2.0 Обзор архитектуры CPU

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

Этот документ представляет краткое изложение функций CPU и периферии dsPIC30F2010. Для полного описания их назначения обращайтесь к “dsPIC30F Family Reference Manual” (DS70046).

2.1 Обзор ядра

Ядро использует 24-битное слово инструкции. Программный счётчик (PC) имеет ширину 23 бита вместе с наименьшим значимым битом (LSb), который всегда очищен (смотреть часть 3.1 “Пространство программных адресов”), и большим значимым битом (MSb) игнорируемым в течении нормального выполнения программы, за исключением специальных инструкций. Таким образом PC может адресовать до 4M слов инструкций в пространстве пользовательской программы. Используемый механизм предварительной выборки инструкции помогает повысить производительность. Конструкции программного цикла, свободные от потерь управления счётчика цикла, поддерживают использование инструкций DO и REPEAT, обе из которых прерываются в любой точке.

Пространство рабочих регистров содержит 16x16-бит регистров, каждый из которых может действовать как регистр данных, адреса или смещения. Один рабочий регистр (W15) работает как программный указатель стека для прерываний и подпрограмм.

Пространство данных имеет 64 кбайт (32K слов) и разделено на два блока, к которым ссылаются как к X и Y памяти данных. Каждый блок имеет независимое устройство генерации адреса (AGU). Большинство инструкций работают исключительно через AGU памяти X, что обеспечивает видимость единого унифицированного пространства памяти. *Умножение с накоплением* (MAC), класс инструкций DSP с двойным источником, работает через оба, X и Y AGU, разделяя пространство адресов данных на две части (смотреть часть 3.2 “Пространство адресов данных”). Граница между пространством данных X и Y зависит от устройства и не может быть изменена пользователем. Каждое слово данных состоит из 2 байт, и большинство инструкций могут обращаться к данным как к словам или байтам.

Есть два метода записи данных в программную память:

• Верхние 32 кбайт пространства памяти данных могут отображаться в нижнюю половину (пространство пользователя) программной памяти при любой 16K границе слов программы, определённое через 8-ми битный регистр страницы видимости пространства программы (PSVPAG). Это позволяет любой инструкции получать доступ к пространству программы, как если бы это было пространство данных, с ограничением в котором доступ требует дополнительного цикла. Кроме этого только нижние 16 бит каждого слова инструкции доступны с использованием этого метода.

• Линейный косвенный доступ к 32K словным страницам в пределах области программы так же возможен с использованием любого рабочего регистра, через инструкции чтения и записи таблицы. Инструкции чтения и записи таблицы могут быть использованы для доступа ко всем 24 битам слова инструкции.

Переполнение-свободный циклический буфер (адресация по модулю) поддерживается в обоих, X и Y, адресных пространствах. Это первоначально предполагалось удалять цикл переполнения для DSP алгоритмов. X AGU также поддерживает бит-реверсную адресацию на эффективные адреса места назначения, что существенно упрощает ввод и вывод данных преобразования для алгоритмов radix-2 FFT. Смотреть часть 4.0 “Устройства генерации адресов” для деталей на модульную адресацию или бит-реверсную адресацию.

Ядро поддерживает присущий (не операнд), относительный, точный, непосредственную память, непосредственную регистровую, косвенную регистровую, регистр сдвиг и точный сдвиг режимы адресации. Инструкции связаны с встроенными режимами адресации, зависящих от их функциональных требований.

Для большинства инструкций, ядро способно выполнять чтение памяти данных (или программных данных), чтение рабочего регистра (данные), запись памяти данных и чтение программной (инструкции) памяти за цикл инструкции. Как результат, поддерживается трёхоперандная инструкция, позволяя действие C = A + B выполнять за один цикл.

Встроенный движок DSP значительно повышает производительность и арифметическую способность ядра. Это представляет высокоскоростной 17 разрядный умножитель, 40-битное ALU, два 40-битных аккумулятора и 40-битовая схема барабанного сдвига, которая в одном цикле может выполнить сдвиг до 15 бит вправо или 16 бит влево. Инструкции DSP работают незаметно с остальными инструкциями и разрабатываются для оптимального исполнения в реальном масштабе времени. Инструкции классов MAC могут одновременно выбирать два операнда данных из памяти, пока перемножаются два W регистра. Чтобы допустить параллельную выборку операндов данных, пространство данных разделено для этих инструкций и линейно для всех остальных. Этим достигается прозрачная и подвижная манера, к предназначать определённые рабочие регистры каждому адресному пространству для инструкций класса MAC.

Ядро не поддерживает многоступенчатый конвейер инструкций. Тем не менее используется механизм одноступенчатой упреждающей выборки инструкции, что доступно и частично декодирует инструкцию в цикле перед выполнением, в порядке увеличения доступного времени выполнения. Большинство инструкций выполняется за один цикл, с определённым исключением.

Характеристики ядра исключительно векторная структура обработки для ловушек и прерываний, с 62 независимыми векторами. Исключения состоят до 8 ловушек (4 из которых зарезервированы) и 54 прерываний. Приоритет каждого прерывания основан на определённых пользователем приоритете между 1 и 7 (1 является самым низким приоритетом и 7 является самым высоким) вместе с предопределенным ‘естественным порядком’. Ловушки имеют фиксированные приоритеты, в пределах от 8 до 15.

2.2 Программная модель

Программная модель показана на рис.2-1 и состоит из 16 16-битных рабочих регистров (от W0 до W15), 2-х 40-битных аккумуляторов (ACCA и ACCB), регистра STATUS (SR), регистра страницы таблицы данных (TBLPAG), регистра страницы видимости пространства программы (PSVPAG), регистров DO REPEAT (DOSTART, DOEND, DCOUNT и RCOUNT) и программного счётчика (PC). Рабочие регистры могут действовать как регистры данных, адресов или смещения. Все регистры отображены в памяти. W0 действует как регистр W для файл регистра адресации. Около этих регистров имеем теневой регистр, связанный с каждым из них, как показано на рис.2-1. Теневой регистр используется как регистр временного хранения и может передавать эти значения в или из главных регистров в случае события. Ни какой из теневых регистров не доступен напрямую. Следующие правила применяются для передачи регистров в и из теневых.

• PUSH.S и POP.S

W0, W1, W2, W3, SR (только биты DC, N, OV, Z и C) переданы.

• Инструкция DO

DOSTART, DOEND, DCOUNT теневые будут проталкиваться в начале цикла, и выталкиваться в конце цикла.

Когда байтовая операция выполняется на рабочем регистре, только наименьший значимый байт целевого регистра будет задействован. Тем не менее, польза отображения памяти рабочих регистров быть что обоими, наименьшим и наибольшим значимыми байтами, можно манипулировать через доступы к пространству памяти байтовых данных

2.2.1 Указатель программного стека/указатель фрейма

Устройства dsPIC® DSC содержат программный стек. W15 предназначен в качестве указателя программного стека (SP), и автоматически изменяется исключительно в случае обработки и вызове подпрограмм и возвратах. Тем не менее, W15 может ссылаться на любую инструкцию в такой же манере, как и другие W регистры. Это упрощает чтение, запись и манипуляции с указателем стека (т.е. создание фреймов стека).

Примечание: В порядке защиты против доступов к стеку с нарушением границ, W15<0> всегда очищен.

После сброса W15 установлен в 0x0800. Пользователь может перепрограммировать SP в течении инициализации в любую позицию в пределах пространства данных. W14 предназначен в качестве указателя фрейма стека и определяется с помощью инструкций LNK и ULNK. Тем не менее W14 может ссылаться на любую инструкцию в такой же манере, как и все другие регистры W.

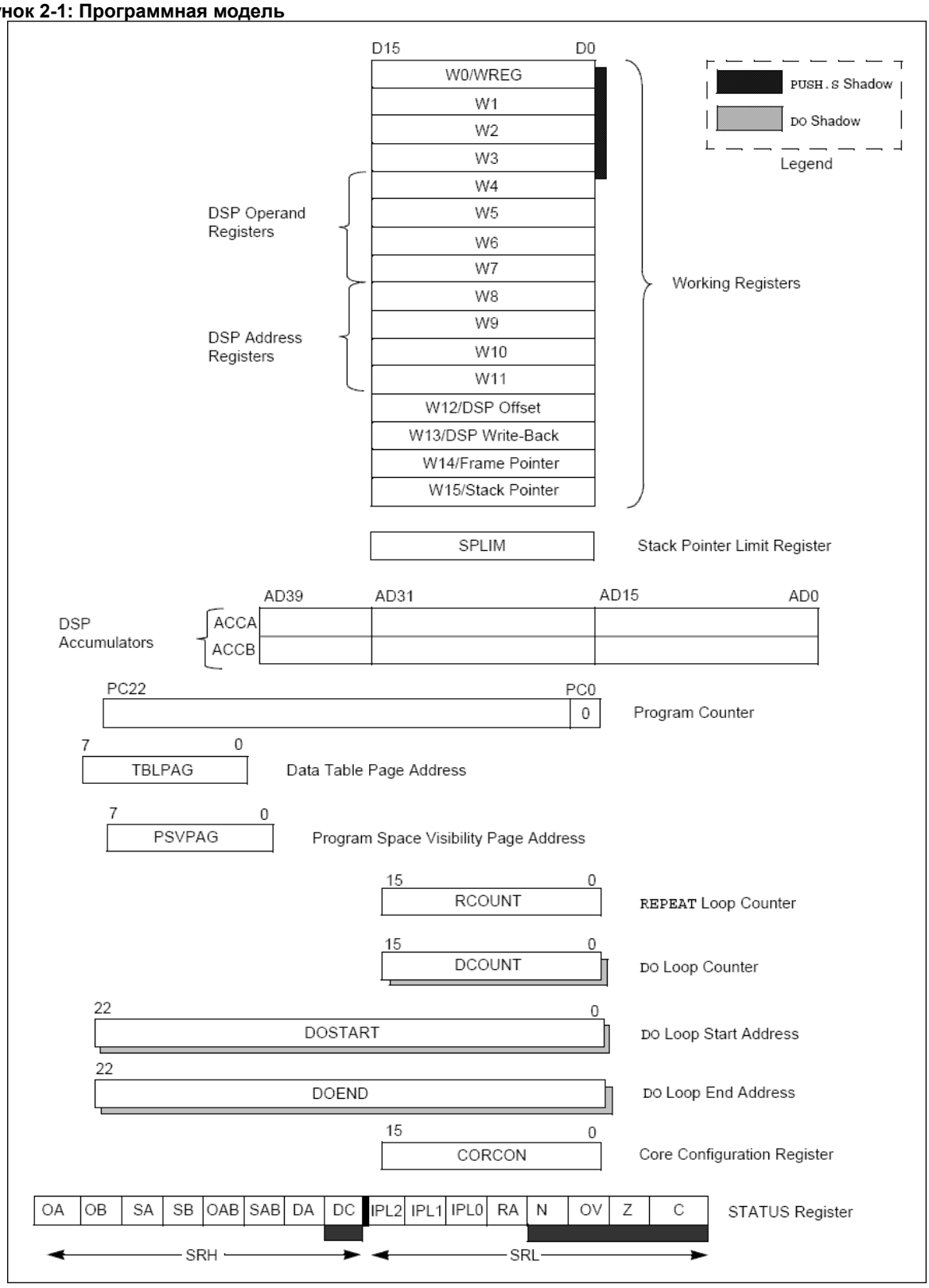
2.2.2 Регистр STATUS

Ядро dsPIC DSC имеет 16-битный регистр STATUS (SR), где LSB отсылает к младшему байту SR (SRL) и MSB как к старшему байту SR (SRH). Смотреть рис.2-1 для формата SR. SRL содержит все рабочие флаги состояния ALU MCU (включая бит Z), так же биты состояния уровня приоритета прерывания CPU, IPL<2:0>, и бит состояния активности REPEAT, RA. Исключительно в течении обработки, SPL есть конкатенирует с MSB программного счётчика PC для формирования полного значения слова что быть затем отправлено в стек. Верхний байт регистра STATUS содержит биты состояния DSP сумматора/вычитателя, бит активности цикла DO (DA) и бит состояния цифрового переноса (DC).

2.2.3 Программный счётчик

Программный счётчик имеет ширину 23 бит. Бит 0 всегда очищен. Следовательно, PC может адресовать до 4M слов инструкций.

Рисунок 2-1: Программная модель



2.3 Поддержка деления

dsPIC DSC устройства представляют 16/16-bit операцию деления дробных со знаком, а так же 32/16-bit и 16/16-bit операции деления целых чисел со знаком и без знака, в форме единственной инструкции итеративных делений. Поддерживаются следующие инструкции и размеры данных:

1. DIVF – 16/16 дробное деление со знаком
2. DIV.sd – 32/16 деление со знаком
3. DIV.ud – 32/16 деление без знака
4. DIV.sw – 16/16 деление со знаком
5. DIV.uw – 16/16 деление без знака

Деление 16/16 подобно 32/16 (то же число итераций), но делимое расширяется нулём или знаком в течении первой итерации.

Инструкции деления должны быть выполнены в течении цикла REPEAT. Любые другие формы выполнения (в том числе последовательность инструкций деления) могут не работать корректно потому что поток инструкции зависит от RCONT. В инструкцию деления не встроена автоматическая установка значения RCONT и поэтому это должно быть явно и правильно определено в инструкции REPEAT, как показано в таблице 2-1(REPEAT может выполнять целевую инструкцию {значение операнда + 1} раз). Счётчик цикла REPEAT может быть установлен до 18 итераций инструкции DIV/DIVF. Таким образом полная операция деления занимает 19 циклов.

Примечание:	Поток деления прерываем. Тем не менее пользователю необходимо сохранить контекст как соответствующий.

Таблица 2-1: Инструкции деления

Инструкции	Функции
DIVF	Дробное деление со знаком: Wm/Wn → W0; Rem → W1
DIV.sd	Деление со знаком: (Wm + 1:Wm)/Wn → W0; Rem → W1
DIV.ud	Деление без знака: (Wm + 1:Wm)/Wn → W0; Rem → W1
DIV.sw (или DIV.s)	Деление со знаком: Wm/Wn → W0; Rem → W1
DIV.uw (или DIV.u)	Деление без знака: Wm/Wn → W0; Rem → W1

2.4 Движок DSP

Движок DSP состоит из высокоскоростного 17-битного умножителя, барабанного сдвигающего устройства и 40-битного сумматора/вычитателя (с двумя целевыми аккумуляторами, логикой округления и насыщения). Движек DSP так же имеет способность выполнять свойственные операции аккумулятор-аккумулятор, что не требует дополнительных данных. Такими инструкциями являются ADD, SUB и NEG.

Движок DSP имеет различные опции выбора через различные биты в регистре конфигурации ядра CPU (CORCON), как указано ниже:

1. Дробное или целое умножение DSP (IF).
2. Умножение DSP со знаком или без знака (US).
3. Стандартное или сходящееся округление (RND).
4. Вкл/Выкл автоматическое насыщение для ACCA (SATA).
5. Вкл/Выкл автоматическое насыщение для ACCB (SATB).
6. Вкл/Выкл автоматическое насыщение для записи в память данных (SATDW).
7. Выбор режима насыщения аккумулятора (ACCSAT).

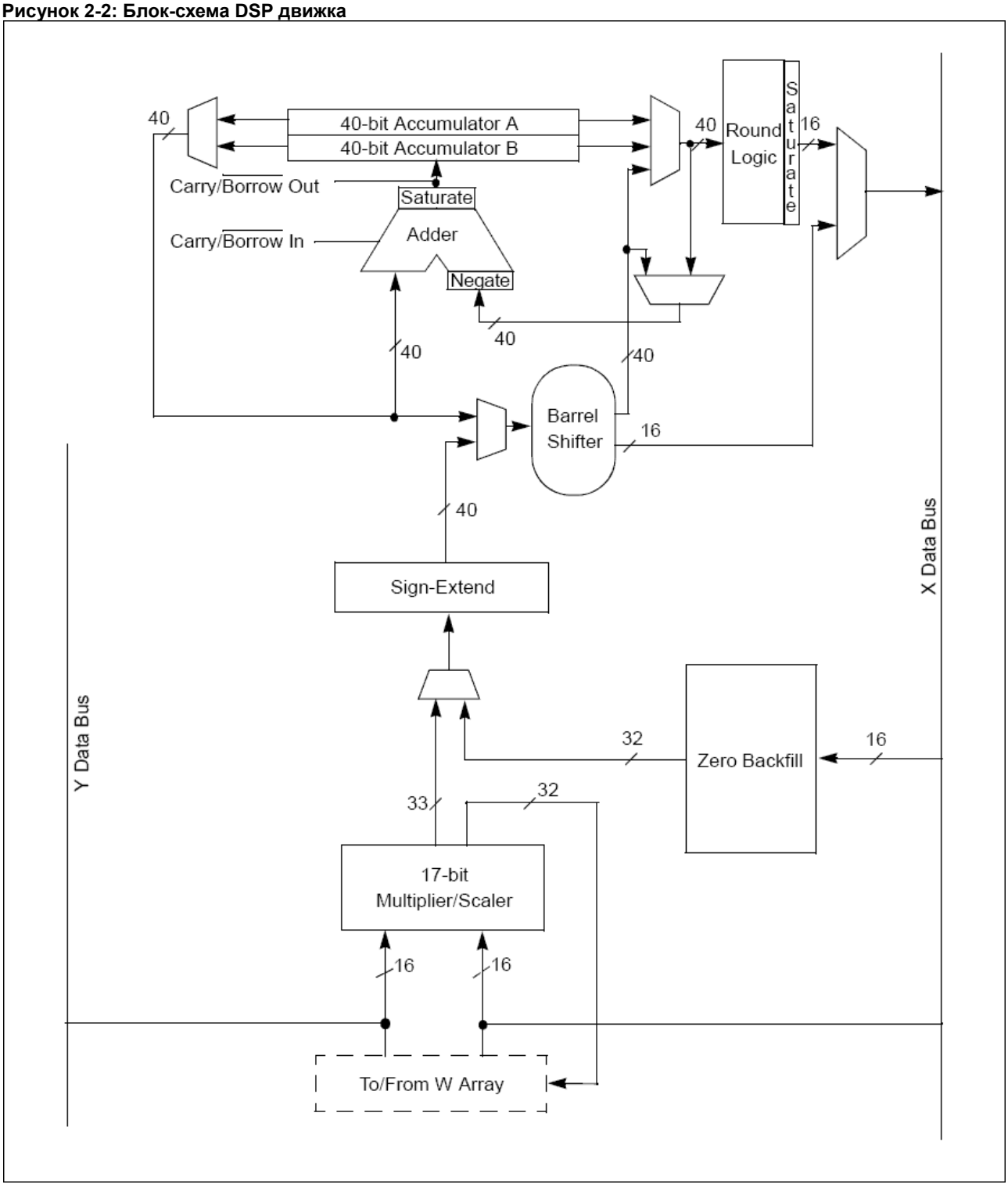
Примечание: Формат CORCON смотреть в таблице 3-3.

Блок-схема движка DSP показана на рисунке 2-2.

Таблица 2-2: Сводка DSP инструкций

Инструкция	Алгебраическая операция	ACC WB?
CLR	A = 0	Да
ED	$A = (x – y)^2$	Нет
EDAC	$A = A + (x – y)^2$	Нет
MAC	A = A + (x * y)	Да
MAC	$A = A + x^2$	Нет
MOVSAC	No change in A	Да
MPY	A = x * y	Нет
MPY.N	A = – x * y	Нет
MSC	A = A – x * y	Да

Рисунок 2-2: Блок-схема DSP движка



2.4.1 Умножитель

17-битный умножитель способен к знаковым и беззнаковым операциям и может мультиплексировать эти выходы используя делитель для поддержки 1.31 дробного (Q31) или 32 битного целого результатов. Без знаковый операнд расширяется нулём до 17-битного входного значения умножителя. Знаковый операнд расширяется знаком до 17-битного входного значения умножителя. На выходе 17-битного умножителя/делителя получаем 33-битное значение, которое расширено знаком до 40 бит. Целые данные в действительности представлены как знаковые двоичные дополненные значения, где MSB определён как знаковый бит. В общем смысле говоря, диапазон N-битного двоично-дополненного целого лежит от -2^{N-1} до $2^{N-1} - 1$. Для 16-битного целого диапазон данных лежит от -32768 (0x8000) до 32767 (0x7FFF), включая 0. Для 32-битного целого, диапазон данных лежит от -2,147,483,648 (0x8000 0000) до -2,147,483,647 (0x7FFF FFFF).

Когда умножитель сконфигурирован для дробного умножения, данные представляются как двоичное дополнение дроби, где MSB определено как знаковый бит и корневая точка подразумевается находится именно после знакового бита (QX format). Область N-битного двоичного дополнения дроби с этим подразумевает корневая точка есть -1.0 до $(1-2^{1-N})$. Для 16-битной дроби, диапазон данных Q15 лежит от -1.0 (0x8000) до 0.999969482(0x7FFF), включая '0' и имея точность 3.01518×10^{-5}. В режиме дроби, операция 16x16 умножения генерирует результат 1.31, что имеет точность 4.65661×10^{-10}.

Такое же умножение используется для поддержки инструкций умножения MCU, что включает 16-битные знаковые целые, без знаковые и смешанные знаковые умножения. Инструкция MUL может быть направленной на использование операнда размером в байт или слово. Байтные операнды могут направлять 16-битный результат, и словные операнды могут направлять 32-битные результаты в специальный регистр(ы) в W пространстве.

2.4.2 Аккумуляторы данных и сумматор/вычитатель

Аккумулятор данных состоит из 40 битного сумматора/вычитателя с логикой автоматического знакового расширения. Это позволяет выбрать два аккумулятора (A или B) как преаккумулятор источник и постаккумулятор приёмник. Для инструкций ADD и LAC, аккумулированные или загруженные данные могут быть произвольно масштабированы через барабанное сдвигающее устройство, до аккумулятора.

2.4.2.1 Сумматор/Вычитатель, переполнение и насыщение

Сумматор/Вычитатель является 40-битным сумматором с дополнительным нулевым входом в с одной стороны и любые, точные или дополненные данные, с дркгого входа. В случае суммирования, вход переноса/заёма имеет активный высокий уровень и на другой вход поданы истинные данные (не дополненные), поскольку в случае вычитания, вход переноса/заёма имеет активно низкий уровень и на другой вход поданы данные в дополнительном коде. Сумматор/Вычитатель генерирует биты SA/SB и OA/OB состояние переполнения, которые защёлкиваются и отражаются в регистре STATUS.

• Переполнение бита 39: это есть катастрофическое переполнение в котором уничтожается знаковый разряд аккумулятора.

• Переполнение битов защиты от 32 до 39: это исправимое переполнение. Этот бит устанавливается всякий раз когда все биты защиты не идентичны друг другу.

Сумматор имеет дополнительный блок насыщения, который, если выбран, контролирует насыщение данных аккумулятора. Использует результат аккумулятора, биты состояния переполнения, описанные выше, и биты режима контроля SATA/B (CORCON<7:6>) и ACCSAT (CORCON<4>) определяющие когда и какое значение насыщено. Шесть битов регистра STATUS обеспечивают поддержку насыщения и переполнения; они есть:

1. OA:

ACCA переполнение в битах защиты

2. OB:

ACCB переполнение в битах защиты

3. SA:

ACCA насыщен (бит 31 переполнен и насыщен) или ACCA переполнены биты защиты и насыщен (бит 39 переполнен и насыщен)

4. SB:

ACCB насыщен (бит 31 переполнен и насыщен) или ACCB переполнены биты защиты и насыщен (бит 39 переполнен и насыщен)

5. OAB:

Логическое или OA и OB

6. SAB:

Логическое или SA и SB

Биты OA и OB модифицируются каждый раз, как данные проходят через сумматор/вычитатель. Когда установлены, они показывают что в самом последнем действии имело место переполнение в аккумуляторных битах защиты (биты с 32 по 39). Биты OA и OB могут так же произвольно генерировать ловушку арифметического предупреждения, когда установлен и разрешён соответствующий бит (OVATE, OVBTE) флага ловушки переполнения в регистре INTCON1 (**обратиться к части 5.9 "Прерывания"**). Это позволяет пользователю принять безотлагательные действия, для примера, для коррекции системного усиления.

Биты SA и SB будут изменены каждый раз когда данные проходят через сумматор вычитатель, но могут быть очищены только пользователем. Когда установлены, они показывают что аккумулятор был переполнен в максимальном диапазоне (бит 31 для бит 32 насыщение, или бит 39 для бита 40 насыщение) и может быть насыщен (если насыщение было разрешено). Когда насыщение не разрешено, SA и SB по умолчанию отображают переполнение бита 39 и таким образом показывают, что случилось катастрофическое переполнение произошло. Если бит COVTE в регистре INTCON1 установлен, биты SA и SB генерируют ловушку арифметического предупреждения, когда насыщение заблокировано.

Биты состояния переполнения и насыщения могут опционально быть рассмотрены регистре статуса (SR) как логическое или SA и SB (в бит SAB). Это позволяет программисту контролировать один бит в регистре STATUS определить, если любой аккумулятор переполнен или один бит определён если любой аккумулятор насыщен. Это должно быть полезно для комплексной арифметики, которая обычно использует оба аккумулятора.

Устройство поддерживает три режима насыщения и переполнения:

1. Бит 39 переполнен и насыщен:

Когда происходит переполнение и насыщение бита 39, логика насыщения загружает максимальное позитивное 9.31 (0x7FFFFFFFFF) или максимальное негативное 9.31 значение (0x8000000000) в целевой аккумулятор. Устанавливается бит SA или SB и остаток устанавливается пока очистит пользователь. Это отсылает к как 'супер насыщение' и обеспечивает защиту против ошибочных данных или проблемы неожиданного алгоритма (в том числе вычисления усиления).

2. Бит 31 переполнен и насыщен:

Когда происходит переполнение и насыщение бита 31, логика насыщения затем загружает максимальное позитивное 1.31 значение (0x007FFFFFFF) или максимальное негативное 1.31 значение (0x0080000000) в целевой аккумулятор. Устанавливается бит SA или SB и остаток устанавливается пока очищен пользователем. Когда это режим насыщения действительно есть, биты защиты не используются (так что биты OA, OB или OAB не будут установлены).

3. Бит 39 катастрофически переполнен

Бит 39 переполняет бит состояния из сумматор быть использован для установки бита SA или SB, которые остаются установленными пока не будут очищены пользователем. Не насыщение операция будет выполнена и аккумулятор быть допускать переполнение (разрушающий этот признак). Если бит COVTE в регистре INTCON1 установлен, катастрофическое переполнение может инициировать ловушку исключения.

2.4.2.2 Аккумулятор ‘Write-Back’ (обратная запись)

Класс инструкций MAC (за исключением MPY, MPY.N, ED и EDAC) могут опционально записывать округлённую версию старшего слова (биты от 31 до 16) аккумулятора, то есть это не нацеливать к инструкция в пространстве памяти данных. Запись выполняется через шину X в объединённое X и Y адресное пространство. Поддерживаются следующие режимы адресации:

1. W13, Регистр направления:

Округлённое содержание не целевого аккумулятора записывается в W13 как дробь 1.15.

2. [W13]+=2, Регистр косвенный с пост-инкрементом:

Округлённое содержание нецелевого аккумулятора записывается в адрес указанный в W13 как 1.15 дробь. W13 затем увеличивается на 2 (для записи слова).

2.4.2.3 Логика округления

Логика округления есть комбинационный блок, который выполняет стандартное (предубеждённое) или сходящееся (объективное) округления функцию в течении записи (сохранения) аккумулятора. Режим округления определяется состоянием бита RND в регистре CORCON. Это генерирует 16 битное, 1.15 значение данных которые проходят в область данных записи логики насыщения. Если округление не указано инструкцией, усечённое 1.15 будет сохранено и наименьшее значимое слово (LSW) будет просто сброшено.

Стандартное округление брать бит 15 аккумулятора, расширить это нулём и добавить в слово ACCxH (биты с 16 по 31аккумулятора). если слово ACCxL (биты с 0 по 15 аккумулятора) быть между 0x8000 и 0xFFFF (включая 0x8000), ACCxH будет увеличен. Если ACCxL быть между 0x0000 и 0x7FFF, ACCxH останется неизменным. Последствие этого алгоритма то, что по последовательности случайных операций округления, значение будет иметь тенденцию быть немного смещённым в плюс.

Сходящийся (или несмещенный) округление функционирует тем же самым способом как условное округление, кроме случая, когда ACCXL равно 0x8000. Если дело обстоит так, исследуется наименьший значащий бит (бит 16 аккумулятора) ACCXH. Если это ‘1’, ACCxH увеличивается. Если это - ‘0’, ACCxH не изменяется. Предполагая, что бит 16 является эффективно случайным в природе, эта схема, удалит любое смещение округления, которое может накапливаться.

Команды SAC И SAC.R сохраняют или усеченную (SAC) или округленную (SAC.R) версии содержания целевого аккумулятора в памяти данных, через X шину (предмет насыщения данных, см. Раздел 2.4.2.4 “Насыщение данных области записи”). Заметьте, что для команд класса MAC, операция обратной записи аккумулятора будет функционировать тем же самым способом, адресуя объединенный MCU (X и Y) пространство данных хотя X шина. Для этого класса инструкций, данные всегда подлежат округлению.

2.4.2.4 Насыщение записи области данных

В дополнение к насыщению сумматора / вычитателя, запись в область данных, может также насыщаться, но без того, чтобы воздействовать на содержание источника аккумулятора. Блок логики насыщения записи области данных принимает 16-битное, 1.15 дробное значение от блока логики округления как его ввод, вместе с состоянием переполнения от первоначального источника (аккумулятора) и 16-разрядного округление сумматора. Они объединены и используются, чтобы выбрать соответствующее 1.15 дробное значение как выход, чтобы записать в область памяти данных.

Если SATDW бит в регистре CORCON установлен, данные (при округлении или усечении) проверен на переполнение и откорректирован соответственно. Для входных данных большее чем 0x007FFF, данные записанные в память вынуждена к максимальному положительному значению 1.15, 0x7FFF. Для входных данных меньше чем 0xFF8000, данные записанные в память вынуждена к отрицательному максимальному значению 1.15, 0x8000. Старший значащий бит источника (бит 39) используется, чтобы определить знак проверяемого операнда.

Если SATDW бит в регистре CORCON не установлен, входные данные всегда пропускаются немодифицированными при всех состояниях.

2.4.3 Барабанное сдвигающее устройство

Барабанное сдвигающее устройство способно к выполнению до 15-разрядных арифметических или логических правых сдвигов в одном цикле, или до 16-разрядных левых сдвигов в единственном цикле. Источник может быть любой из двух DSP аккумуляторов или шина X (чтобы поддержать многоразрядные сдвиги регистра или данных памяти).

Сдвигающее устройство требует двоичное значение определённое обоими величина (число бит) и направления операции сдвига. Положительное значение сдвинет операнд вправо. Отрицательное значение сдвинет операнд в левую сторону. Значение 0 не будет изменять операнд.

Барабанное сдвигающее устройство - шириной 40 битов, таким образом получая 40-разрядный результат для DSP операций сдвига и 16-разрядного результата для MCU операций сдвига. Данные от X шины представлены барабанному сдвигающему устройству между позициями двоичного разряда от 16 до 31 для правых сдвигов, и позиций двоичного разряда от 0 до 15 для левых сдвигов.

3.0 Организация памяти

Примечание: Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

3.1 Адресное пространство программы

Адресное пространство программы - 4M слов инструкций. Это пространство адресуется 24-разрядным значением 23-разрядного PC, эффективными адресами таблицы инструкций (EA) или пространства данных EA, когда пространство программы отображено в область данных, как показано в Таблице 3-1. Заметьте, что адрес области программы увеличивается на два между поочередными словами программы, чтобы обеспечить совместимость с адресацией области данных.

При доступе к области программы пользователя ограничен более низким 4M адресный интервалом инструкций (от 0x000000 до 0x7FFFFE), для всех доступов отличных от TBLRD/TBLWT, которые используют TBLPAG <7>, определённый пользователем или доступ к пространству конфигурации. В Таблице 3-1, командах Read/Write, бит 23 позволяет доступу к ID устройства, ID пользователя и битам конфигурации. Иначе, бит 23 всегда чист.

Примечание: Карта памяти показанная на рис. 3-1 есть концепция, и фактическая конфигурация памяти может измениться для различных устройств в зависимости от доступной памяти.

Рисунок 3-1: Карта пространства памяти программ для dsPIC30F2010

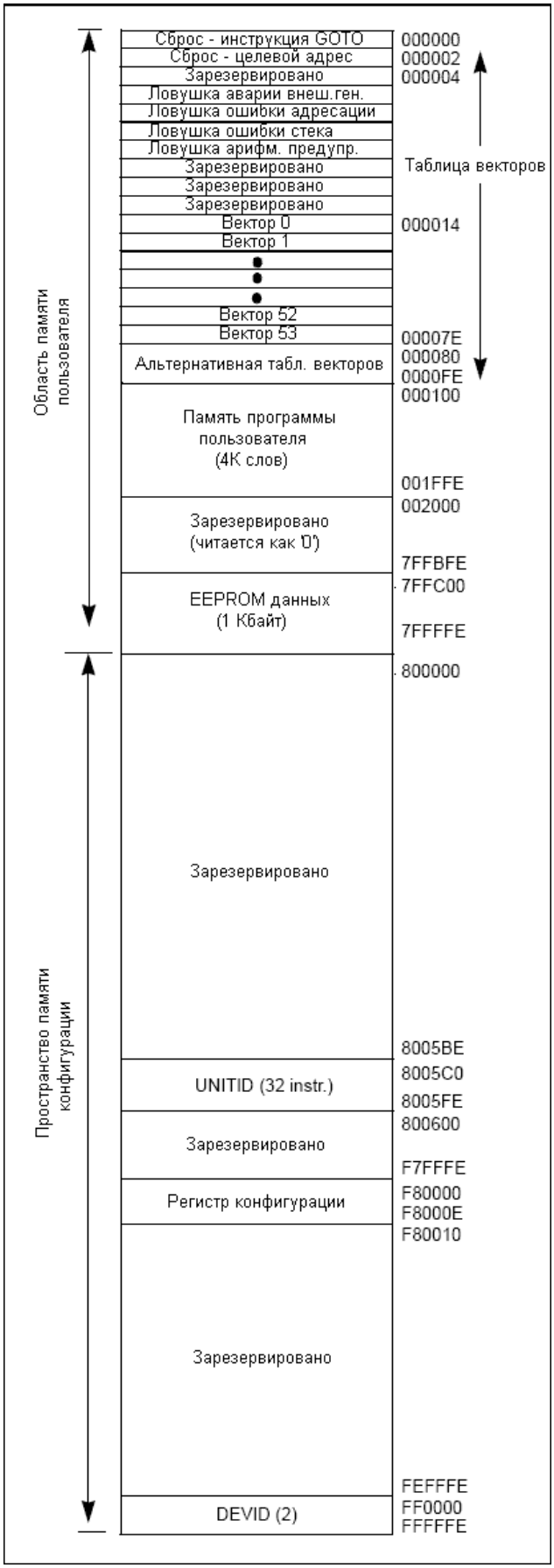
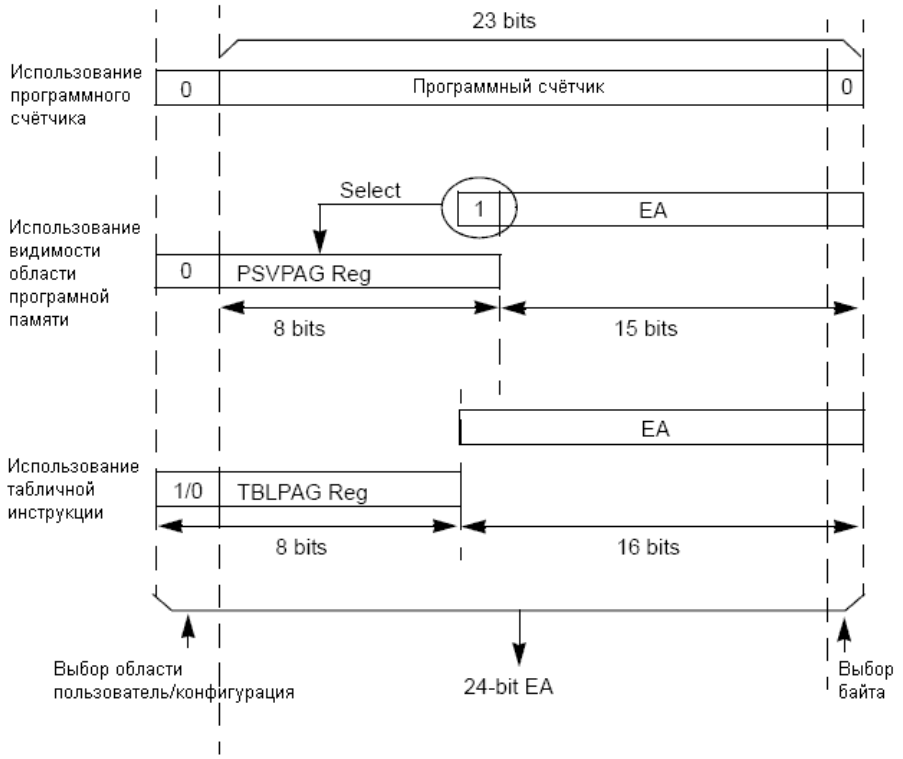


Таблица 3-1: Конструкция адресов области программы

Тип доступа	Область доступа	Адреса области программ				
		<23>	<22:16>	<15>	<14:1>	<0>
Доступ к инструкциям	Пользовательская	0	PC<22:1>			0
TBLRD/TBLWT	Пользовательская (TBLPAG<7> = 0)	TBLPAG<7:0>		Data EA <15:0>		
TBLRD/TBLWT	Конфигурация (TBLPAG<7> = 1)	TBLPAG<7:0>		Data EA <15:0>		
Видимая область программ	Пользовательская	0	PSVPAG<7:0>		Data EA <14:0>	

Рисунок 3-2: Доступ к данным от генерации адресов области программ



Примечание: Видимая область программ не может быть использована для доступа к битам <23:16> слова в памяти программ

3.1.1 Доступ к данным в памяти программ, используя команды таблицы

Эта архитектура выбирает 24-разрядную широкую память программы. Следовательно, команды всегда выравниваются. Однако, поскольку используется модифицированная Гарвардская архитектура, данные могут быть также представлены в области программы.

Имеются два метода обращения к области программы, с помощью которых можно обращаться: через специальные команды таблицы, или через отображение 16К словной области программы в верхнюю половину области данных (см. Раздел 3.1.2 “ Доступ к данным программы с использованием видимости области программы”). Команды TBLRDL и TBLWTL предлагают прямой метод чтения или записи младшего слова любого адреса в пределах области программы, без того, чтобы использовать область данных. Команды TBLRDH и TBLWTH являются единственным методом, посредством которого к верхним 8 битам слова области программы можно обращаться как данным.

PC увеличивается на два для каждого поочередного 24-разрядного слова программы. Это позволяет адресации программы прямо отображается к адресам области данных. Таким образом память программы может расценена как два адресных пространства, каждое из которых имеет ширину одного 16-разрядного, находящихся размер к размеру, каждое с тем же самым адресным интервалом. TBLRDL и TBLWTL обращаются к области, которое содержит самое младшее информационное слово, и TBLRDH, и TBLWTH обращаются к области, которое содержит наиболее старший байт данных. На рисунке 3-2 показано, как работает созданный для таблицы EA и доступ к области данных (PSV = 1). Здесь, P <23:0> обращается к слову области программы, принимая во внимание, что D <15:0> обращается к слову области данных.

Комплект инструкций таблицы обеспечивает перемещение байта или словных данных в или из области программы.

1. TBLRDL: Табличное чтение младшего слова: Читать наименьшее значимое слово по программному адресу;

P<15:0> отображается в D<15:0>.

Byte: Читать один LSBs по программному адресу;

P<7:0> отображается в место назначение байта, когда выбор байта = 0;

P<15:8> отображается в место назначение байта, когда выбор байта = 1.

2. TBLWTL: Табличная запись младшего (обратится к части 6.0 “Flash память программ” для деталей на Flash программирование).

3. TBLRDH: Табличное чтение старшего слова: Читать наибольшее значимое слово программного адреса;

P<23:16> отображается в D<7:0>; D<15:8> всегда = 0.

Байт: читать один MSBs по программному адресу;

P<23:16> отображается в место назначение байта, когда выбор байта = 0;

Байт назначения всегда = 0 когда выбор байта = 1.

4. TBLWTH: Табличная запись старшего (обратится к части 6.0 “Flash память программ” для деталей на Flash программирование).

Рисунок 3-3: Табличный доступ к программным данным (наименьшее значимое слово)

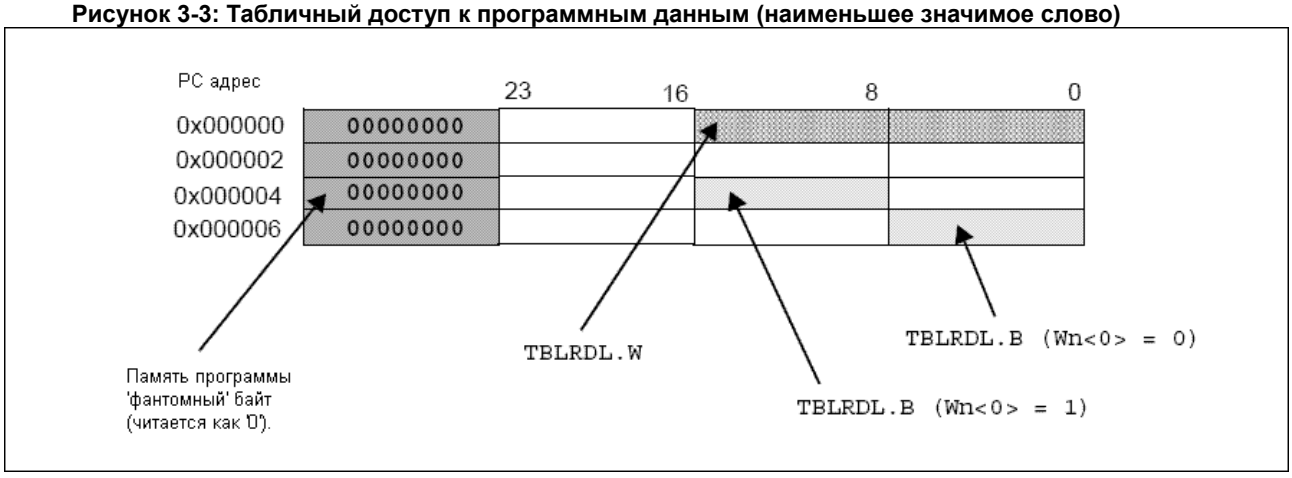
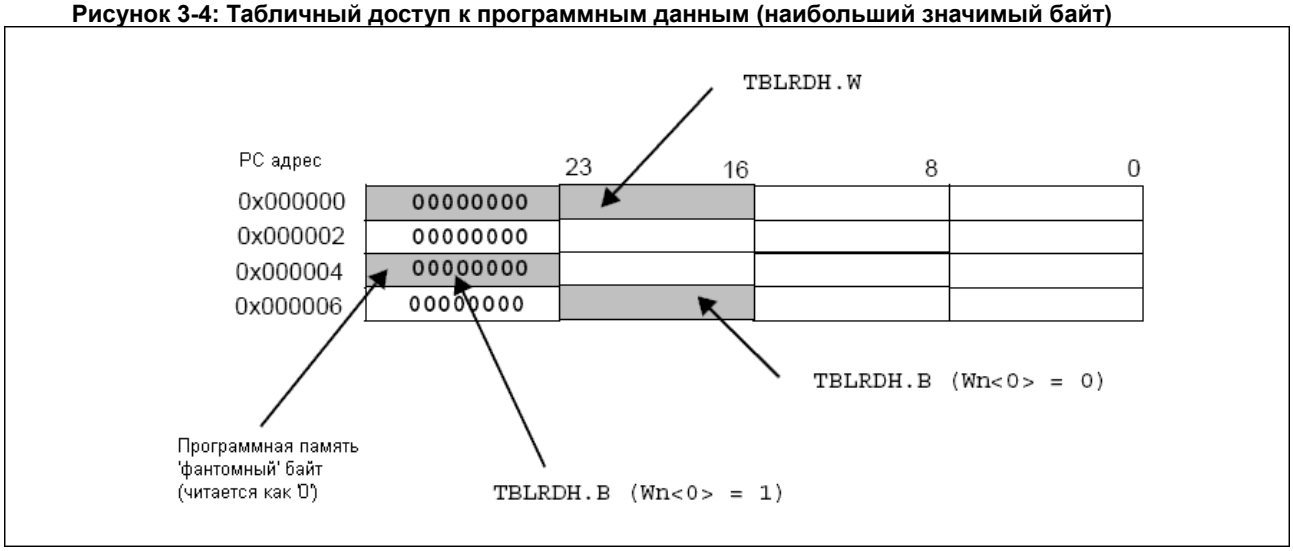


Рисунок 3-4: Табличный доступ к программным данным (наибольший значимый байт)



3.1.2 Доступ к данным из программной памяти, используя видимость программной области

Верхние 32 Кбайта области данных могут опционально быть отображены в любую 16К словную страницу области программы. Это обеспечивает прозрачный доступ к сохранённым константам из X области данных, без необходимости использования специальных инструкций (то есть инструкций TBLRDL/H, TBLWTL/H).

Доступ к области программ происходит через область данных если MSb области данных EA установлен и включена видимость области программы, через установку бита PSV в регистре контроля ядра (CORCON). Функции CORCON обсуждались в части 2.4 “Движок DSP”.

Доступ к данным этой области добавляет дополнительный цикл к выполнению инструкции, поскольку требуется два раза выбирать программную память.

Имейте в виду, что верхняя половина адресов области данных есть всегда часть X области данных. Следовательно , когда DSP операция использует отображение области программы для доступа к этому региону памяти, область Y данных должно обычно содержать состояние (переменную) данных для DSP операций, поскольку X пространство данных должно обычно содержать коэффициенты (константы).

Хотя каждый адрес области данных, 0x8000 и выше, отображается прямо в соответствующий адрес памяти программы (смотреть рис.3-5), только нижние 16 бит 24-битного слова программы используются для содержания данных. Верхние 8 бит должны быть запрограммированы заставлять незаконную инструкцию поддерживать ошибкоустойчивость машины. Обратитесь к “Руководству программиста dsPIC30F/33F” (DS70157) для деталей кодирования инструкций.

Имейте в виду, что для каждого слова памяти программ PC увеличивается на 2, наименьшие значимые 15 бит адресов области данных напрямую отображаются в наименьшие значимые 15 бит в соответствующей области адресов программы. Остальные биты предусмотрены через регистр страницы видимости области программы, PSVPAG<7:0>, как показано на рисунке 3-5.

Примечание: PSV доступ временно блокирован в течении табличного чтения/записи.

Для инструкций использующие PSV, которые выполнены снаружи цикла REPEAT:

• Следующие инструкции требуют один цикл инструкции вдобавок к определённому времени выполнения:

- MAC класс инструкции с выборкой операнда данных
- MOV инструкции
- MOV.D инструкции

• Все другие инструкции требуют два цикла инструкции вдобавок к определённому времени выполнения инструкции.

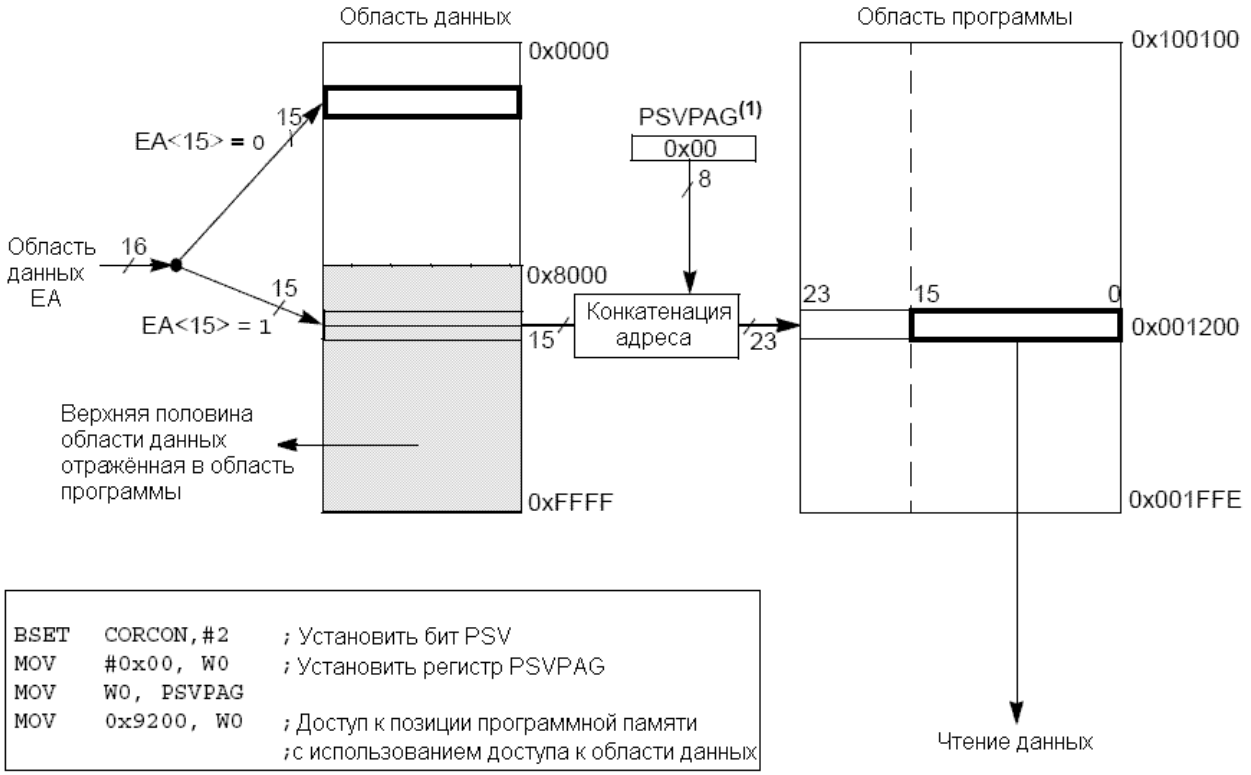
Для инструкций использующих PSV который быть выполненный внутри цикла REPEAT:

• Следующий пример требует два цикла инструкции вдобавок к определённому времени выполнения инструкции:

- Выполнение в первой итерации
- Выполнение в последней итерации
- Выполнение до выхода из цикла подлежащего прерыванию
- Выполнение с входить снова в цикл после обслуженного прерывания

• Любые другие итерации цикла позволены инструкции, доступ к данным используя PSV, выполнение в одном цикле.

Рисунок 3-5: Действие окна пространства памяти на область программы



Примечание: PSVPAG есть 8-битный регистр, содержащий биты <22:15> адресов програмной области (т.е., определяет страницу програмной области которая отображается в верхнюю половину области данных).

3.2 Область адресов данных

Ядро имеет две области данных. Области данных могут быть использоваться отдельно (для некоторых DSP инструкций), или иметь один унифицированный линейный адресный диапазон (для инструкций MCU). Пространства памяти доступны с использованием двух устройств генерации адреса (AGU) и отдельных путей данных.

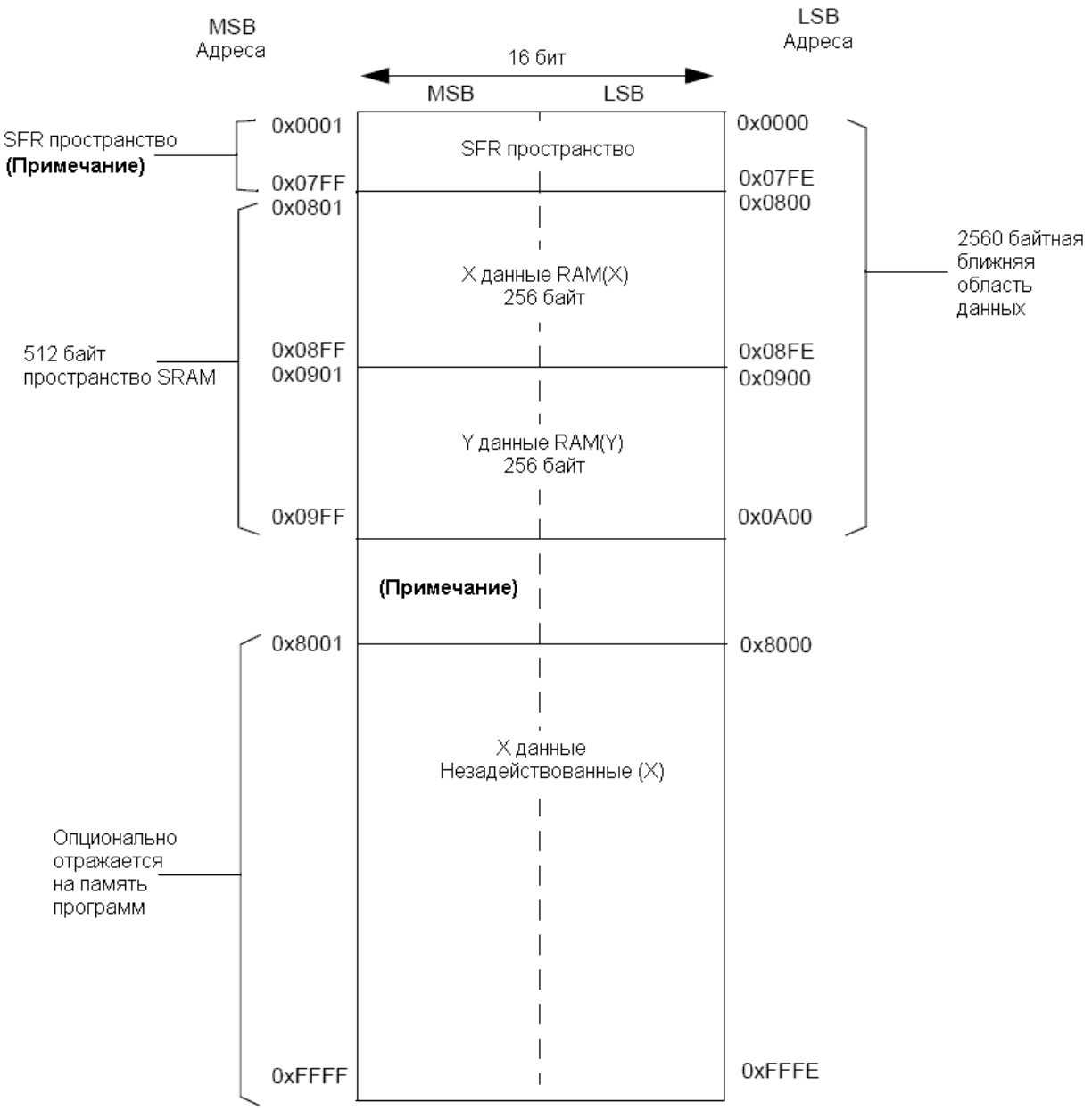
3.2.1 Карта пространства памяти данных

Пространство памяти данных разделено на два блока, X и Y области данных. ключевой элемент этой архитектуры является то, что Y пространство есть подмножество X пространства, и полностью содержится в пределах пространства X. В порядке обеспечения линейной адресации пространства, X и Y области имеют непрерывную адресацию.

Когда выполняется любая инструкция, отличная от инструкций класса MAC, X блок включает 256 байтное пространство адресов данных (включая все Y адреса). Когда выполняется одна из инструкций класса MAC, X блок включает 256 байтное пространство адресов данных исключая адреса блока Y (только для чтения данных). В других словах, все другие инструкции считают целую память данных как одно смешанное адресное пространство. Инструкции MAC класса извлекают адресного пространства Y из пространства данных и эти адреса используют источник EA от w10 И w11. Остальное X пространство данных адресуется с использованием W8 и W9. Оба адресных пространства имеют конкурентный доступ только с инструкциями MAC класса.

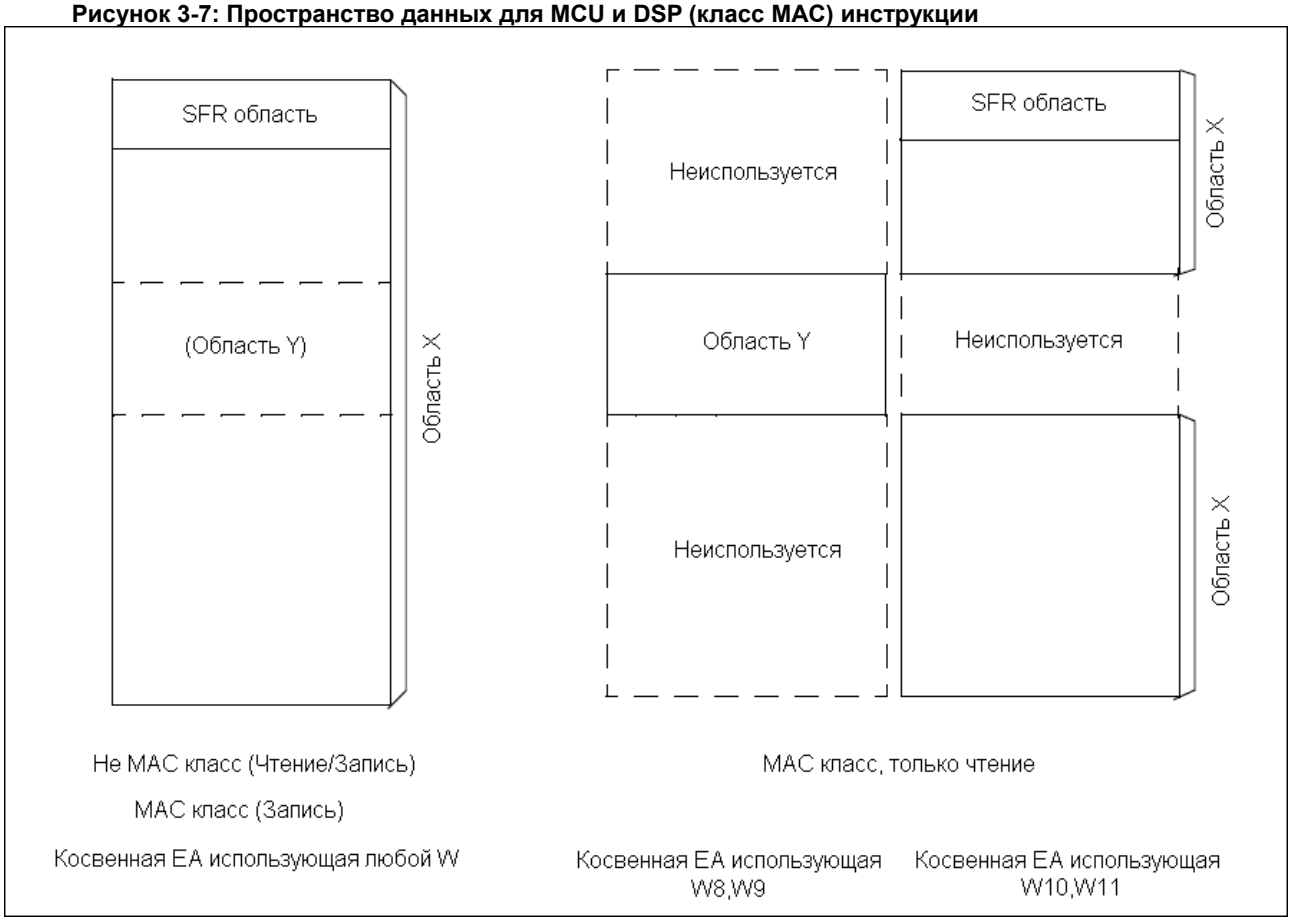
Карта пространства памяти данных показана на рис.3-6.

Рисунок 3-6: Карта пространства памяти данных



Примечание: Незадействованная SFR или SRAM локализации читаются как '0'.

Рисунок 3-7: Пространство данных для MCU и DSP (класс MAC) инструкции



3.2.2 Область данных

X область данных используется всеми инструкциями и поддерживает все режимы адресации. Там есть раздельные шины чтения и записи данных. Шина чтения данных X возвращает данные пути для всех инструкций которые рассматривают пространство данных как комбинированное пространство X и Y адресов. Это есть так же адрес пути к области данных X для инструкции чтения двойного операнда (класс MAC). Шина записи данных X быть только для записи путь к области данных для всех инструкций. Пространство данных X так же поддерживает модульную адресацию для всех инструкций, подлежащих режиму адресации ограничений.

Бит-реверсная адресация поддерживается только для записи в область данных X.

Пространство данных Y используется в концепции с пространством данных X к MAC классу инструкций (CLR, ED, EDAC, MAC, MOVSAC, MPY, MPY.N and MSC) обеспечить два параллельные пути чтения данных. Через шину Y запись не производится. Этот класс инструкций посвящается двум W регистрам указателям, w10 И w11, всегда адресующим пространство данных Y, независимо от пространства данных X, поскольку W8 и W9 всегда адресуют пространство данных X. Имейте в виду, что в течении обратной записи аккумулятора, адрес области данных быть решать комбинацию областей данных X и Y, так чтобы запись происходила через шину X. Следовательно запись может быть в любой адрес в целой области данных.

Пространство данных Y может быть использовано только для выборки данных связанных с MAC классом инструкций. Это так же поддерживает модульную адресацию для автоматических циклических буферов. Конечно, все другие инструкции могут получить доступ к адресам области данных Y через путь данных X, как составная часть линейного пространства.

Граница между пространствами данных X и Y определена как показано на рис.3-6 и не используется программно.

Если указатель EA указывает за пределы данных собственного назначенного адресного пространства, или позиция за пределами физической памяти, возвращается равный нулю байт/слово. Например, ходя адресное пространство Y наблюдается для всех не-MAC инструкций использующих любые режимы адресации, попытка MAC инструкции выбрать данные из этой области, используя W8 и W9(указатели пространства X), вернёт 0x0000.

Таблица 3-2: Результат неправильного доступа к памяти

Предпринимаемые действия	Возвращаемые данные
EA = Незадействованный адрес	0x0000
Доступ к области данных Y в MAC инструкции, используя W8 или W9	0x0000
Доступ к области данных X в MAC инструкции, используя W10 или W11	0x0000

Все эффективные адреса имеют ширину 16 бит и направлены на адресный диапазон 64 Кбайт или 32Кслова.

3.2.3 Ширина пространства данных

Ширина данных ядра 16 бит. Все внутренние регистры организованы как 16-битные слова. Область памяти данных организована в байтной адресации, блоками 16-битной ширины.

3.2.4 Выравнивание данных

Для совместимости с устройствами PIC® MCU и более эффективного использования области памяти данных, система команд поддерживает словные и байтные операции. В области памяти данных и регистрах данные выровнены как слова, но вся область данных EAs разделена на байты. Чтение байта данных происходит при полном чтении слова, которое содержит байт, используя LSb любого EA для определения выбираемого байта. Выбранный байт будет установлен на LSB пути данных X (байтный доступ не возможен из пути данных Y, т.к. MAC класс инструкций может выбирать только слова). Т.е. область данных и регистры организованы как объект двойной байтной ширины с коллективным (слово) декодированием адресов, но раздельными линиями записи. Байт данных записывается только при записи соответствующей стороны массива или регистра который соответствует адресу байта. Как последовательность этого байтного доступа, все эффективные адреса вычисляются (включая генерированные операциями DSP, которые ограничиваются данными словного размера) быть непосредственно масштабированы шаг через память выровненную словами. Например, ядро должно распознавать, что режим косвенной адресации пост-модификацией регистра, [Ws ++], приводит к значению Ws + 1 для байтных операций и Ws + 2 для словных операций.

Все словные доступы должны быть выровнены по чётным адресам. Расположенная с нарушением границ выборка слова не поддерживается, таким образом надо позаботится брать когда смешаны байтные и словные операторы, или транслируются из 8-битного кода MCU. Если предпринимается попытка чтения или записи с нарушением границ, генерируется ловушка ошибки адресации. Если ошибка происходит при чтении, инструкция будет выполнена, при записи инструкция может быть выполненной, но самой записи не произойдёт. В любом случае, ловушка может быть затем выполнена, позволяя системе и/или пользователю проверить состояние машины до выполнения ошибки адресации.

Рисунок 3-8: Выравнивание данных

	15 **MSB** 8	7 **LSB** 0	
0001	Байт 1	Байт 0	0000
0003	Байт 3	Байт 2	0002
0005	Байт 5	Байт 4	0004

Все байтовые загрузки в любой регистр W загружены в LSB. MSB не изменяется.

Расширенная знаком (SE) инструкция позволяет пользователям транслировать 8-битные знаковые данные в 16-битные знаковые значения. Кроме того для 16-битных беззнаковых данных, пользователи могут очистить MSB любого W регистра выполняя расширенную нулём (ZE) инструкцию на соответствующих адресах.

Хотя большинство инструкций позволяют работать с байтными и словными размерами, следует иметь в виду, что некоторые инструкции работают только со словами.

3.2.5 Ближняя область данных

8 Кбайт ‘ближней(near)’ области данных зарезервированы в X адресах памяти между 0x0000 и 0x1FFF, чтобы быть напрямую адресуемыми через 13-битное абсолютное адресное поле внутри всей памяти прямых инструкций. Остальная адресная область X и вся адресная область Y адресуется косвенно. К тому же вся область данных X адресуется с использованием инструкций MOV, которые поддерживают прямую адресацию с 16-битным полем адреса.

3.2.6 Программный стек

Устройство dsPIC DSC содержит программный стек. Регистр W15 используется как указатель стека. Указатель стека всегда указывает на первое доступное свободное слово, и наращивается от низких адресов к высоким адресам. Он пре-декрементируемый для выталкивания из стека (pop) и пост-инкрементируемый для помещения в стек (push), как показано на рисунке 3-9. Имейте в виду, что PC помещается(push) в стек в течении любой инструкции CALL, перед помещением в стек MSB регистра PC расширяется нулём, гарантируя, что MSB всегда очищен.

Примечание:	Помещение в стек PC в течении процесса исключения конкатенирует регистр SRL в MSB PC до помещения в стек.

Регистр ограничителя указателя стека (SPLIM) связан с указателем стека. SPLIM не инициализируется сбросом. Как есть случай для указателя стека, SPLIM<0> ,быть заставлять в ‘0’, потому что все операции стека должны быть выровнены по слову. Всякий раз когда EA генерируется с использованием W15 как указателя источника или цели, адреса таким образом генерированные сравниваются со значением в SPLIM. Если содержание указателя стека (W15) и регистра SPLIM равны и выполняется операция помещения в стек, ловушка ошибки стека не происходит. Ловушка ошибки стека происходит на последующей операции размещения в стеке. Таким образом, например, если желательно вызвать ловушку ошибки стека, когда стек растёт выше адреса 0x2000 в RAM, записать в SPLIM значение 0x1FFE.

Аналогично, ловушка потери значения стека (ошибка стека) генерируется когда адрес указателя стека меньше чем 0x0800, таким образом предохраняя стек от столкновения с пространством специального функционального регистра (SFR). запись в регистр SPLIM должна не быть немедленной следуя к косвенной операции чтения, используя W15.

Рисунок 3-9: Вызов фрейма стека

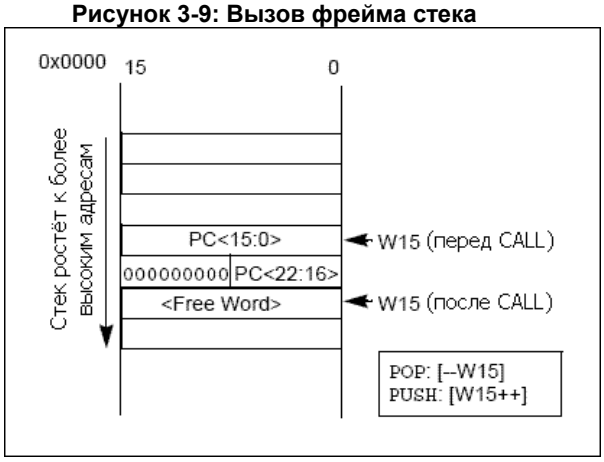


Таблица 3-3: Карта регистров ядра

SFR Имя	Адрес (Дом)	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
W0	0000	W0/WREG																0000 0000 0000 0000
W1	0002	W1																0000 0000 0000 0000
W2	0004	W2																0000 0000 0000 0000
W3	0006	W3																0000 0000 0000 0000
W4	0008	W4																0000 0000 0000 0000
W5	000A	W5																0000 0000 0000 0000
W6	000C	W6																0000 0000 0000 0000
W7	000E	W7																0000 0000 0000 0000
W8	0010	W8																0000 0000 0000 0000
W9	0012	W9																0000 0000 0000 0000
W10	0014	W10																0000 0000 0000 0000
W11	0016	W11																0000 0000 0000 0000
W12	0018	W12																0000 0000 0000 0000
W13	001A	W13																0000 0000 0000 0000
W14	001C	W14																0000 0000 0000 0000
W15	001E	W15																0000 1000 0000 0000
SPLIM	0020	SPLIM																0000 0000 0000 0000
ACCAL	0022	ACCAL																0000 0000 0000 0000
ACCAH	0024	ACCAH																0000 0000 0000 0000
ACCAU	0026	Знаковое расширение (ACCA<39>)								ACCAU								0000 0000 0000 0000
ACCBL	0028	ACCBL																0000 0000 0000 0000
ACCBH	002A	ACCBH																0000 0000 0000 0000
ACCBU	002C	Знаковое расширение (ACCB<39>)								ACCBU								0000 0000 0000 0000
PCL	002E	PCL																0000 0000 0000 0000
PCH	0030	-	-	-	-	-	-	-	-	-	PCH							0000 0000 0000 0000
TBLPAG	0032	-	-	-	-	-	-	-	-	TBLPAG								0000 0000 0000 0000
PSVPAG	0034	-	-	-	-	-	-	-	-	PSVPAG								0000 0000 0000 0000
RCOUNT	0036	RCOUNT																uuuu uuuu uuuu uuuu
DCOUNT	0038	DCOUNT																uuuu uuuu uuuu uuuu
DOSTARTL	003A	DOSTARTL															0	uuuu uuuu uuuu uuu0
DOSTARTH	003C	-	-	-	-	-	-	-	-	-	DOSTARTH							0000 0000 0uuu uuuu
DOENDL	003E	DOENDL															0	uuuu uuuu uuuu uuu0
DOENDH	0040	-	-	-	-	-	-	-	-	-	DOENDH							0000 0000 0uuu uuuu
SR	0042	OA	OB	SA	SB	OAB	SAB	DA	DC	IPL2	IPL1	IPL0	RA	N	OV	Z	C	0000 0000 0000 0000
CORCON	0044	-	-	-	US	EDT	DL2	DL1	DL0	SATA	SATB	SATDW	ACCSAT	IPL3	PSV	RND	IF	0000 0000 0010 0000
MODCON	0046	XMODEN	YMODEN	-	-	BWM<3:0>				YWM<3:0>				XWM<3:0>				0000 0000 0000 0000
XMODSRT	0048	XS<15:1>															0	uuuu uuuu uuuu uuu0
XMODEND	004A	XE<15:1>															1	uuuu uuuu uuuu uuu1
YMODSRT	004C	YS<15:1>															0	uuuu uuuu uuuu uuu0
YMODEND	004E	YE<15:1>															1	uuuu uuuu uuuu uuu1
XBREV	0050	BREN	XB<14:0>															uuuu uuuu uuuu uuuu
DISICNT	0052	-	-	DISICNT<13:0>														0000 0000 0000 0000

Описание: u = неинициализированный бит

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

4.0 Устройства генерации адресов

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

Ядро dsPIC DSC содержит два независимых устройства генерации адресов, это X AGU и Y AGU. Y AGU поддерживает только чтение данных размером в слово для инструкций класса DSP MAC. Оба AGU поддерживают три типа адресации данных:

- Линейную адресацию
- Модульную (Циклическую) адресацию
- Бит-реверсную адресацию

Режимы линейной и модульной адресации данных могут применяться к области данным или области программы. Бит-реверсная адресация может применяться только для адресации в области данных.

4.1 Режимы адресации инструкций

Режимы адресации в таблице 4-1 формируют основу режимов адресации оптимизированную для поддержки специфических особенностей отдельных инструкций. Режимы адресации обеспеченные в инструкциях MAC класса отчасти отличаются от таких же в инструкциях других типов.

4.1.1 Инструкции прямой адресации к регистрам

Большинство инструкций прямой регистрации к регистрам используют 13-битовое поле адреса (f) для непосредственной адресации данных присутствующих в первых 8192 байтах памяти данных (ближняя область данных). Большинство инструкций прямой адресации к регистрам применяют рабочий регистр W0, который обозначается как WREG в этих инструкциях. Местом назначения обычно является тот же файловый регистр или WREG (исключительно с инструкцией MUL), что записывает результат в регистр или регистровую пару. Инструкция MOV позволяет дополнительную гибкость и может получить доступ ко всей области данных.

Таблица 4-1: Основные поддерживаемые режимы адресации

Режим адресации	Описание
Прямая адресация к регистрам	Адрес специального (файлового) регистра определён явно.
Прямая регистровая	Содержимое регистров массива W доступно непосредственно.
Регистровая косвенная	Содержимое Wn используется для формирования эффективного адреса (EA).
Регистровая косвенная с пост-модификацией	Содержимое Wn используется для формирования EA. Затем инкрементируется или декрементируется.
Регистровая косвенная с пре-модификацией	Содержимое Wn инкрементируется или декрементируется, а затем полученное знаковое значение используется для формирования EA.
Регистровая косвенная с регистром смещения	Для формирования EA используется сумма Wn и Wb.
Регистровая косвенная с указанным смещения	Для формирования EA используется сумма Wn и указанного смещения (литерала).

4.1.2 Инструкции MCU

Трёхоперандные инструкции MCU имеют форму:

Операнд 3 = Операнд 1 <функция> Операнд 2

Где операнд 1 всегда является рабочим регистром (т.е., может быть использован только прямой режим адресации), который обозначается Wb. Операндом 2 может быть регистр W, выборка из памяти данных или 5-битный литерал. Результат может быть так же помещён в регистр W или по адресу. Следующие режимы адресации поддерживаются инструкциями MCU:

- Прямая регистровая
- Косвенная регистровая
- Косвенная регистровая с пост-модификацией
- Косвенная регистровая с пре-модификацией
- 5-битный или 10-битный литерал

Примечание:	Не все инструкции поддерживают все режимы адресации указанные выше. отдельные инструкции могут поддерживать различные подмножества этих режимов адресации.

4.1.3 Инструкции перемещения и накопления

Инструкции перемещения и DSP класса накопления обеспечивают большую степень гибкости адресации чем другие инструкции. В добавок к режимам адресации поддерживаемым большинством инструкций MCU, инструкции перемещения и накопления так же поддерживают косвенной регистровый с регистром смещения режим адресации, так же имеют отношение к регистровому индексному режиму.

Примечание:	Для инструкций MOV, режим адресации определённый в инструкции может отличаться для источника и приёмника EA. Тем не менее 4-битная область Wb (регистр смещения) распределена между источником и приёмником (но обычно используется чем-то одним).

В итоге, следующие режимы адресации поддерживаются инструкциями перемещения и накопления:

- Прямая регистровая
- Косвенная регистровая
- Косвенная регистровая с пост-модификацией
- Косвенная регистровая с пре-модификацией
- Косвенная регистровая с регистром смещения (индексная)
- Косвенная регистровая с указанным (литерал) смещением
- 8-битный литерал
- 16-битный литерал

Примечание:	Не все инструкции поддерживают указанные выше режимы адресации. Отдельные инструкции могут поддерживать различное подмножество этих режимов адресации.

4.1.4 Инструкции MAC

DSP инструкции с операндом из двойного источника (CLR, ED, EDAC, MAC, MPY, MPY.N, MOVSAC и MSC), так же относятся к инструкциям MAC, используют упрощённый набор режимов адресации позволяют пользователю эффективно манипулировать указателями данных через регистр косвенных таблиц.

Два регистра источника выбираемого операнда должны быть членами набора {W8, W9, W10, W11}. Для чтения данных, W8 и W9 всегда должны быть направлены в X AGU и W10 и W11 всегда должны быть направлены в Y AGU. Сгенерированные эффективные адреса (перед и после модификации) должны, следовательно, быть правильно адресованы в пределах области данных X для W8 и W9 и области данных Y для W10 и W11.

Примечание:	Косвенный регистр с регистром смещения адресации доступны только для W9 (в X области) и W11 (в Y области).

В итоге следующие режимы адресации поддерживаются инструкциями класса MAC:

- Регистровая косвенная
- Регистровая косвенная с пост-модификацией на 2
- Регистровая косвенная с пост-модификацией на 4
- Регистровая косвенная с пост-модификацией на 6
- Регистровая косвенная с регистром смещения (Индексный)

4.1.5 Другие инструкции

Кроме тех различных режимов адресации очерченных выше, несколько инструкций используют литеральные константы различных размеров. Например, инструкции BRA (ветка) используют 16-битные знаковые литералы для непосредственного определения места назначения ветвления, поскольку инструкция DISI использует 14-битную беззнаковую область литерала. В нескольких инструкциях, таких например как ADD ACC, источник операнда или результат подразумеваются непосредственно через код операции. Определённые операции, такие как NOP, не имеют операндов.

4.2 Модульная адресация

Модульная адресация является методом обеспечения автоматизированное средство поддержки циклических буферов данных, использующее аппаратные средства. Цель в том, чтобы удалить необходимость для программы выполнять адресацию данных проверки границ, когда выполняется плотно зацикленный код, что типично для многих DSP алгоритмов.

Адресация по модулю может работать в области данных или области программы (поскольку механизм указателя данных, по существу, одинаков для обоих). Один циклический буфер поддерживается в каждой X (что так же обеспечивает указатели на область программы) или Y области данных. Модульная адресация может работать на любом регистре указателе W. Тем не менее не желательно использование W14 или W15 для модульной адресации, поскольку эти два регистра используются как указатель фрейма стека и указатель стека, соответственно.

В общем, любой конкретный циклический буфер может быть только сконфигурирован для работы в одном направлении, как там есть определённые ограничения на стартовый адрес буфера (для инкремента буфера) или конечные адреса (для декремента буфера) основываясь на направление буфера.

Только исключение использования ограничений для буферов которые имеют power-of-2 длину. Как эти буферы удовлетворяют начальные и конечные адреса критерия, они могут действовать в двунаправленном режиме, (т.е. проверка границы адресов выполняется на обоих нижнем и верхнем адресах границы).

4.2.1 Начальные и конечные адреса

Схема модульной адресации требует чтобы начальный и конечный адреса были определены и загружены в 16-битные регистры модуля буфера адресов: XMODSRT, XMODEND, YMODSRT и YMODEND (смотреть таблицу 3-3).

Примечание:	Y область модульной адресации EA принимает данные размером в слово (LSb каждого EA всегда очищен).

Длина циклического буфера определена не напрямую. Это определено через различия между соответствующими начальными и конечными адресами. Максимально возможная длина циклического буфера 32К слова (64 Кбайт).

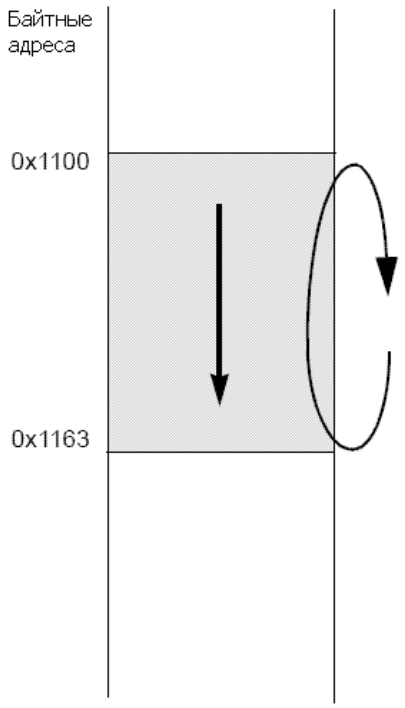
4.2.2 Выбор адресного регистра W

Регистр управления модульным и бит-реверсной адресацией MODCON<15:0> содержит флаги разрешения а так же область W регистра определяющую W регистры адресации. Области XWM и YWM выбираются как регистры обслуживающие модульную адресацию. Если XWM = 15, X RAGU и X WAGU модульная адресация блокирована. Аналогично, если YWM = 15, Y AGU модульная адресация блокирована.

W регистр указателя адресов X области (XWM) применяемый как модульная адресация, останавливается в MODCON<3:0> (смотреть таблицу 3-3). Модульная адресация разрешена для области данных X, когда XWM установлено любое значение за исключением 15 и бит XMODEN установлен MODCON<15>.

W регистр указателя адресов Y области (YWM) который приложен для модульной адресации, хранит MODCON<7:4>. Модульная адресация разрешена для области данных Y когда в YWM установлено любое значение кроме 15 и бит YMODEN установлен MODCON<14>.

Рисунок 4-1: Пример работы модульной адресации




```
MOV     #0x1100,W0
MOV     W0, XMODSRT     ;установка начального модульного адреса
MOV     #0x1163,W0
MOV     W0,MODEND       ;установка конечного модульного адреса
MOV     #0x8001,W0
MOV     W0,MODCON       ;разрешить W1, X AGU для модуля
MOV     #0x0000,W0      ;W0 хранит значение наполняющее буфер
MOV     #0x1110,W1      ;W1 указывает на буфер
DO      AGAIN,#0x31     ;заполнить 50 позиций буфера
MOV     W0, [W1++]      ;заполнить следующие позиции
AGAIN:  INC     W0,W0   ;инкремент заполняющего значения
```

4.2.3 Применение модульной адресации

Модульная адресация может применяться для эффективной адресации вычислений связанных с любым регистром W. При этом важно понимать, что адресные границы проверяются для адресов меньше чем или больше чем верхний (для инкрементируемых буферов) и нижний (для декрементируемых буферов) граничных адресов (не точно равные). Изменением адреса можно, следовательно, переходить за границы и всё же быть настроены правильно.

Примечание:	Модульный корректированный эффективный адрес записывается обратно в регистр только тогда, когда для вычисления эффективного адреса используется пре-модифицированный или пост-модифицированный режим адресации. Когда смещение адреса (т.е. [W7 + W2]) используется, выполняется коррекция модульных адресов, но содержимое регистра остаётся неизменным.

4.3 Бит-реверсная адресация

Бит-реверсная адресация предполагает упрощение переупорядочивания данных для алгоритмов radix-2 FFT. Эта адресация поддерживается только X AGU для записи данных.

Модификатор, который может быть постоянной величиной или содержимым регистра, считает как содержать эти биты в обратном порядке. Адреса источника и приёмника поддерживаются в нормальном порядке. Таким образом только операнд требует перестановки в модификаторе.

4.3.1 Реализация бит-реверсной адресации

Бит-реверсная адресация разрешена, когда:

1. BWM (выбран регистр W) в регистре MODCON имеет любое значение кроме 15 (при использовании бит-реверсной адресации стек может быть недоступным) **и**
2. В регистре XBREV установлен бит BREN **и**
3. Используется регистровый косвенный режим адресации с пре-инкрементом или пост-инкрементом.

Если длина бит-реверсного буфера составляет M = 2N байт, тогда следующие ‘N’битов стартового адреса буфера данных должны быть установлены в ноль.

XB<14:0> есть бит-реверсно модифицированный адрес или ‘точка вращения’ которая обычно постоянная. В случае вычисления FFT, это значение равно половине размера FFT буфера данных.

Примечание:	Все вычисленные бит-реверсные EA принимают данные размером в слово (LSb каждого EA всегда очищен). Значение XB соответственно масштабировано для генерации совместимых (байт) адресов.

Когда разрешено, бит-реверсная адресация может быть только для выполнения косвенной регистровой адресации с пре-инкрементом или пост-инкрементом и для записи данных размером в слово. Это может не функционировать для любых других режимов адресации или для данных байтного размера, и нормальная адреса могут генерироваться в замен. Когда бит-реверсная адресация активна, W адресный указатель всегда добавлен к модификатору адресов (XB) и смещение связанное с регистром режима косвенной адресации может игнорироваться. К тому же, так как требуются данные размером в слово, LSb эффективного адреса (EA) игнорируется (и всегда очищен).

Примечание:	Модульная адресация и бит-реверсная адресация не могут быть разрешены вместе. В случае, если пользователь пытается это сделать, бит-реверсная адресация имеет приоритет когда активен X WAGU, и X WAGU модульная адресация будет запрещена. Тем не менее модульная адресация может продолжать функционировать в X RAGU.

Если бит-реверсная адресация уже разрешена установкой бита BREN (XBREV<15>), тогда запись регистра XBREV не должна немедленно сопровождать операция косвенного чтения использующая регистр W, который определён как бит-реверсный указатель.

Рисунок 4-2: Пример бит-реверсной адресации

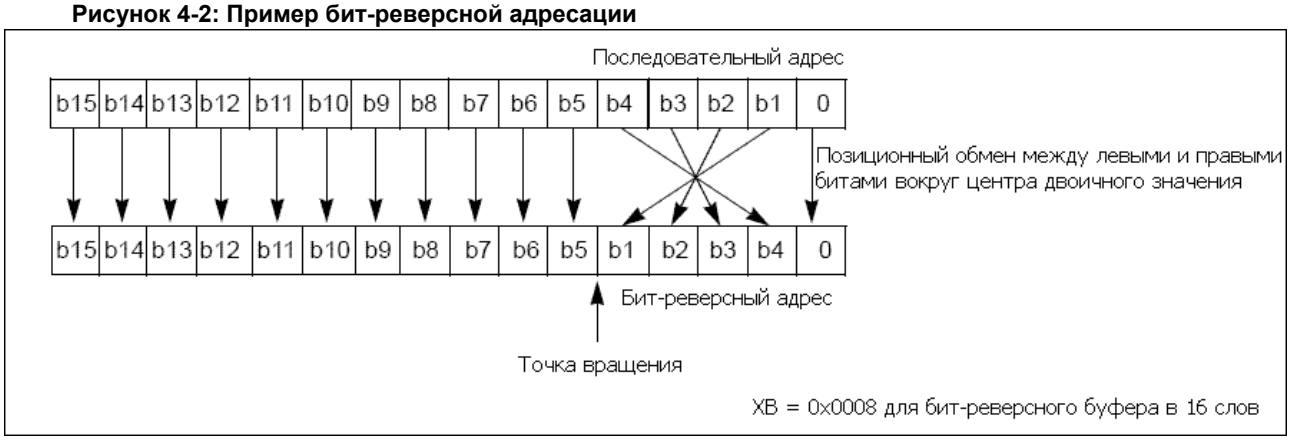


Таблица 4-2: Последовательность бит-реверсной адресации (16-элементная)

Нормальная адресация					Бит-реверсная адресация				
A3	**A2**	**A1**	**A0**	**Десятичное**	**A3**	**A2**	**A1**	**A0**	**Десятичное**
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	8
0	0	1	0	2	0	1	0	0	4
0	0	1	1	3	1	1	0	0	12
0	1	0	0	4	0	0	1	0	2
0	1	0	1	5	1	0	1	0	10
0	1	1	0	6	0	1	1	0	6
0	1	1	1	7	1	1	1	0	14
1	0	0	0	8	0	0	0	1	1
1	0	0	1	9	1	0	0	1	9
1	0	1	0	10	0	1	0	1	5
1	0	1	1	11	1	1	0	1	13
1	1	0	0	12	0	0	1	1	3
1	1	0	1	13	1	0	1	1	11
1	1	1	0	14	0	1	1	1	7
1	1	1	1	15	1	1	1	1	15

Таблица 4-3: Значения для XBREV регистра модифицированные бит-реверсной адресацией

Размер буфера (слов)	**XB<14:0> значение модифицированное бит-реверсной адресацией(1)**
32768	0x4000
16584	0x2000
8192	0x1000
4096	0x0800
2048	0x0400
1024	0x0200
512	0x0100
256	0x0080
128	0x0040
64	0x0020
32	0x0010
16	0x0008
8	0x0004
4	0x0002
2	0x0001

Примечание 1: Модифицируемые значения больше чем 256 слов превышают доступную память данных на устройстве dsPIC30F2010

5.0 Прерывания

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

dsPIC30F2010 имеет 24 источника прерывания по событию и 4 процессорных исключения(ловушки), арбитраж которых осуществляет схема приоритетов.

CPU ответственный за чтение таблицы векторов прерываний (IVT) и передачи адресов, содержащихся в векторе прерывания, в программный счётчик. Вектор прерывания передаётся из шины данных программы в программный счётчик, через мультиплексор шириной 24 бит на вход программного счётчика.

Таблица векторов прерывания (IVT) и таблица альтернативных векторов прерывания (AIVT) расположены близко к началу программной памяти (0x000004). IVT и AIVT показаны на рисунке 5-1.

Диспетчер прерываний является ответственным за обработку прерываний и процессорных исключений, до их передачи процессорному ядру. Периферийные прерывания и ловушки разрешены, для диспетчеризации и обслуживания приоритетов используются централизованные регистры специального назначения:

• IFS0<15:0>, IFS1<15:0>, IFS2<15:0>

Все флаги запроса прерывания содержатся в этих трёх регистрах. Флаги устанавливаются соответствующими периферийными и внешними сигналами, и очищаются программно.

• IEC0<15:0>, IEC1<15:0>, IEC2<15:0>

Управляющие биты разрешения всех прерываний содержатся в этих трёх регистрах. Эти биты управления используются для индивидуального разрешения прерываний от периферии или внешних сигналов.

• IPC0<15:0>... IPC11<7:0>

Назначаемые пользователем уровни приоритетов связанные с каждым из этих прерываний содержатся в этих двенадцати регистрах.

• IPL<3:0> Текущий уровень приоритета CPU явно загруженный в биты IPL. IPL<3> представлен в регистре CORCON, поскольку IPL<2:0> представлены в регистре STATUS(SR) процессорного ядра.

• INTCON1<15:0>, INTCON2<15:0>

Глобальное управление функциями прерывания производится с помощью этих двух регистров. INTCON1 содержит флаги управления и состояния для процессорных исключений. Регистр INTCON2 управляет поведением сигналов запроса внешних прерываний и используется альтернативной таблицей векторов.

Примечание:	Биты флага прерывания устанавливаются когда происходит условие прерывания, не обращая внимание на состояние соответствующих битов разрешения. Программа пользователя должна гарантировать очистку флагов прерывания до разрешения прерывания.

Если бит NSTDIS (INTCON1<15>) установлен, то не допускается вложенность прерываний. Таким образом, если в настоящее время обслуживается прерывание, то обработка нового прерывания не допускается, даже если новое прерывание имеет более высокий приоритет, чем обслуживаемое в настоящий момент.

Примечание:	Биты IPL становятся только для чтения всякий раз когда бит NSTDIS установлен в ‘1’.

Некоторые прерывания имеют специализированные биты управления для представления желаемого края или уровня запуска прерывания, прерывания на изменение. Управление этими характеристиками остаётся в пределах периферийного модуля, который генерирует прерывание.

Инструкцию DISI можно использовать для отключения обработки прерываний приоритетом 6 и ниже для определённого числа инструкций, в течении которых бит DISI (INTCON2<14>) остаётся установленным.

Когда обслуживается прерывание, в PC загружается адрес хранящийся в позиции вектора в программной памяти, который соответствует прерыванию. Там есть 63 различных вектора в пределах IVT (обратитесь к рисунку 5-1). Эти векторы содержатся в позициях программной памяти от 0x000004 до 0x0000FE (обратитесь к рисунку 5-1). Эти позиции содержат 24-битные адреса, и в порядке сохранения ошибкоустойчивости, будет происходить ловушка ошибки адресации, если произойдёт попытка выборки любого из этих слов в течении нормального выполнения программы. Это предохраняет выполнение произвольных данных, как результат случайного декремента PC на пространство векторов, случайного отображения пространства адресов данных на пространство векторов или прокручивания PC выше 0x000000 после достижения конца доступного пространства памяти программы. Выполнение инструкции GOTO в это векторное пространство может так же генерировать ловушку ошибки адресации.

5.1 Приоритет прерывания

Назначаемые пользователем биты (IP<2:0>) приоритета прерывания для каждого индивидуального источника прерывания расположены наименьших значимых 3 битах каждого нибла, в пределах регистра(ов) IPCx. Бит 3 каждого нибла не используется и читается как ‘0’. Эти биты определяют уровень приоритета, назначенный пользователем конкретному прерыванию.

Примечание: Выбираемые пользователем уровни приоритета от 0, как самый низкий приоритет, до уровня 7, как самый высокий приоритет.

Поскольку более чем одному источнику запроса прерывания может быть назначен определённый пользователем уровень приоритета, то предусмотрен способ назначения приоритета в пределах данного уровня. Этот метод называется "Естественный порядок приоритетов" и есть окончательный.

Естественный порядок приоритетов определён позицией прерывания в таблице векторов, и только влияет на управление прерыванием, когда многочисленные прерывания с тем же, определённым пользователем, приоритетом рассматриваются в то же время.

В таблице 5-1 приведён список номеров прерываний и источников прерываний для устройств dsPIC DSC и номеров векторов связанных с ними.

Примечание 1: Схема естественного порядка приоритетов имеет 0 как самый высокий приоритет и 53 как самый низкий приоритет.
2: Номер естественного порядка приоритета совпадает с номером INT.

Возможность пользователю присваивать каждому прерыванию один из семи уровней приоритета означает, что пользователь может назначить очень высокий уровень приоритета прерыванию с низким естественным порядком приоритета. Например, PLVD (обнаружение низкого напряжения) может быть дан приоритет 7. INT0 (внешнее прерывание 0) может быть назначен приоритет с уровнем 1, таким образом давая очень низкий эффективный приоритет.

Таблица 5-1: Таблица векторов прерывания dsPIC30F2010

Номер INT	Номер вектора	Источник прерывания
Самый высокий приоритет естественного порядка		
0	8	INT0 – Внешнее прерывание 0
1	9	IC1 – Вход захвата 1
2	10	OC1 — Выход сравнения 1
3	11	T1 – Таймер 1
4	12	IC2 - Вход захвата 2
5	13	OC2 — Выход сравнения 2
6	14	T2 – Таймер 2
7	15	T3 – Таймер 3
8	16	SPI1
9	17	U1RX – UART1 Приёмник
10	18	U1TX – UART1 Передатчик
11	19	ADC – ADC Преобразование сделано
12	20	NVM – NVM Запись завершена
13	21	SI2C – I2C™ Slave прерывание
14	22	MI2C – I2C Master прерывание
15	23	Прерывание изменения по входу
16	24	INT1 – Внешнее прерывание 1
17	25	IC7 – Вход захвата 7
18	26	IC8 – Вход захвата 8
19	27	Зарезервировано
20	28	Зарезервировано
21	29	Зарезервировано
22	30	Зарезервировано
23	31	INT2 — Внешнее прерывание 2
24	32	Зарезервировано
25	33	Зарезервировано
26	34	Зарезервировано
27	35	Зарезервировано
28	36	Зарезервировано
29	37	Зарезервировано
30	38	Зарезервировано
31	39	Зарезервировано
32	40	Зарезервировано
33	41	Зарезервировано
34	42	Зарезервировано
35	43	Зарезервировано
36	44	INT3 – Внешнее прерывание 3
37	45	Зарезервировано
38	46	Зарезервировано
39	47	PWM – Соответствие периода PWM
40	48	QEI – Прерывание QEI
41	49	Зарезервировано
42	50	Зарезервировано
43	51	FLTA – PWM дефект A
44	52	Зарезервировано
45-53	53-61	Зарезервировано
Самый низкий приоритет естественного порядка		

5.2 Последовательность сброса

Сброс не истинное исключение, потому что диспетчер прерываний не вовлечён в процесс сброса. Процессор инициализирует его регистры в ответ на сброс, который сбрасывает PC в ноль. Затем процессор начинает выполнение программы с позиции 0x000000. С помощью инструкции GOTO, записанной в первой позиции памяти программы, немедленно осуществляется переход по указанному адресу. Процессор выполняет GOTO и начинает работу с указанного стартового адреса.

5.2.1 Источники сброса

Вдобавок к внешнему сбросу и сбросу по включению питания (POR), существует 6 источников условий ошибок, которые 'ловят' вектор сброса.

• Сторожевой таймер:

Окончание счёта сторожевого таймера показывает, что процессор долго не выполняет корректный код.

• Ловушка неинициализированного регистра W:

Попытка использования не инициализированного регистра как адресного указателя вызовет сброс.

• Ловушка запрещённой инструкции:

Перехватывает попытки выполнения любого неиспользованного кода операции. Имейте в виду, что выборка запрещённой инструкции не приводит к ловушке, если эта инструкция сброшена перед выполнением благодаря изменению потока.

• Сброс по провалу питания (BOR):

Если обнаружено быстрое пропадание питания устройства, которое может привести к сбою.

• Ловушка блокировки:

Случай одновременного возникновения многочисленных условий перехвата(ловушек) вызовет сброс.

5.3 Ловушки

Ловушки могут рассматриваться как не маскируемые прерывания указывающие на программную или аппаратную ошибку, которые придерживаются определённых заранее приоритетов, как показано на рисунке 5-1. Они предполагают, что пользователь обеспечил средство коррекции ошибочной операции в течении отладки и когда работает в течении приложения.

Примечание:	Если пользователь не предполагает выбирать поправочное действие в случае ситуации ловушки, эти вектора должны быть загружены с адресами заданной по умолчанию ручки, которая просто содержать инструкцию RESET. Если с другой стороны, вызывается один из векторов содержащий недействительный адрес, генерируется ловушка ошибки адресации.

Имейте в виду, что много условий этих ловушек могут быть обнаруживаться только тогда, когда они происходят. Следовательно сомнительной инструкции будет позволено завершиться прежде обработки ловушки исключения. Если пользователь решает восстановиться после ошибки, результат ошибочных действий, которые вызвали ловушку, можно исправить.

Есть 8 фиксированных уровней приоритета для ловушек: От 8 до 15 уровня, которые означают что IPL3 всегда установлен в течении обработки ловушки.

Если пользователь в настоящее не выполняет ловушку, и он установил биты IPL<3:0> в значение '0111' (Level 7), тогда все прерывания выключены, но ловушки могут всё же обрабатываться.

5.3.1 Источники ловушки

Следующие ловушки предусмотрены с возрастанием приоритета. Тем не менее, поскольку все ловушки могут быть вложенными, приоритеты имеют не большой эффект.

Ловушка математической ошибки:

Ловушка математической ошибки выполняется по следующим четырём обстоятельствам:

1. Если была попытка деления на ноль, операция деления прерывается на границе цикла и происходит ловушка.

2. Если разрешено, ловушка математической ошибки будет происходить когда арифметическая операция на любом аккумуляторе A или B вызывает переполнение из бита 31 и биты защиты аккумулятора не использованы.

3. Если разрешено, ловушка математической ошибки будет происходить когда любом аккумуляторе A или B вызывается катастрофическое переполнение из бита 39 и все насыщения отключены.

4. Если объём сдвига определённый в инструкции сдвига превышает максимум допустимого объёма сдвига, ловушка происходит.

Ловушка ошибки адреса:

Эта ловушка инициируется когда любое из следующих обстоятельств происходит:

1. Попытка получить доступ к слову данных с нарушением границ.
2. Попытка выборки из незадействованных позиций памяти данных.
3. Попытка доступа к данным в незадействованных позициях памяти программ.
4. Попытка выборки команд из области векторов.

Примечание: В инструкциях класса MAC, в которых пространство памяти разделено на X и Y области, где незадействованная X область включает всю Y область, и незадействованная Y область включает всю X область.

5. Выполнение инструкции “BRA #literal” или инструкции “GOTO #literal”, где литерал является адресом в незадействованной области программы.

6. Выполнение инструкции в которой, после модификации PC, тот указывает на неиспользуемую область программ. PC можно модифицировать загрузкой значения в стек и выполнением инструкции RETURN.

Ловушка ошибки стека:

Эта ловушка инициализируется при следующих условиях:

1. Указатель стека загружен значением, которое больше чем (программируемый пользователем) предельное значение записанное в регистр SPLIM (переполнение стека).

2. Указатель стека загружен значением, которое меньше чем 0x0800 (простое опустошение стека).

Ловушка аварии генератора:

Эта ловушка инициируется, если внешний генератор терпит неудачу и работа поддерживается внутренним резервным RC-генератором.

5.3.2 Аппаратные и программные ловушки

Возможно, что многие ловушки могут стать активными в пределах одного и того же цикла (например, расположенное с нарушением границ слово стека записывается в переполненный адрес). В таком случае, осуществляется фиксированный приоритет, показанный на рисунке 5-1, который может требовать, чтобы пользователь, в порядке полного исправления дефекта, проверил, ожидают ли другие ловушки.

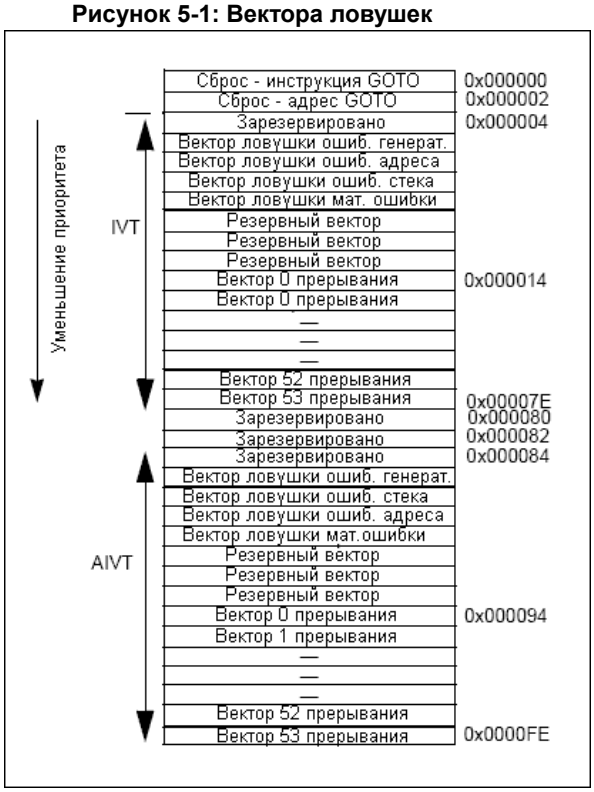
‘Программные’ ловушки включают исключения уровня приоритета от 8 до 11, включительно. Ловушка арифметической ошибки (уровень 11) начинает эту категорию ловушек.

‘Аппаратные’ ловушки включают исключения уровня приоритета от 12 до 15, включительно. Ловушки ошибки адреса (уровень 12), ошибки стека (уровень 13) и ошибки генератора (уровень 14) относятся к этой категории.

Каждая аппаратная ловушка, которая происходит, должна быть подтверждена прежде чем выполнения кода любого типа будет продолжено.

Если происходит аппаратная ловушка более низкого приоритета, пока ловушка с более высоким приоритетом рассматривается, подтверждается или обрабатывается, происходит конфликт аппаратных ловушек. Устройство автоматически сбрасывается в состоянии конфликта аппаратных ловушек. Бит состояния TRAPR (RCON<15>) установлен, когда происходит сброс, с тем чтобы это состояние могло быть обнаружено программным обеспечением.

Рисунок 5-1: Вектора ловушек



5.4 Последовательность прерывания

Все флажки событий прерывания выбраны в начале каждого цикла инструкции через регистры IFSx. Незаконченный запрос прерывания (IRQ) указывается через бит флага равный '1' в регистре IFSx. IRQ заставляет прерывание происходить, если установлен соответствующий бит в регистре разрешения прерывания (IECx). В оставшееся время цикла инструкции приоритеты всех рассматриваемых запросов прерываний

оцениваются.

Если рассматриваемые IRQ имеют уровень приоритета больше, чем приоритетный уровень текущего процесса в битах IPL, процессор будет прерван.

Затем процессор помещает в стек текущий программный счётчик и младший байт регистра STATUS процессора (SRL), как показано на рисунке 5-2. Младший байт регистра статуса содержит уровень приоритета процесса во время до начала цикла прерывания. Процессор затем загружает уровень приоритета для этого прерывания в регистр STATUS. Это действие блокирует все более низкие приоритеты прерывания, пока завершится программа обработки прерывания (ISR).

Рисунок 5-2: Стековый фрейм прерывания

Примечание 1:	Пользователь всегда может уменьшить уровень приоритета, записав новое значение в SR. Программа обработки прерывания должна очистить бит флага прерывания в регистре IFSx прежде уменьшения приоритета прерывания процессора, чтобы избежать рекурсивного прерывания.
2:	IPL3 бит (CORCON<3>) всегда очищен когда обрабатываются прерывания. Бывает установлен только в течении выполнения ловушки.

Инструкция RETFIE (Возврат из прерывания) извлекает из стека программный счётчик и регистр статуса, чтобы вернуть процессору его состояние до прерывания.

5.5 Альтернативная таблица векторов

В памяти программы, таблица векторов прерывания (IVT) сопровождается альтернативной таблицей векторов прерываний (AIVT), как показано на рисунке 5-1. Доступ к альтернативной таблице векторов обеспечивается через бит ALTIVT в регистре NTCON2. Если бит ALTIVT установлен, все прерывания и процессорные исключения используют альтернативные вектора вместо векторов по умолчанию. Альтернативные вектора организованы в той же манере, как и вектора по умолчанию. Обычно AIVT используется для эмуляции и отладки как мера обеспечения средства переключения между приложением и средой поддержки, без требования перепрограммирования векторов прерывания. Эта так же даёт возможность переключатся между приложением для вычислений различных программных алгоритмов во время работы. Если AIVT не требуется, память программы расположенная в AIVT может быть использована для других целей. AIVT является не защищённой частью и может быть свободно программировано пользователем.

5.6 Быстрое сохранение контекста

Опция сохранения контекста доступна при использовании теневых регистров. Теневые регистры доступны для битов DC, N, OV, Z и C в SR, и регистров от W0 до W3. Теневые имеют только один уровень погружения. Теневые регистры доступны только при помощи инструкций PUSH.S и POP.S.

Когда векторы процессора в прерывании, инструкцию PUSH.S можно использовать для сохранения текущего значения вышеупомянутых регистров в их соответствующих теневых регистрах.

Если ISR определённого приоритета использует PUSH.S и POP.S инструкции для быстрого сохранения контекста, затем более высокий приоритет ISR должен не включать такие же инструкции. Пользователь должен сохранить регистры ключа в программе в течении прерывания с более низким приоритетом, если ISR с более высоким приоритетом использует быстрое сохранение контекста.

5.7 Внешние запросы прерывания

Диспетчер прерываний поддерживает пять внешних сигналов запроса прерываний, INT0-INT4. Эти входы чувствительны на изменение сигнала; они требуют низкий-высокий или высокий-низкий переходов для генерации запроса прерывания. Регистр INTCON2 имеет три бита, INT0EP-INT2EP, которые выбирают полярность схемы обнаружения фронта.

5.8 Пробуждение из режимов Спячки и Простоя

Диспетчер прерывания может использоваться для пробуждения процессора из режимов Спячки или Простоя, если режимы Спячки или Простоя активны, когда генерируется прерывание. Если разрешён запрос прерывания достаточного приоритета полученного диспетчером прерываний, затем стандартный запрос прерывания будет представлен в процессор. Если разрешённый запрос прерывания достаточного приоритета получен диспетчером прерываний, затем стандартный запрос прерывания будет представлен процессору. В то же самое время процессор будет пробуждён из режимов Спячки и Простоя и начнёт обрабатывать программу обслуживания прерывания, необходимую для обработки запроса прерывания.

Таблица 5-2: Карта регистров диспетчера прерываний

SFR Имя	Адрес	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
INTCON1	0080	NSTDIS	-	-	-	-	OVATE	OVBTE	COVTE	-	-	-	MATHERR	ADDRERR	STKERR	OSCFAIL	-	0000 0000 0000 0000
INTCON2	0082	ALTIVT	DISI	-	-	-	-	-	-	-	-	-	-	-	INT2EP	INT1EP	INT0EP	0000 0000 0000 0000
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0IF	0000 0000 0000 0000
IFS1	0086	-	-	-	-	-	-	-	-	INT2IF	-	-	-	-	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IFS2	0088	-	-	-	-	FLTAIF	-	-	QEIIF	-	-	-	-	-	-	-	-	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	-	-	-	-	-	-	-	-	INT2IE	-	-	-	-	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IEC2	0090	-	-	-	-	FLTAIE	-	-	QEIIE	PWMIE	-	-	-	-	-	-	-	0000 0000 0000 0000
IPC0	0094	-	T1IP<2:0>			-	OC1IP<2:0>			-	IC1IP<2:0>			-	INT0IP<2:0>			0100 0100 0100 0100
IPC1	0096	-	T31P<2:0>			-	T2IP<2:0>			-	OC2IP<2:0>			-	IC2IP<2:0>			0100 0100 0100 0100
IPC2	0098	-	ADIP<2:0>			-	U1TXIP<2:0>			-	U1RXIP<2:0>			-	SPI1IP<2:0>			0100 0100 0100 0100
IPC3	009A	-	CNIP<2:0>			-	MI2CIP<2:0>			-	SI2CIP<2:0>			-	NVMIP<2:0>			0100 0100 0100 0100
IPC4	009C	-	-	-	-	-	IC8IP<2:0>			-	IC7IP<2:0>			-	INT1IP<2:0>			0100 0100 0100 0100
IPC5	009E	-	INT2IP<2:0>			-	-	-	-	-	-	-	-	-	-	-	-	0100 0000 0000 0000
IPC6	00A0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0000 0000 0000 0000
IPC7	00A2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0000 1000 0000 0000
IPC8	00A4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0000 0000 0000 0000
IPC9	00A6	-	PWMIP<2:0>			-	-	-	-	-	-	-	-	-	-	-	-	0000 0000 0000 0000
IPC10	00A8	-	FLTAIP<2:0>			-	-	-	-	-	-	-	-	-	QEIIP<2:0>			0100 0000 0000 0100
IPC11	00AA	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0000 0000 0000 0000

Описание: u = неинициализированный бит

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

6.0 Flash память программы

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

Семейство устройств содержат внутреннюю Flash память программ для хранения исполняемого кода пользователя. Существует два метода, которые пользователь может использовать для программирования этой памяти:

1. Внутрисхемное последовательное программирование (ICSP)
2. Самопрограммирование во время выполнения программы (RTSP)

6.1 Внутрисхемное последовательное программирование (ICSP)

Устройства dsPIC30F могут периодически программироваться непосредственно в схеме. Для этого достаточно двух линий для данных программирования и синхронизации (которые соответственно называются PGD и PGC), и три других линии - питание (VDD), общий провод (VSS) и мастер очистки (MCLR). Это позволяет потребителю производить плату с незапрограммированными устройствами, и затем программировать их перед самим использованием. Это так же позволяет обновлять программное обеспечение и использовать фирменное программное обеспечение.

6.2 Самопрограммирование во время выполнения программы (RTSP)

RTSP выполняется с использованием инструкций TBLRD (табличное чтение) TBLWT (табличную запись). С RTSP, пользователь может стереть память программ, 32 инструкции (96 байт) за раз и может записать данные в память программ, 32 инструкции (96 байт) за раз.

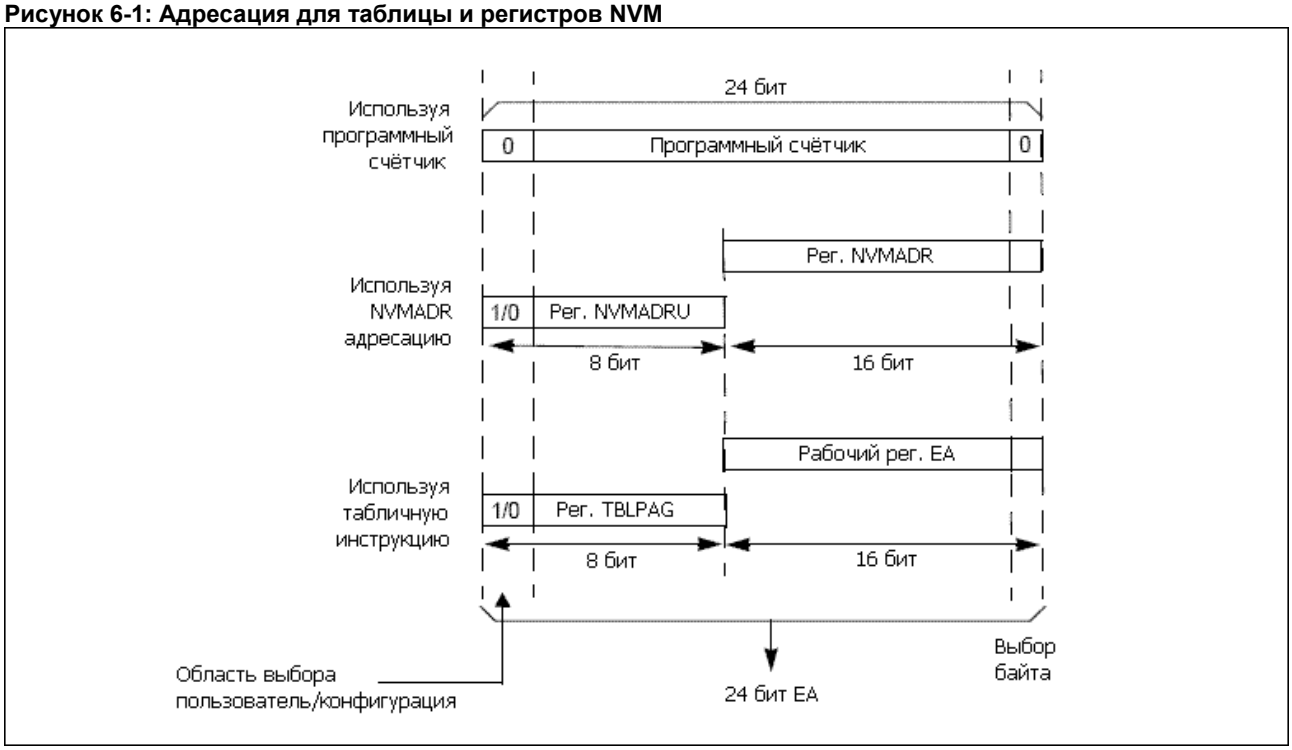
6.3 Обзор действия табличных инструкций

Инструкции TBLRDL и TBLWTL используются для чтения или записи битов<15:0> программной памяти. TBLRDL и TBLWTL позволяют доступ к программной памяти в байтном или словном режимах.

Инструкции TBLRDH и TBLWTH используются для чтения и записи битов<23:16> программной памяти. TBLRDH и TBLWTH позволяют доступ к программной памяти в байтном и словном режимах.

24-битные адреса программной памяти формируются с использованием битов<7:0> регистра TBLPAG и EA из регистра W определённого в табличной инструкции, как показано на рисунке 6-1.

Рисунок 6-1: Адресация для таблицы и регистров NVM



6.4 Операция RTSP

Flash память программ dsPIC30F организована на строки и панели. Каждая строка состоит из 32 инструкций, или 96 байтов. Каждая панель состоит из 128 строк, или 4K x 24 инструкций. RTSP позволяет стереть пользователю одну строку (32 инструкции) за раз и программирования 32 инструкций одновременно. RTSP можно использовать для многократного программирования панелей программной памяти, но табличный указатель должен меняться на границе каждой панели.

Каждая панель программной памяти содержит защёлку записи, которая удерживает 32 инструкции программных данных. До фактической операции программирования, записываемые данные должны быть загружены на защёлки записи панели. Данные программированные на панель загружены в последовательном порядке на защёлки записи; инструкция 0, инструкция 1 и т.д. Загруженные слова инструкций всегда должны быть из границы 32 адресов

Базовая последовательность для RTSP программирования вызывает таблицу указателей, затем делает последовательность TBLWT инструкций загрузки защёлок записи. Программирование выполняется установкой специальных битов в регистре NVMCON. 32 TBLWTL и четыре TBLWTH инструкции необходимо для загрузки 32 инструкций. Если требуется многократное программирование панели, таблицу указателей нужно изменить и следующую установку многократной записи защёлок записать.

Все операции записи таблицы являются записями отдельного слова (2 цикла инструкции), потому что только таблица защёлок записана. Программный цикл необходим для программирования каждой строки.

Flash память программы читаема, записываема выше целого диапазона VDD.

6.5 Управляющие регистры

Для чтения и записи Flash памяти программ используются четыре SFR регистра:

- NVMCON
- NVMADR
- NVMADRU
- NVMKEY

6.5.1 Регистр NVMCON

С помощью регистра NVMCON выбирают какие блоки стираются, какой тип памяти программируется и старт программного цикла.

6.5.2 Регистр NVMADR

Регистр NVMADR используется для хранения нижних двух байтов эффективного адреса. Регистр NVMADR захватывает EA<15:0> последней табличной инструкции, которая была выполнена и выбрала строку записи.

6.5.3 Регистр NVMADRU

Регистр NVMADRU используется для хранения верхнего байта эффективного адреса. Регистр NVMADRU захватывает EA<23:16> последней табличной инструкции, которая была выполнена.

6.5.4 Регистр NVMKEY

Регистр NVMKEY только для записи и используется для записи защиты. В начале программирования или последовательности стирания, пользователь должен последовательно записать 0x55 и 0xAA в регистр NVMKEY. Обратитесь к части 6.6 “Процесс программирования” для больших подробностей.

Примечание:	Пользователь также может непосредственно записывать регистры NVMADR и NVMADRU определяя адрес памяти программ для стирания и программирования.

6.6 Операции программирования

Полная последовательность программирования необходима для программирования и стирания внутренней Flash в режиме RTSP. Длительность операции программирования номинально составляет 2 msec и процессор останавливается (ожидает) пока операция не завершится. Установка бита WR (NVMCON<15>) стартует операцию и бит WR автоматически очищается, когда операция завершается.

6.6.1 Алгоритм программирования FLASH

Пользователь может стереть и программировать одну строку Flash памяти программ за раз. Общий процесс есть:

1. Читать одну строку Flash памяти программ (32 слова инструкций) и запомнить её в RAM данных как “изображение” данных.
2. Дополнить изображение данных желаемыми новыми данными.
3. Стереть строку Flash памяти программы.

a) Установить регистр NVMCON для multi-word, программировать Flash, стерев и установив бит WREN.
b) Адрес записи строки на NVMADRU/NVMDR будет стёрт.
c) Записать ‘55’ в NVMKEY.
d) Записать ‘AA’ в NVMKEY.
e) Установить бит WR. Это начнёт цикл стирания.
f) CPU останавливается на время цикла стирания.
g) Бит WR очищается, когда цикл стирания завершается.

4. Записать 32 слова инструкций из "изображения" RAM данных на защёлки записи Flash.
5. Программировать 32 слова инструкций на Flash программ.

a) Установить регистр NVMCON для multi-word, программировать Flash, стерев и установив бит WREN.
b) Записать ‘55’ в NVMKEY.
c) Записать ‘AA’ в NVMKEY.
d) Установить бит WR. Это начнёт цикл программирования.
e) CPU останавливается на время цикла программирования.
f) Бит WR очищается, когда цикл программирования завершается.

6. Повторять шаги с 1 по 5 пока необходимо записывать желаемый объём Flash памяти программы.

6.6.2 Стирание строки памяти программы

Пример 6-1 показывает кодовую последовательность, которая может быть использована для стирания строки (32 инструкции) памяти программы.

Пример 6-1: Стирание строки памяти программы

```
; Установка NVMCON для операции стирания, многократной записи слова
; выбор памяти программы и разрешение записи
         MOV     #0x4041,W0                ;
         MOV     W0,NVMCON                 ; Init NVMCON SFR
; Установка указателя на стираемую строку
         MOV     #tblpage(PROG_ADDR),W0    ;
         MOV     W0,NVMADRU                ; Инициализировать PM страницы границу SFR
         MOV     #tbloffset(PROG_ADDR),W0  ; Инициализировать указатель на страницу EA[15:0]
         MOV     W0, NVMADR                ; Инициализировать NVMADR SFR
         DISI    #5                        ; Блокировать все прерывания с приоритетом <7
                                           ; для следующих 5-ти инструкций
         MOV     #0x55,W0                  ;
         MOV     W0,NVMKEY                 ; Записать ключ 0x55
         MOV     #0xAA,W1                  ;
         MOV     W1,NVMKEY                 ; Записать ключ 0xAA
         BSET    NVMCON,#WR                ; Начать последовательность стирания
         NOP                               ; Вставить две инструкции NOP после команды
         NOP                               ; стирания
```

6.6.3 Загрузка защёлок записи

Пример 6-2 показывает последовательность инструкций которая может быть использована для загрузки 96 байт защёлок записи. 32 TBLWTL и 32 TBLWTH инструкции необходимо загрузить в защёлки записи выбранные табличным указателем.

Пример 6-2: Загрузка защёлок записи

```
; Установить указатель на первую записываемую ячейку памяти программы
; память программы выбрана и запись разрешена
          MOV       #0x0000,W0          ;
          MOV       W0,TBLPAG           ; Инициализировать страницу PM граница SFR
          MOV       #0x6000,W0          ; Пример адреса памяти программы
; Выполнить инструкции TBLWT записи защёлок
; 0-е программное слово
          MOV       #LOW_WORD_0,W2      ;
          MOV       #HIGH_BYTE_0,W3     ;
          TBLWTL    W2,[W0]             ; Запись младшего слова PM в программную защёлку
          TBLWTH    W3,[W0++]           ; Запись старшего байта PM в программную защёлку
; 1-е программное слово
          MOV       #LOW_WORD_1,W2      ;
          MOV       #HIGH_BYTE_1,W3     ;
          TBLWTL    W2,[W0]             ; Запись младшего слова PM в программную защёлку
          TBLWTH    W3,[W0++]           ; Запись старшего байта PM в программную защёлку
; 2-е программное слово
          MOV       #LOW_WORD_2,W2      ;
          MOV       #HIGH_BYTE_2,W3     ;
          TBLWTL    W2,[W0]             ; Запись младшего слова PM в программную защёлку
          TBLWTH    W3,[W0++]           ; Запись старшего байта PM в программную защёлку
          .
          .
          .
; 31-е программное слово
          MOV       #LOW_WORD_31,W2     ;
          MOV       #HIGH_BYTE_31,W3    ;
          TBLWTL    W2,[W0]             ; Запись младшего слова PM в программную защёлку
          TBLWTH    W3,[W0++]           ; Запись старшего байта PM в программную защёлку
```

Примечание: В примере 6-2, содержание верхнего байта W3 не имеет значения.

6.6.4 Инициализация последовательности программирования

Для защиты, инициализация последовательности записи для NVMKEY должна быть использована чтобы позволить продолжение любого стирания или операцию программирования. После того, как команда программирования была выполнена, пользователь должен ждать время программирования пока программирование не будет завершено. Две инструкции следующие за начальной последовательностью программирования должны быть NOP.

Пример 6-3: Инициализация последовательности программирования

```
          DISI      #5                  ; Блокировать все прерывания с приоритетом <7
                                        ; для следующих 5 инструкций
          MOV       #0x55,W0
          MOV       W0,NVMKEY           ; Записать ключ 0x55
          MOV       #0xAA,W1            ;
          MOV       W1,NVMKEY           ; Записать ключ 0xAA
          BSET      NVMCON,#WR          ; Начать последовательность стирания
          NOP                           ; Вставить две команды NOP после стирания
          NOP                           ;
```

Таблица 6-1: Карта регистров NVM

Имя регистра	Адрес	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
NVMCON	0760	WR	WREN	WRERR	-	-	-	-	TWRI	-	PROGOP<6:0>							0000 0000 0000 0000
NVMADR	0762	NVMADR<15:0>																uuuu uuuu uuuu uuuu
NVMADRU	0764	-	-	-	-	-	-	-	-	NVMADR<23:16>								0000 0000 uuuu uuuu
NVMKEY	0766	-	-	-	-	-	-	-	-	KEY<7:0>								0000 0000 0000 0000

Описание: u = неинициализированный бит

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

7.0 EEPROM память данных

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

EEPROM память данных читается и записывается с помощью нормальных операций выше целого VDD диапазона. EEPROM память данных непосредственно отображается в пространство адресов памяти программы.

Четыре SFR, используемые для чтения и записи Flash памяти программы, так же используются для доступа к EEPROM памяти данных. Как описано в части 4.0, эти регистры есть:

- NVMCON
- NVMADR
- NVMADRU
- NVMKEY

EEPROM память данных допускает чтение и запись одного слова и 16-словных блоков. Когда требуется связь с памятью данных, регистр NVMADR в коньюкции с регистром NVMADRU используется для адресации доступной ячейки EEPROM. Инструкции TBLRDL и TBLWTL используются для чтения и записи данных EEPROM. Устройства dsPIC30F имеют до 1 кбайта данных EEPROM, с адресным диапазоном от 0x7FFC00 до 0x7FFFFE.

Операции записи слова должна предшествовать стирание соответствующей ячейки или ячеек памяти. Для завершения записи обычно требуется 2 ms, но время записи может изменяться в зависимости от напряжения и температуры.

Операция программирование или стирания данных EEPROM не останавливает потока инструкций. Пользователь отвечает за соответствующую длительность времени перед инициализацией другой операции записи/стирания EEPROM. Попытка чтения данных EEPROM во время операции стирания или программирования приведёт к получению неопределённых данных.

Бит управления WR инициализирует операцию записи, аналогично записи в Flash память программы. Этот бит не может быть очищен программно, только установлен. Этот бит очищается аппаратно после завершения операции записи. Невозможность очистки бита WR программно предотвращает случайное или преждевременное завершение операции записи. При включении питания бит WREN очищается. Бит WRERR устанавливается, когда операция записи прервана сбросом по входу MCLR, или сбросом по окончанию счёта WDT, в течении нормальной работы. В этих ситуациях, после сброса пользователь может проверить бит WRERR и перезаписать ячейку. Регистр адреса NVMADR остаётся неизменным.

Примечание:	Бит флага прерывания NVMIF в регистре IFS0 установлен, когда запись завершена. Он должен очищаться программно.

7.1 Чтение памяти данных EEPROM

Инструкция TBLRD читает слово по адресу текущего слова программы. Этот пример использует W0 как указатель на данные EEPROM. Результат будет помещён в регистр W4, как показано в примере 7-1.

Пример 7-1: Чтение данных EEPROM

```
MOV        #LOW_ADDR_WORD,W0    ; Инициализировать указатель
MOV        #HIGH_ADDR_WORD,W1
MOV        W1,TBLPAG
TBLRDL     [ W0 ], W4           ;Чтение данных EEPROM
```

7.2 Стирание данных EEPROM

7.2.1 Стирание блока данных EEPROM

В порядке стирания блока данных EEPROM, регистры NVMADRU и NVMADR должны первоначально указывать на стираемый блок памяти. Сконфигурировать NVMCON для стирания блока данных EEPROM, и установить биты WR и WREN в регистре NVMCON. Установка бита WR инициирует стирание, как показано в примере 7-2.

Пример 7-2: Стирание блока данных EEPROM

```
; Выбор блока данных EEPROM, ERASE, WREN биты
          MOV     #4045,W0            ;
          MOV     W0,NVMCON           ; Инициализация NVMCON SFR
; Старт цикла стирания через установку WR, после записи последовательности ключей
          DISI    #5                  ; Блокировать все прерывания с приоритетом <7
                                      ; для следующих 5-ти инструкций
          MOV     #0x55,W0            ;
          MOV     W0,NVMKEY           ; Записать ключ 0x55
          MOV     #0xAA,W1            ;
          MOV     W1,NVMKEY           ; Записать ключ 0xAA
          BSET    NVMCON,#WR          ; Инициировать последовательность стирания
          NOP
          NOP
; Цикл стирания завершается за 2mS. CPU не останавливается для выполнения цикла стирания данных
; Пользователь может опрашивать бит WR, использовать NVMIF или Timer IRQ для определения завершения
; стирания
```

7.2.2 Стирание слова данных EEPROM

Регистры NVMADRU и NVMADR должны указывать на блок. Выбрать стираемый блок данных Flash, и установить биты WR и WREN в регистре NVMCON. Установка бит WR инициирует стирание, как показано в примере 7-3.

Пример 7-3: Стирание слова данных EEPROM

```
; Выбор слова данных EEPROM, ERASE, WREN биты
          MOV     #4044,W0
          MOV     W0,NVMCON
; Старт цикла стирания через установку WR, после записи последовательности ключей
          DISI    #5                  ; Блокировать все прерывания с приоритетом <7
                                      ; для следующих 5-ти инструкций
          MOV     #0x55,W0            ;
          MOV     W0,NVMKEY           ; Записать ключ 0x55
          MOV     #0xAA,W1            ;
          MOV     W1,NVMKEY           ; Записать ключ 0xAA
          BSET    NVMCON,#WR          ; Инициировать последовательность стирания
          NOP
          NOP
; Цикл стирания завершается за 2mS. CPU не останавливается для выполнения цикла стирания данных
; Пользователь может опрашивать бит WR, использовать NVMIF или Timer IRQ для определения завершения
; стирания
```

7.3 Запись в EEPROM данных

Для записи в ячейку данных EEPROM, должна следовать следующая последовательность:

1. Стереть слово данных EEPROM.
a) Выбрать слово данных EEPROM, стереть и установить бит WREN в регистре NVMCON.
b) Записать в NVMADRU/NVMADR адрес стираемого слова.
c) Разрешить прерывание NVM (опционально).
d) Записать ‘55’ в NVMKEY.
e) Записать ‘AA’ в NVMKEY.
f) Установить бит WR. Это начнёт цикл стирания.
g) Опрашивать бит NVMIF или ждать прерывания для NVMIF.
h) Бит WR очищается, когда цикл стирания окончен.
2. Записать слово данных в защёлки записи EEPROM данных.
3. Программировать 1 слово данных в EEPROM данных.
a) Выбрать слово, EEPROM данных, программировать и установить бит WREN в регистре NVMCON.
b) Разрешить NVM прерывание окончания записи (опционально).
c) Записать ‘55’ в NVMKEY.
d) Записать ‘AA’ в NVMKEY.
e) Установить бит WR. Это начнёт цикл программирования.
f) Опрашивать бит NVMIF или ждать прерывания NVM.
g) Бит WR очищается, когда цикл записи закончен.

Запись не инициируется, если точно не выполнить вышеуказанную последовательность (записать 0x55 в NVMKEY, записать 0xAA в

NVMCON, затем установить бит WR) для каждого слова. Так же строго рекомендуется чтобы прерывания были запрещены в течении этого кодового сегмента.

К тому же, бит WREN в NVMCON должен быть установлен для разрешения записи. Этот механизм предотвращает случайную запись данных EEPROM связанную с выполнением случайного кода. Бит WREN необходимо держать очищенным всё время, кроме когда обновляется EEPROM. Бит WREN не очищается аппаратно.

После того как последовательность записи инициирована, очистка бита WREN не оказывает эффекта на текущий цикл записи. Установка бита WR запрещена, если бит WREN не установлен. Бит WREN должен быть установлен предшествующей инструкцией. Оба бита WR и WREN не могут быть установлены одной и той же инструкцией.

При завершении цикла записи, бит WR будет очищен аппаратно и установлен бит флага (NVMIF) прерывания завершения записи энергонезависимой памяти. Пользователь может использовать это прерывание или опрашивать бит. Бит NVMIF сбрасывается программно.

7.3.1 Запись слова в EEPROM данных

После того как пользователь стёр слово, которое будет запрограммировано, далее используется инструкция табличной записи для записи защёлки записи, как показано в примере 7-4.

Пример 7-4: Запись слова данных EEPROM

```
; указатель на память данных
          MOV     #LOW_ADDR_WORD,W0    ; инициация указателя
          MOV     #HIGH_ADDR_WORD,W1
          MOV     W1,TBLPAG
          MOV     #LOW(WORD),W2        ; Получить данные
          TBLWTL  W2,[ W0]             ; Записать данные
; NVMADR захватывает последний адрес табличного доступа
; Выбрать EEPROM данных для 1 слова
          MOV     #0x4004,W0
          MOV      W0,NVMCON

; Ввести ключи для допуска к операции чтения
          DISI    #5                   ; Блокировать все прерывания с приоритетом <7
                                       ; для следующих 5-ти инструкций
          MOV     #0x55,W0             ;
          MOV     W0,NVMKEY            ; Записать ключ 0x55
          MOV     #0xAA,W1             ;
          MOV     W1,NVMKEY            ; Записать ключ 0xAA
          BSET    NVMCON,#WR           ; Инициировать последовательность программирования
          NOP
          NOP
; Цикл записи завершается за 2mS. CPU не останавливается во время цикла записи данных
; Пользователь может опрашивать бит WR, использовать NVMIF или Timer IRQ для определения завершения
; записи
```

7.3.2 Запись блока данных EEPROM

Для записи блока данных EEPROM, надо сначала записать все шестнадцать защёлок, затем установить регистр NVMCON и программировать блок.

Пример 7-5: Запись блока в EEPROM данных

```
MOV       #LOW_ADDR_WORD,W0        ; Инициировать указатель
MOV       #HIGH_ADDR_WORD,W1
MOV       W1,TBLPAG
MOV       #data1,W2                ; Получить 1-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data2,W2                ; Получить 2-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data3,W2                ; Получить 3-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data4,W2                ; Получить 4-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data5,W2                ; Получить 5-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data6,W2                ; Получить 6-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data7,W2                ; Получить 7-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data8,W2                ; Получить 8-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data9,W2                ; Получить 9-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data10,W2               ; Получить 10-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data11,W2               ; Получить 11-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data12,W2               ; Получить 12-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data13,W2               ; Получить 13-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data14,W2               ; Получить 14-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data15,W2               ; Получить 15-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные
MOV       #data16,W2               ; Получить 16-ые данные
TBLWTL    W2,[ W0]++               ; Записать данные. NVMADR захватывает последний адрес
                                   ; табличного доступа.
MOV       #0x400A,W0               ; Выбрать данные EEPROM для многословной операции
MOV       W0,NVMCON                ; Обслужить ключи для допуска к операции программирования
DISI      #5                       ; Блокировать все прерывания с приоритетом <7
                                   ; для следующих 5 инструкций
MOV       #0x55,W0                 ;
MOV       W0,NVMKEY                ; Записать ключ 0x55
MOV       #0xAA,W1                 ;
MOV       W1,NVMKEY                ; Записать ключ 0xAA
BSET      NVMCON,#WR               ; Начать цикл записи
NOP
NOP
```

7.4 Проверка записи

В зависимости от прикладной программы, хорошая практика программирования может требовать, чтобы значения данных, записанных в память, надо было сравнивать с оригинальными значениями. Это должно использоваться в прикладных программах, где чрезмерное количество записей может подвергать биты близко к определённым пределам.

7.5 Защита против случайной записи

Могут сложится условия, когда устройство не нуждается в записи данных в EEPROM память. В устройство встроены различные механизмы, запрещающие случайную запись EEPROM. При включении, бит WREN очищен; также таймер включения предотвращает запись EEPROM.

Инициация последовательности записи и бит WREN вместе помогают предотвратить случайную запись в течении кратковременных провалов питания, глюков питания и программных сбоев.

8.0 Порты ввода/вывода

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

Все ножки устройства (за исключением VDD, VSS, MCLR и OSC1/CLKI) распределены между периферией и параллельными портами ввода/вывода.

Все порты ввода/вывода имеют входы с триггером Шмидта для улучшения помехоустойчивости.

8.1 Параллельные порты ввода/вывода (PIO)

Когда периферия разрешена и активно использует связанную с ней ножку, использование ножки как обычной ножки вывода запрещено. Ножка ввода/вывода может быть прочитана, но выходной драйвер для бита параллельного порта отключен. Если периферия включена, но периферия не использует активно ножку, то ножка может быть использована как порт.

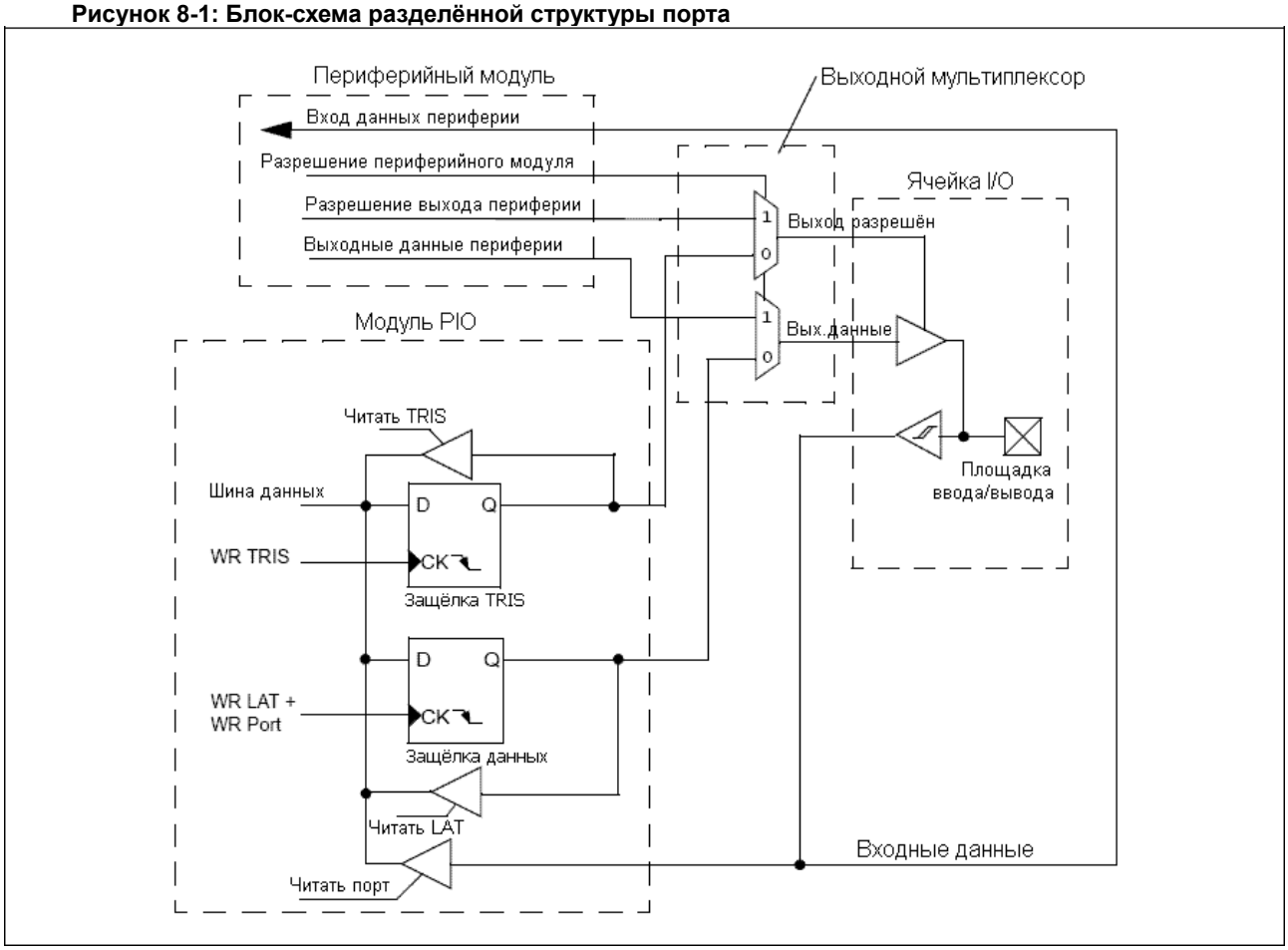
Все ножки порта имеют три регистра напрямую связанные с работой ножки порта. Регистр направления данных (TRISx) определяет для каждой ножки в каком режиме, ввода или вывода, она будет использоваться. Если бит направления данных равен ‘1’, то ножка используется на ввод. Все ножки порта определены как вводы после сброса. Чтение из защёлки (LATx), читает защёлку. Запись в защёлку, записывает защёлку (LATx).

Любой бит и связанные с ним данные и регистры контроля, которые не правильные для конкретного устройства, могут быть заблокированы. Это означает, что соответствующие LATx и TRISx регистры и ножки порта будут читаться как нули.

Когда ножка распределена с другой периферией или функцией которая определена как только ввод, это тем не менее считается как преданный порт, потому что там нет другого конкурирующего источника выходов. В качестве примера рассмотрим INT4 ножку.

Параллельный порт ввода/вывода (PIO), который делит ножку с периферией, в основном, подчинён периферии. Выходные буферы данных периферии и сигналы управления обеспечены спаривание мультиплексоров. Мультиплексоры выбирают, имеют ли периферийное устройство или связанный порт монопольное использование выходных данных и сигналов управления контактной площадки ввода/вывода. На рисунке 8-1 показано, как порты разделены с другими внешними устройствами, и соответствующей ячейкой ввода/вывода с который они связаны. Таблица 8-1 показы форматы регистраторов для общедоступных портов, от PORTB до PORTF.

Рисунок 8-1: Блок-схема разделённой структуры порта



8.2 Конфигурация ножек аналогового порта

Используйте регистры ADPCFG и TRIS для управления работой ножек A/D порта. Для ножек порта, которые нужно сделать аналоговыми входами, нужно установить соответствующий бит TRIS (ввод). Если бит TRIS очищен (выход), то ножка превращается в выход цифровых уровней (VOH или VOL).

При попытке чтения регистра порта, все ножки которого конфигурированы как аналоговые входные каналы, будет читаться низкий логический уровень. Ножки, сконфигурированные как цифровые входы, не преобразуют аналоговый вход. Аналоговые уровни на любой ножке, которая определена как цифровой вход (включая ANx ножки), могут заставить входные буферы потреблять ток, который превышает допустимые для устройства.

8.2.1 Синхронизация записи/чтения порта ввода/вывода

Один цикл инструкции необходим между изменениями направления передачи порта или между операциями записи и чтения того же самого порта. Обычно для создания этой паузы используют инструкцию NOP.

Пример 8-1: Пример записи/чтения порта

```
MOV       0xFF00, W0          ; Конфигурируем ножки PORTB<15:8> как входы
MOV       W0, TRISBB          ; и ножки PORTB<7:0> как выходы
NOP                           ; 1 цикл задержки
BTSS      btssPORTB, #13      ; Следующая инструкция
```

8.3 Модуль регистрации изменения входа

Модуль регистрации изменения входа обеспечивает устройствам dsPIC30F способность генерировать запрос прерывания процессора в ответ на изменение состояния на выбранных входных ножках. Этот модуль способен обнаруживать изменение состояния входа даже в режиме Спячки, когда отключено тактирование. Существует до 22 внешних сигналов (от CN0 до CN21) которые могут быть выбраны (разрешены) для генерации запроса прерывания на изменение состояния.

Таблица 8-1: Карта регистра порта dsPIC30F2010

SFR Имя	Адрес	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
TRISB	02C6	-	-	-	-	-	-	-	-	-	-	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	0000 0000 0011 1111
PORTB	02C8	-	-	-	-	-	-	-	-	-	-	RB5	RB4	RB3	RB2	RB1	RB0	0000 0000 0000 0000
LATB	02CA	-	-	-	-	-	-	-	-	-	-	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	0000 0000 0000 0000
TRISC	02CC	TRISC15	TRISC14	TRISC13	-	-	-	-	-	-	-	-	-	-	-	-	-	1110 0000 0000 0000
PORTC	02CE	RC15	RC14	RC13	-	-	-	-	-	-	-	-	-	-	-	-	-	0000 0000 0000 0000
LATC	02D0	LATC15	LATC14	LATC13	-	-	-	-	-	-	-	-	-	-	-	-	-	0000 0000 0000 0000
TRISD	02D2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TRISD1	TRISD0	0000 0000 0000 0111
PORTD	02D4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RD1	RD0	0000 0000 0000 0000
LATD	02D6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	LATD1	LATD0	0000 0000 0000 0000
TRISE	02D8	-	-	-	-	-	-	-	TRISE8	-	-	TRISE5	TRISE4	TRISE3	TRISE2	TRISE1	TRISE0	0000 0001 0011 1111
PORTE	02DA	-	-	-	-	-	-	-	RE8	-	-	RE5	RE4	RE3	RE2	RE1	RE0	0000 0000 0000 0000
LATE	02DC	-	-	-	-	-	-	-	LATE8	-	-	LATE5	LATE4	LATE3	LATE2	LATE1	LATE0	0000 0000 0000 0000
TRISF	02DE	-	-	-	-	-	-	-	-	-	-	-	-	TRISF3	TRISF2	-	-	0000 0000 0000 1100
PORTF	02E0	-	-	-	-	-	-	-	-	-	-	-	-	RF3	RF2	-	-	0000 0000 0000 0000
LATF	02E2	-	-	-	-	-	-	-	-	-	-	-	-	LATF3	LATF2	-	-	0000 0000 0000 0000

Описание: u = неинициализированный бит

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

Таблица 8-2: Карта регистра сообщений о изменениях входов (BITS 15-0)

SFR Имя	Адрес	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
CNEN1	00C0	CN15IE	CN14IE	CN13IE	CN12IE	CN11IE	CN10IE	CN9IE	CN8IE	CN7IE	CN6IE	CN5IE	CN4IE	CN3IE	CN2IE	CN1IE	CN0IE	0000 0000 0000 0000
CNEN2	00C2	-	-	-	-	-	-	-	-	-	-	CN21IE	CN20IE	CN19IE	CN18IE	CN17IE	CN16IE	0000 0000 0000 0000
CNPU1	00C4	CN15PUE	CN14PUE	CN13PUE	CN12PUE	CN11PUE	CN10PUE	CN9PUE	CN8PUE	CN7PUE	CN6PUE	CN5PUE	CN4PUE	CN3PUE	CN2PUE	CN1PUE	CN0PUE	0000 0000 0000 0000
CNPU2	00C6	-	-	-	-	-	-	-	-	-	-	CN21PUE	CN20PUE	CN19PUE	CN18PUE	CN17PUE	CN16PUE	0000 0000 0000 0000

Описание: u = неинициализированный бит

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

9.0 Модуль TIMER1

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

В этом разделе описывается 16-битный модуль универсального Timer1 и связанные с ним режимы работы. На рисунке 9-1 изображена упрощённая блок-схема 16-битного модуля Timer1.

Примечание:	Timer1 есть таймер ‘типа A’r. Пожайлуста обратитесь к описанию для таймера типа A в части 22.0 “Электрические характеристики” этого документа.

Следующие части обеспечивают детальное описание режимов работы таймеров, включая регистры установки и управления вместе с связанными блок-схемами.

Модуль Timer1 является 16-битным таймером который может служить как счётчик времени для часов реального времени, или работать как свободно бегущий интервальный таймер/счётчик. 16-битный таймер имеет следующие режимы:

- 16-битный таймер
- 16-битный синхронный счётчик
- 16-битный асинхронный счётчик

Далее, следующие характеристики поддерживаются:

- Работа управляемым таймером
- Установки выбираемого предделителя
- Работа таймера в течении режимов Спячки и Простоя CPU
- Прерывание при регистрировании соответствия 16-битного периода или падающего фронта внешнего тактового сигнала.

Эти режимы работы определяются установкой соответствующих бит в 16-битном SFR, T1CON. Рисунок 9-1 представляет блок-схему 16 битного модуля таймера.

16-битный режим таймера: В 16-битном режиме таймера, таймер инкрементируется на каждый цикл инструкции до соответствующего значения, предварительно загруженный на период регистр PR1, затем сбрасывается в ‘0’ и продолжает считать.

Когда CPU впадает в режим Простоя, таймер может остановить приращение, если TSIDL (T1CON<13>) бит = 0. Если TSIDL = 1, модуль таймера логически может продолжать последовательность инкрементирования с завершением режима Простоя CPU.

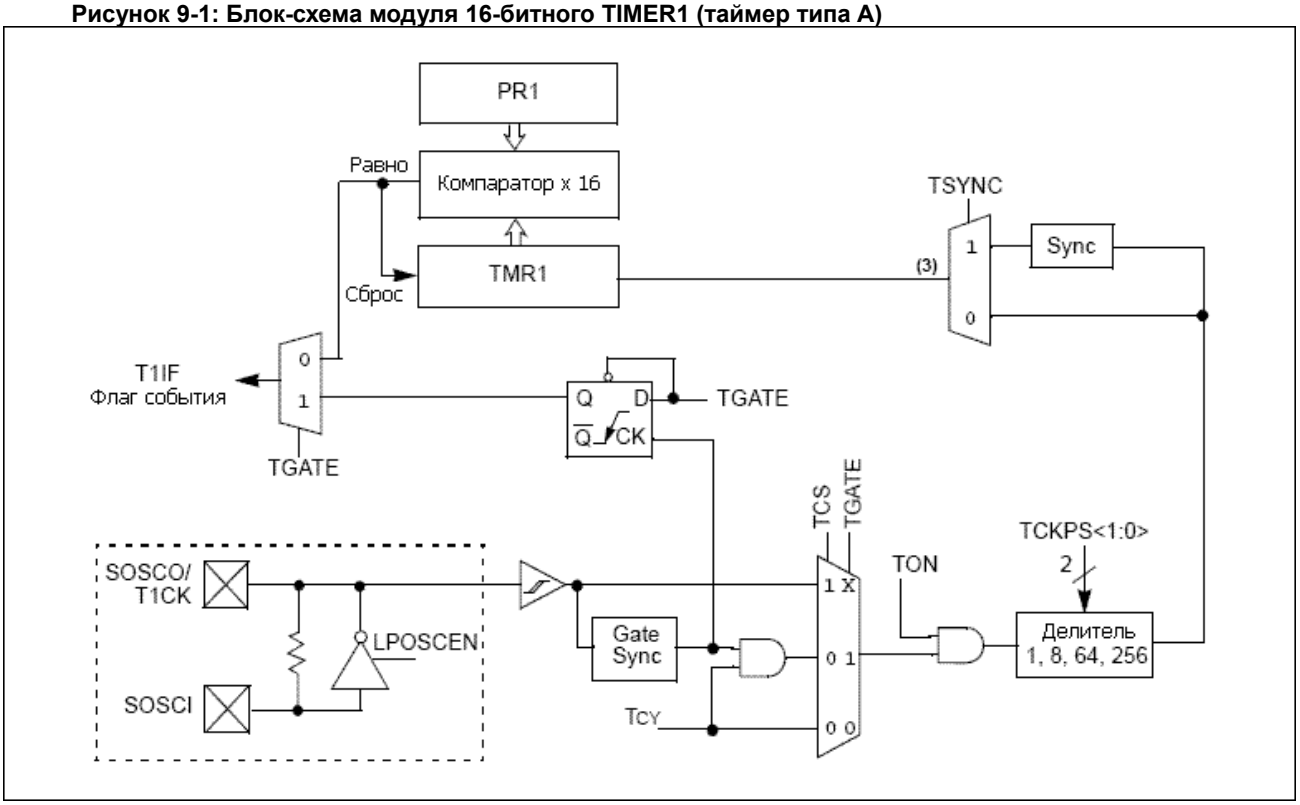
Режим 16-битного синхронного счётчика: В режиме 16-битного синхронного счётчика, таймер инкрементируется на нарастающем фронте прилагаемого внешнего тактового сигнала, который синхронизирован с фазой внутреннего тактирования. Таймер считает до соответствующего значения, загруженного в PR1, затем сбрасывается в ‘0’ и продолжает.

Когда CPU впадает в режим Простоя, таймер останавливает инкрементирование, если соответствующий TSIDL бит = 0. Если TSIDL = 1, модуль таймера логически продолжает последовательность инкрементирования с завершением режима Простоя CPU.

Режим 16-битного асинхронного счётчика: В режиме 16-битного асинхронного счётчика, таймер инкрементируется на каждый нарастающий фронт прилагаемого внешнего тактового сигнала. Таймер считает до соответствующего значения, загруженного в PR1, затем сбрасывается в ‘0’ и продолжает.

Когда таймер конфигурирован для асинхронного режима работы и CPU впадает в режим Простоя, таймер остановит инкрементирование, если TSIDL = 1.

Рисунок 9-1: Блок-схема модуля 16-битного TIMER1 (таймер типа A)



9.1 Timer Gate Operation

16-битный таймер может быть установлен в режим вентильного накопителя времени. Этот режим позволяет внешнему сигналу TCY увеличивать соответствующий таймер, когда управляющий входной сигнал (ножка T1CK) имеет высокий уровень. Бит управления TGATE (T1CON<6>) должен быть установлен для разрешения этого режима. Таймер должен быть разрешён (TON = 1) и установлен внутренний источник тактирования таймера (TCS = 0).

Когда CPU впадает в режим Простоя, таймер останавливает приращение, если TSIDL = 0. Если TSIDL = 1, таймер продолжает последовательность приращения с остановкой режима Простоя CPU.

9.2 Предделитель таймера

Входной тактовый сигнал (FOSC/4 или внешнее тактирование) 16-битного таймера, имеет опции предделителя 1:1, 1:8, 1:64, и

1:256 выбираемые с помощью битов управления TCKPS<1:0> (T1CON<5:4>). Счётчик предделителя очищен, когда любое из следующих условий происходит:

• запись в регистр TMR1

• очистка бита TON (T1CON<15>)

• сброс устройства, такой как POR и BOR

Тем не менее, если таймер отключен (TON = 0), предделитель таймера не может быть сброшен, т.к. тактирование предделителя остановлено.

TMR1 не очищается при записи T1CON. он очищается записью в регистр TMR1.

9.3 Работа таймера в течении режима Спячки

В течении режима Спячки CPU, таймер работает если:

• Модуль таймера разрешён (TON = 1) и

• Выбран внешний Источник тактирования (TCS = 1) и

• Бит TSYNC (T1CON<2>) переведён в логический ‘0’, который определяет внешний источник тактирования как асинхронный

Когда все три условия выполнены, таймер может продолжать считать до регистра периода и, после чего, сброшен в 0x0000.

Когда соответствие между таймером и регистром периода состоится, генерируется прерывание, если установлен соответствующий бит разрешения прерывания таймера.

9.4 Прерывание таймера

16-битный таймер имеет способность генерировать прерывание на соответствие периоду. Когда счётчик таймера соответствует регистру периода, будет установлен бит T1IF и, если разрешено, сгенерировано прерывание. Бит T1IF должен быть очищено программно. Флаг прерывания таймера T1IF расположен в регистре управления IFS0 диспетчера прерывания.

Когда режим вентильного накопителя времени разрешён, прерывание так же может генерироваться по падающему фронту сигнала управления (конец цикла накопления).

Разрешение прерывания выполняется через соответствующий бит разрешения прерывания, T1IE. Бит разрешения прерывания таймера расположен в регистре управления IEC0 диспетчера прерывания.

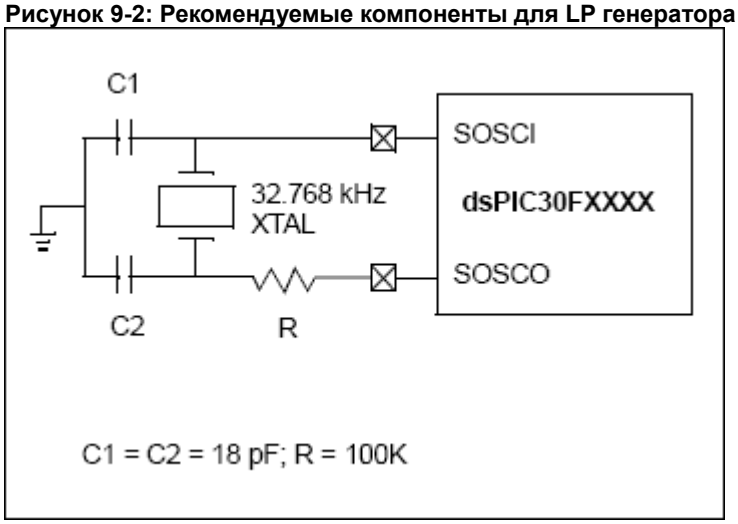
9.5 Часы реального времени

Timer1, когда работает в режиме часов реального времени (RTC), обеспечивает время дня и возможность отмечать время. Ключевые характеристики RTC:

• Работа от 32 kHz LP генератора
• 8-битный предделитель
• Низкое потребление энергии
• Прерывания от часов реального времени

Эти режимы работы определены через установку соответствующих бит в регистре управления.

Рисунок 9-2: Рекомендуемые компоненты для LP генератора RTC TIMER1



9.5.1 Работа генератора RTC

Когда TON = 1, TCS = 1 и TGATE = 0, таймер увеличивается на нарастающий фронт выходного сигнала LP генератора 32 kHz, до значения определённого в регистре периода, и затем сбрасывается в ‘0’.

Бит TSYNC должен быть установлен в логический ‘0’ (асинхронный режим) для правильного функционирования.

Разрешение LPOSCEN (OSCCON<1>) блокирует нормальные режимы таймера и счётчика, и разрешает таймерный перенос пробуждения случай.

Когда CPU входит в режим Спячки, RTC продолжает работу, обеспеченный активным внешним кварцевым генератором 32 kHz и имеет не изменённые биты управления. Бит TSIDL должен быть очищен в ‘0’ в порядке для продолжения работы RTC в режиме Простоя.

9.5.2 Прерывания RTC

Когда происходят события прерывания, соответствующий флаг прерывания, T1IF, устанавливается и прерывание генерируется, если разрешено. Бит T1IF должен быть очищен программно. Соответствующий флаг прерывания таймера, T1IF, расположен в регистре состояния IFS0 диспетчера прерываний.

Разрешение прерывания выполняется через соответствующий бит разрешения прерывания таймера, T1IE. Бит разрешения прерывания расположен в регистре управления IEC0 диспетчера прерываний.

Таблица 9-1: Регистровая карта TIMER1

SFR Имя	Адрес	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
TMR1	0100	Регистр Timer 1																uuuu uuuu uuuu uuuu
PR1	0102	Регистр периода 1																1111 1111 1111 1111
T1CON	0104	TON	-	TSIDL	-	-	-	-	-	-	TGATE	TCKPS1	TCKPS0	-	TSYNC	TCS	-	0000 0000 0000 0000

Описание: u = неинициализированный бит

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

10.0 Модуль TIMER2/3

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

В этой части описывается 32-битный универсальный модуль таймера (Timer2/3) и связанные с ним режимы работы. На рисунке 10-1 изображена упрощённая блок-схема 32-битного модуля Timer2/3. На рисунке 10-2 и рисунке 10-3 показана конфигурация Timer2/3, как два независимых 16-битных таймера; Timer2 и Timer3 соответственно.

Примечание:	Timer2 является таймером ‘типа B’ и Timer3 является таймером ‘типа C’. Пожайлуста обратитесь к соответствующему типу таймера в части 22.0 “Электрические характеристики” этого документа.

Модуль Timer2/3 является 32-битным таймером, который может быть сконфигурирован как два 16-битных таймера, с избирательными режимами работы. Эти таймеры используются в других периферийных модулях, таких как:

• Вход захвата
• Выход сравнения/Простой PWM

Следующие части предоставляют детальное описание, включающее регистры установки и управления, вместе с соответствующими блок-схемами для режимов работы таймеров.

32-битный таймер имеет следующие режимы:

• Два независимых 16-битных таймера (Timer2 и
Timer3) со всеми 16-битными режимами работы (кроме режима асинхронного счётчика)
• Работа в качестве одиночного 32-битного таймера
• Одиночного 32-битного синхронного счётчика

Далее, следующие рабочие характеристики поддерживаются:

• ADC триггер события
• Работа управляемого таймера
• Выбираемые параметры настройки предделителя
• Работа таймера в режимах Простоя и Спячки
• Прерывание на совпадение с 32-битным регистром периода

Эти рабочие режимы определяются установкой соответствующих битов в 16-битном T2CON и T3CON SFR.

Для работы 32-битного таймера/счётчика, Timer2 является наименьшим значимым словом и Timer3 является наибольшим значимым словом 32-битного таймера.

Примечание:	Для работы 32-битного таймера, биты управления T3CON игнорируются. Только биты управления T2CON используются для установки и управления. Тактовый и управляющий входы Timer 2 используются для 32-битного модуля таймера, но прерывание генерируется с флагом прерывания Timer3 (T3IF), и прерывание разрешается битом разрешения прерывания Timer3 (T3IE).

16-битный режим: В 16-битном режиме, Timer2 и Timer3 могут быть сконфигурированы как два независимых 16-bit таймера. Каждый таймер может быть установлен в любой 16-битный режим таймера или режим 16-битного синхронного счётчика. Смотреть часть 9.0 “Модуль Timer1” для деталей на эти два режима работы.

Функциональная разница между Timer2 и Timer3 только в том, что Timer2 обеспечивает тактового выхода предделителя. Это используется для высокочастотных внешних тактовых входов.

32-битный режим таймера: В 32-битном режиме таймера, таймер приращается на каждый цикл инструкции до совпадения с значением, предварительно загруженным в объединённый 32-битный регистр периода PR3/PR2, затем сбрасывается в ‘0’ и продолжает счёт.

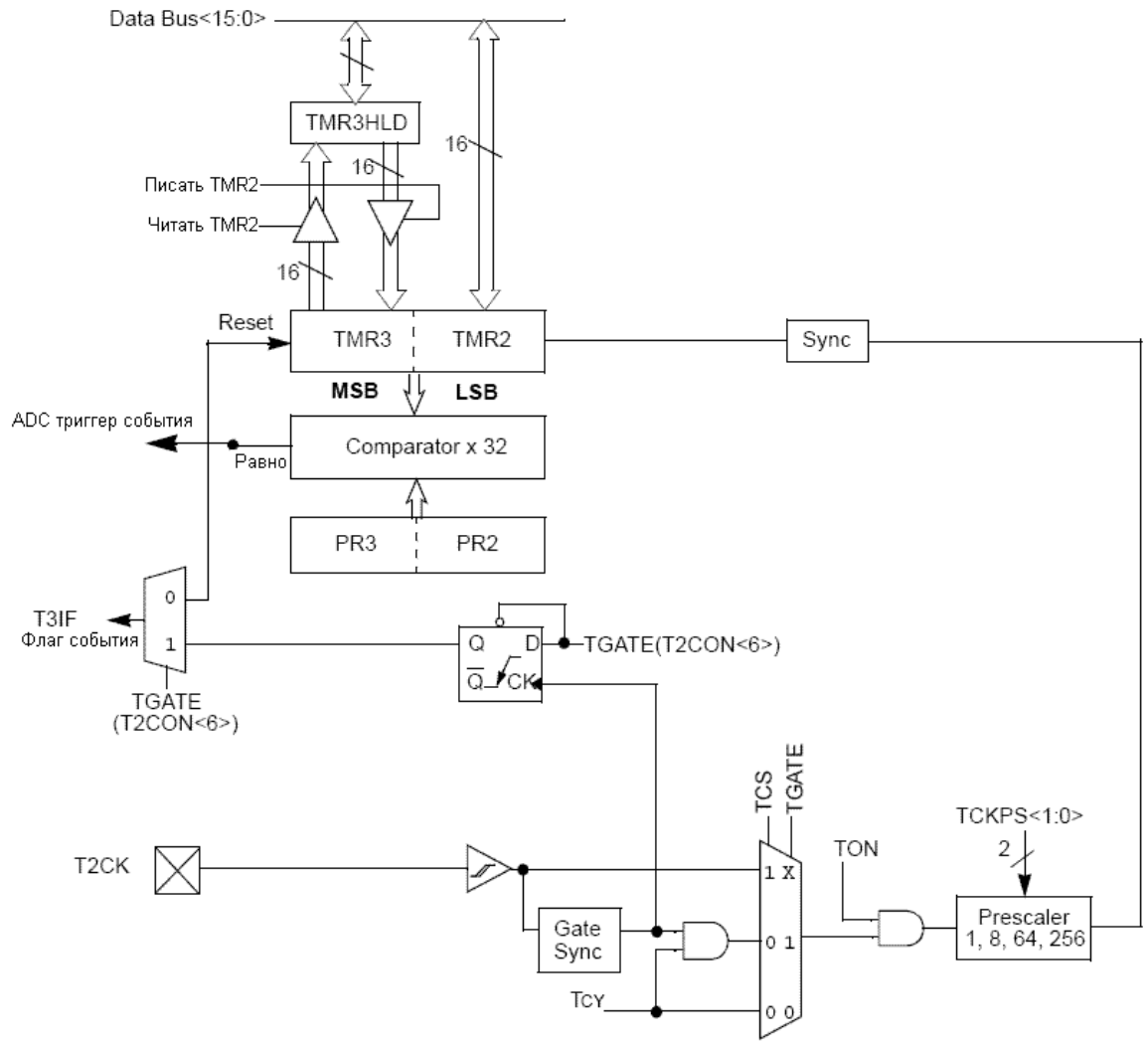
Для синхронного чтения пары Timer2/Timer3, чтение наименьшего значимого слова (TMR2 регистр) заставляет наибольшее значимое слово (msw) прочитаться и защёлкнуться в 16-битный регистр хранения, называемый TMR3HLD.

Для синхронной 32-битной записи, регистр хранения (TMR3HLD) должен быть записан первым. В то время как следующий передаётся и защёлкивается в MSB 32-битного таймера (TMR3).

Режим 32-битного синхронного счётчика: В режиме 32-битного синхронного счётчика, таймер увеличивается по нарастающему фронту прилагаемого внешнего тактового сигнала, который синхронизирован с фазой внутреннего тактового генератора. Таймер считает до совпадения значения предварительно загруженного в объединённый 32-битный регистр периода PR3/PR2, затем сбрасывается в ‘0’ и продолжает.

Когда таймер сконфигурирован для работы в режиме синхронного счётчика и CPU уходит в режим Простоя, таймер останавливает приращение, если TSIDL (T2CON<13>) бит = ‘0’. Если TSIDL = ‘1’, модуль таймера логически продолжает последовательность приращения с завершением режима Простоя CPU.

Рисунок 10-1: Блок-схема 32-битного TIMER2/3



Прмечание: Бит конфигурации таймера T32, T2CON(<3>) должны быть установлены в 1 для работы 32-битным таймером/счётчиком. Все биты управления соответствуют регистру T2CON.

Рисунок 10-2: Блок схема 16-битного TIMER2 (таймер типа B)

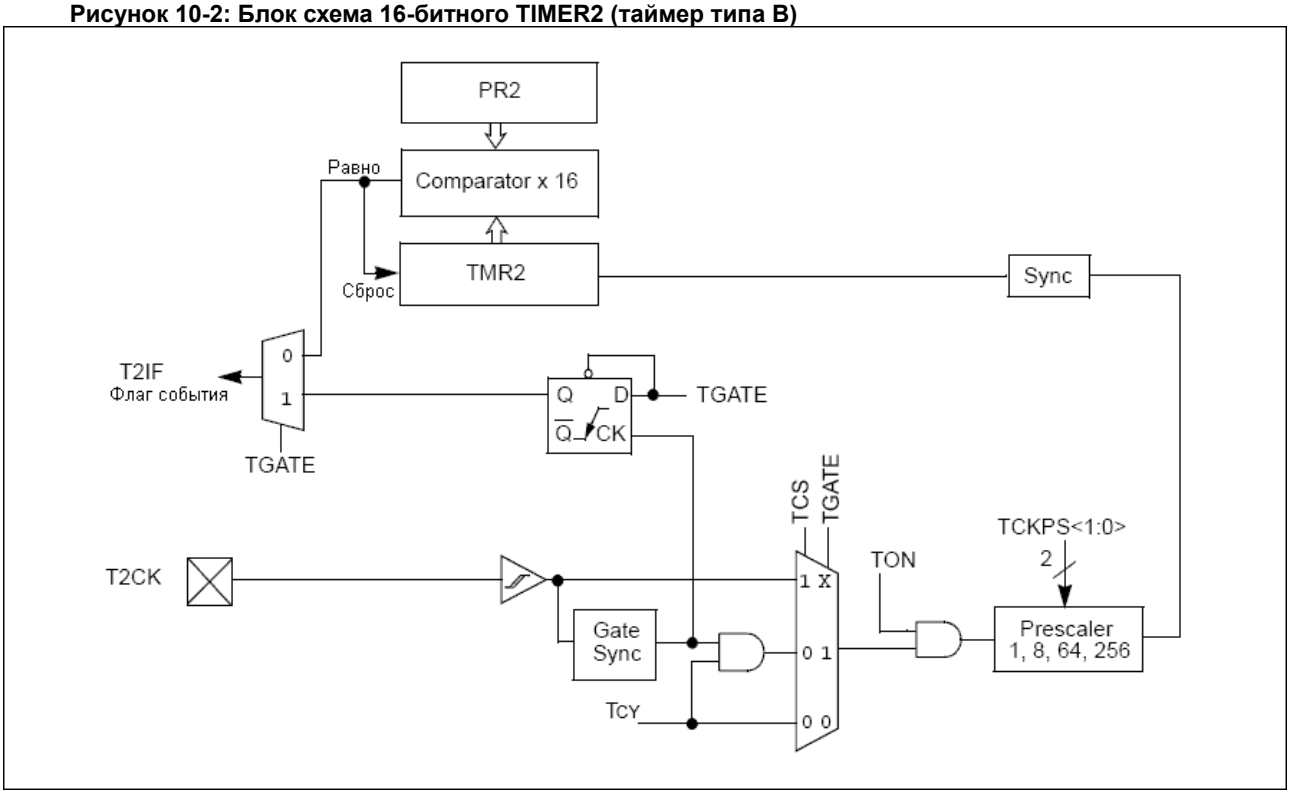
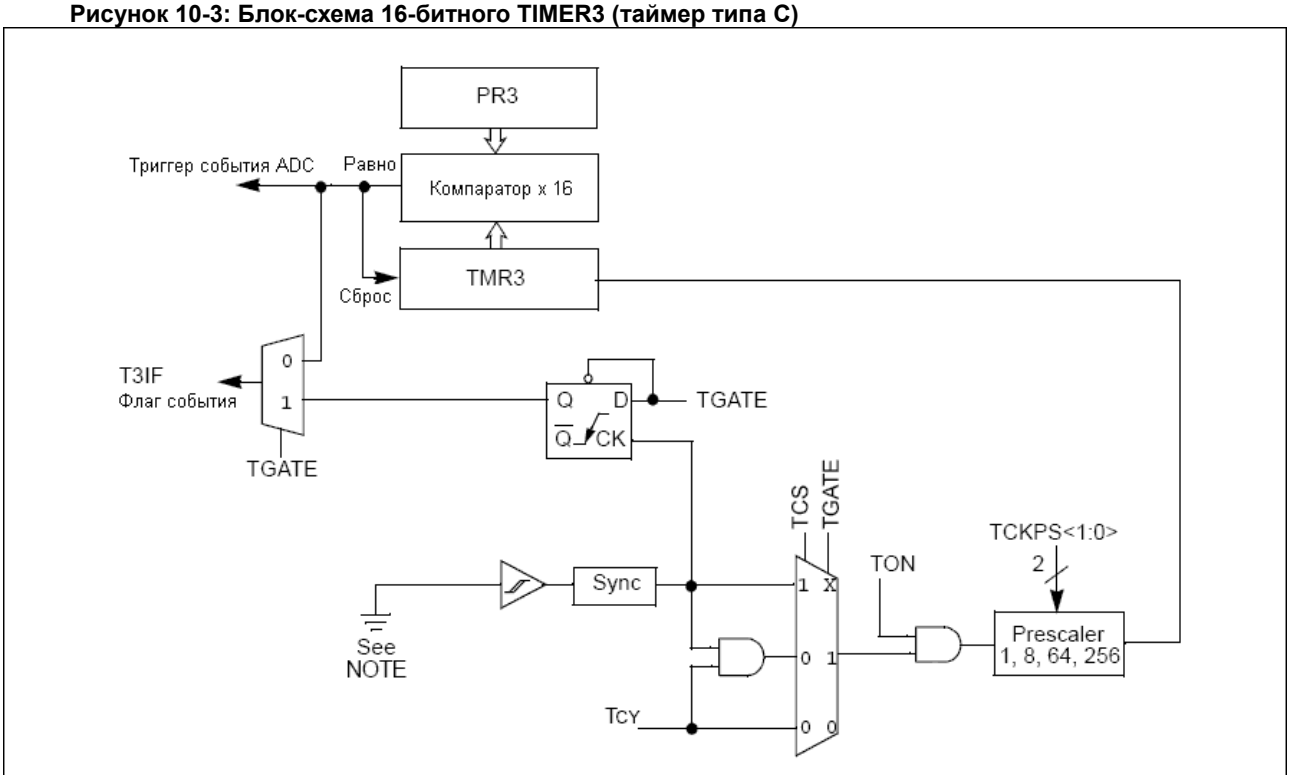


Рисунок 10-3: Блок-схема 16-битного TIMER3 (таймер типа C)



Примечание: dsPIC30F2010 не имеет внешнюю ножку для входа TIMER3. Следующие режимы не должны использоваться:
1. TCS = 1
2. TCS = 0 и TGATE = 1 (управляемый аккумулятор таймера)

10.1 Timer Gate Operation

32-битный таймер может быть установлен в режим вентильного накопителя времени. Этот режим позволяет внешнему сигналу TCY приращать соответствующий таймер, когда управляющий входной сигнал (ножка T2CK) имеет высокий уровень. Бит управления TGATE (T2CON<6>) должен быть установлен для разрешения этого режима. Когда в этом режиме, Timer2 порождает тактовый источник. Установка TGATE игнорируется для Timer3. Таймер должен быть разрешён (TON = 1) и установлен внутренний источник тактирования таймера (TCS = 0).

Падающий фронт внешнего сигнала завершает операцию счёта, но не сбрасывает таймер. Пользователь может сбросить таймер в порядке начала счёта с нуля.

10.2 Событие запуска ADC

Когда происходит совпадение между 32-битным таймером (TMR3/TMR2) и 32-битным объединённым регистром периода (PR3/PR2), специальный сигнал события запуска ADC генерируется Timer3.

10.3 Предделитель таймера

Входной тактовый сигнал (FOSC/4 или внешний тактовый сигнал) таймера имеет опции предварительного деления 1:1, 1:8, 1:64, и 1:256, выбираемые битами управления TCKPS<1:0> (T2CON<5:4> и T3CON<5:4>). Для работы 32-битного таймера, порождающим тактовым источником является Timer2. Операция предварительного деления для Timer3 в этом режиме не предусмотрена. Счётчик предделителя очищается, когда следующее происходит:

• запись в регистр TMR2/TMR3
• очистка в ‘0’ любого из битов TON (T2CON<15> или T3CON<15>)
• аппаратный сброс, как например POR и BOR

Тем не менее, если таймер отключен (TON = 0), то предделитель Timer2 не может быть сброшен. TMR2/TMR3 не очищаются, когда T2CON/T3CON записываются.

10.4 Работа таймера в течении режима Спячки

В течении режима Спячки CPU, таймер не работает, потому что отключен внутренний тактовый сигнал.

10.5 Прерывание таймера

Модуль 32-битного таймера может генерировать прерывание на совпадение периода, или на падающий фронт внешнего управляющего сигнала. Когда 32-битный таймер считает совпадения соответствующего 32-битного регистра периода, или детектирует падающий фронт внешнего сигнала “gate”, бит T3IF (IFS0<7>) установлен и прерывание может генерироваться , если разрешено. В этом режиме, флаг прерывания T3IF используется как источник прерывания. Бит T3IF должен быть очищен программно.

Разрешение прерывания выполняется через соответствующий бит разрешения прерывания таймера, T3IE (IEC0<7>).

Таблица 10-1: Карта регистров TIMER2/3

SFR Имя	Адрес	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
TMR2	0106	Регистр Timer 2																uuuu uuuu uuuu uuuu
TMR3HLD	0108	Регистр хранения Timer3 (Только для 32-битных таймерных операций)																uuuu uuuu uuuu uuuu
TMR3	010A	Регистр Timer 3																uuuu uuuu uuuu uuuu
PR2	010C	Регистр периода 2																1111 1111 1111 1111
PR3	010E	Регистр периода 3																1111 1111 1111 1111
T2CON	0110	TON	-	TSIDL	-	-	-	-	-	-	TGATE	TCKPS1	TCKPS0	T32	-	TCS	-	0000 0000 0000 0000
T3CON	0112	TON	-	TSIDL	-	-	-	-	-	-	TGATE	TCKPS1	TCKPS0	-	-	TCS	-	0000 0000 0000 0000

Описание: u = неинициализированный бит

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

11.0 Модуль захвата входа

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

В этой части описывается модуль захвата входа и его режимы работы. Характеристики, обеспечиваемые этим модулем полезны в приложениях требующих измерения частоты (периода) и длительности импульса. На рисунке 11-1 изображена блок-схема модуля захвата входа. Захват входа полезен для таких режимов как:

• Измерения Частоты/Периода/Импульса
• Дополнительный источник внешних прерываний

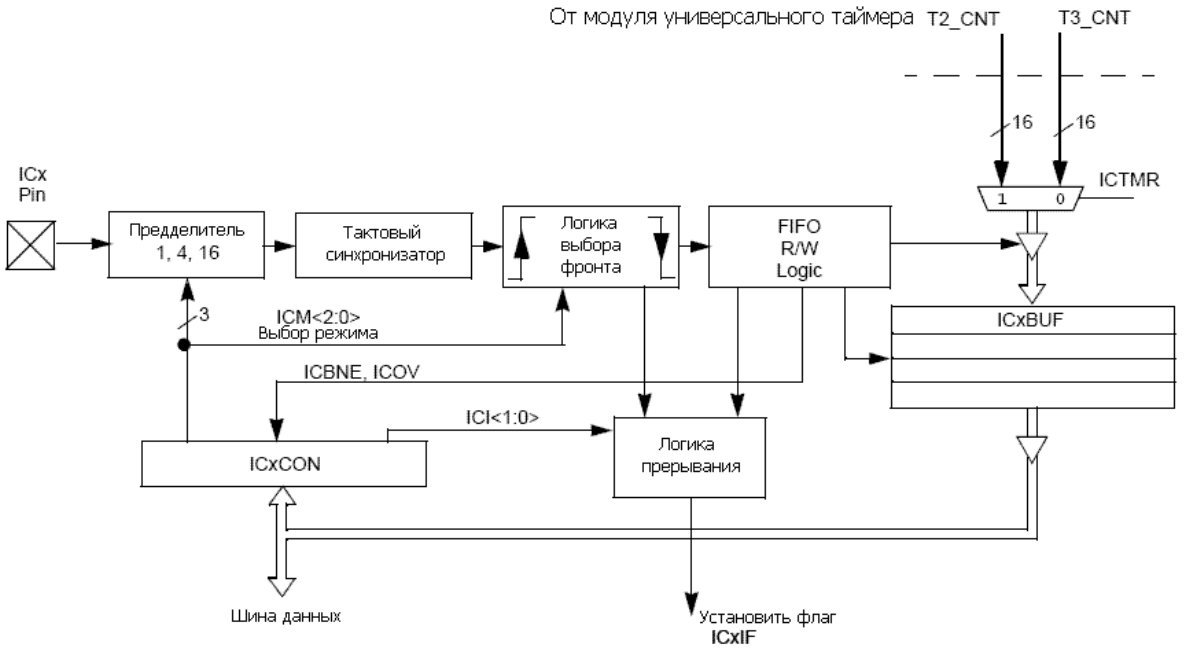
Ключевыми рабочими характеристиками модуля захвата входа являются:

• Режим простого захвата события
• Выбор режима Timer2 и Timer3
• Прерывание по случаю захвата входа

Эти режимы работы определяются через установку соответствующих битов регистра ICxCON (где x = 1,2,...,N). Устройство dsPIC DSC содержит до 8 каналов захвата, (т.е., максимальное значение N равно 8).

Примечание:	Устройство dsPIC30F2010 имеет четыре входа захвата – IC1, IC2, IC7 и IC8. Присвоение имён этим четырём каналам захвата было не случайным и сохраняет программную совместимость с другими устройствами dsPIC DSC.

Рисунок 11-1: Блок-схема режима захвата входа



Примечание: Там где показан 'x', подразумевается ссылка на регистры или биты связанные с соответствующими каналами захвата входа от 1 до N.

11.1 Режим простого захвата события

В семействе устройств dsPIC30F реализованы следующие виды простого захвата событий:

• Захват каждого падающего фронта
• Захват каждого нарастающего фронта
• Захват каждого 4-го нарастающего фронта
• Захват каждого 16-го нарастающего фронта
• Захват каждого нарастающего и падающего фронтов

Эти простые режимы захвата входа конфигурируются установкой соответствующих битов ICM<2:0> (ICxCON<2:0>).

11.1.1 Предварительный делитель захвата

Есть четыре настройки предделителя захвата входа, определяемые битами ICM<2:0> (ICxCON<2:0>). Всякий раз, когда канал захвата выключен, счётчик предделителя очищен. В дополнение, любой сброс очищает счётчик предделителя.

11.1.2 Операционный буфер захвата

Каждый канал захвата имеет связанный с ним FIFO буфер, который имеет глубину в четыре 16-битных слова. Там имеется два флага состояния, которые обеспечивают индикацию состояние FIFO буфера:

• ICBNE – Буфер захвата входа не пустой
• ICOV – Захват входа переполнен

ICBFNE устанавливается первым событием захвата входа и остаётся установленным, пока все события захвата не будут прочитаны из FIFO. После чтения каждого слова из FIFO, оставшиеся слова продвигаются на одну позицию в пределах буфера.

В случае, когда FIFO заполнен четырьмя событиями захвата и пятое событие захвата происходит до чтения FIFO, случится состояние переполнения и бит ICOV будет установлен в логическую ‘1’. Пятый случай захвата пропадает и не запоминается в FIFO. Дополнительные события не могут захватываться пока все четыре события не будут прочитаны из буфера.

Если чтение FIFO выполняется после последнего чтения и новый захват события не был принят, чтение может давать неопределённый результат.

11.1.3 Режим выбора TIMER2 и TIMER3

Модуль захвата входа содержит до 8 каналов захвата входа. Каждый канал выбирается между двумя таймерами для временной базы, Timer2 or Timer3.

Выбор таймерного ресурса выполняется через SFR бит ICTMR (ICxCON<7>). Timer3 является таймерным ресурсом доступным по умолчанию для модуля захвата входа.

11.1.4 Режим сенсора Холла

Когда модуль захвата входа установлен для захвата каждого фронта, нарастающего и падающего, ICM<2:0> = 001, следующие действия выполняются логикой захвата входа:

• Флаг прерывания захвата входа устанавливается на каждый фронт, нарастающий или падающий.
• Биты установки включения прерывания режима захвата, ICI<1:0>, игнорируются, поскольку каждый захват генерирует прерывание.
• Состояние переполнения захвата не генерируется в этом режиме.

11.2 Работа захвата входа в течении режимов Простоя и Спячки

Случай захвата входа может генерировать пробуждение устройства или прерывание, если разрешено, если устройство находится в режиме Простоя CPU или Спячки.

Independent of the timer being enabled, the input

Модуль захвата пробуждает из режима CPU Спячки или Простоя, когда событие захвата происходит, если ICM<2:0> = 111 и установлен бит разрешения прерывания. То же самое пробуждение может генерировать прерывание, если условие для обработки прерывания соответствует. Особенность пробуждения полезна как метод дополнения ножек с которых возможны внешние прерывания.

11.2.1 Захват входа в режиме CPU Спячки

Режим CPU Спячки позволяет модулю захвата входа работать с уменьшенной функциональностью. В режиме CPU Спячки, биты ICI<1:0> не применяются, модуль захвата входа может функционировать только как источник внешнего прерывания.

Модуль захвата должен быть сконфигурирован для прерывания только по нарастающему фронту (ICM<2:0> = 111), в порядке для модуля захвата входа, используется пока устройство находится в режиме Спячки. Установки предделителя 4:1 или 16:1 не применяются в этом режиме.

11.2.2 Захват входа в режиме Простоя CPU

Режим Простоя CPU позволяет модулю захвата входа работать с полной функциональностью. В режиме Простоя CPU, режим прерывания выбирается битами ICI<1:0> применяется, как хорошо как 4:1 и 16:1 установки предделителя захвата, который определён битами управления ICM<2:0>. Этот режим чтобы выбранный таймер был разрешён. Кроме того, бит ICSIDL должен быть переведён в логический ‘0’.

Если модуль захвата входа определён как ICM<2:0> = 111 в режиме Простоя CPU, ножка захвата входа обслуживается только как ножка внешнего прерывания.

11.3 Прерывания захвата входа

Каналы захвата входа имеют способность генерировать прерывание, основанное на выбранном числе событий захвата. Число выбирается установкой битов управления ICI<1:0> (ICxCON<6:5>). Каждый канал обеспечивает бит флага прерывания (ICxIF). Флаг прерывания соответствующего канала захвата расположен в соответствующем регистре статуса IFSx.

Разрешение прерывания выполняется через соответствующий каналу захвата бит разрешения прерывания (ICxIE). Бит разрешения прерывания захвата расположен в соответствующем регистре управления IEC.

Таблица 11-1: Карта регистров захвата входа

SFR Имя	Адрес	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
IC1BUF	0140	Регистр захвата входа 1																uuuu uuuu uuuu uuuu
IC1CON	0142	-	-	ICSIDL	-	-	-	-	-	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC2BUF	0144	Регистр захвата входа 2																uuuu uuuu uuuu uuuu
IC2CON	0146	-	-	ICSIDL	-	-	-	-	-	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC3BUF	0148	Регистр захвата входа 3																uuuu uuuu uuuu uuuu
IC3CON	014A	-	-	ICSIDL	-	-	-	-	-	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC4BUF	014C	Регистр захвата входа 4																uuuu uuuu uuuu uuuu
IC4CON	014E	-	-	ICSIDL	-	-	-	-	-	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC5BUF	0150	Регистр захвата входа 5																uuuu uuuu uuuu uuuu
IC5CON	0152	-	-	ICSIDL	-	-	-	-	-	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC6BUF	0154	Регистр захвата входа 6																uuuu uuuu uuuu uuuu
IC6CON	0156	-	-	ICSIDL	-	-	-	-	-	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC7BUF	0158	Регистр захвата входа 7																uuuu uuuu uuuu uuuu
IC7CON	015A	-	-	ICSIDL	-	-	-	-	-	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC8BUF	015C	Регистр захвата входа 8																uuuu uuuu uuuu uuuu
IC8CON	015E	-	-	ICSIDL	-	-	-	-	-	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000

Описание: u = неинициализированный бит

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

12.0 Модуль сравнения выхода

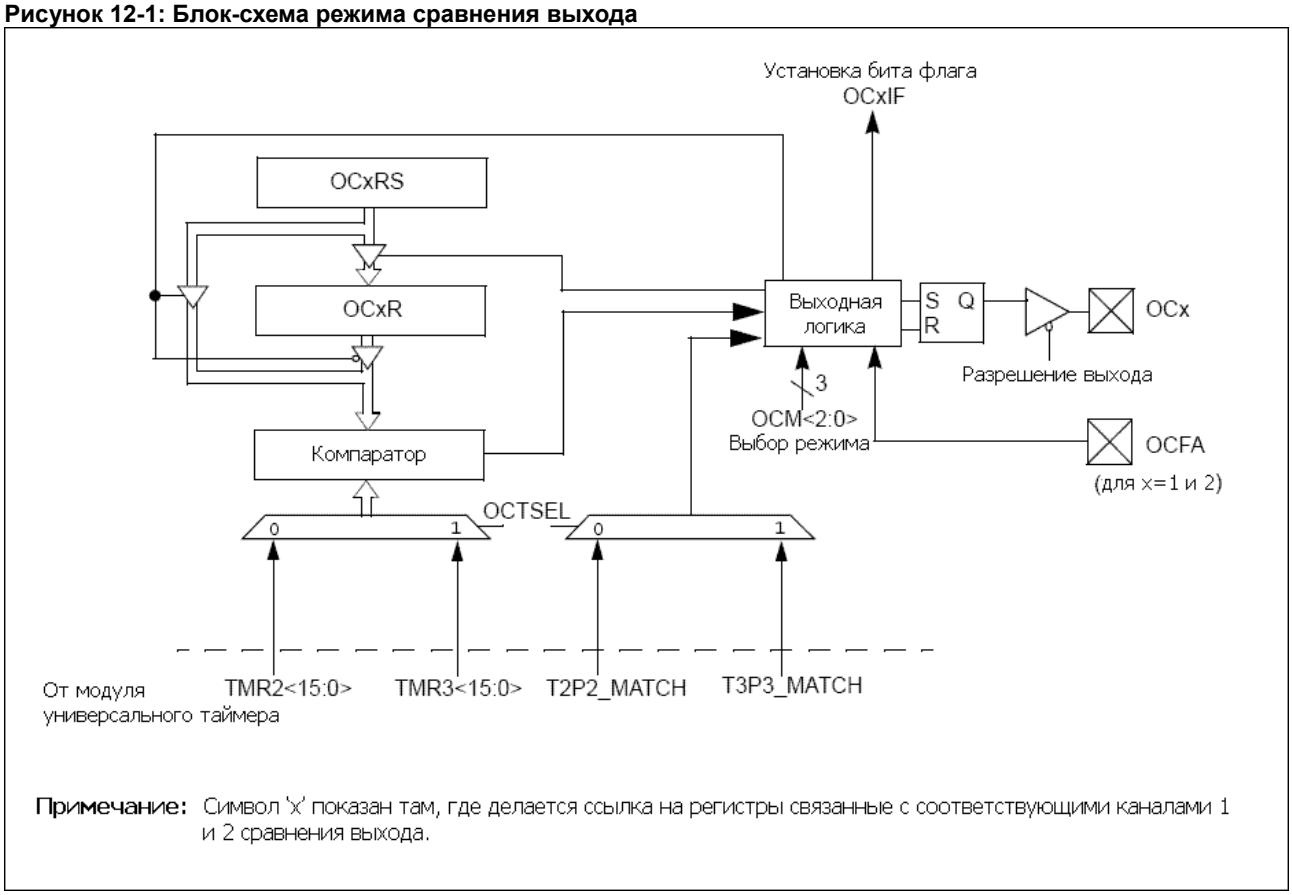
Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

В этой части описывается модуль сравнения выхода и связанные с ним режимы работы. Характеристики, обеспечиваемые этим модулем полезны в приложениях требующих оперативных режимов, таких как:

- Генерация выходных импульсов переменной ширины
- Коррекция коэффициента мощности

На рисунке 12-1 изображена блок-схема модуля сравнения выхода.

Рисунок 12-1: Блок-схема режима сравнения выхода



Примечание: Символ 'x' показан там, где делается ссылка на регистры связанные с соответствующими каналами 1 и 2 сравнения выхода.

12.1 Режим выбора Timer2 и Timer3

Каждый канал сравнения выхода может выбирать один из двух 16-битных таймеров: Timer2 или Timer3. Выбор таймеров управляется битом (OCxCON<3>). Timer2 является таймером источником, принятым по умолчанию, для модуля сравнения выхода.

12.2 Простой режим сравнения соответствия выхода

Когда биты управления OCM<2:0> (OCxCON<2:0>) = 001, 010 или 011, выбранный канал сравнения выхода сконфигурирован для одного из трёх простых режимов сравнения совпадения выхода:

- Компаратор переводит ножку I/O в низкий уровень
- Компаратор переводит ножку I/O в высокий уровень
- Компаратор переключает ножку I/O

Регистр OCxR используется в этих режимах. Регистр OCxR загружается значением и сравнивается с выбранным инкрементируемым счётчиком таймера. Когда происходит сравнение, один из этих компраторов происходит режимы соответствия. Если счётчик сбрасывается в ноль перед достижением значения в OCxR, состояние ножки OCx остаётся неизменным.

12.3 Двойной режим сравнения соответствия выхода

Когда биты управления OCM<2:0> (OCxCON<2:0>) = 100 или 101, выбранный канал сравнения выхода сконфигурирован для одного из двух режимов двойного сравнения выхода, который есть:

- Простой режим выходного импульса
- Непрерывный режим выходного импульса

12.3.1 Режим единственного импульса

Если пользователь конфигурирует модуль для генерации одиночного выходного импульса, требуются следующие шаги (допустим таймер отключен):

- Определить время выполнения цикла инструкции TCY.
- Вычислить желаемую ширину импульса в величине основанной на TCY.
- Вычислить время начала импульса из стартового значения таймера 0x0000.
- Записать ширину импульса и время остановки в OCxR и регистр сравнения OCxRS (x означает канал 1, 2).
- Установить в качестве времени регистра периода значение эквивалентное, или больше чем, значению в регистре сравнения.
- Установить OCM<2:0> = 100.
- Разрешить таймер, TON (TxCON<15>) = 1.

Инициировать другой одиночный импульс, передачей другой записи установив OCM<2:0> = 100.

12.3.2 Непрерывный импульсный режим

Если пользователь конфигурирует модуль для генерации непрерывного потока выходных импульсов, следующие шаги необходимы:

- Определить время цикла инструкции TCY.
- Вычислить желаемое значение импульса основанное на TCY.
- Вычислит время старта ширины импульса из таймерного стартового значения 0x0000.
- Записать старт ширины импульса и время остановки в OCxR и OCxRS (x обозначает канал I 1, 2) регистров сравнения, соответственно.
- Установит время в регистре периода в значение равное, или больше чем, значению в регистре компаратора.
- Установить OCM<2:0> = 101.
- Разрешить таймер, TON (TxCON<15>) = 1.

12.4 Простой режим PWM

Когда биты управления OCM<2:0> (OCxCON<2:0>) = 110 или or 111, выбранный выходной канал компаратора сконфигурирован для работы в режиме PWM. Когда сконфигурирован для работы в режиме PWM, OCxR есть основная защёлка (только чтение) и OCxRS является второй защёлкой. Это позволяет уменьшить импульсную помеху PWM переходов.

Пользователь должен выполнить следующие шаги в порядке конфигурации модуля сравнения выхода для работы PWM:

1. Установить период PWM записав соответствующий регистр периода.
2. Установить PWM заполнение, записав регистр OCxRS.
3. Конфигурировать модуль сравнения выхода для работы PWM.
4. Установить значение предделителя TMRx и разрешить таймер, TON (TxCON<15>) = 1.

12.4.1 Защита от дефекта входной ножки для PWM

Когда биты управления OCM<2:0> (OCxCON<2:0>) = 111, выбранный канал сравнения выхода снова сконфигурирован для PWM режима работы, с дополнительной характеристикой защиты от ошибки входа. Пока в этом режиме, если логический ‘0’ обнаружен на ножке OCFA/B, соответствующая выходная ножка PWM устанавливается в высокоимпедансное состояние входа. Бит OCFLT (OCxCON<4>) независимо указывает на то что произошел дефект. Это состояние может быть поддерживаться пока оба следующих события произошли:

- Состояние внешнего дефекта было удалено.
- Режим PWM был переразрешён записью соответствующих бит управления.

12.4.2 Период PWM

Период PWM определяется записью регистра PRx. PWM период может быть вычислен с использованием формулы 12-1.

Формула 12-1: Период PWM

Период PWM = [(PRx) + 1] • 4 • TOSC • (TMRx значение предделителя)

Частота PWM определяется как 1 / [период PWM].

Когда выбранный TMRx равен соответствующему регистру периода, PRx, следующие четыре события происходят на следующий цикл приращения:

• TMRx очищен.

• Ножка OCx устанавливается.

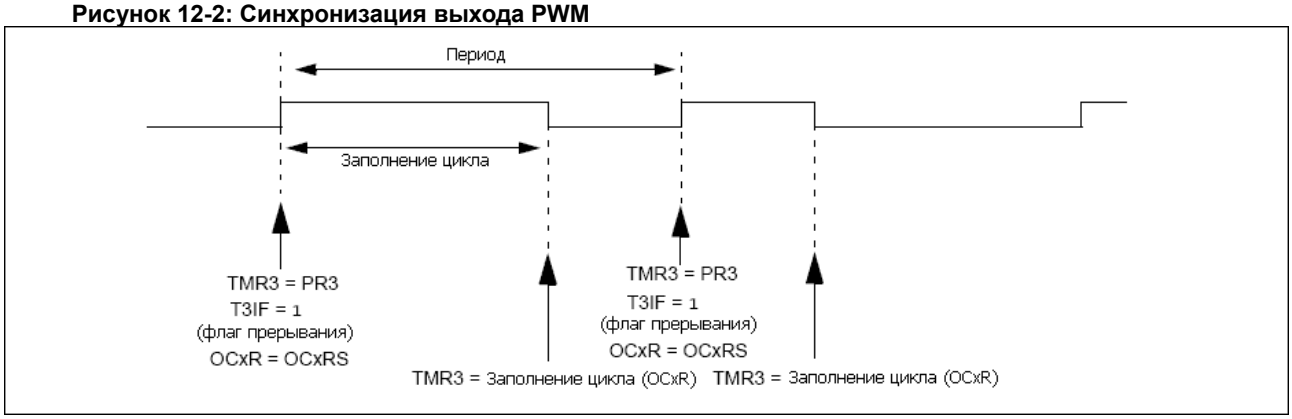
- Исключение 1: Если заполнение цикла PWM есть 0x0000, на ножке OCx остаётся низкий уровень.
- Исключение 2: Если заполнение цикла больше чем PRx, на ножке остаётся высокий уровень.

• Заполнение цикла PWM защёлкивается из OCxRS в OCxR.

• Соответствующий флаг прерывания таймера установлен.

Смотреть рисунок 12-2 для ключевых сравнений периода PWM. Timer3 отсылает к рисунку для ясности.

Рисунок 12-2: Синхронизация выхода PWM



12.5 Работа сравнения выхода в течении режима Спячки CPU

Кгда CPU входит в режим Спячки, всё внутреннее тактирование останавливается. Следовательно, когда CPU входит в состояние Speep, канал сравнения выхода переводит ножку в то активное состояние, которое наблюдалось перед входом CPU в состояние Спячки.

Например, если ножка была в высоком состоянии, когда CPU входил в состояние Спячки, ножка остаётся высокой. Подобно, если ножка была в низком состояние, когда CPU входил в состояние Спячки, ножка остаётся низкой. В любом случае, модкль сравнения выхода будет продолжать работать когда устройство проснётся.

12.6 Работа сравнения выхода в течении режима Простоя CPU

Когда CPU входит в режим Простоя, модуль сравнения выхода может работать полно функционально.

Канал сравнения выхода работает в течении режима Простоя CPU если бит (OCxCON<13>) находится в логическом ’0’ и выбранный основной таймер (Timer2 или Timer3) разрешён и бит TSIDL выбранного таймера установлен в логический ‘0’.

12.7 Прерывания сравнения выхода

Каналы сравнения выхода имеют способность генерировать прерывание на сравнение соответствие, для какого угодно выбранного режима соответствия.

Для всех режимов, за исключением режима PWM, когда сравнение событие происходит, соответствующий флаг прерывания (OCxIF) утверждается и генерируется прерывание, если разрешено. Бит OCxIF расположен в соответствующем регистре состояния IFS, и должен очищаться программно. Прерывание разрешается через соответствующий бит разрешения прерывания сравнения (OCxIE), расположенный в соответствующем управляющем регистре IEC.

Для режима PWM, когда событие происходит, соответствующий флаг прерывания таймера (T2IF или T3IF) утверждается и прерывание генерируется, если разрешено. Бит IF расположен в регистре состояния IFS0, и должен быть очищен программно. Прерывание разрешается через соответствующий бит разрешения прерывания таймера (T2IE или T3IE), расположенный в регистре управления IEC0. Флаг прерывания сравнения выхода не устанавливается в течении работы в режиме PWM.

Таблица 12-1: Регистровая карта сравнения выхода

SFR Имя	Адрес	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
OC1RS	0180	Сравнение выхода 1 Мастер регистр																0000 0000 0000 0000
OC1R	0182	Сравнение выхода 1 Помощник регистр																0000 0000 0000 0000
OC1CON	0184	-	OCFRZ	OCSIDL	-	-	-	-	-	-	-	-	OCFLT1	OCTSEL1	OCM<2:0>			0000 0000 0000 0000
OC2RS	0186	Сравнение выхода 2 Мастер регистр																0000 0000 0000 0000
OC2R	0188	Сравнение выхода 2 Помощник регистр																0000 0000 0000 0000
OC2CON	018A	-	OCFRZ	OCSIDL	-	-	-	-	-	-	-	-	OCFLT1	OCTSEL1	OCM<2:0>			0000 0000 0000 0000

Описание: u = неинициализированный бит

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

13.0 Модуль интерфейса квадратурного энкодера (QEI)

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

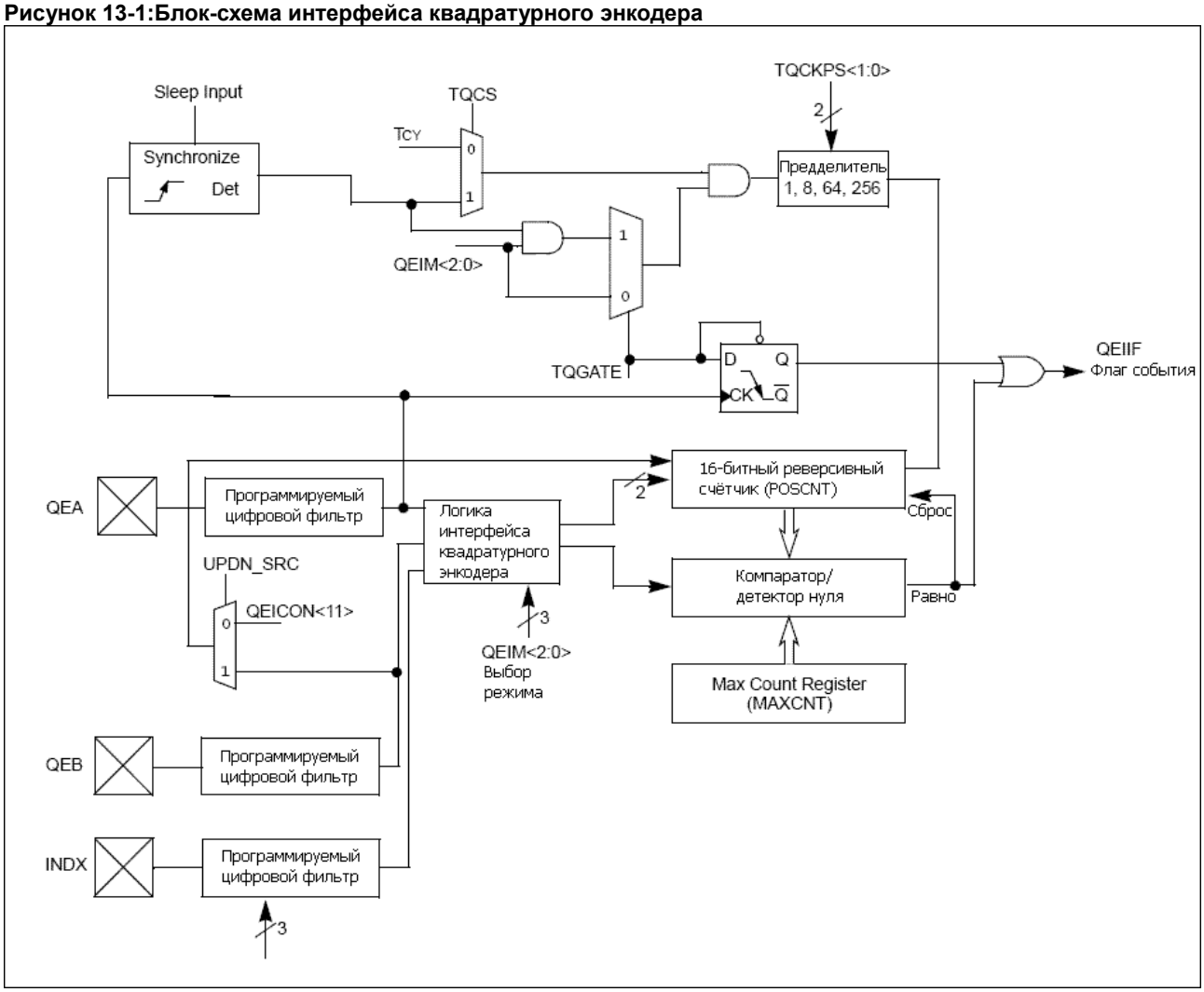
В этой части описывается модуль интерфейса квадратурного энкодера (QEI) и связанные с ним режимы работы. Модуль QEI обеспечивает интерфейс с шаговыми энкодерами для получения данных о положении ротора мотора. Шаговый энкодер очень часто используется в различных приложениях управления двигателями.

Интерфейс квадратурного энкодера (QEI) является ключевым элементом различных приложений управления двигателями, такими как вентильный двигатель (SR) и асинхронный двигатель (ACIM). Действующие характеристики QEI есть, но не ограничивать в:

- Три входных канала для двухфазного сигнала и индексного импульса
- 16-битный реверсивный счётчик положения
- Счётчик состояния направления
- Режим измерения положения (x2 и x4)
- Программируемый цифровой фильтр шума на входах
- Режим изменения 16-битного таймера/счётчика
- Прерывания интерфейса квадратурного энкодера

Эти режимы работы определяются установкой соответствующих бит QEIM<2:0> (QEICON<10:8>). На рисунке 13-1 изображена блок-схема интерфейса квадратурного энкодера.

Рисунок 13-1:Блок-схема интерфейса квадратурного энкодера



13.1 Логика интерфейса квадратурного энкодера

Типовой пошаговый (оптический) энкодер имеет три выхода: Фаза A, Фаза B, и индексный импульс. Эти сигналы используются и часто требуются для контроля скорости и положения ротора ACIM и SR двигателей.

Два канала, фаза A (QEA) и фаза B (QEB), имеют уникальное взаимоотношение. Если фаза A опережает фазу B, значит направление (ротора) считается положительным или прямым. Если фаза A отстаёт от фазы B, значит направление (ротора) считается отрицательным или обратным.

Третий канал, называемый индексный импульс, случается один раз за оборот и используется как ссылка для определения абсолютного положения. Индексный импульс совпадает с фаза A и фаза B, обе имеют низкий уровень.

13.2 Режим 16-битного реверсивного счётчика положения

16-битный реверсивный счётчик считает вверх или вниз на каждый счётный импульс, который генерируется разницей входных сигналов фазы A и фазы B. Счётчик действует как интегратор, который считает значение пропорциональное положению. Направление счёта определяется сигналом UPDN, который генерируется логикой интерфейса квадратурного энкодера

13.2.1 Проверка ошибки счётчика положения

Проверка ошибки счёта в QEI предусмотрена для и указывается битом CNTERR (QEICON<15>). Проверка ошибки применяется только когда счётчик положения сконфигурирован для сброса на режимах индексного импульса (QEIM<2:0> = ‘110’ или ‘100’). В этих режимах, содержимое регистра POSCNT сравнивается с значениями (0xFFFF или MAXCNT + 1, в зависимости от направления). Если это значение определено, состояние ошибки генерируется установкой бита CNTERR и прерывание ошибки счёта QEI генерируется. Прерывание ошибки счёта QEI может быть запрещено установкой бита CEID (DFLTCON<8>). Счётчик положения продолжает считать фронты энкодера после того как ошибка была обнаружена. Регистр POSCNT продолжает считать вверх/вниз пока настоящий rollover/underflow. Для настоящего события rollover/underflow прерывание не генерируется. Бит CNTERR читается, записывается и сбрасывается программой пользователя.

13.2.2 Сброс счётчика положения

Бит разрешения сброса счётчика положения, POSRES (QEI<2>) управляет независимо счётчиком положения и сброшен, когда обнаружен индексный импульс. Этот бит применяется только когда QEIM<2:0> = ‘100’ или ‘110’.

Если бит POSRES установлен в ‘1’, тогда счётчик положения сбрасывается при обнаружении индексного импульса. Если бит POSRES установлен в ‘0’, тогда счётчик положения не сбрасывается при обнаружении индексного импульса. Счётчик положение продолжает счёт вверх или вниз, и будет сброшен состоянием переполнения(rollover) или потери значимости(underflow).

Когда выбранный сигнал INDX сбросит счётчик положения (POSCNT), пользователь определяет состояниями входных ножек QEA и QEB. Эти состояния должны соответствовать в порядке для происхождения сброса. Эти состояния выбираются битом IMV<1:0> в регистре DFLTCON <10:9>.

Бит IMV<1:0> (Соответствующее индексу значение) позволяет пользователю определить состояние входных ножек QEA и QEB в течение индексного импульса, когда регистр POSCNT будет сброшен.

В режиме 4X счёта квадратур:

IMV1 = требуемое состояние фазы B входного сигнала для соответствия индексному импульсу

IMV0 = Требуемое состояние фазы A входного сигнала для соответствия индексному импульсу

В режиме 2X счёта квадратур:

IMV1 = Выбранный входной сигнал фазы для соответствия состоянию индекса (0 = фаза A, 1 = фаза B)

IMV0 = Требуемое состояние выбранного входного фазового сигнала для соответствия индексному импульсу

Прерывание всё же генерируется на обнаружение индексного импульса и не на переполнение/(потерю значимости) счётчика положения.

13.2.3 Состояние направления счёта

Как упомянутая в предыдущей части, логика QEI генерирует сигнал UPDN, основанный на отношении между фазами A и B. В дополнение выходная ножка, состояние этого внутреннего сигнала UPDN снабжает бит SFR UPDN (QEICON<11>) как только для чтения бит.

Примечание:	Ножки QEI мультиплексируются с аналоговыми входами. Пользователь должен обеспечить, чтобы все связанные с QEI ножки были настроены как цифровые входы в регистре ADPCFG.

13.3 Режим измерения положения

Есть два режима измерения, которые поддерживаются и названы x2 и x4. Эти режимы выбираются битами QEIM<2:0> выбора режима, расположенными в SFR QEICON<10:8>.

Когда биты управления QEIM<2:0> = 100 или 101, выбран x2 режим измерения и логика QEI только смотрит на вход фазы A для темпа приращения счётчика положения. Каждый нарастающий и падающий фронт сигнала фазы A вызывает увеличение или уменьшение счётчика положения. Сигнал фазы B попрежнему используется для определения направления счётчика, так же как в режиме x4.

В пределах режима измерения x2, есть два варианта сброса счётчика положения:

1. Счётчик положения сбрасывается при обнаружении индексного импульса, QEIM<2:0> = 100.
2. Счётчик положения сбрасывается по совпадению с MAXCNT, QEIM<2:0> = 101.

Когда биты управления QEIM<2:0> = 110 или 111, выбран x4 режим измерения и логика QEI рассматривает оба фронта входных сигналов фазы A и B.

Каждый фронт обоих сигналов заставляет счётчик положения увеличиваться или уменьшаться.

В пределах режима измерения x4, есть два варианта сброса счётчика положения:

1. Счётчик положения сбрасывается при обнаружении индексного импульса, QEIM<2:0> = 110.
2. Счётчик положения сбрасывается по совпадению с MAXCNT, QEIM<2:0> = 111.

Режим измерения x4 обеспечивает большее разрешение данных (больше позиций счёта) для определения позиции ротора.

13.4 Программируемые цифровые фильтры шума

Секция цифрового фильтра шума отвечает за подавление шума поступающего захвата или квадратурных сигналов. Входы триггеров Шмидта и фильтр с трёхтактным циклом задержки объединены для отсеивания шума с низким и большим уровнями, коротких пиков, которые обычно происходят в приложениях склонных к шуму, таких как моторные системы.

Фильтр гарантирует, что отфильтрованный выходной сигнал не позволяет изменения пока стабильное значение регистрируется для трёх последовательных тактовых циклов.

Для ножек QEA, QEB и INDX, тактовый делитель частоты для цифрового фильтра запрограммирован битами QECK<2:0> (DFLTCON<6:4>) и происходит от базового цикла инструкции TCY.

Для разрешения выхода фильтра для каналов QEA, QEB и INDX, бит QEOUT должен быть установлен в ‘1’. Сеть фильтра для всех каналов выключается на POR и BOR.

13.5 Изменение 16-битного таймера/счётчика

Когда модуль QEI не сконфигурирован для QEI режима QEIM<2:0> = 001, модуль может быть сконфигурирован как простой 16-битный таймер/счётчик. Установка и управление вспомогательным таймером выполняется через регистр QEICON SFR. Эти таймерные функции идентичны Timer1. Ножка QEA используется как тактовый вход таймера.

Когда сконфигурирован как таймер, регистр POSCNT служит как счётный регистр таймера. Когда регистр таймер/период совпадение происходит, флаг прерывания QEI утверждается.

В отличие от универсальных таймеров, в этот таймер добавлена особенность внешнего входа выбора вверх/вниз. Когда ножка UPDN утверждена высоко, таймер инкрементируется вверх. Когда ножка UPDN утверждена вниз, таймер декрементируется.

Примечание:	Изменение режима работы (т.е., из QEI в таймер или наоборот), не влияет на содержимое регистра Таймер/Счётчик положения

Бит управление/состояния UPDN (QEICON<11>) может быть использован для выбора состояния направления счётчика регистра таймера. Когда UPDN = 1, таймер считает вверх. Когда UPDN = 0, таймер считает вниз.

Вдобавок, бит управления UPDN_SRC (QEICON<0>) определяет, основано ли состояние направления таймера на логическом состоянии, записанном в бит управления/состояния UPDN (QEICON<11>), или состоянии вывода QEB. Когда UPDN_SRC = 1, направление счёта таймера управляется от ножки QEB. Аналогично, когда UPDN_SRC = 0, направление счёта таймера управляется битом UPDN.

Примечание:	В этот таймер не встроена поддержка работы в режиме внешнего асинхронного счётчика. Если используется внешний тактовый источник, тактирование автоматически синхронизируется с внутренним циклом инструкции.

13.6 Работа модуля QEI в течении режима Спячки CPU

13.6.1 Работа в течении режима Спячки CPU

Модуль QEI остановлен в течении режима Спячки CPU.

13.6.2 Работа таймера в течении режима Спячки CPU

В течении режима Спячки CPU таймер не работает, потому что внутренний тактовый сигнал отключен.

13.7 Работа модуля QEI в течении режима Простоя CPU

Поскольку модуль QEI может функционировать как интерфейс квадратурного энкодера или как 16-битный таймер, в следующих частях описывается работа модуля в обоих режимах.

13.7.1 Работа QEI в течении режима Простоя CPU

Когда CPU установлен в режим Простоя, модуль QEI работает, если бит QEISIDL (QEICON<13>) = 0. Этот бит по умолчанию находится в логическом ‘0’ при выполнении POR и BOR. Для остановки модуля QEI в течении режима Простоя CPU, QEISIDL надо установить в ‘1’.

13.7.2 Работа таймера в течении режима Простоя CPU

Когда CPU установлен в режим Простоя и модуль QEI сконфигурирован в режим 16-битного таймера, 16-битный таймер может работать, если бит QEISIDL (QEICON<13>) = 0. Этот бит по умолчанию находится в логическом ‘0’ при выполнении POR и BOR. Для остановки модуля таймера в течении режима Простоя CPU, QEISIDL должен быть установлен в ‘1’.

Если бит QEISIDL очищен, таймер нормально функционирует, как будто режим Простоя CPU не был введён.

13.8 Прерывания интерфейса квадратурного энкодера

Интерфейс квадратурного энкодера имеет способность генерировать прерывание по случаю следующих событий:

- Прерывание на переполнение/(потеря значимости) 16-битного реверсивного счётчика положения
- Обнаружение пригодного индексного импульса или если бит CNTERR установлен
- Событие совпадения периода таймера (переполнение/потеря значимости)
- Событие управление накоплением

Бит флага прерывания QEI, QEIIF, утверждён на случай любого из вышеуказанных событий. Бит QEIIF должен быть очищен программно. QEIIF расположен в регистре статуса IFS2.

Разрешение прерывания выполняется через разрешение соответствующего бита, QEIIE. Бит QEIIE расположен в регистре управления

IEC2.

Таблица 13-1: Регистровая карта QEI

SFR Имя	Адрес	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
QEICON	0122	CNTERR	-	QEISIDL	INDX	UPDN	QEIM2	QEIM1	QEIM0	SWPAB	-	TQGATE	TQCKPS1	TQCKPS0	POSRES	TQCS	UPDN_SRC	0000 0000 0000 0000
DFLTCON	0124	-	-	-	-	-	IMV1	IMV0	CEID	QEOUT	QECK2	QECK1	QECK0	-	-	-	-	0000 0000 0000 0000
POSCNT	0126	Счётчик положения Counter<15:0>																0000 0000 0000 0000
MAXCNT	0128	Maximun Счётчик<15:0>																1111 1111 1111 1111

Описание: u = неинициализированный бит

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

14.0 Модуль PWM управления мотором

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

Этот модуль упрощает задачу генерации многочисленных, синхронизированных широтно-импульсно модулированных (PWM) выходов. В особенности, PWM модуль модуль поддерживает следующие энергетические и управления движением приложения:

- Трёхфазный асинхронный двигатель
- Вентильный двигатель (SR)
- Безщёточный двигатель постоянного тока (BLDC)
- Бесперебойный источник питания (UPS)

PWM модуль имеет следующие характеристики:

- 6 PWM ножек ввода/вывода с 3 генераторами заполнения цикла
- Разрешение до 16 бит
- ‘Непрерывное’ изменение частоты PWM
- Фронтально и центрально выровненные режимы выхода
- Режим генерации одиночного импульса
- Поддержку прерывания для несимметричной коррекции в центрально выровненном режиме
- Аннулирование выходного контроля для работы электрически коммутируемого двигателя (ECM)
- Компаратор ‘специального события’ для назначения других периферийных событий
- Ножками FLTA опциональное управление каждой из PWM выходных ножек в определённом состоянии

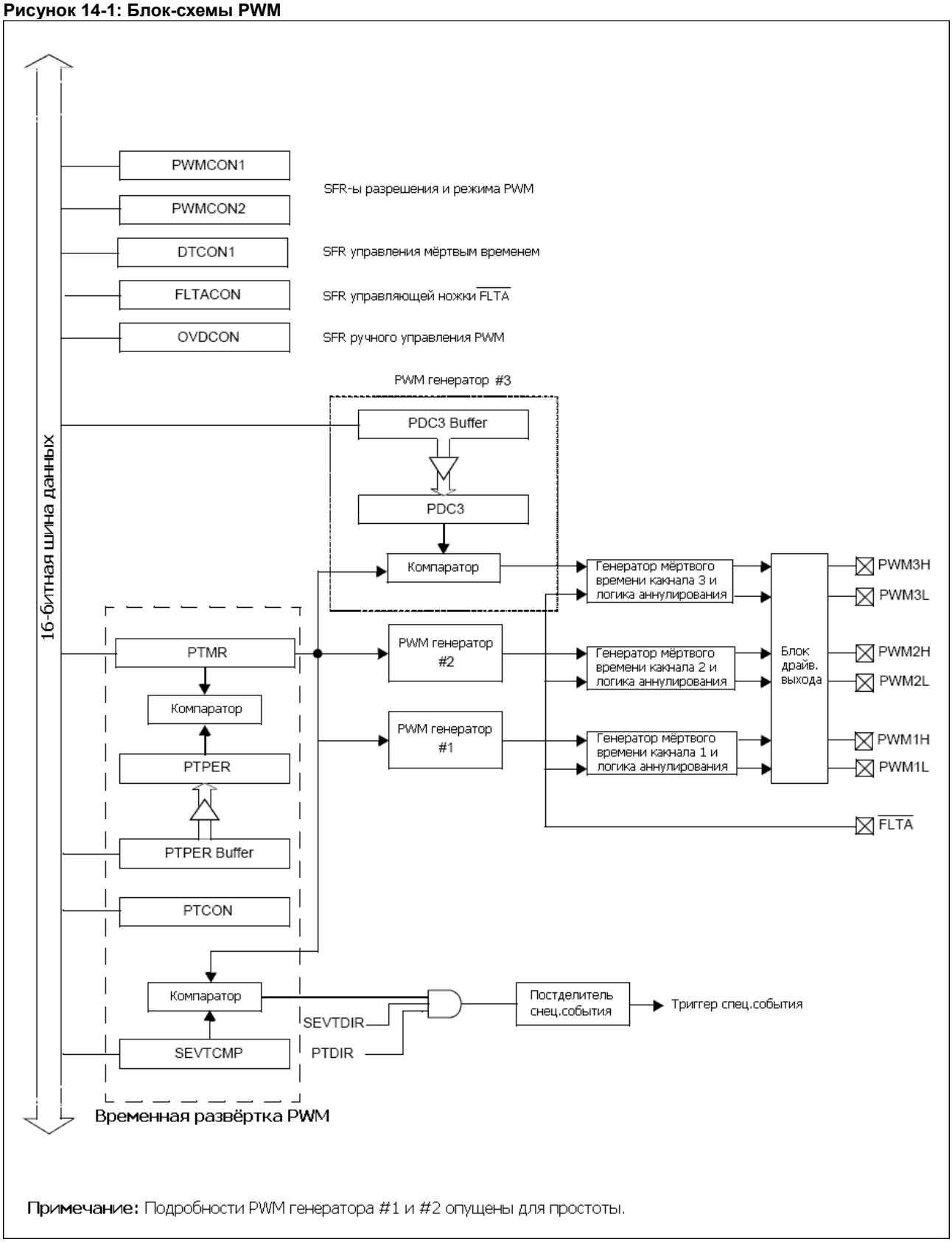
Этот модуль содержит 3 генератора заполнения цикла, нумеруемых от 1 до 3. Модуль имеет 6 выходных ножек PWM, нумеруемых от PWM1H/PWM1L до PWM3H/PWM3L.

Шесть ножек ввода/вывода сгруппированы высокий/низкий нумерованные пары, обозначаемые суффиксом H или L, соответственно. Для комплиментарных нагрузок, низкие PWM ножка всегда комплиментарна соответствующей высокой ножке ввода/вывода.

Упрощённая блок-схема модуля PWM управления двигателем показана на рисунке 14-1.

Модуль PWM позволяет несколько режимов работы, которые полезны для особых приложений управления мощностью.

Рисунок 14-1: Блок-схемы PWM



Примечание: Подробности PWM генератора #1 и #2 опущены для простоты.

14.1 Временная развёртка PWM

Временная развёртка PWM обеспечивается 15-битным таймером с предделителем и постделителем. Временная развёртка доступна через PTMR SFR. PTMR<15> есть только для чтения бит состояния, PTDIR, который показывает настоящее направление счёта временной развёртки PWM. Если PTDIR очищен, PTMR считает вверх. Если PTDIR установлен, PTMR считает вниз. Временная развёртка PWM конфигурируется через PTCON SFR. Временная развёртка разрешается/запрещается установкой/сбросом бита PTEN в PTCON SFR. PTMR не очищается, когда бит PTEN очищается программно.

PTPER SFR устанавливают период счёта для PTMR. Пользователь должен записать 15-битное значение в PTPER<14:0>. Когда значение в PTMR<14:0> соответствует значению в PTPER<14:0>, временная развёртка сбрасывает в ‘0’, или меняет направление счёта на следующем происходящем тактовом цикле. Действие принимается в зависимости от режима работы временной развёртки.

Примечание:	Если регистр периода установлен в 0x0000, таймер остановит счёт и прерывание и триггер специального события при этом не генерируются, даже если значение специального события также 0x0000. Модуль не обновляет регистр периода, если он уже 0x0000; следовательно, пользователь должен отключить модуль, чтобы модифицировать регистр периода.

ременная развёртка PWM может быть сконфигурирована для четырёх различных режимов работы:

- Режим свободного прогона
- Режим одиночного кадра
- Режим непрерывного счёта вверх/вниз
- Режим непрерывного счёта вверх/вниз с прерываниями для двойной коррекции

Эти четыре режима выбираются битами PTMOD<1:0> в PTCON SFR. Режимы счёта вверх/вниз поддерживают центрально-выровненную генерацию. Режим одиночного кадра позволяет модулю PWM поддерживать управление импульсом определённых электронно-коммутируемых моторов (ECMs).

Сигнал прерывания генерируется временной развёрткой PWM в зависимости от битов выбора режима (PTMOD<1:0>) и битов постделителя (PTOPS<3:0>) в PTCON SFR.

14.1.1 Режим свободного прогона

В режиме свободного прогона, временная развёртка PWM считает вверх пока значение в регистре периода (PTPER) временной развёртки будет соответствовать. Регистр PTMR сбрасывается на следующий входной тактовый фронт и временная развёртка продолжает считать вверх при условии если бит PTEN остаётся установленным.

Когда временная развёртка PWM находится в режиме свободного прогона (PTMOD<1:0> = 00), событие прерывания генерируется каждый раз когда происходит соответствия с регистром PTPER и регистр PTMR сбрасывается в ноль. Биты выбора постделителя могут быть использованы в этом режиме таймера, чтобы уменьшить частоту событий прерывания.

14.1.2 Режим одиночного кадра

В режиме счёта одиночного кадра, временная развёртка PWM начинает счёт вверх, когда бит PTEN установлен. Когда значение в регистре PTMR соответствует регистру PTPER, регистр PTMR сбрасывается на следующий входной тактовый фронт и бит PTEN очищается аппаратно для остановки временной развёртки.

Когда временная развёртка PWM находится в режиме одиночного кадра (PTMOD<1:0> = 01), событие прерывания генерируется, когда происходит соответствие с регистром PTPER, регистр PTMR сбрасывается в ноль на следующий входной тактовый фронт, и бит PTEN очищается. Биты выбора постделителя не имеют эффекта в этом режиме таймера.

14.1.3 Режимы непрерывного счёта вверх/вниз

В режимах непрерывного счёта вверх/вниз, временная развёртка PWM считает вверх, пока значение в регистре PTPER соответствует. Таймер начнёт считать вниз на следующий входной тактовый фронт. Бит PTDIR в PTCON SFR только для чтения и показывает направление счёта. Бит PTDIR установлен, когда таймер считает вниз.

В режиме счёта верх/вниз (PTMOD<1:0> = 10), событие прерывания генерируется каждый раз, когда значение регистра PTMR становится нулевым и временная развёртка PWM начинает считать вверх. Биты выбора постделителя могут быть использованы в этом режиме таймера для уменьшения частоты событий прерываний.

14.1.4 Режим двойной коррекции

В режиме двойной коррекции (PTMOD<1:0> = 11), событие прерывания генерируется каждый раз, как регистр PTMR равен нулю, а так же каждый раз, как соответствие периода происходит. Биты выбора постделителя не имеют эффекта в этом режиме таймера.

Режим двойной коррекции обеспечивает две дополнительные функции пользователю. Во-первых, ширина полосы частот цикла управления удвоена, потому что заполнение цикла PWM может быть скорректировано, дважды за период. Во-вторых, может генерироваться несимметричная центрально-выровненная форма PWM, которая быть полезной для минимизации искажений выходной формы волны в определённых приложениях управления двигателем.

Примечание:	Программирование значения 0x0001 в регистре периода может генерировать продолжительный импульс прерывания, и следовательно, должно избегаться.

14.1.5 Предделитель временной развёртки PWM

Входное тактирование в PTMR (FOSC/4), имеет опции предделителя 1:1, 1:4, 1:16 или 1:64, выбираемые битами управления PTCKPS<1:0> в PTCON SFR. Счётчик предделителя очищается, когда любое из следующего происходит:

• запись в регистр PTMR
• запись в регистр PTCON
• любой сброс устройства

Регистр PTMR не очищается, когда записан PTCON.

14.1.6 Постделитель временной развёртки PWM

Соответствующий выход PTMR может быть опционально постделён 4-бытным постделителем (который даёт деление от 1:1 до 1:16).

Счётчик постделителя очищен, когда любое из следующего происходит:

• запись в регистр PTMR
• запись в регистр PTCON
• любой сброс устройства

Регистр PTMR не очищается, когда записан PTCON.

14.2 Период PWM

PTPER является 15-битным регистром и используется для установки периода счёта для временной развёртки PWM. PTPER является регистром с двойным буферированием. Содержимое буфера PTPER загружается в регистр PTPER в следующих примерах:

• Режим свободного прогона и режим одиночного кадра: Когда регистр PTMR сброшен в ноль после соответствия с регистром PTPER.

• Режимы счёта вверх/вниз: Когда регистр PTMR равен нулю.

Значение, удерживаемое в буфере PTPER автоматически загружается в регистр PTPER когда временная развёртка PWM блокирована (PTEN = 0).

Период PWM можно определить, используя формулу 14-1:

Формула 14-1: Период PWM

$$T_{PWM} = \frac{T_{CY} \bullet (PTPER + 1)}{(PTMR\_значение\_предделителя)}$$

Если временная развёртка PWM сконфигурирована для одного из режимов счёта вверх/вниз, период PWM можно найти, используя формулу 14-2.

Формула 14-2: Период PWM (режим счёта вверх/вниз)

$$T_{PWM} = \frac{T_{CY} \bullet 2 \bullet (PTPER + 0.75)}{(PTMR\_значение\_предделителя)}$$

Максимальное разрешение (в битах) для генератора данного устройства и частоты PWM можно определить, используя формулу 14-3:

Формула 14-3: Разрешение PWM

$$Разрешение = \frac{\log(2 \bullet T_{PWM} / T_{CY})}{\log(2)}$$

14.3 Фронтально выровненный PWM

Фронтально-выровненные PWM сигналы производятся модулем, временная развёртка которого работает в режиме свободного прогона или режиме одиночного кадра. Для фронтально-выровненных PWM выходов, выход имеет период определённый значением в PTPER и заполнение цикла в соответствующем регистре заполнения цикла (смотреть рисунок 14-2). Выход PWM приведён в активное состояние в начале периода (PTMR = 0) и приведён в неактивное состояние, когда значение регистра заполнения цикла соответствует PTMR.

Если значение в особенном регистре заполнения цикла нулевое, то выход на соответствующей ножке PWM будет неактивным для целого периода PWM. Кроме того, выход на ножке PWM может быть активным для целого периода PWM если значение в регистре заполнения цикла больше чем значение, удерживаемое в регистре PTPER.

Рисунок 14-2: Фронтально-выровненный PWM

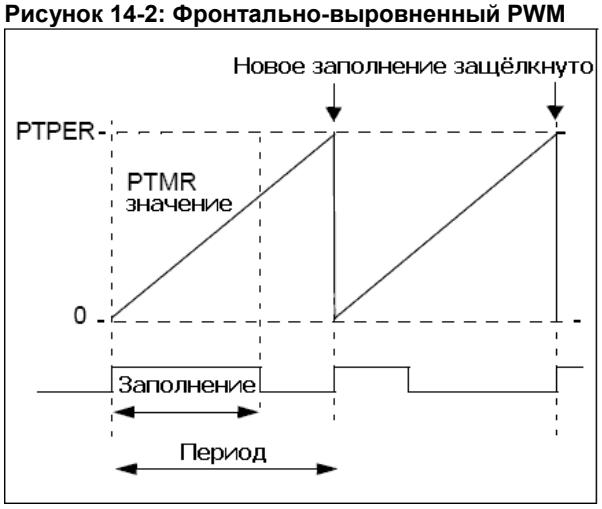
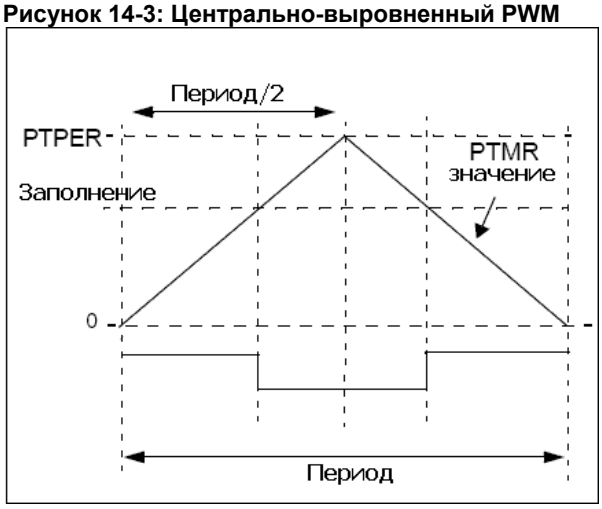


Рисунок 14-3: Центрально-выровненный PWM



14.4 Центрально-выровненный PWM

Центрально-выровненный PWM сигналы производятся модулем при временной развёртке PWM сконфигурированной в режиме счёта вверх/вниз (смотреть рисунок 14-3).

PWM сравниваются, выход приводится к активному состоянию, когда значение регистра заполнения цикла соответствует значению PTMR и временная развёртка PWM считается вниз (PTDIR = 1). PWM сравниваются, выход приводится в неактивное состояние, когда временная развёртка считается вверх (PTDIR = 0) и значение регистра PTMR соответствует заполнению цикла.

Если значение в особенном регистре заполнения цикла нулевое, тогда выход на соответствующей ножке PWM будет неактивным для целого периода PWM. Кроме того, выход на ножке PWM будет активным для целого периода PWM, если значение в регистре заполнения цикла равно значению, удерживаемому в регистре PTPER.

14.5 Устройства сравнения заполнения цикла PWM

Есть четыре 16-битных специальных регистра (PDC1, PDC2, PDC3 и PDC4) используемые для определения значений заполнения цикла для модуля PWM.

Значение в каждом регистре заполнения цикла определяет объём времени которое выход PWM находился в активном состоянии. Регистры заполнения цикла имеют ширину 16 бит. LSb регистра заполнения цикла независимо определяет фронт PWM происходит в начале. Таким образом, разрешение PWM эффективно удвоено.

14.5.1 Буферы регистра заполнения цикла

Четыре регистра заполнения цикла PWM имеют двойное буфферирование позволяющие коррекцию понижающую помехи на выходах PWM. Для каждого цикла заполнения, есть регистр цикла заполнения который доступен пользователю и второй регистр заполнения цикла который удерживает актуальное сравниваемое значение используемое в обеспечении периода PWM.

Для фронтально-выровненного выхода PWM, новое значение заполнения цикла будет обновляться каждый раз, когда произойдёт совпадение с регистром PTPER и PTMR будет сброшен. Содержимое буферов заполнения цикла будет автоматически загружено в регистр заполнения цикла, когда временная развёртка PWM блокирована (PTEN = 0) и бит UDIS очищен в PWMCON2.

Когда временная развёртка PWM находится в режиме счёта вверх/вниз, новые значения заполнения цикла модифицируются, когда значение регистра PTMR нулевое и временная развёртка PWM начинает считать вверх. Содержимое буферов заполнения цикла автоматически загружается в регистры цикла заполнения, когда временная развёртка PWM заблокирована (PTEN = 0).

Когда временная развёртка PWM в режиме счёта вверх/вниз с двойной коррекцией, новые значения заполнения цикла модифицируются, когда значение регистра PTMR нулевое, и когда значение регистра PTMR совпадает с значением в регистре PTPER. Содержимое буферов заполнения цикла автоматически загружается в регистры заполнения цикла, когда временная развёртка PWM блокирована (PTEN = 0).

14.6 Комплиментарная работа PWM

В комплиментарном режиме работы каждая пара выходов PWM получает комплиментарные сигналы PWM. Мёртвое время может опционально включатся между переключениями устройства, когда оба выхода неактивны в течении короткого периода (обратитесь к части 14.7 “Генераторы мёртвого времени”).

В комплиментарном режиме, устройства сравнения заполнение цикла назначены на выходы PWM следующим образом:

- PDC1 регистр управляет выходами PWM1H/PWM1L
- PDC2 регистр управляет выходами PWM2H/PWM2L
- PDC3 регистр управляет выходами PWM3H/PWM3L

Комплиментарный режим выбирается для каждой пары PWM ножек ввода/вывода очисткой соответствующего бита PMODx в PWMCON1 SFR. PWM ножки ввода/вывода установлены по умолчанию в комплиментарный режим после сброса устройства.

14.7 Генераторы мёртвого времени

Генератор мёртвого времени могут быть предусмотрены, когда любая пара PWM ножек ввода/вывода работает в комплиментарном режиме выхода. Выходы PWM используют двухтактные схемы драйвера. Из-за невозможности мощных выходных устройств переключатся мгновенно, необходимо обеспечить некоторое время между моментом выключения одного выхода PWM в комплиментарной паре и моментом включения другого транзистора.

14.7.1 Генераторы мёртвого времени

Каждая комплиментарная пара выходов модуля PWM иметь 6-битный вычитающий счётчик, который используется для создания вставки мёртвого времени. Как показано на рисунке 14-4, устройство мёртвого времени имеет детектор нарастающего и падающего фронта подключенный к выходу сравнения заполнения цикла.

14.7.2 Диапазон мёртвого времени

Объём мёртвого времени обеспечиваемый устройством мёртвого времени выбирается определением величиной входного тактирования предделителя и 6-битным беззнаковым значением.

Четыре входных тактирования предделителя выбор обеспечен, чтобы позволить подходящий диапазон мёртвого времени, основанный на рабочей частоте устройства. Значение тактирования предделителя мёртвого времени выбирается с использованием DTAPS<1:0> и

DTBPS<1:0> битов управления в DTCON1 SFR. Одна из четырёх тактовых опций предделителя (TCY, 2TCY, 4TCY или 8TCY) выбирается для значения мёртвого времени.

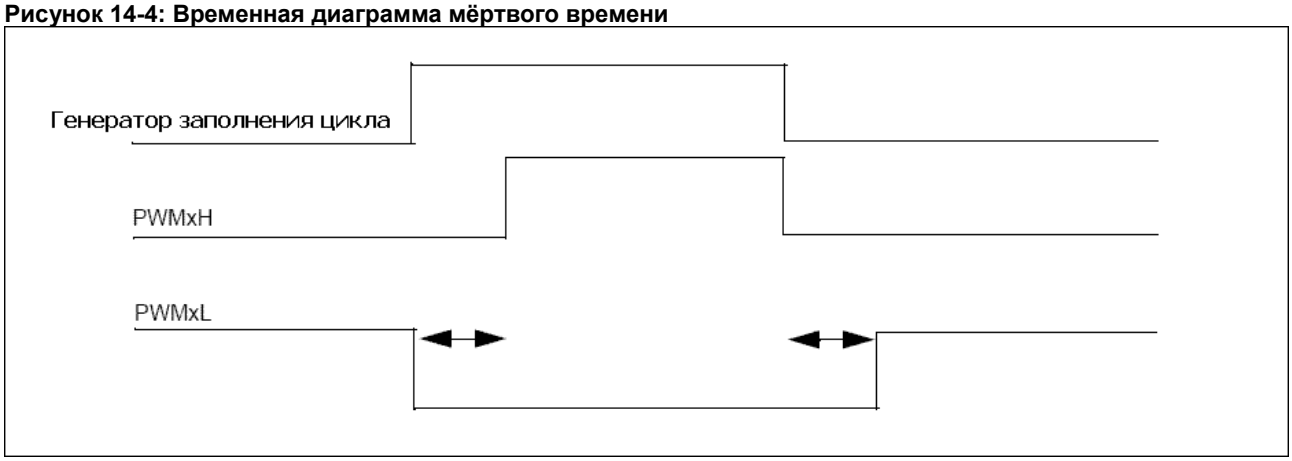
После того как значение предделителя выбрано, мёртвое время настраивается загрузкой 6-битного беззнакового значения в DTCON1 SFR.

Предделитель устройства мёртвого времени очищается по следующим событиям:

- На загрузку вниз таймера благодаря событию фронта сравнения цикла заполнения.
- На запись в регистр DTCON1.
- На любой сброс устройства.

Примечание:	Пользователь не должен изменять значение DTCON1 пока модуль PWM работает (PTEN = 1). Иначе можно получить неожиданный результат.

Рисунок 14-4: Временная диаграмма мёртвого времени



14.8 Независимый выход PWM

Режим независимого выхода PWM требуется для управления определённым типом нагрузок. В частности выходная пара PWM находится в независимом режиме вывода, когда соответствующий бит PMOD в регистре PWMCON1 установлен. Ни какой контроль мёртвого времени не осуществляется между смежными ножками ввода/вывода PWM, когда модуль работает в независимом режиме и обе ножки ввода/вывода могут быть активными одновременно.

В независимом режиме, каждый генератор заполнения цикла подключен к обоим PWM ножкам ввода/вывода в выходной паре. Используя связанный регистр заполнения цикла и соответствующие биты в регистре OVDCON, пользователь может выбрать следующие опции выходного сигнала для работы каждой PWM ножки ввода/вывода в независимом режиме:

• Ножка ввода/вывода выводит сигнал PWM
• Ножка ввода/вывода неактивная
• Ножка ввода/вывода активная

14.9 Операция одиночного импульса PWM

Модуль PWM производит на выводах одиночный импульс, когда биты управления PTCON PTMOD<1:0> = 10. Только фронтальновыровненные выходы могут производить в режиме одиночного импульса. В режиме одиночного импульса, PWM ножка(и) ввода/вывода приводится в активное состояние, когда бит PTEN установлен. Когда происходит соответствие с регистром заполнения цикла, PWM ножка ввода/вывода приводится в неактивное состояние. Когда происходит соответствие с регистром PTPER, регистр PTMR очищается, все активные PWM ножки ввода/вывода приводятся в неактивное состояние, бит PTEN очищается, и генерируется прерывание.

14.10 Отмена выхода PWM

Биты отмены выхода PWM позволяют пользователю вручную привести PWM ножки ввода/вывода в определённое логическое состояние, независимо от устройств сравнения заполнения цикла.

Все биты управления связанные с отменной функцией выходов PWM расположены в регистре OVDCON. Верхняя половина регистра OVDCON содержит шесть бит, POVDxH<3:1> и POVDxL<3:1>, определяющих какие PWM ножки ввода/вывода будут отменены. Нижняя половина регистра OVDCON содержит шесть бит, POUTxH<3:1> и POUTxL<3:1>, определяющих состояние PWM ножек ввода/вывода, когда конкретный выход будет отменён через биты POVD.

14.10.1 Режим комплиментарного выхода

Когда ножка PWMxL делается активной через регистр OVDCON, выходной сигнал вынуждается быть комплиментарным сигналу на соответствующей ножке PWMxH в паре. Вставка мёртвого времени всё же выполняется, когда PWM каналы отменены вручную.

14.10.2 Отмена синхронизации

Если бит OSYNC в регистре PWMCON2 установлен, все отмены выхода, выполненные через регистр OVDCON, синхронизированы с временной развёрткой PWM. Синхронные отмены вывода происходят в следующие моменты:

• В фронтально-выровненном режиме, когда PTMR нулевой.
• В центрально-выровненных режимах, когда PTMR нулевой и когда значения PTMR соответствуют PTPER.

14.11 Управление выходом и полярностью PWM

Имеются три бита конфигурации устройства, связанные с модулем PWM, которые обеспечивают управление выходной ножкой PWM:

• Бит конфигурации HPOL
• Бит конфигурации LPOL
• Бит конфигурации PWMPIN

Эти три бита в регистре конфигурации FPORBOR (смотреть часть 21) работают в сочетании с тремя битами разрешения PWM (PWMEN<3:1>) расположенными в PWMCON1 SFR. Биты конфигурации и биты разрешения PWM гарантируют, что ножки PWM будут в корректных состояниях после того как происходит сброс устройства. Предохранение конфигурации PWMPIN позволяет выходам модуля PWM быть опционально разрешёнными на сброс устройства. Если PWMPIN = 0, выходы PWM будут приведены в их неактивные состояния при сбросе. Если PWMPIN = 1 (по умолчанию), выходы PWM будут в третьем состоянии. Бит HPOL определяет полярность для выходов PWMxH, поскольку бит LPOL определяет полярность для выходов PWMxL.

14.11.1 Управление выходной ножкой

Биты управления PEN<3:1>H и PEN<3:1>L в PWMCON1 SFR разрешают каждую ножку высокого выхода PWM и каждую ножку низкого выхода PWM, соответственно. Если конкретная выходная ножка для PWM не разрешена, то используется как ножка универсального ввода/вывода.

14.12 Ножки PWM FLTA

Есть одна FLTA ножка (FLTA) связанная с модулем PWM. Когда утверждено, эта ножка может произвольно управлять каждой PWM ножкой ввода/вывода в определённое состояние.

14.12.1 Биты разрешения дефектной ножки

FLTACON SFR имеет 4 бита управления, которые независимо определяют какая конкретная пара PWM ножек ввода/вывода управляется ножкой ввода FLTA. Чтобы разрешить определённую PWM пару ножек ввода/вывода для отмены FLTA, соответствующий бит должен быть установлен в регистре FLTACON. Если все биты разрешения были очищены в регистре FLTACON, то входная ножка FLTA не оказывает эффекта на модуль PWM и ножка может быть использована как ножка универсального ввода/вывода или прерывания.

Примечание:	Логика ножки FLTA может работать независимо от логики PWM. Если все биты разрешения в регистре FLTACON очищены, то ножка(и) FLTA может быть использована как универсальная ножка(и) прерывания. Каждая ножка FLTA имеет вектор прерывания, бит флага прерывания и биты приоритета прерывания связанные с этим.

14.12.2 Состояния дефекта

Специальный функциональный регистр FLTACON имеет 8 бит, которые определяют состояние каждой PWM ножки ввода/вывода, когда это отменено входом. Когда эти биты очищены, PWM ножка ввода/вывода приводится в неактивное состояние. Если бит установлен, PWM ножка ввода/вывода приводится в активное состояние. Активное и неактивное состояния упомянуты определённой полярностью для каждой PWM ножки ввода/вывода (биты управления полярностью HPOL и LPOL).

14.12.3 Режимы дефектного входа

Входная ножка FLTA имеет два режима работы:

• Запираемый режим: Когда ножка FLTA приведена в низ, выходы PWM переходят в состояние определённое в регистре FLTACON. Выходы PWM остаются в этом состоянии пока ножка FLTA будет переведена вверх и соответствующий флаг прерывания будет очищен программно. Когда обе эти действия произойдут, выходы PWM будут возвращены в нормальную работу в начале следующего цикла PWM или на границе середины цикла. Если флаг прерывания был очищен перед состоянием окончания FLTA, модуль PWM будет ожидать пока ножка FLTA будет недолго утверждённой, чтобы восстановить выходы.

• Поцикловый режим: Когда входная ножка FLTA приведена вниз, выходы PWM остаются в определённых FLTA состояниях для так долго, как ножка FLTA будет удерживаться низкой. После того как ножка FLTA приведена вверх, выходы PWM возвращаются к нормальной работе в начале следующего цикла PWM или на границе половины цикла.

Режим работы для входной ножки FLTA выбирается с использованием бита управления FLTAM в специально функциональном регистре FLTACON.

Ножкой FLTA можно управлять вручную в программе.

14.13 Обновление блокировки PWM

Для комплексных приложений PWM, пользователь может нуждаться в записи до четырёх регистров цикла заполнения и регистра временной развёртки периода, PTPER, в данное время. В нескольких приложениях, это важно чтобы все буферные регистры были записаны перед новым заполнением цикла и значения периода загружены для использования модулем.

Обновление блокировки PWM особенность разрешается установкой бита управления UDIS в PWMCON2 SFR. Бит UDIS влияет на все регистры буфера заполнения цикла и буфер временной развёртки периода PWM, PTPER. Ни какие изменения заполнения цикла или значения периода не производят эффекта пока UDIS = 1.

14.14 Специальный триггер события PWM

Модуль PWM имеет специальный триггер события, который позволяет A/D преобразования синхронизировать с временной развёрткой PWM. Выборка A/D и время преобразования могут быть запрограммированы происходить в любой момент в течении периода PWM. Специальный триггер события позволяет использовать минимальную задержку между временем, когда приобретается результат преобразования A/D, и временем когда обновлено значение цикла заполнения.

Специальный триггер события PWM имеет SFR имя SEVTCMP, и пять служебных битов управляют его работой. Значение PTMR для которого специальный триггер события должен произойти загружено в регистр SEVTCMP. Когда временная развёртка PWM в режиме счёта вверх/вниз, дополнительный бит управления необходим для определения фазы счёта для специального триггера события. Фаза счёта выбирается с использованием бита управления SEVTDIR в SEVTCMP SFR. Если бит SEVTDIR очищен, специальный триггер события произойдёт при счёте вверх цикла временной развёртки PWM. Если бит SEVTDIR установлен, специальный триггер события произойдёт при счёте вниз цикла временной развёртки PWM. Бит управления SEVTDIR не оказывает эффекта, если временная развёртка PWM не сконфигурирована для режима счёта вверх/вниз.

14.14.1 Постделитель специального триггера событий

Специальный триггер события PWM имеет постделитель, который позволяет диапазон постделения от 1:1 до 1:16. Постделитель конфигурируется записью битов управления SEVOPS<3:0> в .

Выход постделителя специального события очищен на следующие события:

- Любая запись в регистр SEVTCMP
- Любой сброс устройства

14.15 Работа PWM в течении режима Спячки CPU

Входные ножки FLTA A и FLTA B имеют способность пробуждать CPU из режима Спячки. Модуль PWM генерирует прерывание, если любая ножка FLTA будет приведена вниз в течении Спячки.

14.16 Работа PWM в течении режима Простоя CPU

PTCON SFR содержит бит управления PTSIDL. Этот бит определяет, если модуль PWM продолжает работать или остановлен как только устройство войдёт в режим Простоя. Если PTSIDL = 0, модуль продолжает работать. Если PTSIDL = 1, модуль останавливает работу до тех пор, пока CPU остаётся в режиме Простоя.

Таблица 14-1: Регистровая карта PWM

SFR Имя	Адрес	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
PTCON	01C0	PTEN	-	PTSIDL	-	-	-	-	-	PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>		0000 0000 0000 0000
PTMR	01C2	PTDIR	Значение счётчика таймера PWM															0000 0000 0000 0000
PTPER	01C4	-	Регистр временной развёртки PWM															0011 1111 1111 1111
SEVTCMP	01C6	SEVTDIR	Регистр специального события сравнения PWM															0000 0000 0000 0000
PWMCON1	01C8	-	-	-	-	-	PTMOD3	PTMOD2	PTMOD1	-	PEN3H	PEN2H	PEN1H	-	PEN3L	PEN2L	PEN1L	0000 0000 0111 0111
PWMCON2	01CA	-	-	-	-	SEVOPS<3:0>				-	-	-	-	-	-	OSYNC	UDIS	0000 0000 0000 0000
DTCON1	01CC	DTBPS<1:0>		DTB<5:0>						DTAPS<1:0>		Dead Time A Value						0000 0000 0000 0000
FLTACON	01D0	-	-	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L	FLTAM	-	-	-	-	FAEN3	FAEN2	FAEN1	0000 0000 0000 0000
OVDCON	01D4	-	-	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L	-	-	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L	0011 1111 0000 0000
PDC1	01D6	Регистр заполнения цикла PWM #1																0000 0000 0000 0000
PDC2	01D8	Регистр заполнения цикла PWM #2																0000 0000 0000 0000
PDC3	01DA	Регистр заполнения цикла PWM #3																0000 0000 0000 0000

Описание: u = неинициализированный бит

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

18.0 Модуль 10-битного высокоскоростного аналогово-цифрового преобразователя (ADC)

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

10-битный высокоскоростной аналогово-цифровой преобразователь (ADC) позволяет преобразовать аналоговый входной сигнал в 10-битный цифровой код. Работа этого модуля основана на архитектуре регистра последовательной аппроксимации (SAR), и обеспечивает максимальный темп выборок в 500 ksps.

Модуль ADC имеет до 16 аналоговых входов, которые мультиплексированы на четыре усилителя выборки хранения. Выход усилителя выборки хранения подаётся на вход преобразователя, который генерирует результат. В качестве аналогового опорного напряжения программно можно выбрать любое из питающих устройство напряжений (AVDD/AVSS) или уровень напряжения на ножке (VREF+/VREF-). ADC имеет уникальную способность быть в состоянии работать, пока устройство находится в режиме спячки.

Модуль ADC имеет шесть 16-битных регистров:

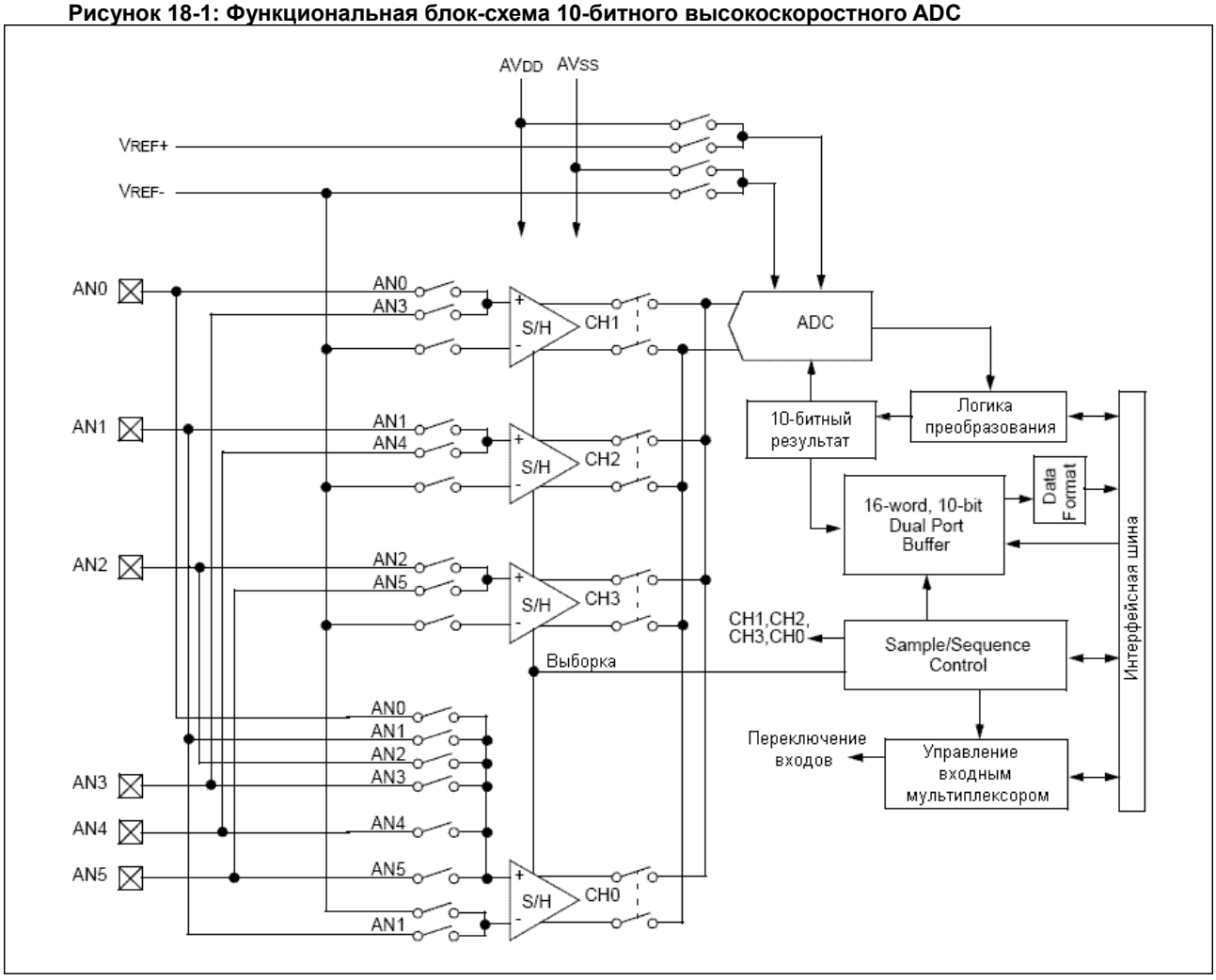
- A/D Регистр1 управления (ADCON1)
- A/D Регистр2 управления (ADCON2)
- A/D Регистр3 управления (ADCON3)
- A/D Регистр выбора входа (ADCHS)
- A/D Регистр конфигурации порта (ADPCFG)
- A/D Регистр выбора сканируемого входа (ADCSSL)

Регистры управления ADCON1, ADCON2 и ADCON3 управляют работой модуля ADC. Регистр ADCHS выбирает какие входные каналы будут преобразованы. Регистр ADPCFG конфигурирует ножки порта как аналоговые выводы или как цифровой ввод/вывод. Регистр ADCSSL выбирает входы для сканирования.

Примечание:	Биты SSRC<2:0>, ASAM, SIMSAM, SMPI<3:0>, BUFM и ALTS bits, так же как регистры ADCON3 и ADCSSL, не могут быть записаны пока ADON = 1. Это может приводить к неопределённым результатам.

Блок-схема модуля ADC показана на рисунке 18-1.

Рисунок 18-1: Функциональная блок-схема 10-битного высокоскоростного ADC



18.1 Буфер результата A/D

Модуль содержит 16-словный двух портовый только читаемый буфер, под названием ADCBUF0...ADCBUFF, буферирующий результаты ADC. RAM шириной 10 бит, но читается в различном формате 16-битных слов. Буферные регистры, от ADCBUF0 до ADCBUFFThe, содержащие шестнадцать результатов преобразования ADC, не могут быть записаны из программы пользователя.

18.2 Операция преобразования

После того, как ADC модуль сконфигурирован, приобретение выборки стартует после установки бита SAMP. Различные источники, такие как программируемый бит, таймаут таймера и внешние события, завершат приобретение и запустят преобразование. Когда A/D преобразование завершится, результат будет загружен в ADCBUF0...ADCBUFF, и флаг ADIF прерывания и бит DONE будут установлены после числа выборок определённых битом SMPI. Следующие шаги должны быть сделаны для A/D преобразования:

1. Конфигурирование модуля ADC:
- Сконфигурировать аналоговые ножки, источник опорного напряжения и цифровой ввод/вывод
- Выбрать входные каналы A/D
- Выбрать тактирование A/D преобразования
- Выбрать триггер A/D преобразования
- Включить модуль A/D
2. Конфигурирование прерывания A/D (если необходимо):
- Очистить бит ADIF bit
- Выбрать приоритет прерывания A/D
3. Старт выборки.
4. Выждать время, требуемое для выборки.
5. Триггер окончания выборки, старта преобразования
6. Ожидать одного из двух событий завершения A/D преобразования:
- Ожидание прерывания A/D
- Ожидание установки бита DONE
7. Читать буфер результата A/D, очистить ADIF, если требуется.

18.3 Последовательность выбора и преобразования

Различные группы битов управления выбирают последовательность, в которой A/D подключает входы к каналам выборки/хранения, каналам преобразования, записывает буфер памяти, и генерирует прерывания. Последовательность управляется временами выборки.

Бит SIMSAM управляет последовательностью приобретение/преобразование многих каналов. Если бит SIMSAM равен ‘0’, два или четыре выбранных канала приобретают и конвертируют последовательно, с двумя или четырьмя временами выборки. Если бит SIMSAM равен ‘1’, два или четыре выбранных канала приобретаются одновременно, с одним временем выборки. Каналы затем конвертируются последовательно. Очевидно, если только 1 канал выбран, бит SIMSAM не применим.

Биты CHPS определяют, сколько каналов производят выборку. Это может измениться от 1, 2 или 4 каналов. Если CHPS выбирает 1 канал, канал CH0 будет производит выборку в время выборки и конвертирован. Результат будет запомнен в буфере. Если выбраны 2 канала, каналы CH0 и CH1 будут выбраны и конвертированы. Если CHPS выбрал 4 канала, то каналы CH0, CH1, CH2 и CH3 будут выбраны и конвертированы.

Биты SMPI выбирают число приобретение/выборка последовательностей, которые должны быть выполнены перед тем как произойдёт прерывание. Это может меняться от 1 выборки на прерывание до 16 выборок на прерывание.

Пользователь не может программировать комбинацию CHPS и SMPI бит, которая определяет более 16 преобразований на прерывание, или 8 преобразований на прерывание, в зависимости от бита BUFM. Бит BUFM, когда установлен, разделяет 16-словный буфер результата (ADCBUF0...ADCBUFF) на две 8-словных группы. Запись 8-словных буферов будет чередоваться на каждое событие прерывания. Использование бита BUFM зависит от того, сколько времени является доступным для перемещения данных вне буферов после прерывания, как определено прикладной программой.

Если процессор может быстро разгрузить полный буфер в пределах времени, требуемого чтобы приобрести и преобразовать один канал, бит BUFM может быть ‘0’, и до 16 преобразований может быть сделано в прерывание. Процессор будет иметь одно выборки и преобразования время, чтобы переместить эти шестнадцать преобразований.

Если процессор не может разгрузить буфер в пределах времени выборки и преобразования, бит BUFM должен быть ‘1’. Для примера, если SMPI<3:0> (ADCON2<5:2>) = 0111, тогда восемь преобразований могут быть загружены в 1/2 буфера, после которого произойдёт прерывание. Следующие восемь преобразований будут загружены в другую 1/2 буфера. Процессор может иметь целое время между прерываниями, чтобы переместить восемь преобразований.

ALTS бит может использоваться, чтобы чередовать вводы, выбранные в течение выборочной последовательности. Входной мультиплексор имеет два набора вводов выборки: MUX A и MUX B. Если ALTS бит - ‘0’, только MUX A входы выбраны для осуществления выборки. Если ALTS бит ‘1’ и SMPI<3:0> = 0000, на первой последовательности выборка/преобразование MUX A входы выбраны, и на следующей последовательности выборка/преобразование, выбраны входы MUX B.

Бит CSCNA (ADCON2 < 10 >) позволит входам канала CH0 быть поочередно отсканированным поперек выбранного числа аналоговых входных сигналов для MUX группы. Входы выбраны регистром ADCSSL. Если конкретный бит в регистре ADCSSL ‘1’, соответствующий вход выбран. Входы всегда сканируются от более низкого до высокого пронумерованного входа, стартуя после каждого прерывания. Если число выбранных входов больший чем число выборок, принятых за прерывание, выше пронумерованные входы не используются.

18.4 Программирование триггера запуска преобразования

Триггер преобразования завершает приобретение выборки и стартует запрос преобразований.

Биты SSRC<2:0> выбирают источник триггера преобразования.

Биты SSRC обеспечивают до 5 альтернативных источников триггера преобразования.

Когда SSRC<2:0> = 000, триггер преобразования по программному управлению. Очистка бита SAMP вызывает триггер преобразования

Когда SSRC<2:0> = 111 (режим авто-старта), триггер преобразования подчиняется A/D управлению тактирования. Биты SAMC выбирают число тактов A/D между стартом приобретения выборки и стартом преобразования. Это обеспечивает наиболее быстрый темп преобразования на многих каналах. SAMC должен всегда быть по крайней мере 1 тактовый цикл.

Другие триггерные источники могут приходить из модулей таймера, модуля PWM управления двигателем или внешние прерывания.

Примечание:	При Работе A/D с максимальной скоростью преобразования, опция триггера авто-преобразования должна быть выбрана (SSRC = 111) и биты времени авто-выборки должны быть установлены в 1 TAD (SAMC = 00001). Эта конфигурация даёт общий период преобразования (выборка + преобразование) 13 TAD. Использование любых других триггеров преобразования приводят к дополнительным TAD циклам для синхронизации внешних событий с A/D.

18.5 Прерывание преобразования

Очистка бита ADON в течении преобразования прервёт текущее преобразование и остановит последовательность выборки. ADCBUF не будет обновлён частично завершённым A/D преобразованием выборки. То есть, ADCBUF продолжит содержать значение прошлой завершённым преобразованием (или прошлым значением записанным в ADCBUF регистр).

Если очистка бита ADON совпадает с авто-стартом, очистка имеет высший приоритет.

После того, как A/D преобразование прервано, требуется ждать 2 TAD перед тем как следующая выборка стартует через установку бита SAMP.

Если последовательная выборка определена, A/D продолжает следующий импульс выборки, который соответствует следующему преобразованию канала. Если определена одновременная выборка, A/D может продолжить последовательность преобразования со следующей многоканальной группой.

18.6 Выбор тактирования A/D преобразования

A/D преобразование требует 12 TAD. Источник тактирования A/D преобразования выбирается программно, используя шести битный счётчик. Есть 64 возможные опции для TAD.

Формула 18-1: Тактирование A/D преобразования

$$T_{AD} = T_{CY} \cdot (0.5 \cdot (ADCS(5:0)+1))$$

$$ADCS(5:0) = 2 \cdot \frac{T_{AD}}{T_{CY}} - 1$$

Внутренний RC генератор выбирается установкой бита ADRC.

Для корректных A/D преобразований, тактирование A/D преобразования (TAD) должно быть выбрано так, чтобы гарантировать минимальное время TAD в 83.33 nsec (для VDD = 5V). Обратитесь к части 22.0 "Электрические характеристики" для минимального TAD в других условиях эксплуатации.

Пример 18-1 показывает расчёт выборки для битов ADCS<5:0>, в предположении что скорость работы устройства 30 MIPS.

Пример 18-1: Расчёт тактирования A/D преобразования

$$T_{AD} = 84\ nsec$$
$$T_{CY} = 33\ nsec\ (30\ MIPS)$$

$$ADCS(5:0) = 2 \cdot \frac{T_{AD}}{T_{CY}} - 1 = 2 \cdot \frac{84\ nsec}{33\ nsec} - 1 = 4.09$$

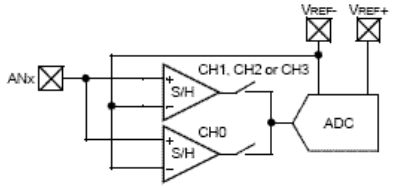
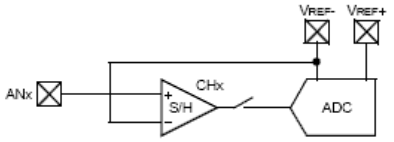
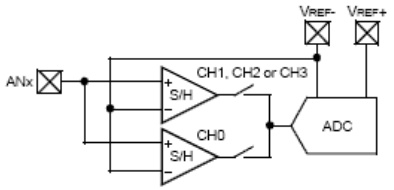
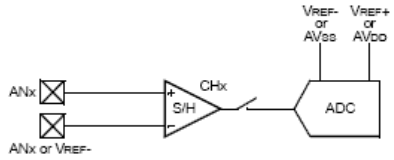
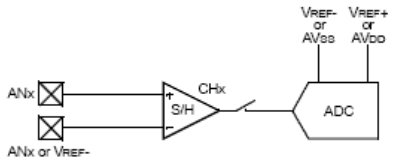
$$Следовательно,$$
$$Set\ ADCS(5:0) = 5$$

$$Фактически\ T_{AD} = \frac{T_{CY}}{2} \cdot (ADCS(5:0)+1) = \frac{33\ nsec}{2} \cdot (5+1) = 99\ nsec$$

18.7 Скорость A/D преобразования

Спецификация на 10-битный ADC dsPIC30F разрешает максимальный темп выборок 1 Msps. Таблица 18-1 резюмирует скорости преобразования для 10-битного ADC устройства dsPIC30F и требуемые условия работы.

Таблица 18-1: Параметры темпа 10-битного A/D преобразования

Темпы преобразования 10-битного конвертера dsPIC30F						
A/D скорость	**TAD минимум, нсек**	**Мин. Время выборки**	**RS Max**	**VDD, Вольт**	**Температура, °C**	**A/D Channels Configuration**
До 1 Msps(1)	83.33	12 TAD	500	4.5 - 5.5	-40 ... +85	
До 750 ksps(1)	95.24	2 TAD	500	4.5 - 5.5	-40 ... +85	
До 600 ksps(1)	138.89	12 TAD	500	3.0 - 5.5	-40 ...+125	
До 500 ksps(1)	153.85	1 TAD	5к	4.5 - 5.5	-40 ...+125	
До 300 ksps(1)	256.41	1 TAD	5к	3.0 - 5.5	-40 ...+125	

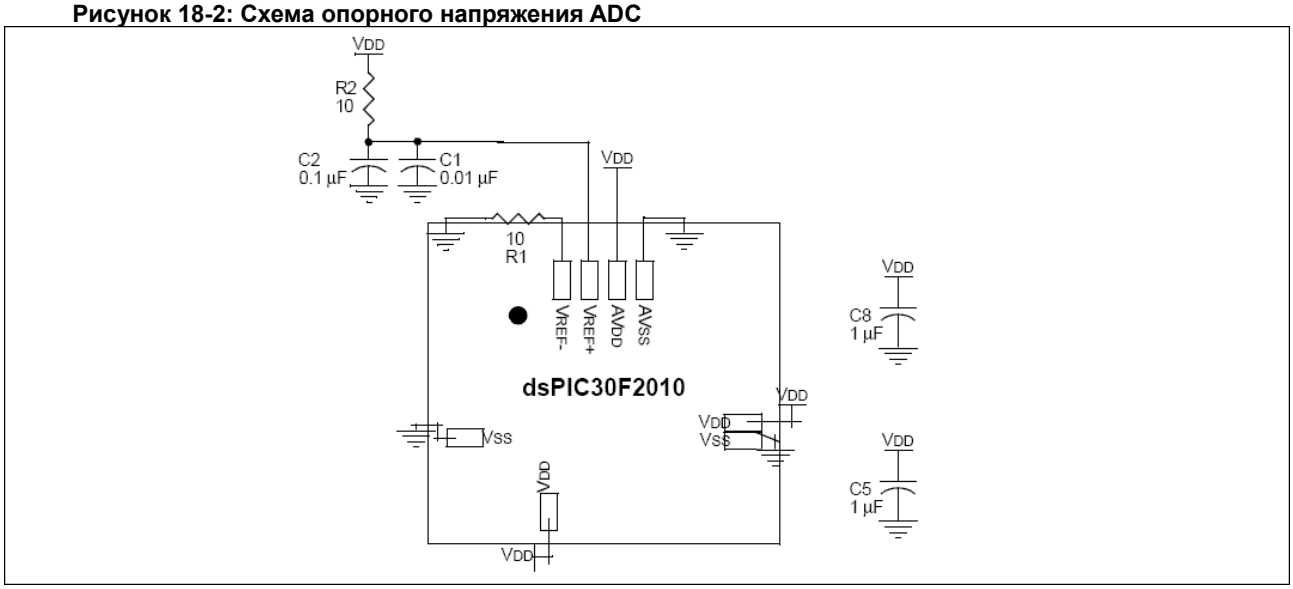
Примечание 1: Внешние ножки VREF- и VREF+ должны использоваться для правильной работы. Смотреть рисунок 18-2 для рекомендуемой схемы.

Рекомендации конфигурации дают требуемые значения установки для скоростей преобразования выше 500 ksps, поскольку они требуют использования ножек внешнего VREF и там есть некоторые отличия в процедуре конфигурации.

Детали конфигурации, которые не критичные для скорости преобразования опущены.

На Рисунке 18-2 изображена рекомендованная схема для пересчёта темпа выше 500 ksps.

Рисунок 18-2: Схема опорного напряжения ADC



18.7.1 1 Msps **Руководящие принципы конфигурирования**

Конфигурация для операции 1 Msps зависит, будет выбран единственная независимая входная ножка или независимые многократные ножки будут выбраны.

18.7.1.1 Единственный аналоговый вход

Для преобразования 1 Msps для единственного аналогового входа, по крайней мере два канала выборки-хранения должны быть разрешены. Аналоговый входной мультиплексор должен быть сконфигурирован с тем чтобы одну и ту же входную ножку подключать к обоим каналам выборки-хранения. Когда на одном канале производится A/D преобразование значения удерживаемого на канале выборки-хранения, другой канал приобретает новую входную выборку.

18.7.1.2 Многократные аналоговые входы

A/D преобразование может также быть использовано для выборки многократных аналоговых входов, используя многократные каналы выборки-хранения. В этом случае, общий темп преобразования 1 Msps делится среди различных входных сигналов. Например, на четырёх входах может производиться выборка в темпе 250 ksps для каждого сигнала или на двух входах может производиться выборка в темпе 500 ksps для каждого сигнала. Последовательная выборки можеть быть использована в этой конфигурации позволяя требуемое время выборки для каждого входа.

18.7.1.3 1 Msps пункты конфигурации

Следующие пункты конфигурации необходимы для достижения темпа преобразования 1 Msps.

• Подчиняться с условиями предусмотренными в таблице 18-2

• Подключить внешние VREF+ и VREF- ножки следуя рекомендациям схемы показанной на рисунке 18-2

• Для разрешения опции авто-преобразования, установить SSRC<2:0> = 111 в регистре ADCON1

• Разрешить автоматическую выборку, установив бит управления ASAM в регистре ADCON1

• Разрешить последовательную выборку, очистив бит SIMSAM в регистре ADCON1

• Разрешить по крайней мере два канала выборки-хранения, записав биты управления CHPS<1:0> в регистре ADCON2

• Записать биты управления SMPI<3:0> в регистре ADCON2 для желаемого числа преобразований между прерываниями. Как минимум, установить SMPI<3:0> = 0001 с тех пор по крайней мере два канала выборки-хранения должны быть разрешены

• Конфигурировать тактовый период A/D равный:

$$\frac{1}{12 \cdot 1.000,000} = 83.33\text{nS}$$

записью битов управления ADCS<5:0> в регистр ADCON3

• Конфигурировать время выборки равный 2 TAD записью: SAMC<4:0> = 00010

• Выбрать по крайней мере два канала на аналоговый вход, записью регистра ADCHS

18.7.2 750 ksps Руководящие принципы конфигурирования

Следующие пункты конфигурации необходимы для достижения темпа преобразования 750 ksps. Эта конфигурация предполагает, что единственный аналоговый вход выбран.

• Подчиняться с условиями предусмотренными в таблице 18-2

• Подключить внешние VREF+ и VREF- ножки следуя рекомендациям схемы показанной на рисунке 18-2

• Для разрешения опции авто-преобразования, установить SSRC<2:0> = 111 в регистре ADCON1

• Разрешить автоматическую выборку, установив бит управления ASAM в регистре ADCON1

• Разрешить один канал выборки-хранения, установив CHPS<1:0> = 00 в регистре ADCON2

• Записать управляющие биты SMPI<3:0> в регистре ADCON2 для желаемого числа преобразований между прерываниями

• Конфигурировать тактовый период A/D равный:

$$\frac{1}{(12+2)\cdot 750{,}000} = 95.24\text{nS}$$

записью битов управления ADCS<5:0> в регистр ADCON3

• Конфигурировать время выборки равный 2 TAD записью: SAMC<4:0> = 00010

18.7.3 600 ksps Руководящие принципы конфигурирования

Конфигурация для операции 600 ksps зависит, будет выбран единственная независимая входная ножка или независимые многократные ножки будут выбраны.

18.7.3.1 Единственный аналоговый вход

Когда выполняются преобразования 600 ksps для единственного аналогового входа, по крайней мере два канала выборки-хранения должны быть разрешены. Аналоговый входной мультиплексор должен быть сконфигурирован с тем, чтобы та же самая входная ножка была подключена к обоим каналам выборки-хранения. Когда на одном канале производится A/D преобразование значения удерживаемого на канале выборки-хранения, другой канал приобретает новую входную выборку.

18.7.3.2 Многократные аналоговые входы

A/D преобразование может также быть использовано для выборки многократных аналоговых входов, используя многократные каналы выборки-хранения. В этом случае, общий темп преобразования 600 ksps делится среди различных входных сигналов. Например, на четырёх входах может производиться выборка в темпе 150 ksps для каждого сигнала или на двух входах может производиться выборка в темпе 300 ksps для каждого сигнала. Последовательная выборки можеть быть использована в этой конфигурации позволяя требуемое время выборки для каждого входа.

18.7.3.3 600 ksps пункты конфигурации

Следующие пункты конфигурации необходимы для достижения темпа преобразования 600 ksps.

• Подчиняться с условиями предусмотренными в таблице 18-2

• Подключить внешние VREF+ и VREF- ножки следуя рекомендациям схемы показанной на рисунке 18-2

• Для разрешения опции авто-преобразования, установить SSRC<2:0> = 111 в регистре ADCON1

• Разрешить автоматическую выборку, установив бит управления ASAM в регистре ADCON1

• Разрешить последовательную выборку, очистив бит SIMSAM в регистре ADCON1

• Разрешить по крайней мере два канала выборки-хранения, записав биты управления CHPS<1:0> в регистре ADCON2

• Записать биты управления SMPI<3:0> в регистре ADCON2 для желаемого числа преобразований между прерываниями. Как минимум, установить SMPI<3:0> = 0001 с тех пор по крайней мере два канала выборки-хранения должны быть разрешены

• Конфигурировать тактовый период A/D равный:

$$\frac{1}{12\cdot 600{,}000} = 138.89\text{nS}$$

записью битов управления ADCS<5:0> в регистр ADCON3

• Конфигурировать время выборки равный 2 TAD записью: SAMC<4:0> = 00010

Конфигурировать по крайней мере два канала на аналоговый входную ножку, записью регистра ADCHS.

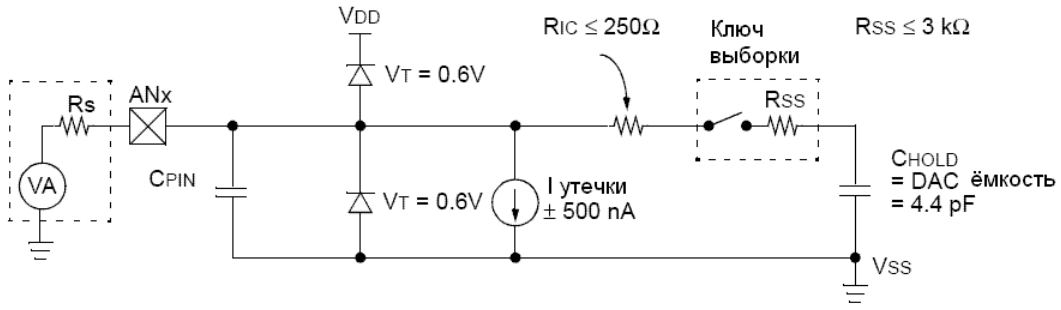
18.8 Требования приобретения A/D

Модель аналогового входа 10-битного ADC показана на рисунке 18-3. Общее время выборки для A/D есть функция от времени установки внутреннего усилителя, устройства VDD времени зарядки конденсатора хранения.

Для обеспечения точности A/D определённой спецификацией, зарядка конденсатора хранения (CHOLD) должна позволять полную зараядку до уровня напряжения на аналоговой входной ножке. Сопротивление источника (RS), сопротивление связи (RIC), и сопротивление внутреннего ключа выборки (RSS) объединяются и непосредственно влияют на время требуемое для зарядки конденсатора CHOLD. Объединённое сопротивление аналоговых источников должно следовательно быть достаточно небольшим для обеспечения полной зарядки конденсатора хранения в течении определённого времени выборки. Для минимизации влияния токов утечки ножки на точность A/D преобразования, максимальное рекомендуемое сопротивление RS будет 5 kΩ. После того, как аналоговый канал входа выбран (изменён), эта функция выборки должна быть завершена до начала преобразования. Внутренний конденсатор хранения будет в разряженном состоянии раньше каждой операции выборки.

Пользователь должен позволять по крайней мере 1 период TAD времени выборки, TSAMP, между преобразованиями позволять приобретать каждой выборке. ТЭто время выборки может контролироваться вручную в программе путём установки/сброса бита SAMP, или это может контролироваться автоматически A/D преобразователем. В автоматической конфигурации, пользователь должен позволить достаточное время между запусками преобразования с тем чтобы минимальное время выборки могло быть удовлетворено. Обратитесь к Электрическим спецификациям для требований TAD и времени выборки.

Рисунок 18-3: Модель аналогового входа ADC



Описание:	CPIN	= входная ёмкость
	VT	= пороговое напряжение
	I leakage	= ток утечки ножки из-за различных соединений
	RIC	= сопротивление связи
	RSS	= сопротивление ключа выборки
	CHOLD	= ёмкость выборки-хранения (из DAC)

Примечание: CPIN значение зависит от упаковки устройства и не тестировалось. Эффект от CPIN незначителен, если Rs ≤ 5 kΩ.

18.9 Модуль режимов выключения

Модуль имеет 3 внутренних режима потребления. Когда бит ADON есть ‘1’, модуль находится в активном режиме; это есть полное потребление и функциональность. Когда ADON есть ‘0’, модуль находится в режиме отключения. Цифровые и аналоговые части схемы блокированы для для максимального хранения тока. В порядке возврата в активный режим из отключенного режима, пользователь должен ждать для стабилизации схемы ADC.

18.10 Работа A/D в течении режимов спячки и простоя CPU

18.10.1 Работа A/D в течении режима спячки CPU

Когда устройство входит в режим спячки, все тактовые источники модуля отключаются и остаются в логическом ‘0’.

Если спячка происходит в середине преобразования, преобразование прерывается. Преобразователь не будет продолжать частично завершённое преобразование при выходе из режима спячки.

Содержимое регистра не тронуто при входе или выходе устройства из спячки.

Модуль A/D может работать в режиме спячки, если в качестве тактового источника A/D установлен RC (ADRC = 1). Когда выбран RC тактовый источник, модуль ждёт один цикл инструкции перед началом преобразования. Это позволяет выполнение инструкции SLEEP, что устраняет все цифровые шумы переключения, чтобы преобразовывать. Когда преобразование завершено, бит DONE устанавливается и результат загружается в регистр ADCBUF.

Если разрешено прерывание A/D, устройство пробуждается из спячки. Если прерывание A/D не разрешено, то модуль A/D затем будет отключен, хотя бит ADON останется установленным.

18.10.2 Работа A/D в течении режима простоя CPU

Бит ADSIDL выбран если модуль остановился на простое или продолжает на простое. Если ADSIDL = 0, модуль продолжит работу на утверждении режима простоя. Если ADSIDL = 1, модуль остановится на простое.

18.11 Результаты сброса

Сброс устройства заставляет все регистры перейти в их состояние сброса. Это заставит модуль A/D отключиться, и любая последовательность преобразования и приобретения будет прервана. Значения, которые были в регистрах ADCBUF не будут изменены. Регистр результата будут содержать неизвестные данные после сброса по включению питания.

18.12 Выходные форматы

Результат A/D имеет ширину 10-бит. RAM буфера данных также имеет 10-битную ширину. 10-битные данные могут читаться в одном из четырёх форматов. Биты FORM<1:0> выбирают формат. Каждый из выходных форматов транслирует в 16-битный результат на шине данных.

Запись данных всегда будет выровненный по правому краю (целый) формат.

Рисунок 18-4: A/D Выходные форматы данных

Содержимое RAM:							d09	d08	d07	d06	d05	d04	d03	d02	d01	d00
Чтение в Шину:																
Дробь со знаком (1.15)	$\overline{d09}$	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0
Дробь (1.15)	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0
Целое со знаком	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	d08	d07	d06	d05	d04	d03	d02	d01	d00
Целое	0	0	0	0	0	0	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

18.13 Конфигурация ножек аналогового порта

Использование ADPCFG и TRIS регистров управления работой ножек порта A/D. Ножки порта, которые нужны как аналоговые входы, должны иметь установленным их соответствующий бит TRIS (вход). Если бит TRIS очищен (выход), цифровой выходной уровень (VOH или VOL) будет преобразован.

Работа A/D не зависит от состояния битов CH0SA<3:0>/CH0SB<3:0> и битов TRIS.

Когда читается регистр PORT, все ножки, сконфигурированы как аналоговые входные каналы, читаются как очищенные.

Ножки сконфигурированные как цифровые входы не преобразуют аналоговый вход. Аналоговые уровни на любых ножках, которые определены как цифровые входы (включая ножки ANx), могут заставлять входной буфер поглощать ток, который превышает допустимый для устройства.

18.14 Соображения подключения

Аналоговые входы имеют диоды к VDD и VSS как защиту. Это требует, чтобы чтобы аналоговые входы были между VDD и VSS. Если входное напряжение превышает эту область более чем на 0.3V (в любом направлении), один из диодов становится прямо смещённым и это может повредить устройство если допустимый входной ток будет превышен.

Внешний RC фильтр иногда добавляется для сглаживания входного сигнала. Компонент R должен быть выбран гарантируя, что требование времени выборки будут удовлетворены. Любые подключенные внешние компоненты (через высокое сопротивление) к аналоговой входной ножке (конденсатор, стабилитрон, и т.д.) должны иметь очень маленькую утечку тока ножки.

Таблица 18-2: Регистровая карта ADC

SFR Имя	Адрес	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
ADCBUF0	0280	—	—	—	—	—	—	ADC буфер данных 0										0000 00uu uuuu uuuu
ADCBUF1	0282	—	—	—	—	—	—	ADC буфер данных 1										0000 00uu uuuu uuuu
ADCBUF2	0284	—	—	—	—	—	—	ADC буфер данных 2										0000 00uu uuuu uuuu
ADCBUF3	0286	—	—	—	—	—	—	ADC буфер данных 3										0000 00uu uuuu uuuu
ADCBUF4	0288	—	—	—	—	—	—	ADC буфер данных 4										0000 00uu uuuu uuuu
ADCBUF5	028A	—	—	—	—	—	—	ADC буфер данных 5										0000 00uu uuuu uuuu
ADCBUF6	028C	—	—	—	—	—	—	ADC буфер данных 6										0000 00uu uuuu uuuu
ADCBUF7	028E	—	—	—	—	—	—	ADC буфер данных 7										0000 00uu uuuu uuuu
ADCBUF8	0290	—	—	—	—	—	—	ADC буфер данных 8										0000 00uu uuuu uuuu
ADCBUF9	0292	—	—	—	—	—	—	ADC буфер данных 9										0000 00uu uuuu uuuu
ADCBUFA	0294	—	—	—	—	—	—	ADC буфер данных A										0000 00uu uuuu uuuu
ADCBUFB	0296	—	—	—	—	—	—	ADC буфер данных B										0000 00uu uuuu uuuu
ADCBUFC	0298	—	—	—	—	—	—	ADC буфер данных C										0000 00uu uuuu uuuu
ADCBUFD	029A	—	—	—	—	—	—	ADC буфер данных D										0000 00uu uuuu uuuu
ADCBUFE	029C	—	—	—	—	—	—	ADC буфер данных E										0000 00uu uuuu uuuu
ADCBUFF	029E	—	—	—	—	—	—	ADC буфер данных F										0000 00uu uuuu uuuu
ADCON1	02A0	ADON	—	ADSIDL	—	—	—	FORM<1:0>		SSRC<2:0>			—	SIMSAM	ASAM	SAMP	DONE	0000 0000 0000 0000
ADCON2	02A2	VCFG<2:0>			—	—	CSCNA	CHPS<1:0>		BUFS	—	SMPI<3:0>				BUFM	ALTS	0000 0000 0000 0000
ADCON3	02A4	—	—	—	SAMC<4:0>					ADRC	—	ADCS<5:0>						0000 0000 0000 0000
ADCHS	02A6	CH123NB<1:0>		CH123SB	CH0NB	CH0SB<3:0>				CH123NA<1:0>		CH123SA	CH0NA	CH0SA<3:0>				0000 0000 0000 0000
ADPCFG	02A8	—	—	—	—	—	—	—	—	—	—	PCFG5	PCFG5	PCFG5	PCFG5	PCFG5	PCFG5	0000 0000 0000 0000
ADCSSL	02AA	—	—	—	—	—	—	—	—	—	—	CSSL5	CSSL5	CSSL5	CSSL5	CSSL5	CSSL5	0000 0000 0000 0000

Описание: u = неинициализированный бит

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

19.0 Системная интеграция

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

Есть различные характерные черты предполагающие расширение возможностей системы и минимизировать стоимость, путём устранения внешних компонентов, обеспечить экономичные режимы работы и предложить защиту кода:

• Выбор генератора
• Сброс
- Сброс при включении питания (POR)
- Таймер включения питания (PWRT)
- Таймер запуска генератора (OST)
- Программируемый сброс при нарушении питания (BOR)
• Сторожевой таймер (WDT)
• Энергосберегающие режимы (Спячки и Простоя)
• Защита кода
• Позиции ID устройства
• Возможность внутрисхемного последовательного программирования (ICSP)

Устройства dsPIC30F имеют сторожевой таймер, который постоянно разрешён через биты конфигурации или может управляться программно. Таймер работает от собственного RC генератора для увеличения надёжности. Имеются два таймера, которые обеспечивают необходимые задержки при включении питания. Один - таймер запуска генератора (OST), удерживает чип в состоянии сброса пока стабилизируется кварцевый генератор. Другой - таймер включения питания (PWRT), который обеспечивает только задержку на включение питания, разработанный чтобы удерживать часть в состоянии сброса, пока источник питания стабилизируется. С этими двумя внутричиповыми таймерами, большинство приложений не нуждаются во внешней схеме сброса.

Режим Спячки разработан, чтобы обеспечить очень низкий потребляемый ток в выключенном режиме. Пользователь может пробудится из Спячки через внешний сброс, пробуждение от сторожевого таймера или через прерывание. Также несколько доступных опций генератора позволяют части соответствовать широкой разновидности приложений. В режиме Простоя, тактовые источники по прежнему активны, но CPU выключен. Опция RC генератора сохраняет системный расход, пока опция LP кварца сохраняет питание.

19.1 Обзор системного генератора

Системный генератор dsPIC30F имеет следующие модули и характеристики:

• Различные опции внешнего и внутреннего генератора как тактового источника
• Внутричиповый PLL повышающий внутреннюю рабочую частоту
• Механизм переключения между различными тактовыми источниками
• Программируемый тактовый постделитель для сохранения системной энергии
• Монитор защиты тактирования (FSCM), который обнаруживает отказ тактирования и принимает защитные меры
• Регистр управления тактированием OSCCON
• Биты конфигурации для выбора главного генератора

Таблица 19-1 обеспечивает суммарные данные по режимам работы генератора dsPIC30F. Упрощённая схема системы генератора показана на рисунке 19-1.

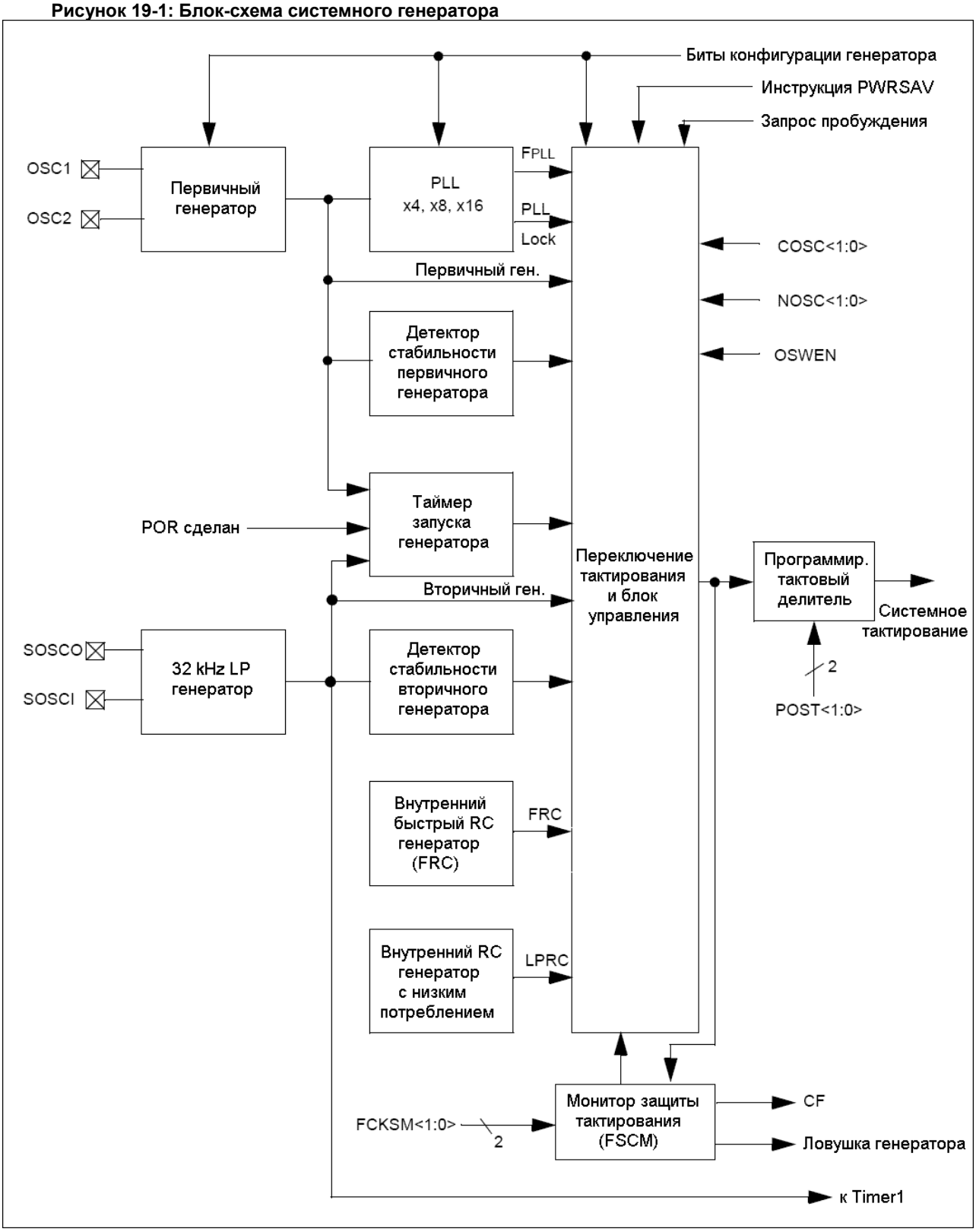
Биты конфигурации определяют тактовый источник работающий с сбросом при включении питания (POR) и сбросом при провале питания (BOR). Согласно этому, тактовый источник может быть изменён между допустимыми тактовыми источниками. Регистр OSCCON управляет переключением тактирования и отражает связанные биты состояния системного тактирования.

Таблица 19-1: Режимы работы генератора

Режим генератора	Описание
XTL	200 kHz-4 MHz кварц на OSC1:OSC2.
XT	4 MHz-10 MHz кварц на OSC1:OSC2.
XT w/ PLL 4x	4 MHz-10 MHz кварц на OSC1:OSC2. 4x PLL разрешён.
XT w/ PLL 8x	4 MHz-10 MHz кварц на OSC1:OSC2. 8x PLL разрешён.
XT w/ PLL 16x	4 MHz-10 MHz crystal on кварц на OSC1:OSC2. 16x PLL разрешён(1).
LP	32 kHz кварц на SOSCO:SOSCI(2).
HS	10 MHz-25 MHz кварц.
EC	Внешний тактовый вход (0-40 MHz).
ECIO	Внешний тактовый вход (0-40 MHz). OSC2 pin is I/O.
EC w/ PLL 4x	Внешний тактовый вход (0-40 MHz). OSC2 pin is I/O. 4x PLL разрешён(1).
EC w/ PLL 8x	Внешний тактовый вход (0-40 MHz). OSC2 pin is I/O. 8x PLL разрешён(1).
EC w/PLL 16x	Внешний тактовый вход (0-40 MHz). OSC2 pin is I/O. 16x PLL разрешён(1).
ERC	Внешний RC генератор. Ножка OSC2 является выходом(3) FOSC/4.
ERCIO	Внешний RC генератор. Ножка OSC2 является входом/выходом(3)
FRC	7.37 MHz внутренний RC генератор.
LPRC	512 kHz внутренний RC генератор.

Примечание **1:** Встречаются dsPIC30F с максимальной рабочей частотой до 120 MHz.
2: LP генератор может быть распределён как системный тактовый источник, а так же как источник тактирования часов реального времени для Timer1.
3: Требуются внешние R и C. Рабочая частота до 4 MHz.

Рисунок 19-1: Блок-схема системного генератора



19.2 Конфигурации генератора

19.2.1 Выбор источника тактирования инициализации

При выходе из сброса по включению питания или по провалу питания, устройство выбирает этот источник тактирования основываясь на:

a) Битах конфигурации FOS<1:0>, которые выбирают одну из четырёх групп генераторов.

b) И битах конфигурации FPR<3:0>, которые выбирают один из 13 генераторных выборов в пределах первичной группы.

Выбор делается как показано в таблице 19-2.

Таблица 19-2: Значение бита конфигурации для выбора тактирования

Режим генератора	Генератор источник	FOS1	FOS0	FPR3	FPR2	FPR1	FPR0	Функция OSC2
EC	Первичный	1	1	1	0	1	1	CLKO
ECIO	Первичный	1	1	1	1	0	0	I/O
EC w/ PLL 4x	Первичный	1	1	1	1	0	1	I/O
EC w/ PLL 8x	Первичный	1	1	1	1	1	0	I/O
EC w/ PLL 16x	Первичный	1	1	1	1	1	1	I/O
ERC	Первичный	1	1	1	0	0	1	CLKO
ERCIO	Первичный	1	1	1	0	0	0	I/O
XT	Первичный	1	1	0	1	0	0	OSC2
XT w/ PLL 4x	Первичный	1	1	0	1	0	1	OSC2
XT w/ PLL 8x	Первичный	1	1	0	1	1	0	OSC2
XT w/ PLL 16x	Первичный	1	1	0	1	1	1	OSC2
XTL	Первичный	1	1	0	0	0	x	OSC2
HS	Первичный	1	1	0	0	1	x	OSC2
LP	Вторичный	0	0	-	-	-	-	(Прим. 1,2)
FRC	Внутр. FRC	0	1	-	-	-	-	(Прим. 1,2)
LPRC	Внутр. LPRC	1	0	-	-	-	-	(Прим. 1,2)

19.2.2 Генератор таймера включения питания (OST)

В порядке гарантии, что кварцевый генератор (или керамический резонатор) стартовал и стабилизировался, генератор таймера включения питания включен. Это простой 10-битный счётчик, который считает 1024 TOSC цикла перед разрешением тактовому генератору работать с системой. Перерыв в работе определяется как TOST. Время TOST вызывается при каждом рестарте генератора (т.е., на POR, BOR и пробуждение от Спячки). Таймер запуска генератора применяется для LP генератора, XT, XTL, и HS режимов (с пробуждением из Спячки, POR и BOR) для первичного генератора.

19.2.3 Управление LP генератором

Разрешение LP генератора управляется двумя элементами:

1. Биты текущей группы генераторов COSC<1:0>.
2. Бит LPOSCEN (регистр OSCCON).

LP генератор включен (даже в течении режима Спячки) если LPOSCEN = 1. LP генератор тактирует устройство если:

• COSC<1:0> = 00 (LP выбран как главный генератор)

и

• LPOSCEN = 1

Держание LP генератора включенным всегда учитывает быстрое переключение на системное тактирование 32 kHz для операций с низким потреблением энергии. Возврат к быстрому главному генератору будет требовать времени запуска.

19.2.4 Фазовая автоподстройка (PLL)

PLL умножает тактовую частоту, которая генерируется первичным генератором или быстрым RC генератором. PLL настраивается чтобы иметь требуемое увеличение x4, x8, и x16. Входной и выходной диапазоны частоты подытожены в таблице 19-3.

Таблица 19-3: Диапазон частоты PLL

Fвх	PLL умножение	Fвых
4 MHz-10 MHz	x4	16 MHz-40 MHz
4 MHz-10 MHz	x8	32 MHz-80 MHz
4 MHz-7.5 MHz	x16	64 MHz-120 MHz

PLL показывает выход захвата, который утверждён когда PLL входит в состояние захвата фазы. Если контур выпадает из захвата (например из-за шума), сигнал захвата отменяется. Состояние этого сигнала отображается в только читаемом бите LOCK в регистре OSCCON.

19.2.5 Быстрый RC генератор (FRC)

FRC генератор является быстрым (7.37 MHz ±2% номинал) внутренним RC генератором. Этот генератор предназначен, чтобы обеспечить разумную скорость работы устройства без использования внешней кварцевой, керамической или RC цепей.

dsPIC30F работает от генератора FRC когда биты выбора текущего генератора в регистре OSCCON (OSCCON<13:12>) установлены в ‘01’.

Четырехбитная область, указанная TUN<3:0> (OSCCON<15:14> и OSCCON<11:10>), позволяет пользователю настроить внутренний быстрый RC генератор (номинал 7.37MHz). Пользователь может настроить FRC генератор в пределах диапазона от -12% (или -960 kHz) до +10.5% (или +840 kHz) шагами по 1.50% относительно калибровки сделанной на заводе, смотреть таблицу 19-4.

Таблица 19-4: Настройка FRC

Биты TUN<3:0>	Частота FRC
0111	+ 10.5%
0110	+ 9.5%
0101	+ 7.5%
0100	+ 6.0%
0011	+ 4.5%
0010	+ 3.0%
0001	+ 1.5%
0000	Центральная частота (генератор работает на откалиброванной частоте)
1111	- 1.5%
1110	- 3.0%
1101	- 4.5%
1100	- 6.0%
1011	- 7.5%
1010	- 9.5%
1001	- 10.5%
1000	- 12.0%

19.2.6 RC генератор с низким потреблением (LPRC)

LPRC генератор является частью сторожевого таймера (WDT) и генерирует номинальную частоту 512 kHz. LPRC генератор является тактовым источником для схемы таймера включения (PWRT), WDT, и монитора защиты тактирования. Он так же может быть использован для обеспечения опции низкочастотного источника тактирования для приложений критичным к потребляемой мощности и не требуют высокой точности тактирования.

LPRC генератор всегда разрешён сбросом по включению питания, потому что является тактовым источником для PWRT. После окончания PWRT, LPRC генератор остаётся включенным если одно из далее перечисленного справедливо:

• Монитор защиты тактирования разрешён
• WDT разрешён
• LPRC генератор выбран как системный тактовый источник через биты управления COSC<1:0> в регистре OSCCON

Если ни одно из вышеуказанных условий не является истинным, LPRC будет выключен после истечения PWRT.

Примечание 1: Функция ножки OSC2 определяется выбором режима первичного генератора (FPR<3:0>).

2: Ножка OSC1 не может быть использована как ножка ввода/вывода даже если вторичный генератор или внутренний источник тактирования выбраны всегда.

19.2.7 Монитор защиты тактирования

Монитор защиты тактирования (FSCM) позволяет устройству продолжать работать даже в случае отказа генератора. Функция FSCM программируется соответствующими битами конфигурации FCKSM (биты переключения тактирования и выбора монитора) в регистре конфигурации устройства FOSC. Если функция FSCM разрешена, внутренний генератор LPRC работает всё время (за исключением режима Спячки) и не управляется битом SWDTEN.

В случае отказа генератора, FSCM генерирует событие ловушки отказа тактирования и переключает системное тактирование к FRC генератору. Пользователь затем имеет возможность попытаться перезапустить генератор или выполнить управляемое завершение работы. Пользователь может решить обратиться к ловушке, так как тёплый сброс просто загружает адрес сброса в вектор ловушки отказа генератора. В этом случае, бит состояния (отказ тактирования) CF (OSCCON<3>) устанавливается всякий раз, когда отказ тактирования признаётся.

В случае отказа тактирования, WDT не затрагивается и продолжает работать на тактировании LPRC.

В случае если генератор имеет очень длительное время запуска следующее после POR, BOR или Спячки, может случится, что таймер PWRT окончит счёт перед тем, как генератор стартует. В таких случаях, FSCM будет активирован и FSCM инициирует ловушку отказа тактирования, и биты COSC<1:0> будут загружены выбранным генератором FRC. Это эффективно отключит настоящий генератор, который делал попытку запуска.

Пользователь может определить эту ситуацию и перезапустить генератор в ловушке отказа тактирования ISR.

С определением отказа тактирования, модуль FSCM инициирует переключение тактирования к генератору FRC как следует:

1. Биты COSC (OSCCON<13:12>) загружены с значением выбора генератора FRC.
2. Бит CF установлен (OSCCON<3>).
3. Управляющий бит OSWEN (OSCCON<0>) очищен.

Для успеха переключения тактирования, источник тактирования выбирается в четырёх группах:

1. Первичный
2. Вторичный
3. Внутренний FRC
4. Внутренний LPRC

Пользователь может переключаться между этими функциональными группами, но не может переключаться между опциями в пределах группы. Если выбрана первичная группа, тогда выбор в пределах группы всегда определяется битами конфигурации FPR<3:0>.

Регистр OSCCON хранит биты управления и состояния связанные с переключением тактирования.

• COSC<1:0>: Только читаемые биты состояния, которые отражают текущую действующую группу генератора.

• NOSC<1:0>: Биты управления, которые записываются для указания новой группы генератора.

- На POR и BOR COSC<1:0> и NOSC<1:0> оба загружаются значениями бита конфигурации FOS<1:0>.

• LOCK: Бит состояния LOCK показывает захват PLL.

• CF: Только читаемый бит состояния индицирующий если произошло обнаружение отказа тактирования.

• OSWEN: Бит управления изменяющийся от ‘0’ к ‘1’, когда инициирована последовательность тактового перехода. очистка бита управления OSWEN прерывает процесс тактового перехода (используется для создания ситуации зависания).

Если биты конфигурации FCKSM<1:0> = 1x, тогда функции переключения тактирования и монитора защиты тактирования будут блокированы. Это есть установка по умолчанию бита конфигурации.

Если переключение тактирования блокировано, тогда биты FOS<1:0> и FPR<3:0> непосредственно управляют выбором генератора и биты COSC<1:0> не управляют выбором тактирования. Тем не менее эти биты отражают выбор тактового источника.

Примечание:	Приложение не должно пытаться переключить тактирование с частотой ниже чем 100 Khz, когда монитор защиты тактирования разрешён. Если такое переключение тактирования выполнено, устройство может генерировать ловушку отказа генератора и переключится на быстрый RC генератор.

19.2.8 Защита против случайной записи в OSCCON

Умышленную запись в регистр OSCCON сделать трудно, потому что он контролирует переключение тактирования и масштабирование тактирования.

Для записи младшего байта OSCCON, следующая кодовая последовательность должна быть выполнена без любой другой инструкции между:

• Запись байта “0x46” в младший OSCCON

• Запись байта “0x57” в младший OSCCON

Запись байта предусмотрена для одного цикла инструкции. Записать желаемое значения или использовать инструкцию манипуляции битами.

Для записи старшего байта OSCCON, следующие инструкции должны быть выполнены без любой другой инструкцией между:

• Запись байта “0x78” в старший OSCCON

• Запись байта “0x9A” в старший OSCCON

Запись байта предусмотрена для одного цикла инструкции. Записать желаемое значения или использовать инструкцию манипуляции битами.

19.3 Сброс

Различия между различными типами сброса dsPIC30F2010:

a) Сброс при включении питания (POR)

b) Сброс MCLR в течении нормальной работы

c) Сброс MCLR в течении Спячки

d) Сброс от сторожевого таймера (WDT) в течении нормальной работы

e) Programmable Brown-out Reset (BOR)

f) Инструкция RESET

g) Сброс вызываемый блокировкой ловушки (TRAPR)

h) Сброс вызываемый незаконным кодом операции, или использованием неинициализированного регистра W как адресного указателя (IOPUWR)

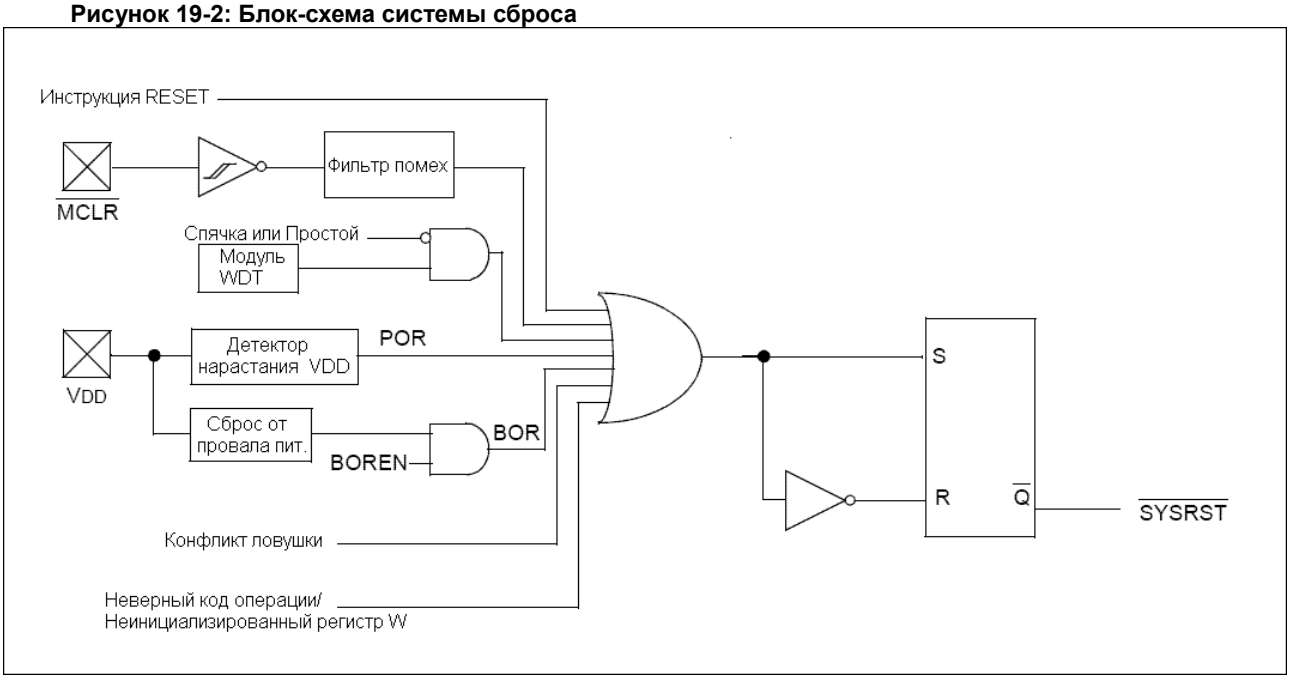
Различные регистры затрагиваются в различных путях различных условий сброса. Большинство резисторов не задействовано в WDT пробуждении, поскольку это рассматривается как возобновление нормальной работы. Биты состояния регистра RCON установлены или сброшены иначе в различных ситуациях сброса, как показано в таблице 19-5. Эти биты используются в программе для определения характера сброса.

Блок схема внутричиповой схемы сброса показана на рисунке 19-2.

Фильтр шума /MCLR предусмотрен в маршруте сброса. Фильтр определяет и игнорирует маленькие пульсации.

Внутренние сбросы не приводят ножку MCLR вниз.

Рисунок 19-2: Блок-схема системы сброса



19.3.1 POR: Сброс при включении питания

Событие включения питания генерирует внутренний импульс, когда обнаруживается нарастание VDD. Импульс сброса произойдёт в пороговом напряжении (VPOR) схемы POR, которое номинально равно 1.85V. Характеристики напряжения питания устройства должны выполнить указанное стартовое напряжение и требование темпа нарастания. Импульс POR сбрасывает таймер POR и устанавливает устройство в состояние сброса. POR также выбирает источник тактирования устройства определённый битами конфигурации генератора.

Схема POR включает маленькую задержку, TPOR, которая номинально составляет 10 µs и гарантирует смещение схем устройства будет стабильным. Кроме этого, пользователь выбирает прилагаемый перерыв включения питания. Параметр TPWRT основан на битах конфигурации устройства и может быть 0 ms(нет задержки), 4 ms, 16 ms или 64 ms. Общая задержка включения устройства равна TPOR + TPWRT. Когда эта задержка истекает, SYSRST может быть отвергнут на следующий передний фронт такта Q1, и PC прыгнет на вектор сброса.

Временная диаграмма сигнала SYSRST показана на рисунках 19-3, 19-4, 19-5.

Рисунок 19-3: Последовательность перерыва на включение питания (/MCLR связан с VDD)

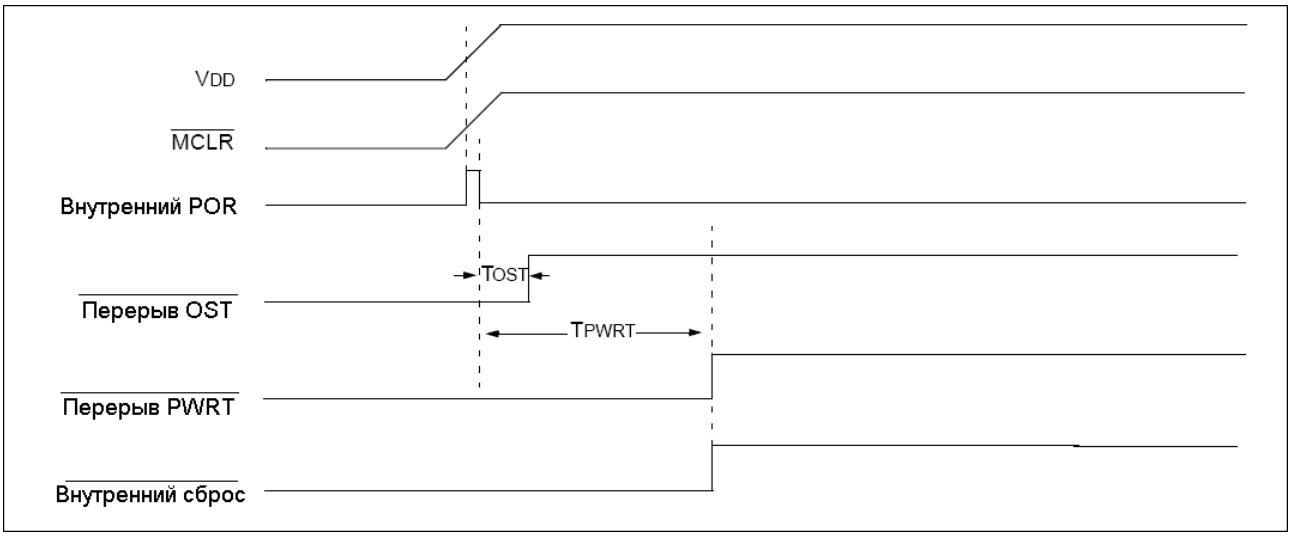


Рисунок 19-4: Последовательность перерыва на включение питания (/MCLR не связан с VDD): Случай 1

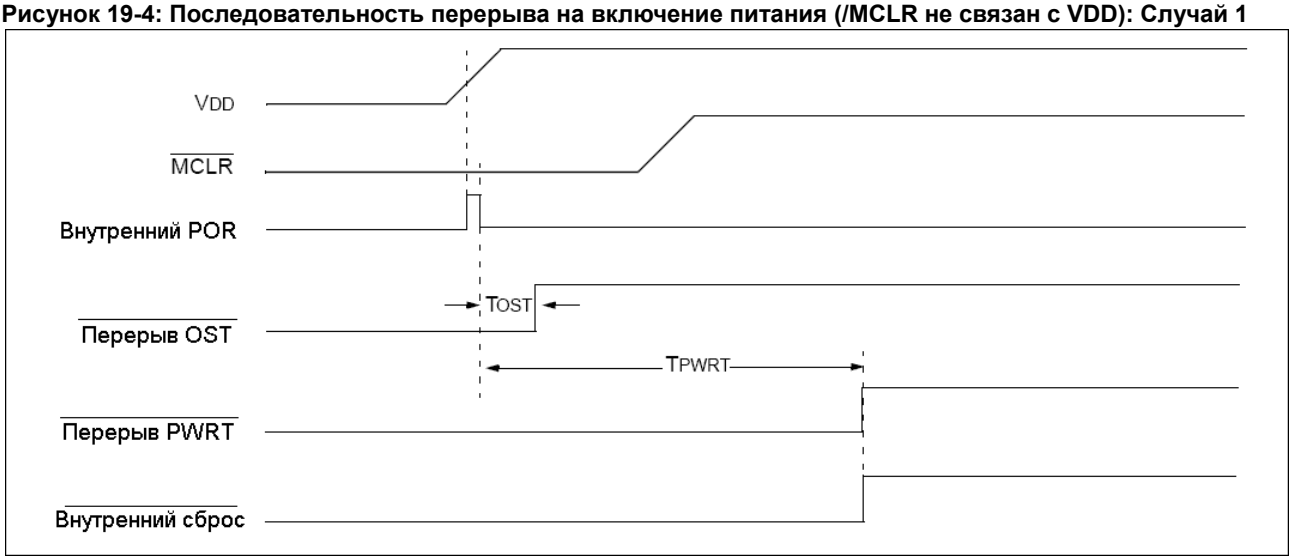
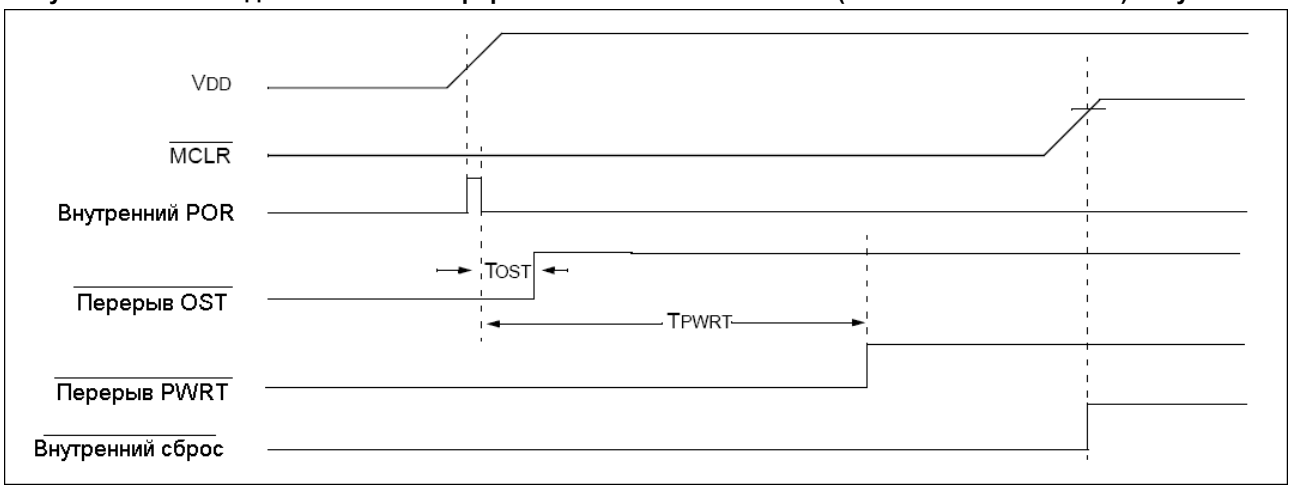


Рисунок 19-5: Последовательность перерыва на включение питания (/MCLR не связан с VDD): Случай 2



19.3.1.1 POR с длительным временем запуска кварца (с разрешённым FSCM)

Схема запуска генератора не связана с POR схемой. Некоторые схемы кварцевых генераторов (особенно на низкочастотных кварцах) могут иметь относительно длительное время запуска. Следовательно, одно или более следующих условий возможны после истекания времени POR и PWRT:

• Схема генератора не начала генерировать.
• Таймер запуска генератора не истёк (если используется кварцевый генератор).
• PLL не достигла захвата (если используется PLL).

Если FSCM разрешён и одно из вышеуказанных событий справедливо, то может произойти ловушка отказа тактирования. Устройство автоматически переключится на FRC генератор и пользователь может переключится на желаемый кварцевый генератор в ловушке ISR.

19.3.1.2 Работа без FSCM и PWRT

Если FSCM блокирован и таймер включения питания (PWRT) также блокирован, то устройство быстро выйдет из из сброса на включение питания. Если в качестве тактового источника используется FRC, LPRC, EXTRC или EC, это будет активировано немедленно.

Если FSCM блокирован и системное тактирование не стартовало, устройство окажется в замороженном состоянии вектора сброса, пока системное тактирование стартует. Из перспективы пользователя, устройство оказывается будет в состоянии сброса, пока будет достигнуто системное тактирование.

19.3.2 BOR: Программируемый сброс по провалу питания

Модуль BOR основывается на внутренней схеме опорного напряжения. Главной целью модуля BOR является генерация сброса устройства когда состояние провала питания происходит. Условия провала питания обычно являются причиной отказа на магистралях переменного тока (т.е., пропуск части формы цикла AC благодаря плохой передачи энергии по линиям или снижению напряжения благодаря чрезмерному току потребления, когда подключается большая индуктивная нагрузка).

Модуль BOR допускает выбор одного из следующих точек падения напряжения:

• 2.6V-2.71V
• 4.1V-4.4V
• 4.58V-4.73V

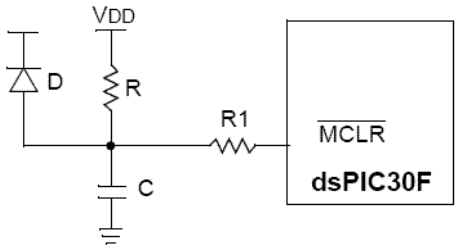
Примечание:	Указанные здесь, номинальные значения точек падения напряжения BOR предусмотрены только для руководства разработкой.

BOR генерирует импульс сброса, который сбрасывает устройство. BOR выбирает источник тактирования, основываясь на значениях бит конфигурации устройства (FOS<1:0> и FPR<3:0>). Кроме того, если режим генератора выбран, BOR активизирует таймер запуска генератора (OST). Системное тактирование приостанавливается пока не истечёт OST. Если используется PLL, тогда тактирование может быть приостановлено пока бит захвата (OSCCON<5>) не ‘1’.

Одновременно, перед освобождением внутреннего сброса, будут применяться POR перерыв (TPOR) и PWRT перерыв (TPWRT). Если TPWRT = 0 и кварцевый генератор используется, тогда номинальная задержка TFSCM = 100 μs применяется. Общая задержка в этом случае составляет (TPOR + TFSCM).

Бит состояния BOR (RCON<1>) будет установлен для отображения, что происходит BOR. Схема BOR, если разрешена, может продолжать работать в режимы Спячки или Простоя и сбрасывает устройство если VDD падает ниже порогового напряжения BOR.

Рисунок 19-6: Внешняя схема сброса по включению питания (для замедления нарастания VDD)



Примечание 1:	Внешняя схема сброса по включению питания необходима только если скорость нарастания VDD слишком маленькая. Диод D помогает быстро разрядить конденсатор, когда питание отключается.
2:	R должен быть правильно выбран для того чтобы быть уверенным что падение напряжения на R не вызовет нарушения электрических технических требований устройства.
3:	R1 должен быть правильно выбран для того, чтобы ограничить любой ток втекающий в MCLR из внешнего конденсатора C, в случае пробоя ножки MCLR/VPP из-за разряда статического электричества (ESD) или перенапряжения (EOS).

Примечание:	Специализированные устройства супервизоры, как например MCP1XX и MCP8XX, могут также быть использованы в качестве внешней схемы сброса на включение питания.

Таблица 19-5 показывает условия сброса для регистра RCON. Посколку биты управления в пределах регистра RCON быть R/W, информация в таблице подразумевает что все биты инвертированы до действия указанного в столбце состояния.

Таблица 19-5: Состояние инициализации для регистра RCON в случае 1

Состояние	Программный счётчик	TRAPR	IOPUWR	EXTR	SWR	WDTO	IDLE	SLEEP	POR	BOR
Сброс на включение	0x000000	0	0	0	0	0	0	0	1	1
Сброс по провалу питания	0x000000	0	0	0	0	0	0	0	0	1
/MCLR сброс в течении нормальной работы	0x000000	0	0	1	0	0	0	0	0	0
Программный сброс в течении нормальной работы	0x000000	0	0	0	1	0	0	0	0	0
/MCLR сброс в течении Спячки	0x000000	0	0	1	0	0	0	1	0	0
/MCLR сброс в течении Простоя	0x000000	0	0	1	0	0	1	0	0	0
WDT перерыв сброса	0x000000	0	0	0	0	1	0	0	0	0
WDT пробуждение	PC + 2	0	0	0	0	1	0	1	0	0
Прерывание пробуждающее из Спячки	PC + 2(1)	0	0	0	0	0	0	1	0	0
Ловушка отказа тактирования	0x000004	0	0	0	0	0	0	0	0	0
Ловушка сброса	0x000000	1	0	0	0	0	0	0	0	0
Ловушка не верной операции	0x000000	0	1	0	0	0	0	0	0	0

Описание: u = неизменный, x = неизвестный, - = не задействованный бит, читается как ‘0’

Примечание 1: Когда пробуждение происходит благодаря разрешённому прерывания, в PC загружается соответствующий вектор прерывания.

В таблице 19-6 показан второй пример состояния битов для регистра RCON. В этом случае, не принято чтобы пользователь устанавливал/очищал биты до действия указанного в столбце состояния.

Таблица 19-6: Состояние инициализации для регистра RCON в случае 2

Состояние	Программный счётчик	TRAPR	IOPUWR	EXTR	SWR	WDTO	IDLE	SLEEP	POR	BOR
Сброс на включение	0x000000	0	0	0	0	0	0	0	1	1
Сброс по провалу питания	0x000000	0	0	0	0	0	0	0	0	1
/MCLR сброс в течении нормальной работы	0x000000	0	0	1	0	0	0	0	0	0
Программный сброс в течении нормальной работы	0x000000	0	0	0	1	0	0	0	0	0
/MCLR сброс в течении Спячки	0x000000	0	0	1	0	0	0	1	0	0
/MCLR сброс в течении Простоя	0x000000	0	0	1	0	0	1	0	0	0
WDT перерыв сброса	0x000000	0	0	0	0	1	0	0	0	0
WDT пробуждение	PC + 2	0	0	0	0	1	0	1	0	0
Прерывание пробуждающее из Спячки	PC + 2(1)	0	0	0	0	0	0	1	0	0
Ловушка отказа тактирования	0x000004	0	0	0	0	0	0	0	0	0
Ловушка сброса	0x000000	1	0	0	0	0	0	0	0	0
Ловушка не верной операции	0x000000	0	1	0	0	0	0	0	0	0

Описание: u = неизменный, x = неизвестный, - = не задействованный бит, читается как ‘0’

Примечание 1: Когда пробуждение происходит благодаря разрешённому прерывания, в PC загружается соответствующий вектор прерывания.

19.4 Сторожевой таймер (WDT)

19.4.1 Работа сторожевого таймера

Первичной функцией сторожевого таймера (WDT) является сброс процессора в случае программного сбоя. WDT является свободно бегущим таймером, который работает от RC-генератора, расположенного на чипе, не требуя ни каких навесных компонентов. Следовательно, таймер WDT может продолжать работать в случае отказа тактирования (т.е. кварцевого генератора) главный процессора.

19.4.2 Разрешение и блокировка WDT

Сторожевой таймер может быть “разрешён” или “блокирован” только через бит конфигурации (FWDTEN) в регистре конфигурации FWDT.

Установка FWDTEN = 1 разрешает сторожевой таймер. Разрешение делается при программировании устройства. По умолчанию, после стирания чипа, FWDTEN bit = 1. Любое устройство программирования способное программировать устройства dsPIC30F позволяет программировать эти и другие биты конфигурации.

Если разрешён, WDT инкрементируется до переполнения или "таймаута". Таймаут вызовет сброс устройства (кроме в течении Спячки). Для предотвращения таймаута WDT, пользователь должен очищать сторожевой таймер используя инструкцию CLRWDT.

Если происходит таймаут WDT в течении Спячки, устройство пробуждается. Бит WDTO в регистре RCON должно быть очищено для индикации пробуждения в результате таймаута WDT.

Установка FWDTEN = 0 позволяет программе пользователя разрешить/блокировать сторожевой таймер через бит управления SWDTEN (RCON<5>).

19.5 Режимы экономичного потребления

Существует два режима экономичного потребления, которые могут быть введены через выполнение специальной инструкции, PWRSAV.

Это есть: Спячки и Простоя.

Формат инструкции PWRSAV такой как следует ниже:

PWRSAV <параметр>, где ‘параметр’ определяет режим Спячки или Простоя.

19.5.1 Режим Спячки

В режиме Спячки, тактирование CPU и периферии отключено. Если используется генератор на чипе, это завершение.

Монитор защиты от сбоя тактирования не работает в течении Спячки, поскольку там не тактируется монитор. Тем не менее, тактирование LPRC остаётся активным, если WDT работает в течении Спячки.

Если разрешены, схема защиты от провала питания и схема определения низкого напряжения, то продолжают функционировать в течении Спячки.

Процессор просыпается из Спячки если по крайней мере одно из следующих условий происходит:

- любое прерывание, которое индивидуально разрешено и удовлетворяет требуемому уровню приоритета
- любой сброс (POR, BOR и MCLR)
- таймаут WDT

На пробуждение из режима Спячки, процессор осуществляет рестарт с тем же самым тактированием, которое было активным перед входом в режим Спячки. Когда переключение тактирования разрешено, биты COSC<1:0> определяют генератор источник, который используется при пробуждении. Если переключение тактирования блокировано, то там только одно системное тактирование.

Примечание:	Если произошли POR или BOR, выбор генератора основан на битах конфигурации FOS<1:0> и FPR<3:0>.

Если источник тактирования есть генератор, тактирование устройства будет задерживаться до момента OST (индикация устойчивости генератора). Если используется PLL, системное тактирование будет задерживаться выключенным, пока LOCK = 1 (индикация, что PLL стабилен). В любом случае, TPOR, TLOCK и TPWRT задержки прилагаются.

Если EC, FRC, LPRC или ERC генераторы используются, то задержка TPOR (~ 10 µs) приложена. Это есть минимальная задержка возможная для пробуждения из Спячки.

Кроме того, если LP генератор активный в течении Спячки, и LP является генератором используемым при пробуждении, то задержка запуска будет равна TPOR. Задержка PWRT и время задержки OST не применены. В порядке получения минимальной возможной задержки запуска при пробуждении из Спячки, один из этих параметров пробуждения должен быть выбран перед входом в Спячку.

Любое прерывание, которое индивидуально разрешено (используя соответствующий бит IE) и обладает преобладающим уровнем приоритета в состоянии пробудить процессор. Процессор обрабатывает прерывание и переходит на ISR. Бит состояния Sleep(спячка) в регистре RCON установлен при пробуждении.

Примечание:	Не смотря на различные прилагаемые задержки (TPOR, TLOCK и TPWRT), кварцевый генератор (и PLL) могут не быть активными в конце таймаута (т.е., для низкочастотных кристаллов. В этих случаях), если FSCM разрешён, тогда устройство может определить это как отказ тактирования и обработает ловушки отказа тактирования, генератор FRC будет разрешён, и пользователь должен вновь давать возможность кварцевому генератору. Если FSCM не разрешён, то устройство просто приостановит выполнение кода, пока тактирование стабилизируется, и останется в Спячке пока тактовый генератор не стартует.

Все сбросы пробуждают процессор из режима Спячки. Любой сброс, за исключением POR, устанавливает бит состояния Sleep(спячка). В POR, бит состояния Sleep(спячка) очищен.

Если сторожевой таймер разрешён, тогда процессор пробуждается из режима Спячки с WDT задержкой. Оба бита Sleep(спячка) и WDTO устанавливаются.

19.5.2 Режим Простоя

В режиме Простоя, тактирование CPU выключено, пока периферия работает. В отличие от режима Спячки, тактовый источник остаётся активным.

Различные периферийные устройства имеют бит управления в каждом модуле, которые позволяют им работать в режиме Простоя.

LPRC защита тактирования остаётся активной если разрешено обнаружение отказа тактирования.

Процессор пробуждается из Простоя если справедливо по крайней мере одно из следующих условий:

• на любое прерывание, которое индивидуально разрешено (бит IE установлен в ‘1’) и имеет необходимый уровень приоритета

• на любой сброс (POR, BOR, MCLR)

• на WDT задержку

С пробуждением из режима Простоя, тактирование используется в CPU и немедленно начинается выполнение инструкций, стартуя с инструкции следующей за инструкцией PWRSAV.

Любой прерывание которое индивидуально разрешено (используя бит IE) и имеет необходимый уровень приоритета, в состоянии пробудить процессор. Процессор обработает прерывание и перейдёт к ISR. Бит состояния Idle(Простой) в регистре RCON устанавливается с пробуждением.

Любой сброс, за исключением POR, устанавливают бит состояния Idle(Простой). На POR, бит Idle очищен.

Если сторожевой таймер разрешён, тогда процессор пробуждается из режима простоя с WDT таймаутом. Оба бита состояния, Idle(Простой) и WDTO, устанавливаются.

В отличие от режима Спячки, здесь нет никаких запаздываний, включенных в пробуждение из режима Простоя.

19.6 Регистры конфигурации устройства

Биты конфигурации в каждом регистре конфигурации определяют некоторые режимы устройства и программируются программированием устройства, или используя внутрисхемное последовательное программирование (ICSP). Каждый регистр является 24-битным регистром, но только нижние 16 бит каждого регистра используются для хранения данных конфигурации. Имеются четыре регистра конфигурации устройства доступные пользователю:

1. FOSC (0xF80000): Регистр конфигурации генератора
2. FWDT (0xF80002): Регистр конфигурации сторожевого таймера
3. FBORPOR (0xF80004): Регистр конфигурации BOR и POR
4. FGS (0xF8000A): Регистр конфигурации общего кодового сегмента

Размещение битов конфигурации делается автоматически, когда вы выбираете устройство в вашем программисте устройства. Желаемое состояние битов конфигурации может быть определено в исходном коде (зависит от используемого инструментального языка), или через интерфейс программирования. После того, как устройство запрограммировано, прикладное программное обеспечение может читать значения бит конфигурации, через инструкции чтения таблицы. Для дополнительной информации пожайлуста обращайтесь к инструкциям по программированию устройства.

Примечание:	Если биты конфигурации защиты кода (FGS<GCP> и FGS<GWRP>) запрограммированы, возможно только полное стирание защищённого кодом устройства при напряжении VDD ≥ 4.5V.

19.7 Внутрисхемный отладчик

Когда в качестве отладчика выбран MPLAB® ICD2, разрешается функция внутрисхемной отладки. Эта функция позволяет просто осуществлять функции отладки при использовании MPLAB IDE. Когда разрешена эта функция устройства, некоторые ресурсы становятся недоступными для общего использования. Эти ресурсы включают первые 80 байт RAM данных и две ножки ввода/вывода.

Одна из четырёх пар ножек ввода/вывода может быть выбрана для использования, с использованием опции конфигурации в MPLAB IDE. Эти пары ноже называются EMUD/EMUC, EMUD1/EMUC1, EMUD2/EMUC2 и EMUD3/EMUC3.

В каждом случае, выбранная ножка EMUD будет линией Эмуляция/Данные отладки. Эти ножки обеспечивают интерфейс доступа к модулю MPLAB ICD 2 из Microchip. Выбранная пара ножек ввода/вывода отладки используется MPLAB

ICD 2 для посылки комманд и приёма ответов, а так же для посылки и приёма данных. Использующая функции внутрисхемной отладки устройства, разработка должна осуществлять связь с MCLR, VDD, VSS, PGC, PGD и выбранной парой ножек EMUDx/EMUCx.

Это вызывает две возможности:

1. Если EMUD/EMUC выбраны как ножки ввода/вывода отладки, тогда требуется только 5-ножечный интерфейс, поскольку функции ножек EMUD и EMUC мультиплексированы с функциями ножек PGD и PGC во всех устройствах dsPIC30F.

2. Если EMUD1/EMUC1, EMUD2/EMUC2 или EMUD3/EMUC3 выбраны как пары ножек ввода/вывода отладки, тогда требуется 7-ножечный интерфейс, поскольку функции ножек EMUDx/EMUCx (x = 1, 2 или 3) не мультиплексированы с функциями ножек PGD и PGC.

Таблица 19-7: Карта регистра системной интеграции

SFR Имя	Адрес	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Состояние после сброса
RCON	0740	TRAPR	IOPUWR	BGST	-	-	-	-	-	EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR	В зависимости от типа сброса
OSCCON	0742	TUN3	TUN2	COSC<1:0>		TUN1	TUN0	NOSC<1:0>		POST<1:0>		LOCK	—	CF	—	LPOSCEN	OSWEN	В зависимости от бит конфиг.

Описание: u = неинициализированный бит
Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

Таблица 19-8: Карта регистра конфигурации устройства

SFR Имя	Адрес	Bits 23-16	Бит 15	Бит 14	Бит 13	Бит 12	Бит 11	Бит 10	Бит 9	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
FOSC	F80000	—	FCKSM<1:0>		—	—	—	—	FOS<1:0>		—	—	—	—	FPR<3:0>			
FWDT	F80002	—	FWDTEN	—	—	—	—	—	—	—	—	—	FWPSA<1:0>		FWPSB<3:0>			
FBORPOR	F80004	—	MCLREN	—	—	—	—	PWMPIN	HPOL	LPOL	BOREN	—	BORV<1:0>		—	—	FPWRT<1:0>	
FGS	F8000A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	GCP	GWRP

Примечание: Обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046) для описания битовых полей регистра.

20.0 Краткое изложение системы команд

Примечание:	Это описание представляет общие характеристики группы устройств dsPIC30F и не претендует на полную информативность. Для большей информации на CPU, периферию, регистры и общее функционирование устройства, обратитесь к “Справочному руководству на семейство dsPIC30F” (DS70046). Для большей информации на систему команд устройства и программирование обратитесь к "Руководству программиста dsPIC30F/33F" (DS70157).

Система команд dsPIC30F сильно расширена по сравнению с системой команд PIC® MCU, но позволяет свободную миграцию с системы команд PIC MCU.

Большинство инструкций представляют из себя единственное слово памяти программ (24 бит). Только три инструкции требуют две позиции программной памяти.

Каждая однословная инструкция является 24-битным словом и разделена на 8-битный код операции, который определяет тип инструкции, и один или более операндов, которые определяют действие инструкции.

Система команд является высоко ортогональной и сгруппирована в пять основных категорий:

• Словно или байтно ориентированные операции
• Бит ориентированные операции
• Операции с литералом
• DSP операции
• Операции управления

В таблице 20-1 показаны основные символы, используемые в описываемых инструкциях.

Система команд dsPIC30F сведена в таблицу 20-2, включающую все инструкции вместе с флагами состояния, на которые влияет каждая инструкция.

Большинство словно или байтно ориентированных инструкций регистра W (включая инструкции барабанного сдвига) имеют три операнда:

• Первый операнд источник, которым обычно является регистр ‘Wb’ без какого-то адресного модификатора
• Второй операнд источник, которым обычно является регистр ‘Ws’ с или без адресного модификатора
• Место назначения результата, которым обычно является регистр ‘Wd’ с или без адресного модификатора

Тем не менее, словно или байтно ориентированные инструкции файловых регистров имеют два операнда:

• Файловый регистр определённый значением ‘f’
• Место назначения, которым также может быть файловый регистр ‘f’ или регистр W0, который обозначен как ‘WREG’.

Большинство бит ориентированных инструкций (включая простые инструкции вращения/сдвига) имеют два операнда:

• Регистр W (с или без адресного модификатора) или файловый регистр (определяется значением ‘Ws’ или ‘f’)
• Бит в регистре W или файловом регистре (определён значением литерала, или косвенно содержимым регистра ‘Wb’)

Литеральные инструкции, которые включают перемещение данных могут использовать некоторые из следующих операндов:

• Значение литерала загружено в регистр W или файловый регистр (определённый через значение ‘k’)
• Регистр W или файловый регистр, куда значение литерала было загружено (определён через ‘Wb’ или ‘f’)

Тем не менее, литеральная инструкция, которая включает арифметические или логические операции использует некоторые из следующих операндов:

• Первый операнд источник, которым является регистр ‘Wb’ без какого-то адресного модификатора
• Второй операнд источник, который является значением литерала
• Место назначения результата (только если не тоже самое, как первый операнд источник), которое обычно является регистром ‘Wd’ с или без адресного модификатора

MAC класс инструкций DSP может использовать некоторые из следующих операндов:

• Аккумулятор (A или B) будет использован (требуемый операнд)
• Регистр W будет использован как два операнда
• Операции упреждающей выборки адресного пространства X и Y
• Назначения упреждающей выборки адресного пространства X и Y
• Аккумулятор назначения обратной записи

Другие инструкции DSP не включают любые умножения, и могут включать:

• Аккумулятор, который нужно использовать (требуемый)
• операнд источник или приёмник (определён как Wso или Wdo, соответственно) с или без адресного модификатора
• Количество сдвигов, определённое регистром W ‘Wn’ или значением литерала

Инструкции управления могут использовать некоторые из следующих операндов:

• Адреса программной памяти
• Инструкции режима чтения и записи таблицы

Все инструкции имеют длину в одно слово, за исключением некоторых инструкций, которые были сделаны такими с тем, чтобы вся необходимая информация была доступна в этих 48 битах. Во втором слове 8 MSbs равны ‘0’. Если это второе слово будет выполнена как инструкция (отдельно), то будет выполнено как NOP.

Большинство инструкций длиной в слово выполняются в одном цикле инструкции, за исключением условных при истинном тесте или изменение программного счётчика как результат инструкции. В этих случаях инструкция требует два цикла инструкции с дополнительным циклом(и) инструкции выполняемыми как NOP. Известные исключения являются BRA (безусловный/вычисленный переход), косвенный CALL/GOTO, все табличные чтения и записи и инструкции RETURN/RETFIE, которые являются инструкциями длиной в слово, но требуют два или три цикла. Определённые инструкции, которые вызывают пропуск последующей инструкции, также требуют двух или трёх циклов если выполняется пропуск, в зависимости от того на которой инструкции, однословной или двухсловной, делается пропуск. Кроме того, перемещение двойного слова требует двух циклов. Двухсловные инструкции выполняются в два цикла инструкции.

Примечание:	Для больших деталей на систему команд обращайтесь к “Справочному руководству программиста dsPIC30F/33F” (DS70157).

Таблица 20-1: Символы используемые в описании кодов операций

Область	Описание
#text	Означает литерал определённый как “текст“
(text)	Означает “содержимое текста“
[text]	Означает “позицию адресованную текстом”
{ }	Необязательное поле или операция
<n:m>	Битовая область регистра
.b	Выбор байтного режима
.d	Выбор режима двойного слова
.S	Выбор теневого регистра
.w	Выбор словного режима (по умолчанию)
Acc	Один из двух аккумуляторов {A, B}
AWB	Аккумулятор обратной записи назначения адреса регистра ∈ {W13, [W13] + = 2}
bit4	4-разрядное поле выбора бита (используется в инструкциях адресации слова) ∈ {0...15}
C, DC, N, OV, Z	Биты состояния MCU: (C)перенос, (DC)цифровой перенос, (N)отрицательный, (O)переполнение, (Z)ноль
Expr	Абсолютный адрес, метка или выражение (решенный компоновщиком)
f	Адрес файлового регистра ∈ {0x0000...0x1FFF}
lit1	1-битный без знаковый литерал ∈ {0,1}
lit4	4- битный без знаковый литерал ∈ {0...15}
lit5	5- битный без знаковый литерал ∈ {0...31}
lit8	8- битный без знаковый литерал ∈ {0...255}
lit10	10- битный без знаковый литерал ∈ {0...255} для байтного режима, {0:1023} для словного режима
lit14	14- битный без знаковый литерал ∈ {0...16384}
lit16	16- битный без знаковый литерал ∈ {0...65535}
lit23	23- битный без знаковый литерал ∈ {0...8388608}; LSB must be 0
None	Поле не требует ввода, может быть пустым
OA, OB, SA, SB	Биты состояния DSP: ACCA переполнение, ACCB переполнение, ACCA насыщение, ACCB насыщение
PC	Программный счётчик
Slit10	10-битный знаковый литерал ∈ {-512...511}
Slit16	16- битный знаковый литерал ∈ {-32768...32767}
Slit6	6- битный знаковый литерал ∈ {-16...16}

Таблица 20-1: Символы используемые в описании кодов операций (продолжение)

Область	Описание
Wb	Основной W регистр ∈ {W0..W15}
Wd	Регистр W места назначения ∈ { Wd, [Wd], [Wd++], [Wd--], [++Wd], [--Wd] }
Wdo	Регистр W места назначения ∈ { Wnd, [Wnd], [Wnd++], [Wnd--], [++Wnd], [--Wnd], [Wnd+Wb] }
Wm,Wn	Рабочая регистровая пара делимого и делителя (прямая адресация)
Wm*Wm	Рабочая регистровая пара множимого и множителя для квадратных инструкций ∈ {W4 * W4,W5 * W5,W6 * W6,W7 * W7}
Wm*Wn	Рабочая регистровая пара множимого и множителя для DSP инструкций ∈ {W4 * W5,W4 * W6,W4 * W7,W5 * W6,W5 * W7,W6 * W7}
Wn	Один из 16 рабочих регистров ∈ {W0..W15}
Wnd	Один из 16 рабочих регистров места назначения ∈ {W0..W15}
Wns	Один из 16 рабочих регистров источников ∈ {W0..W15}
WREG	W0 (рабочий регистр используемый в инструкциях файловых регистров)
Ws	Регистр источник W ∈ { Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws] }
Wso	Регистр источник W ∈ { Wns, [Wns], [Wns++], [Wns--], [++Wns], [--Wns], [Wns+Wb] }
Wx	Пространство данных X инструкций упреждающей выборки адрес регистра для DSP ∈ {[W8] + = 6, [W8] + = 4, [W8] + = 2, [W8], [W8] - = 6, [W8] - = 4, [W8] - = 2, [W9] + = 6, [W9] + = 4, [W9] + = 2, [W9], [W9] - = 6, [W9] - = 4, [W9] - = 2, [W9 + W12],none}
Wxd	Пространство данных X инструкций упреждающей выборки назначения для DSP ∈ {W4..W7}
Wy	Пространство данных Y инструкций упреждающей выборки адрес регистра для DSP ∈ {[W10] + = 6, [W10] + = 4, [W10] + = 2, [W10], [W10] - = 6, [W10] - = 4, [W10] - = 2, [W11] + = 6, [W11] + = 4, [W11] + = 2, [W11], [W11] - = 6, [W11] - = 4, [W11] - = 2, [W11 + W12], none}
Wyd	Пространство данных Y инструкций упреждающей выборки назначения для DSP ∈ {W4..W7}

Таблица 20-2: Обзор системы команд

Номер инстр.	Ассебл. Мнемон.	Ассемблерный синтаксис		Описание	слов	циклов	Влияет на флаги
1	ADD	ADD	Acc	Сложить аккумуляторв	1	1	OA,OB,SA,SB
		ADD	f	f = f + WREG	1	1	C,DC,N,OV,Z
		ADD	f,WREG	WREG = f + WREG	1	1	C,DC,N,OV,Z
		ADD	#lit10,Wn	Wd = lit10 + Wd	1	1	C,DC,N,OV,Z
		ADD	Wb,Ws,Wd	Wd = Wb + Ws	1	1	C,DC,N,OV,Z
		ADD	Wb,#lit5,Wd	Wd = Wb + lit5	1	1	C,DC,N,OV,Z
		ADD	Wso,#Slit4,Acc	16-битное знаковое добавить в аккумулятор	1	1	OA,OB,SA,SB
2	ADDC	ADDC	f	f = f + WREG + (C)	1	1	C,DC,N,OV,Z
		ADDC	f,WREG	WREG = f + WREG + (C)	1	1	C,DC,N,OV,Z
		ADDC	#lit10,Wn	Wd = lit10 + Wd + (C)	1	1	C,DC,N,OV,Z
		ADDC	Wb,Ws,Wd	Wd = Wb + Ws + (C)	1	1	C,DC,N,OV,Z
		ADDC	Wb,#lit5,Wd	Wd = Wb + lit5 + (C)	1	1	C,DC,N,OV,Z
3	AND	AND	f	f = f .AND. WREG	1	1	N,Z
		AND	f,WREG	WREG = f .AND. WREG	1	1	N,Z
		AND	#lit10,Wn	Wd = lit10 .AND. Wd	1	1	N,Z
		AND	Wb,Ws,Wd	Wd = Wb .AND. Ws	1	1	N,Z
		AND	Wb,#lit5,Wd	Wd = Wb .AND. lit5	1	1	N,Z
4	ASR	ASR	f	f = Правый арифметический сдвиг f	1	1	C,N,OV,Z
		ASR	f,WREG	WREG = Правый арифметический сдвиг f	1	1	C,N,OV,Z
		ASR	Ws,Wd	Wd = Правый арифметический сдвиг Ws	1	1	C,N,OV,Z
		ASR	Wb,Wns,Wnd	Wnd = Правый арифметический сдвиг Wb к Wns	1	1	N,Z
		ASR	Wb,#lit5,Wnd	Wnd = Правый арифметический сдвиг Wb к lit5	1	1	N,Z
5	BCLR	BCLR	f,#bit4	Очистить бит f	1	1	Никакой
		BCLR	Ws,#bit4	Очистить бит Ws	1	1	Никакой
6	BRA	BRA	C,Expr	Переход, если Перенос	1	1(2)	Никакой
		BRA	GE,Expr	Переход, если больше или равно	1	1(2)	Никакой
		BRA	GEU,Expr	Переход, если без знака больше или равно	1	1(2)	Никакой
		BRA	GT,Expr	Переход, если больше	1	1(2)	Никакой
		BRA	GTU,Expr	Переход, если без знака больше	1	1(2)	Никакой
		BRA	LE,Expr	Переход, если меньше или равно	1	1(2)	Никакой
		BRA	LEU,Expr	Переход, если без знака меньше или равно	1	1(2)	Никакой
		BRA	LT,Expr	Переход, если меньше	1	1(2)	Никакой
		BRA	LTU,Expr	Переход, если без знака меньше	1	1(2)	Никакой
		BRA	N,Expr	Переход, если отрицательное	1	1(2)	Никакой
		BRA	NC,Expr	Переход, если нет переноса	1	1(2)	Никакой
		BRA	NN,Expr	Переход, если не отрицательное	1	1(2)	Никакой
		BRA	NOV,Expr	Переход, если нет переполнения	1	1(2)	Никакой
		BRA	NZ,Expr	Переход, если не ноль	1	1(2)	Никакой
		BRA	OA,Expr	Переход, если переполнение аккумулятора A	1	1(2)	Никакой
		BRA	OB,Expr	Переход, если переполнение аккумулятора B	1	1(2)	Никакой
		BRA	OV,Expr	Переход, если переполнение	1	1(2)	Никакой
		BRA	SA,Expr	Переход, если аккумулятор A насыщен	1	1(2)	Никакой
		BRA	SB,Expr	Переход, если аккумулятор B насыщен	1	1(2)	Никакой
		BRA	Expr	Безоговорочный переход	1	2	Никакой
		BRA	Z,Expr	Переход, если ноль	1	1(2)	Никакой
		BRA	Wn	Вычисленный переход	1	2	Никакой

Таблица 20-2: Обзор системы команд (продолжение)

Номер инстр.	Ассебл. Мнемон.	Ассемблерный синтаксис		Описание	слов	циклов	Влияет на флаги
7	BSET	BSET	f,#bit4	Установить бит f	1	1	Никакой
		BSET	Ws,#bit4	Установить бит Ws	1	1	Никакой
8	BSW	BSW.C	Ws,Wb	Записать бит C в Ws<Wb>	1	1	Никакой
		BSW.Z	Ws,Wb	Записать бит Z в Ws<Wb>	1	1	Никакой
9	BTG	BTG	f,#bit4	Переключить бит f	1	1	Никакой
		BTG	Ws,#bit4	Переключить бит Ws	1	1	Никакой
10	BTSC	BTSC	f,#bit4	Проверить бит f и пропустить если очищен	1	1 (2 или 3)	Никакой
		BTSC	Ws,#bit4	Проверить бит Ws и пропустить если очищен	1	1 (2 или 3)	Никакой
11	BTSS	BTSS	f,#bit4	Проверить бит f и пропустить если установлен	1	1 (2 или 3)	Никакой
		BTSS	Ws,#bit4	Проверить бит Ws и пропустить если установлен	1	1 (2 или 3)	Никакой
12	BTST	BTST	f,#bit4	Проверить бит f	1	1	Z
		BTST.C	Ws,#bit4	Бит Ws проверить в C	1	1	C
		BTST.Z	Ws,#bit4	Бит Ws проверить в Z	1	1	Z
		BTST.C	Ws,Wb	Бит Ws<Wb> проверить в C	1	1	C
		BTST.Z	Ws,Wb	Бит Ws<Wb> проверить в Z	1	1	Z
13	BTSTS	BTSTS	f,#bit4	Бит f проверить, затем установить	1	1	Z
		BTSTS.C	Ws,#bit4	Бит Ws проверить в C, затем установить	1	1	C
		BTSTS.Z	Ws,#bit4	Бит Ws проверить в Z, затем установить	1	1	Z
14	CALL	CALL	lit23	Вызвать подпрограмму	2	2	Никакой
		CALL	Wn	Вызвать косвенно подпрограмму	1	2	Никакой
15	CLR	CLR	f	f = 0x0000	1	1	Никакой
		CLR	WREG	WREG = 0x0000	1	1	Никакой
		CLR	Ws	Ws = 0x0000	1	1	Никакой
		CLR	Acc,Wx,Wxd, Wy,Wyd,AWB	Очистить аккумулятор	1	1	OA,OB,SA,SB
16	CLRWDT	CLRWDT		Очистить сторожевой таймер	1	1	WDTO,Sleep
17	COM	COM	f	f = /f	1	1	N,Z
		COM	f,WREG	WREG = /f	1	1	N,Z
		COM	Ws,Wd	Wd = /Ws	1	1	N,Z
18	CP	CP	f	Сравнить f с WREG	1	1	C,DC,N,OV,Z
		CP	Wb,#lit5	Сравнить Wb с lit5	1	1	C,DC,N,OV,Z
		CP	Wb,Ws	Сравнить Wb с Ws (Wb - Ws)	1	1	C,DC,N,OV,Z
19	CP0	CP0	f	Сравнить f с 0x0000	1	1	C,DC,N,OV,Z
		CP0	Ws	Сравнить Ws с 0x0000	1	1	C,DC,N,OV,Z
20	CPB	CPB	f	Сравнить f с WREG, с заёмом	1	1	C,DC,N,OV,Z
		CPB	Wb,#lit5	Сравнить Wb с lit5, с заёмом	1	1	C,DC,N,OV,Z
		CPB	Wb,Ws	Сравнить Wb с Ws, с заёмом (Wb - Ws - /C)	1	1	C,DC,N,OV,Z
21	CPSEQ	CPSEQ	Wb, Wn	Сравнить Wb с Wn, пропустить если =	1	1 (2 или 3)	Никакой
22	CPSGT	CPSGT	Wb, Wn	Сравнить Wb с Wn, пропустить если >	1	1 (2 или 3)	Никакой
23	CPSLT	CPSLT	Wb, Wn	Сравнить Wb с Wn, пропустить если <	1	1 (2 или 3)	Никакой
24	CPSNE	CPSNE	Wb, Wn	Сравнить Wb с Wn, пропустить если <>	1	1 (2 или 3)	Никакой
25	DAW	DAW	Wn	Wn = десятичная настройка Wn	1	1	C

Таблица 20-2: Обзор системы команд (продолжение)

Номер инстр.	Ассебл. Мнемон.	Ассемблерный синтаксис		Описание	слов	циклов	Влияет на флаги
26	DEC	DEC	f	f = f - 1	1	1	C,DC,N,OV,Z
		DEC	f,WREG	WREG = f -1	1	1	C,DC,N,OV,Z
		DEC	Ws,Wd	Wd = Ws - 1	1	1	C,DC,N,OV,Z
27	DEC2	DEC2	f	f = f - 2	1	1	C,DC,N,OV,Z
		DEC2	f,WREG	WREG = f -2	1	1	C,DC,N,OV,Z
		DEC2	Ws,Wd	Wd = Ws - 2	1	1	C,DC,N,OV,Z
28	DISI	DISI	#lit14	Блокировать прерывания для k циклов инструкций	1	1	Никакой
29	DIV	DIV.S	Wm,Wn	Деление со знаком 16/16-битного целого	1	18	N,Z,C, OV
		DIV.SD	Wm,Wn	Деление со знаком 32/16-битного целого	1	18	N,Z,C, OV
		DIV.U	Wm,Wn	Деление без знакового 16/16-битного целого	1	18	N,Z,C, OV
		DIV.UD	Wm,Wn	Деление без знакового 32/16-битного целого	1	18	N,Z,C, OV
30	DIVF	DIVF	Wm,Wn	Деление со знаком 16/16-битной дроби	1	18	N,Z,C, OV
31	DO	DO	#lit14,Expr	Делать код в PC+Expr, lit14 + 1 раз	2	2	Никакой
		DO	Wn,Expr	Делать код в PC+Expr, (Wn) + 1 раз	2	2	Никакой
32	ED	ED	Wm*Wm,Acc, Wx,Wy,Wxd	Евклидово расстояние (не аккумулируется)	1	1	OA,OB,OAB, SA,SB,SAB
33	EDAC	EDAC	Wm*Wm,Acc, Wx,Wy,Wxd	Евклидово расстояние	1	1	OA,OB,OAB, SA,SB,SAB
34	EXCH	EXCH	Wns,Wnd	Обменять Wns с Wnd	1	1	Никакой
35	FBCL	FBCL	Ws,Wnd	Найти изменение бита с левой (MSb) стороны	1	1	C
36	FF1L	FF1L	Ws,Wnd	Найти первую единицу с левой (MSb) стороны	1	1	C
37	FF1R	FF1R	Ws,Wnd	Найти первую единицу с правой (LSb) стороны	1	1	C
38	GOTO	GOTO	Expr	Перейти на адрес	2	2	Никакой
		GOTO	Wn	Перейти косвенно	1	2	Никакой
39	INC	INC	f	f = f + 1	1	1	C,DC,N,OV,Z
		INC	f,WREG	WREG = f + 1	1	1	C,DC,N,OV,Z
		INC	Ws,Wd	Wd = Ws + 1	1	1	C,DC,N,OV,Z
40	INC2	INC2	f	f = f + 2	1	1	C,DC,N,OV,Z
		INC2	f,WREG	WREG = f + 2	1	1	C,DC,N,OV,Z
		INC2	Ws,Wd	Wd = Ws + 2	1	1	C,DC,N,OV,Z
41	IOR	IOR	f	f = f .ИЛИ. WREG	1	1	N,Z
		IOR	f,WREG	WREG = f .ИЛИ. WREG	1	1	N,Z
		IOR	#lit10,Wn	Wd = lit10 .ИЛИ. Wd	1	1	N,Z
		IOR	Wb,Ws,Wd	Wd = Wb .ИЛИ. Ws	1	1	N,Z
		IOR	Wb,#lit5,Wd	Wd = Wb .ИЛИ. lit5	1	1	N,Z
42	LAC	LAC	Wso,#Slit4,Acc	Загрузить аккумулятор	1	1	OA,OB,OAB, SA,SB,SAB
43	LNK	LNK	#lit14	Указатель связи фрейма	1	1	Никакой
44	LSR	LSR	f	f = Логический сдвиг вправо f	1	1	C,N,OV,Z
		LSR	f,WREG	WREG = Логический сдвиг вправо f	1	1	C,N,OV,Z
		LSR	Ws,Wd	Wd = Логический сдвиг вправо Ws	1	1	C,N,OV,Z
		LSR	Wb,Wns,Wnd	Wnd = Логический сдвиг вправо Wb к Wns	1	1	N,Z
		LSR	Wb,#lit5,Wnd	Wnd = Логический сдвиг вправо Wb к lit5	1	1	N,Z
45	MAC	MAC	Wm*Wn,Acc,Wx, Wxd,Wy,Wyd, AWB	Умножение и накопление	1	1	OA,OB,OAB, SA,SB,SAB
		MAC	Wm*Wm,Acc, Wx,Wxd,Wy,Wyd	Возведение в квадрат и накопление	1	1	OA,OB,OAB, SA,SB,SAB

Таблица 20-2: Обзор системы команд (продолжение)

Номер инстр.	Ассебл. Мнемон.	Ассемблерный синтаксис		Описание	слов	циклов	Влияет на флаги
46	MOV	MOV	f,Wn	Переместить f в Wn	1	1	Никакой
		MOV	f	Переместить f в f	1	1	N,Z
		MOV	f,WREG	Переместить f в WREG	1	1	N,Z
		MOV	#lit16,Wn	Переместить 16-битный литерал в Wn	1	1	Никакой
		MOV.b	#lit8,Wn	Переместить 8-битный литерал в Wn	1	1	Никакой
		MOV	Wn,f	Переместить Wn в f	1	1	Никакой
		MOV	Wso,Wdo	Переместить Ws в Wd	1	1	Никакой
		MOV	WREG,f	Переместить WREG в f	1	1	N,Z
		MOV.d	Wns,Wd	Переместить двойню из W(ns):W(ns+1) в Wd	1	2	Никакой
		MOV.d	Ws,Wnd	Переместить двойню из Ws в W(nd+1):W(nd)	1	2	Никакой
47	MOVSAC	MOVSAC	Acc,Wx,Wxd, Wy,Wyd,AWB	Выборка и сохранение аккумулятора	1	1	Никакой
48	MPY	MPY	Wm*Wn,Acc, Wx,Wxd,Wy,Wyd	Произведение Wm и Wn в аккумулятор	1	1	OA,OB,OAB, SA,SB,SAB
		MPY	Wm*Wm,Acc, Wx,Wxd,Wy,Wyd	Квадрат Wm в аккумулятор	1	1	OA,OB,OAB, SA,SB,SAB
49	MPY.N	MPY.N	Wm*Wn,Acc,Wx, Wxd,Wy,Wyd	-(Произведение Wm и Wn) в аккумулятор	1	1	Никакой
50	MSC	MSC	Wm*Wm,Acc, Wx,Wxd,Wy, Wyd,AWB	Умножить и вычесть из аккумулятора	1	1	OA,OB,OAB, SA,SB,SAB
51	MUL	MUL.SS	Wb,Ws,Wnd	{Wnd + 1, Wnd} = знаковый(Wb) * знаковый(Ws)	1	1	Никакой
		MUL.SU	Wb,Ws,Wnd	{Wnd + 1, Wnd} = знаковый(Wb) * без знаковый(Ws)	1	1	Никакой
		MUL.US	Wb,Ws,Wnd	{Wnd + 1, Wnd} = без знаковый(Wb) * знаковый(Ws)	1	1	Никакой
		MUL.UU	Wb,Ws,Wnd	{Wnd + 1, Wnd} = без знаковый(Wb) * без знаковый(Ws)	1	1	Никакой
		MUL.SU	Wb,#lit5,Wnd	{Wnd + 1, Wnd} = знаковый(Wb) * без знаковый(lit5)	1	1	Никакой
		MUL.UU	Wb,#lit5,Wnd	{Wnd + 1, Wnd} = без знаковый(Wb) * без знаковый(lit5)	1	1	Никакой
		MUL	f	W3:W2 = f * WREG	1	1	Никакой
52	NEG	NEG	Acc	Изменить знак аккумулятора	1	1	OA,OB,OAB, SA,SB,SAB
		NEG	f	f = /f + 1	1	1	C,DC,N,OV,Z
		NEG	f,WREG	WREG = /f + 1	1	1	C,DC,N,OV,Z
		NEG	Ws,Wd	Wd = /Ws + 1	1	1	C,DC,N,OV,Z
53	NOP	NOP		Нет операции	1	1	Никакой
		NOPR		Нет операции	1	1	Никакой
54	POP	POP	f	Вытолкнуть f из вершины стека (TOS)	1	1	Никакой
		POP	Wdo	Вытолкнуть из вершины стека (TOS) в Wdo	1	1	Никакой
		POP.D	Wnd	Вытолкнуть из вершины стека (TOS) в W(nd):W(nd + 1)	1	2	Никакой
		POP.S		Вытолкнуть теневые регистры	1	1	Все
55		PUSH	f	Втолкнуть f в вершину стека (TOS)	1	1	Никакой
		PUSH	Wso	Втолкнуть Wso в вершину стека (TOS)	1	1	Никакой
		PUSH.D	Wns	Втолкнуть W(ns):W(ns + 1) в вершину стека (TOS)	1	2	Никакой
		PUSH.S		Втолкнуть теневые регистры	1	1	Никакой
56	PWRSAV	PWRSAV	#lit1	Идти в режим спячки или простоя	1	1	WDTO,Sleep

Таблица 20-2: Обзор системы команд (продолжение)

Номер инстр.	Ассебл. Мнемон.	Ассемблерный синтаксис		Описание	слов	циклов	Влияет на флаги
57	RCALL	RCALL	Expr	Относительный вызов	1	2	Никакой
		RCALL	Wn	Вычисленный вызов	1	2	Никакой
58	REPEAT	REPEAT	#lit14	Повторить инструкцию Next lit14 + 1 раз	1	1	Никакой
		REPEAT	Wn	Повторить инструкцию Next (Wn) + 1 раз	1	1	Никакой
59	RESET	RESET		Программный сброс устройства	1	1	Никакой
60	RETFIE	RETFIE		Возврат из прерывания	1	3 (2)	Никакой
61	RETLW	RETLW	#lit10,Wn	Возврат с литералом в Wn	1	3 (2)	Никакой
62	RETURN	RETURN		Возврат из подпрограммы	1	3 (2)	Никакой
63	RLC	RLC	f	f = Вращать влево через перенос f	1	1	C,N,Z
		RLC	f,WREG	WREG = Вращать влево через перенос f	1	1	C,N,Z
64	RLNC	RLNC	f	f = Вращать влево (не через перенос) f	1	1	N,Z
		RLNC	f,WREG	WREG = Вращать влево (не через перенос) f	1	1	N,Z
		RLNC	Ws,Wd	Wd = Вращать влево (не через перенос) Ws	1	1	N,Z
65	RRC	RRC	f	f = Вращать вправо через перенос f	1	1	C,N,Z
		RRC	f,WREG	WREG = Вращать вправо через перенос f	1	1	C,N,Z
		RRC	Ws,Wd	Wd = Вращать вправо через перенос Ws	1	1	C,N,Z
66	RRNC	RRNC	f	f = Вращать вправо (не через перенос) f	1	1	N,Z
		RRNC	f,WREG	WREG = Вращать вправо (не через перенос) f	1	1	N,Z
		RRNC	Ws,Wd	Wd = Вращать вправо (не через перенос) Ws	1	1	N,Z
67	SAC	SAC	Acc,#Slit4,Wdo	Хранить аккумулятор	1	1	Никакой
		SAC.R	Acc,#Slit4,Wdo	Хранить округлённый аккумулятор	1	1	Никакой
68	SE	SE	Ws,Wnd	Wnd = расширенный знаком Ws	1	1	C,N,Z
69	SETM	SETM	f	f = 0xFFFF	1	1	Никакой
		SETM	WREG	WREG = 0xFFFF	1	1	Никакой
		SETM	Ws	Ws = 0xFFFF	1	1	Никакой
70	SFTAC	SFTAC	Acc,Wn	Арифметический сдвиг аккумулятора на (Wn)	1	1	OA,OB,OAB, SA,SB,SAB
		SFTAC	Acc,#Slit6	Арифметический сдвиг аккумулятора на Slit6	1	1	OA,OB,OAB, SA,SB,SAB
71	SL	SL	f	f = Левый сдвиг f	1	1	C,N,OV,Z
		SL	f,WREG	WREG = Левый сдвиг Shift f	1	1	C,N,OV,Z
		SL	Ws,Wd	Wd = Левый сдвиг Ws	1	1	C,N,OV,Z
		SL	Wb,Wns,Wnd	Wnd = Левый сдвиг Wb на Wns	1	1	N,Z
		SL	Wb,#lit5,Wnd	Wnd = Левый сдвиг Wb на lit5	1	1	N,Z
72	SUB	SUB	Acc	Вычитать аккумуляторы	1	1	OA,OB,OAB, SA,SB,SAB
		SUB	f	f = f - WREG	1	1	C,DC,N,OV,Z
		SUB	f,WREG	WREG = f - WREG	1	1	C,DC,N,OV,Z
		SUB	#lit10,Wn	Wn = Wn - lit10	1	11	C,DC,N,OV,Z
		SUB	Wb,Ws,Wd	Wd = Wb - Ws	1	1	C,DC,N,OV,Z
		SUB	Wb,#lit5,Wd	Wd = Wb - lit5	1	1	C,DC,N,OV,Z
73	SUBB	SUBB	f	f = f - WREG - (/C)	1	1	C,DC,N,OV,Z
		SUBB	f,WREG	WREG = f - WREG - (/C)	1	1	C,DC,N,OV,Z
		SUBB	#lit10,Wn	Wn = Wn - lit10 - (/C)	1	1	C,DC,N,OV,Z
		SUBB	Wb,Ws,Wd	Wd = Wb - Ws - (/C)	1	1	C,DC,N,OV,Z
		SUBB	Wb,#lit5,Wd	Wd = Wb - lit5 - (/C)	1	1	C,DC,N,OV,Z

Таблица 20-2: Обзор системы команд (продолжение)

Номер инстр.	Ассебл. Мнемон.	Ассемблерный синтаксис		Описание	слов	циклов	Влияет на флаги
74	SUBR	SUBR	f	f = WREG - f	1	1	C,DC,N,OV,Z
		SUBR	f,WREG	WREG = WREG - f	1	1	C,DC,N,OV,Z
		SUBR	Wb,Ws,Wd	Wd = Ws - Wb	1	1	C,DC,N,OV,Z
		SUBR	Wb,#lit5,Wd	Wd = lit5 - Wb	1	1	C,DC,N,OV,Z
75	SUBBR	SUBBR	f	f = WREG - f - (/C)	1	1	C,DC,N,OV,Z
		SUBBR	f,WREG	WREG = WREG - f - (/C)	1	1	C,DC,N,OV,Z
		SUBBR	Wb,Ws,Wd	Wd = Ws - Wb - (/C)	1	1	C,DC,N,OV,Z
		SUBBR	Wb,#lit5,Wd	Wd = lit5 - Wb - (/C)	1	1	C,DC,N,OV,Z
76	SWAP	SWAP.b	Wn	Wn = обмен ниблами Wn	1	1	Никакой
		SWAP	Wn	Wn = обмен байтами Wn	1	1	Никакой
77	TBLRDH	TBLRDH	Ws,Wd	Читать Prog<23:16> в Wd<7:0>	1	2	Никакой
78	TBLRDL	TBLRDL	Ws,Wd	Читать Prog<15:0> в Wd	1	2	Никакой
79	TBLWTH	TBLWTH	Ws,Wd	Записать Ws<7:0> в Prog<23:16>	1	2	Никакой
80	TBLWTL	TBLWTL	Ws,Wd	Записать Ws в Prog<15:0>	1	2	Никакой
81	ULNK	ULNK		Расцепить указатель фрейма	1	1	Никакой
82	XOR	XOR	f	f = f .XOR. WREG	1	1	N,Z
		XOR	f,WREG	WREG = f .XOR. WREG	1	1	N,Z
		XOR	#lit10,Wn	Wd = lit10 .XOR. Wd	1	1	N,Z
		XOR	Wb,Ws,Wd	Wd = Wb .XOR. Ws	1	1	N,Z
		XOR	Wb,#lit5,Wd	Wd = Wb .XOR. lit5	1	1	N,Z
83	ZE	ZE	Ws,Wnd	Wnd = Расширенный нолём Ws	1	1	C,Z,N