертирован
<b>SIFCAX</b>	Бит 12	Выбор входа компаратора. Этот бит используется для выбора группы каналов входа компаратора. 0 – вход компаратора подключен к одному из каналов SIFCHx, выбранных логикой управления каналами 1 – вход компаратора подключен к одному из каналов SIFCI, определяемому логикой управления каналами и битами SIFCISEL и SIFCACI3.
<b>SIFCISEL</b>	Бит 11	Выбор входа компаратора. Этот бит используется совместно с битом SIFCACI3 для выбора входа компаратора при SIFCAX = 1. 0 – вход компаратора подключен к одному из каналов SIFCIx, выбранных логикой управления каналами и битом SIFCACI3. 1 – вход компаратора подключен к каналу SIFCI
<b>SIFCACI3</b>	Бит 10	Выбор входа компаратора. Этот бит используется для выбора входа компаратора при SIFCAX = 1 и SIFCISEL =0. 0 – вход компаратора выбирается логикой управления каналами 1 – вход компаратора подключен к каналу SIFCI3
<b>SIFVSS</b>	Бит 9	Подключение SIFVSS к УВХ. 0 – «земляной» вывод запоминающего конденсатора УВХ подключен к SIFVSS независимо от управляющих сигналов TSM. 1 – подключение «земляного» вывода запоминающего конденсатора УВХ управляемся сигналами TSM.
<b>SIFVCC2</b>	Бит 8	Разрешение буфера средней точки. 0 – буфер средней точки выключен 1 – буфер средней точки включен, если SIFSH=0
<b>SIFSH</b>	Бит 7	Разрешение УВХ. 0 – УВХ выключен 1 – УВХ включен
<b>SIFTEN</b>	Бит 6	Разрешение схемы возбуждения. 0 – схема возбуждения выключена 1 – схема возбуждения включена
<b>SIFCH1x</b>	Биты 5-4	Этими битами определяется вход компаратора для измерительного канала 1. 00 – входом компаратора является SIFCH0 если SIFCAX = 0 входом компаратора является SIFC10 если SIFCAX = 1 01 – входом компаратора является SIFCH1 когда SIFCAX = 0 входом компаратора является SIFC11 когда SIFCAX = 1 10 – входом компаратора является SIFCH2 когда SIFCAX = 0 входом компаратора является SIFC12 когда SIFCAX = 1 11 – входом компаратора является SIFCH3 когда SIFCAX = 0 входом компаратора является SIFC13 когда SIFCAX = 1

<b>SIFTC0x</b>	Биты 3-2	Этими битами определяется вход компаратора для измерительного канала 0. 00 – входом компаратора является SIFCH0 если SIFCAX = 0 входом компаратора является SIFCIO если SIFCAX = 1 01 – входом компаратора является SIFCH1 когда SIFCAX = 0 входом компаратора является SIFCI1 когда SIFCAX = 1 10 – входом компаратора является SIFCH2 когда SIFCAX = 0 входом компаратора является SIFCI2 когда SIFCAX = 1 11 – входом компаратора является SIFCH3 когда SIFCAX = 0 входом компаратора является SIFCI3 когда SIFCAX = 1
<b>SIFCH1OUT</b>	Бит 1	Выход блока аналоговых формирователей для измерительного канала 1
<b>SIFCH0OUT</b>	Бит 0	Выход блока аналоговых формирователей для измерительного канала 0

### SIFCTL3, Регистр управления модулем Scan IF №3

15	14	13	12	11	10	9	8
<b>SIFS2x</b>		<b>SIFS1x</b>		<b>SIFIS2x</b>		<b>SIFIS1x</b>	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
<b>SIFCS</b>		<b>SIFIFGSETx</b>		<b>SIF3OUT</b>	<b>SIF2OUT</b>	<b>SIF1OUT</b>	<b>SIFOOUT</b>
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

<b>SIFS2x</b>	Биты 15-14	Выбор источника сигнала S2. Этими битами выбирается источник сигнала S2 для блока PSM когда SIFCS = 1. 00 – SIFOOUT является источником сигнала S2. 01 – SIF1OUT является источником сигнала S2. 10 – SIF2OUT является источником сигнала S2. 11 – SIF3OUT является источником сигнала S2.
<b>SIFS1x</b>	Биты 13-12	Выбор источника сигнала S1. Этими битами выбирается источник сигнала S1 для блока PSM когда SIFCS = 1. 00 – SIFOOUT является источником сигнала S1. 01 – SIF1OUT является источником сигнала S1. 10 – SIF2OUT является источником сигнала S1. 11 – SIF3OUT является источником сигнала S1.
<b>SIFIS2x</b>	Биты 11-10	Источник флага прерываний SIFIFG4. 00 – SIFIFG4 выставляется по каждому счётному импульсу счётчика SIFCNT2. 01 – SIFIFG4 выставляется, когда (SIFCNT2 по модулю 4) = 0. 10 – SIFIFG4 выставляется, когда (SIFCNT2 по модулю 64) = 0. 11 – SIFIFG4 выставляется при декременте SIFCNT2 от 1 до 0
<b>SIFIS1x</b>	Биты 9-8	Источник флага прерываний SIFIFG3. 00 – SIFIFG3 выставляется по каждому счётному импульсу любого направления счётчика SIFCNT1. 01 – SIFIFG3 выставляется, когда (SIFCNT1 по модулю 4) = 0. 10 – SIFIFG3 выставляется, когда (SIFCNT1 по модулю 64) = 0. 11 – SIFIFG3 выставляется при переходе SIFCNT1 от OFFh к 0

<b>SIFS1x</b>	Бит 7	Выбор выхода компаратора /входа таймера Timer_A 0 – сигнал SIFEX <sub>(tsm)</sub> и выход компаратора подключены к входам TACCRx. 1 – выходы SIFxOUT подключены к входам TACCRx, определяемым битами SIFS1x и SIFS2x.
<b>SIFIFGSETx</b>	Биты 6-4	Выбор источника флага прерывания SIFIGO. Этими битами определяется условие выставления бита SIFIGO. 000 – SIFIGO выставляется при выставлении SIFOOUT. 001 – SIFIGO выставляется при сбросе SIFOOUT. 010 – SIFIGO выставляется при выставлении SIF1OUT. 011 – SIFIGO выставляется при сбросе SIF1OUT. 100 – SIFIGO выставляется при выставлении SIF2OUT. 101 – SIFIGO выставляется при сбросе SIF2OUT. 110 – SIFIGO выставляется при выставлении SIF3OUT. 111 – SIFIGO выставляется при сбросе SIF3OUT.
<b>SIF3OUT</b>	Бит 3	Выходной бит 3 блока аналоговых формирователей
<b>SIF3OUT</b>	Бит 2	Выходной бит 2 блока аналоговых формирователей
<b>SIF3OUT</b>	Бит 1	Выходной бит 1 блока аналоговых формирователей
<b>SIF3OUT</b>	Бит 0	Выходной бит 0 блока аналоговых формирователей

**SIFCTL4, Регистр управления модулем Scan IF №4**

15	14	13	12	11	10	9	8
<b>SIFCNTRST</b>	<b>SIFCNT2EN</b>	<b>SIFCNT1ENM</b>	<b>SIFCNT1ENP</b>	<b>SIFQ7EN</b>	<b>SIFQ6EN</b>	<b>SIFDIV3Bx</b>	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
<b>SIFDIV3Bx</b>	<b>SIFDIV3Ax</b>			<b>SIFDIV2x</b>	<b>SIFDIV1x</b>		
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
<b>SIFCNTRST</b>	Бит 15	Сброс счётчика. Установка этого бита разрешает сброс регистра SIFCNT после его чтения. 0 – регистр SIFCNT не сбрасывается после чтения 1 – регистр SIFCNT сбрасывается после чтения					
<b>SIFCNT2EN</b>	Бит 14	Разрешение счётчика SIFCNT2. 0 – счётчик SIFCNT2 запрещён 1 – счётчик SIFCNT2 разрешён					
<b>SIFCNT1ENM</b>	Бит 13	Разрешение декремента счётчика SIFCNT1. 0 – декремент счётчика SIFCNT1 запрещён 1 – декремент счётчик SIFCNT1 разрешён					
<b>SIFCNT1ENP</b>	Бит 12	Разрешение инкремента счётчика SIFCNT1. 0 – инкремент счётчика SIFCNT1 запрещён 1 – инкремент счётчик SIFCNT1 разрешён					

<b>SIFQ7EN</b>	Бит 11	Разрешение бита Q7. Этим битом разрешается использование бита Q7 для вычисления следующего состояния автомата PSM при SIFQ6EN = 1. 0 – Q7 не используется для вычисления следующего состояния автомата PSM 1 – Q7 используется для вычисления следующего состояния автомата PSM																																																																																	
<b>SIFQ6EN</b>	Бит 10	Разрешение бита Q6. Этим битом разрешается использование бита Q6 для вычисления следующего состояния автомата PSM 0 – Q6 не используется для вычисления следующего состояния автомата PSM 1 – Q6 используется для вычисления следующего состояния автомата PSM																																																																																	
<b>SIFDIV3Bx</b>	Биты 9-7	Делитель частоты ACLK триггера запуска автомата TSM. Этими битами совместно с битами SIFDIV3Ax определяется коэффициент деления частоты ACLK при запуске триггера автомата TSM.																																																																																	
		<b>SIFDIV3Bx</b> <b>SIFDIV3Ax</b> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th></th><th><b>000</b></th><th><b>001</b></th><th><b>010</b></th><th><b>011</b></th><th><b>100</b></th><th><b>101</b></th><th><b>110</b></th><th><b>111</b></th></tr> </thead> <tbody> <tr><td><b>000</b></td><td>2</td><td>6</td><td>10</td><td>14</td><td>18</td><td>22</td><td>26</td><td>30</td></tr> <tr><td><b>001</b></td><td>6</td><td>18</td><td>30</td><td>42</td><td>54</td><td>66</td><td>78</td><td>90</td></tr> <tr><td><b>010</b></td><td>10</td><td>30</td><td>50</td><td>70</td><td>90</td><td>110</td><td>130</td><td>150</td></tr> <tr><td><b>011</b></td><td>14</td><td>42</td><td>70</td><td>98</td><td>126</td><td>154</td><td>182</td><td>210</td></tr> <tr><td><b>100</b></td><td>18</td><td>54</td><td>90</td><td>126</td><td>162</td><td>198</td><td>234</td><td>270</td></tr> <tr><td><b>101</b></td><td>22</td><td>66</td><td>110</td><td>154</td><td>198</td><td>242</td><td>286</td><td>330</td></tr> <tr><td><b>110</b></td><td>26</td><td>78</td><td>130</td><td>182</td><td>234</td><td>286</td><td>338</td><td>390</td></tr> <tr><td><b>111</b></td><td>30</td><td>90</td><td>150</td><td>210</td><td>270</td><td>330</td><td>390</td><td>450</td></tr> </tbody> </table>		<b>000</b>	<b>001</b>	<b>010</b>	<b>011</b>	<b>100</b>	<b>101</b>	<b>110</b>	<b>111</b>	<b>000</b>	2	6	10	14	18	22	26	30	<b>001</b>	6	18	30	42	54	66	78	90	<b>010</b>	10	30	50	70	90	110	130	150	<b>011</b>	14	42	70	98	126	154	182	210	<b>100</b>	18	54	90	126	162	198	234	270	<b>101</b>	22	66	110	154	198	242	286	330	<b>110</b>	26	78	130	182	234	286	338	390	<b>111</b>	30	90	150	210	270	330	390	450
	<b>000</b>	<b>001</b>	<b>010</b>	<b>011</b>	<b>100</b>	<b>101</b>	<b>110</b>	<b>111</b>																																																																											
<b>000</b>	2	6	10	14	18	22	26	30																																																																											
<b>001</b>	6	18	30	42	54	66	78	90																																																																											
<b>010</b>	10	30	50	70	90	110	130	150																																																																											
<b>011</b>	14	42	70	98	126	154	182	210																																																																											
<b>100</b>	18	54	90	126	162	198	234	270																																																																											
<b>101</b>	22	66	110	154	198	242	286	330																																																																											
<b>110</b>	26	78	130	182	234	286	338	390																																																																											
<b>111</b>	30	90	150	210	270	330	390	450																																																																											
<b>SIFDIV2x</b>	Биты 3-2	Делитель частоты ACLK для автомата TSM. Этими битами выбирается коэффициент деления частоты ACLK для автомата TSM. 00/1 01/2 10/4 11/8																																																																																	
<b>SIFDIV1x</b>	Биты 1-0	Делитель частоты SMCLK для автомата TSM. Этими битами выбирается коэффициент деления частоты SMCLK для автомата TSM. 00/1 01/2 10/4 11/8																																																																																	

### SIFCTL5, Регистр управления модулем Scan IF №5

15	14	13	12		11	10	9	8
<b>SIFCNT3x</b>								
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0	
<b>SIFTSMRP</b> <b>SIFCLFOx</b> <b>SIFFNOM</b> <b>SIFFCLKGON</b> <b>SIFCLKEN</b>								
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

SIFCNT3x	Биты 15-8	Счётчик встроенного генератора. SIFCNT3 считает число тактов встроенного генератора за период частоты ACLK при SIFFNOM = 0 или за 4 периода ACLK при SIFFNOM = 1 после установки как SIFCLKGON так и SIFCLKEN
SIFTSMRP	Бит 7	Режим повторения автомата TSM 0 – каждая последовательность автомата TSM запускается делителем частоты ACLK, установленным битами SIFDIV3Ax и SIFDIV3Bx. 1 – последовательность автомата TSM запускается незамедлительно по завершению предыдущей.
SIFCLFQx	Биты 6-3	Подстройка частоты встроенного генератора. Этими битами управляется частота встроенного генератора. Каждый инкремент или декремент этих бит изменяет частоту генератора примерно на 5%. 0000 – минимальная частота : 1000 – номинальная частота : 1111 – максимальная частота
SIFFNOM	Бит 2	Номинальная частота встроенного генератора 0 4 МГц 1 1 МГц
SIFFCLKGON	Бит 1	Управление встроенным генератором. Когда SIFCLKGON = 1 и SIFCLKEN = 1, запускается калибровка встроенного генератора. SIFCLKGON не используется когда SIFCLKEN = 0. 0 – калибровка встроенного генератора не используется. 1 – калибровка встроенного генератора включена при SIFCLKEN = 1.
SIFCLKEN	Бит 0	Разрешение встроенного генератора. Этим битом определяется высокочастотный источник тактирования для автомата TSM. 0 – высокочастотным источником тактирования для автомата TSM является SMCLK. 1 – высокочастотным источником тактирования для автомата TSM является встроенный генератор модуля Scan IF.

**SIFDACRx, Регистры ЦАП модуля Scan IF**

15	14	13	12	11	10	9	8
0	0	0	0	0	0	<b>Данные ЦАП</b>	
r0	r0	r0	r0	r0	r0	rw	rw
7	6	5	4	3	2	1	0
<b>Данные ЦАП</b>							
rw	rw	rw	rw	rw	rw	rw	rw
<b>Не используются</b>		Биты 15-10	Не используются. Эти биты всегда читаются как «0», при записи в эти биты состояние выхода ЦАП не изменяется.				
<b>Данные ЦАП</b>		Биты 9-0	10-битные данные ЦАП				

## SIFTSMx, Регистры автомата TSM модуля Scan IF

15	14	13	12		11	10	9	8
<b>SIFREPEATx</b>					<b>SIFACLK</b>	<b>SIFSTOP</b>	<b>SIFDAC</b>	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4		3	2	1	0
<b>SIFTTESTS1</b>	<b>SIFRSON</b>	<b>SIFCLKON</b>	<b>SIFCA</b>	<b>SIFEX</b>	<b>SIFLCEN</b>	<b>SIFCHx</b>		
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
<b>SIFREPEATx</b>	Биты 15-11	Этими битами совместно с битом SIFACLK определяется длительность состояния. Битами SIFREPEATx выбирается число тактов для состояния. Оно равно SIFREPEATx + 1.						
<b>SIFACLK</b>	Бит 10	Этим битом выбирается источник тактирования автомата TSM. 0 – источником тактирования автомата TSM является высокочастотный генератор, определяемый битом SIFCLKEN. 1 – источником тактирования автомата TSM является частота ACLK						
<b>SIFSTOP</b>	Бит 9	Этот бит означает завершение последовательности автомата TSM. Длительность такого состояния всегда равна одному периоду высокочастотного генератора, вне зависимости от настроек SIFACLK и SIFREPEATx. 0 – последовательность автомата TSM продолжается на следующем состоянии 1 – завершение последовательности TSM						
<b>SIFDAC</b>	Бит 8	Включение ЦАП. Этим битом включается ЦАП для данного состояния, если SIFDACON = 0. 0 – в данном состоянии ЦАП выключен. 1 – в данном состоянии ЦАП включен.						
<b>SIFTTESTS1</b>	Бит 7	Управление тестовым состоянием автомата TSM. Этими битами выбирается для данного состояния, какие из бит, управляющих каналами, и какие регистры ЦАП будут использованы. 0 – каналами управляют биты SIFTCHOx, для ЦАП используется регистр SIFDACR6 1 – каналами управляют биты SIFTCH1x, для ЦАП используется регистр SIFDacr7						
<b>SIFRSON</b>	Бит 6	Разрешение встроенных регистров-зашёлок. Этими битами разрешаются встроенные регистры-зашёлки для выходного каскада блока аналоговых формирователей AFE. 0 – встроенные регистры-зашёлки запрещены 1 – встроенные регистры-зашёлки разрешены						
<b>SIFCLKON</b>	Бит 5	Разрешение высокочастотного генератора. Установкой этого бита для данного состояния включается высокочастотный генератор при SIFACLK = 1, даже если этот генератор не используется для TSM в текущем состоянии. Когда в качестве высокочастотного генератора используется DCO, генератор DCO будет принудительно включен на время текущего состояния независимо от низкопотребляющего режима MSP430. 0 – высокочастотный генератор выключен в текущем состоянии, если SIFACLK = 1						

<b>SIFCLKON</b>	Бит 5	1 – высокочастотный генератор включен в текущем состоянии, если SIFACLK = 1
<b>SIFCA</b>	Бит 4	Включение компаратора блока TSM. Установкой этого бита осуществляется включение компаратора для текущего состояния, если SIFCAON = 0. 0 – в текущем состоянии компаратор выключен 1 – в текущем состоянии компаратор выключен
<b>SIFEX</b>	Бит 3	Управление схемой возбуждения и УВХ. Этот бит, совместно с битами SIFSH и SIFTEN, разрешает работу схемы возбуждения либо УВХ для текущего состояния. SIFLCEN должен быть установлен в 1 если SIFEX = 1. 0 – схема возбуждения отключена при SIFSH = 0 и SIFTEN = 1. УВХ запрещено при SIFSH = 1 и SIFTEN = 0. 1 – схема возбуждения включена при SIFSH = 0 и SIFTEN = 1. УВХ включено при SIFSH = 1 и SIFTEN = 0.
<b>SIFLCEN</b>	Бит 2	Разрешение LC-датчика. Установкой этого бита отключается демпфирующий транзистор, разрешая на время текущего состояния колебания в LC-датчике при SIFTEN = 1. 0 – все каналы SIFCHx демпфированы встроенными демпферами. Колебания в LC-датчиках отсутствуют. 1 – активный канал SIFCHx не демпфирирован. Колебания в LC-датчик присутствуют.
<b>SIFCHx</b>	Биты 1-0	Выбор входного канала. Этими битами выбирается активный канал для текущего состояния, который будет запитан от схемы возбуждения либо подключен к измерительной схеме. 00 – SIFCH0 01 – SIFCH1 10 – SIFCH2 11 – SIFCH3

Таблица состояний автомата PSM (в памяти микроконтроллера MSP430)

7	6	5	4	3	2	1	0
Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
<b>Q7</b>	Бит 7	Когда Q7 = 1, будет установлен флаг прерывания SIFIFG6. Если SIFQ6EN = 1 и SIFQ7EN = 1 и Q7 = 1, автомат PSM переходит к следующему состоянию незамедлительно, вне зависимости от сигнала SIFSTOP <sub>(tsm)</sub> , Q7 при этом используется для вычисления следующего состояния.					
<b>Q6</b>	Бит 6	Когда Q6 = 1, будет установлен флаг прерывания SIFIFG5. Если SIFQ6EN = 1, Q6 принимает участие в вычислении следующего состояния					
<b>Q5</b>	Бит 5	Бит 5 следующего состояния					
<b>Q4</b>	Бит 4	Бит 4 следующего состояния					
<b>Q3</b>	Бит 3	Бит 3 следующего состояния					
<b>Q2</b>	Бит 2	Когда Q2 = 1, счётчик SIFCNT1 декрементируется, если SIFCNT1ENM = 1, а счётчик SIFCNT2 декрементируется, если SIFCNT2EN = 1.					
<b>Q1</b>	Бит 1	Когда Q1 = 1, счётчик SIFCNT1 инкрементируется, если SIFCNT1ENP = 1.					
<b>Q0</b>	Бит 0	Бит 2 следующего состояния					



## Семейство микроконтроллеров MSP430x4xx

### Руководство пользователя

Руководитель проекта

*Таранков И.В.*

Дизайн обложки

*Георгадзе Е.С.*

Графика

*Писанко В.А.*

Верстка

*Торочков Е.В.*

Подписано в печать 07.06.2005 г. Формат 62×90/16  
Печать офсетная. Бумага ролевая. Гарнитура «HeliosCondensed»  
Обложка – Бумага мел. матовая. Формат 62×64/8  
Усл. печ. л. 26. Тираж 2000 экз. Зак. № 144

Отпечатано в типографии «Гран При», г. Рыбинск.  
Тел. представительства в г. Москва: (095) 180-9500