Перевод проекта с контроллером PIC16xxx на контроллер PIC18xxx2.

При появлении контроллеров расширенного семейства PIC18Fxx2 встает вопрос о модификации существующих программ под новые контроллеры. Преимущества контроллеров расширенного семейства PIC18Fxx2 очевидны:

- увеличенный объем flash-памяти программ до 32КБайт (16Кслов), ОЗУ и памяти данных;
- приоритеты прерываний;
- аппаратный умножитель 8*8;
- возможность подключения второго кварца к Таймеру1 и Таймеру3;
- повышенное разрешение модулей захвата (до 6.25нс) и сравнения (до 100нс);
- программируемые детектор пониженного напряжения (с возможностью прерывания) и генератор сброса;
- встроенный умножитель частоты (PLL), частота до 40МГц; возможность переключения на вспомогательный генератор тактовой частоты;
- меньшая стоимость.

ВВЕДЕНИЕ.

При разработке микроконтроллеров семейства PIC18xxx2 уделялось особое внимание, чтобы переход от микроконтроллеров среднего семейства к современным микроконтроллерам PIC18xxx2 выполнялся настолько легко, насколько это возможно: изменения названий регистров и отдельных битов, а также смена расположения битов были сведены к минимуму. Микроконтроллеры семейства PIC18xxx2 совместимы по выводам с контроллерами среднего семейства, которые имеют корпус с 28 и 40 выводами.

В данном документе описаны минимальные изменения, необходимые для перевода инженерного проекта с микроконтроллера среднего семейства (например, PIC16C74A) на более продвинутый контроллер (например, PIC18C442).

Изменения в основном касаются переименования регистров и битов, переноса битов в новые регистры и размещения переменных в соответствующих областях оперативной памяти. Дополнительные изменения отмечены особо: предложены программные решения.

Примечание: *при переходе к новому семейству не следует изменять биты, не определённые в микроконтроллерах семейства PIC16xxx, по крайней мере, до тех пор, пока не будут досконально известны последствия этих изменений.*

ТАКТИРОВАНИЕ.

Может потребоваться изменение схемы автогенератора, либо изменение конфигурации устройства в соответствии с режимом тактирования. В любом случае следует удостовериться в том, что автогенератор запускается и работает как надо.

Возможно, в кварцевом автогенераторе придётся заменить навесные конденсаторы и/или изменить режим тактирования. При использовании прежних номиналов совместно с PIC18xxx2 возможна ситуация, когда RC-автогенератор будет работать на иной частоте, нежели можно было ожидать.

Примечание 1: *несмотря на идентичность условий проверки совместимых устройств (микроконтроллеров), характеристики этих устройств могут отличаться вследствие неидеальных условий производства. Эти изменения не окажут влияния на правильно спроектированную систему, то есть когда все её компоненты работают без превышения предельно допустимых технических ограничений. Если же некоторые элементы системы работают «не в режиме» или на грани выхода за предельные рамки, то вполне возможно различное поведение устройств (микроконтроллеров).*

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»*
(3432) 237-079, факс: (3432) 237-038
http://www.averon.ru
ic@averon.ru

Примечание 2: *Следует убедиться в правильности работы автогенератора. Может потребоваться подбор номиналов навесных конденсаторов и/или настройка режима тактирования.*

РЕЖИМЫ ТАКТИРОВАНИЯ.

В микроконтроллерах семейства PIC18xxx2 предусмотрено больше режимов тактирования, нежели в контроллерах семейства PIC16xxxx. Два режима **RC**-автогенератора влияют на использование одного вывода. **OSC1**, как и раньше, используется для подключения **RC**-автогенератора, **OSC2** может использоваться или как тактовый выход ($F_{OSC}/4$), или в качестве цифрового входа/выхода RA6. В режимах **EC** или **ECIO** частота внешнего тактирования может быть до 40МГц, использование вывода RA6 аналогично режимам с **RC**-генератором.

Теперь имеется два «высокоскоростных» режима **HS**. В режиме **HS/PL**L (с умножением частоты на основе ФАПЧ) можно использовать кварц на 4..10 МГц, в то время как в обычном режиме HS имеется возможность использовать кварц с частотой вплоть до 25 МГц.

При сбросе по включению питания задержки **OST** (1024 такта) и **PWRT** (например, 72 мс) остаются такими же, что и в режиме **HS**. В режиме умножения частоты с использованием ФАПЧ (PLL) добавляется дополнительная задержка на 2 мс с тем, чтобы система ФАПЧ микроконтроллера успела «захватить» и стабилизировать частоту кварца.

В режиме **HS/PLL** система ФАПЧ микроконтроллера используется для умножения частоты кварца на 4, тем самым обеспечивается возможность получения частот вплоть до 40 МГц. Эта частота делится с целью получения нужного количества тактов, в результате частота выполнения команд составит 10 МГц.

Чтобы перезапустить контроллер в режиме **HS/PLL** при выходе из режима **SLEEP**, для автогенератора и системы ФАПЧ требуются задержки: **OST** и **PLL** (2 мс) соответственно.

ПЕРЕКЛЮЧЕНИЕ ИСТОЧНИКОВ ТАКТИРОВАНИЯ.

В семействе PIC18xxx2 предусмотрена возможность использования автогенератора модуля **Timer1** в качестве автогенератора всей системы в целом. В этом случае при работе от автогенератора модуля **Timer1** автогенератор системы прекращает работу так же, как и в режиме **SLEEP**, поэтому для выхода из этого режима потребуются те же самые задержки. Такой режим позволяет длительное время работать на очень низкой скорости ($F_{ТАКТОВ}$ = 32 кГц / 4 = 8 кГц), при этом потребляемая мощность оказывается почти такой же низкой, как и в режиме **SLEEP**. Переключение источников тактовых импульсов возможно между любыми режимами тактирования и автогенератором модуля Timer1, кроме режима системного автогенератора.

Для организации переключения источников тактирования необходимо выполнить оба из следующих условий:

- автогенератор модуля **Timer1** должен быть включен посредством установки бита **T1OSCEN** (3-й бит в регистре **T1CON**);
- само переключение источников тактирования должно быть разрешено посредством сброса бита **OSCSEN** (5-й бит в регистре конфигурации **CONFIG1H**).

Непосредственно переключение источников тактирования осуществляется битом **SCS** (0-й бит в регистре **OSCCON**). Если этот бит сброшен, то в качестве источника тактирования используется автогенератор микроконтроллера, а если установлен – то автогенератор модуля **Timer1**.

Следует помнить, что изменение тактовой частоты повлияет на работу периферийных устройств, тактирование которых зависит от тактирования системы.

Слово конфигурации.

В семействе контроллеров PIC18xxx слово конфигурации разделено на несколько байт по адресам с 0x300000 по 0x30000D, за назначением конкретных регистров обращайтесь к документации на конкретный контроллер и к описаниям регистров в заголовочных файлах, например **«p18f252.inc»**.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

ИЗМЕНЕНИЯ В СИСТЕМЕ КОМАНД.

При переходе к новому семейству пользователь должен быть в курсе изменений системы команд и должен выполнить соответствующую коррекцию программного обеспечения. Следует обратить внимание на те участки программы, где присутствуют логические ветвления, зависящие от состояния какого-либо бита регистра **STATUS**. Возможны ситуации, когда новые команды упрощают код.

В ***Приложении 3*** приведены команды, выполнение которых по-новому влияет на состояние битов состояния (битов регистра **STATUS**). ***Таблица 3-1*** иллюстрирует команды, взятые из системы команд среднего семейства, но влияющие на новые биты состояния. В ***Таблицу 3-2*** сведены команды, заменяющие старые. В ***Таблице 3-***3 приводятся новые команды, биты состояния, на которые эти команды влияют и краткое описание команд.

ОПЕРАТИВНАЯ ПАМЯТЬ.

В микроконтроллерах семейства PIC16xxxx все переменные и регистры специального назначения (**SFR**) размещаются в двух банках. Каждый банк содержит до 128 адресов. Первые 32 адреса в каждом банке зарезервированы под регистры специального назначения, а остальные используются для размещения регистров общего назначения (**GPR**).

В микроконтроллерах семейства PIC18xxx2 память данных разбита на банки по 256 байт. Все регистры специального назначения размещаются в банке 15. Все переменные могут храниться в банке 0, причём по тем же адресам, что и в микроконтроллерах среднего семейства. Адрес должен состоять из 12 разрядов и записываться в виде «0x0nn» (для банка 0 первые 4 бита всегда нулевые, nn = адрес, используемый в PIC16xxxx).

При работе с микроконтроллерами семейства PIC18xxx2 директива **`BANKSEL`** и команды, связанные с переключением между банками, оказываются ненужными, поскольку все переменные могут быть размещены в банке 0. Участки кода, относящиеся к вышеупомянутой директиве можно выделить в комментарий, удалить, либо оставить в том случае, когда её присутствие не имеет значения. Поскольку требования к памяти данных постоянно растут, то вполне возможна ситуация, кода потребуется переключение между банками – в этом случае вместо директивы **`BANKSEL`** можно воспользоваться макросом или специально написанной функцией. Вашему вниманию предлагается рекомендуемая последовательность действий при коррекции программного обеспечения.

1. Следует удостовериться, что все переменные оперативной памяти расположены в банке 0 и имеют 12-разрядные адреса.
2. Участок кода, где впервые сбрасываются биты **RP1** и **RP0** регистра **STATUS**, следует заменить на строку **`clrf BSR`**.
3. Все строки программы, касающиеся битов **RP1** и **RP0**, следует поместить в комментарий, либо просто удалить.
4. Все строки программы, касающиеся директивы **`BANKSEL`**, следует поместить в комментарий, либо просто удалить (необязательно).

Доступ к банкам выполняется автоматически в случае, если операнд содержит 12-разрядный адрес, расположенный ниже 0x080 (регистры общего назначения в нижней половине банка 0), или в 0xF80 и выше (регистры специального назначения в верхней половине банка 15). При адресации операндов в диапазоне 0x000..0x07F или 0xF80..0xFFF **BSR** игнорируется. Адреса нижней части диапазона лежат в банке 0, в то время как адреса верхней части лежат в банке 15.

В контроллерах семейства PIC16xxxx регистры специального назначения располагаются по нижним 32 адресам каждого банка. А в контроллерах семейства PIC18xxx2 все регистры специального назначения расположены по 12-разрядным адресам в верхней части банка 15 (0xF80..0xFFF).

При обращении к регистрам специального назначения следует использовать их символьные имена. При этом **BSR** игнорируется, и для адресации регистров специального назначения автоматически выбирается банк 15. Большинство названий регистров специального назначения или те же, что и в PIC16xxxx, или очень похожи (см. ***Приложение 1***).

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

Регистр **FSR** заменён двумя регистрами **FSR0H**:**FSR0L** и должен содержать 12-разрядный адрес, требуемый для обращения к любому из регистров общего или специального назначения в любом банке. Поэтому в программе вместо **FSR** следует использовать **FSR0L**, а в **FSR0H** нужно прописать нули. Для обращения к регистрам или адресам оперативной памяти, указанным в **FSR0**, следует использовать **INDF0** в качестве операнда команд адресации. Однако практика показывает, что использование **FSR0** для косвенной адресации регистров специального назначения неэффективно. Оказывается, что прямая адресация делает программный код более экономичным, да и пользоваться ей гораздо удобнее. Поэтому при изменении программы рекомендуется следовать нижеприведённому алгоритму.

1. Если для адресации регистров специального назначения используется регистр **FSR**, то следует изменить соответствующий операнд **INDF** на имя **SFR**.
2. Все команды **`bsf STATUS,IRP`** и **`bcf STATUS,IRP`** нужно заменить командами **`clrf FSR0H`**.
3. Все строки типа **`BANKISEL`** `имя_переменной` следует поменять на **`clrf FSR0H`**.
4. Название **FSR** нужно изменить на **FSR0L**.
5. **INDF** переименовать в **INDF0**.

Регистр **FSR0L** содержит 8 разрядов, поэтому команды инкремента **INCF** или декремента **DECF** могут привести к его переполнению и повлиять на биты состояния (биты регистра **STATUS**). При этом переполнение регистра **FSR0L** никак не скажется на **FSR0H** (см. ***Пример 1***).

ПРИМЕР 1: Инкремент FSR0L и эффект переполнения.

```
clrf  FSR0H           ; прописываем нули в FSR0H.
movlw 0xFF
movwf FSR0L           ; записываем в FSR0L максимальное
                      ; 8-разрядное число (255).
incf  FSR0L,F         ; инкремент FSR0L --> FSR0L=0x00,
                      ; STATUS, C=1 (бит переноса),
                      ; STATUS, Z=1 (результат операции=0),
                      ; FSR0H=0x00 (никаких изменений).
```

ПАМЯТЬ ПРОГРАММ.

Для адресации слов памяти программ в микроконтроллерах семейства PIC16xxxx используется 13-разрядный программный счётчик. А в контроллерах семейства PIC18xxx2 программный счётчик содержит уже 21 разряд и адресует ***байты*** памяти программ, причём при выборе *команды значение счётчика увеличивается на два*. Любые попытки записать в **PCL<0>** 1 приведут к результату **PCL<0>** = 0. Поэтому при обращении к памяти программ самый младший разряд программного счётчика всегда установлен в нуль.

Запись в **PCL**, как и в контроллерах среднего семейства, приводит к перезаписи содержимого **PCLATH** в **PCH**, а содержимого **PCLATU** – в **PCU**. При чтении PCL всё происходит с точностью до наоборот: содержимое регистров **PCH** и **PCU** переписывается в **PCLATH** и в **PCLATU** соответственно.

Команды **`CALL`** и **`GOTO`** содержат всю необходимую информацию, касающуюся адресации. Поэтому для подготовки к переходам по программе не требуется каких бы то ни было манипуляций с регистром **PCLATH**, кроме того, они не будут иметь никакого практического эффекта. Фрагменты кода, относящиеся к упомянутым моментам, могут быть оставлены без изменения, помещены в комментарий, либо попросту удалены.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

КОМАНДЫ С ИСПОЛЬЗОВАНИЕМ СИМВОЛА $.

Иногда программисты пользуются символом «**$**» для обозначения адреса данной команды в памяти программ. Само по себе обозначение «**$**» означает то же, что и раньше. Однако, поскольку теперь программный счётчик адресует память ***побайтно***, а не отдельные слова, то любые адресные смещения (сдвиги) необходимо удвоить с тем, чтобы получить правильный адрес (см. ***Пример 2***).

ПРИМЕР 2: Удвоенные смещения адресов.

```
goto  $-6        ; вместо goto   $-3.
goto  $+0x2E     ; вместо goto   $+0x17.
```

ТАБЛИЦЫ RETLW.

При использовании семейства PIC18xxx2 требуется изменить подпрограммы вычисляемых переходов **GOTO** для таблиц **RETLW** в смысле вычисления необходимого сдвига по таблице, которое осуществляется до вызова таблицы (выше по тексту программы), либо нужно скорректировать сдвиг внутри самой подпрограммы.

Поскольку вычисляемый переход **GOTO** вызывает команду **RETLW**, то бит **PCL<0>** должен быть сброшен. Поэтому для правильного перехода по таблице необходимо, чтобы величина сдвига была удвоена.

Пример 3 иллюстрирует способ модификации подпрограммы с целью реализации поддержки 256-значной таблицы (сдвиг меняется от 0 до 255): при таком способе не имеет значения, ни как вызывалась таблица, ни возможные проблемы выхода кода за пределы страницы. Необходимы лишь временное расположение в оперативной памяти и метка в начале таблицы.

ПРИМЕР 3: Модифицированная программа для поддержки входной таблицы на 256 байт.

```
            movf  OFFSET,w          ; Сдвиг меняется от 0x00 до 0xFF.
            call  table
            . . .

            ORG   0x3C0
table       movwf taboff                ; Сохранить сдвиг по таблице.
            bcf         STATUS,C    ; Сброс бита переноса.
            rlcf        taboff,f    ; Умножение на 2,результат – в
                                    ; taboff.

            movlw HIGH(tab_st1)     ; Берём старший байт адреса
                                    ; таблицы.
            btfsc STATUS,C          ; Проверка бита переноса.
            incf        WREG,W
            movwf PCLATH            ; Если требуется, то изменяем
                                    ; содержимое регистра PCLATH.
            movlw LOW(tab_st1)      ; Берём младший байт адреса
                                    ; таблицы.
            addwf taboff,w          ; Добавляем сдвиг.
            btfsc STATUS,C          ; Переполнение было?
            incf        PCLATH,F    ; Инкремент, если не было.
            movwf PCL               ; Если было, то осуществляем
```


*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»*
(3432) 237-079, факс: (3432) 237-038
http://www.averon.ru
ic@averon.ru

```
                                ; переход, PCLATH и PCLATU
                                ; переписываются в PCH и PCU.
tab_st1     retlw 0x00          ; «Тело» таблицы, первое
                                ; значение, сдвиг=0.
            retlw 0x01          ; (256 значение, 256 слово/
                                ; /512 байт).
            . . .
            . . .
```

ТАБЛИЦЫ В ПАМЯТИ ПРОГРАММ.

Вычисляемые переходы **GOTO** с использованием таблиц **RETLW** можно организовать в рамках системы команд контроллеров семейства PIC18xxx2, но в этом случае в каждой команде памяти программ (одна команда – 16 разрядов) можно разместить только один байт данных, а размеры таблицы ограничены 256 значениями.

Вместе с тем, табличные операции позволяют размещать в каждом слове памяти программ два 8-битных байта данных и, таким образом, уплотнять табличные данные в два раза. При этом размер таблицы определяется только размером памяти программ. Кроме всего прочего, память программ может быть прочитана, что очень удобно при вычислении контрольной суммы программы с целью определения целостности кода.

ЧТЕНИЕ ТАБЛИЦ.

21-разрядный табличный указатель на память программ загружается вместе с адресом байта данных, подлежащих чтению. Этот указатель хранится в **TBLPTRU<4:0>**, **TBLPTRH<7:0>** и в **TBLPTRL<7:0>**. В результате выполнения команды **TBLRD*** данные, расположенные по текущему адресу, помещаются в регистр **TABLAT**, где они уже могут быть использованы программой именно как данные.

TBLPTRU<4:0>, **TBLPTRH<7:0>** и в **TBLPTRL<7:0>** в области памяти представляют собой регистры специального назначения. Не требуется, чтобы бит **TBLPTRL<0>** всегда был сброшен, как в случае с **PCL<0>**. Для регистров группы **TBLPTR** предусмотрен автоматический инкремент или декремент с использованием вариантов команды **TBLRD***. В ***Таблице 1*** приведены команды и то, как они влияют на регистры **TABLAT** и **TBLPTR**. Операции инкремента или декремента указателя влияют на каждый из 21 разрядов регистров группы **TBLPTR**.

ТАБЛИЦА 1: Команды чтения таблиц и их влияние на табличный указатель.

Команда	Действие команды.
TBLRD*	Помещает копию байта памяти программ в регистр TABLAT.
TBLRD*+	Помещает копию байта памяти программ в регистр TABLAT. Инкремент TBLPTR после чтения.
TBLRD*-	Помещает копию байта памяти программ в регистр TABLAT. Декремент TBLPTR после чтения.
TBLRD+*	Инкремент TBLPTR перед чтением. Помещает копию байта памяти программ в регистр TABLAT.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»*
(3432) 237-079, факс: (3432) 237-038
http://www.averon.ru
ic@averon.ru

АЛГОРИТМ ЧТЕНИЯ ТАБЛИЦ В ПАМЯТИ ПРОГРАММ.

1. Записать в табличный указатель требуемый адрес. (по желанию программиста **TBLPTR** может быть чётным, а может быть и нечётным).
2. Выполнить команду **TBLRD**.
3. Прочитать байт данных, скопированный в регистр **TABLAT** из памяти программ.

ПРИМЕР 4: Код, реализующий чтение таблицы.

```
RdStr       movlw HIGH(string)
            movwf TBLPTRH           ; Загружаем старший байт
                                    ; указателя (0x12).
            movlw LOW(string)
            movwf TBLPTRL           ; Загружаем младший байт
                                    ; указателя(0x34).
read        tblrd*+                 ; Считываем байт из памяти
                                    ; программ, сдвигаем указатель
                                    ; на один байт.
            movff TABLAT,PORTB      ; Перемещаем один байт из
                                    ; табличного регистра-защёлки
                                    ; в выходной порт B.
            tstfsz      TABLAT      ; Байт прочитан?
            goto        read        ; Если нет, то читаем снова.
            return                  ; Если да, то выходим.

            ORG         0x1234
String      DW          "Ntcn.",0x00    ; Текстовая строка.
```

ПРЕРЫВАНИЯ.

Сброс и система прерываний контроллеров семейства PIC18xxx2 совместимы с PIC16xxxx. Единственное различие состоит в том, что в контроллерах среднего семейства начало вектора прерываний располагалось по адресу 0x0004, а в микроконтроллерах новой серии оно переместилось в адрес 0x0008.

ВЫВОДЫ ВНЕШНИХ ПРЕРЫВАНИЙ.

Вывод **RB0/INT**, присутствующий у микроконтроллеров семейства PIC16xxxx, в PIC18xxx2 переименован в **RB0/INT0**. Кроме **RB0/INT0** в контроллерах нового семейства для поддержки дополнительных функций, касающихся прерываний, предусмотрены ещё два вывода: **RB1/INT1** и **RB2/INT2** (прежние **RB1** и **RB2**).

РЕГИСТРЫ, УПРАВЛЯЮЩИЕ ПРЕРЫВАНИЯМИ.

Регистр **INTCON** практически не изменился. Биты **INTE** и **INTF** переименованы соответственно в **INT0IE** и **INT0IF**.

Регистр **INTCON2** содержит биты, которые определяют уровни срабатывания прерываний **INT0**, **INT1**, **INT2**. Приоритет прерываний от модуля **Timer0** и от линий **PORTB** выставляется здесь же. Кроме того, биты регистра **INTCON2** управляют «подтяжкой» резисторами **PORTB**.

В регистре **INTCON3** находятся биты разрешения прерываний **INT1** и **INT2**, флаги прерываний, а также биты, определяющие приоритет прерываний.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»*
(3432) 237-079, факс: (3432) 237-038
http://www.averon.ru
ic@averon.ru

Для флагов прерываний, расположенных в регистре **PIR1**, имеются биты разрешения соответствующих прерываний в регистре **PIE2** и биты приоритета в регистре **IPR1**. Аналогичная ситуация наблюдается для регистров **PIR2**, **PIE2** и **IPR2**.

Чтобы адаптировать старую программу под микроконтроллер нового семейства, необходимо выполнить следующие несложные шаги.

1. Стартовый адрес вектора прерываний нужно изменить с 0x0004 на 0x0008.
2. Биты **INT**, **INTF** и **INTE** следует переименовать соответственно в **INT0**, **INT0IF** и **INT0IE**.
3. Бит **OPTION<NOT_RBPU>** нужно заменить битом **INTCON2<NOT_RBPU>**.
4. Бит **OPTION<INTEDG>** следует поменять на **INTCON2<INTEDG0>**.

Прерывание **INT0** всегда обладает наивысшим приоритетом. Соответствующий этому прерыванию бит разрешения **INT0IE** и флаг **INT0IF** находятся в регистре **INTCON**.

Для унификации работы с прерываниями в семействе PIC18xxx2 все остальные регистры, имена битов и их расположение остались теми же, что и в среднем семействе PIC16xxxx.

ПРИОРИТЕТ ПРЕРЫВАНИЙ.

В микроконтроллерах семейства PIC18xxx2 организована двухуровневая система приоритетов: каждому уровню соответствует свой вектор прерываний. Адреса прерываний, которым назначен более высокий приоритет, расположены в «векторе высшего приоритета», начиная с 0x0008, а прерывания с более низким приоритетом располагаются в «векторе низшего приоритета», начиная с 0x0018. Приоритет прерываний включается установкой бита **IPEN (RCON<7>)**.

Когда бит **IPEN** сброшен, всем прерываниям назначен высокий приоритет, поэтому все прерывания располагаются в «векторе высшего приоритета». Этот режим одинаковых приоритетов совместим с режимом организации приоритетов контроллеров семейства PIC16xxxx.

Когда бит **IPEN** установлен, активным является режим разных приоритетов. В этом случае меняется назначение битов **GIE (INTCON<7>)** и **PEIE (INTCON<6>)**. Бит общего разрешения прерываний **GIE** превращается в бит общего разрешения прерываний высшего приоритета **GIEH**, а бит разрешения периферийных прерываний **PEIE** трансформируется в бит общего разрешения прерываний низшего приоритета **GIEL**. При сброшенном бите **GIEL** все прерывания низшего приоритета запрещены. Если же сброшен бит **GIEH**, то запрещёнными оказываются не только прерывания высшего, но и прерывания низшего приоритета.

Не лишним будет подчеркнуть, что и при сброшенном бите **IPEN** (то есть в режиме одинаковых приоритетов) сброс бита **GIEH** также **запрещает ВСЕ прерывания**.

Каждому источнику прерывания назначен свой бит установки приоритета. Установленный бит означает высокий приоритет, сброшенный – низкий. Если флаг прерывания располагается в регистре **PIR1**, то соответствующий бит установки приоритета расположен в **IPR1**, а если флаг находится в регистре **PIR2**, то бит установки приоритета – в **IPR2**. Остальные биты установки приоритета размещены в регистрах **INTCON2** и **INTCON3**.

Приоритеты прерываний для **Timer0** и **RBIF** устанавливаются битами **TMR0IP (INTCON2<2>)** и **RBIP (INTCON<2>)** соответственно. Приоритеты для **INT1** и **INT2** назначаются битами **INT1IP (INTCON3<6>)** и **INT2IP (INTCON3<7>**) соответственно. Сброс этих битов означает низкий приоритет.

Биты **INT1IE (INTCON3<3>)** и **INT2IE (INTCON3<4>**) разрешают соответственно прерывания **INT1** и **INT2**. Этим прерываниям назначены флаги **INT1IF (INTCON3<3>)** и **INT2IF (INTCON3<4>)**.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

СТЕК ВОЗВРАЩАЕМОГО АДРЕСА.

Контроллеры семейства PIC18xxx2 обладают стеком, в котором организован 31 уровень, в то время как контроллеры среднего семейства имеют лишь 8-уровневый стек. Кроме того, в новом семействе предусмотрен доступ к указателю стека, битам ошибок, а также к содержанию верхней части стека. Поэтому для работы с содержимым стека и с указателем в систему команд были добавлены инструкции **PUSH** и **POP**.

У PIC16xxxx стек состоит из 8 уровней и функционирует как кольцевой буфер: запись в стек, следующая после 8-й записи, записывается на первое место, следующая запись – на второе, и так далее; извлечение из стека происходит в обратном порядке, то есть 8-я, запись, 7-я, и так по первую запись включительно, после 1-й вновь извлекается 8-я запись, 7-я, и так далее. Индикация состояния стека отсутствует. Содержимое стека программе недоступно.

Стек PIC18xxx2 отличается не только тем, что имеет 31 уровень, но и принципом работы: он функционирует как линейный буфер. 31-я запись в стек вызывает установку бита переполнения стека **STKFUL (STKPTR<7>)**. 32-я запись производится на место 31-й. Записи извлекаются, начиная с 31-й, и заканчивая 1-й, затем каждая следующая команда извлечения возвращает 0x0000, инициирует установку бита опустошения стека **STKUNF (STKPTR<6>)** и запускает программу с начала, при этом сброса микроконтроллера не происходит. По желанию пользователя можно организовать сброс микроконтроллера при переполнении или опустошении стека (см. раздел **СИСТЕМА СБРОСА МИКРОКОНТРОЛЛЕРА**.). Биты переполнения и опустошения стека сбрасываются либо при включении питания, либо программно. Такая возможность позволяет программно организовать адекватную реакцию системы на ошибки, возникающие при работе со стеком.

По завершении инициализации контроллера в указателе стека **STKPTR** содержатся нули, поэтому **STKPTR** указывает на адрес стека, по которому прописано значение 0x00000, соответствующее вектору сброса. Любая из команд **PUSH** и **CALL**, или любое прерывание вызовет инкремент указателя стека, то есть указатель сместится на более высокое положение, которое будет символизировать новую «вершину» стека и куда затем запишется содержимое программного счётчика. Команда возврата (**RETURN** или **RETFIE**) приведёт к перезаписи содержимого верхней части стека в программный счётчик и к декременту указателя стека. Команда **POP** просто-напросто декрементирует указатель, сбрасывая содержимое верхней части стека.

«Вершина» стека, содержащая 21 разряд, доступна через регистры верхней части стека **TOSU<4:0>**, **TOSH<7:0>** и **TOSL<7:0**>. Она может быть подвержена процедурам чтения, записи, сохранения данных и восстановления данных.

СТЕК БЫСТРЫХ РЕГИСТРОВ (ТЕНЕВОЙ СТЕК).

Стек быстрых регистров представляет собой группу регистров, в которых каждый раз в случае прерывания или вызова подпрограммы сохраняется содержимое регистров **WREG**, **STATUS** и **BSR**. Это одноуровневый стек, и пользователю он недоступен. При выполнении команды **retfie FAST** содержимое стека быстрых регистров переписывается обратно в регистры **WREG**, **STATUS** и **BSR**.

Вызов подпрограммы также может выполняться с параметром **FAST**. Команда типа **call label, FAST** тоже сохраняет содержимое регистров **WREG**, **STATUS** и **BSR** в стеке быстрых регистров. Для восстановления контекста требуется соответствующая команда возврата из подпрограммы: **return FAST**.

Если разрешены какие-либо прерывания, то для возврата из подпрограммы стек быстрых регистров использовать уже нельзя. Если во время выполнения подпрограммы происходит прерывание, то в быстрые регистры автоматически записывается текущее содержимое регистров **WREG**, **STATUS** и **BSR**. После возврата из прерывания стек быстрых регистров будет по-прежнему хранить содержимое регистров **WREG**, **STATUS** и **BSR**, которое имело место при возникновении прерывания. При попытке

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

возврата из подпрограммы с использованием стека быстрых регистров в регистрах **WREG**, **STATUS** и **BSR** попросту будет восстановлен неверный контекст.

В случае режима разных приоритетов стек быстрых регистров могут использовать только прерывания высшего приоритета. Следовательно, при обработке прерываний более низкого приоритета необходимо самостоятельно программно осуществлять сохранение контекста. Прерывание более высокого приоритета в любой момент может прервать обработку прерывания более низкого приоритета.

СИСТЕМА СБРОСА МИКРОКОНТРОЛЛЕРА.

В PIC18xxx2 организованы те же самые источники сброса, что и в PIC16xxxx.

Следует отметить, что регистр **PCON** получил новое имя **RCON**. Биты **TO** и **PD** перенесены из регистра **STATUS** в регистр **RCON**. В остальном все биты выполняют те же функции, что и в контроллерах среднего семейства.

СБРОС ПРИ ВКЛЮЧЕНИИ ПИТАНИЯ (POR).

Как уже отмечалось, регистр **PCON** переименован в **RCON** и содержит бит **POR**. Действие этого бита не изменилось.

СБРОС ПРИ СНИЖЕНИИ НАПРЯЖЕНИЯ ПИТАНИЯ (BOR).

Бит **BOR** содержится в регистре **RCON** и функционирует точно так же, как и в контроллерах семейства PIC16xxxx.

В PIC18xxx2, как и в PIC16xxxx, модуль **BOR**, ответственный за сброс при снижении питающего напряжения может быть как включен, так и выключен. Когда модуль **BOR** включен, автоматически оказывается включенным и таймер запуска **PWRT**, при этом состояние бита включения таймера запуска **PWRTEN (CONFIG2L<0>)** игнорируется. В отличие от среднего семейства в новых микроконтроллерах вместо одного порога срабатывания детектора снижения питания организовано целых четыре, которые задаются комбинацией битов **BORV1:BORV0 (CONFIG2L<3:2>)**. Время (параметр 35, $\mathbf{T_{BOR}}$), в течение которого должно поддерживаться пониженное напряжение питания (параметр D005, $\mathbf{V_{BOR}}$) для гарантированного наступления сброса, увеличено по сравнению с PIC16xxxx, где оно составляло 100 микросекунд.

Ниже следуют требуемые шаги по коррекции текста программы.

1. Название регистра **PCON** следует поменять на **RCON**.
2. Бит **STATUS<NOT_TO>** нужно заменить битом **RCON<NOT_TO**>.
3. Бит **STATUS<NOT_PD>** подлежит замене на **RCON<NOT_PD>.**
4. С помощью битов **BORV1:BORV0 (CONFIG2L<3:2>)** нужно сконфигурировать требуемый порог снижения напряжения питания.

СБРОС ПО СИГНАЛУ СБРОСА (MCLR).

Сигнал сброса, реализованный в микроконтроллерах семейства PIC18xxx2, работает так же, как и в контроллерах среднего семейства. Следовательно, никаких программных или аппаратных изменений не требуется.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

СБРОС ПО СИГНАЛУ ОТ СТОРОЖЕВОГО ТАЙМЕРА (WDT).

В контроллерах PIC18xxx2, в отличие от PIC16xxxx, у сторожевого таймера имеется собственный делитель частоты, независимый от предварительного делителя модуля **Timer0**. Если бит **WDTEN (CONFIG2H<0>)** установлен, то сторожевой таймер включен, если бит сброшен, то таймер выключен. Коэффициент деления частоты (1:1…1:128) определяется комбинацией битов **WDTPS2:WDTPS0 (CONFIG2H<3:1>**). Вместо битов **TO (STATUS<4>)** и **PD (STATUS<3>)** теперь используются биты **TO (RCON<3>) PD (RCON<2>)**. Действие этих битов не изменилось.

Если сторожевой таймер был *отключен* посредством сброса бита **WDTEN**, то его можно вновь включать и впоследствии выключать, программно контролируя соответственно установку и сброс бита **SWDTE (WDTCON<0>)**. Что касается коэффициента деления частоты, то в ходе выполнения программы его изменить не удастся.

Если же сторожевой таймер был *включен* при помощи бита **WDTEN**, то манипуляции битом **SWDTE** не приведут к желаемому результату.

ЭФФЕКТЫ ПЕРЕПОЛНЕНИЯ И ОПУСТОШЕНИЯ СТЕКА.

Как известно, в PIC16xxxx реализован 8-уровневый стек, и если программный счётчик выполнит 8 записей в стек, то 9-я запись будет произведена поверх 1-й, при этом никакая генерация ошибки не предусмотрена. В стеке PIC18xxx2 – 31 уровень. Когда стек почти заполнен (содержит 30 записей), и производится 31-я запись, то в этой ситуации выставляется флаг переполнения стека **STKFUL**, и каждая последующая запись будет осуществляться поверх 31-й.

В противоположной ситуации, когда стек пуст (все записи извлечены), и имеет место попытка извлечения, выставляется флаг опустошения стека **STKUNF**, а в программный счётчик записывается адрес вектора **RESET**. При этом происходит перезапуск программы, но сам контроллер не сбрасывается.

Система может быть сконфигурирована таким образом, что при выставлении флагов переполнения или опустошения стека осуществляется сброс микроконтроллера. Для этого необходимо установить бит **SVTREN (CONFIG4L<0>)**. Сами биты **STKFUL** и **STKUNF** сбрасываются только программно, либо при включении питания***. Сброс, вызванный ошибкой стека (переполнение или опустошение), эти биты не сбросит!*** Чтобы определить причину сброса, пользователь должен опросить биты **STKFUL** и **STKUNF**, а при проведении корректирующих действий – сбросить их.

КОМАНДА СБРОСА `RESET`.

Для PIC18xxx2 введена команда **`RESET`**, которая осуществляет сброс контроллера подобно сигналу **MCLR**. При этом сбрасывается вся периферия, а выполнение кода продолжается, начиная с вектора сброса.

В регистре **RCON** содержится бит **RI**, который устанавливается при сбросе по включению питания, при сбросе по снижению питающего напряжения и при сбросе по сигналу от сторожевого таймера, а сбрасывается этот бит при выполнении команды **`RESET`**. На биты **TO**, **PD**, **BOR** и **POR** команда **`RESET`** влияния не оказывает. Опрос бита **RI (RCON<4>)** позволяет определить причину сброса.

МОДУЛЬ TIMER0.

Режим работы модуля **Timer0** PIC18xxx2 аналогичен PIC16xxxx. В PIC18xxx2 счётчик **Timer0** доступен для чтения и записи через регистр **TMR0L**, в отличие от **TMR0** в PIC16xxxx. Регистр **OPTION_REG** переименован в **T0CON**.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

Бит **PSA (T0CON<3>)** функционирует так же, как и в контроллерах среднего семейства: бит сброшен – предварительный делитель частоты модуля **Timer0** включен, бит установлен – предварительный делитель частоты модуля **Timer0** отключен.

Если понадобится иной коэффициент деления частоты, чем определяемый битами **PS2:PS0 (OPTION<2:0>)**, то на этот случай предусмотрены биты **T0PS2:T0PS0 (T0CON<2:0>)**, которые работают точно также.

Модуль **Timer0** выставляет свой флаг прерывания (**T0IF**) при переполнении счётного регистра **TMR0L** (в PIC16xxxx флаг выставлялся при переполнении **TMR0**).

Содержимое регистра **T0CON** не влияет на делитель частоты сторожевого таймера (см. раздел **СТОРОЖЕВОЙ ТАЙМЕР**).

Бит **RBPU (OPTION<7>)** переименован и перенесён в **RBPU (INTCON2<7>)**, а **бит INTEDG (OPTION<6>)** – в **INT0EDG (INTCON2<6>)**.

Таким образом, для перехода от PIC16xxxx к PIC18xxx2 потребуются следующие изменения в программе.

1. Нужно переместить бит **OPTION<NOT_RBPU**> в **INTCON2<NOT_RBPU>**.
2. Бит **OPTION_REG<INTEDG>** следует переместить с переименованием в **INTCON2<INTEDG0>.**
3. Имя регистра **OPTION_REG** необходимо сменить на **T0CON**.
4. Биты **PS2:PS0** подлежат переименованию в **T0PS2:T0PS0**.
5. Фрагменты программы, относящиеся к модификации битов **PS2:PS0** и **PSA** применительно к **WDT** следует поместить в комментарий либо удалить. Вместо этих фрагментов нужно дописать строки, должным образом инициализирующие регистр **CONFIG2H**.
6. В операциях чтения и записи модуля **Timer0** имя **TMR0** необходимо изменить на **TMR0L**.

Сбросив бит **T08BIT (T0CON<6>)**, можно заставить модуль **Timer0** работать в качестве 16-разрядного таймера. В этом режиме, если в **TMR0L** уже что-то записано, **TMR0H** записывается в старший байт модуля **Timer0**. Когда регистр **TMR0L** прочитан, содержимое **TMR0H** обновляется из старшего байта **Timer0**. В режиме 16-разрядного счётчика прерывание от модуля **Timer0** происходит в момент переполнения **Timer0** из 0xFFFF в 0x0000.

МОДУЛЬ TIMER1.

Модуль PIC16xxxx **Timer1** совместим «вверх» с аналогичным модулем PIC18xxx2, то есть в контроллерах нового семейства поддерживаются все прежние особенности модуля.

При установленном бите **RD16 (T1CON<7>)** чтение **TMR1L** приводит к записи в **TMR1H** старшего байта **Timer1**. Запись в **TMR1L** приводит к перезаписи содержимого регистра **TMR1H** в старший байт **Timer1**. В этом режиме пользователю нет нужды проверять младший байт на предмет переполнения в процессе чтения старшего байта, или останавливать работу таймера при записи в него константы.

Автогенератор модуля **Timer1**, реализованный в PIC18xxx2, в принципе не отличается от варианта, представленного в PIC16xxxx. Тем не менее, вследствие некоторых изменений настройки автогенератора придётся скорректировать с тем, чтобы он заработал должным образом.

Примечание: *коррекция настройки автогенератора может потребоваться даже в том случае, если пользователь не вносил никаких изменений в колебательные цепи. В частности, может потребоваться подбор номиналов навесных конденсаторов.*

МОДУЛЬ TIMER2.

Никаких изменений не требуется, поскольку эти модули в среднем и новом семействе абсолютно одинаковы.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

МОДУЛЬ TIMER3.

В PIC18xxx2 реализован четвёртый по счёту таймер, которого ещё не было в PIC16xxxx. Этот модуль идентичен модулю **Timer1** PIC18xxx2. Оба этих модуля могут служить времязадающей основой для модуля захвата и сравнения **CCP**, а также могут для своих нужд использовать автогенератор модуля **Timer1**. Оба таймера могут быть сброшены по сигналу от специального событийного триггера модуля захвата и сравнения.

МОДУЛЬ ЗАХВАТА, СРАВНЕНИЯ И ШИРОТНО-ИМПУЛЬСНОЙ МОДУЛЯЦИИ (CCP).

Функции захвата, сравнения и ШИМ PIC16xxxx полностью совместимы с аналогичными функциями PIC18xxx2. Поэтому никаких программных или аппаратных изменений не требуется.

В модуле **CCP** PIC18xxx2 реализован дополнительный режим, отсутствовавший у представителей среднего семейства: в случае совпадения в режиме сравнения состояние выхода модуля **CCP** инвертируется.

МОДУЛЬ АНАЛОГО-ЦИФРОВОГО ПРЕОБРАЗОВАНИЯ (A/D).

В случае сброса модуль АЦП контроллеров нового семейства принимает то же состояние, что и аналогичный модуль контроллеров семейства PIC16xxxx. В программе требуется лишь одно изменение. Поскольку АЦП PIC18xxx2 имеет 10 разрядов, то для представления 10-разрядного результата преобразования требуется два 8-разрядных регистра. Поэтому регистр результата **ADRES** превратился в регистры **ADRESH** и **ADRESL**, которые содержат старшие и младшие разряды соответственно. Принцип действия и устройство регистров **ADCON0** и **ADCON1** остались прежними, но в **ADCON1** добавлено три новых бита. Новые биты и новый регистр обеспечивают дополнительные функции, которые не были доступны в PIC16xxxx. Бит **ADFM** (ADCON1<7>) управляет выравниванием 10-разрядного результата в **ADRESH:ADRESL**. По желанию пользователя, можно организовать представление результата преобразования в виде 8 разрядов. Для этого нужно сбросить бит **ADFM** (бит перейдёт в состояние сброса **RESET**, совместимое с PIC16xxxx), при этом старшие 8 разрядов результата помещаются в регистр **ADRESH**, а оставшиеся 2 разряда – в **ADRESL**<7:6>. Биты **ADRESL**<6:0> будут в сброшенном состоянии. Такой формат представления позволяет пользователю применять по отношению к результату восьмиразрядные математические операции.

При установленном бите **ADFM** в силу вступает 10-разрядный формат представления результата преобразования. В этом режиме 8 младших разрядов сохраняются в **ADRESL0**<7:0>, а 2 старших разряда – в **ADRESH**<1:0>. При этом биты **ADRESH**<7:2> сброшены. Такой формат удобен при использовании 10-разрядного результата в квазишестнадцатиразрядных математических операциях.

В контроллерах нового семейства возможен машинный цикл длиной 100 нс (когда автогенератор на 10 МГц управляет модулем ФАПЧ с умножением частоты). Ресурсы автогенератора, доступные через биты **ADCS1**:**ADCS0** (ADCON0<7:6>), не позволяют на столь высокой частоте установить минимальное время преобразования $\mathbf{T_{AD}}$, равное 1,6 мкс. Третий бит, который отвечает за выбор тактовой частоты АЦП, называется **ADCS2** (ADCON1<6>). Когда он установлен, частота выполнения команд делится пополам, при этом уже можно вполне корректно настроить время преобразования T_{AD}. Все эти манипуляции никак не влияют на встроенный в АЦП RC-автогенератор.

Пользоваться внешней опорой для аналого-цифрового преобразования можно как в PIC16xxxx, так и в PIC18xxx2. Контроллеры нового семейства, кроме того, позволяют использовать низкие напряжения. Комбинация битов **PCFG3:PCFG0** (ADCON1<3:0>) определяет три возможных варианта: внутренняя опора и внутренняя «земля», внешнее высокое опорное напряжение и внутренняя «земля» и, наконец, внешние высокое и низкое опорные напряжения.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

Опорные напряжения не могут превышать напряжение питания, зато могут изменять диапазон преобразования, внося положительный сдвиг и уменьшая полное входное напряжение. Следует обратить внимание на ограничения, которые накладываются на величину опорного напряжения (параметры A20, A20A, A21, A22 и A25).

Пример 5 иллюстрирует последовательность действий при инициализации модуля АЦП.

ПРИМЕР 5: Настройка АЦП.

АЦП настраивается при следующих условиях.

На выводе **AN0** – любое напряжение из диапазона 0,5..3,5 В. Текущее напряжение на этом входе – 2,296 В. Микроконтроллер использует автогенератор на 8 МГц и ФАПЧ с умножением частоты (PLL). Предполагается использовать 10-разрядный результат преобразования.

Источник тактирования определяется битами **ADCS2:ADCS0** = b“110”. Частота 32 МГц на выходе PLL (после умножения) делится на 64 для получения времени преобразования $\mathbf{T_{AD}}$ = 2 мкс.

С помощью битов **PCFG3:PCFG0** = b“1101” выводы **AN0** и **AN1** сконфигурированы как аналоговые входы, **AN2** – как низкий опорный вход ($\mathbf{V_{REF-}}$), AN3 – как высокий опорный вход ($\mathbf{V_{REF+}}$). На **AN3** подаётся 3,5 В, на **AN2** – 0,5 В. Эти опорные напряжения используются для всех преобразований по всем разрешённым каналам.

Установленный бит **ADFM** определяет выравнивание результата по правому краю (8 бит – в **ADRESL**, и 2 бита – в **ADRESH**).

Канал **AN0** задаётся сброшенными битами **CHS2:CHS0**. Преобразование начинается при установке бита **GO/DONE** (ADCON0<2>). По окончании преобразования, когда бит **GO/DONE** сбросится, регистр **ADRESH:ADRESL** будет содержать число 0x0265.

На этом пример закончен.

УНИВЕРСАЛЬНЫЙ СИНХРОННО-АСИНХРОННЫЙ ПРИЁМОПЕРЕДАТЧИК (USART).

USART PIC16xxxx совместим «вверх» с аналогичным модулем PIC18xxx2. Никаких изменений кода не требуется.

Приёмопередатчик может контролировать последовательно передаваемые данные в 9-битном режиме. Установка бита **ADDEN** (RCSTA<3>) приводит к тому, что приёмопередатчик генерирует прерывание только в том случае, если установлен 9-й бит принятых данных, вместо того, чтобы прерывать выполнение основной программы каждый раз при получении нового байта данных. Пока 9-й бит посылки установлен, содержимое регистра **RCREG** не меняется: в нём хранится принятый адрес.

Такая организация чрезвычайно удобна для сетевых протоколов: установленный 9-й бит сигнализирует о том, что остальные 8 разрядов посылки содержат адрес абонента, а сброшенный 9-й бит говорит о том, что посылка содержит 8 бит данных.

Чтобы сконфигурировать выводы **RC6/TX/CK** и **RC7/RX/DT** соответственно как линии передачи и приёма, необходимо выполнить следующее:

- установить бит **SPEN** (RCSTA<7>);
- установить бит **TRISC**<7>;
- и сбросить бит **TRISC**<6>.

Следует обратить внимание на тот факт, что флаг TXIF сбрасывается отнюдь не сразу после помещения данных в передающий буфер TXREG. Этот бит становится доступен только во втором по счёту после команды записи в TXREG командном цикле (см. Пример 6). Таким образом, опрос флага TXIF, произведённый сразу после записи в TXREG, смысла не имеет.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

ПРИМЕР 6: Вариант корректного опроса флага TXIF.

```
movwf TXREG      ; Запись данных в передающий буфер.
nop              ; Выжидаем один командный цикл (на этом
                 ; месте может стоять любая команда).
btfss PIR1,TXIF  ; Опрашиваем флаг во втором командном
                 ; цикле, поскольку теперь он доступен.
```

ПОСЛЕДОВАТЕЛЬНЫЙ СИНХРОННЫЙ ПОРТ (SSP).

Регистр **SSPCON** переименован в **SSPCON1**. В остальном модуль **SSP** PIC16xxxx совместим «вверх» с модулем **MSSP** PIC18xxx2.

РЕЖИМ ПОСЛЕДОВАТЕЛЬНОГО ИНТЕРФЕЙСА (SPI).

В отношении **SPI** наблюдается полная совместимость среднего и нового семейств, поэтому никаких изменений кода не требуется.

РЕЖИМ I^2C.

Изменений кода не требуется, так как модуль SSP PIC16xxxx совместим «вверх» с модулем **MSSP** PIC18xxx2.

Модуль **SSP**, используемый в PIC16xxxx, обеспечивает ограниченную поддержку «ведущего» режима I^2C. В PIC18xxx2 используется модуль **MSSP**, который на аппаратном уровне поддерживает «ведущий» режим I^2C и режим нескольких «ведущих».

Был добавлен «ведущий» режим. Для аппаратной поддержки «ведущих» режимов был введён регистр **SSPCON2**. В контроллерах нового семейства ведомые устройства поддерживает общий вызов.

РЕЖИМ «ВЕДУЩИЙ» (MASTER).

Ведущий модуль **SSP** (MSSP), используя регистр **SSPCON2**, аппаратно поддерживает «ведущий» режим I^2C. Предусмотрено администрирование в режиме нескольких «ведущих». Для получения сигнала **SCL** используется специальный генератор.

В контроллерах семейства PIC16xxxx «ведущий» режим был реализован таким образом, что программно контролировал выводы **SCL** и **SDA**. Прерывания по стартовому (S) и стоповому (P) битам были организованы аппаратно.

Контроллеры семейства PIC18xxx2 по-прежнему поддерживают такой режим работы и не требуют изменения кода, но «ведущий» режим и режим нескольких «ведущих» уже организуются на аппаратном уровне.

В «ведущем» режиме, когда **SSPM3:SSPM0** (SSPCON1<3:0>) = b“1011” и когда **SSPEN** (SSPCON1<5>) = 1, выводы **SCL** и **SDA** управляются модулем **SSP**. Данные и ведомые адреса (7 или 10 разрядов) вместе с битом **R/W** пересылаются в буферный регистр **SSPBUF**. Скорость передачи определяется комбинацией битов **SSPADD**<6:0>.

Две операции в одно и то же время выполняться не могут: сначала завершается одна операция, и только после этого начинается следующая. В случае успешного завершения выставляется флаг **SSPIF**, при этом модуль **SSP** работает вхолостую. Возможные операции: старт, повторный старт, остановка, посылка одного байта данных и посылка сигнала подтверждения (**ACK/NACK**). Когда модуль **SSP** занят, любая попытка выполнения следующей операции игнорируется, при этом выставляется бит состояния **WCOL**

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

(SSPCON1<7>), а бит **SSPIF** остаётся сброшенным. Бит **WCOL** сбрасывается программно. Если шина была занята, то выставляется флаг **BCLIF** (PIR2<3>), и происходит прерывание.

ПОДДЕРЖКА ОБЩЕГО ВЫЗОВА В «ВЕДОМОМ» РЕЖИМЕ.

Поддержка общего вызова активизируется установкой бита **GCEN** (SSPCON2<7>). В случае совпадения адреса (либо адреса ведомого устройства, либо общего адреса 0x00) происходят следующие события.

- Принятый адрес помещается в регистр **SSPBUF**.
- Выставляется флаг **BF** (SSPSTAT<0>), сигнализирующий о том, что буфер полон.
- Генерируется импульс подтверждения (ACK).
- С установкой флага **SSPIF** (PIR1<3>) происходит прерывание.

При обслуживании прерывания необходимо прочитать буфер **SSPBUF** с тем, чтобы определить причину прерывания, поскольку его могло вызвать либо ведомое устройство, либо оно произошло вследствие совпадения с адресом общего вызова.

В ситуации, когда ведомое устройство настроено на работу с 10-разрядным адресом, бит **GCEN** установлен, а система идентифицировала принятый адрес как адрес общего вызова, бит **UA** (SSPSTAT<1>) не выставляется. Вместо этого ведомое устройство начинает приём данных после отправления сигнала подтверждения (ACK).

ЗАКЛЮЧЕНИЕ.

Итак, перевод инженерного проекта с контроллера среднего семейства PIC16xxxx на микроконтроллер нового семейства PIC18xxx2 заключается в следующем.

- Следует удостовериться в правильности работы основного автогенератора и автогенератора модуля **Timer1** (если он используется).
- Все переменные необходимо поместить в банке 0 и закрепить за ними 12-разрядные адреса.
- Нужно скорректировать подпрограммы чтения таблиц, использующие вычисляемые переходы `goto`.
- Следует должным образом изменить названия некоторых битов и регистров и скорректировать их расположение.
- В случае разрешённого сброса по снижению напряжения питания необходимо задать порог срабатывания детектора.

По желанию можно усовершенствовать проект, используя дополнительные функции PIC18xxx2, среди которых можно выделить:

- дополнительные режимы тактирования системы;
- переключение источников тактирования для снижения потребляемой энергии;
- дополнительную память программ и данных;
- восстановление данных с использованием памяти программ;
- дополнительные выводы для внешних прерываний;
- наличие приоритетов прерываний;
- возможность доступа к стеку и регистру состояния системы;
- возможность сброса при возникновении ошибки в работе стека;
- наличие теневого стека (стека быстрых регистров) для сохранения контекста;
- наличие команды сброса `RESET`;
- возможность работы модуля **Timer0** в режиме 8- или 16-разрядного таймера-счётчика;
- наличие четвёртого по счёту таймера – **Timer3**, повторяющего модуль **Timer1**;
- 10-разрядный модуль АЦП;
- полная поддержка «ведущего» режима I^2C.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

ПРИЛОЖЕНИЕ 1.

ИЗМЕНЕНИЯ В ИМЕНАХ И РАСПОЛОЖЕНИИ БИТОВ И РЕГИСТРОВ.

ТАБЛИЦА 1-1: Изменения в именах и расположении битов и регистров.

PIC16xxxx		PIC18xxx2		Примечания
Регистр	**Бит**	**Регистр**	**Бит**	
OPTION_REG	NOT_RBPU	INTCON2	NOT_RBPU	
OPTION_REG	INTEDG	INTCON2	INTEDG0	
OPTION_REG	T0CS	T0CON	T0CS	
OPTION_REG	TOSE	T0CON	T0SE	
OPTION_REG	PSA	T0CON	PSA	Сторожевой таймер WDT имеет собственный делитель частоты. Включается битом WDTEN (CONFIG2H<0>)
OPTION_REG	PS2	T0CON	T0PS2	Коэффициент деления частоты WDT устанавливается битами WDTPS2:WDTPS0 (CONFIG2H<3:1>)
OPTION_REG	PS1	T0CON	T0PS1	
OPTION_REG	PS0	T0CON	T0PS0	
PCON	NOT_POR	RCON	NOT_POR	
PCON	NOT_BOR	RCON	NOT_BOR	
STATUS	NOT_TO	RCON	NOT_TO	
STATUS	NOT_PD	RCON	NOT_PD	
INDF	—	INDF0	—	
FSR	—	FSR0L	—	
TMR0	—	TMR0L	—	
SSPCON	—	SSPCON1	—	
ADRES	—	ADRESH	—	

ПРИЛОЖЕНИЕ 2.

ИЗМЕНЕНИЯ В ПРОГРАММНОМ КОДЕ.

```
;     Код для PIC18xxx2:          ;     Код для PIC16xxxx:

      clrf  BSR                   ;     bcf   STATUS,RP0
                                  ;     bcf   STATUS,RP1
      movf  INDf0,w               ;     movf  INDF,w
      movf  TMR0L,w               ;     movf  TMR0,w
      movf  FSR0L,w               ;     movf  FSR,w
      movf  SSPCON1,w             ;     movf  SSPCON,w
      movf  ADRESH,w              ;     movf  ADRES,w
      movf  T0CON,w               ;     movf  OPTION_REG,w
      movf  RCON,w                ;     movf  PCON,w
```


*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»*
(3432) 237-079, факс: (3432) 237-038
http://www.averon.ru
ic@averon.ru

ПРИЛОЖЕНИЕ 3.

ИЗМЕНЕНИЯ В СИСТЕМЕ КОМАНД.

ТАБЛИЦА 3-1: Различия в функционировании битов состояния.

Команды	Биты состояния (биты регистра STATUS)		Примечания
	PIC16xxxx	PIC18xxx2	
ADDLW	C, DC, Z	C, DC, Z, **OV**, **N**	
ADDWF	C, DC, Z	C, DC, Z, **OV**, **N**	
ANDLW	Z	Z, **N**	
ANDWF	Z	Z, **N**	
COMF	Z	Z, **N**	
DECF	Z	**C**, **DC**, Z, **OV**, **N**	
INCF	Z	**C**, **DC**, Z, **OV**, **N**	
IORLW	Z	Z, **N**	
IORWF	Z	Z, **N**	
MOVF	Z	Z, **N**	
RETFIE	GIE	GIE/GIEH, PEIE/GIEL	Имена и функции этих битов изменены вследствие введения приоритетов прерываний.
RLF	C, **DC**, Z	C, Z, **N**	Сдвиг влево с использованием бита переноса.
RRF	C, **DC**, Z	C, Z, **N**	Сдвиг вправо с использованием бита переноса.
SUBLW	C, DC, Z	C, DC, Z, **OV**, **N**	
SUBWF	C, DC, Z	C, DC, Z, **OV**, **N**	
XORLW	Z	Z, **N**	
XORWF	Z	Z, **N**	

Примечание: биты, на которые оказывается влияние при переходе от PIC16xxxx к PIC18xxx2 выделены **жирным** шрифтом.

ТАБЛИЦА 3-2: Команды, вышедшие из употребления.

Команды PIC16xxxx	Биты состояния PIC16xxxx	Что нужно сделать
CLRW	**Z**	Вместо этой команды использовать CLRF WREG.

Примечание: биты, на которые оказывается влияние при переходе от PIC16xxxx к PIC18xxx2 выделены **жирным** шрифтом.

ТАБЛИЦА 3-3: Новые команды, которые были введены в PIC18xxx2.

Команды.	Биты состояния (биты регистра STATUS)	Примечания
ADDWFC	C, DC, Z, OV, N	Сложить содержимое регистров WREG, F и бит переноса.
BC		Условное ветвление, зависящее от бита переноса.
BN		Условное ветвление, зависящее от негативного бита состояния.
BNC		Условное ветвление, зависящее от бита переноса.
BNN		Условное ветвление, зависящее от негативного бита состояния.
BNOV		Условное ветвление, зависящее от бита OV.
BNZ		Условное ветвление, зависящее от бита Z.
BOV		Условное ветвление, зависящее от бита OV.
BRA		Безусловное ветвление.
BTG		Инверсия бита в регистре
BZ		Условное ветвление, зависящее от бита Z.
CPFSEQ		Ветвление, зависящее от результата беззнакового вычитания.
CPFSGT		Ветвление, зависящее от результата беззнакового вычитания.
CPFSLT		Ветвление, зависящее от результата беззнакового вычитания.
DAW	C	Десятичное выравнивание WREG.
DCFSNZ		Декремент регистра, пропуск следующей команды в случае ненулевого результата.
INFSNZ		Инкремент регистра, пропуск следующей команды в случае ненулевого результата.
LFSR		Переместить константу по адресу, указанному в FSR.
MOVF	Z, N	Переместить содержимое регистра.
MOVFF		Переместить содержимое одного регистра в другой.
MOVLB		Переместить константу в регистр выбора банка (BSR).
MULLW		Умножение константы на содержимое WREG.
MULWF		Перемножение содержимого регистра и WREG.
NEGF	C, DC, Z, OV, N	Дополнение до 2-х, инверсия знака 8-разрядных данных.
POP		Сброс верхней части стека, декремент указателя стека.
PUSH		Инкремент указателя стека, копирование программного счётчика в верхнюю часть стека.
RCALL		Вызвать подпрограмму по смещению.
RESET	Все биты	Сбросить контроллер.
RLNCF	Z, N	Сдвинуть влево без использования бита переноса.
RRNCF	Z,N	Сдвинуть вправо без использования бита переноса.
SETF		Установить все биты регистра.
SUBFWB	C, DC, Z, OV, N	WREG-f с заёмом (переносом).
SUBWFB	C, DC, Z, OV, N	f-WREG с заёмом (переносом).
TBLRD		Считать байт из таблицы в памяти программ.
TBLWT		Записать байт в таблицу в памяти программ.
TSTFSZ		Проверить регистр, пропустить следующую команду, если регистр = 0.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»*
(3432) 237-079, факс: (3432) 237-038
http://www.averon.ru
ic@averon.ru

ВОЗМОЖНЫЕ ПРОБЛЕМЫ И СПОСОБЫ ИХ РЕШЕНИЯ.

При написании программы помимо документации на применяемый контроллер изучайте **«Silicon and Datasheet Errata»** – документы с описанием обнаруженных ошибках в работе контроллера или допущенных неточностей в документации. Полный список подобных документов Вы можете найти на сайте **Microchip**: **http://www.microchip.com/1010/suppdoc/errata/index.ht**m

1. Проблемы, возникающие при записи в таблицу (для PIC18Fxx2).

Если во время записи в таблицу происходит периферийное прерывание, то возможно повреждение данных. Такая ситуация имеет место только в том случае, если был установлен бит разрешения соответствующего периферийного прерывания (в регистре **PIE** или **INTCON**).

Проблема решается следующим образом. Перед выполнением каких бы то ни было процедур записи в таблицу, следует **запретить ВСЕ периферийные прерывания**. Для этого нужно сбросить все биты в трёх регистрах, отвечающих за управление прерываниями (**INTCON**, **INTCON2**, **INTCON3**), а также в обоих регистрах, отвечающих за разрешение периферийных прерываний (**PIE1**, **PIE2**). По завершении записи в таблицу необходимо восстановить прежние значения вышеупомянутых регистров.

2. Проблемы, возникающие при записи и стирании FLASH-памяти (для PIC18Fxx2).

После сброса состояние бита **CFGS** (EECON1<6>), отвечающего за выбор нужной области памяти, не определено. Поэтому перед выполнением процедур записи или стирания необходимо должным образом сконфигурировать этот бит, чтобы гарантировать доступ именно к той области памяти, которая требуется. Применительно к памяти программ упомянутое конфигурирование бита означает его сброс.

ПРИМЕР 7: Код, реализующий процедуру стирания определённой области FLASH-памяти программ.

```
ERASE_ROW

      bsf   EECON1, EEPGD    ; Указывает на FLASH-память
                             ; программ.
      bcf   EECON1, CFGS     ; Доступ к FLSH-памяти программ
                             ; открыт.
      bsf   EECON1, WREN     ; Разрешить запись в память.
      bsf   EECON1, FREE     ; Разрешить операцию стирания.
```


*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru

ПРИМЕР 8: Код, реализующий процедуру записи в определённую область FLASH-памяти программ.

```
ERASE_BLOCK

     bsf   EECON1, EEPGD    ; Указывает на FLASH-память программ.
     bcf   EECON1, CFGS     ; Доступ к FLASH-памяти программ открыт.
     bsf   EECON1, WREN     ; Разрешить запись в память.
     bcf   EECON1, FREE     ; Разрешить операцию стирания.

. . .

PROGRAM_MEMORY

     bsf   EECON1, EEPGD    ; Указывает на FLASH-память программ.
     bcf   EECON1, CFGS     ; Доступ к FLASH-памяти программ открыт.
     bsf   EECON1, WREN     ; Разрешить запись в память.
     bcf   INTCON, GIE      ; Запретить прерывания.
```

ПАМЯТЬ EEPROM ДАННЫХ (для PIC18Fxx2).

В процессе чтения памяти EEPROM данных во время второго по счёту после выставления бита **RD** (EECON1<0>) командного цикла может произойти повреждение содержимого регистра **EEDATA**. Собственно содержимое EEPROM при этом не страдает.

Предлагается следующий вариант решения. Для гарантированного сохранения целостности содержимого регистра **EEDATA** он должен быть прочитан сразу после выставления бита **RD**. Целесообразно воспользоваться командами **`movf`** и **`movff`** (см. ***Пример 9***).

Кроме того, перед чтением следует запретить все прерывания.

ПРИМЕР 9: Предлагаемый алгоритм чтения EEDATA.

```
bcf   INTCON, GIEH     ; Запретить все прерывания.
bsf   EECON1,RD        ; Начать процедуру чтения.
movf  EEDATA,W         ; Сохранить EEDATA в рабочем регистре.
bsf   INTCON,GIEH      ; Разрешить прерывания.
. . .
```

Данная статья составлена на основе следующих документов:
AN716: Migrating Designs from PIC16C74A/74B to PIC18C442 (DS00716A)
PIC16Fxxx2 Rev.B3 Silicon/Data Sheet Errata (DS80127A)

По вопросам и замечаниям обращайтесь на e-mail: **support@averon.ru**

Июль 2002г.

*620086, г. Екатеринбург, ул. Чкалова, 3, «**АВЕРОН**»* ✉
(3432) 237-079, факс: (3432) 237-038 ☎
http://www.averon.ru
ic@averon.ru