Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Оренбургский государственный университет»

Кафедра промышленной электроники и
информационно-измерительной техники

А.В. Хлуденев

РАЗРАБОТКА И КОДИРОВАНИЕ АЛГОРИТМОВ ДЛЯ PIC-МИКРО

Методические указания

Рекомендовано к изданию редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего образования «Оренбургский государственный университет» для обучающихся по образовательной программе высшего образования по направлению подготовки 11.03.04 Электроника и наноэлектроника

Оренбург
2018

УДК 004.4 : 681.3.068
ББК 32.973.26
Х 60

Рецензент - кандидат технических наук, доцент С.А. Сильвашко

Хлуденев, А.В.
Х 60 Разработка и кодирование алгоритмов для PIC-микро: методические указания / А.В. Хлуденев; Оренбургский гос. ун-т. – Оренбург: ОГУ, 2018.

Методические указания содержат рекомендации по выполнению практических заданий по дисциплине "Отладочные средства микропроцессорных систем". Рассмотрены типовые задачи разработки программных средств для микроконтроллеров PIC-micro.

Методические указания предназначены для обучающихся по образовательной программе высшего образования по направлению подготовки 11.03.04 Электроника и наноэлектроника.

ББК 32.973.26
УДК 004.4 : 681.3.068

Содержание

1 Разработка простых программ

1.1 Описание задания

Необходимо разработать и кодировать на языке Ассемблера микроконтроллера (МК) PIC-micro алгоритм программы генератора чисел. Числовая последовательность формируется на линиях порта PORTD в соответствии с вариантом задания по таблице 1.1.

Таблица 1.1

Вариант	Задание
1	Формирователь последовательности чисел Фибоначчи 0, 1,…,233
2	Вычитающий двоичный счетчик по модулю 16
3	Суммирующий восьмиразрядный счетчик в унитарном коде
4	Вычитающий восьмиразрядный счетчик в унитарном коде
5	Суммирующий восьмиразрядный счетчик в коде Джонсона
6	Вычитающий восьмиразрядный счетчик в коде Джонсона

1.2 Рекомендации по выполнению

Алгоритм – точный набор инструкций, описывающих последовательность действий некоторого исполнителя для достижения результата, решения некоторой задачи за конечное время. Если исполнителем является процессорное ядро МК, то в конечном итоге набор инструкций необходимо представить в виде последовательности машинных команд – загрузочного кода программы. Кодированное представление алгоритма получают путем последовательного преобразования исходного описания, отличающегося меньшей степенью детализации и представленного в форме, удобной для восприятия человеком.

Исходное представление алгоритм программы обычно формируют по результатам анализа ее функциональной спецификации. По заданию основные функции программы:

- модификация содержимого ячейки Cnt (в соответствии с вариантом задания);
- вывод содержимого ячейки Cnt в порт PORTD.

На рисунке 1.1 приведен пример - схема программы, реализующей алгоритм формирования на линиях порта PORTD числовой последовательности по закону

суммирующего десятичного счетчика. При старте программа выполняет инициализацию порта вывода, инициализацию ячейки счетчика Cnt. После чего в теле цикла программы выполняется модификация (инкремент) содержимого ячейки Cnt и вывод ее содержимого в порт PORTD. Если содержимое счетчика превышает допустимое значение, определяемое модулем счета, счетчик обнуляется.

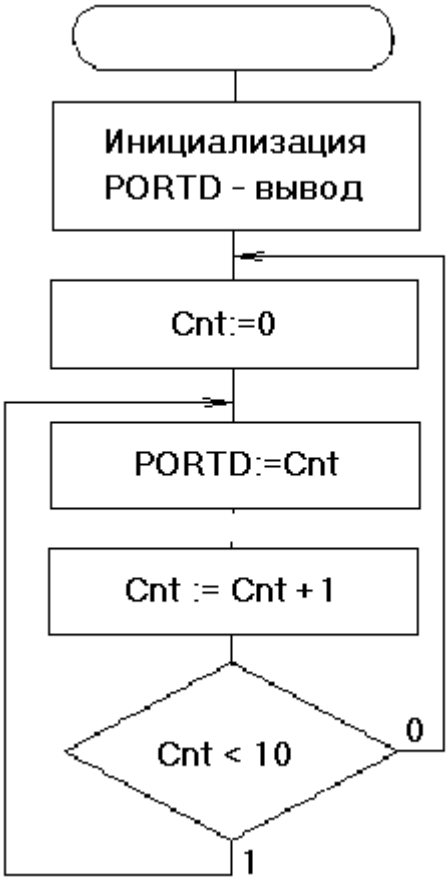

Рисунок 1.1

Теперь перейдем к кодированию алгоритма на языке Ассемблера - формированию исходного текста программы. Сводная таблица команд МК PIC16F87X приведена в приложении А, подробное описание приведено в документации изготовителя [1]. Рекомендуемые структуры исходного текста программ на языке Ассемблера для МК PIC-micro приводятся в шаблонах системы MPLAB, которые находятся в директории ..\MPASM Suite\Template\. Варианты шаблонов для получения абсолютного кода программы находятся в папке ..\Code. Для МК типа PIC16F877A шаблон находится в файле 16F877ATEMP.asm.

Упрощенный вариант шаблона с комментариями для программы, не использующей прерывания, приведен ниже:

```
        list    p=16f877a,st = OFF      ; директива листинга
        #include <p16f877a.inc>         ; определение переменных МК

        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON &
    _HS_OSC & _WRT_OFF& _LVP_OFF & _DEBUG_ON & _CPD_OFF
    ; директива установки разрядов слова конфигурации

;***** Определение переменных
; директивы определения переменных
;*****************************************************************
        ORG     0x000           ; вектор старта
        nop                     ; требуется для ICD

main
; операторы программы
        END                     ; директива окончания программы
```

Текст программы состоит из следующих разделов:

- директивы листинга, определения модели МК;
- директивы определения переменных;
- директивы указания вектора старта;
- операторы программы;
- директива окончания программы.

В программе используется одна переменная Cnt, выделим для нее первую свободную ячейку в банке 0, используя директиву EQU

```
Cnt     EQU     0x20            ; счетчик
```

Последовательность операторов программы должна соответствовать алгоритму. При старте программы должны выполняться операторы инициализации, а затем операторы основного цикла.

Для настройки линий PORTD на вывод в регистр TRISD необходимо загрузить управляющее слово b'00000000' или очистить регистр TRISD [1]. Регистр TRISD находится в банке 1, поэтому перед использованием оператора очистки регистра TRISD необходимо переключить соответствующие разряды (RP1,RP0) регистра

STATUS. После очистки регистра TRISD необходимо вернуть исходное состояние измененных разрядов (RP0) регистра STATUS. Для наглядности вместо инструкций установки и сброса разрядов регистра STATUS можно использовать директивы

```
BANKSEL  TRISD          ; банк 1 (адреса 80h - FFh),
BANKSEL  PORTD          ; банк 0 (адреса 00h - 7Fh).
```

При трансляции эти директивы будут заменены соответствующими инструкциями.

Особенностью системы команд МК PIC-micro является отсутствие команд сравнения и условных переходов. Поэтому эти действия необходимо выполнить, используя реализованные операции системы команд.

Сравнение слов информации обычно реализуют путем вычитания без сохранения разности. Результат сравнения можно оценить по значениям признаков выполнения операции (флагов): C – признака переноса и Z – признака нулевого результата. Вычитание константы можно заменить сложением с константой противоположного знака. В результате сложения:

```
movlw    -.10           ; число -10
addwf    cnt,w          ; прибавить к содержимому счетчика
```

флаг C установится, если условие cnt < 10 не выполняется. Ветвление алгоритма можно реализовать с помощью команд:

```
btfss    STATUS,C       ; если C=1
goto     inc            ; пропустить, иначе выполнить
goto     loop           ;.
```

Если C=0 (cnt<10), будет выполнен переход на метку Inc, иначе переход на метку Loop.

Сформированный исходный текст программы приведен ниже

```
list     p=16f877a,st = OFF          ; директива листинга
#include <p16f877a.inc>              ; определение переменных МК

    __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON &
_HS_OSC & _WRT_OFF& _LVP_OFF & _DEBUG_ON & _CPD_OFF
; директива установки разрядов слова конфигурации
```

```
;***** Определение переменных
Cnt         EQU         0x20                ; счетчик
;*****************************************************************
            ORG         0x000               ; вектор старта
Main        nop
            BANKSEL     TRISD               ; банк 1 (адреса 80h - FFh)
            clrf        TRISD^80            ; все линии PORTD на вывод
            BANKSEL     PORTD               ; банк 0 (адреса 00h - 7Fh)
Loop        clrf        Cnt                 ; обнулить счетчик
Inc         movf        Cnt,w               ;
            movwf       PORTD
            incf        Cnt,f               ; инкремент счетчика
            movlw       -.10                ; число -10
            addwf       Cnt,w               ; прибавить к содержимому счетчика
            btfss       STATUS,C            ; для сравнения
            goto        Inc                 ;
            goto        Loop
            END                             ; директива окончания программы.
```

2 Дискретный ввод-вывод

2.1 Описание задания

Необходимо разработать и кодировать на языке Ассемблера алгоритм программы дискретного ввода-вывода для МК типа PIC16F877A.

Нажатие с последующим отпусканием кнопки RC1 должно вызывать модификацию содержимого ячейки Cnt. Содержимое ячейки Cnt должно отображаться на светоизлучающих диодах, подключенных к линиям порта PORTD. Варианты заданий приведены в таблице 1.1. Для вариантов 1 – 2 информация должна быть представлена в десятичном коде, для вариантов 3 – 6 в шестнадцатеричном коде.

2.2 Рекомендации по выполнению

Функции программы:

- опрос кнопки, подключенной к линии порта PORTC (RC1);
- модификация содержимого ячейки Cnt;
- вывод содержимого ячейки Cnt в порт PORTD.

На рисунке 2.1 приведен пример - схема программы, реализующая алгоритм счетчика по модулю 10 числа нажатий на кнопку RC1.

При старте программы выполняется инициализация портов ввода и вывода, инициализация ячейки счетчика Cnt. После чего в теле цикла программы выполняется вывод содержимого ячейки Cnt в порт PORTD. Состояние линий порта PORTC копируется в ячейку PCh0.

Активному (нажатому) состоянию кнопки соответствует низкий потенциал на линии PORTC. Пассивному (отжатому) состоянию кнопки соответствует высокий потенциал, который обеспечивается подтягивающими резисторами. Все механические кнопки имеют негативное свойство, известное как «дребезг» контактов, которое обусловлено колебаниями упругих контактов при их замыкании и размыкании. Длительность колебаний составляет несколько миллисекунд. В устройствах на базе МК обычно используют программные способы подавления «дребезга» контактов. Простейший из них основан на ограничении минимальных интервалов времени между последовательными операциями анализа состояния входных линий порта. Для надежного считывания состояний кнопки величина этих интервалов не должна быть меньше 5 – 10 мс. Для подавления «дребезга» контактов в цикл основной программы включен оператор вызова подпрограммы задержки D5ms.

По комбинации значений 0 – 1 – 1 соответствующего разряда в ячейках PCh0, PCh1, PCh2 (копии PORTC в текущем и двух предыдущих циклах программы) определяется активность линии входного порта. Для этого вычисляется значение переменной

$$Key = \overline{PCh0} \,\&\, PCh1 \,\&\, PCh2$$

и анализируется состояние ее разряда Key,1. В случае если Key,1 = 1 (кнопка нажата), изменяется содержимое ячейки Cnt.

При вызове подпрограммы задержки D5ms происходит инициализация счетчика миллисекунд Count_ms. Подпрограмма выполняется, пока Count_ms не достигнет значения 0. Декремент переменной Count_ms через одну миллисекунду выполняет подпрограмма обработки прерываний INT.

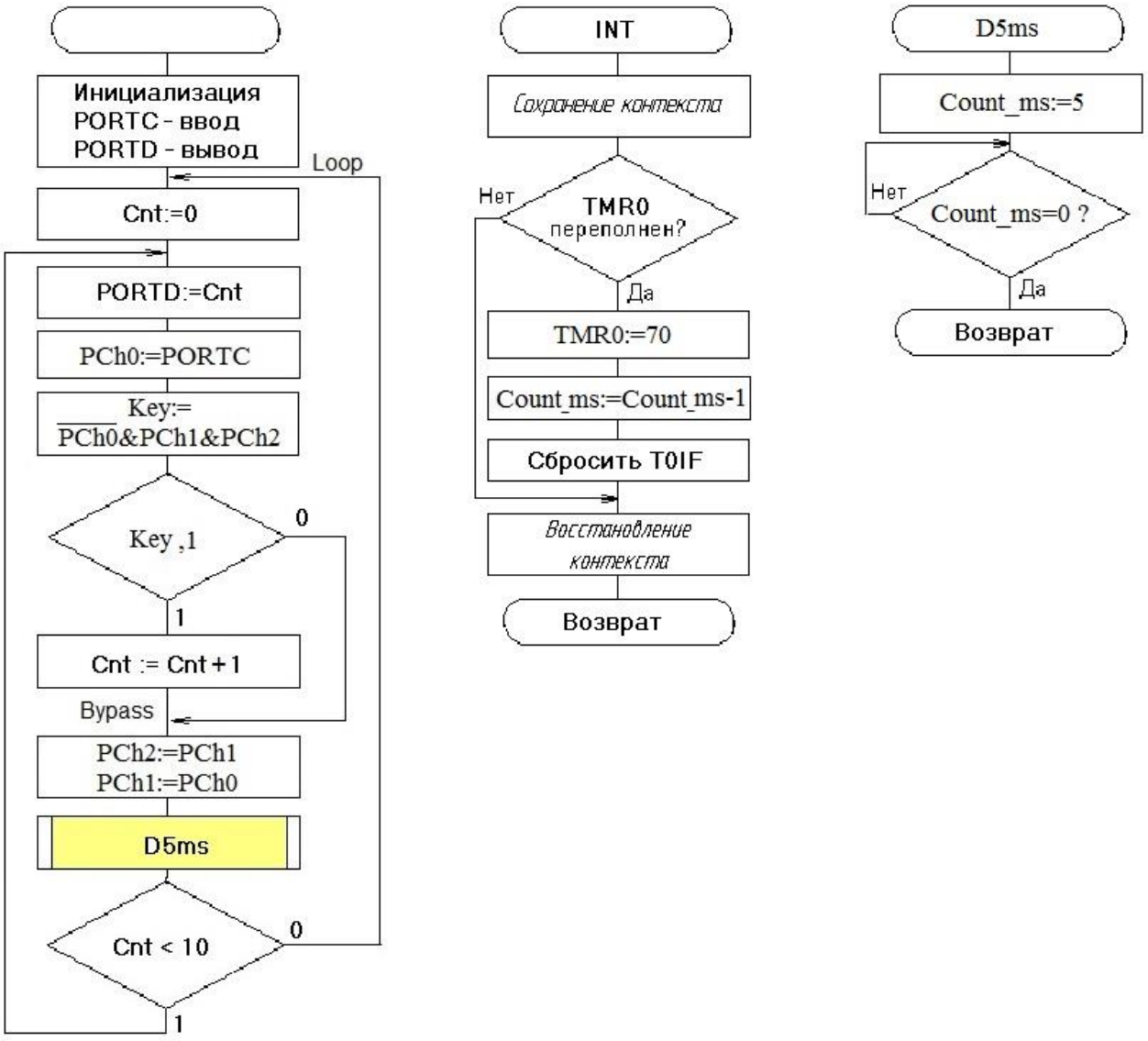

Рисунок 2.1

Функциональная схема таймера приведена на рисунке 2.2.

Режим работы таймера определяется состоянием разрядов регистра OPTION_REG (таблица 2.1).

Если бит T0CS=0, то TMR0 работает от внутреннего тактового генератора, инкремент счетчика происходит в каждом машинном цикле. На входе TMR0 может быть включен предделитель частоты, если бит PSA=0. Коэффициент деления определяется значениями разрядов PS2:PS0. Значение коэффициента деления частоты 1:16 получается при <PS2:PS0> = 011. Для настройки TMR0 на данный режим в регистр OPTION_REG необходимо загрузить управляющее слово b'10000011'.

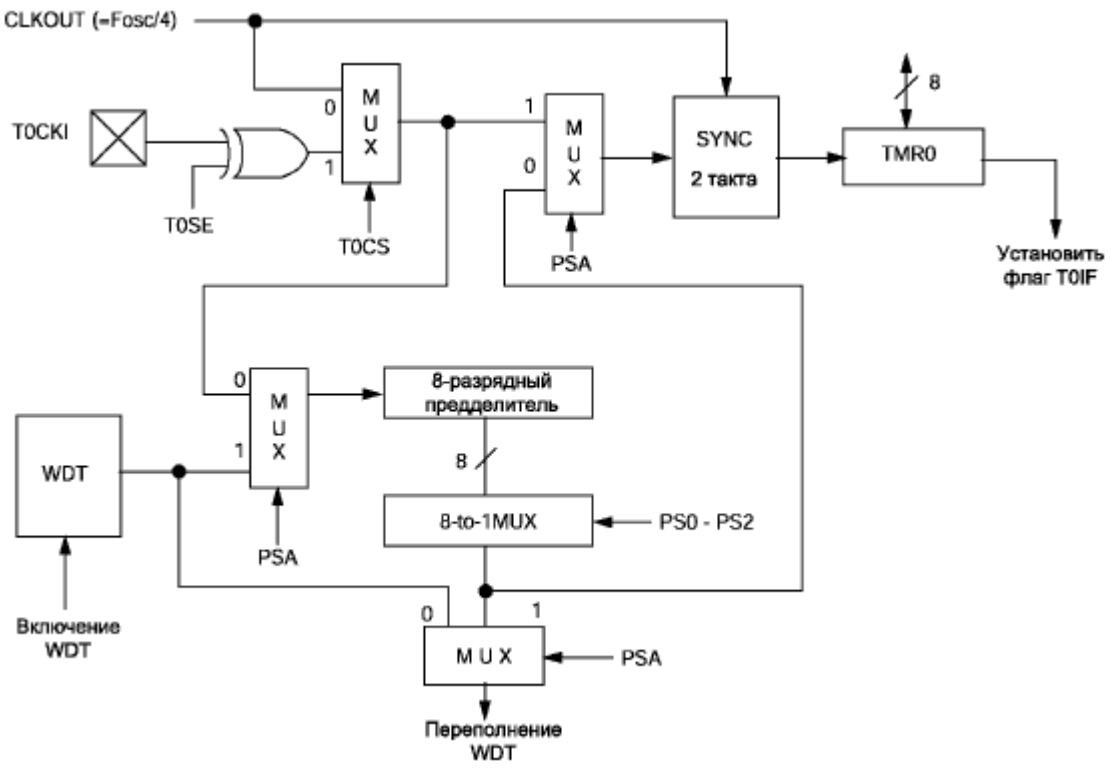

Рисунок 2.2

Таблица 2.1

Адрес	Имя	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
01h,101h	TMR0	Регистр таймера 0							
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
81h,181h	OPTION_REG	-RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

При переполнении таймера устанавливается флаг T0IF (регистр INTCON). После анализа флага, его необходимо программно сбросить и выполнить загрузку константы в таймер TMR0. При загрузке константы 70 при тактовой частоте МК F=12 МГц интервал времени между двумя переполнениями таймера составит:

$$T_{TMR} = 4 \cdot 2^4 \cdot 186/F = 4 \cdot 16 \cdot (256\text{-}70)/12 \cdot 10^6 = 0.992 \text{ мс}.$$

Погрешность формирования интервала 1 мс не превысит

$$(1 - 0.992) \cdot 100\ \% = 0.8\ \%.$$

При возникновении прерывания МК выполняет текущую команду, запоминает адрес следующей команды (точки возврата) в стеке и передает управление подпрограмме обработки прерывания. После обработки прерывания из стека извлекается

адрес точки возврата, и управление передается прерванному процессу. Это позволяет МК своевременно реагировать на различные события, при условии, что обработка прерываний занимает не слишком много времени. При возникновении нескольких прерываний от различных источников, их обработка выполняется последовательно в соответствии с приоритетами.

Для данной задачи прерывания будут возникать при переполнении таймера TMR0. Микроконтроллеры семейства PIC16F87x имеют 14 источников прерываний [1]. Логическая схема формирования запроса прерывания приведена на рисунке 2.3. Регистр INTCON содержит флаги отдельных прерываний, биты разрешения этих прерываний и бит глобального разрешения прерываний GIE.

Если бит GIE (INTCON,7) установлен в ‘1’, разрешены все немаскированные прерывания. Если GIE=0, то все прерывания запрещены. Каждое прерывание в отдельности может быть разрешено/запрещено установкой/сбросом соответствующего бита разрешения. При сбросе микроконтроллера бит GIE сбрасывается в ‘0’.

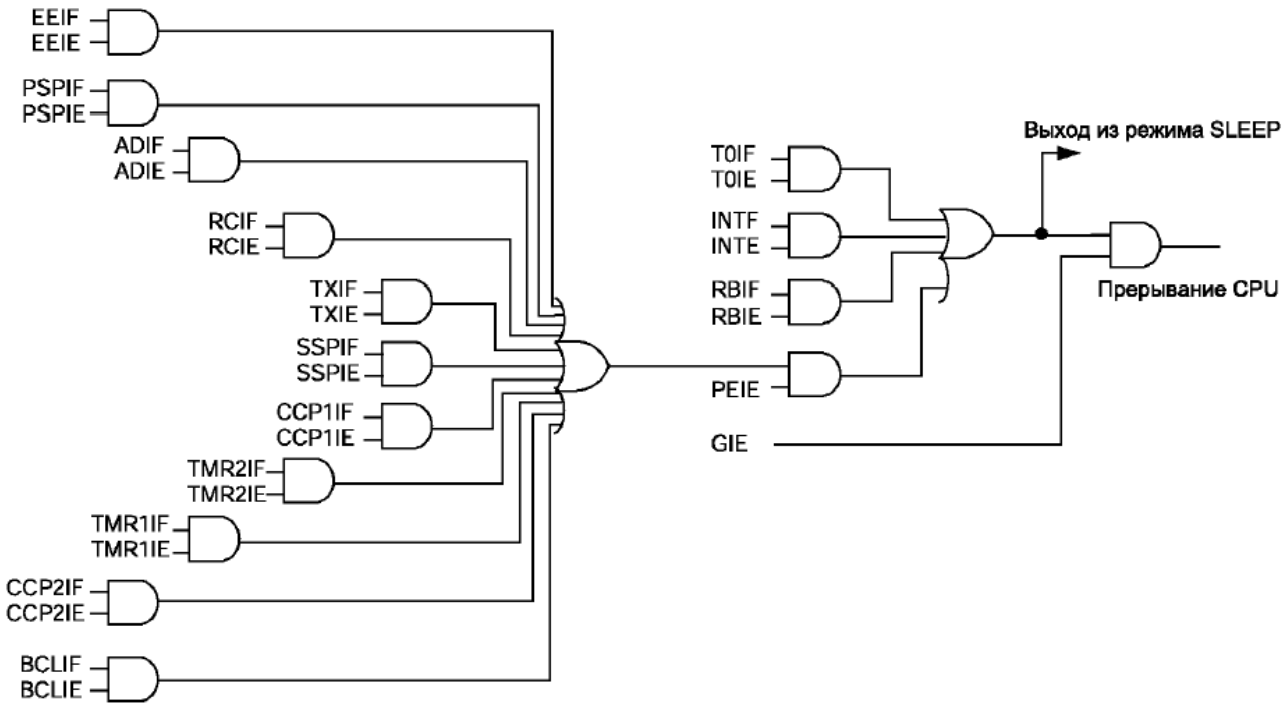

Рисунок 2.3

В регистре INTCON находятся флаги прерываний: от внешнего сигнала INT, от изменения уровня сигнала на входах RB7:RB4, от переполнения TMR0 (таблица 2.2). В регистрах PIR1, PIR2 содержатся флаги прерываний периферийных модулей

микроконтроллера, а в регистрах PIE1, PIE2 соответствующие биты разрешения прерываний. В регистре INTCON находится бит разрешения прерываний от периферийных модулей PEIE.

Таблица 2.2

Адрес	Имя	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0Ch	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1F	TMR2IF	TMR1IF
8Ch	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1E	TMR2IE	TMR1IE
0Dh	PIR2	-		-	EEIF	BCLIF	-	-	CCP2IF
8Dh	PIE2	-		-	EEIE	BCLIE	-	-	CCP2IE

При переходе в подпрограмму обработки прерываний бит GIE аппаратно сбрасывается в ‘0’, запрещая прерывания, адрес возврата из подпрограммы обработки прерываний помещается в стек, а в счетчик команд PC загружается вектор прерывания 0x0004. Источник прерываний можно определить проверкой флагов прерываний. Чтобы избежать повторного перехода на обработку прерывания необходимо сбросить установленный флаг обрабатываемого прерывания.

При возвращении из подпрограммы обработки прерывания, по команде RETFIE, бит GIE аппаратно устанавливается в ‘1’, разрешая все немаскированные прерывания. Прерывание при переполнении таймера TMR0 можно запретить, сбросив бит T0IE в регистре INTCON.

Микроконтроллеры PIC16F87x не имеют специальных команд для сохранения содержимого аккумулятора и специальных регистров во время обработки прерывания с последующим их восстановлением. Автоматически сохраняется только значение программного счетчика для возврата из подпрограммы. Поэтому необходимо программно сохранять контекст (содержимое аккумулятора W и регистра STATUS). С учетом этой особенности шаблон программы с комментариями приведен ниже:

```
list    p=16f877a, st = OFF          ; директива листинга
        #include <p16f877a.inc>      ; определение переменных МК

    __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON &
_HS_OSC & _WRT_OFF& _LVP_OFF & _DEBUG_ON & _CPD_OFF
```

```
; директива установки разрядов слова конфигурации

;***** Определение переменных
w_temp      EQU     0x70        ; переменные для сохранения контекста
status_temp EQU     0x71        ;
; директивы определения остальных переменных
;*****************************************************************
            ORG     0x000           ; вектор старта
            nop                     ; требуется для ICD
            goto    Main            ; переход на начало программы

            ORG     0x004           ; вектор прерывания
            movwf   w_temp          ; сохранение контекста
            movf    STATUS,w        ;
            movwf   status_temp     ;
; операторы подпрограммы обработки прерывания
Ret         movf    status_temp,w   ; восстановление контекста
            movwf   STATUS          ;
            swapf   w_temp,f
            swapf   w_temp,w        ;
            retfie                  ; возврат из подпрограммы

Main
; операторы программы
; операторы подпрограмм
            END                     ; директива окончания программы
```

Использование команды swapf для загрузки и выгрузки значений обусловлено тем, что эта команда не изменяет состояние флага нулевого результата Z регистра STATUS. Применение команды movf сократило бы программный код, но тогда при сохранении аккумулятора может быть изменено состояние бита Z, что в общем случае недопустимо. Корректное программирование требует, чтобы при возвращении из подпрограммы прерывания значения регистров W и STATUS были восстановлены абсолютно точно.

Операторы подпрограммы обработки прерывания:

```
            btfss   INTCON,T0IF     ;прерывание по переполнению TMR0?
            goto    Ret             ; нет
            movlw   .70
            movwf   TMR0            ;загрузить таймер
            decf    Count_ms,f      ;мс прошла
```

```
        bcf     INTCON,T0IF     ;сбросить флаг прерывания.
```

Для настройки линий PORTC на ввод в регистр TRISC необходимо загрузить управляющее слово b'11111111'. Для настройки линий PORTD на вывод в регистр TRISD необходимо загрузить управляющее слово b'00000000'.

Операторы инициализации:

```
Main    clrf    PORTD           ;инициализация модулей МК
        BANKSEL TRISD           ;bank1
        movlw   b'11111111'     ;все ввод
        movwf   TRISC^80h
        movlw   b'00000000'     ;все вывод
        movwf   TRISD^80h
        movlw   b'10000011'     ;режимы TMR0
        movwf   OPTION_REG^80h
        BANKSEL PORTD           ;bank0
        bsf     INTCON,GIE      ;разрешить прерывания
        bsf     INTCON,T0IE     ;разрешить прерывания от TMR0.
```

Операторы основного цикла программы:

```
Loop    clrf    Cnt             ;основной цикл
Inc     movf    Cnt,w
        movwf   PORTD           ;вывод на линейный индикатор
        movf    PORTC,w
        movwf   PCh0
        comf    PCh0,w
        andwf   PCh1,w
        andwf   PCh2,w
        movwf   Key
        btfss   Key,1           ;опрос кнопки "+"
        goto    Bypass          ;кнопка "+" не нажата
        incf    Cnt,f           ;инкремент счетчика
Bypass  movf    PCh1,w
        movwf   PCh2
        movf    PCh0,w
        movwf   PCh1
        call    D5ms            ;задержка
        movlw   -.10            ;число -10
        addwf   Cnt,w           ;прибавить к содержимому счетчика
        btfss   STATUS,C        ;для сравнения
        goto    Inc
        goto    Loop            ;в начало цикла.
```

Исходный текст подпрограммы задержки:

```
D5ms    movlw   .5
        movwf   Count_ms        ;загрузить счетчик миллисекунд
        movf    Count_ms,f      ;время истекло?
        btfss   STATUS,Z
        goto    $-2             ; нет
        return                  ;возврат в основную программу.
```

3 Вывод символьной информации

3.1 Описание задания

Необходимо разработать и кодировать на языке Ассемблера алгоритм программы вывода символьной информации для МК типа PIC16F877A.

Нажатие с последующим отпусканием кнопки RC1 должно вызывать модификацию содержимого ячейки Cnt. Содержимое ячейки Cnt должно отображаться на 7-сегментных индикаторах в десятичном коде для вариантов 1 – 3 и в шестнадцатеричном коде для вариантов 4 -6. Варианты заданий приведены в таблице 1.1.

3.2 Рекомендации по выполнению

Символьные индикаторы являются элементом человеко-машинного интерфейса (HMI). В простых задачах используют 7-сегментные индикаторы для отображения десятичных цифр и других символов, например, некоторых букв латинского алфавита. Схема включения 4-разрядного 7-сегментного индикатора с общим катодом приведена на рисунке 3.1. Аноды светоизлучающих сегментов всех разрядов подключим к линиям порта PORTD. Общий катод каждого разряда управляются транзисторным ключом. Управление ключами будут обеспечивать четыре линии порта PORTB.

Данное включение предполагает использование динамической индикации: на линии порта PORTD поочередно выводятся коды символов разрядов и открывается соответствующий транзисторный ключ. Чтобы мерцание индикаторов не было заметно, частота сканирования разрядов должна быть не менее 50 Гц. Если принять,

что в цикле программы длительностью 5 мс будет выводиться символ одного разряда, то период сканирования составит 20 мс.

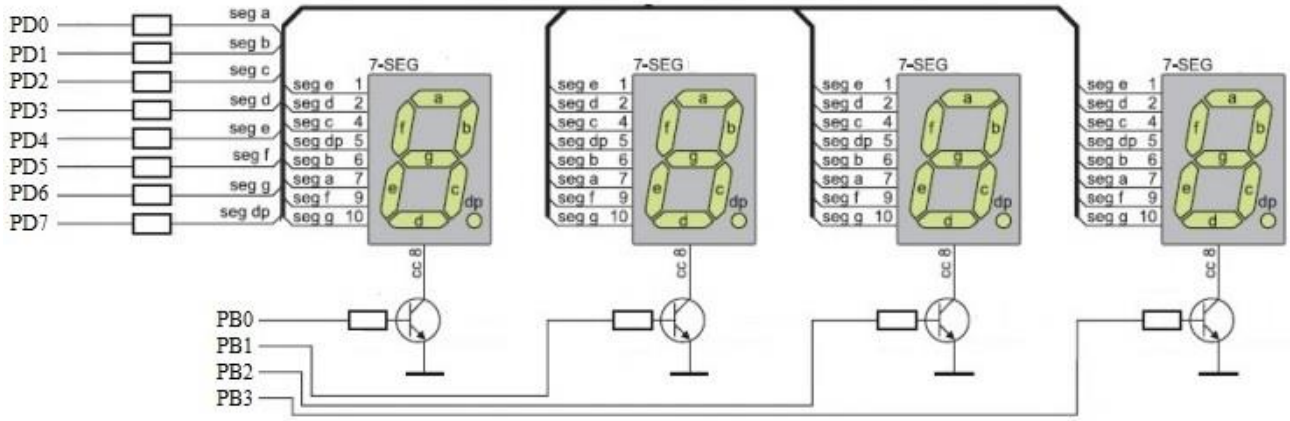

Рисунок 3.1

Схема алгоритма примера программы вывода в десятичном коде значения декадного счетчика приведена на рисунке 3.2. Выводимая информация – значение переменной Cnt преобразуется и выводится на 7-сегментные индикаторы.

Для этого используются:

- подпрограмма B2_BCD для преобразования двоичного кода в двоично-десятичный (аргумент функции двухбайтный двоичный код в ячейках H_byte, L_byte; значение функции трехбайтный BCD в ячейках R0,R1,R2);

- подпрограмма Pack, выполняющая переформатирование данных (аргумент функции двухбайтный BCD в ячейках R1,R2; значение функции 7-сегментный код разрядов в ячейках Dig4 – Dig1);

- подпрограмма вывода данных через порты Dis7seg.

Алгоритм функции B2_BCD и функции BINto7Seg, вызываемой из подпрограммы Pack для преобразования 4-разрядного двоичного кода символа в 7-сегментный код, на рисунке 3.2 не приведены. Программа использует подпрограмму задержки D5ms и подпрограмму обработки прерываний Int (алгоритм приведен на рисунке 2.2).

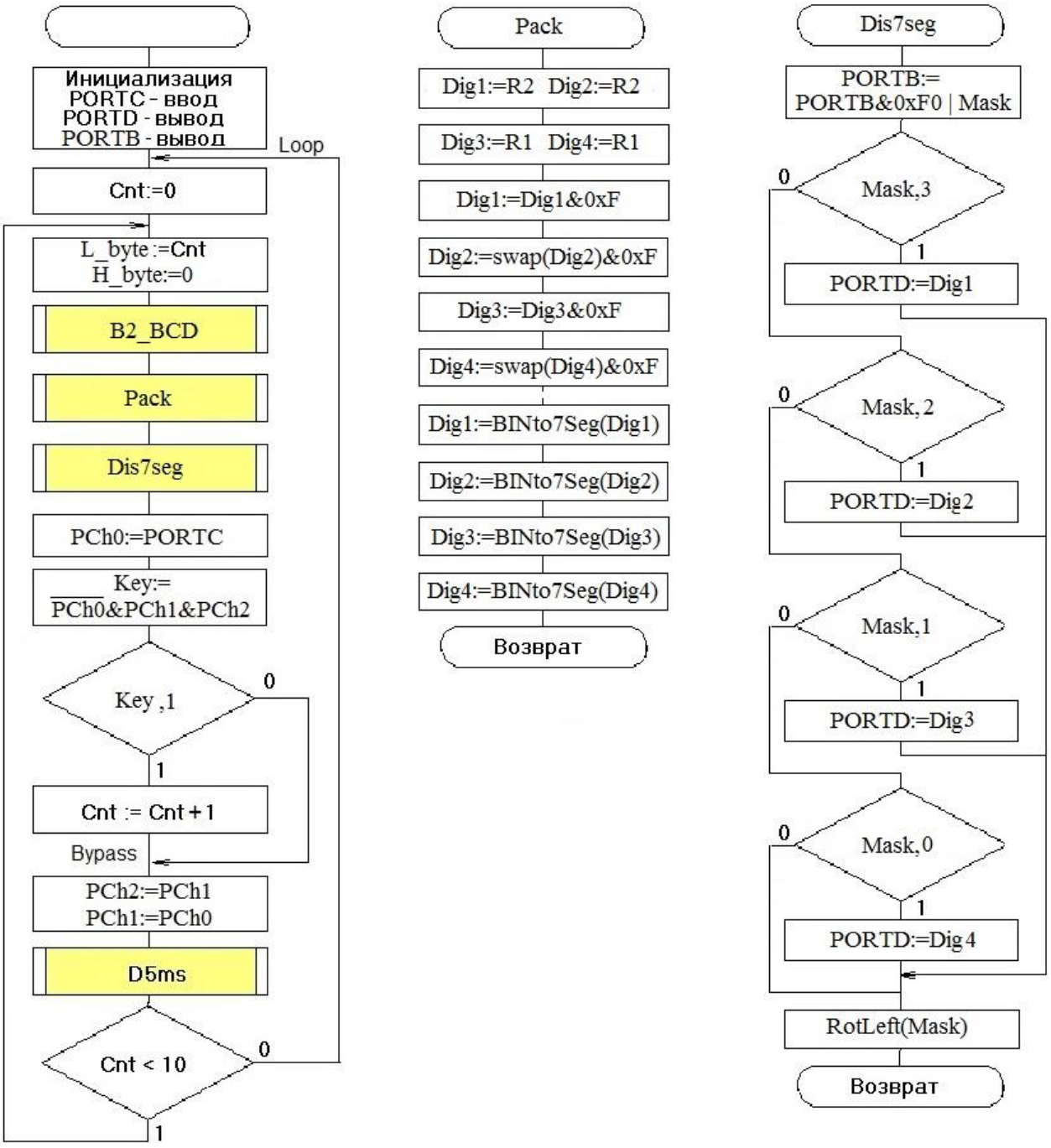

Рисунок 3.2

При кодировании алгоритма следует учесть, что линии PB3 – PB0 порта PORTB необходимо настроить на вывод, остальные на ввод:

```
movlw     b'11110000'     ;3-0 вывод
movwf     TRISB^80h
```

Подпрограмма преобразования в двоично-десятичный код B2_BCD должна использоваться, если информацию необходимо отображать в десятичном формате.

Текст подпрограммы:

```
        B2_BCD  bcf     STATUS,0
                movlw   .16
                movwf   count
                clrf    R0
                clrf    R1
                clrf    R2
loop16          rlf     L_byte,f
                rlf     H_byte,f
                rlf     R2,f
                rlf     R1,f
                rlf     R0,f
                decfsz  count,f
                goto    adjDEC
                retlw   0
adjDEC          movlw   R2
                movwf   FSR
                call    adjBCD
                movlw   R1
                movwf   FSR
                call    adjBCD
                movlw   R0
                movwf   FSR
                call    adjBCD
                goto    loop16

adjBCD          movlw   3
                addwf   0,w
                movwf   temp
                btfsc   temp,3
                movwf   0
                movlw   30
                addwf   0,w
                movwf   temp
                btfsc   temp,7
                movwf   0
                retlw   0
```

Если информация должна отображаться в шестнадцатеричном коде, то значение переменной Cnt необходимо копировать в ячейку R2. Ячейку R1 необходимо очистить. Текст подпрограммы переформатирования данных:

```
Pack        movf    R2,w
            movwf   Dig1
            movwf   Dig2
            swapf   Dig2,f
            movf    R1,w
            movwf   Dig3
            movwf   Dig4
            swapf   Dig4,f
            movlw   b'00001111'
            andwf   Dig4,f
            andwf   Dig3,f
            andwf   Dig2,f
            andwf   Dig1,f
            movf    Dig4,w
            call    BINto7Seg
            movwf   Dig4
            movf    Dig3,w
            call    BINto7Seg
            movwf   Dig3
            movf    Dig2,w
            call    BINto7Seg
            movwf   Dig2
            movf    Dig1,w
            call    BINto7Seg
            movwf   Dig1
            return
```

При реализации подпрограммы BINto7Seg использован метод табличных вычислений. Значение аргумента передается в функцию через аккумулятор и суммируется с младшим байтом программного счетчика PCL. Функция возвращает через аккумулятор значение 7-сегментного кода при выполнении инструкции возврата retlw с соответствующим параметром. Текст подпрограммы преобразования BIN - 7 Seg

```
BINto7Seg   addwf   PCL,f
            retlw   0x3F
            retlw   0x06
            retlw   0x5b
            retlw   0x4f
            retlw   0x66
            retlw   0x6d
            retlw   0x7d
            retlw   0x07
```

```
        retlw   0x7f
        retlw   0x6f
        retlw   0x77
        retlw   0x7e
        retlw   0x39
        retlw   0x5e
        retlw   0x79
        retlw   0x71
```

До основного цикла кроме инициализации портов, таймера и системы прерываний необходимо выполнить инициализацию переменной Mask

```
        clrf    Mask
        incf    Mask,f
```

Текст подпрограммы вывода на индикатор:

```
Dis7seg movlw   0xF0
        andwf   PORTB
        movlw   0x0F
        andwf   Mask,w
        iorwf   PORTB
        btfss   Mask,3
        goto    $+4
        movf    Dig1,w      ;вывести
        movwf   PORTD       ;символ
        goto    Shift
        btfss   Mask,2
        goto    $+4
        movf    Dig2,w      ;вывести
        movwf   PORTD       ;символ
        goto    Shift
        btfss   Mask,1
        goto    $+4
        movf    Dig3,w      ;вывести
        movwf   PORTD       ;символ
        goto    Shift
        btfss   Mask,0
        goto    Shift
        movf    Dig4,w      ;вывести
        movwf   PORTD       ;символ
Shift   bcf     STATUS,C
        rlf     Mask,f
        btfsc   Mask,4
```

```
        swapf   Mask,f
        return
```

4 Аналоговый ввод

4.1 Описание задания

Необходимо разработать и кодировать на языке Ассемблера алгоритм программы аналогового ввода для МК типа PIC16F877A. Нажатие с последующим отпусканием кнопки RC1 должно вызывать инкремент переменной Cnt. Значение трех младших разрядов Cnt должно выводиться на линии RE2 – RE0 порта PORTE. Программа должна выполнять запуск модуля аналого-цифрового преобразователя (АЦП), получать результаты преобразования, обрабатывать и выводить на 7-сегментные индикаторы. Варианты заданий приведены в таблице 4.1. Для вариантов 1 – 3 должен использоваться программный опрос готовности результата аналого-цифрового преобразования, для вариантов 4 – 6 обработка прерывания по запросу модуля АЦП.

Таблица 4.1

Вариант	Задание
1	Информация из канала AN0 в шестнадцатеричной форме (0 .. 3FF)
2	Информация из канала AN0 в шестнадцатеричной форме (0 .. FF)
3	Информация из канала AN0 в десятичной форме (0 .. 1023)
4	Информация из канала AN0 в десятичной форме (0 .. 255)
5	Информация из канала AN0 в десятичной форме (0.000 .. 4.095)
6	Информация из канала AN0 в десятичной форме (-128 .. 127)

4.2 Рекомендации по выполнению

В таблице 4.2 приведены регистры специального назначения, связанные с модулем АЦП. Результат аналого-цифрового преобразования сохраняется в регистрах ADRESH (старший байт) и ADRESL (младший байт).

Таблица 4.2

Адрес	Имя	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0Ch	PIR1	PSPIF*	ADIF	RCIF	TXIF	SSPIF	CCP1F	TMR2IF	TMR1IF
8Ch	PIE1	PSPIE*	ADIE	RCIE	TXIE	SSPIE	CCP1E	TMR2IE	TMR1IE
1Eh	ADRESH	Старший байт результата преобразования							
9Eh	ADRESL	Младший байт результата преобразования							
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/ -DONE	-	ADON
9Fh	ADCON1	ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0
85h	TRISA	-	-	Регистр направления данных PORTA					
05h	PORTA	-	-	Регистр защелки PORTA					
89h*	TRISE	IBF	OBF	IBOV	PSPM	-	Регистр напр. PORTE		
09h*	PORTE	-	-	-	-	-	RE2	RE1	RE0

Разряды регистра управления ADCON0 определяют:

- биты 7-6 ADCS1:ADCS0 - выбор источника тактового сигнала (00 = $F_{OSC}/2$; 01 = $F_{OSC}/8$; 10 = $F_{OSC}/32$; 11 = F_{RC} - внутренний RC генератор модуля АЦП);

- биты 5-3 CHS2:CHS0 - выбор аналогового канала AN0.. AN7;

- бит 2 GO/-DONE - бит статуса модуля АЦП (установка бита вызывает начало преобразования, аппаратно сбрасывается по завершении преобразования);

- бит 1 не используется;

- бит 0 ADON - включение модуля АЦП (1 - модуль включен, 0 - модуль выключен и не потребляет ток).

Разряды регистра управления ADCON1 определяют:

- бит 7 ADFM - формат сохранения 10-разрядного результата (пояснения на рисунке 4.1);

- биты 6-4 не используются**,** читаются как ‘0’;

- биты 3-0 PCFG3:PCFG0 - управляющие биты настройки каналов АЦП и многофункциональных выводов (таблица 4.3).

Когда преобразование завершено, 10-разрядный результат аналого-цифрового преобразования записывается в регистры ADRESH:ADRESL, после чего сбрасывается флаг GO/-DONE (ADCON0,2) и устанавливается флаг прерывания ADIF в регистре PIR1. Сброс флага GO/-DONE в '0' во время преобразования приведет к его прекращению.

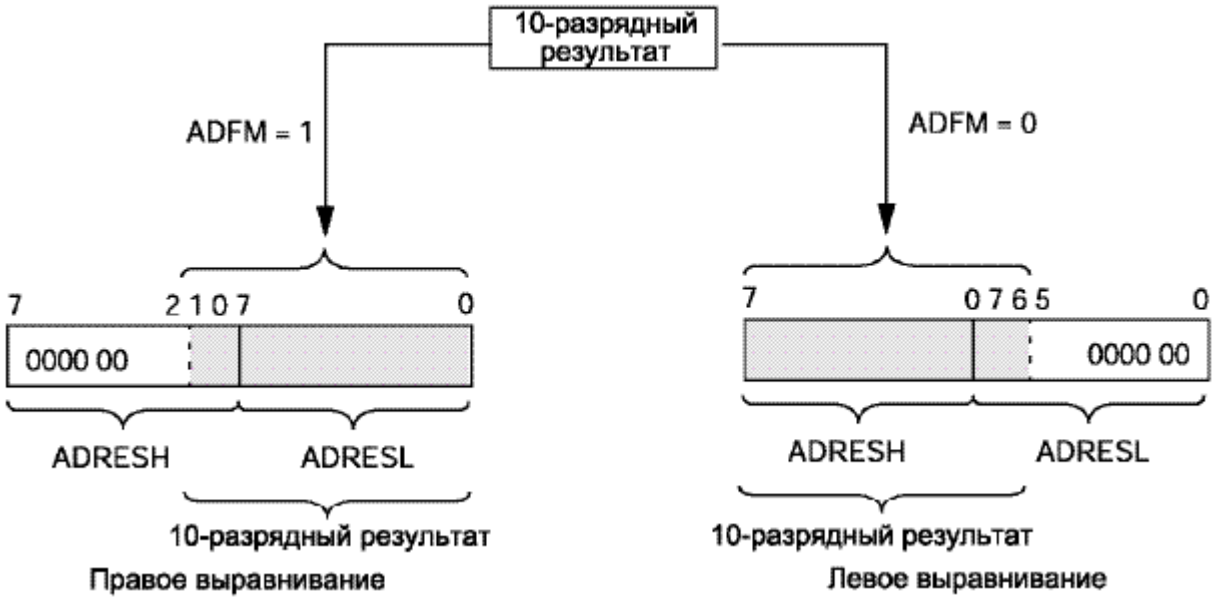

Рисунок 4.1

Таблица 4.3

PCGF3: PCGF0	AN7 RE2	AN6 RE1	AN5 RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-
0000	A	A	A	A	A	A	A	A	VDD	VSS
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS
0010	D	D	D	A	A	A	A	A	VDD	VSS
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS
0100	D	D	D	D	A	D	A	A	VDD	VSS
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS
011x	D	D	D	D	D	D	D	D	VDD	VSS
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2
1001	D	D	A	A	A	A	A	A	VDD	VSS
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2
1110	D	D	D	D	D	D	D	A	VDD	VSS
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2

Время получения одного бита результата определяется параметром T_{AD}. Для 10-разрядного результата требуется как минимум $12T_{AD}$. Для получения корректного результата преобразования необходимо выбрать источник тактового сигнала АЦП, обеспечивающий время T_{AD} не менее 1.6 мкс. В таблице 4.4 указано макси-

мальное значение тактовой частоты микроконтроллера для каждого режима синхронизирующего сигнала АЦП.

Таблица 4.4

Выбор T_{AD}		F_{OSC}, максимум
Режим	ADCS1:ADCS0	
$2T_{OSC}$	00	1.25 МГц
$8T_{OSC}$	01	5 МГц
$32T_{OSC}$	10	20 МГц
RC генератор	11	не более 1 МГц

На рисунке 4.2 приведена схема программы для вариантов с программным опросом флага готовности GO/-DONE и с использованием прерываний при установке флага ADIF.

При инициализации модуля АЦП выполняется загрузка регистров управления ADCON0 и ADCON1. Для вариантов заданий, где требуется разрешение 10 разрядов, следует выбирать правое выравнивание. Для вариантов заданий, где требуется разрешение 8 разрядов, следует выбирать левое выравнивание, при этом два младших разряда не используются. Регистр TRISA должен настраивать линии порта PORTA на вход. При использовании прерываний необходимо разрешить прерывания от ADIF.

При использовании программного опроса готовности результата в основном цикле необходимо:

- активировать аналого-цифровое преобразование - установить бит GO/-DONE;

- ожидать окончания преобразования (пока бит GO/-DONE не будет сброшен);

- считать результат преобразования из регистров ADRESH:ADRESL.

При использовании прерываний в основном цикле необходимо выполнить проверку, что аналого-цифровое преобразование не выполняется. И только в этом случае активировать его - установить бит GO/-DONE. В подпрограмме обработки прерываний проверить флаг ADIF в регистре PIR1. Если ADIF установлен, считать результат преобразования из регистров ADRESH:ADRESL, сбросить флаг ADIF.

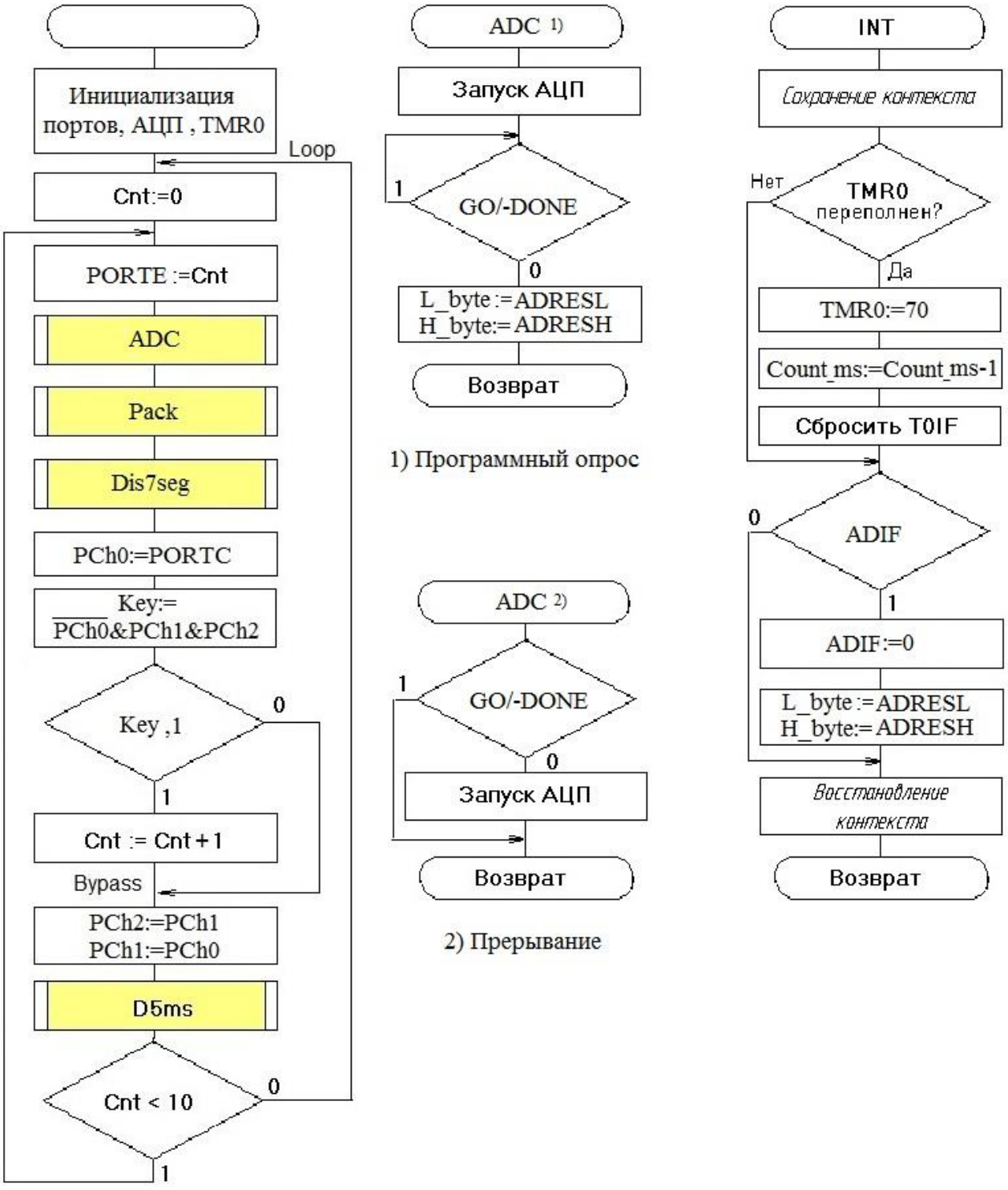
Инициализация портов, АЦП , TMR0
Loop
Cnt:=0
PORTE :=Cnt
ADC
Pack
Dis7seg
PCh0:=PORTC
Key:= PCh0&PCh1&PCh2
Key ,1
0
1
Cnt := Cnt + 1
Bypass
PCh2:=PCh1
PCh1:=PCh0
D5ms
Cnt < 10
0
1
ADC 1)
Запуск АЦП
GO/-DONE
1
0
L_byte :=ADRESL
H_byte:=ADRESH
Возврат
1) Программный опрос
ADC 2)
GO/-DONE
1
0
Запуск АЦП
Возврат
2) Прерывание
INT
Сохранение контекста
Нет
TMR0 переполнен?
Да
TMR0:=70
Count_ms:=Count_ms-1
Сбросить T0IF
ADIF
0
1
ADIF:=0
L_byte :=ADRESL
H_byte:=ADRESH
Восстановление контекста
Возврат

Рисунок 4.2

Список использованных источников

1 PIC16F87X [Электронный ресурс]: Техническое описание. – М.: ООО «Микро-Чип», 2001. – 61 с. – Режим доступа : http://www.microchip.ru/files/d-sheets-rus/pic16f87x.pdf /. – 26.04.2018.

2 MPASM [Электронный ресурс]: Руководство пользователя. – М.: ООО «Микро-Чип», 2001. – 183 с. – Режим доступа: http://www.microchip.ru/files/d-sheets-rus/mpasm.pdf /. – 26.04.2018.

Приложение А
(справочное)

Система команд МК PIC16F87X

Мнемоника		Описание	Циклов	Флаги
ADDWF	f,d	Сложение W и f	1	C,DC,Z
ANDWF	f,d	Побитное 'И' W и f	1	Z
CLRF	f	Очистить f	1	Z
CLRW	-	Очистить W	1	Z
COMF	f,d	Инвертировать f	1	Z
DECF	f,d	Вычесть 1 из f	1	Z
DECFSZ	f,d	Вычесть 1 из f и пропустить если 0	1(2)	
INCF	f,d	Прибавить 1 к f	1	Z
INCFSZ	f,d	Прибавить 1 к f и пропустить если 0	1(2)	
IORWF	f,d	Побитное 'ИЛИ' W и f	1	Z
MOVF	f,d	Переслать f	1	Z
MOVWF	f	Переслать W в f	1	
NOP	-	Нет операции	1	
RLF	f,d	Циклический сдвиг f влево через перенос	1	C
RRF	f,d	Циклический сдвиг f вправо через перенос	1	C
SUBWF	f,d	Вычесть W из f	1	C,DC,Z
SWAPF	f,d	Поменять местами полубайты в регистре f	1	
XORWF	f,d	Побитное 'исключающее ИЛИ' W и f	1	Z
BCF	f,b	Очистить бит b в регистре f	1	
BSF	f,b	Установить бит b в регистре f	1	
BTFSC	f,b	Проверить бит b в f, пропустить если 0	1(2)	
BTFSS	f,b	Проверить бит b в f, пропустить если 1	1(2)	
ADDLW	k	Сложить константу с W	1	C,DC,Z
ANDLW	k	Побитное 'И' константы и W	1	Z
CALL	k	Вызов подпрограммы	2	
CLRWDT	-	Очистить WDT	1	-TO,-PD
GOTO	k	Безусловный переход	2	
IORLW	k	Побитное 'ИЛИ' константы и W	1	Z
MOVLW	k	Переслать константу в W	1	
RETFIE	-	Возврат из подпрограммы с разрешением прерываний	2	
RETLW	k	Возврат из подпрограммы с загрузкой константы в W	2	
RETURN	-	Возврат из подпрограммы	2	
SLEEP	-	Перейти в режим SLEEP	1	-TO,-PD
SUBLW	k	Вычесть W из константы	1	C,DC,Z
XORLW	k	Побитное 'искл. ИЛИ' константы и W	1	Z