МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального 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
государственного образовательного учреждения высшего профессионального
образования «Оренбургский государственный университет»

Оренбург 2009

УДК 621.3 : 681.31
ББК 32.85+32.973
Х 60

Рецензент
доктор технических наук, профессор В.Н. Булатов

Хлуденев, А.В.
Х 60 **Разработка и кодирование алгоритмов для PIC-микро: методические указания / А.В.Хлуденев. – Оренбург: ИПК ГОУ ОГУ, 2009. - 46 с.**

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 студентов, обучающихся по программам высшего профессионального образования по специальности 210106 «Промышленная электроника», а также могут быть использованы студентами других специальностей, связанных с использованием и разработкой микропроцессорных средств управления и обработки информации.

ББК 32.85+32.973

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	Суммирующий двоичный счетчик по модулю 16
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.

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 C:\Program Files\ Microchip\MPASM Suite\Template\. Варианты шаблонов для получения абсолютного кода программы находятся в папке ..\Code. Для МК типа PIC16F877 шаблон находится в файле 16F877TEMP.asm.

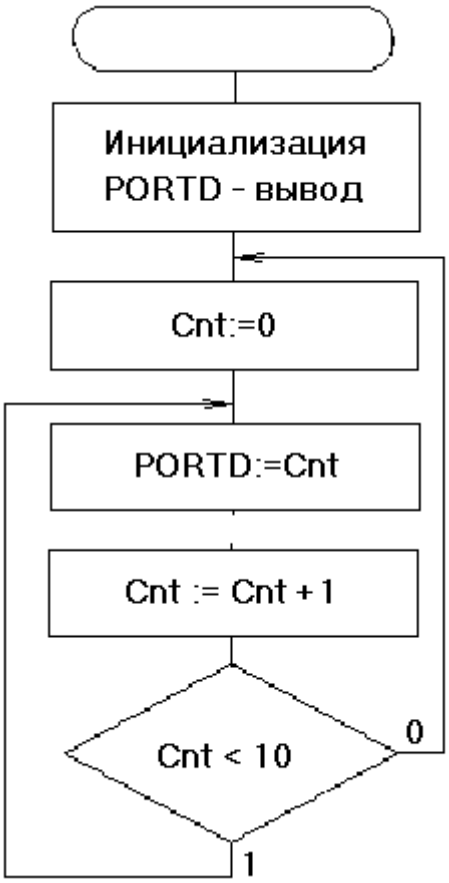


Рисунок 1.1

Данный шаблон с комментариями на русском языке приведен ниже:

```
        list    p=16f877, st = OFF      ; директива листинга
        #include <p16f877.inc>          ; определение микроконтроллера

        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON & _PWRTE_ON
    & _XT_OSC & _WRT_ENABLE_ON & _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
        clrf    PCLATH          ; очистка программного счетчика
        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
        movf    status_temp,w   ; восстановление контекста
        movwf   STATUS          ;
        swapf   w_temp,f
        swapf   w_temp,w        ;
        retfie                  ; возврат из подпрограммы

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

В общем случае текст программы на Ассемблере состоит из следующих разделов:

- директив листинга, определения модели МК;
- директив определения переменных;
- директивы указания вектора старта;
- оператора переход на начало программы;
- директивы указания вектора прерывания;
- операторов подпрограммы обработки прерывания;
- операторов программы;
- директивы окончания программы.

В данном случае механизм прерываний не используется, поэтому из шаблона можно удалить раздел подпрограммы обработки прерывания и команду перехода на операторы основной программы (goto main).

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 (IRP, RP1,RP0) регистра STATUS. После очистки регистра TRISD необходимо вернуть исходное состояние измененных разрядов (RP0) регистра STATUS.

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 нуле-

вого 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 loop, иначе переход на метку inc.

Сформированный исходный текст приведен ниже

```
        list    p=16f877, st = OFF      ; директива листинга
        #include <p16f877.inc>          ; определение микроконтроллера

        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON & _PWRTE_ON
    & _XT_OSC & _WRT_ENABLE_ON & _LVP_OFF & _DEBUG_ON &
    _CPD_OFF        ; директива установки разрядов слова конфигурации

;***** Определение переменных
Cnt     EQU     0x20            ; счетчик
;*****************************************************************
        ORG     0x000           ; вектор старта
        clrf    PCLATH          ; очистка программного счетчика

        bsf     STATUS,RP0      ; банк 1 (адреса 80h - FFh)
        bcf     STATUS,IRP
        bcf     STATUS,RP1
        clrf    TRISD^80        ; все линии PORTD на вывод
        bcf     STATUS,RP0      ; 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 Разработка программ сложной структуры

2.1 Описание задания

Необходимо разработать и кодировать на языке Ассемблера МК PIC-micro алгоритм программы вычисления арифметического выражения в соответствии с вариантом задания по таблице 2.1.

Таблица 2.1

Вариант	Задание
1	18*26 + 34*78
2	56*24 + 37*98 +1000
3	81*62 - 43*87
4	65*42 + 73*89 - 100
5	174*78 - 85*103
6	38*55 - 237*5 -132

2.2 Рекомендации по выполнению

Часто при реализации алгоритма приходится выполнять одни и те же последовательности операторов (например, при выполнении типовых арифметических вычислении). С целью сокращения объема кода программы рационально последовательно выполнять один и тот же участок кода, вместо того, чтобы дублировать в программе этот участок кода несколько раз.

Участок программы, к которому можно обращаться из различных мест программы для выполнения некоторых действий называется ***подпрограммой***. Вызов подпрограммы выполняется по команде ***call*** , возврат из подпрограммы - по команде ***return***. Описанную ситуацию иллюстрирует рисунок 2.1.

Программы, содержащие подпрограммы, имеют сложную структуру. Ассемблер MPASM / 2 / может транслировать такие программы в двух режимах:

- для генерации абсолютного кода, который может быть загружен непосредственно в память программ микроконтроллера;
- для генерации объектных кодов, которые необходимо связывать с другими объектными модулями при помощи редактора связей MPLINK.

По умолчанию MPASM работает в режиме генерации абсолютного кода (рисунок 2.2). В этом случае текст всей программы, включая подпрограммы, должен храниться в одном файле. При трансляции исходного файла в этом режиме, все значения базовых адресов используемых ячеек должны быть явно указаны как параметры директив:

- ORG для памяти программ;
- EQU или CBLOCK для памяти данных.

Если компиляция выполнена без ошибок, то будет создан HEX файл кода программы, который можно использовать для непосредственного программирования микроконтроллера.

Рисунок 2.1

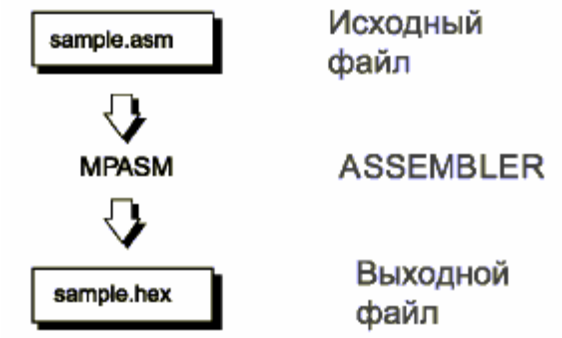


Рисунок 2.2

Рассмотрим формирование исходного текста программы сложной структуры на примере задачи вычисления суммы произведений чисел 53*45 + 13*11. В системе команд МК семейств PIC16 нет команды умножения, поэтому приходится реализовать эту операцию программно. На рисунке 2.3 приведена схема подпрограммы умножения восьмиразрядных чисел методом поразрядной обработки разрядов множителя. Сначала обнуляются ячейки, в которых формируется шестнадцатиразрядное произведение (H_byte, L_byte), а также в счетчик циклов Count загружается необходимое число шагов. В теле циклического алгоритма, пока Count > 0, анализируется текущее значение младшего разряда множителя Mulplr. Если он равен 1, то к старшему байту произведения H_byte прибавляется множимое Mulcnd. Затем выполняется сдвиг вправо текущего шестнадцатиразрядного произведения (H_byte, L_byte), сдвиг вправо множителя Mulplr и декремент счетчика циклов Count.

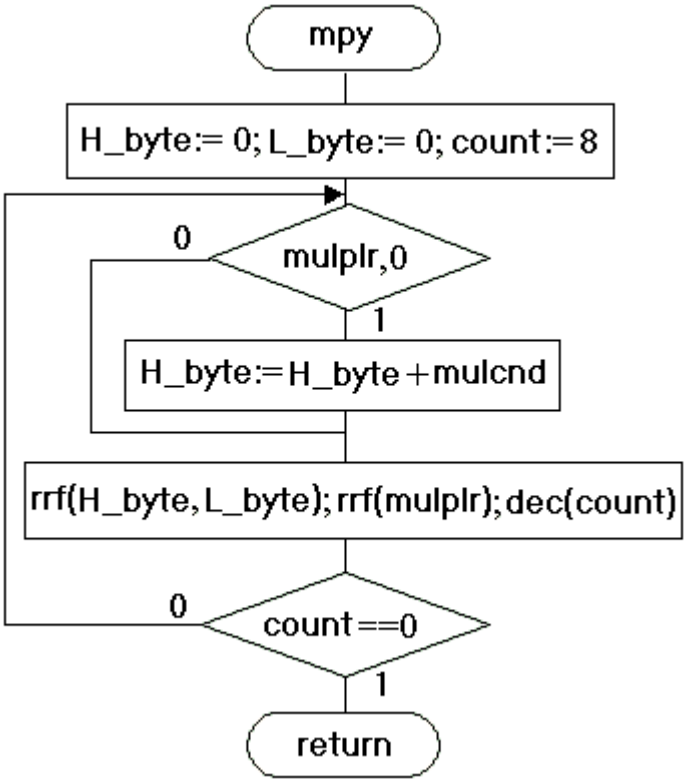


Рисунок 2.3

Исходный текст варианта программы для прямой трансляции в абсолютный код приведен ниже:

```
            LIST    p=16F877, st=OFF
            #include "P16F877.INC"
            CBLOCK 0x20     ; Структура данных
            Mulcnd          ; множимое
            Mulplr          ; множитель
            H_byte          ; старший байт произведения
            L_byte          ; младший байт произведения
            Count           ; счетчик циклов
            H_sum           ; старший байт суммы
            L_sum           ; младший байт суммы
            ENDC

            ORG     0x000       ;вектор старта
            goto    Start
;************* Подпрограмма умножения ************
Mpy         clrf    H_byte
            clrf    L_byte
            movlw   8
            movwf   Count
            movf    Mulcnd,w
            bcf     STATUS,C
```

```
Loop    rrf     Mulplr,f
        btfsc   STATUS,C
        addwf   H_byte,f
        rrf     H_byte,f
        rrf     L_byte,f
        decfsz  Count,f
        goto    Loop
        retlw   0
;***********************************************
Start   clrw
Main    clrf    H_sum           ;очистить сумму
        clrf    L_sum
        movlw   0x35
        movwf   Mulplr          ;35h умножить на 2Dh
        movlw   0x2D
        movwf   Mulcnd
        call    Mpy             ;вызов подпрограммы
        movf    H_byte,w
        movwf   H_sum
        movf    L_byte,w
        movwf   L_sum
        movlw   0x0D
        movwf   Mulplr          ;0Dh умножить на 0Bh
        movlw   0x0B
        movwf   Mulcnd
        call    Mpy             ;вызов подпрограммы
        movf    L_byte,w
        bcf     STATUS,C
        addwf   L_sum,f         ;сложить мл.байты
        movf    H_byte,w
        addwf   H_sum,f         ;сложить ст.байты
        goto    Main
        END
```

MPASM также может генерировать объектные модули, которые могут быть связаны друг с другом с использованием линкера MPLINK для окончательного формирования исполняемого (абсолютного) кода (рисунок 2.4). Данный подход позволяет многократно использовать отлаженные модули программы. Объектные файлы могут быть сгруппированы в библиотечные файлы с помощью программы MPLIB. Библиотеки могут указываться в качестве параметра во время редактирования связей (линковки) и, таким образом, в исполняемый код будут включены только необходимые процедуры.

Написание исходного текста программы, который будет транслироваться в объектный файл, несколько отличается от создания программы с непосред-

ственной трансляцией в HEX файл. Подпрограммы, разработанные для трансляции непосредственно в HEX файл, потребуют незначительных изменений для получения корректного перемещаемого объектного модуля.

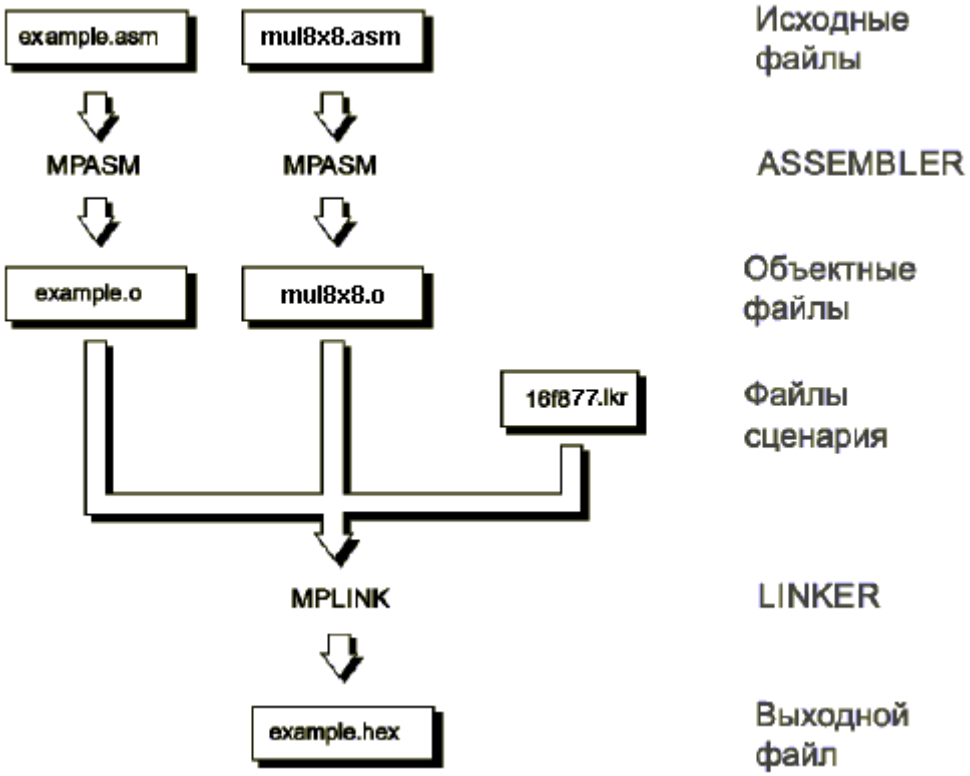


Рисунок 2.4

При трансляции исходных файлов в этом режиме базовые адреса используемых ячеек задаются в параметрах директив / 2 /:

- CODE для памяти программ;
- UDATA (или UDATA_OVR, UDATA_SHR, IDATA) для памяти данных.

Фирма Microchip разработала стандартные файлы сценария (например, 16f877.inc), в которых приведена информация о распределении адресных пространств программ и данных для конкретной модели микроконтроллера.

Операторам программы должна предшествовать директива CODE, определяющая секцию перемещаемого кода.

```
PROG1   CODE            ;сегмент PROG1 (0x0000 - 0x07FF)
        nop
        goto    Start
……….
PROG1   CODE            ;сегмент PROG1
Start   clrw
```

Физический адрес кода в памяти программ может быть установлен, указав необязательный параметр директивы

CODE <ROM address>.

Указание физического адреса может быть необходимо, например, в случаях определение вектора прерывания.

Распределение ячеек оперативной памяти должно быть выполнено в секции данных. Существует пять типов секций данных, в данном задании используются UDATA – неинициализированные данные. Это наиболее общий тип размещения данных. Ячейки, зарезервированные в этой секции, не инициализируются, а обращение к данным выполняется только с использованием меток или косвенной адресацией. Адрес инициализации данных в памяти можно установить, указав необязательный параметр <RAM address>.

Метки, которые определены в одном модуле и используются в других объектных модулях, должны быть отмечены директивой GLOBAL после их объявления. Модули, использующие эти метки, должны объявить их директивой EXTERN. Исходный текст варианта программы для трансляции в относительный код приведен ниже.

Подключаемый модуль (подпрограмма умножения):

```
            LIST    p=16F877, st = OFF
            #include "P16F877.INC"
            UDATA           ; Структура данных
Mulcnd      RES     1       ; множимое
Mulplr      RES     1       ; множитель
H_byte      RES     1       ; старший байт произведения
L_byte      RES     1       ; младший байт произведения
Count       RES     1       ; счетчик циклов

            GLOBAL  mulcnd, mulplr, H_byte, L_byte

PROG1       CODE            ; сегмент кода PROG1 (0x0000 - 0x07FF)
Mpy         GLOBAL  mpy
            clrf    H_byte
            clrf    L_byte
            movlw   8
            movwf   Count
            movf    Mulcnd, W
            bcf     STATUS,C  ; Очистка флага C.
Loop        rrf     Mulplr,f
            btfsc   STATUS,C
            addwf   H_byte,f
            rrf     H_byte,f
            rrf     L_byte,f
            decfsz  Count,f
            goto    Loop
            retlw   0
            END
```

Основной модуль

```
        LIST    p=16F877, st = OFF
        #include "P16F877.INC"
        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON &
  _PWRTE_ON & _XT_OSC & _WRT_ENABLE_ON & _LVP_OFF &
  _DEBUG_ON & _CPD_OFF

        UDATA                   ; Структура данных
H_sum   RES   1                 ; Старший байт суммы
L_sum   RES   1                 ; Младший байт суммы

        EXTERN  Mulcnd, Mulplr, H_byte, L_byte
        EXTERN    Mpy

PROG1   CODE                    ;сегмент PROG1
        nop
        start       clrw
Main    clrf        H_sum       ;очистить сумму
        clrf        L_sum
        movlw       0xFF
        movwf       Mulplr      ;35h умножить на 2Dh
        movlw       0x01
        movwf       Mulcnd
        call        Mpy         ;вызов подпрограммы
        movf        H_byte,w
        movwf       H_sum
        movf        L_byte,w
        movwf       L_sum
        movlw       0xFF
        movwf       Mulplr      ;0Dh умножить на 0Bh
        movlw       0xFF
        movwf       Mulcnd
        call        Mpy                 ;вызов подпрограммы
        movf        L_byte,w
        addwfL_     Sum,f       ;сложить мл.байты
        btfsc       STATUS,C
        incf        H_sum,f
        movf        H_byte,w
        addwf       H_sum,f     ;сложить ст.байты
        goto        Main
        END
```

3 Формирование интервалов времени

3.1 Описание задания

Необходимо разработать и кодировать на языке Ассемблера алгоритм программы для МК типа PIC16F877, которая управляет светодиодом, подключенным к линии RD0 порта PORTD. Время включенного и выключенного состояния светодиода формируется в соответствии с вариантом задания (таблица 3.1).

Факультативное задание – автомат, управляющий дорожным светофором (красный, зеленый свет – 20 секунд, желтый свет - 2 секунды).

Таблица 3.1

Вариант	Задание
1	Время включенного состояния нарастает от 0.5 до 60 секунд с шагом 0.5 секунды, время выключенного состояния – 1 секунда.
2	Время включенного состояния нарастает от 1 до 60 секунд с шагом 1 секунда, время выключенного состояния – 2 секунды.
3	Время включенного состояния спадает от 100 до 2 секунд с шагом 2 секунды, время выключенного состояния – 2 секунды.
4	Время включенного состояния спадает от 100 до 5 секунд с шагом 5 секунд, время выключенного состояния – 1 секунда.
5	Время включенного состояния нарастает от 1 до 60 секунд с шагом 1 секунда, время выключенного состояния спадает от 60 до 1 секунды с шагом 1 секунда.
6	Время включенного состояния спадает от 50 до 2 секунд с шагом 2 секунды, время выключенного состояния нарастает от 2 до 50 секунд с шагом 2 секунды.

3.1 Рекомендации по выполнению

Одной из наиболее часто встречающихся функций микропроцессорных устройств (МПУ) является отсчет интервалов времени.

В учебно-отладочном стенде используется кварцевая стабилизация частоты тактового генератора, и она составляет 3.6864 МГц. Длительность машинного цикла МК составляет 4 такта, поэтому тактирование встроенных периферийных модулей происходит с частотой 3.6864 / 4 = 0.9216 МГц. Время выполнения одной команды МК также составляет 4 такта.

Функцию формирования интервалов времени можно реализовать в следующих вариантах:

- программный;
- аппаратный;
- программно-аппаратный.

Программный способ формирования интервала времени основан на выполнении процессорным ядром некоторой фиксированной последовательности

команд. При этом общее время их выполнения равно формируемому интервалу времени. Главным недостатком этого способа является загрузка процессорного ядра. Поэтому программный способ используют в случаях, когда формируемый интервал времени не слишком длительный, и процессорное ядро на этом интервале времени свободно от выполнения других функций.

Аппаратный способ формирования интервала времени реализуется на основе специализированных периферийных модулей МК – таймеров. Процессорное ядро в течение формируемого интервала времени свободно для выполнения других функций, если используется механизм прерывания по переполнению таймера. Недостатком этого способа является сложность формирования длительных интервалов времени на таймерах с небольшим числом разрядов. В какой-то степени эту проблему удается решить путем понижения частоты счетных импульсов.

Программно-аппаратный способ позволяет формировать длительные интервалы времени путем многократного программного перезапуска таймера. При этом средняя загрузка процессорного ядра в течение формируемого интервала времени получается небольшой, если исключить программный опрос флага переполнения таймера.

В составе МК семейства PIC16F87x имеются три таймера [1]:

- 8-разрядный таймер/счетчик TMR0 с 8-разрядным программируемым предделителем;
- 16-разрядный таймер/счетчик TMR1 с 8-разрядным программируемым предделителем и возможностью подключения внешнего резонатора;
- 8-разрядный таймер/счетчик TMR2 с программируемым предделителем и выходным делителем (по 4 разряда).

Оценим возможность использования таймера TMR0. Функциональная схема таймера приведена на рисунке 3.1.

Режим работы таймера определяется состоянием разрядов регистра OPTION_REG (таблица 3.2).

Таблица 3.2

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

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. Максимальное значение коэффициента деления частоты 1:256 получается при <PS2:PS0> = 111. Для настройки TMR0 на данный режим в регистр OPTION_REG необходимо загрузить управляющее слово b'10000111'.

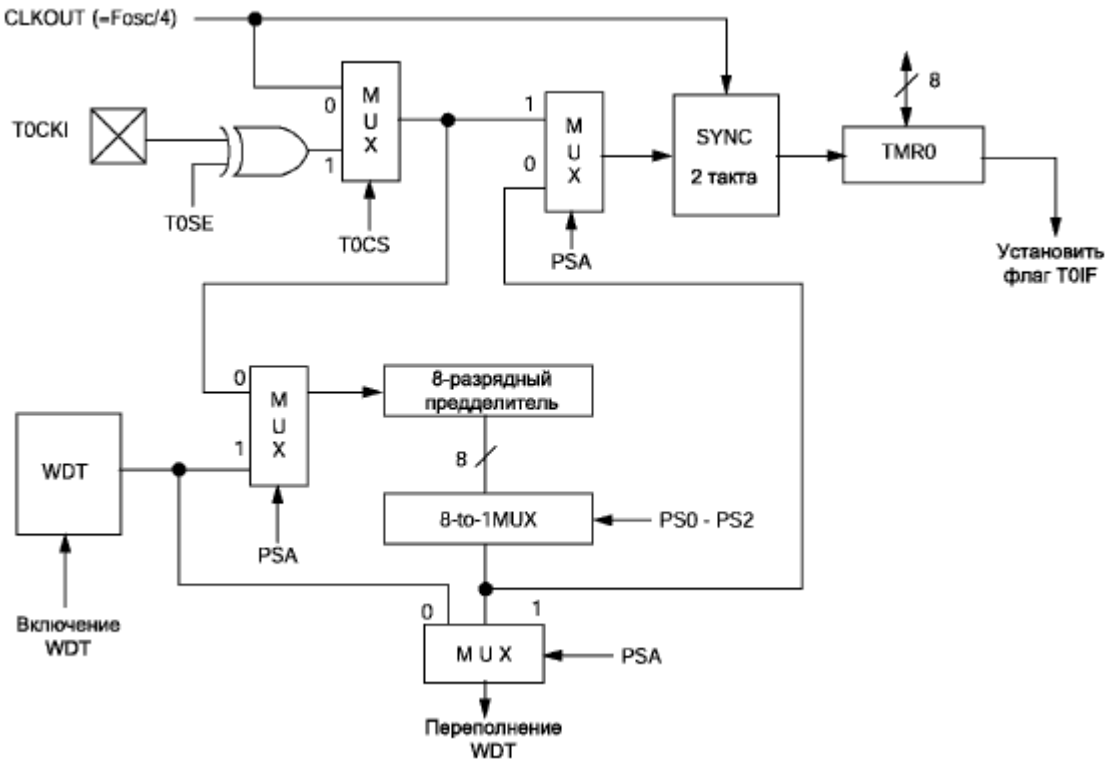


Рисунок 3.1

При переполнении таймера устанавливается флаг T0IF (регистр INTCON). После анализа флага, его необходимо программно сбросить.

При тактовой частоте МК F=3.6864 МГц интервал времени между двумя переполнениями таймера составит:

$$T_{TMR} = 4 \cdot 2^8 \cdot 2^8 / F = 4 \cdot 256 \cdot 256 / 3.6864 \cdot 10^6 = 71.11 \text{ мс}.$$

Таким образом, при заданном значении тактовой частоты 3.6864 МГц аппаратный вариант на основе таймера TMR0 позволяет формировать интервалы не более 71.11 мс.

Для формирования интервала 0.5 секунды требуемое число переполнений составит

$$500 / 71.11 \approx 7.$$

Погрешность формирования интервала 0.5 секунды не превысит

$$[(500 - 71.11 \cdot 7) / 500] \cdot 100\ \% = 0.446\ \%.$$

Последовательно инициируя от 1 до 255 серий из 7 переполнений таймера TMR0, можно формировать интервал времени в диапазоне от 0.5 до 127.5 секунды с шагом 0.5 секунды.

Для подсчета количества переполнений таймера введем переменную Count. Для подсчета половин секунд используем переменную Count05s.

На рисунке 3.2 приведена схема программы, которая управляет светодиодом, подключенным к линии RD0 порта PORTD. Время включенного и выключенного состояния светодиода определяется значением целочисленного параметра *n*, который передается в подпрограмму формирования временного интервала DLY.

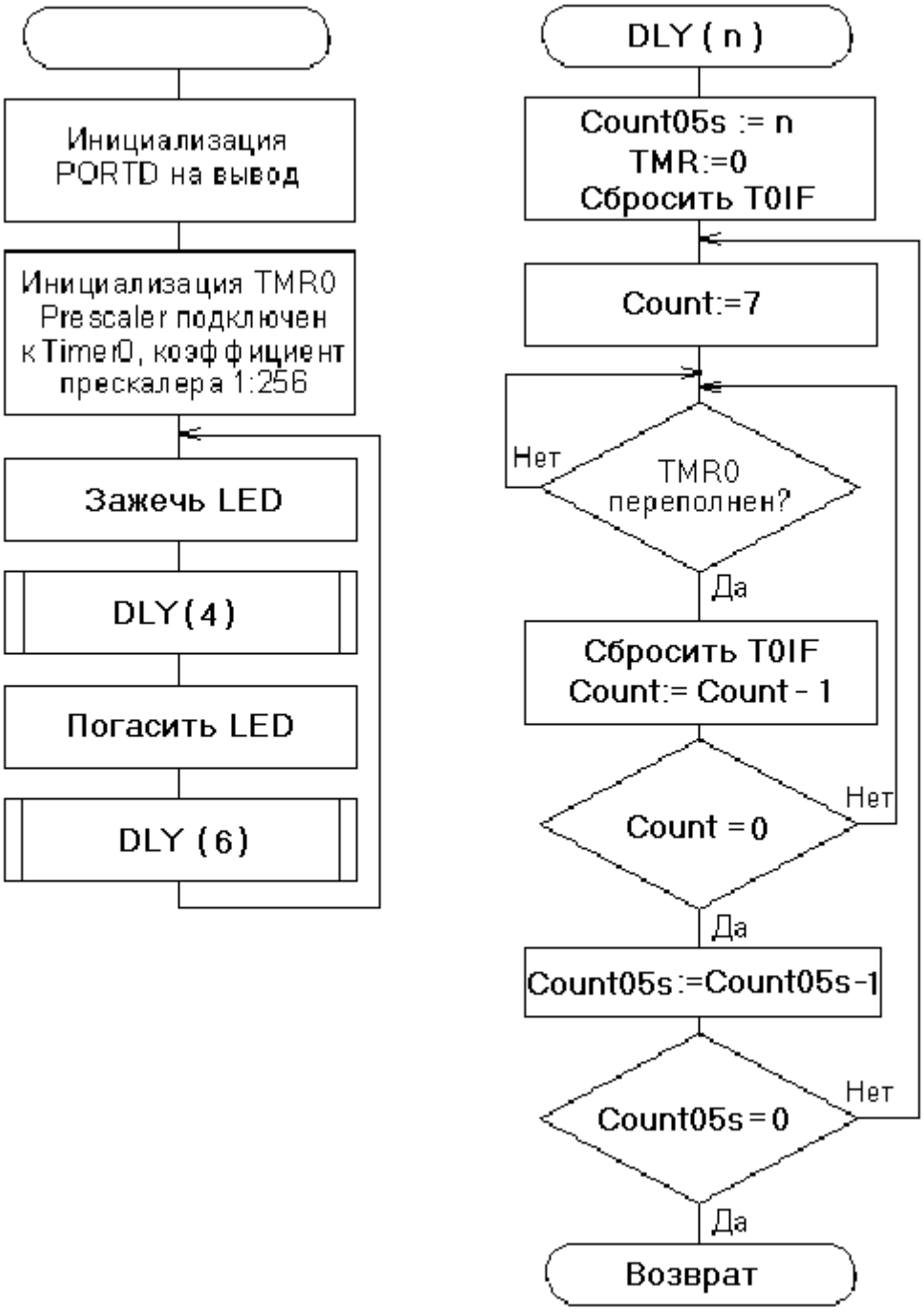


Рисунок 3.2

Ниже приводится исходный текст программы. Значения интервалов времени (в единицах 0.5 секунды), в течение которых светодиод выключен и включен, задаются значениями констант Delay0 и Delay1 соответственно.

```
        list    p=16f877, st = OFF
        include "p16f877.inc"
;---------------------- Константы --------------------------
Delay0      EQU         .4                  ;задержка в единицах 0.5 с
Delay1      EQU         .6                  ;задержка в единицах 0.5 с
;---------------------- Переменные -------------------------
            CBLOCK      H'020'
            Count                           ;счетчик переполнения таймера
            Count05s                        ;счетчик половин секунд
            ENDC
            #define Led  PORTD,0            ;выходной управляющий сигнал
;----------------- Начало кода программы --------------------
            ORG         0x000               ;вектор сброса
            nop                             ;необходимо для MICD
            clrf        PORTD               ;инициализация модулей МК
            bsf         STATUS,RP0          ;банк 1 (адреса 80h - FFh)
            bcf         STATUS,IRP
            bcf         STATUS,RP1
            movlw       b'10000111'         ;режимы TMR0
            movwf       OPTION_REG^80
            clrf        TRISD^80            ;все линии PORTD на вывод
            bcf         STATUS,RP0          ;банк 0 (адреса 00h - 7Fh)
Main        bsf         Led                 ;зажечь светодиод
            movlw       Delay1
            call        Dly
            bcf         Led                 ;погасить светодиод
            movlw       Delay0
            call        Dly
            goto        main
;------------------ Подпрограмма задержки ---------------------
Dly         movwf       Count05s            ;загрузить счетчик половин секунд
            clrf        TMR0                ;инициализация TMR0
            bcf         INTCON,T0IF
T05s        movlw       .500000/.256/.256
            movwf       Count               ;загрузить счетчик переполнений
T71ms       btfss       INTCON,T0IF
            goto        T71ms               ;ожидание переполнения таймера
            bcf         INTCON,T0IF
            decfsz      Count,f             ;полсекунды прошло?
            goto        T71ms
            decfsz      Count05s,f          ;время закончилось?
            goto        T05s
            return                          ;возврат в основную программу
            END
```

4 Дискретный ввод-вывод

4.1 Описание задания

Необходимо разработать и кодировать на языке Ассемблера алгоритм программы дискретного ввода-вывода для МК типа PIC16F877.

Нажатие с последующим отпусканием кнопок RB1 и RB2 должно вызывать разнонаправленную модификацию содержимого ячейки Cnt. Содержимое ячейки Cnt должно отображаться на светоизлучающих диодах, подключенных к линиям порта PORTD. Активному (нажатому) состоянию кнопки соответствует низкий потенциал на выводе порта PORTB. Пассивному (отжатому) состоянию кнопки соответствует высокий потенциал, который обеспечивается встроенными подтягивающими резисторами. Варианты заданий приведены в таблице 4.1.

Таблица 4.1

Вариант	Задание
1	Инкремент и декремент с ограничением в диапазоне 0 .. 255
2	Циклический инкремент и декремент по модулю 10
3	Циклическое вращение единицы
4	Вращение единицы с ограничением
5	Циклический инкремент и декремент в коде Джонсона
6	Инкремент и декремент в коде Джонсона с ограничением

4.2 Рекомендации по выполнению

Функции программы:

- опрос кнопок, подключенных к выводам порта PORTB (RB1 и RB2);
- модификация содержимого ячейки Cnt;
- вывод содержимого ячейки Cnt в порт PORTD.

На рисунке 4.1 приведен пример - схема программы, реализующая алгоритм реверсивного счетчика числа нажатий на кнопки RB1 и RB2 (по модулю 256). Нажатие с последующим отпусканием кнопки RB1 вызывает декремент содержимого ячейки Cnt, нажатие с последующим отпусканием кнопки RB2 вызывает инкремент ее содержимого.

При старте программа выполняет инициализацию портов ввода и вывода, инициализацию ячейки счетчика Cnt. После чего в теле цикла программы анализируются состояния кнопок RB1 и RB2 и, в случае их нажатия, изменяется содержимое ячейки Cnt.

Все механические кнопки имеют одно негативное свойство, известное как «дребезг» контактов, которое обусловлено колебаниями упругих контактов при их замыкании и размыкании. Длительность колебаний составляет всего 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 входной линии порта. Для надежного считывания состояний кнопки величина этих интервалов не должна быть меньше 15 – 30 мс. Для подавления «дребезга» контактов в цикл основной программы включены операторы вызова подпрограммы задержки D15ms.

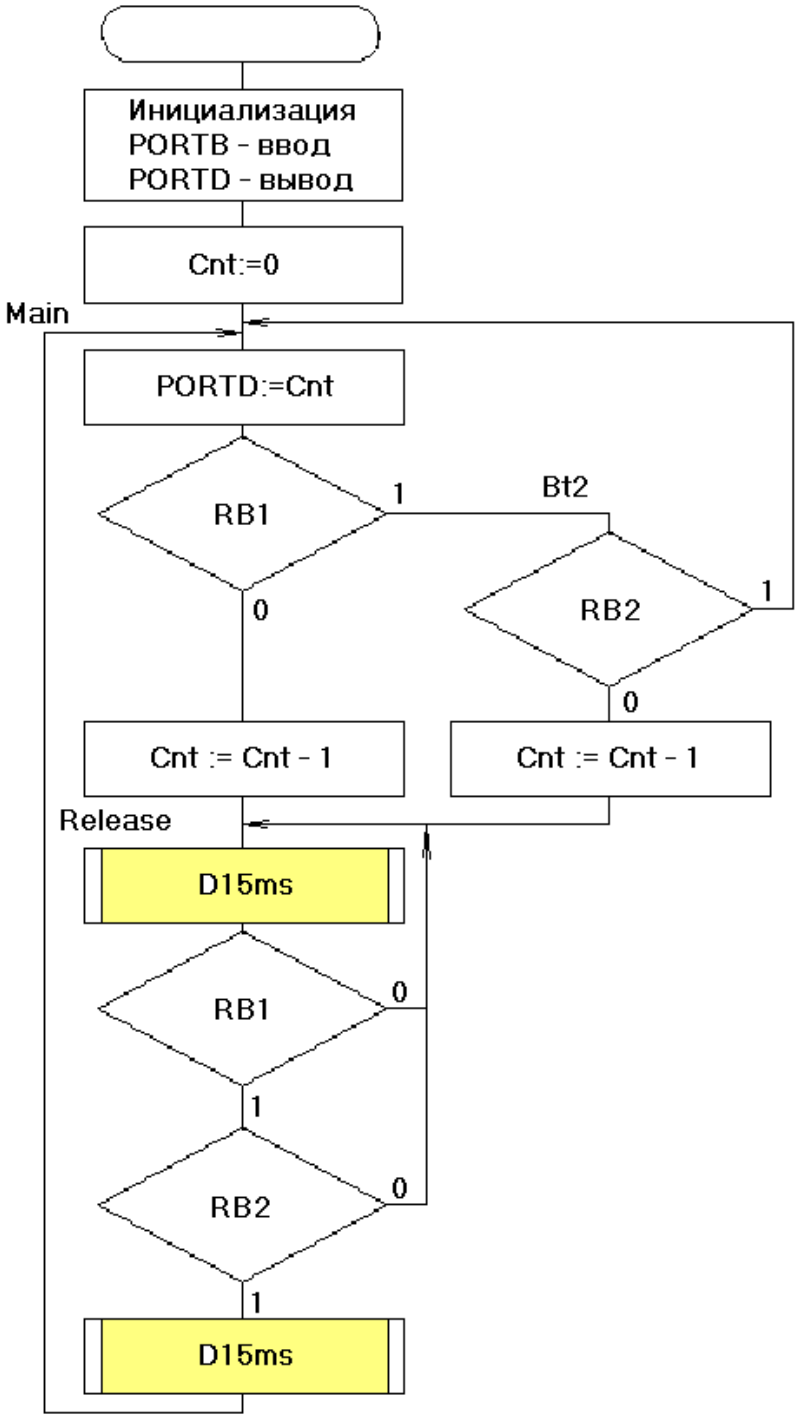


Рисунок 4.1

В таблицах 4.1 и 4.2 приведены регистры специального назначения, связанные с портами PORTB и PORTD. Для настройки линий PORTB на ввод в регистр TRISB необходимо загрузить управляющее слово b'11111111'. Для вклю-

чения подтягивающих резисторов к линиям PORTB в разряд 7 регистра OPTION_REG необходимо загрузить 0. Для настройки линий PORTD на вывод в регистр TRISD необходимо загрузить управляющее слово b'00000000'.

Таблица 4.1

Адрес	Имя	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
86h, 186h	TRISB	Регистр направления данных PORTB							
81h, 181h	OPTION_REG	-RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

Таблица 4.2

Адрес	Имя	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
88h	TRISD	Регистр направления данных PORTD							
89h	TRISE	IBF	OBF	IBOV	PSPMODE	-	Рег. напр. данных PORTE		

Ниже приводится исходный текст программы реверсивного счетчика по модулю 256.

```
            #include    p16f877.inc
            LIST    p=16F877, st=OFF
            __CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON &
      _PWRTE_ON & _XT_OSC & _WRT_ENABLE_ON & _LVP_OFF &
      _DEBUG_ON & _CPD_OFF
;---------------------- Переменные -------------------------
            CBLOCK  0x20
            Cnt                         ;счетчик
            Temp1
            Temp2
            ENDC
;---------------------- Константы --------------------------
Init_PB     EQU     b'11111111'         ;все ввод
Init_PD     EQU     b'00000000'         ;все вывод
Init_OPT    EQU     b'00000000'         ;pull-up R
;------------------ Битовые переменные ---------------------
#define     RB1     PORTB,1             ;кнопка "-"
#define     RB2     PORTB,2             ;кнопка "+"
;------------------ Начало кода программы -------------------
            ORG     0x000               ;вектор сброса
            nop                         ;необходимо для MICD
            clrf    PORTD               ;инициализация модулей МК
            clrf    Cnt
            bsf     STATUS,RP0          ;bank1
```

```
            bcf     STATUS,IRP
            bcf     STATUS,RP1
            movlw   Init_PB
            movwf   TRISB^80h
            movlw   Init_PD
            movwf   TRISD^80h
            movlw   Init_OPT
            movwf   OPTION_REG^80h
            bcf     STATUS,RP0      ;bank0
Main        movf    Cnt,W           ;основной цикл
            movwf   PORTD           ;вывод на линейный индикатор
            btfsc   RB1             ;опрос кнопки "-"
            goto    Bt2             ;кнопка "-" не нажата
            decf    Cnt,1           ;декремент счетчика
            goto    Release
Bt2         btfsc   RB2
            goto    Main            ;кнопка "+" не нажата
            incf    Cnt,1           ;инкремент счетчика
Release                             ;опрос кнопок (отжатие)
            call    D15ms
            btfss   RB1             ;опрос кнопки "-"
            goto    Release         ;кнопка не отжата
            btfss   RB2             ;опрос кнопки "+"
            goto    Release         ;кнопка не отжата
            call    D15ms           ;задержка
            goto    Main            ;кнопка отжата
;------------------ Подпрограмма задержки ---------------------
D15ms                               ;подпрограмма задержки 15 мс
            movlw   .30
            movwf   Temp2
_15ms       call    D500us
            decfsz  Temp2,1
            goto    _15ms
            return
;-------------------------------------------------------------
D500us                              ;подпрограмма задержки 500 мкс
            movlw       .151
            movwf   Temp1
_500us      decfsz  Temp1,1
            goto    _500us
            return
            END
```

5 Аналоговый ввод

5.1 Описание задания

Необходимо разработать и кодировать на языке Ассемблера алгоритм программы аналогового ввода для МК типа PIC16F877. Программа должна выполнять запуск модуля АЦП, получать результаты преобразования, обрабатывать и выводить на индикаторы. Варианты заданий приведены в таблице 5.1.

Таблица 5.1

Вариант	Задание
1	Среднее арифметическое из каналов – AN1 и AN3 в PORTD
2	Разность из каналов – AN0 и AN1 в PORTD, заем в RC5
3	Разность из каналов – AN1 и AN3 в PORTD, заем в RC5
4	Меньшее значение из каналов – AN0 и AN1 в PORTD
5	Большее значение из каналов – AN0 и AN1 в PORTD
6	Меньшее значение из каналов – AN1 и AN3 в PORTD

5.2 Рекомендации по выполнению

В таблице 5.2 приведены регистры специального назначения, связанные с модулем АЦП.

Таблица 5.2

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

Результат аналого-цифрового преобразования сохраняется в регистрах ADRESH (старший байт) и ADRESL (младший байт);

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 5.1);

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

- биты 3-0 PCFG3:PCFG0 - управляющие биты настройки каналов АЦП и многофункциональных выводов (см. таблицу 5.3).

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 бита GO/-DONE в '0' во время преобразования приведет к его прекращению.

Таблица 5.3

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

После включения и конфигурации АЦП выбирается рабочий аналоговый канал. Соответствующие биты TRIS аналоговых каналов должны настраивать линию порта на вход. Рекомендованная последовательность действий для работы с АЦП:

1) настроить и включить модуль АЦП;

2) выдержать паузу, необходимую для зарядки конденсатора C_{HOLD} , не менее 20 мкс [1];

3) начать аналого-цифровое преобразование - установить бит GO/-DONE;

4) ожидать окончания преобразования (пока бит GO/-DONE не будет сброшен или флаг ADIF в регистре PIR1 не будет установлен);

5) считать результат преобразования из регистров ADRESH:ADRESL;

6) для следующего преобразования необходимо выполнить шаги, начиная с пункта 1 или 2.

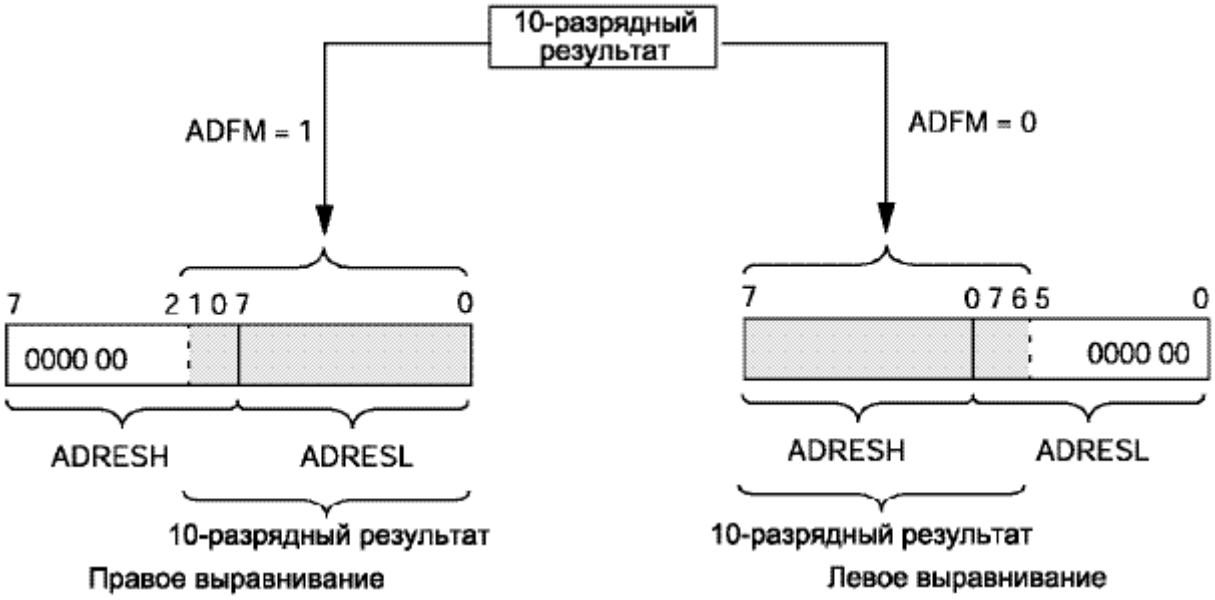


Рисунок 5.1

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 5.4 указано максимальное значение тактовой частоты микроконтроллера для каждого режима синхронизирующего сигнала АЦП.

Таблица 5.4

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

Чтобы обеспечить требуемые значения паузы, необходимой для зарядки конденсатора C_{HOLD} , а также периода дискретизации аналогового сигнала, в цикле программы между операторами выбора канала и запуска преобразования необходимо выполнять программные задержки.

На рисунке 5.2 приведена схема программы, которая после инициализации модуля АЦП циклически выполняет:

- последовательный запуск модуля АЦП для выполнения преобразования напряжений из каналов AN0 и AN1;

- вычисляет среднее арифметическое значение полученных результатов;

- выводит среднее арифметическое значение в PORTD.

Период дискретизации аналоговых сигналов определяется полным временем выполнения цикла.

Рисунок 5.2

Исходный текст программы

```
list p=16f877, st = OFF
include "p16f877.inc"
```

```
        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON & _PWRTE_ON &
_XT_OSC & _WRT_ENABLE_ON & _LVP_OFF & _DEBUG_ON & _CPD_OFF

        CBLOCK 0x20
        Ad0
        Cnt
        ENDC

        ORG  0x000
        nop
Init    BANKSEL  PORTD
        clrf     PORTD
        movlw    B'01000001';Fosc/8,канал AN0,разрешить A/D
        movwf    ADCON0
        BANKSEL OPTION_REG
        clrf     TRISD^80   ;PORTD все линии на выход
        movlw    B'00000100'        ;Рез-т влево, каналы AN0,AN1,AN3
        movwf    ADCON1^80          ;опорное напряжение VDD и VSS
        BANKSEL  PORTD
Main    movlw    .38
        call     Dly
        bsf      ADCON0,GO          ;Начало A/D преобразования
Wait0   btfsc    ADCON0,GO          ;Ожидание завершения
        goto     Wait0        ;     A/D преобразования
        bsf      ADCON0,CHS0
        movf     ADRESH,W
        movwf    Ad0
        movlw    .4
        call     Dly
        bsf      ADCON0,GO          ;Начало A/D преобразования
Wait1   btfsc    ADCON0,GO          ;Ожидание завершения
        goto     Wait1        ;     A/D преобразования
        bcf      ADCON0,CHS0
        movf     ADRESH,W
        addwf    Ad0,F
        rrf      Ad0,W
        movwf    PORTD
        goto     Main
Dly     movwf    Cnt
Dlp     decfsz   Cnt,F
        goto     Dlp
        return
        END
```

6 Вывод символьной информации

6.1 Описание задания

Необходимо разработать и кодировать на языке Ассемблера алгоритм программы вывода символьной информации на 7-сегментные индикаторы для МК типа PIC16F877. Варианты заданий приведены в таблице 6.1.

Таблица 6.1

Вариант	Задание
1	Информация из канала AN0 в шестнадцатеричной форме (0 .. 3FF)
2	Информация из канала AN0 в десятичной форме (0 .. 1023)
3	Информация из канала AN0 в шестнадцатеричной форме (0 .. FF)
4	Информация из канала AN0 в десятичной форме (-128 .. 127)
5	Цифровой вольтметр (рабочая шкала 0.000 .. 5.000 В)
6	Цифровой термометр (рабочая шкала 0 .. 125 °С)

6.2 Рекомендации по выполнению

Для управления индикаторами в отладочном стенде применен специализированный контроллер MC14489BP, схема включения приведена на рисунке 6.1. Эта микросхема обеспечивает вывод информации на 7-сегментные индикаторы (до пяти разрядов) в режиме динамического управления.

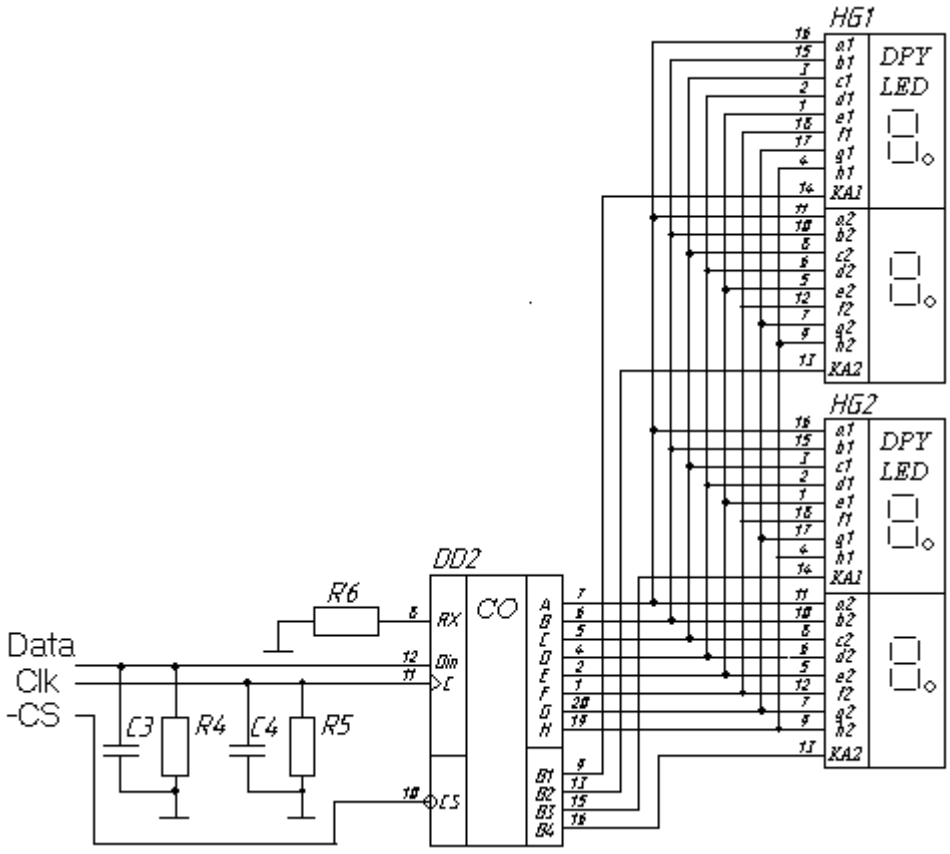


Рисунок 6.1

Данные от МК к контроллеру MC14489BP передаются через синхронный последовательный интерфейс по трем линиям:

- RE1 (DATA IN) – данные;
- RE0 (Clk) - синхроимпульсы;
- RB3 (-CS) – выборка кристалла.

В отладочном стенде установлены четыре разряда индикатора. Каждый из разрядов индикатора можно запрограммировать на работу в одном из режимов (таблица 6.2):

- стандартного hex-декодирования;
- специального декодирования;
- без декодирования (используются только сегменты a – d).

Таблица 6.2

Код разряда	Символ		Единичные индикаторы No Decode			
	Hex Decode	Special Decode	d	c	b	a
L L L L	0					
L L L H	1	c				on
L L H L	2	H			on	
L L H H	3	h			on	on
L H L L	4	J		on		
L H L H	5	L		on		on
L H H L	6	n		on	on	
L H H H	7	o		on	on	on
H L L L	8	P	on			
H L L H	9	r	on			on
H L H L	A	U	on		on	
H L H H	b	u	on		on	on
H H L L	C	У	on	on		
H H L H	d	-	on	on		on
H H H L	E	=	on	on	on	
H H H H	F	°	on	on	on	on

Режим работы определяется словом конфигурации Диаграммы сигналов для приема, а также формат конфигурационного слова приведены на рисунке 6.2. Разряды слова конфигурации C7 – C0 определяют:

- режим потребляемой мощности (C0);
- режимы декодирования разрядов (C1 – C7).

При этом разряды C1 – C5 позволяют выбрать либо режим стандартного hex-декодирования, либо иной режим для каждого разряда индикатора. Разряды C6 и C7 позволяют выбрать либо режим специального декодирования, либо режим без декодирования для групп разрядов 1-3 и 4-5 соответственно.

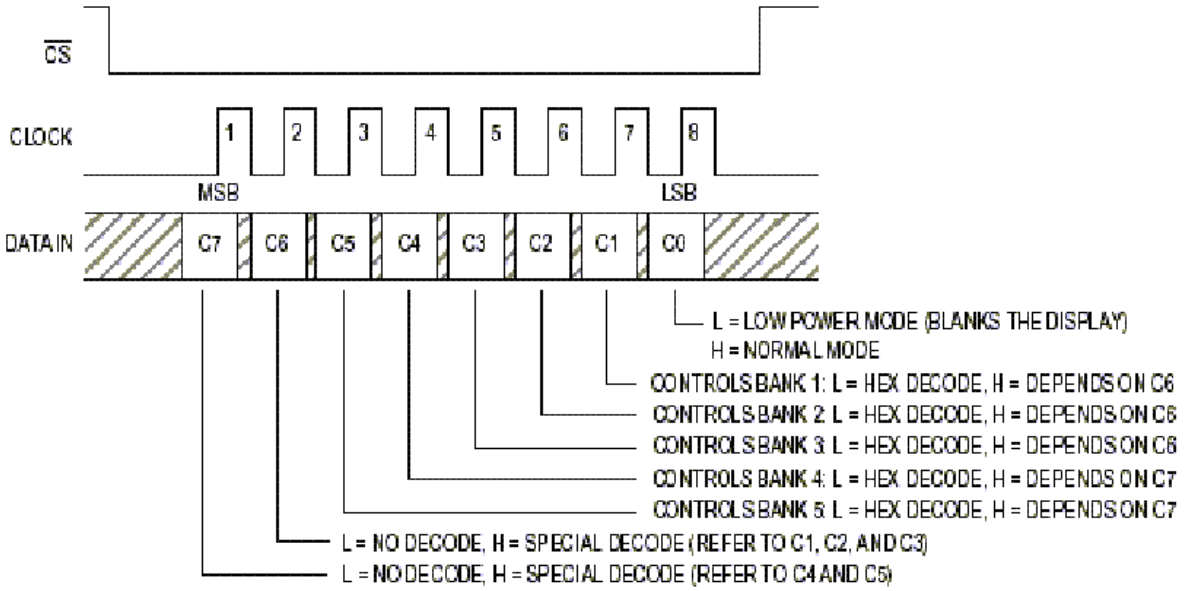


Рисунок 6.2

Диаграммы сигналов для приема, а также формат информационного слова приведены на рисунке 6.3. Информационное слово должно содержать 24 разряда, из них младшие 20 разрядов (D0 – D19) содержат пять тетрад кода отображаемых символов, разряды D20 – D22 управляют сегментами h (точка), разряд D23 – управляет яркостью свечения сегментов.

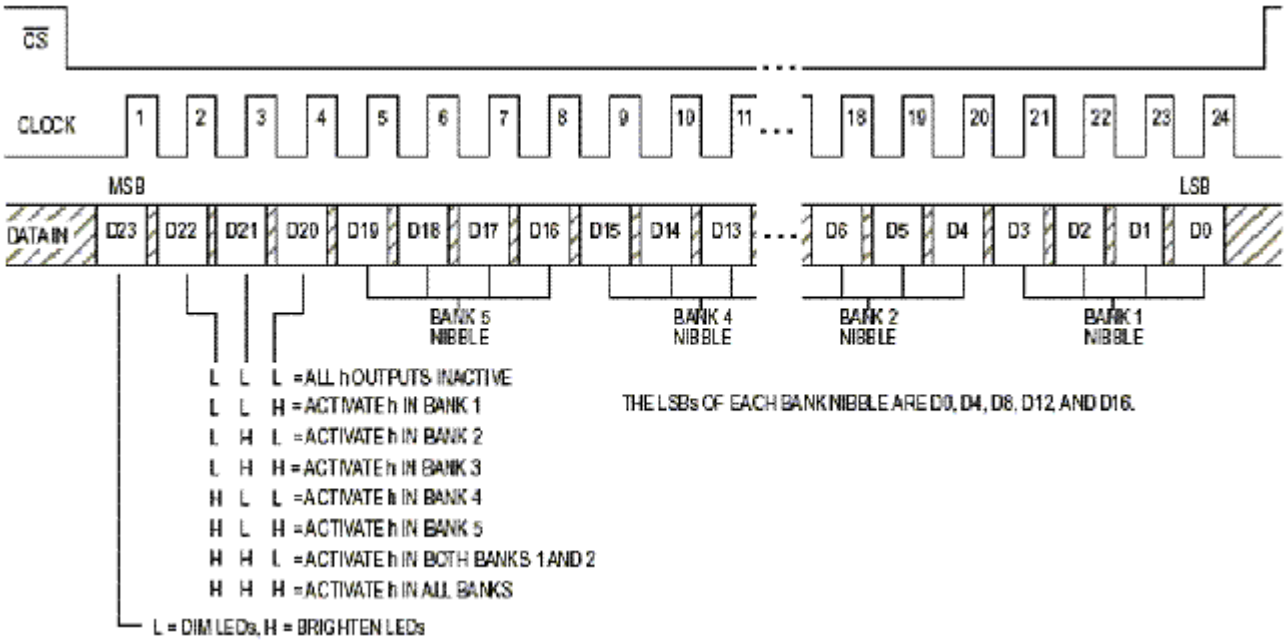


Рисунок 6.3

Для работы с индикатором необходимо линии 3 PORTB и 1-0 PORTE настроить на вывод. Для загрузки в регистры ADCON1, TRISB и TRISE можно использовать слова:

```
Init_AD1    EQU     b'10001110'     ;включены линии PORTE
Init_PB     EQU     b'11110111'     ;бит 3 - вывод
Init_PE     EQU     b'00000100'     ;линии 1-0 – вывод.
```

Для программного управления состоянием сигнальных линий удобно определить одноразрядные переменные:

```
#define     Data_in     PORTE,1         ;сигналы MC14489
#define     Clk         PORTE,0         ;
#define     Cs          PORTB,3         ;
```

Для хранения передаваемого по последовательному каналу байта информации используем ячейку Indval. В соответствии с рисунком 6.2 инициализацию MC14489 будет выполнять подпрограмма Control_MC14489:

```
Control_MC14489                 ;подпрограмма инициализации MC14489
            bcf         Cs
            movlw       b‘00000001’
            call        Send8
            bsf         Cs
            return --.
```

В соответствии с рисунком 6.3 вывод 24-х разрядного слова информации, из ячеек Fig2, Fig1, Fig0 будет выполнять подпрограмма Dysplay

```
Dysplay     bcf         Cs          ;подпрограмма отображ. 7сегм. символов
            movf        Fig2,w
            call        Send8
            movf        Fig1,w
            call        Send8
            movf        Fig0,w
            call        Send8
            bsf         Cs
            return --.
```

Подпрограмма вывода однобайтного слова Send8:

```
Send8       bcf         Clk         ;подпрограмма отображения  символа
            movwf       Indval
            movlw       8
            movwf       Cntind
Sendloop    bcf         Data_in
            rlf         Indval,f
            btfsc       STATUS,C
            bsf         Data_in
            nop
            bsf         Clk
            nop
            bcf         Clk
            decfsz      Cntind,f
            goto        Sendloop
            return --.
```

Для вариантов заданий 2, 4 – 6 потребуется подпрограмма преобразования двоичного кода в двоично-десятичный B2_BCD. Вход - двухбайтное число в ячейках H_byte, L_byte, выход – двоично-кодированное число в ячейках R0, R1, R2.

```
        ;------------------- Подпрограмма преобразования кода ----------------------
B2_BCD  bcf     STATUS,0
        movlw   .16
        movwf   count
        clrf    R0
        clrf    R1
        clrf    R2
loop16  rlf     L_byte
        rlf     H_byte
        rlf     R2
        rlf     R1
        rlf     R0
        decfsz  count
        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
        addwf0,w
        movwf   temp
        btfsc   temp,3
        movwf   0
        movlw   30
        addwf0,w
        movwf   temp
        btfsc   temp,7
        movwf   0
        retlw   0 --.
```

7 Обработка прерываний

7.1 Описание задания

Необходимо разработать и кодировать на языке Ассемблера алгоритм программы, в которой используется механизм прерываний. Программа должна выполнять две функции:

- отсчет времени и его отображение в секундах на 7-сегментных индикаторах;
- разнонаправленную модификацию содержимого ячейки Cnt при нажатии с последующим отпусканием кнопок RB1 и RB2 и вывод ее содержимого в порт PORTD (варианты заданий приведены в таблице 4.1).

7.2 Рекомендации по выполнению

Рассмотрим решение задачи на примере программы, реализующей алгоритм реверсивного счетчика числа нажатий на кнопки RB1 и RB2 по модулю 256. Возьмем за основу результаты, полученные при выполнении заданий 3, 4 и 6. Секундные интервалы времени будем формировать путем подсчета количества переполнений таймера TMR0. Тогда МК должен реагировать на следующие события:

- переполнение таймера TMR0;
- нажатие и отпускание кнопок RB1 и RB2.

Использование программного опроса флага T0IF и линий порта RB1 и RB2 может привести к пропуску переполнения таймера и, как следствие, появлению погрешности формирования секундного интервала в моменты времени, когда кнопки могут быть нажаты. Или, что менее вероятно, пропуску кратковременного нажатия на кнопки при ожидании переполнения таймера. Решается эта проблема путем использования механизма прерываний.

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 примем, что прерывания будут возникать при переполнении таймера TMR0, анализ состояния линий RB1 и RB2 будет выполняться путем программного опроса в главном цикле основной программы.

Микроконтроллеры PIC16F87X имеют 14 источников прерываний [1]. Логика прерываний приведена на рисунке 7.1. Регистр INTCON содержит флаги отдельных прерываний, биты разрешения этих прерываний и бит глобального разрешения прерываний GIE.

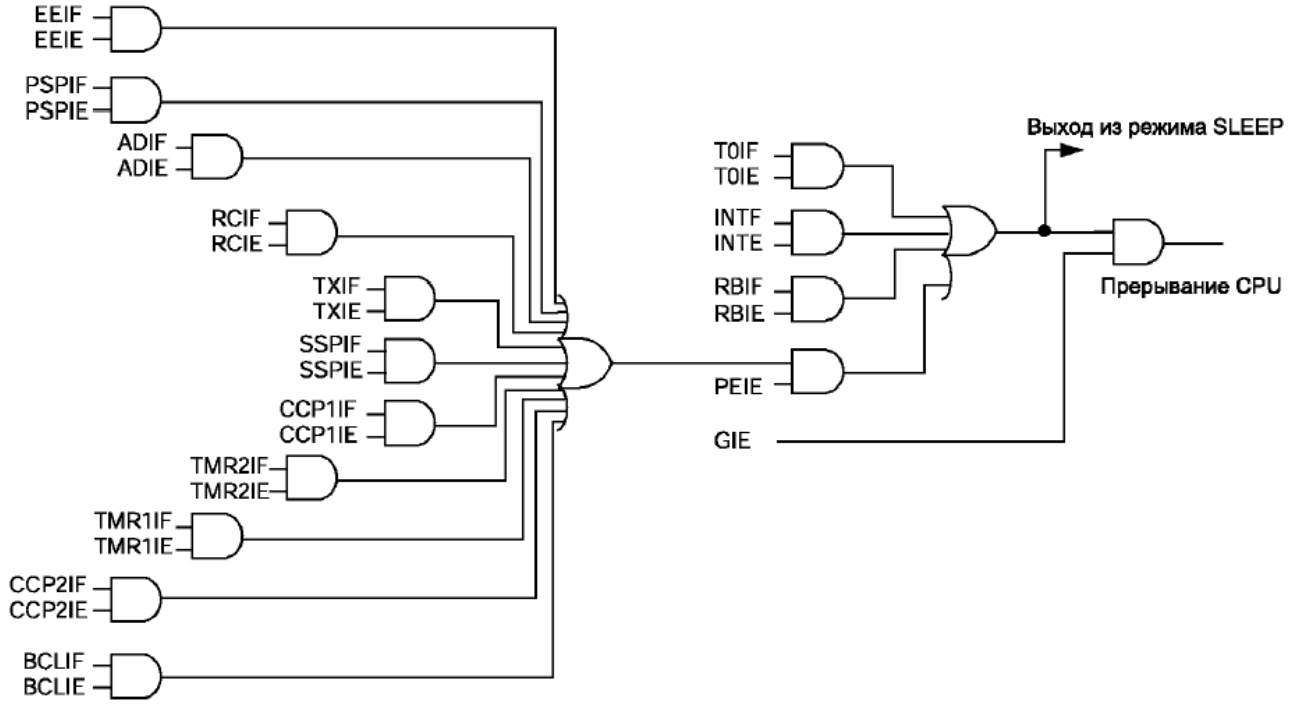


Рисунок 7.1

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 в регистрах INTCON, PIE1 и PIE2. При сбросе микроконтроллера бит GIE сбрасывается в ‘0’.

В регистре INTCON находятся флаги следующих прерываний: внешнего сигнала INT, изменения уровня сигнала на входах RB7:RB4, переполнения TMR0 (таблица 7.1). 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.

При переходе на 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 0004h. Источник прерываний можно определить проверкой флагов прерываний. Их необходимо сбросить программно перед разрешением прерываний, чтобы избежать повторного вызова.

Таблица 7.1

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 возвращении из подпрограммы обработки прерывания, по команде RETFIE, бит GIE аппаратно устанавливается в ‘1’, разрешая все немаскированные прерывания.

Обработка запроса внешнего прерывания может занимать три или четыре командных цикла процессора, в зависимости от того, в какой момент времени обнаружен запрос.

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 пользователь должен программно сохранять контекст (содержимое аккумулятора W и регистра STATUS). Пример приведен ниже:

```
Push    movwf   w_tenp          ;Сохранить содержимое W в W_TEMP
        movf    STATUS, w       ; Загрузить содержимое STATUS в W
        movwf   Status_Temp     ; сохранить W в STATUS_TEMP

        …….                     ; тело подпрограммы
        …….                     ; обработки прерывания

Pop     movwf   Status_Temp,w   ; Загрузить сохраненное значение
        movwf   STATUS          ; и восстановить его в рег. STATUS
        swapf   W_Temp, f       ; восстановить содержимое W
        swapf   W_Temp,w
```

Использование команды swapf для загрузки и выгрузки значений обусловлено тем, что эта команда не изменяет состояние флаг-бита нулевого результата Z регистра STATUS. Применение команды movf сократило бы программный код, но тогда при сохранении значения 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.

В качестве примера рассмотрим программу, которая:

- выполняет счет секунд, результат счета (переменная Time) выводится на семисегментные светодиодные индикаторы.

- реализует алгоритм реверсивного счетчика числа нажатий на кнопки RB1 и RB2 (по модулю 256), результат счета (переменная Cnt) выводится в PORTD.

За основу алгоритма программы можно взять схему на рисунке 4.1. При старте программа должна выполнить инициализацию портов ввода и вывода, контроллера MC14489, инициализацию ячеек Fig0-Fig2 и Cnt. В это время все прерывания должны быть запрещены. После инициализации прерывания необ-

ходимо разрешить. После чего в теле цикла программы анализируются состояния кнопок RB1 и RB2 и, в случае их нажатия, изменяется содержимое ячейки Cnt.

Схема подпрограммы обработки прерывания приведена на рисунке 7.2. Режим работы таймера и модуль счетчика переполнений таймера CntT определяется в соответствии с рекомендациями, изложенными в разделе 3.

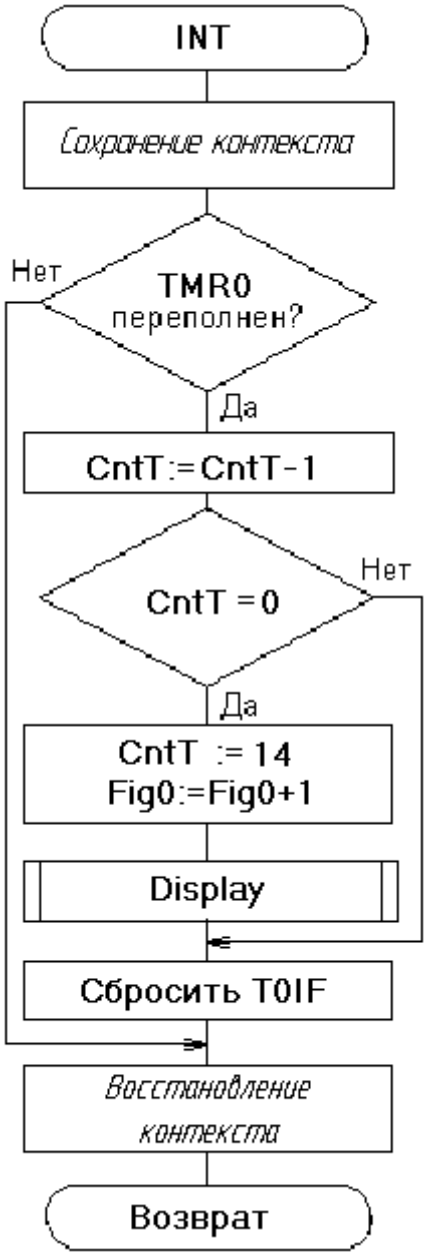


Рисунок 7.2

Ниже приводится текст основы программы, полученной с учетом принятых решений. Операторы отлаженных на других занятиях участков алгоритма представлены комментариями.

```
            LIST p=16f877, st = OFF
            INCLUDE "p16f877.inc"
            __CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON &
_PWRTE_ON & _XT_OSC & _WRT_ENABLE_ON & _LVP_OFF &
_DEBUG_ON & _CPD_OFF

            CBLOCK   0x20
            CntT              ;счетчик переполнений таймера
            Cnt               ;счетчик
            Fig0              ;разряды 1-0 7сегм. (время)
            Fig1              ;разряды 3-2 7сегм.
            Fig2              ;точки      7сегм.
            IndVal            ;слово в MC14489
            CntInd            ;счетчик разрядов MC14489
            Temp1             ;
            Temp2             ;
            w_temp            ;переменная для хранения w
            status_temp       ;переменная для хранения STATUS
            ENDC

Init_OPT    EQU  b'10000111'      ;настройка TMR0
Init_PB     EQU  b'11010111'      ;биты 3,5 - вывод
Init_PD     EQU  b'00000000'      ;все - вывод
Init_PE     EQU  b'11111100'      ;биты 1,0 - вывод

;------------------------------ Битовые переменные ---------------------------------
            #define     RB1       PORTB,1        ;кнопка "-"
            #define     RB2       PORTB,2        ;кнопка "+"
            #define     Data_in   PORTE,1        ;сигналы MC14489
            #define     Clk       PORTE,0        ;
            #define     Cs        PORTB,3        ;

            ORG         0x000             ;вектор старта
            nop
            goto        Begin

;---------------------- Подпрограмма обработки прерывания----------------------------
            ORG         0x004             ;вектор прерывания
            movwf       w_temp            ;сохранение контекста
            movf        STATUS,w
            movwf       status_temp
            btfss       INTCON,T0IF       ;прерывание по переполнению TMR0?
            goto        Ret               ; нет
```

```
            decfsz      CntT,f          ;секунда прошла?
            goto        Cf              ; нет
            movlw       .14
            movwf       CntT            ;загрузить счетчик переполнений
            incf        Fig0,f          ;инкремент секунд
            call        Dysplay
Cf          bcf         INTCON,T0IF     ;обнулить флаг прерывания
Ret                                     ;завершить обработку прерывания
            movf        status_temp,w   ;восстановление контекста
            movwf       STATUS
            swapf       w_temp,f
            swapf       w_temp,w
            retfie                      ; возврат из прерывания

      ;операторы подпрограммы отображ. 7сегм. символов Display
      ;операторы подпрограммы инициализации MC14489 Control_MC14489
      ;операторы подпрограммы задержки 15 мс  D15ms
;=================================================================
Begin                                   ;начало программы
            clrf        INTCON          ;запретить все прерывания

      ;операторы инициализации портов, таймера, MC14489

            clrf        Fig0            ;инициализация ячеек
            clrf        Fig1
            clrf        Fig2
            clrf        Cnt
            movlw       .14
            movwf       CntT            ;загрузить счетчик переполнений
            bsf         INTCON,GIE      ;разрешить прерывания
            bsf         INTCON,T0IE     ;разрешить прерывания от TMR0
Main                                    ;основной цикл

      ;операторы основного цикла

            goto        Main
            end
```

Полный текст программы сохранен в файле LR7.asm.

8 Программная реализация микропрограммного автомата

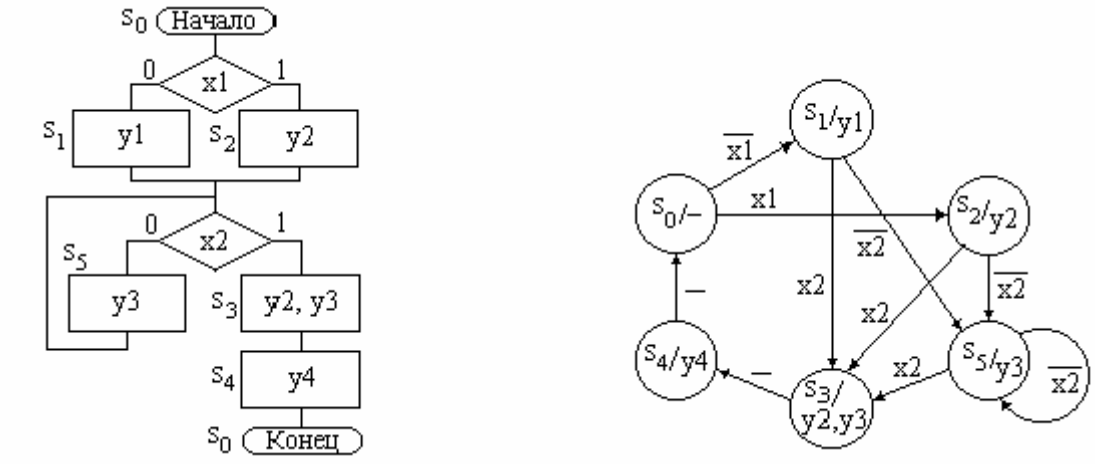
8.1 Описание задания

Необходимо разработать и кодировать на языке Ассемблера алгоритм программной реализации микропрограммного управляющего автомата.

На входные линии PORTE,2 и PORTA,5 МК поступает набор входных сигналов **X=<**x_1,x_2>. Текущие значения набора выходных сигналов **Y=<**y_1,y_2, y_3,y_4> формируются на линиях PORTD. Смена внутренних состояний автомата происходит при формировании положительного фронта внешнего сигнала Clk на линии PORTB,0. Микропрограммы заданы в форме граф-схемы алгоритма, варианты заданий приведены в таблице 8.1.

8.2 Рекомендации по выполнению

Рассмотрим пример реализации управляющего автомата, выполняющего микропрограмму, заданную граф-схемой алгоритма (ГСА) на рисунке 8.1. На ГСА выполнена отметка состояний для построения автомата Мура, рядом представлен граф автомата Мура.



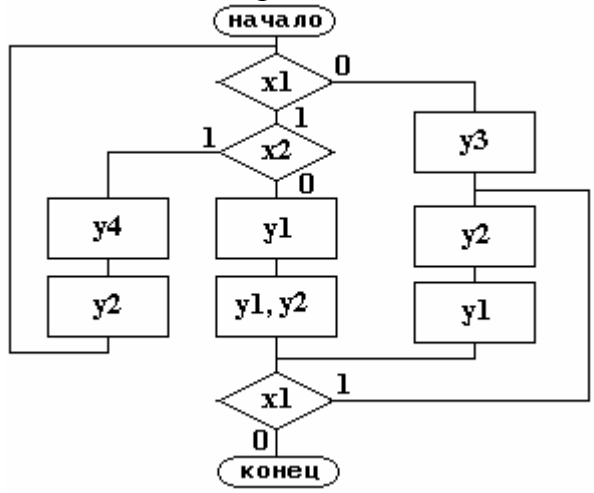
а) б)

Рисунок 8.1

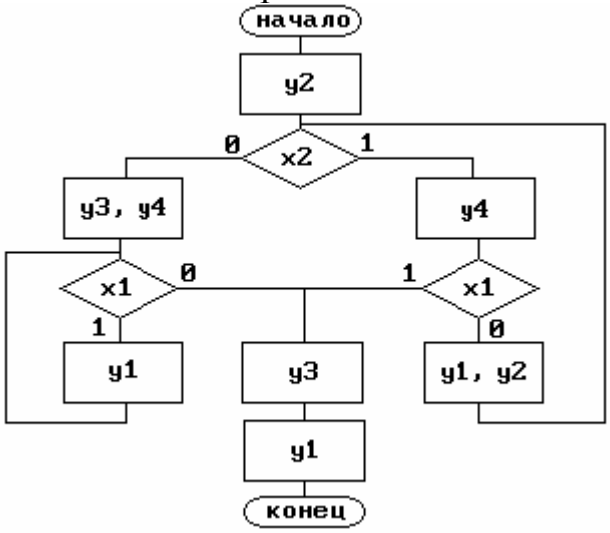
Если в графе автомата преобладают условные переходы, то для программной реализации более рационально использовать табличный метод. Табличный метод реализует представление алгоритма автомата в форме таблицы переходов-выходов. Значения функций переходов и выходов автомата зависят от входных сигналов и внутреннего состояния s_k. Каждому сочетанию входного набора $\boldsymbol{X}$ и кода текущего внутреннего состояния $\boldsymbol{Q^k}$ можно поставить в соответствие код следующего состояния $\boldsymbol{Q^{k+1}}$ и текущий выходной набор $\boldsymbol{Y}$. Общее число таких комбинаций составит $2^n \cdot N_s$, где n - число входных сигналов, N_s - число состояний автомата.

Таблица 8.1

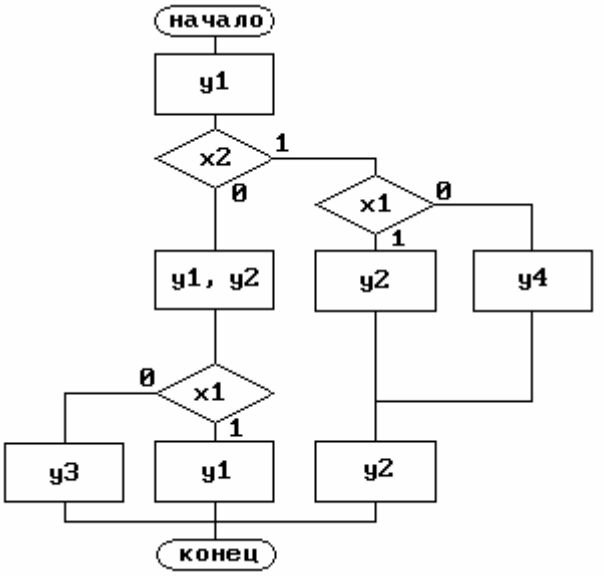
Вариант 1



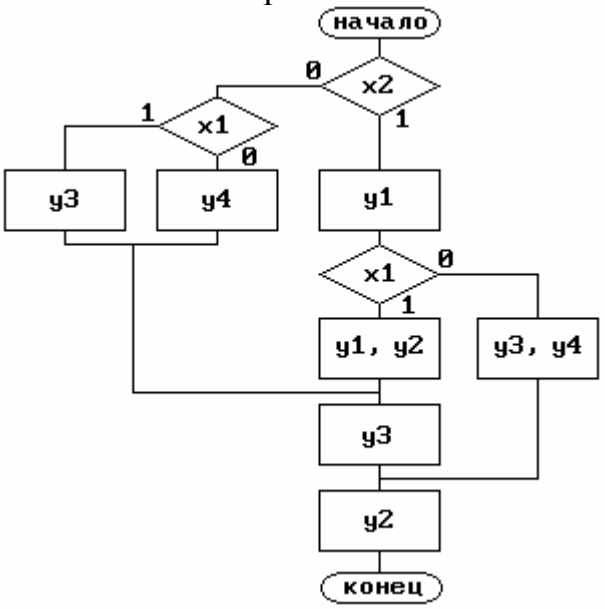
Вариант 2



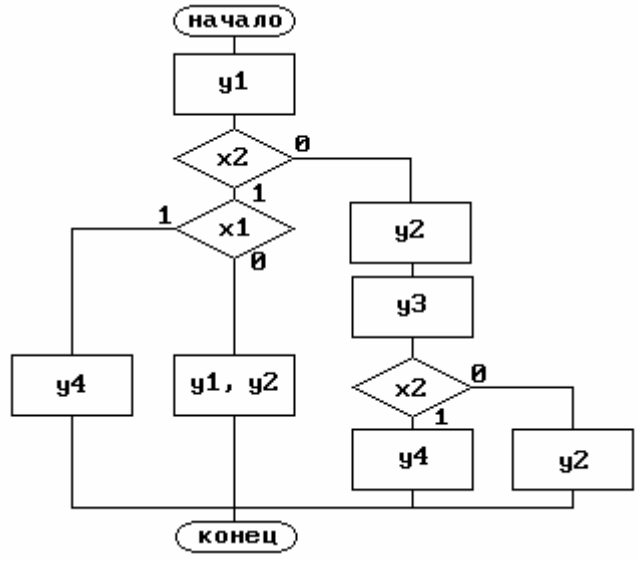
Вариант 3



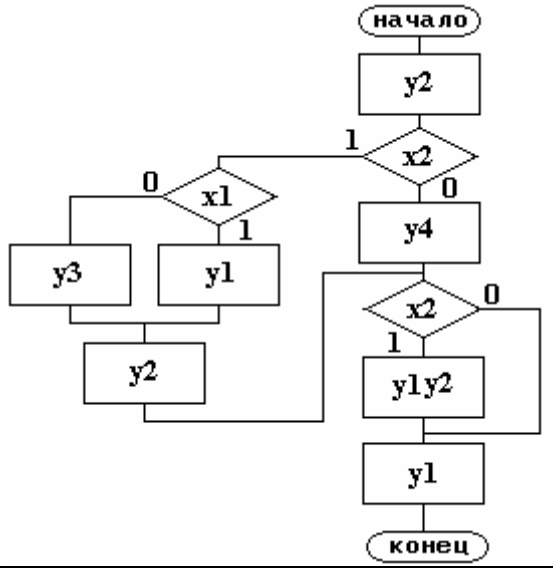
Вариант 4



Вариант 5



Вариант 6



Программная реализация этого метода сводится к формированию в зависимости от входного слова адреса, по которому в памяти находится соответствующее выходное слово. Под входным словом понимается конкатенация (результат соединения) входного набора и кода текущего состояния автомата ($\boldsymbol{Q^k}$, $\boldsymbol{X}$). Под выходным словом понимается конкатенация кода следующего состояния автомата $\boldsymbol{Q^{k+1}}$ и соответствующего ему выходного набора $\boldsymbol{Y}$.

Символьное представление функций переходов и выходов приведено в таблице 8.2.

Таблица 8.2

s	$x_2 x_1$				**Y**
	00	01	10	11	
s_0	s_1	s_2	s_1	s_2	-
s_1	s_5	s_5	s_3	s_3	y1
s_2	s_5	s_5	s_3	s_3	y2
s_3	s_4	s_4	s_4	s_4	y2y3
s_4	s_0	s_0	s_0	s_0	y4
s_5	s_5	s_5	s_3	s_3	y3

Выполним кодирование внутренних состояний. Число разрядов кода внутренних состояний l определяется числом состояний автомата N_s

$$l = ent(log_2 N_s),$$

где *ent* – операция округления до ближайшего большего целого числа. Результат кодирования состояний приведен в таблице 8.3.

Таблица 8.3

s	q_2	q_1	q_0
s_0	0	0	0
s_1	0	0	1
s_2	0	1	0
s_3	0	1	1
s_4	1	0	0
s_5	1	0	1

Для экономии памяти целесообразно, чтобы выходные слова хранились в ячейках со смежными адресами. Поэтому для полностью определенного автомата и $2^l > N_s$ сформируем входное слово в формате: $000\ q_2\ q_1\ q_0\ \ x_2\ x_1$. Для выходного слова используем формат $y_4\ y_3\ y_2\ y_1\ \ q_2\ q_1\ q_0\ 0$. Требуемое соответствие сформированных входных и выходных слов приведено в таблице 8.4.

Следует отметить, что позиции разрядов кода состояния во входном и выходном словах смещены на один разряд. Это необходимо учитывать при анализе кода текущего состояния автомата.

Таблица 8.4

Входное слово $000\ q_2\ q_1\ q_0\ \ x_2\ x_1$	Выходное слово $y_4\ y_3\ y_2\ y_1\ \ q_2\ q_1\ q_0\ 0$
000 000 00	0001 001 0
000 000 01	0010 010 0
000 000 10	0001 001 0
000 000 11	0010 010 0
000 001 00	0100 101 0
000 001 01	0100 101 0
000 001 10	0110 011 0
000 001 11	0110 011 0
000 010 00	0100 101 0
000 010 01	0100 101 0
000 010 10	0110 011 0
000 010 11	0110 011 0
000 011 00	1000 100 0
000 011 01	1000 100 0
000 011 10	1000 100 0
000 011 11	1000 100 0
000 100 00	0000 000 0
000 100 01	0000 000 0
000 100 10	0000 000 0
000 100 11	0000 000 0
000 101 00	0100 101 0
000 101 01	0100 101 0
000 101 10	0110 011 0
000 101 11	0110 011 0

Система команд микроконтроллеров PIC оптимизирована для выполнения табличных вычислений. Для идентификаторов, определенных в директиве

```
CBLOCK   0x20
InWord           ;входное слово конечного автомата
OutWord          ;выходное слово конечного автомата
…….
ENDC
```

алгоритм табличного вычисления выходного слова на языке ассемблера имеет вид

```
……..
StateMachine                        ; конечный автомат
        clrf        InWord
        btfsc       PORTE,2         ; анализ линии x1
        bsf         InWord,0        ;
```

```
        btfsc   PORTA,5         ; анализ линии x2
        bsf     InWord,1
        rlf     OutWord,w       ; сдвиг влево
        andlw   b'00011100'     ; маска кода состояния
        iorwf   InWord,w        ; текущее входное слово
        call    Tabl            ;
        movwf   OutWord         ; текущее выходное слово
        movwf   PORTD
        ……..

Tabl    addwf   PCL,f           ;смещение в таблице
        retlw   b'00010010'     ;первый элемент таблицы s0->s1
        ……..
        retlw   b'01100110'     ;последний элемент таблицы s5->s3
```

При разработке программы следует следить, чтобы при вычислении адреса ячейки таблицы не происходило переполнение. Если это возможно, то следует позаботиться о размещении первого элемента таблицы. Максимально возможный объем таблицы значений – 256 ячеек.

В соответствии с заданием переходы автомата должны выполняться по фронту сигнала Clk. Поэтому приведенный участок кода должен выполняться при изменении потенциала на соответствующей линии порта ввода. Возможен программный опрос этой линии или обработка прерывания.

Список использованных источников

1 PIC16F87X [Электронный ресурс]: Техническое описание. – М.: ООО «Микро-Чип», 2001. – 61 с. – Режим доступа : WWW.URL : http: // www.microchip.ru/. – 27.06.2008.

2 MPASM [Электронный ресурс]: Руководство пользователя. – М.: ООО «Микро-Чип», 2001. – 183 с. – Режим доступа: WWW.URL: http: // www.microchip.ru/. – 27.06.2008.

Приложение А
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 регистре f, пропустить если 0	1(2)	
BTFSS	f,b	Проверить бит b в регистре f, пропустить 1 если	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