

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ДГТУ

Кафедра «Приборостроение»

ФОРМИРОВАНИЕ ИНТЕРВАЛОВ ВРЕМЕНИ

Методические указания к лабораторной работе по дисциплинам
«Автоматизированные системы сбора и обработки измерительной
информации», «Микропроцессорная техника»

Ростов-на-Дону 2012

УДК 004.382.7

Составители: к.т.н., доц. А.В. Литвин
к.т.н., доц. К.А. Мороз
ст. преподаватель И.Н. Нестеренко
ст. преподаватель В.Н. Сыроватка

Формирование интервалов времени: Метод. указания к лабораторной работе по дисциплинам «Автоматизированные системы сбора и обработки измерительной информации» и «Микропроцессорная техника». – Ростов н/Д: Издательский центр ДГТУ. 2012. – 16 с.

В методических указаниях приводятся цель работы, краткое описание способов формирования задержек, содержание и порядок выполнения лабораторной работы, рассмотрены примеры фрагментов программ для микроконтроллера. Предназначены для студентов очной и заочной форм обучения специальностей «Приборостроение» и «Стандартизация и сертификация».

Печатается по решению методической комиссии факультета «Приборостроение и техническое регулирование»

Рецензент к.т.н., доцент П. С. Обухов

Научный редактор к.т.н., проф. В.Н. Ананченко

© Издательский центр ДГТУ, 2012

1. Цель работы. Изучение способов формирования интервалов времени программными и аппаратными средствами микроконтроллера (МК) PIC16F628A. Приобретение практических навыков по созданию фрагментов программ формирующих заданные интервалы времени (задержки).

2. Краткие сведения из теории.

Работа МК и его взаимодействие с внешними устройствами происходят во времени, поэтому практически в каждой программе производится выдержка интервалов времени или его измерение.

Существует два способа формирования интервалов времени, которые условно можно разделить на программный и аппаратный. Программный способ основан на том, что каждая команда программы выполняется в течение определенного времени. Объединяя команды в группы и используя циклы можно сформировать любой интервал времени, который будет равен времени выполнения данных команд. Несмотря на простоту и универсальность метода, он обладает существенным недостатком - процессор МК занят выполнением данных команд, и не может выполнять команды основной программы. Поэтому, формируемые данным методом интервалы времени называют также задержками.

Аппаратный способ основан на использовании встроенных в МК модулей таймеров (TMR0, TMR1, TMR2). Аппаратный способ более универсален, и позволяет реализовывать не только задержки, как при программном способе, но и отсчитывать интервалы времени одновременно с исполнением основной программы, для чего необходимо задействовать прерывания по переполнению таймеров. В данной лабораторной работе использование системы прерываний не рассматривается.

2.1. Программный способ формирования задержек

Приступая к созданию задержек данным способом необходимо определить время одного машинного цикла, а также знать за какое количество машинных циклов выполняется та или иная команда.

Время одного машинного цикла для МК серии PIC16 определяется как величина обратная частоте следования данных циклов. При этом частота циклов получается делением на 4 частоты тактовых импульсов. Например, если частота тактовых импульсов равна 4 МГц (20 МГц), то частота машинных циклов будет равна 1 МГц (5 МГц), при этом время одного машинного цикла составит 1 мкс (200 нс).

В силу того, что МК серии PIC16 обладают RISC архитектурой, большинство их команд выполняются за 1 машинный цикл (м.ц.).

Исключением являются команды перехода (call и goto) (по 2 м.ц.), возврата из подпрограмм (return, retlw, retfie) (по 2 м.ц.) и команды ветвления (decfsz, incfsz, btfsz, btfs), которые выполняются за 1 или 2 м.ц. в зависимости от сценария их работы.

В качестве примера приведем фрагмент программы, формирующий программную задержку длительностью в 1 секунду. Пусть частота следования тактовых импульсов равна 4 МГц, следовательно, время одного машинного цикла составит 1 мкс.

Фрагмент программы:

```

.... ; команды основной программы
call Pause1s ; Вызов подпрограммы (ПП) Pause1s
.... ; команды основной программы
; Подпрограмма формирования задержки в 999998 мкс.
Pause1s movlw .249 ; Начало п.п. Запись числа 249 в регистр W
movwf n1 ; Скопировать 249 из W в регистр n1
X2 call Pause1ms ; Переход на ПП Pause1ms
call Pause1ms ; Переход на ПП Pause1ms
call Pause1ms ; Переход на ПП Pause1ms
call Pause1ms ; Переход на ПП Pause1ms
decfsz n1,1 ; Декремент содержимого регистра n1=n1-1
goto X2 ; Переход на новый цикл, если n1 ≠ 0.
call Pause1ms ; Если n1 = 0, то далее вызов ПП Pause1ms
call Pause1ms ; Переход на ПП Pause1ms
call Pause1ms ; Переход на ПП Pause1ms
movlw .82 ; Запись числа 82 в регистр W
movwf n2 ; Скопировать 82 из W в регистр n2
X3 decfsz n2,1 ; Декремент содержимого регистра n2=n2-1
goto X3 ; Переход на новый цикл, если n2 ≠ 0.
nop ; Калибровочный (пустой)машинный цикл
return ; Если n2 = 0, то выход из ПП Pause1s
; Подпрограмма формирования задержки в 998 мкс.
Pause1m movlw .199 ; Запись числа 199 в регистр W
s
movwf n ; Скопировать 199 из W в регистр n
X1 nop ; Калибровочный (пустой)машинный цикл
nop ; Калибровочный (пустой)машинный цикл
decfsz n,1 ; Декремент содержимого регистра n=n-1
goto X1 ; Переход на новый цикл, если n ≠ 0.
return ; Если n = 0, то выход из ПП Pause1ms

```

Рассмотрим работу представленного фрагмента программы. По условию задания с момента начала исполнения команды call Pause1s до момента выхода из подпрограммы (ПП) Pause1s должна пройти ровно 1 секунда. При времени исполнения одного машинного цикла в 1 мкс за одну секунду необходимо исполнить 1000000 машинных циклов. Данную

задачу невозможно решить линейно в силу необходимости большого количества памяти программ для списка выполняемых команд задержки, поэтому необходимо организовать циклы задержки. Данные циклы объединены в подпрограммы с именами `Pause1s` и `Pause1ms`. Первая вызывается в основной программе, вторая – многократно во время работы первой. Вызов любой ПП осуществляется командой `call`. Это команда условного перехода на ПП. Условным переход называется по тому, что при выполнении данной команды производится два действия, во-первых, происходит переход на ПП, во-вторых, в вершину стека записывается адрес возврата, то есть адрес той ячейки памяти, с которой продолжится выполнение программы после завершения работы ПП. В случае вызова ПП `Pause1s`, в стек будет занесен адрес ячейки памяти которая следует за ячейкой с командой `call Pause1s`. В конце ПП всегда следует команда возврата `return`, которая извлекает адрес возврата из стека и загружает его в счетчик команд. И работа программы продолжается с команды, следующей за командой `call`, вызвавшей условный переход. Стек имеет восемь уровней, что позволяет вызывать ПП внутри ранее вызванных ПП, что и происходит в тексте ПП `Pause1s`, где многократно вызывается ПП `Pause1ms`. При этом, адрес возврата также помещается в вершину стека, а тот что там находился ранее перемещается в ячейку следующую за вершиной. Такую операцию можно повторить не более восьми раз, иначе будут потеряны адреса возврата из ПП. (Указателя переполнения стека не существует.) Также необходимо учитывать, что из подпрограмм необходимо выходить в последовательности обратной той последовательности, в которой происходил вход. Поскольку, адреса возврата извлекаются только из вершины стека.

Команда `call` выполняется за 2 м.ц., следовательно команды ПП `Pause1s` должны выполняться в течение 999998 м.ц. Организовать такой продолжительный цикл возможно, но не рационально. Более рациональным представляется организовать часть команд в отдельную ПП `Pause1ms` со временем исполнения в 1 мс и многократно вызвать ее в ПП `Pause1s`.

Время выполнения строки программы `call Pause1ms` будет происходить ровно в течение 1 мс. Из этого времени, 2 мкс будет выполняться команда `call`, вызывающая ПП, а оставшиеся 998 мкс будут выполняться команды ПП.

Формат команды:

`call name` – вызов ПП с именем `name`, с помещением адреса следующей инструкции в вершину стека.

return – возврат из ПП, вершина стека загружается в счетчик команд.

Пример условного перехода :

call Pause1s ; Вызов ПП Pause1s, запись в вершину стека
адреса
; команды X.
X ; команда, следующая за командой call .
..... ; команды.
Pause1s ; Начало ПП с именем Pause1s
..... ; команды, составляющие ПП.
return ; возврат из ПП , извлечение из стека адреса
; инструкции X и загрузка его в счетчик команд
; переход на выполнение команды X.

ПП Pause1ms начинается со строки с именем Pause1ms в крайнем левом столбце программы. Имя ПП может содержать любые латинские буквы и цифры. Далее командой movlw записывается константа 199 в аккумулятор, а затем из аккумулятора командой movwf копируется в ячейку памяти ОЗУ с именем n. Число в n может принимать значение от 1 до 255. Данная ячейка необходима для организации цикла задержки, в ней будет сохраняться переменная цикла задержки. Следующая строка ПП начинается с метки X1, с которой начинаются команды цикла, который будет повторяться многократно. И далее идут две пустые команды пор, которые не выполняют никаких действий и лишь служат для организации задержки в 1 м.ц. каждая. Следом, выполняется команда decfsz n,1 , которая с одной стороны уменьшает значение содержимого регистра-счетчика n на единицу и сохраняет полученное значение в нем же, с другой стороны, проверяет, является ли полученный результат нулевым. Если результат операции не равен нулю, то выполняется команда goto X1 и ПП переходит на новый цикл задержки. Если – равен нулю, то команда goto X1 пропускается, и происходит выполнение команды return и выход из ПП. Таким образом, с помощью команды decfsz возможна организация ветвлений программы, то есть организация различных сценариев её выполнения.

Команда goto осуществляет безусловный переход на метку. Является командой управления. Выполняется за 2 м. ц.. Безусловным переход называется потому, что никаких возвратов не предусматривается, программа просто переходит (совершает скачок) на строку, которая начинается с указанной метки и выполнение программы продолжается с команды расположенной на этой строке.

Формат команды:

goto Metka – переход без условия на строку с именем Metka.

Пример перехода:

goto Metka ; безусловный переход на строку с именем Metka.

..... ; команды

Metka ; строка программы, с именем Metka в левом столбце

Вычислим время выполнения подпрограммы. Команды mov и por выполняются каждая за 1 м. ц., то есть за 1 мкс каждая. Команды goto и return – за 2 м. ц., то есть за 2 мкс. Команда decfsz выполняется за 1 м. ц. (за 1 мкс), если результат декремента не равен нулю и за 2 м. ц. (за 2 мкс), если результат равен нулю и пропускается следующая инструкция.

2mov 2nop decfsz goto return

$2 + \{2 + 1 + 2\} * 199 - 1 + 2 = 998 \text{ м.ц. или } 998 \text{ мкс.}$

В формуле происходит умножение времени выполнения группы команд составляющих цикл задержки на величину константы 199. Минус 1 в формуле, означает, что в момент, когда результат операции равен нулю, команда decfsz выполняется за 2 м.ц. цикла и пропускает команду goto. В то время, как при ненулевом результате операции команда decfsz выполняется за 1 м.ц., а команда goto – за 2 м.ц. Нулевой результат появляется лишь единожды. Следовательно, необходимо вычесть единицу. Таким образом, сформирована ПП задержки в 1 мс.

Аналогичным образом формируется ПП Pause1s. Вызов ПП Pause1s производится командой call, которая выполняется за 2 м.ц., следовательно команды ПП Pause1s должны выполняться в течение 999998 м.ц. Помимо выполнения команд в тексте ПП Pause1s вызывается и ПП Pause1ms. Подпрограмма начинается со строки обозначенной именем Pause1s в крайнем левом столбце. Первыми командами происходит загрузка константы 249 в регистр-счетчик (ячейку ОЗУ) n1. Это две команды mov. Со строки с меткой X2 начинается группа команд, составляющих цикл задержки. Четыре команды call Pause1ms четыре раза вызывают на исполнение подпрограмму задержки в 1 мс. Далее, стоит команда decfsz, которая организует декремент и проверку на ноль содержимого регистра-счетчика проходов n1. Пока результат операции не нулевой, происходит переход на метку X2 с помощью команды goto. Как только результат равен нулю, команда goto пропускается, и происходит выполнение следующей команды.

Формат команды:

decfsz f,d – декрементировать содержимое регистра f с пропуском следующей инструкции если результат равен 0. Если результат не равен

0, то следующая инструкция выполняется. Если указатель $d = 0$, результат сохраняется в регистре W . Если $-d = 1$, результат сохраняется в регистре f .

`incfsz f,d` – инкрементировать содержимое регистра f с пропуском следующей инструкции если результат равен 0.

Пример организации ветвления программы :

`decfsz n,1` ; $n = n-1$, ($d=1$), и проверка n на ноль. -

`goto X1` ; если $n \neq 0$, переход на метку $X1$.

`goto Y1` ; если $n = 0$, то переход на метку $Y1$.

..... ; команды.

$X1$; строка с меткой $X1$ в крайнем левом столбце.

..... ; команды.

$Y1$; строка с меткой $Y1$ в крайнем левом столбце.

..... ; команды.

Определим время выполнения данного фрагмента подпрограммы:

`2mov 4call Pause1ms decfsz goto return`

$2 + \{4 \cdot 1000 + 1 + 2\} \cdot 249 - 1 + 2 = 996750 \text{ м.ц.}$

Полученный результат означает, что после рассмотренных команд, и до команды `return` необходимо вставить группу команд, суммарное время выполнения которых составило бы $999998 - 996750 = 3248$ мкс. Три мс. из данного числа компенсируются троекратной обработкой ПП задержки в 1 мс. (Три строки `call Pause1ms`). Оставшиеся 248 мкс. компенсируются обработкой локального цикла задержки. Загружается константа 82 в регистр-счетчик проходов $n2$. Цикл задержки реализован лишь на основе многократного выполнения команд `decfsz` и `goto`. Определим время выполнения команд, оставшихся не учтенными:

`2mov decfsz goto`

$2 + \{1 + 2\} \cdot 82 - 1 = 247 \text{ м.ц. или } 247 \text{ мкс}$

Для компенсации недостающего времени в 1 м.ц. вставим одну команду `por` перед командой `return`. Таким образом, будет сформирована временная задержка в 1 секунду с точностью до 1 мкс.

2.1. Аппаратный способ формирования задержек

При данном способе формирования задержек используются аппаратные таймеры: $TMR0$, $TMR1$, $TMR2$. Рассмотрим особенности применения $TMR0$ и $TMR1$.

Таймер $TMR0$ - 8 разрядный таймер-счетчик с возможностью чтения и записи его текущего содержимого. Регистр ($TMR0$) данного таймера расположен в банке 0 по адресу $01h$. Если какое-либо число

записывается в него, то данная процедура называется предустановкой таймера. Для расширения емкости счета совместно с таймером может использоваться 8 разрядный программируемый предделитель. Управление настройками таймера происходит через регистр Option_Reg (адрес 81h, банк 1), (см. [6], приложение Г). Подключением предделителя управляет бит 3 PSA, при установке которого в 0 происходит его подключение перед таймером. Выбор коэффициента деления предделителя происходит битами 2-0 PS2-PS0, который может принимать стандартные значения кратные 2. Так, например, если коэффициент равен 4, то на счетчик таймера будет поступать каждый четвертый импульс тактового сигнала. При этом данный сигнал может поступать с внешнего источника на вывод МК, или же с внутренних цепей МК. В последнем случае его частота будет равна частоте машинных циклов МК. Выбор источника сигнала происходит битом 5 T0SC. Именно после установки данного бита в 0 начинается отсчет импульсов от внутреннего источника импульсов. Соответственно установка в 1 бита 5 останавливает счет. При этом для исключения начала счета импульсов поступающих на линию RA4 рекомендуется данную линию порта настроить на выход.

После переполнения таймера флаг прерывания по переполнению TMR0 устанавливается в единицу. Данный флаг находится в составе регистра управления прерываниями INTCON (адрес 0Bh, банк 0), под именем T0IF и номером 2 (см. приложение А) Но поскольку в данной работе прерывания не рассматриваются то необходимо организовать программную проверку данного флага. После переполнения таймера данный флаг необходимо программно сбросить в ноль для возможности идентификации следующего переполнения.

Таймер TMR1 - 16 разрядный таймер-счетчик с возможностью чтения и записи его текущего содержимого. Содержит два восьми разрядных регистра с именами TMR1H и TMR1L, расположенных в банке 0 по адресам 0Fh и 0Eh соответственно. Младшие восемь разрядов таймера расположены в регистре TMR1L, старшие – в TMR1H. В оба регистра можно записывать данные, осуществляя, таким образом, предустановку таймера. Управление данным модулем производится посредством изменения содержимого регистра T1CON (адрес 10h, банк 0), (см. [6], приложение Д). Совместно с таймером используется предделитель с коэффициентами деления входной частоты на 1,2,4,8 соответственно. Выбор значения коэффициента производится установкой битов 4 и 5. Далее, при настройке указывается источник тактового сигнала установкой бита TMR1CS под номером 1. В данной работе выберем внутренний источник сигнала (импульсы следуют с частотой м.ц.). Бит номер 1 равен 0. Включение таймера производится установкой

в 0 бита TMR1ON под номером 0. После чего таймер начинает счет. Для контроля момента переполнения таймера необходимо проверять флаг прерывания по переполнению TMR1 с именем TMR1IF, который расположен под номером 0 в регистре флагов прерываний PIR1 (адрес 0Ch, банк 0), (см. приложение Б). После переполнения данный флаг необходимо сбросить программно.

В качестве примера использования таймеров составим ПП задержки для таймера 0 и таймера 1.

Фрагмент программы с примером использования таймера 0:

```

....                ; команды основной программы
call    PauseTMR0   ; Вызов ПП PauseTMR0
....                ; команды основной программы
; Подпрограмма формирования задержки с использованием TMR0
PauseTMR clrf        TMR0      ; Очистка регистра таймера
0
                clrf    INTCON    ; Запрет прерываний и очистка флага 2
                bsf     Status,5   ; Бит № 5 = 1, переход в 1 банк.
                movlw   b'00000011' ; Кдел. = 16, биты 0-2, бит №3=0, №5=0
                movwf   Option_Reg ; Настройка TMR0, начало счета.
                bcf     Status,5    ; Бит № 5 = 0, переход в 0 банк.
xx1         btfss    INTCON,2      ; Проверка флага 2 переполнения TMR0
                goto    xx1        ; Флаг 2 = 0, повторный анализ флага
                bsf     Status,5    ; Флаг 2 = 1, переполнение.Переход 1 б.
                bsf     Option_Reg, ; Бит № 5 = 0, остановка счета имп.
                    5
                bcf     Status,5    ; Бит № 5 = 0, переход в 0 банк.
                return              ; Выход из ПП PauseTMR0

```

Рассмотрим текст подпрограммы с примером использования таймера 0. Вызов ПП производится из основной программы командой call PauseTMR0. ПП PauseTMR0 начинается с очистки регистра таймера TMR0 командой clrf TMR0. Действие данной команды аналогично записи в регистр TMR0 нуля. Точно также очищается регистр управления прерываниями INTCON. После выполнения данной команды все биты регистра INTCON (см. приложение А) установлены в ноль, этим достигается как запрет прерываний (биты 7-3), а также и сброс флага 2 прерывания по переполнению TMR0. Далее производится настройка таймера 0, путем изменения содержимого регистра Option_Reg (см. [6], приложение Г). Для работы с данным регистром производится переход в банк 1. Битами 0-2 производим выбор коэффициента делителя, установкой бита №3 в 0 подключаем делитель перед таймером 0, а установкой бита №5 в 0 выбираем внутренний источник тактового сигнала. В тексте программы сформированное число, соответствующее

Формат команды:

clrf f – очистить регистр f.

Пример обнуления регистра :

clrf CCPCON ; очистка (обнуление) регистра CCPCON.

Примечание: очистка (обнуление) аккумулятора производится командой clrw, очистка сторожевого таймера - командой clrwtd.

данной настройке копируется в аккумулятор, а затем из него в регистр Option_Reg. Именно по завершению выполнения команды movwf Option_Reg происходит применение настроек и начинается счет импульсов. Далее, происходит переход в банк 0. Таймер начинает подсчитывать поступающие импульсы. Он считает их непрерывно и после переполнения (переход от значения в регистре TMR0 от FF к 00) продолжает новый цикл счета (с 00 и далее к FF). Для определения момента переполнения таймера необходимо организовать опрос флага переполнения, который при этом устанавливается в 1. Проверку данного флага возможно организовать с помощью бит ориентированной команды ветвления, предназначенной для проверки значения одного бита в регистре. Существует две такие команды btfss и btfsc.

Формат команды:

btfss f,b – проверить бит № b в регистре с именем f . Если бит № b равен 1 , то пропустить следующую инструкцию (пропуск если единица).

btfsc f,b – проверить бит № b в регистре с именем f . Если бит № b равен 0 , то пропустить следующую инструкцию (пропуск если ноль).

Пример организации ветвления программы :

btfss INTCON,2 ; Проверка бита (флага) № 2 регистра INTCON.

goto X1 ; если бит № 2 = 0, переход на метку X1.

goto Y1 ; если бит № 2 = 1, то пропуск инструкции
; goto X1 и переход на метку Y1.

..... ; команды.

В строке btfss INTCON,2 производится проверка значения бита №2. Если оно равно 0 (не было переполнения) происходит переход на повторный опрос данного флага, до тех пор пока он не установится в 1 (прерывание произошло). В последнем случае будет пропущена инструкция goto xx1 и ПП продолжит свою работу. После переполнения таймера целесообразно остановить его счет. Что и производится установкой бита №5 регистра Option_Reg в единицу. При этом счетный вход таймера 0 подключается к выводу RA4 МК. И для того чтобы таймер

0 не продолжил счет импульсов, но на этот раз поступающих с вывода RA4, данный вывод должен быть настроен на выход, с тем, чтобы он не мог быть приемником сигнала с внешней схемы. Далее, происходит выход из ПП.

Произведем расчет временных характеристик ПП PauseTMR0. Команда call PauseTMR0, вызывающая ПП PauseTMR0 выполняется за 2 м.ц. Команды clrf, bsf, mov выполняются за 1 м.ц. каждая, с учетом их количества, все вместе они выполняются за 5 м.ц. После запуска счета импульсов, командой movwf Option_Reg, дальнейшее время выполнения подпрограммы будет определяться не временем исполнения команд, а емкостью регистра таймера 0. Так как регистр TMR0 8-разрядный, то от нуля до переполнения таймера может быть отсчитано 256 м.ц. Однако в рассмотренном примере, перед таймером включен предделитель с $K_{\text{дел}} = 16$. Следовательно 256 необходимо умножить на 16. (если предделитель не подключен, то $K_{\text{дел}} = 1$). Работа цикла проверки флага №2 переполнения таймера 0 продолжается до тех пор, пока он не установится в 1. При этом команда btfss INTCON,2 пропустит команду goto xx1 и будет выполнена за 2 м.ц. Данное время, как и время выполнения последующих команд необходимо учитывать, так как ожидание повторного переполнения таймера программой не предусмотрено. После выхода из цикла проверки производится остановка счета импульсов. При этом, для перехода из банка в банк и работы с регистром Option_Reg используется 2 команды bsf и одна – bcf. В сумме они выполняются за 3 м.ц. Выход из ПП происходит при выполнении команды return (за 2 м.ц.). В итоге ПП PauseTMR0 выполняется за 4110 м.ц. (мкс).

```
call 2clrf bsf 2mov btfss      2bsf bcf return
2 + 2 + 1 + 2 + 2 + { $K_{\text{дел}} * 256$ } + 2 + 1 + 2 = 14 + {16 * 256} = 4110 м.ц.
```

Рассмотрим текст подпрограммы с примером использования таймера 1. После вызова ПП PauseTMR1 производится обнуление старшего (TMR1H) и младшего (TMR1L) регистров таймера 1. Далее производится запрет прерываний обнулением регистра INTCON. Перед началом счета импульсов сбрасывается в исходное состояние (в 0) флаг переполнения таймера 1. Он расположен в регистре флагов прерываний PIR1 (адрес 0Ch, 1 банк, см. приложение Б) под номером № 0, и может быть сброшен (установлен в исходное состояние) только программным путем. Затем, формируется двоичное число для настройки таймера 1 на требуемый режим работы и копируется в аккумулятор (W). Управление таймером 1 производится через регистр T1CON [6], в него помещается ранее сформированное число. При этом, коэффициент предделителя выбирается равным 8 (биты №4 = №5 = 1), а счет таймер будет

производить от внутреннего источника импульсов, следующих с частотой машинных циклов (бит №1 = 0). Для включения таймера бит №0 устанавливается в 1. Таким образом, после записи ранее сформированного числа в T1CON таймер начинает свою работу согласно заданным параметрам.

Фрагмент программы с примером использования таймера 1:

```

.... ; команды основной программы
call    PauseTMR ; Вызов ПП PauseTMR1
      1
.... ; команды основной программы
; Подпрограмма формирования задержки с использованием TMR1
PauseTMR clrf    TMR1L ; Очистка младшего регистра таймера
1
      clrf    TMR1H ; Очистка старшего регистра таймера
      clrf    INTCON ; Запрет прерываний
      clrf    PIR1 ; Бит №0 = 0, сброс флага прерыв. TMR1
      movlw   b'00110001 ; Кдел. = 8, биты 5.4, бит 1 в 0, бит 0 в 1
      ,
      movwf   T1CON ; Настройка TMR1, начало счета имп.
      btfss   PIR1,0 ; Проверка флага №0 (PIR1) переп. тайм.
      goto    $-1 ; Флаг № 0 = 0, повторный анализ флага
      bcf     T1CON,0 ; №0=1,остановка счета, №0=0(T1CON)
      return ; Выход из ПП PauseTMR1

```

Момент переполнения таймера 1 определяется путем постоянной проверки командой `btfss PIR1,0` флага №0 (переполнения таймера 1) расположенного в регистре PIR1. Пока переполнение не произошло, данный флаг равен 0, и выполняется следующая команда `goto $-1`. Действие данной команды аналогично действию команды перехода на метку, только в данном случае метка не указывается, а происходит переход на команду расположенную на 1 адрес ранее чем данная команда, то есть на команду `btfss PIR1,0`. В момент переполнения таймера1 соответствующий флаг установится в 1. И команда `btfss PIR1,0` пропустит команду `goto $-1`, цикл проверки флага будет завершен. Далее, производится остановка счета импульсов таймером, для этого обнуляется бит №0 регистра T1CON. В ПП производится работа только в банке 0, поскольку все регистры, с содержимым которых выполняются действия, находятся в банке 0.

Расчет временных характеристик ПП `PauseTMR1` аналогичен расчету для ПП `PauseTMR0`. Основное отличие в том, что таймер 1 шестнадцатиразрядный, и содержит два регистра: TMR1L и TMR1H. Приращение числа во втором из них происходит при переполнении первого. Представленная ПП `PauseTMR1` обеспечит задержку в 524301м.ц.

call 4clrf 2mov btfss L H bcf return
 $2 + 4 + 2 + 2 + \{K_{\text{дел}} * 256 * 256\} + 1 + 2 = 13 + \{8 * 256 * 256\} = 524301 \text{ м.ц.}$

При использовании таймеров, возможно получить другие значения задержек. Это достигается следующими путями:

- выбор значений $K_{\text{дел}}$ предделителя;
- организация циклов, в которых происходит заданное количество переполнений таймера;
- однократная или многократная (на каждом цикле счета) предустановка значений в регистре(ах) таймера. При предустановке числа копируются в регистры (TMR0, TMR1H, TMR1L) до момента начала счета импульсов и тем самым, сокращая емкость регистра данного таймера.

Фактически полученные значения задержек, в приведенных примерах, будет отличаться от расчетных на 1-3 м.ц. в сторону увеличения. Это связано с тем, что переполнение таймера может произойти в любой момент цикла проверки флага переполнения. А данный цикл (команды btfss и goto) выполняется в течение 3 м.ц. Для точного измерения времени (до 1 м.ц.) необходимо использовать прерывания при переполнении таймера. Кроме того, использование прерываний позволяет осуществлять работу таймеров в «фоновом» режиме, одновременно с выполнением команд основной программы. Это достигается за счет того, что изменение состояния флага переполнения фиксируется системой прерываний, а не путем выполнения цикла проверки командами ассемблера, как в рассмотренных случаях.

3. Порядок выполнения работы

3.1. Согласно варианту выберите по таблице значения задержки, необходимой к реализации.

3.2. Составьте подпрограмму на ассемблере, реализующую программную задержку.

3.3. Приведите расчеты времени выполнения составленной подпрограммы.

3.4. Повторите пункты 3.2 и 3.3 для случая организации ранее заданной задержки с использованием аппаратных средств МК – таймеров: для четных вариантов задействовать таймер 0 , для нечетных – таймер 1.

Таблица – Варианты задания. Значения задержек.

Вар.	Время	Вар	Время	Вар.	Время	Вар.	Время
1	124,325 мс	16	426,913 мс	31	0,612 с	46	978 мкс
2	726 мкс	17	534 мс	32	379,24 мс	47	99642 мкс

3	27,130 мс	18	14,257 мс	33	78,013 мс	48	44 мс
4	1,146112 с	19	46,273 мс	34	7632 мкс	49	1,36 с
5	725,140 мс	20	1,101143 с	35	5,463 мс	50	425,365 мс
6	1,174 с	21	95,474 мс	36	9,56 мс	51	341,258 мс
7	5,413 мс	22	145,734 мс	37	0,97624 с	52	644,2 мс
8	2,000104 с	23	675 мкс	38	163245 мс	53	3645 мкс
9	513,513 мс	24	6,431 с	39	9563 мкс	54	36,452 мс
10	474,014 мс	25	31,012 мс	40	246,563 мс	55	904,503 мс
11	1925 мкс	26	391,415 мс	41	73,265 мс	56	604,78 мс
12	1,001400 с	27	431,926 мс	42	1,00012 с	57	33,045 мс
13	147,456 мс	28	735 мкс	43	624,345 мс	58	1,00455 с
14	7,425 мс	29	1,040826 с	44	2345 мс	59	376,451 мс
15	1,473456 с	30	113,465 мс	45	44561 мкс	60	924,112 мс

4. Содержание отчета

- 4.1. Цель работы.
- 4.2. Сведения о командах ассемблера, используемых в работе.
- 4.3. Описание порядка настройки таймеров.
- 4.4. Тексты подпрограмм, задающих интервал времени.
- 4.5. Расчет временных параметров составленных подпрограмм.

5. Контрольные вопросы для проверки готовности студентов к выполнению лабораторной работы.

- 5.1. Какие регистры называются регистрами специального назначения?
- 5.2. Опишите состав и назначение регистра STATUS?
- 5.3. В чем отличие рабочих битов от флагов? Перечислите известные Вам флаги.
- 5.4. Опишите особенности команд movlw, movf, movwf?
- 5.5. Опишите команды bcf, bsf?
- 5.6. Для каких целей используются команды decfsz и incfsz в программах? Приведите примеры использования данных команд.

6. Контрольные вопросы по итогам лабораторной работы.

- 6.1. Опишите состав и назначение регистров INTCON, PIR1, Option_Reg?
- 6.2. Для чего предназначены регистры TMR0, TMR1H, TMR1L?
- 6.3. Какие действия выполняют команды btfs и btfsc?
- 6.4. Какими командами можно заменить команду clrf?
- 6.5. В чем отличие команд call и goto?
- 6.6. Опишите порядок работы с таймерами?
- 6.7. Как рассчитать временные характеристики подпрограммы?
- 6.8. Для чего предназначен СТЕК, опишите его структуру?

7. ЛИТЕРАТУРА.

- 7.1. Сайт компании MICROCHIP - www.microchip.ru
- 7.2. Корабельников Е.А. Руководство по конструированию устройств на микроконтроллерах. 2006. www.ikarab.narod.ru
- 7.3. Предько М.Справочник по PIC-микроконтроллерам.— М.: ДМК Пресс, 2002.
- 7.4. Ульрих В.А. Микроконтроллеры PIC16X7XX. — СПб.: Наука и техника, 2002.
- 7.5. Катцен Сид PIC-микроконтроллеры. Полное руководство. — М.: Додэка-XXI, 2010.
- 7.6. Литвин А.В. и др. Методические указания к лабораторной работе «Регистры специального назначения». Ростов-на-Дону: ДГТУ, 2007.

Редактор А.А. Литвинова

ЛР N 020639 от 26.04.96. В набор . .11. В печать . .11.
Офсет. Объем 1.00 усл.п.л., уч. - изд.л. Формат 60х80/16
Бумага тип N3. Заказ N . Тираж . Цена

Издательский центр ДГТУ

Адрес университета и полиграфического предприятия:

344010, Ростов-на-Дону, пл. Гагарина, 1

ПРИЛОЖЕНИЕ А

Регистр **INTCON** (адрес 0Bh)

GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
7	6	5	4	3	2	1	Бит 0
Бит 7	GIE	– Глобальное разрешение прерываний					
	1 =	разрешены все немаскированные прерывания					
	0 =	все прерывания запрещены					
Бит 6	PEIE	– Разрешение прерываний от периферийных модулей					
	1 =	разрешены все немаскированные прерывания периферийных модулей					
	0 =	прерывания от периферийных модулей запрещены					
Бит 5	TOIE	– Разрешение прерывания по переполнению TMR0					
	1 =	прерывание разрешено					
	0 =	прерывание запрещено					

Бит 4	INTE	– Разрешение внешнего прерывания INT 1 = прерывание разрешено 0 = прерывание запрещено
Бит 3	RBIE	– Разрешение прерывания по изменению сигнала на входах RB7:RB4 PORTB 1 = прерывание разрешено 0 = прерывание запрещено
Бит 2	T0IF	– Флаг прерывания по переполнению TMR0 1 = произошло переполнение TMR0 (сбрасывается программно) 0 = переполнения TMR0 не было
Бит 1	INTF	– Флаг внешнего прерывания INT 1 = выполнено условие внешнего прерывания на выводе RB0/INT 0 = внешнего прерывания не было
Бит 0	RBIF	– Флаг прерывания по изменению уровня сигнала на входах RB4:RB7 PORTB 1 = зафиксировано изменение уровня сигнала на одном из входов RB7:RB4 (сбрасывается программно) 0 = не было изменения уровня сигнала ни на одном из входов RB7:RB4

ПРИЛОЖЕНИЕ Б

Регистр **PIR1** (адрес 0Ch)

EEIF	CMIF	RCIF	TXIF	-	CCP1IF	TMR2IF	TMR1IF
7	6	5	4	3	2	1	Бит 0

Бит 7	EEIF	– Флаг прерывания по окончании записи в EEPROM данных 1 = запись в EEPROM данных завершена (сбрасывается программно) 0 = запись в EEPROM данных не завершена или не была начата
Бит 6	CMIF	– Флаг прерывания от компараторов 1 = изменилось состояние вывода компаратора 0 = состояние вывода компаратора не изменялось
Бит 5	RCIF	– Флаг прерывания от приемника USART 1 = буфер приемника USART полон 0 = буфер приемника USART пуст

Бит 4	TXIF	– Флаг прерывания от передатчика USART
	1 =	буфер передатчика USART пуст
	0 =	буфер передатчика USART полон
Бит 3	Не	реализован: читается как '0'
Бит 2	CCP1IF	– Флаг прерывания от модуля CCP1
Режим	1 =	выполнен захват значения TMR1 (сбрасывается программно)
захвата	0 =	захвата значения TMR1 не происходило
Режим	1 =	значение TMR1 достигло указанного в регистрах CCPR1H:CCPR1L(сбрасывается программно)
сравнени	0 =	значение TMR1 не достигло указанного в регистрах CCPR1H:CCPR1L
я		
Режим	ШИМ	Не используется
Бит 1	TMR2IF	– Флаг прерывания по переполнению TMR2
	1 =	произошло переполнение TMR2 (сбрасывается программно)
	0 =	переполнения TMR2 не было
Бит 0	TMR1IF	– Флаг прерывания по переполнению TMR1
	1 =	произошло переполнение TMR1 (сбрасывается программно)
	0 =	переполнения TMR1 не было
