МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Омский государственный технический университет»

РАЗРАБОТКА И АССЕМБЛИРОВАНИЕ ПРОГРАММ В MPLAB IDE

Учебное текстовое электронное издание
локального распространения

Омск
Издательство ОмГТУ
2017

Сведения об издании: 1, 2

Составитель *В. Ф. Нестерук*, канд. техн. наук

Разработка и ассемблирование программ в MPLAB IDE : метод указания / Минобрнауки России, ОмГТУ ; [сост. В. Ф. Нестерук]. – Омск : Изд-во ОмГТУ, 2017.

Представлены рекомендации по применению IDE (integrated development environment) – интегрированной среды разработки MPLAB 8.6 для создания программ на языке ассемблера для микроконтроллеров PIC фирмы Microchip Technology. Рассмотрен процесс создания нового проекта для работы в IDE, составления и ассемблирования программы, а также её отладки в возможных режимах.

Издание предназначено для обучающихся по направлению подготовки бакалавров «Информатика и вычислительная техника» и направлению подготовки магистров «Отказоустойчивые вычислительные системы и компьютерный анализ данных» при выполнении лабораторных работ, заданий к практическим занятиям, курсовых проектов и расчётно-графических работ по дисциплинам «Архитектура микроконтроллеров» и «Микропроцессорные технологии».

Рекомендовано редакционно-издательским советом
Омского государственного технического университета

1 электронный оптический диск

Оригинал-макет издания выполнен в Microsoft Office Word 2007/2010
с использованием возможностей Adobe Acrobat Reader.

Минимальные системные требования:

- процессор Intel Pentium 1,3 ГГц и выше;
- оперативная память 256 Мб и более;
- свободное место на жестком диске 260 Мб и более;
- операционная система Microsoft Windows XP/Vista/7/10;
- разрешение экрана 1024×768 и выше;
- акустическая система не требуется;
- дополнительные программные средства Adobe Acrobat Reader 5.0 и выше.

Редактор *М. А. Болдырева*
Компьютерная верстка *Ю. П. Шелехиной*

Сводный темплан 2017 г.
Подписано к использованию 26.04.17.
Объем 5,58 Мб.

Издательство ОмГТУ.
644050, г. Омск, пр. Мира, 11; т. 23-02-12
Эл. почта: info@omgtu.ru

ВВЕДЕНИЕ

Настоящее издание подготовлено с целью ознакомления студентов с организацией процесса разработки и отладки ассемблерных программ для микроконтроллеров PIC фирмы Microchip Technology с использованием IDE (integrated development environment) – интегрированной среды разработки MPLAB 8.6. Данная IDE кроме создания программы и её отладки позволяет имитировать взаимодействие микроконтроллера с периферийным оборудованием как в синхронном, так и в асинхронном режиме.

Исходным документом для работы в среде MPLAB 8.6 является схема алгоритма решения требуемой задачи. Программная реализация алгоритма может быть представлена как на языке ассемблера, так и на языке С. В качестве примера рассматривается задача опроса матричной клавиатуры размером 4 х 4 с вызовом прерывания при нажатии на любую клавишу. Клавиатура ориентирована на использование в системе реального времени с разделением времени по флагам. В качестве языка программирования используется ассемблер.

Разработка и отладка программы осуществляются в рамках соответствующего проекта, под которым понимается генерация соответствующей среды, ориентированной на решение конкретной задачи. При создании проекта определяются: путь размещения проекта, тип моделируемого микроконтроллера, средства разработки и отладки, способы стимулирования периферийных воздействий и пр. Методика создания нового проекта рассматривается в данных методических указаниях.

СОЗДАНИЕ НОВОГО ПРОЕКТА В MPLAB 8.6 IDE

Перед тем как создавать проект, необходимо определить путь к папке проекта. Можно создать пустую папку в какой-либо директории при условии, что путь к папке не содержит символов кириллицы. Далее в этой папке разместить пустой или рабочий файл с расширением .asm.

На следующем этапе через соответствующую иконку на рабочем столе осуществляется вызов MPLAB 8.6. В ответ открывается окно IDE (рис. 1).

Для создания нового проекта вызывается мастер проекта через меню Project –> Project Wizard (рис. 2). Соответствующее окно показано на рис. 3. Нажимая на кнопку «Далее», вы вызываете выпадающее окно выбора симулируемого устройства, где по центру размещается вспомогательное окошко с вертикальным скроллингом (рис. 4). В качестве примера в окошке «Device» выбран микроконтроллер PIC16C56. В общем случае IDE MPLAB 8.6 позволяет осуществлять разработку и отладку программ для всех архитектур микроконтроллеров PIC фирмы Microchip Technology, начиная с младших подсемейств, по общей методике.

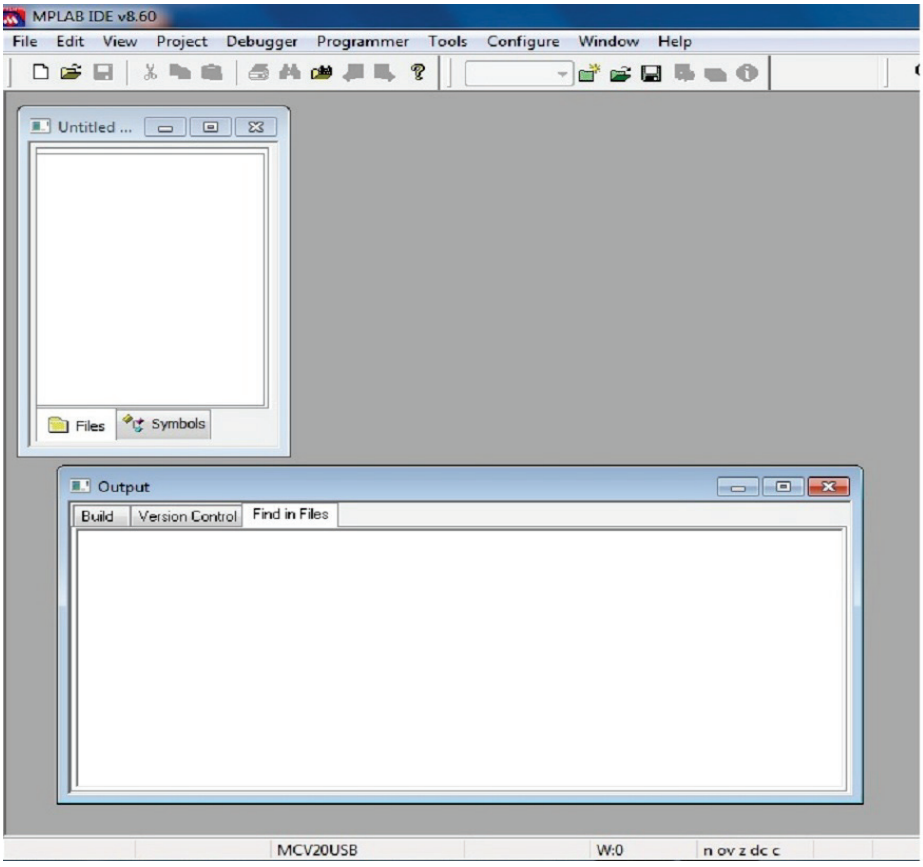

Рис. 1

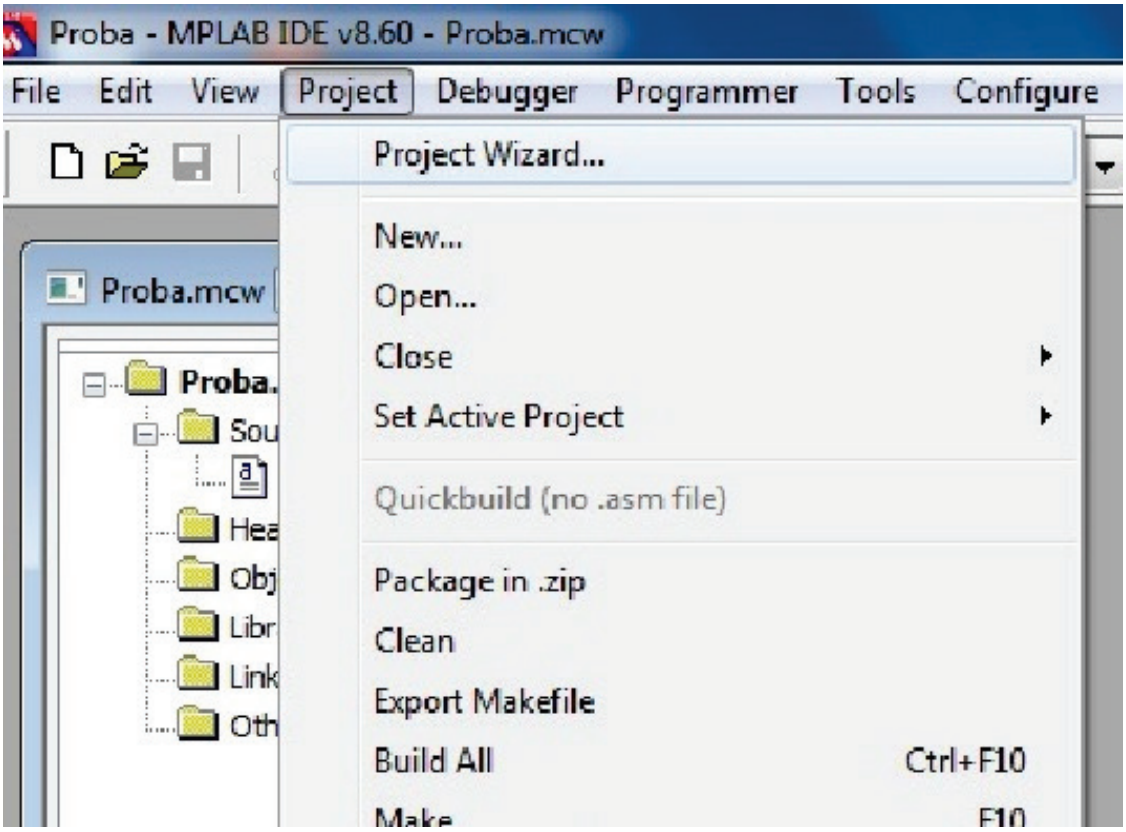
Proba - MPLAB IDE v8.60 - Proba.mcw
File Edit View Project Debugger Programmer Tools Configure
Project Wizard...
New...
Open...
Close
Set Active Project
Quickbuild (no .asm file)
Package in .zip
Clean
Export Makefile
Build All Ctrl+F10
Proba.mcw

Рис. 2

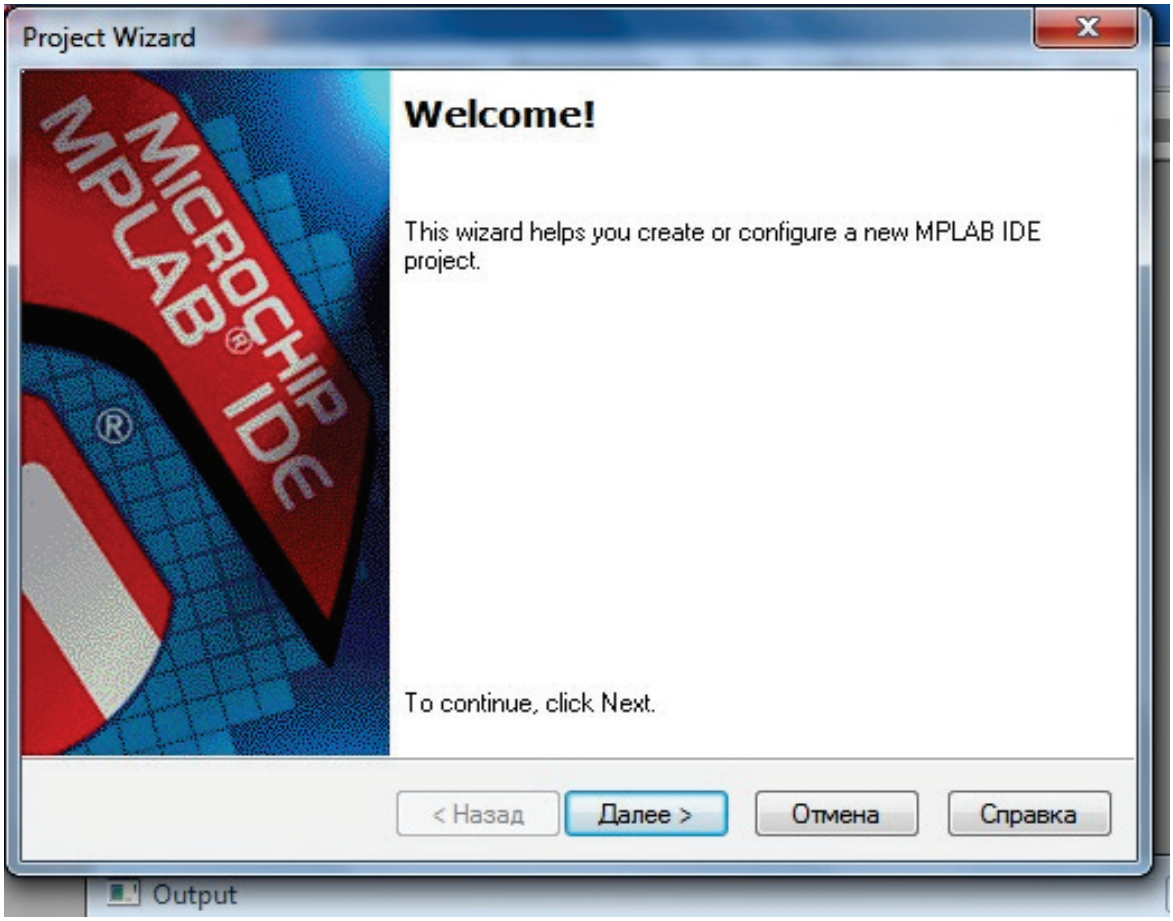
Project Wizard
Welcome!
This wizard helps you create or configure a new MPLAB IDE project.
To continue, click Next.
< Назад
Далее >
Отмена
Справка
Output

Рис. 3

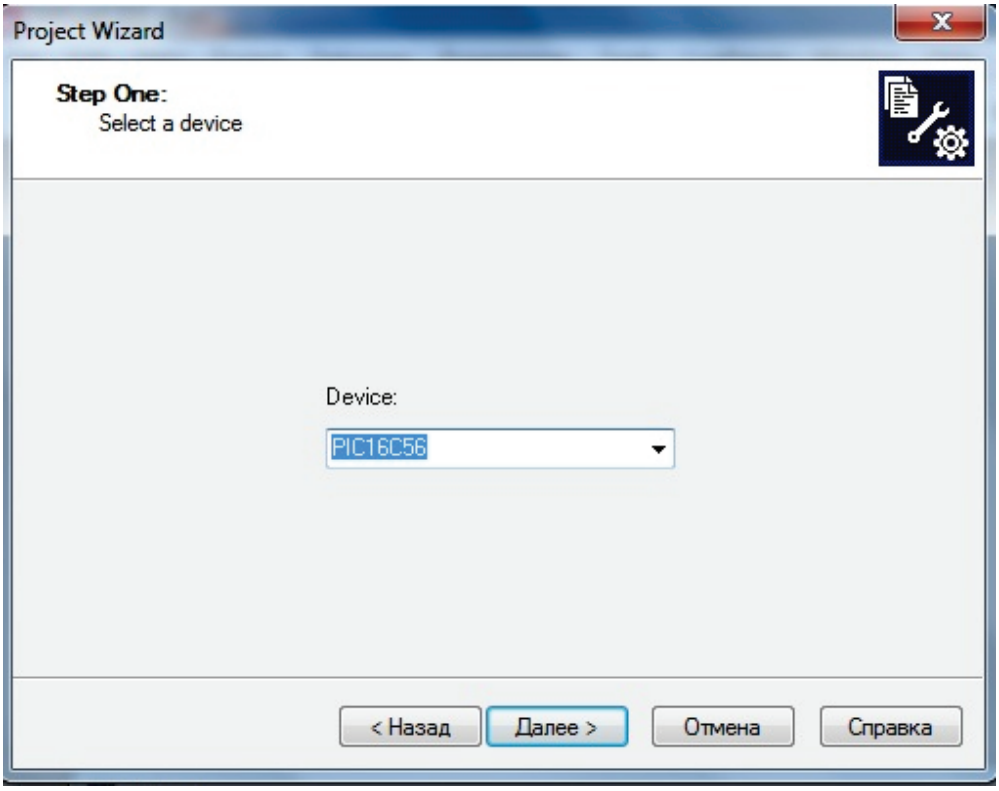

Рис. 4

Переход к очередной настройке производится по кнопке «Далее». Открывается выпадающее окошко выбора средств ассемблирования (рис. 5).

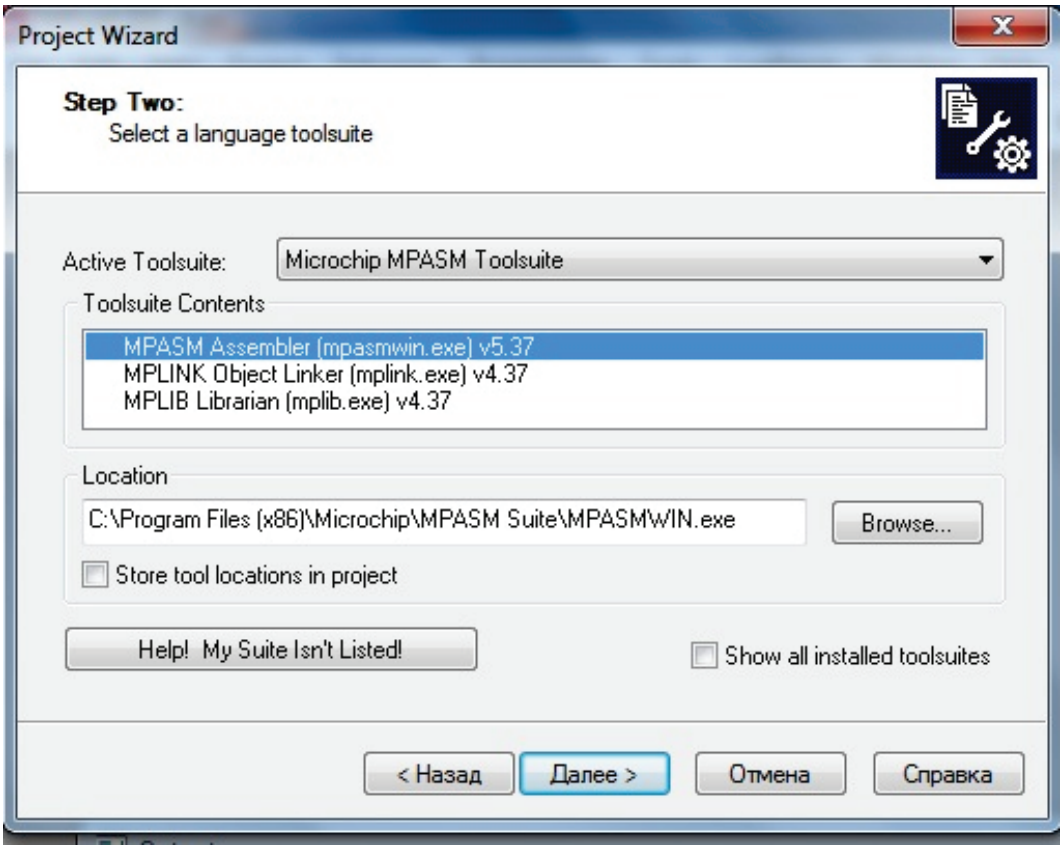

Рис. 5

На следующем шаге настройки в выпадающем окошке (рис. 6) задаётся путь к проекту.

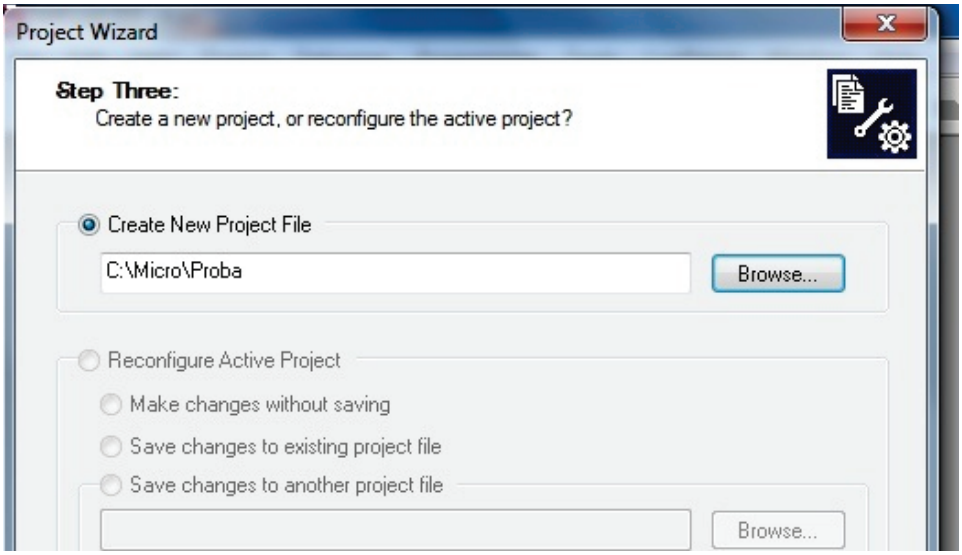

Рис. 6

По кнопке «Далее» открывается (рис. 7) окошко добавления в проект исходного .asm-файла. Для этого в дереве файлов левой части окошка отмечается требуемый файл и при нажатии на кнопку «Add>>» он переносится в правую часть. В примере это файл proba.asm.

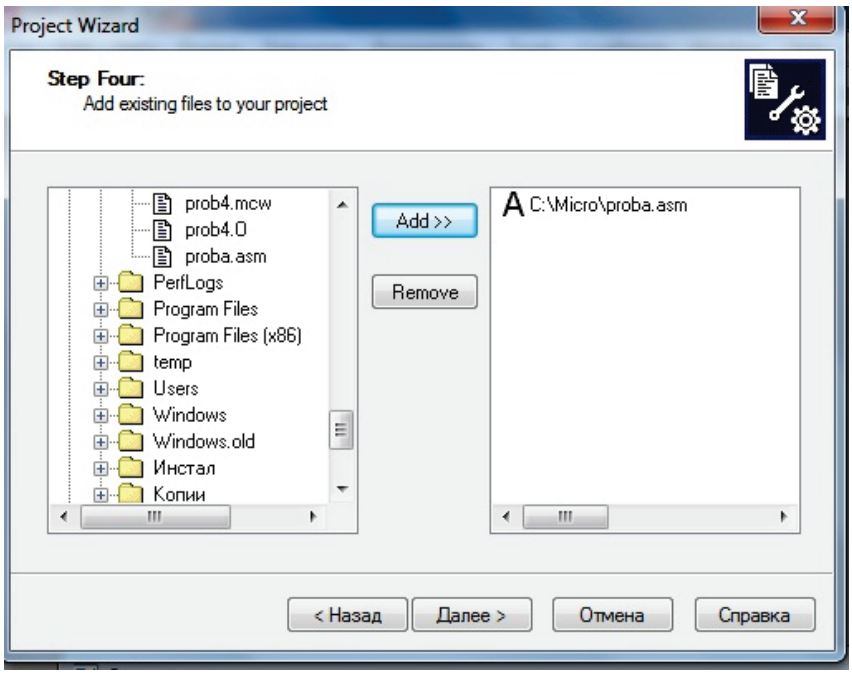

Рис. 7

На этом создание проекта завершается, и по кнопке «Далее» выпадает окошко (рис. 8) завершения работы мастера проекта, в котором нажимаем

на кнопку «Готово». Открывается окно проекта (см. рис. 1), и выводится запрос (рис. 9) о способе размещения файла – абсолютном или перемещаемом. Нажимаем на кнопку «Relocatable».

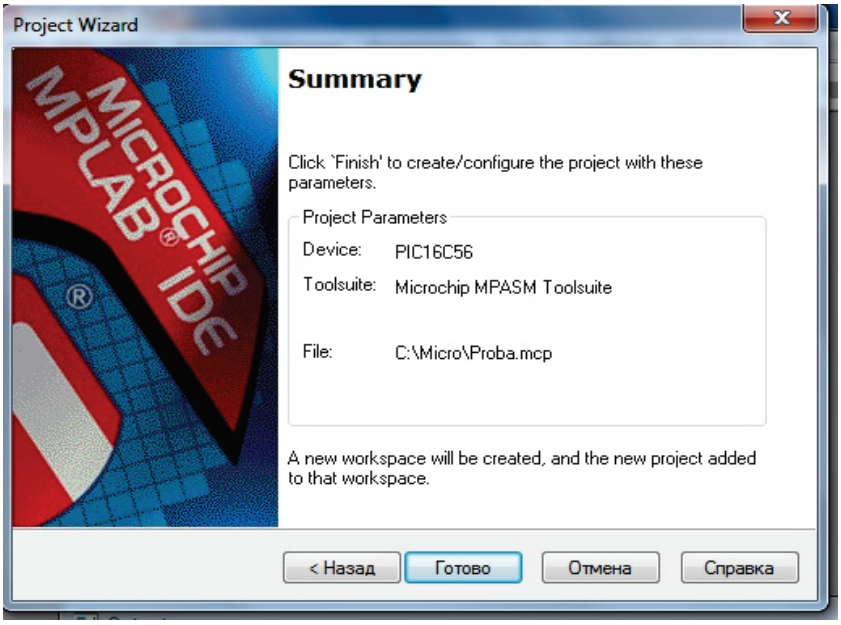

Рис. 8

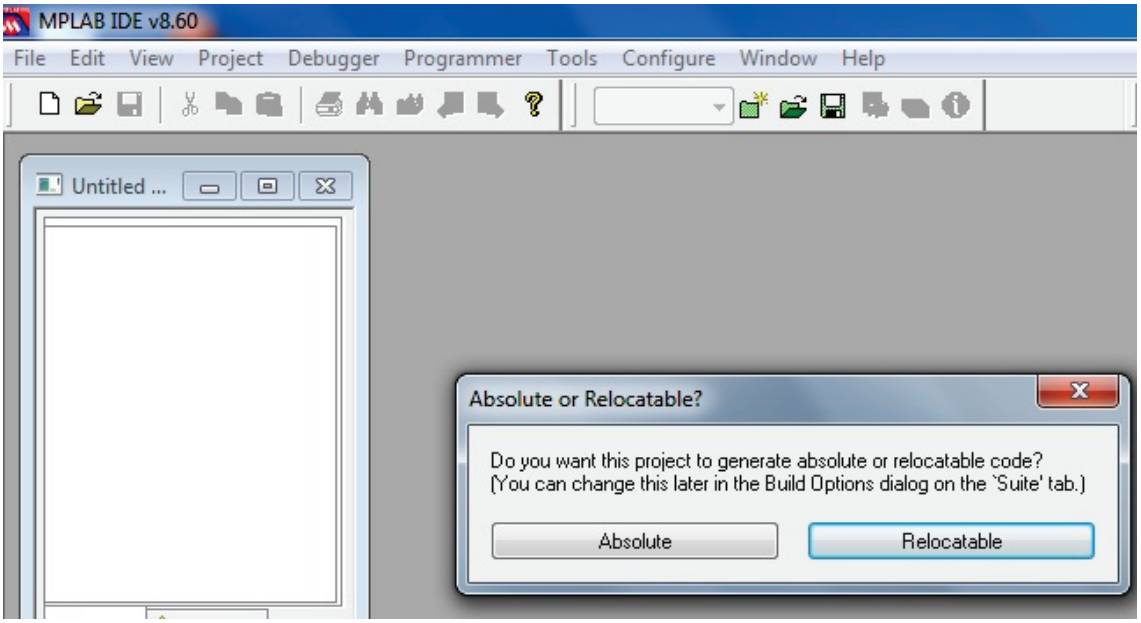

Рис. 9

В результате открывается рабочее окно проекта (рис. 10) с двумя выпадающими окошками, в одном из которых размещается дерево подключаемых файлов, а в окошке «Output» – выходные данные текущих процессов. Подключаемый .asm-файл, который был заранее создан и размещён в папке проекта (см. рис. 7), выбираем в дереве проекта в папке «Source Files.

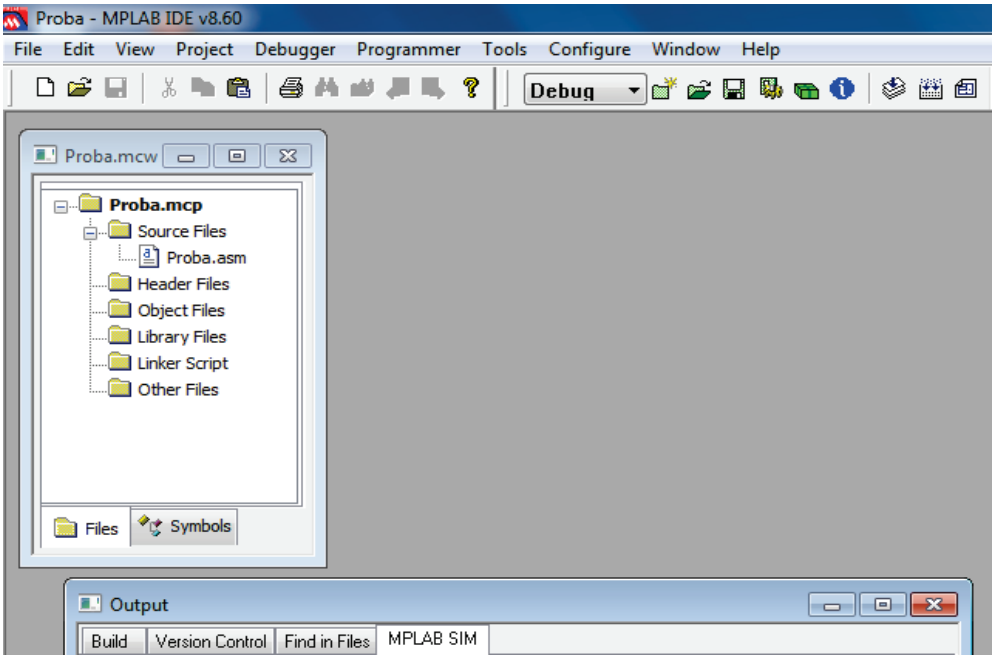

Рис. 10

Для вызова файла в рабочее окно проекта (рис. 11) достаточно выполнить два щелчка левой кнопкой мыши на иконке .asm-файла в дереве проекта. Если вызываемый файл был пустым, откроется пустое окошко, в котором с помощью встроенного текстового редактора можно разместить отлаживаемую программу.

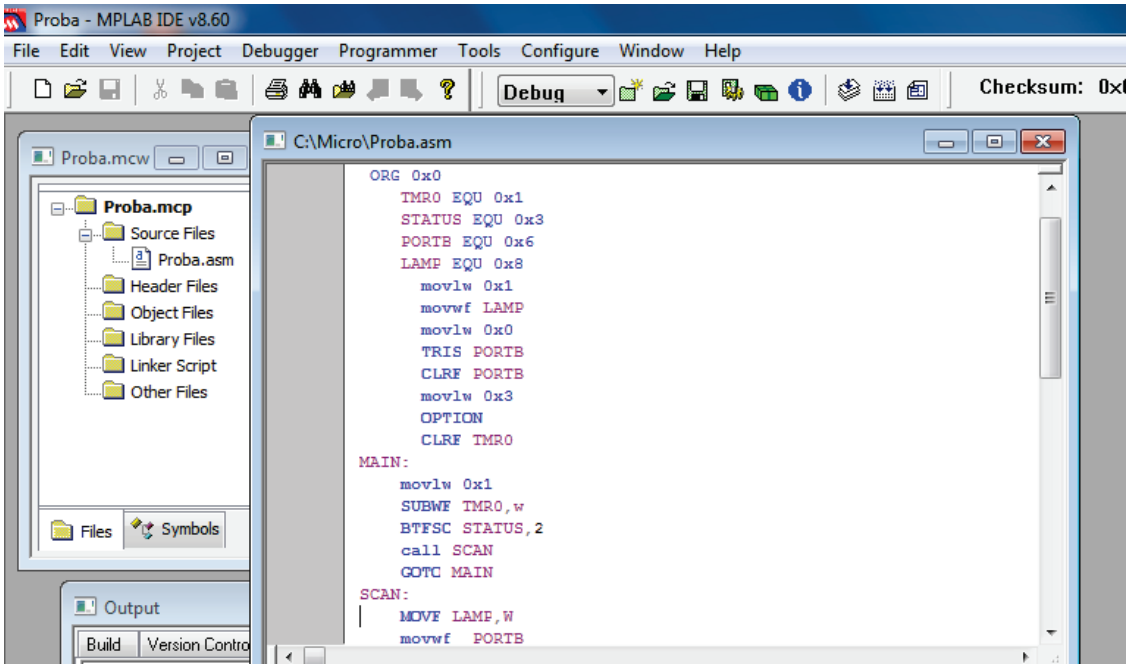

Рис. 11

Если текст программы был размещён в .asm-файле заранее, он также допускает редактирование в ходе отладки. Предварительное ассемблиро-

вание вызывается через меню Project –> Build All. Результаты ассемблирования размещаются в окошке «Output». Если ошибок нет, выводится сообщение «BUILD SUCCEEDED» (рис. 12). Панель инструментов рабочего окна при этом расширяется на иконки управления режимами отладки (рис. 13). В противном случае перечисляются ошибки ассемблирования, выделенные красным цветом.

Запуск режима отладки осуществляется через меню Debugger –> Select Tool –> MPLAB SIM. Для наблюдения за средой микропроцессора в ходе отладки меню View –> File Registers и View –> Special Function Registers позволяют вызвать в рабочее окно (рис. 14) соответствующие выпадающие окошки, в которых отображается текущее состояние упомянутых регистровых файлов. Состояние программной памяти может быть отображено через вызов меню View –> Program Memory.

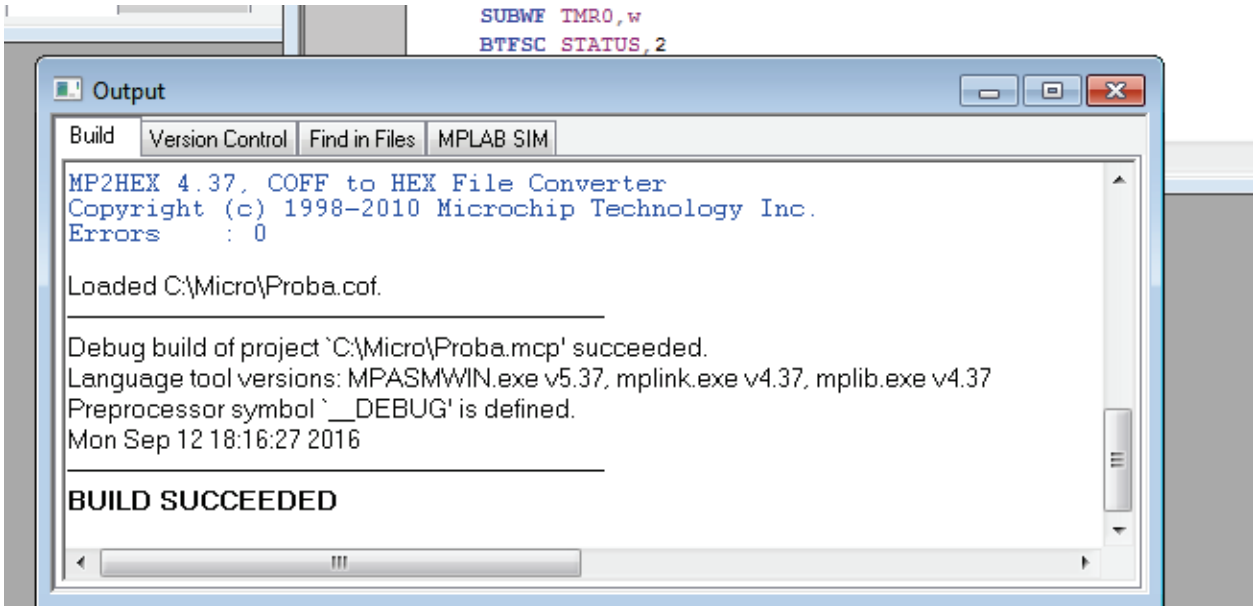

Рис. 12

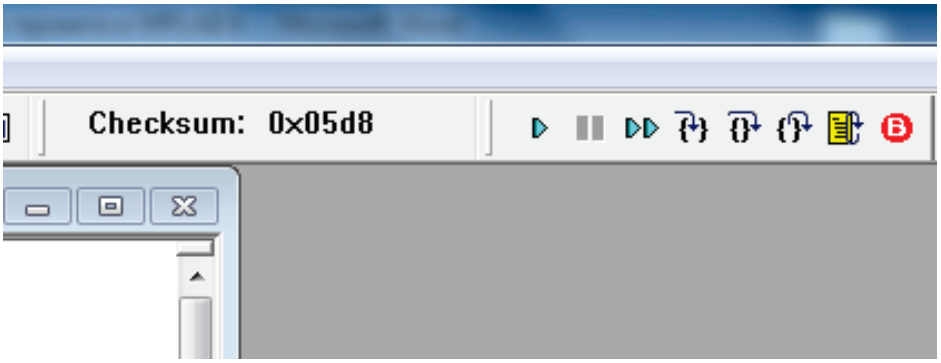

Рис. 13

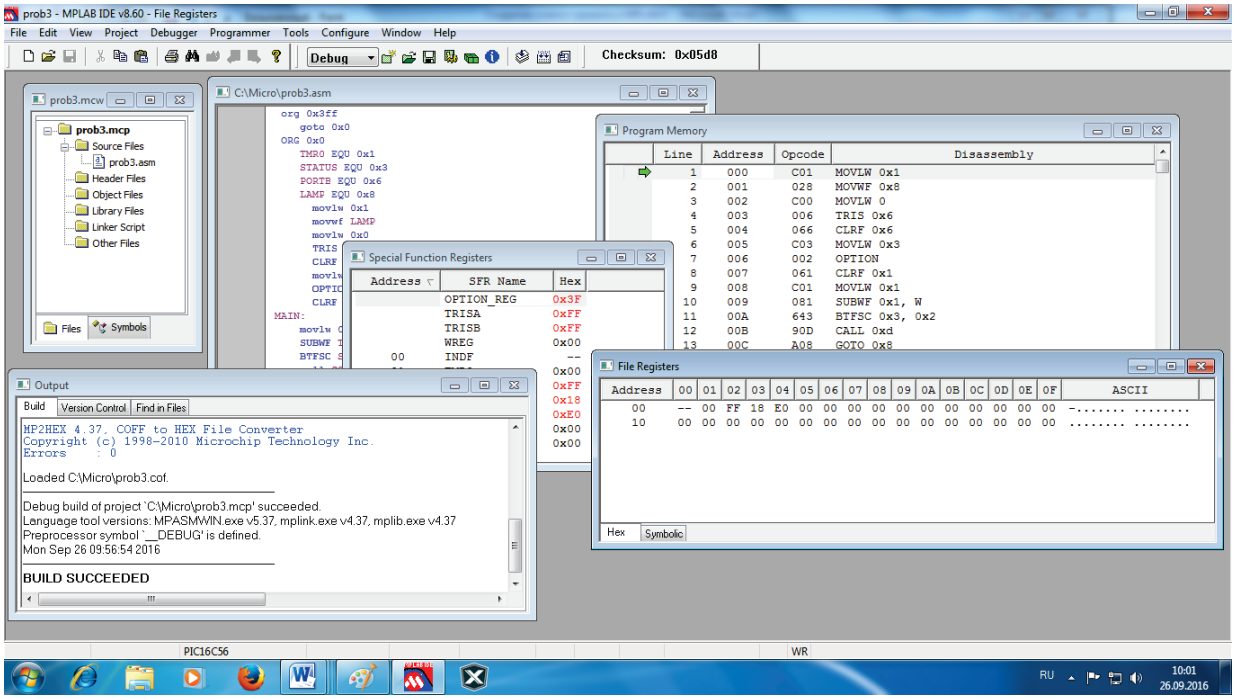

Рис. 14

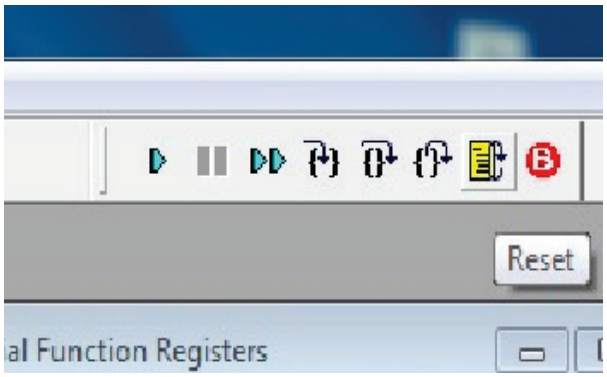

Рис. 15

Управление режимами отладки производится щелчком левой кнопки мыши по соответствующей иконке панели инструментов. Так, сброс отладчика на начало отлаживаемой программы осуществляется по иконке «Reset» (рис. 15). Сброс сопровождается появлением слева от начальной строки программы зелёной стрелки, которая указывает на исполняемую в текущем цикле отладки команду при пошаговом режиме отладки или на начальную команду автоматического прогона при отладке.

Рис. 16

В пошаговом режиме исполнение текущей команды запускается при щелчке по иконке «Step Into» – «Шаг с заходом» (рис. 16), что трактуется как «Пошаговое исполнение с заходом в процедуры». При этом как команды основной программы, так и команды процедуры выполняются

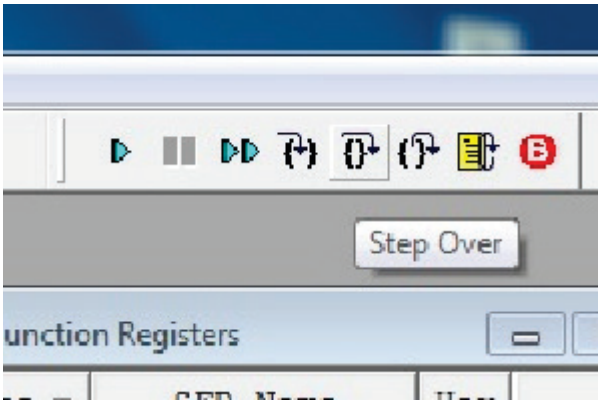

Рис. 17

Рис. 18

Рис. 19

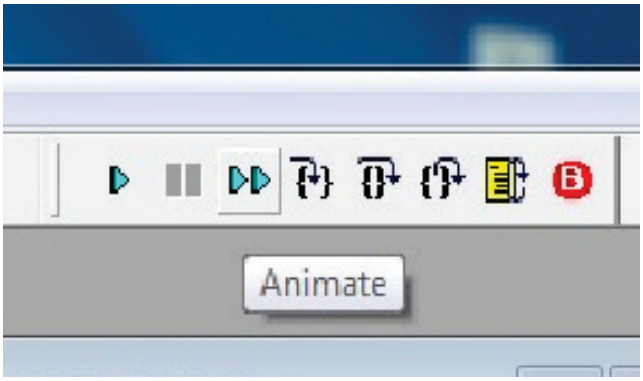

Рис. 20

пошагово, что позволяет отследить текущее состояние регистров после исполнения очередной команды.

Щелчок по иконке «Step Over» – «Шаг с обходом» (рис. 17) обеспечивает исполнение текущей команды основной программы в пошаговом режиме, кроме команд вызова процедур (вызываемая процедура выполняется в автоматическом режиме с остановом на очередной команде основной программы после возврата из процедуры).

Щелчок по иконке «Step Out» – «Шаг с выходом» (рис. 18) позволяет при пошаговом исполнении текущей процедуры закончить процедуру в автоматическом режиме и вернуться в основную программу с остановом на очередной команде основной программы после возврата из процедуры.

Щелчком по иконке «Run» – «Выполнить» (рис. 19) запускается автоматический прогон программы до директивы End.

Иконка «Animate» (рис. 20) – «Анимация» управляет автоматическим исполнением программы с выдержкой некоторого времени перед запуском очередной команды, что позволяет отслеживать результат работы предыдущей команды.

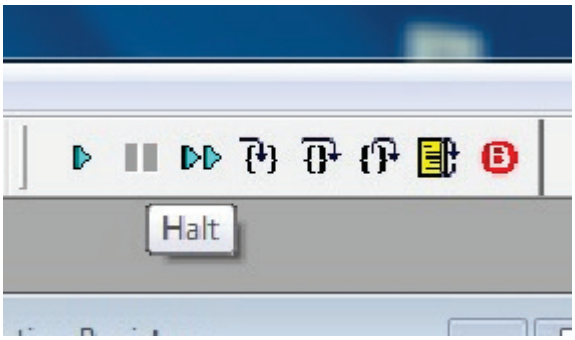

Рис. 21

Щелчком по иконке «Halt» – «Останов» (рис. 21) останавливается текущий режим отладки.

Иконка «Breakpoints» – «Точки останова» (рис. 22) позволяет двойным щелчком устанавливать точки останова на требуемой строке программы, что даёт возможность останова на соответствующей команде при автоматическом прогоне. Установка точек производится перед запуском программы и отображается красной меткой слева от текста команды (рис. 23). Повторное двойное нажатие на иконку «Breakpoints» сбрасывает ранее установленную точку останова. Если в окне текста .asm-файла было установлено несколько точек останова, то при пуске прогона после очередного останова выполнение программы продолжится до очередной точки останова, на что укажет курсор.

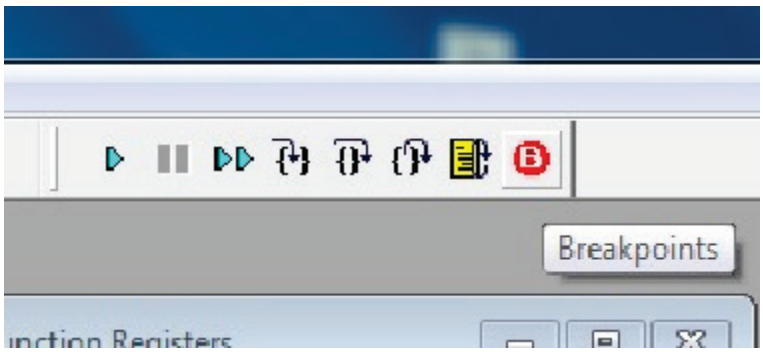

Рис. 22

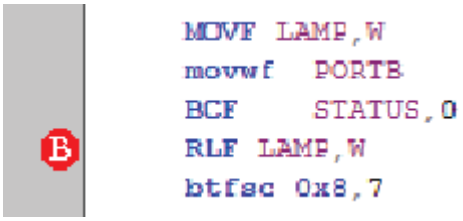

Рис. 23

МОДЕЛИРОВАНИЕ ВЗАИМОДЕЙСТВИЯ С ПЕРИФЕРИЙНЫМ ОБОРУДОВАНИЕМ

В качестве объекта управления в примере используется матричная клавиатуру 4 × 4. Опрос клавиатуры будем осуществлять по прерыванию, вызываемому нажатием на одну из клавиш клавиатуры. Схема клавиатуры представлена на рис. 24.

В каждой точке пересечения вертикальных и горизонтальных шин размещается контактная группа соответствующей клавиши. В исходном состоянии контактные группы разомкнуты и на выходных вертикальных шинах RB<7-4> присутствуют потенциалы логической единицы.

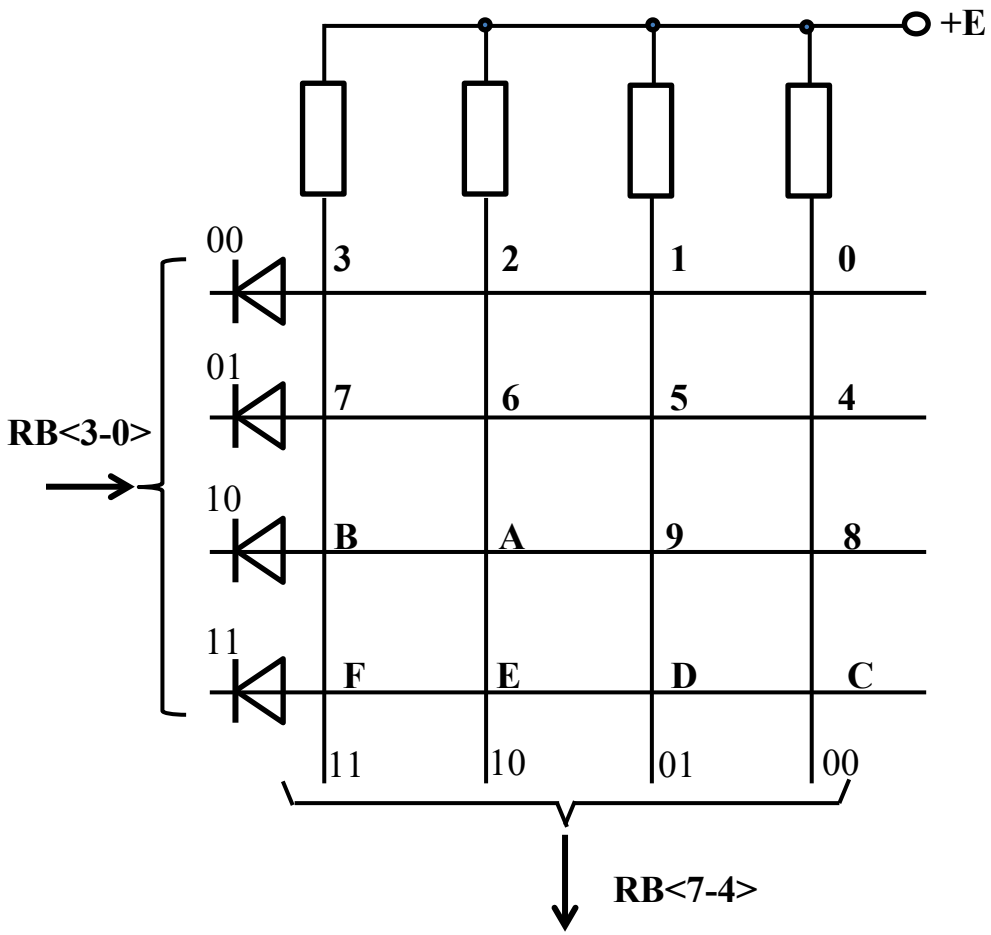

Рис. 24

Входные горизонтальные шины RB<3-0> разделены защитными диодами, предохраняющими выходные каскады порта управления этими шинами от взаимного замыкания при случайном нажатии на две соседние клавиши какой-либо вертикальной шины.

В качестве управляющего микроконтроллера будем использовать PIC16C74, к шинам порта RB которого подключена данная матричная клавиатура. В системе прерывания PIC16C74 имеется режим реакции на изменение сигналов в шинах порта RB<7-4>, что позволяет не использовать дополнительных логических схем для обнаружения факта нажатия на одну из клавиш клавиатуры.

Учитывая эту особенность системы прерывания, сканирование горизонтальных шин клавиатуры следует осуществлять логическим нулём по входам RB<3-0>. В этом случае нажатие на одну из клавиш будет вызывать изменение потенциала на соответствующей шине RB<7-4> с уровня логической единицы на уровень логического нуля и, как следствие, – прерывание микроконтроллера.

Будем предполагать, что микроконтроллер управляет и другими периферийными объектами кроме клавиатуры. Поэтому для создания эффекта квазипараллельного обслуживания периферийных объектов используем режим разделения времени по флагам состояния системы.

Анализ текущего состояния флагов и вызов соответствующих процедур осуществляются в фоновом цикле. Соответствующий алгоритм представлен на рис. 25. Фоновый цикл начинается после инициализации системы и включает в себя три раздела: сканирование, анализ флагов и зону прерываний.

Сканирование заключается в выводе в каждом очередном цикле сигнала нулевого уровня на соответствующую шину RB<3-0> и подготовке кода сканирования для следующего цикла. Соответственно, код сканирования 0EEh формирует 0 на шине RB0, а коды 0DDh, 0BBh и 077h позволяют обнулять шины RB1, RB2 и RB3. Для обеспечения подобного закона формирования кодов сканирования достаточно сдвигать циклически влево байт 0EEh в каждом очередном фоновом цикле.

В свою очередь, при формировании кода номера нажатой клавиши потребуется знать номер позиции нуля в коде сканирования на момент нажатия. Начальное значение счётчика позиции нуля для кода 0EEh равно 00b, а для кодов 0DDh, 0BBh и 077h – 01b, 10b и 11b, что соответствует счёту по mod4.

В разделе анализа флагов проверяются три флага разделения времени: f0, f1 и f2. Флаг f0 = 1, если произошло нажатие на одну из клавиш, флаг f1

устанавливается в 1 по истечении времени дребезга контактов, а f3 = 1 по истечении времени реакции оператора от момента нажатия на клавишу. Начальное состояние флагов нулевое. Единичное значение флагов f0, f1 и f2 активизирует вызов соответствующей процедуры.

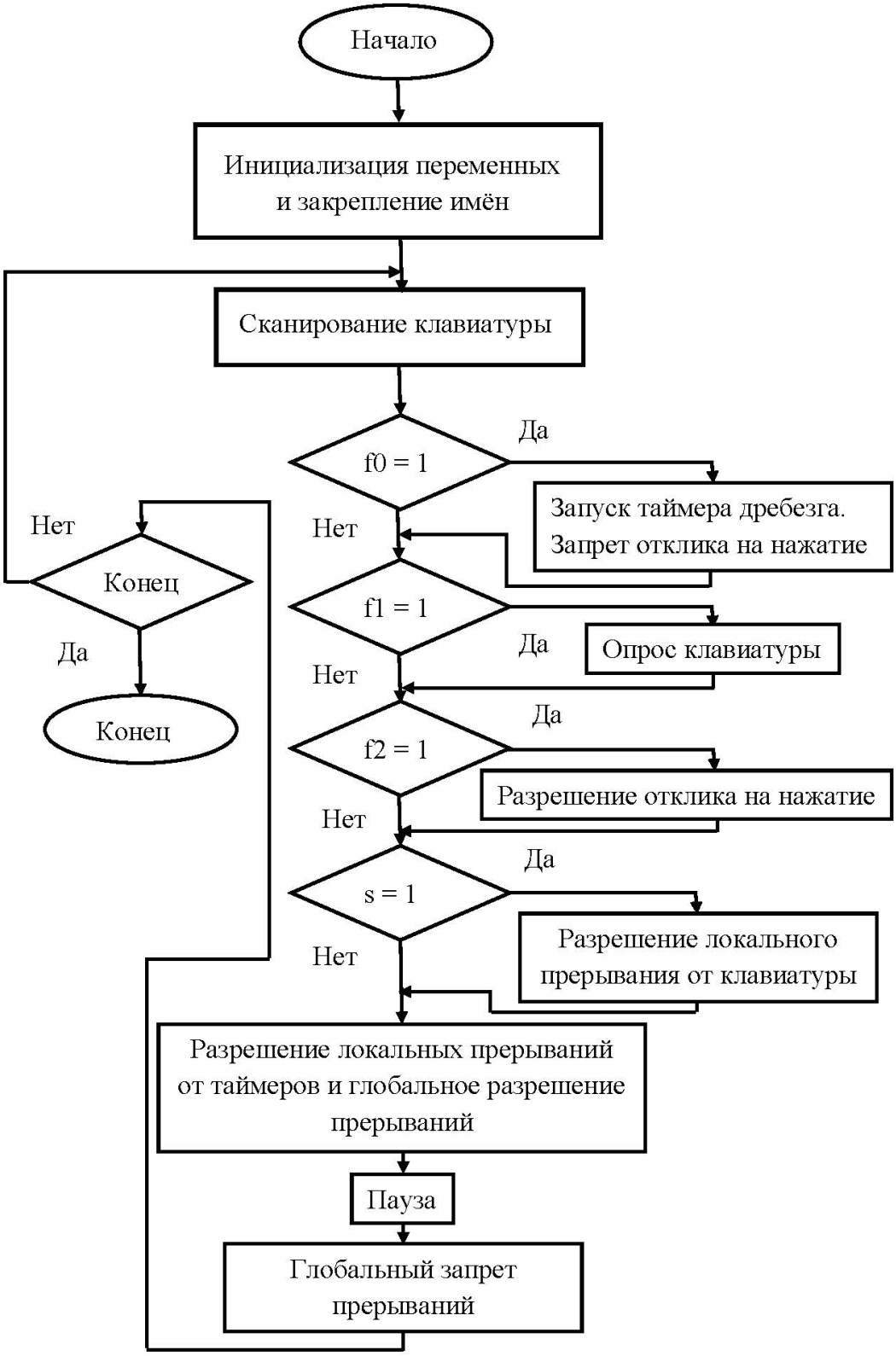

Рис. 25

В разделе зоны прерываний разрешаются или запрещаются локальные и глобальное прерывания. К локальным относятся прерывания: по нажатию на клавишу, по срабатыванию таймера дребезга и по срабатыванию таймера реакции оператора. Глобальное разрешение прерываний позволяет реагировать на все разрешённые запросы локальных прерываний, а глобальный запрет прерываний отключает реакцию на все запросы прерываний.

В рассматриваемом алгоритме в фоновый цикл введена вершина анализа признака **s**, единичное значение которого позволяет разрешать локальное прерывание по нажатию на клавишу, а нулевое значение – запрещать. Подобная организация зоны прерываний позволяет реагировать на запрос по нажатию при устойчивом состоянии кода сканирования и счётчика позиции нуля в коде сканирования. Признак **s** устанавливается первоначально при инициализации, сбрасывается после реакции на нажатие и вновь устанавливается после срабатывания таймера реакции оператора, разрешая реакцию на очередное нажатие.

Алгоритмы процедур прерывания приведены на рис. 26.

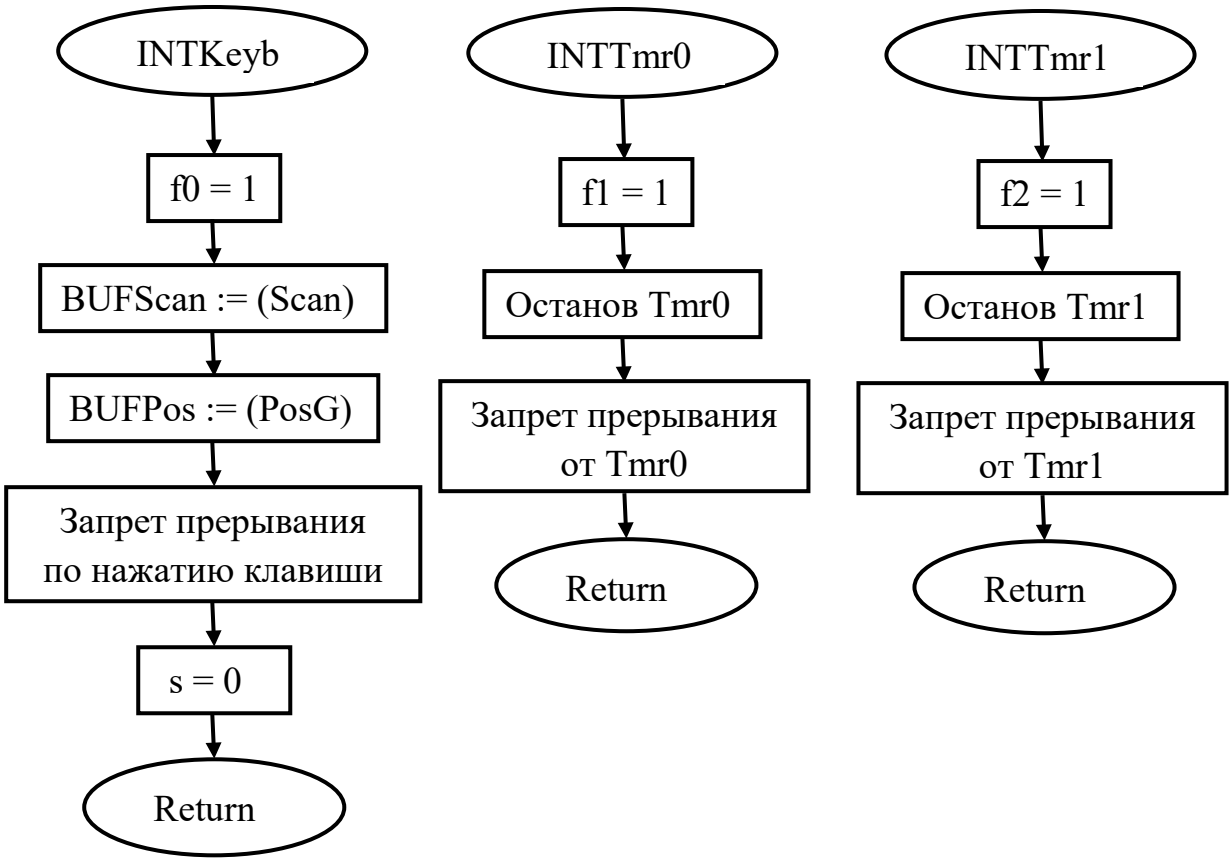

Рис. 26

Процедура INTKeyb вызывается по запросу прерывания при нажатии на клавишу. В процедуре INTKeyb устанавливается флаг f0, сохраняются код сканирования Scan в буфере BUFScan, счётчик позиции нуля в коде сканирования PosG в буфере BUFPos на момент прерывания по нажатию на клавишу, запрещается реакция на повторные нажатия и сбрасывается признак **s**.

Процедура INTTmr0 вызывается по переполнению таймера дребезга, что говорит о завершении выдержки времени ожидания успокоения контактной пары нажатой клавиши. В процедуре устанавливается флаг f1, останавливается таймер Tmr0, запрещаются повторные прерывания от Tmr0.

Процедура INTTmr1 вызывается по переполнению таймера Tmr1, что говорит о завершении выдержки времени ожидания реакции оператора (защита от повторного ввода). В процедуре устанавливается флаг f2, останавливается таймер Tmr1, запрещаются повторные прерывания от Tmr1.

Процедура Timstr запуска таймера дребезга, вызываемая по флагу f0, представлена на рис. 27. В этой процедуре выполняется загрузка таймера Tmr0 на стандартную выдержку времени в 20 мс, сбрасывается процессорный флаг RBIF, устанавливаемый по прерыванию, вызываемому изменением сигналов на шинах RB<7-4> при нажатии на клавишу, запускается Tmr0 и сбрасывается флаг f0 вызова данной процедуры из фонового цикла.

Процедура Keyinp опроса клавиатуры, вызываемая по флагу f1, приведена на рис. 28. Флаг f1 устанавливается в процедуре INTTmr0 и говорит о завершении выдержки времени дребезга контактной пары нажатой клавиши и о возможности считывания устойчивых выходных сигналов на шинах RB<7-4>.

В начале процедуры Keyinp осуществляется вывод на шины RB<3-0> хранимого в BUFScan кода сканирования, зафиксированного на момент прерывания. Далее с шин RB<7-4> считывается ответный код клавиатуры в регистр WReg. В ответном коде содержится 0 в одном из битов, который соответствует вертикальной шине матрицы с нажатой клавишей.

Для формирования двоичного кода номера клавиши необходимо определить позицию нуля в ответном коде. С этой целью организуется цикл сдвига вправо содержимого WReg и приращение счётчика PosV до появления нуля на выходе сдвига. Искомый код клавиши CodK формируется из сдвинутого влево на два разряда содержимого счётчика позиции нуля в коде сканирования, хранимого в BUFPos, и полученного содержимого счётчика PosV. Шестнадцать клавиш клавиатуры 4 × 4 пронумерованы шестнадцатеричными цифрами от 0 до Fh, десять из которых от 0 до 9 рассматриваются как клавиши вводимых чисел, а остальные – от Ah до Fh – как командные клавиши. В дальнейшем двоичные коды вводимых чисел размещаются в буфере данных по текущим адресам ADDR, состоящим из базы буфера данных BD и смещения Off. После ввода текущего CodK смещение инкрементируется и сворачивается по требуемому модулю в зависимости от глубины буфера данных (на рис. 28 – по mod4) для подготовки к последующему вводу данных. Каждый очередной ввод данных сопровождается установкой флага данных fd.

Если была нажата клавиша команды, то CodK заносится в регистр команды ComR и выставляется флаг команды fc. Любой ввод заканчивается сбросом флага f1, после чего запускается таймер TMR1 выдержки времени реакции оператора.

TMR1 отсчитывает время реакции оператора, т. е. время, необходимое для отпускания нажатой клавиши, после чего разрешается очередное нажатие на требуемую клавишу. Переполнение TMR1 вызывает установку f2 (см. рис. 26), по которому из фонового цикла вызывается процедура Reakc (см. рис. 29). В этой процедуре устанавливается признак **s**, разрешающий в фоновом цикле локальное прерывание по нажатию клавиши (см. рис. 25), и сбрасывается флаг вызова f2.

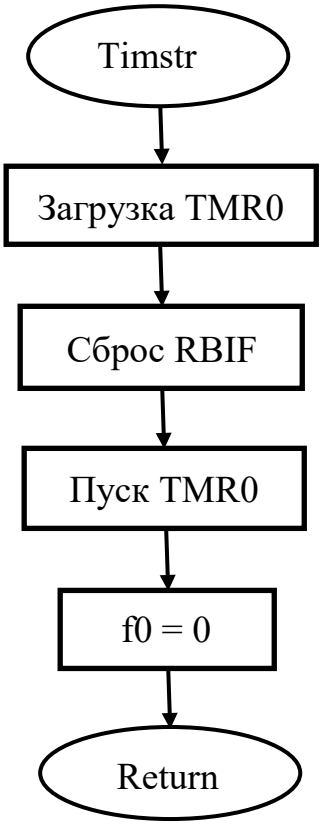

Рис. 27

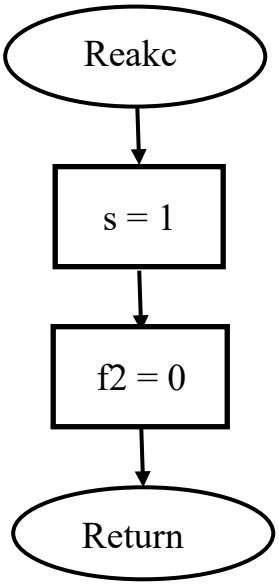

Рис. 29

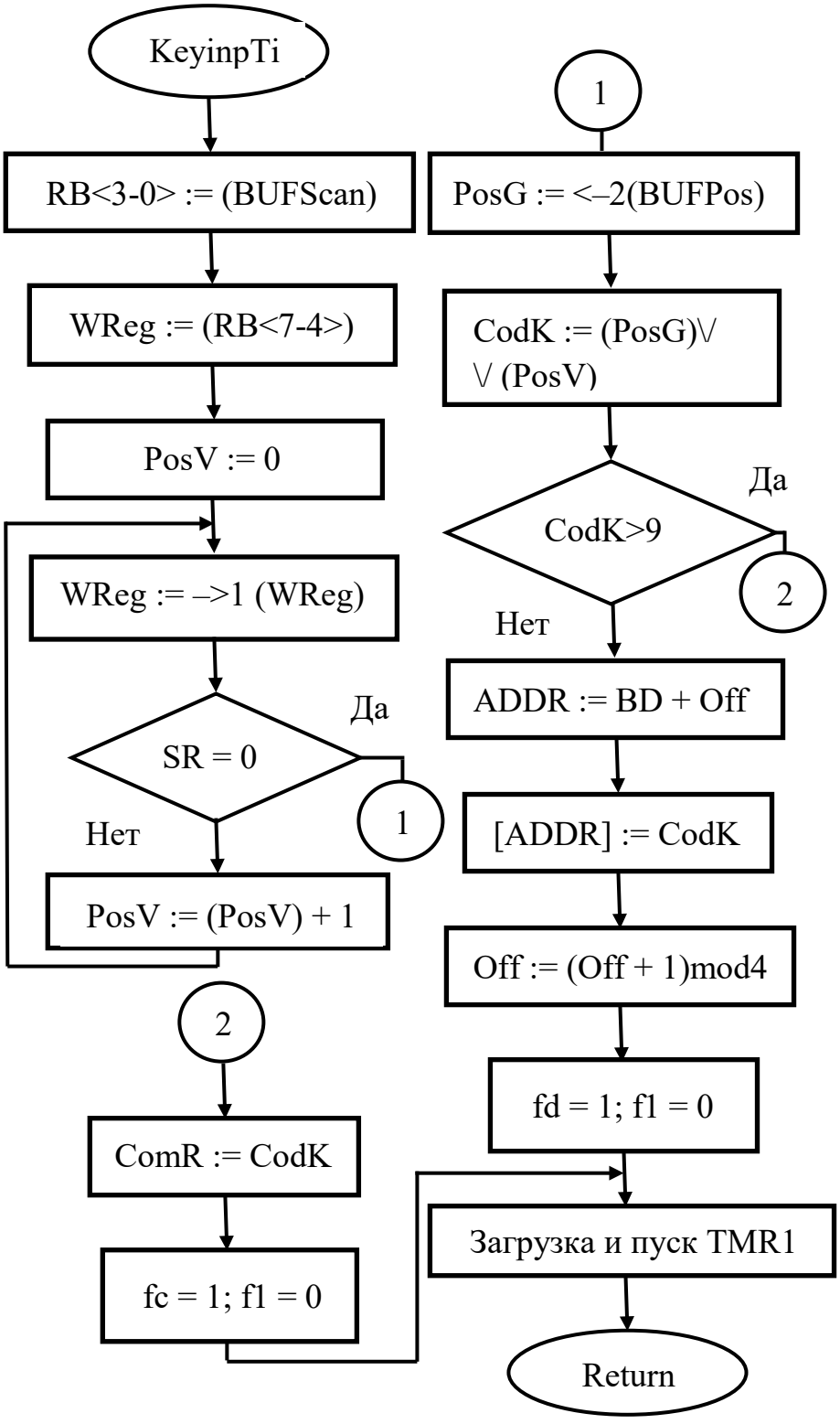

Рис. 28

Процедурой Reakc завершается процесс ввода символа с клавиатуры. Ассемблерная программа для микроконтроллера PIC16C74A, соответствующая рассмотренному алгоритму, приведена ниже.

```
        #include "P16C74A.inc" ; подключение .inc-файла
                       ; закрепление символьных имён
scan equ 0x20          ; регистр кода сканирования
poscnt equ 0x21        ; счётчик позиции нуля в коде сканирования
stk equ 0x22           ; регистр временного хранения
keydat equ 0x23        ; регистр ответного кода клавиатур
vertcnt equ 0x24       ; счётчик позиции нуля в ответном коде
flag equ 0x25          ; регистр флагов и признаков фонового цикла
codkey equ 0x26        ; регистр кода номера клавиши
bufscan equ 0x27       ; буфер кода сканирования
bufpos equ 0x28        ; буфер позиции нуля в коде сканирования
base equ 0x30          ; база буфера данных
flagdan equ 0x34       ; регистр флага данных
comreg equ 0x35        ; регистр кода команды
flagcom equ 0x36       ; регистр флага команды
        Инициализация среды процессора.
        ORG 0x0
                goto Start
        ORG 0x4
                goto PRER
        ORG 0x5
        Start:
                bsf    STATUS, 0x5      ; выбор банка 1
                movlw  0x0              ; очистка WREG
                movwf  INTCON           ; запрет прерываний
                movwf  TRISB            ; настройка RB<7:0>  на вывод
                bcf    STATUS, 0x5      ; выбор банка 0
                movlw  0x0
                movwf  PORTB            ; очистка защёлки PORTB
                movwf  flag             ; сброс регистра флагов          0x25
                bsf   STATUS, 0x5       ; выбор банка 1
                movlw  0xf0
                movwf  TRISB            ; настройка RB<3:0>  на вывод
                                        ; RB<7:4> на ввод
```

```
        bcf   STATUS, 0x5       ; выбор банка 0
        movlw  0xEE
        bsf  STATUS,0x0
        movwf   scan        ; инициализация кода сканирования       0x20
        movlw  base         ; загрузка базы буфера данных
        movwf  FSR          ; в регистр косвенной адресации
        movlw  0x0
        movwf  poscnt       ; сброс счётчика позиции                0x21
        bsf  flag,0x7       ; установка признака разрешения локального
                            ; прерывания по нажатию клавиши
; Фоновый цикл
; Состоит из разделов: сканирование, анализ флагов разделения
; времени, зона разрешения прерываний.

;Сканирование.
cikl:                   ;фоновый цикл
        rlf  scan,w         ; циклический сдвиг кода сканирования
                            ; через флаг CF                         0x20
        btfsc  STATUS,0x0
        goto  one
        movwf  scan; 0x20
        bcf scan, 0x0       ; сброс 0-бита кода сканирования        0x20
m1:
        bcf  STATUS, 0x5; выбор банка 0
        movf  scan,w        ; сохранение кода сканирования          0x20
        movwf  PORTB        ; вывод кода сканирования
        incf  poscnt        ; инкремент счётчика позиции            0x21
        movlw 0x3
        andwf  poscnt       ; свёртка по mod4                       0x21
                            ; завершение цикла сканирования

; Анализ флагов разделения времени.
        btfsc  flag,0x0     ; анализ флага нажатия клавиши          0x25
        goto  timstr
        btfsc  flag,0x1     ; анализ флага прерывания таймера 0     0x25
        goto  keyinp
        btfsc  flag,0x2     ; анализ флага прерывания таймера 1     0x25
        goto  reakc
        goto  inten
```

```
one:
        movwf  scan
        bsf  scan,0x0        ; установка 0-бита кода сканирования          0x20
        goto  m1

        ; Зона разрешения прерываний.
inten:
        btfsc  flag,0x7      ; анализ признака разрешения реакции на нажатие
        bsf  INTCON,0x3 ; разрешение локального прерывания от клавиатуры
        bsf  INTCON,0x5 ; разрешение локального прерывания таймера 0
        bsf  INTCON,0x6 ; разрешение периферийного прерывания от таймера 1
        bsf  INTCON,0x7 ; глобальное разрешение прерываний
        nop
        nop
        nop
        movlw 0x0
        movwf  INTCON ; запрещение глобального, локального и
        goto  cikl ; периферийного прерываний от клавиатуры и таймеров

        ; Вызов процедур прерывания.
ORG 0x60
PRER:                ; анализ запросов прерывания
        bsf  STATUS,0x5 ; выбор банка 1
        btfsc  PIR1, 0x0
        goto  tim1
        bcf  STATUS,0x5 ; выбор банка 0
        btfsc  INTCON, 0x0
        goto  keyb
        goto  tim0

        ; Процедуры прерывания.
keyb:                ; процедура прерывания по нажатию клавиши
        bcf  INTCON,0x3 ; запрет локального прерывания от клавиатуры
        bsf  flag, 0x0       ; установка флага нажатия клавиши          0x25
        movf  scan,w         ; 0x20
        movwf  bufscan       ; запись в буфер кода сканирования          0x27
        movf  poscnt,w       ; 0x21
        movwf  bufpos        ; запись в буфер счётчика позиции          0x28
        bcf  flag, 0x7       ; сброс признака разрешения реакции на нажатие
```

```
        retfie                  ; возврат из прерывания
tim0:           ; процедура прерывания по переполнению таймера  дребезга
        bcf  INTCON,0x2 ; сброс флага прерывания от таймера 0
        bcf  INTCON,0x5 ;  запрет локального прерывания от таймера 0
        bsf  flag,0x1           ; установка флага срабатывания таймера 0
        bcf  OPTION_REG,0x5  ; останов таймера 0
        retfie                  ; возврат из прерывания
tim1:  ; процедура прерывания по переполнению таймера реакции оператора
        bcf  STATUS,0x5 ; выбор банка 0
        bcf  PIR1,0x0           ; сброс флага прерывания таймера 1
        bcf  T1CON, 0x0  ; останов таймера 1
        bsf  flag, 0x2          ; установка флага срабатывания таймера 1
        retfie                  ; возврат из прерывания

        ; Процедура запуска таймера дребезга.
timstr:
        movlw  0xf0
        movwf  TMR0             ; загрузка таймера 0
        bcf  INTCON,0x0         ; сброс флага RBIF прерывания от клавиатуры
        bsf  STATUS,0x5         ; выбор банка 1
        movlw 0x80
        movwf  OPTION_REG  ; режим таймера 0, запуск
        bcf  STATUS,0x5         ; выбор банка 0
        bcf  flag,0x0           ; сброс флага нажатия клавиши
        goto  inten

        ; Процедура опроса клавиатуры.
keyinp:
        movwf  stk              ; сохранение  wreg                      0x22
        bsf    STATUS, 0x5      ; выбор банка 0
        movf  bufscan,w                                                 ; 0x27
        movwf  PORTB            ; вывод кода сканирования
        nop
        nop
        nop
        movf  PORTB,w           ; ввод ответного кода клавиатуры
        andlw  0xf0             ; выделение старшей тетрады
        movwf  keydat                                                   ; 0x23
        swapf  keydat           ; смена тетрад ответного кода
```

```
        clrf  vertcnt           ; очистка счётчика позиции вертикали      0x24
shift:
        rrf  keydat             ; сдвиг ответного кода                    0x23
        btfsc STATUS,0x0        ; анализ выхода сдвига
        goto  two
        incf  vertcnt           ; инкремент счётчика позиции вертикали    0x24
        goto  shift
two:
        rlf  bufpos             ; формирование кода клавиши               0x28
        rlf  bufpos
        movf  bufpos,w
        andlw  0xfc
        iorwf  vertcnt,w                                                  ; 0x24
        movwf  codkey           ; сохранение кода клавиши                 0x26
        movf  stk,w             ; восстановление wreg                     0x22
        bcf  flag,1              ; сброс флага f1                         0x25
        goto  bufdat

bufdat:
        movlw  0xa
        subwf  codkey,w         ; сравнение с 10 0x26
        movwf  0x37
        rlf  0x37
        btfss  STATUS,0x0
        goto  comm
        movf  codkey,w                                                    ; 0x26
        movwf  INDF                     ; запись в буфер данных
        incf  FSR           ; инкремент указателя косвенной адресации
        movf  FSR,w
        andlw  0x33         ; свёртка указателя косвенной адресации  по mod4
        movwf  FSR
        bsf  flagdan,0x0    ; установка флага данных                    0x34
        goto  opt
comm:
        movf  codkey,w                                                    ; 0x26
        movwf  comreg           ; запись в регистр команды                0x35
        bsf  flagcom,0x0        ; установка флага команды                 0x36
opt:
        bcf  STATUS, 0x5; выбор банка 0
```

```
movlw  0xff        ; загрузка таймера 1
movwf  TMR1H
movlw  0xf0
movwf  TMR1L
bsf  T1CON, 0x2
bsf  T1CON, 0x0          ; пуск таймера 1
bsf   STATUS, 0x5        ; выбор банка 1
bsf  PIE1, 0x0           ; разрешение прерывания таймера 1
goto  inten

; Процедура окончания реакции оператора
reakc:
bcf  flag, 0x2      ;сброс флага f2                              0x25
bsf  flag, 0x7      ; установка признака разрешения реакции на нажатие
goto  inten
```

При составлении программы учитывались следующие особенности архитектуры микроконтроллера PIC16C74A:

1. Регистры специальных функций SFR и регистры общего назначения GPR размещаются в двух банках (рис. 30).

2. Некоторые регистры доступны в обоих банках, например INDF, STATUS и др.

3. Большинство регистров закреплены за конкретными банками, например OPTION, PORTA и др.

Данные особенности должны учитываться при программировании. Обращение к регистру, находящемуся в другом банке, а не в текущем, вызовет обращение к регистру текущего банка по смежному адресу. Так, если вы находитесь в банке 0 и обращаетесь к регистру OPTION, то реальные операции будут осуществляться с регистром TMR0.

Учитывая вышесказанное, перед обращением к регистру OPTION необходимо изменить биты кодировки RP1и RP0 в регистре STATUS (рис. 31), которые содержат код номера банка: 00 – банк 0; 01 – банк 1; 10 – банк 2; 11 – банк 3. Так как в PIC16C74A имеется только два банка, то выбираются кодировки 00 или 01. При включении или при сбросе микроконтроллера эти биты обнуляются, то есть идет обращение к банку 0.

Bank 0		Bank 1	
File Address	SFR	File Address	SFR
00h	INDF	80h	INDF
01h	TMR0	81h	OPTION
02h	PCL	82h	PCL
03h	STATUS	83h	STATUS
04h	FSR	84h	FSR
05h	PORTA	85h	TRISA
06h	PORTB	86h	TRISB
07h	PORTC	87h	TRISC
08h	PORTD	88h	TRISD
09h	PORTE	89h	TRISE
0Ah	PCLATH	8Ah	PCLATH
0Bh	INTCON	8Bh	INTCON
0Ch	PIR1	8Ch	PIE1
0Dh	PIR2	8Dh	PIE2
0Eh	TMR1L	8Eh	PCON
0Fh	TMR1H	8Fh	
10h	T1CON	90h	
11h	TMR2	91h	
12h	T2CON	92h	PR2
13h	SSPBUF	93h	SSPADD
14h	SSPCON	94h	SSPSTAT
15h	CCPR1L	95h	
16h	CCPR1H	96h	
17h	CCP1CON	97h	
18h	RCSTA	98h	TXSTA
19h	TXREG	99h	SPBRG
1Ah	RCREG	9Ah	
1Bh	CCPR2L	9Bh	
1Ch	CCPR2H	9Ch	
1Dh	CCP2CON	9Dh	
1Eh	ADRES	9Eh	
1Fh	ADCON0	9Fh	ADCON1
20h … 7Fh	General Purpose Register (GPR) Bank 0	A0h … FFh	General Purpose Register (GPR) Bank 1

Рис. 30

IRP	RP1	RP0	!TO	!PD	ZF	DCF	CF

Рис. 31

В приведённой выше программе выбор банка 1 осуществляется по команде «bsf STATUS, 0x5», устанавливающей бит RP0 в единицу, а выбор банка 0 – по команде «bcf STATUS, 0x5», сбрасывающей бит RP0 в ноль. В начале программы директивами «ORG 0x0» и «ORG 0x4» заданы

соответственно вектор пуска и вектор прерывания. На эти векторы микроконтроллер переходит автоматически соответственно при сбросе и при любом прерывании. По вектору 0x0 размещена команда «goto Start» перехода на фрагмент инициализации среды процессора, а по вектору 0x4 – команда перехода «goto PRER» на блок анализа флагов запросов от источников прерывания.

При инициализации среды процессора запрещаются прерывания путём сброса регистра INTCON, сбрасывается регистр флагов flag, PORTB потетрадно настраивается на вывод по шинам RB<3:0> и на ввод по шинам RB<7:4>, загружается регистр кода сканирования scan кодом 0EEh и счётчик позиции нуля в коде сканирования poscnt соответственно сбрасывается в ноль, загружается база буфера данных base в регистр косвенной адресации FSR и устанавливается признак разрешения локального прерывания по нажатию клавиши.

В системе прерывания PIC16C74A имеется только один вектор прерывания 0x4, поэтому при любом прерывании необходимо выполнить анализ состояния флагов запросов прерывания от возможных источников. В данном случае возможны прерывания: по нажатию клавиши (изменение сигналов на шинах RB<7-4>), по переполнению таймера TMR0 выдержки времени дребезга контактной пары клавиши, по переполнению таймера TMR1 выдержки времени реакции оператора.

Два первых из перечисленных запросов прерывания входят в основную группу запросов и анализируются по соответствующим флагам RBIF и T0IF регистра INTCON (рис. 32).

GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

Рис. 32

Запрос прерывания от TMR1 относится к периферийным запросам, и соответствующий флаг TMR1IF размещается в регистре периферийных прерываний PIR1 (рис. 33). Поэтому в процедуре PRER анализируются соответствующие флаги.

–	ADIF	–	–	SSPIF	CCP1IF	TMR2IF	TMR1IF

Рис. 33

Просмотр флагов начинается командой «btfsc PIR1, 0x0» с переходом на метку tim1 при установленном флаге TMR1IF. В противном случае выполняется команда «btfsc INTCON, 0x0» с переходом на метку keyb при установленном флаге RBIF, иначе с переходом на метку tim0 вызывается прерывающая процедура, соответствующая оставшемуся запросу по флагу T0IF от TMR0.

В процедуре keyb командой «bcf INTCON,0x3» сбрасывается бит RBIE разрешения локального прерывания по нажатию клавиши, что запрещает данное прерывание при дальнейших нажатиях на клавишу. Командой «bsf flag, 0x0» устанавливается фоновый флаг f0. Далее командами «movf scan,w; movwf bufscan; movf poscnt,w; movwf bufpos» описывается сохранение кода сканирования и позиции нуля в коде сканирования на момент нажатия на клавишу. В конце процедуры по команде «bcf flag, 0x7» сбрасывается фоновый признак s, запрещающий реакцию на нажатие клавиши, и по команде «retfie» осуществляется возврат из прерывания.

В процедуре tim0 командами «bcf INTCON,0x2; bcf INTCON,0x5» сбрасывается флаг запроса прерывания T0IF от таймера 0 и запрещается локальное прерывание от таймера 0 сбросом бита T0IE в регистре INTCON.

Далее командой «bsf flag,0x1» устанавливается фоновый флаг f1, командой «bcf OPTION_REG,0x5» останавливается таймер 0 и осуществляется возврат из прерывания.

В процедуре tim1командами «bcf PIR1,0x0; bcf T1CON, 0x0; bsf flag, 0x2; retfie» осуществляется сброс флага запроса прерывания TMR1IF от таймера 1 в регистре PIR1, останов таймера 1 путём сброса бита TMR1ON в регистре T1CON (рис. 34), установка фонового флага f2 и возврат из прерывания.

–	–	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON

Рис. 34

Раздел «*Сканирование»* фонового цикла включает фрагменты под метками cikl, m1, one. В данном разделе осуществляется циклический ле-

вый сдвиг кода сканирования (см. рис. 25). Особенностью реализации операции сдвига является то, что в системе команд PIC16C74A отсутствует команда непосредственно циклического сдвига, а подобный сдвиг выполняется через процессорный флаг CF. В результате вместо 8-разрядного циклического сдвига выполняется 9-разрядный циклический сдвиг, через который требуется имитировать 8-разрядный циклический сдвиг, что существенно усложняет данную процедуру.

Процесс сдвига производится следующим образом. Группой команд «rlf scan,w ; btfsc STATUS,0x0; goto one; movwf scan; bcf scan, 0x0» выполняется циклический левый сдвиг кода сканирования через флаг CF, анализируется состояние CF в регистре STATUS с переходом на метку one при CF = 1 либо с сохранением сдвинутого кода сканирования в регистре scan и сбросом в 0 младшего бита регистра при CF = 0. При переходе на метку one группой команд «movwf scan; bsf scan,0x0; goto m1» также сохраняется сдвинутый код сканирования в регистре scan, но с установкой в 1 младшего бита регистра.

Далее по метке m1 группой команд «bcf STATUS, 0x5; movf scan,w; movwf PORTB; incf poscnt; movlw 0x3; andwf poscnt» производится пересылка кода сканирования из регистра scan на шины RB<3:0> PORTB и наращивание счётчика poscnt со свёрткой по mod4. На этом очередной цикл сканирования заканчивается, и начинается раздел «*Анализ флагов разделения времени*».

В этом разделе анализируются фоновые флаги f0, f1и f2 регистра flag. Единичное значение любого из флагов вызывает переход на соответствующую процедуру. При f0 = 1 осуществляется переход на процедуру timstr запуска таймера дребезга. При f1 = 1 производится переход на процедуру keyinp опроса клавиатуры, а при f2 = 1 выполняется переход на процедуру reakc окончания реакции оператора.

В процедуре timstr группой команд «movlw 0xf0; movwf TMR0; bcf INTCON,0x0; bsf STATUS,0x5; movlw 0x80; movwf OPTION_REG; bcf STATUS,0x5; bcf flag,0x0; goto inten» производится загрузка таймера TMR0 на выдержку 20 мс – времени дребезга контактной пары (конкретный код загрузки определяется с учётом рабочей частоты микроконтрол-

лера), осуществляется сброс флага RBIF прерывания от клавиатуры в регистре INTCON, загружается код режима работы и запуска таймера TMR0 в регистр OPTION_REG, сбрасывается флаг f0 вызова данной процедуры и выполняется переход на метку inten начала раздела «*Зона разрешения прерываний*».

В процедуре keyinp командами «movf bufscan,w; movwf PORTB» из буфера кода сканирования bufscan на шины RB<3:0> выводится код сканирования на момент прерывания по нажатию клавиши, что позволяет ввести по шинам RB<7:4> ответный код с вертикальных шин клавиатуры. Ввод осуществляется по команде «movf PORTB,w». Далее по командам «andlw 0xf0; movwf keydat; swapf keydat» выделяется старшая тетрада ответного кода, соответствующая шинам RB<7:4>, она заносится в регистр ответного кода keydat, где перемещается на место младшей тетрады.

Перед началом цикла shift изменения позиции нуля в ответном коде командой «clrf vertcnt» обнуляется счётчик позиции нуля в ответном коде. Цикл реализуется группой команд «rrf keydat; btfsc STATUS,0x0; goto two; incf vertcnt; goto shift». В ходе цикла выполняется правый сдвиг содержимого регистра keydat с последующим анализом сигнала на выходе сдвига, фиксируемого во флаге CF регистра STATUS. При CF = 0 осуществляется переход на метку two, в противном случае инкрементируется счётчик vertcnt с возвратом на начало цикла по метке shift.

С метки two начинается формирование кода codkey номера нажатой клавиши. Для этого командами «rlf bufpos; rlf bufpos; movf bufpos,w; ANDLW 0xfc» организуется левый сдвиг на два разряда хранимого в буфере bufpos кода позиции нуля в коде сканирования на момент нажатия клавиши. После чего сбрасываются в 0 два младших разряда буфера bufpos. В эти два разряда командой «iorwf vertcnt,w» вставляется содержимое счётчика vertcnt, полученное в предшествующем цикле shift. Сформированный код номера клавиши командой «movwf codkey» загружается в регистр codkey. Процедура завершается сбросом фонового флага f1 и переходом на метку bufdat.

Полученный код номера клавиши рассматривается как десятичная цифра в диапазоне 0…9 либо как код команды в диапазоне 0Ah…0Fh. Подобное разбиение выполняется в разделе под меткой bufdat. Командами «movlw 0xa; subwf codkey,w» производится неразрушающее вычитание

из содержимого codkey числа 0Ah, результат операции командами «movwf temp; rlf temp» сохраняется в регистре temp и сдвигается влево для загрузки знака во флаг CF. По анализу флага CF регистра STATUS командой «btfss STATUS,0x0» выполняется переход «goto comm» на сохранение кода номера командной клавиши либо клавиша классифицируется как цифровая и код её номера заносится в буфер данных.

Сохранение в буфере данных реализуется с использованием косвенной адресации командами «movf codkey,w; movwf INDF», в которых выполняется обращение к регистру INDF, что автоматически выбирает текущий адрес буфера из регистра косвенной адресации FSR. Начальное содержимое FSR (базовый адрес буфера данных) определялось при инициализации загрузкой в него значения base. После каждой текущей загрузки в буфер данных содержимое FSR инкрементируется командой «incf FSR» и округляется по mod4 относительно исходного значения base командами «movf FSR,w; andlw 0x33;mod4; movwf FSR» исходя из глубины буфера, равной 4. Запись в буфер данных завершается установкой флага признака введённых данных в регистре flagdan и переходом на метку opt по командам «bsf flagdan,0x0; goto opt».

Если ранее был выполнен переход на метку comm, что говорит о нажатой командной клавише, то по командам «movwf codkey,w; movwf comreg; bsf flagcom,0x0» содержимое регистра codkey заносится в регистр кода команды comreg и в регистре flagcom выставляется флаг признака введённой команды. Далее следует переход на метку opt.

По метке opt размещаются команды «bcf STATUS, 0x5; movlw 0xff; movwf TMR1H; movlw 0xf0; movwf TMR1L» загрузки таймера выдержки времени реакции оператора TMR1, команды «bsf T1CON, 0x2; bsf T1CON, 0x0» установки битов режима и запуска TMR1в управляющем регистре T1CON и команда «bsf PIE1, 0x0» разрешения локального прерывания по переполнению TMR1, устанавливающая бит TMR1IE в регистре PIE1 (рис. 35).

–	ADIE	–	–	SSPIE	CCP1IE	TMR2IE	TMR1IE

Рис. 35

Завершается процедура запуска TMR1 переходом по команде «goto inten» на метку inten раздела «*Зона разрешения прерываний*».

Код загрузки TMR1 определяется из расчёта времени реакции оператора 0,5 секунды и рабочей частоты микроконтроллера. По истечении данного интервала после пуска TMR1 происходит прерывание по переполнению TMR1 с установкой фонового флага f2. По этому флагу вызывается процедура завершения реакции оператора с выполнением перехода на метку reakc.

В процедуре reakc по командам «bcf flag, 0x2; bsf flag, 0x7; goto inten» сбрасывается флаг f2, выставляется фоновый признак s и осуществляется переход к разделу «*Зона разрешения прерываний».*

В разделе «*Зона разрешения прерываний*» выделяется временной интервал для реакции на возникающие запросы прерываний при стабильном состоянии кода сканирования и счётчика позиции нуля в коде сканирования, что позволяет избегать сбойных ситуаций из-за асинхронного появления запросов прерывания.

Управление прерываниями в PIC16C74A осуществляется на трёх уровнях: глобальном, локальном и периферийном. Установка или сброс бита GIE в регистре INTCON (см. рис. 32) определяют соответственно глобальное разрешение или запрет режима прерывания в системе. При GIE = 1 разрешена реакция на локальные запросы прерываний. К локальным запросам относятся: прерывание по входу RB0, прерывание по изменению сигналов на входах RB<7:4>, прерывание по запросу TMR0 и прерывание по общему запросу от периферии. Управление источниками локальных запросов осуществляется соответственно битами INTE, RBIE, T0IE и PEIE регистра INTCON. Установка или сброс этих битов разрешает или запрещает реакцию на соответствующие локальные запросы прерываний. На третьем уровне при PEIE = 1 (общее разрешение периферийных прерываний) управление отдельными периферийными запросами прерываний осуществляется через управляющие биты регистров PIE1 (см. рис. 35) и PIE2. В рассматриваемом примере работы с клавиатурой для разрешения прерывания по запросу TMR1 в регистре необходимо установить бит TMR1IE.

В разделе «*Зона разрешения прерываний*» кроме перечисленных управляющих битов микроконтроллера используется программно-управляемый фоновый признак s, установка которого позволяет реагировать на разрешённый локальный запрос по нажатию клавиши, а сброс – блокировать локальное разрешение до окончания обработки текущего нажатия, что не допускает многократного ввода с клавиатуры в течение 0,5 секунды.

Команда «btfsc flag,0x7» в начале раздела позволяет пропускать следующую команду «bsf INTCON,0x3» разрешения локального прерывания по нажатию клавиатуры, если s = 0. В противном случае данная команда выполняется. Следующие две команды «bsf INTCON,0x5; bsf INTCON,0x6» разрешают локальные прерывания от TMR0 и от периферии, в частности от TMR1 (бит выбора TMR1IE источника периферийного запроса в регистре PIE1 устанавливается при запуске TMR1 в процедуре под меткой opt).

Далее следуют три команды «nop; nop; nop» выдержки интервала времени для восприятия запросов прерывания. В конце раздела командами «movlw 0x0; movwf INTCON» обнуляются разрешающие биты регистра INTCON, что приводит к локальным и глобальному запретам прерываний до следующего входа в раздел. Последняя команда раздела «goto cikl» определяет переход на начало фонового цикла.

ЭТАПЫ ОТЛАДКИ ПРОГРАММЫ

На первом этапе через рабочее окно MPLAB 8.6 (см. рис. 1) вызывается мастер проекта (см. рис. 2, 3). Далее в сопровождении мастера определяется тип целевого микроконтроллера (см. рис. 4). В нашем случае необходимо выбрать PIC16C74A. На следующем шаге задаются средства ассемблирования и создания исполняемого файла (см. рис. 5).

На следующем этапе (см. рис. 6) прописывается путь к папке проекта. Папка создаётся до вызова IDE в какой-либо директиве. Далее к проекту подключается (см. рис. 7) ранее созданный в каком-либо текстовом редакторе .asm-файл отлаживаемой программы. В частном случае .asm-файл может быть пустым. На этом создание проекта завершается, о чём информирует выпадающее окошко (см. рис. 8). По кнопке «Готово» открывается окно проекта с запросом о способе размещения файла (см. рис. 9).

Далее открывается рабочее окно проекта (см. рис. 10) с окошком дерева файлов проекта, в котором выбирается .asm-файл программы, отображаемый в соответствующем окошке (см. рис. 11). Файл может быть отредактирован с помощью встроенного текстового редактора или создан заново. При этом необходимо будет сохранить изменённый текст, о чём говорит звёздочка в имени окошка.

На следующем шаге производится предварительное ассемблирование с выводом окошка сообщения о результатах (см. рис. 12). В сообщении будут указаны выявленные ошибки программы, которые необходимо устранить путём редактирования .asm-файла. При отсутствии ошибок выводится соответствующее сообщение и панель инструментов расширяется иконками управления режимами отладки (см. рис. 13).

После запуска режима отладки в рабочее окно необходимо вызвать через соответствующие меню окошки состояния аппаратной среды микроконтроллера (см. рис. 14). Запуск сопровождается повторным ассемблированием и активизацией симулятора IDE. При этом в окошке .asm-файла появляется зелёный курсор, указывающий на текущую строку отлаживаемой программы.

Выбирая иконки режима отладки «Reset», «Step Into», «Step Over», «Step Out», «Run», «Animate», «Halt», «Breakpoints» (см. рис. 15–22), можно управлять ходом выполнения программы и просматривать состояние аппаратной среды.

Для моделирования внешнего запроса прерывания при нажатии на клавишу клавиатуры используется асинхронная стимуляция состояния шин RB<7:4>. Воздействие на шины RB<7:4> возможно после активизации выпадающего окошка асинхронной стимуляции (рис. 37). В качестве примера на этом рисунке в первой строке окошка прописано формирование на шине RB6 импульса низкого уровня в течение трёх командных циклов при щелчке в колонке Fire данной строки, что соответствует нажатию клавиши, подключённой к вертикальной шине матрицы клавиатуры с позицией 10 (см. рис. 24).

Содержимое второй строки этого окошка определяет соответствующий эффект, но на шине RB5. Это эквивалентно нажатию одной из клавиш, подключённых к вертикальной шине матрицы клавиатуры с позицией 01. Так как начальное состояние уровней сигналов на шинах RB<7:4> до нажатия на клавишу является высоким, то подобная стимуляция вызывает прерывание по изменению сигналов на шинах RB<7:4>.

При отладке программы результатом асинхронной стимуляции будет прерывание микроконтроллера, когда выполняется раздел «*Зона разрешения прерываний*» фонового цикла при фоновом признаке s = 1 и выполнена команда «bsf INTCON,0x7», устанавливающая флаг глобального разрешения прерываний.

СТИМУЛИРОВАНИЕ ВНЕШНИХ ВОЗДЕЙСТВИЙ

Для моделирования внешних входных воздействий от клавиатуры используется режим Stimulus. Вызов режима осуществляется через меню Debugger –> Stimulus –> New Workbook (рис. 36). Временная картина процесса стимуляции определяется установкой требуемых параметров в выпадающем окошке Stimulus (рис. 37).

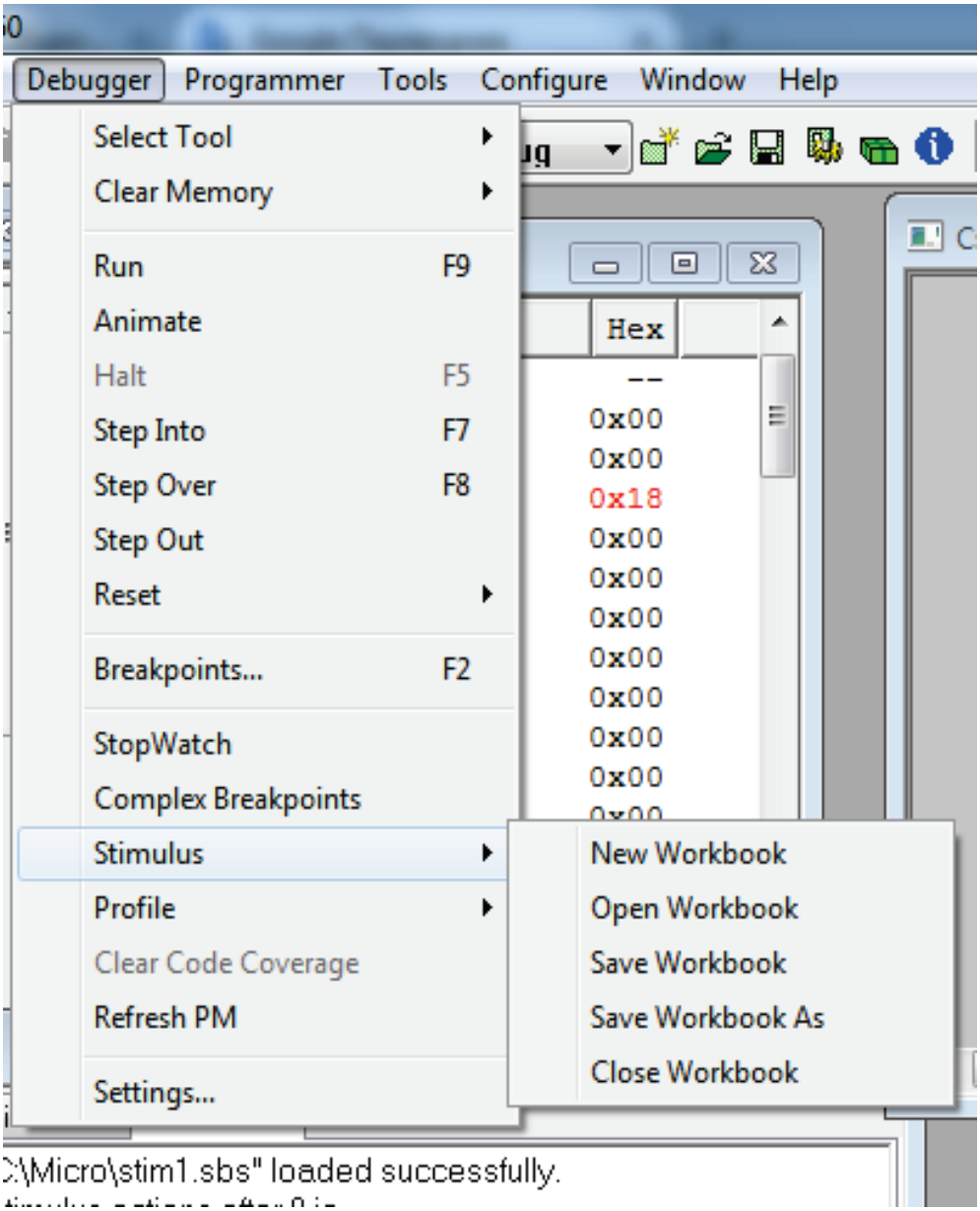

Рис. 36

Асинхронная стимуляция задаётся при нажатии кнопки Asynch. В выпадающей таблице осуществляется построчное заполнение параметров для возможных источников стимуляции.

Заполнение начинается с колонки Pin/SFR текущей строки. Нажатие на соответствующую клетку активизирует список возможных источников, щелчком по одному из которых в клетку заносится имя источника. Так на рис. 37 выбраны шины RB6 и RB5 порта PORTB.

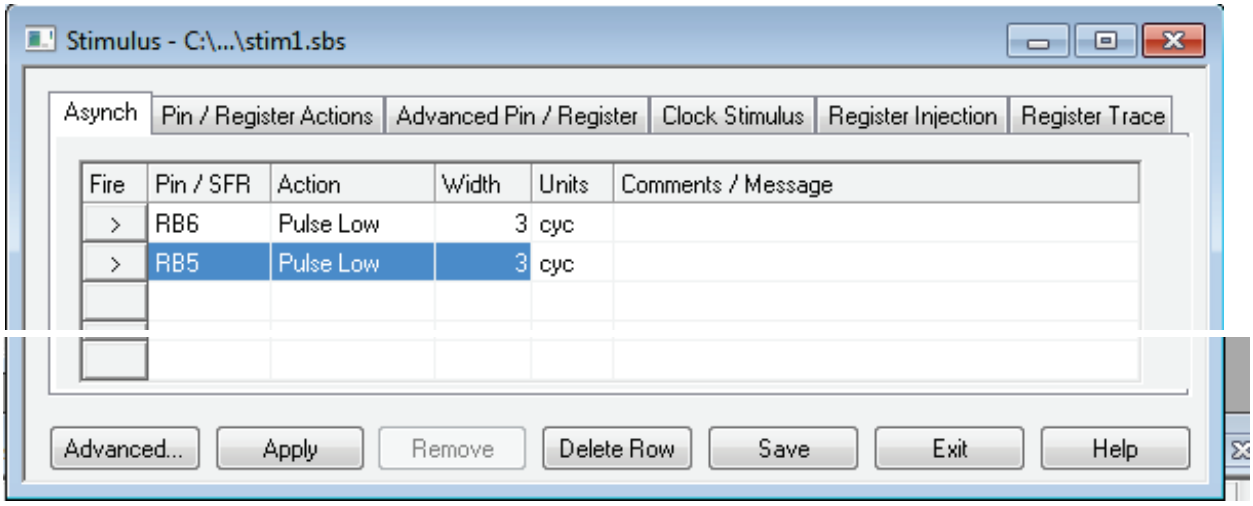

Рис. 37

При нажатии на клетку колонки Action вызывается список вида сигнала стимуляции: Set High (Установить высокий уровень), Set Low (Установить низкий уровень), Toggle (Сменить уровень), Pulse High (Положительный импульс), Pulse Low (Отрицательный импульс). В примере (см. рис. 37) определены отрицательные импульсы на шинах RB6 и RB5.

Нажатие на клетку колонки Width позволяет установить длительность импульса в единицах отсчёта, определяемых в колонке Units, при нажатии на клетку которой выпадает список: **cyc** instruction cycle (командный цикл)**,** **ns** nanosecond (наносекунда), **us** microsecond (микросекунда), **ms** millisecond (миллисекунда), **sec** second (секунда). В примере (см. рис. 37) отрицательные импульсы на шинах RB6 и RB5 длятся по 3 командных цикла.

Активизация асинхронного сигнала на требуемом шаге покомандной отладки производится по нажатию на клетку колонки Fire в соответствующей строке.

При нажатии на кнопку Pin / Register Actions (рис. 38) выпадает таблица, в которой задаются параметры синхронной стимуляции. Заполнение таблицы начинается с первой строки. В колонке Time задаётся номер периода, в котором требуется выполнить стимуляцию. Единица измерения

длительности каждого периода (**cyc** instruction cycle – командный цикл, **h:m:s** Hours:Minutes:Seconds – Часы:Минуты:Секунды, **ms** millisecond – миллисекунда, **us** microsecond – микросекунда, **ns** nanosecond – наносекунда) определяется в окошке Time Units. Далее по нажатию на заголовок колонки Click here to Add Signals выпадает окошко Add/Remove Pin/Registers со списком объектов стимуляции (рис. 39).

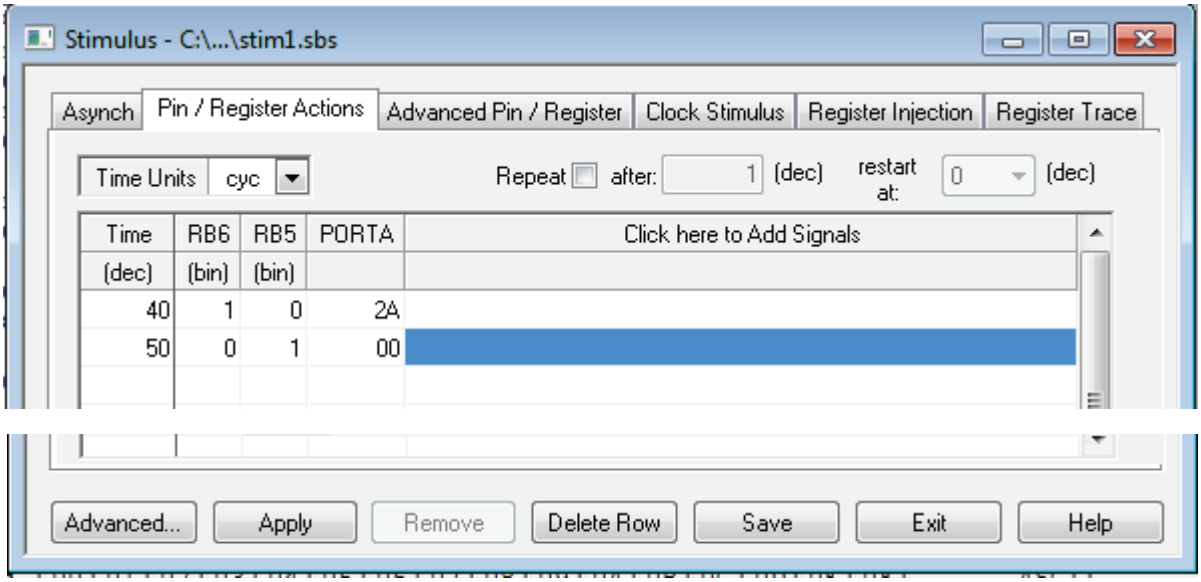

Рис. 38

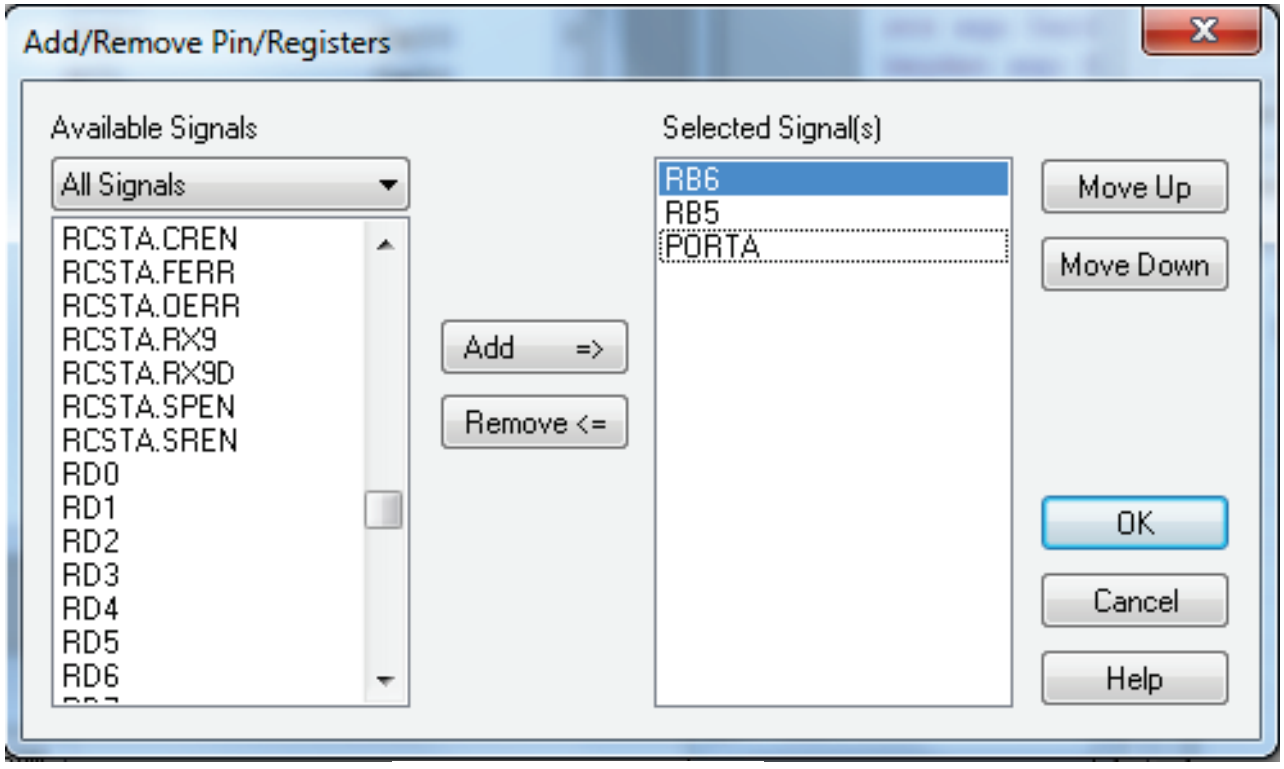

Рис. 39

В качестве объекта может быть шина порта ввода-вывода, внешняя битовая шина, порт ввода-вывода, биты регистров специальных функций, регистры специальных функций. Список объектов может быть отфильтрован по типам объектов при нажатии на кнопку фильтра под заголовком Available Signals. Признаками типов являются: SFR and Bitfield (регистры специальных функций и битовые поля регистров), Pin Onli (только состояние внешних выводов), All Signals (все сигналы). Требуемый объект выбирается щелчком левой кнопки мыши и по нажатию на кнопку Add => заносится в список с заголовком Selected Signal(s), объекты которого выбираются по кнопкам просмотра Move Up и Move Down. Отмеченный объект может быть удалён из списка по кнопке Remove <=. Окошко закрывается по кнопке Ok, а имя каждого объекта списка становится заголовком соответствующей колонки таблицы Pin / Register Actions (см. рис. 38). В качестве примера (см. рис. 39) в список включены биты порта RB6, RB5 и порт ввода-вывода PORTA. Биты в соответствующих клетках строки проставляются в бинарном коде, а содержимое регистров – в шестнадцатеричном коде. В данном примере определено, что на сороковом командном цикле RB6 = 1, RB5 = 0, PORTA = 2Ah, на пятидесятом цикле RB6 = 0, RB5 = 1, PORTA = 00.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Нестерук, В. Ф. Микропроцессорные системы сбора и первичной обработки сигналов : учеб. пособие / В. Ф. Нестерук. – Омск : Изд-во ОмГТУ, 2007. – 136 с.

2. Нестерук, В. Ф. Моделирование периферийного оборудования в интегрированной среде разработки Proteus [Электронный ресурс] : учеб. пособие / В. Ф. Нестерук ; Минобрнауки России, ОмГТУ. – Электрон. текст. дан. (4,59 Мб). – Омск : Изд-во ОмГТУ, 2014. – 1 электрон. опт. диск.