Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра «Электроснабжение промышленных предприятий»

ОСНОВЫ ЦИФРОВОЙ ТЕХНИКИ (ОСНОВЫ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ)

Методические указания
к выполнению лабораторных работ
для студентов специальностей 5В071800 – Электроэнергетика и 5В081200-Энергообеспечение сельского хозяйства

Алматы 2015

СОСТАВИТЕЛИ: М.В.Акименков. Основы цифровой (микропроцессорной) техники. Методические указания к выполнению лабораторных работ для студентов специальностей 5В071800 – Электроэнергетика и 5В081200-Энергообеспечение сельского хозяйства - Алматы: АУЭС, 2015. – 84 с.

Данная разработка включает методические указания по выполнению лабораторных работ по дисциплинам «Основы цифровой техники» и «Основы микропроцессорной техники». В них основное внимание уделено вопросу практического использования цифровой техники в энергетике. Практические примеры базируются на реальных задачах автоматизации процессов управления в энергетике. Во время лабораторных работ студенты осваивают применение микроконтроллеров и технологию разработки для них программ на примере микроконтроллера PIC16F877A.

Илл. – 26, табл. – 23, библиогр. – 11названий.

Рецензент: канд.техн.наук., доц., Гали К.О.

Печатается по плану издания некоммерческого акционерного общества «Алматинский университет энергетики и связи» на 2015 год.

Содержание

Введение

Современное состояние технологии в электроэнергетике требует от специалистов знания микроконтроллеров. В настоящее время в составе выпускаемых изделий многих фирм в дальнем и ближнем зарубежье содержатся микроконтроллеры, и область их применения постоянно увеличивается.

Микроконтроллеры широкого назначения выпускаются многочисленными зарубежными фирмами: Motorola, NEC Corporation, Siemens, Microchip и другими.

Применение методов и технических средств обработки информации цифровой вычислительной техникой в релейной защите и автоматике (РЗА) привело к созданию интегрированных комплексов, выполняющих все функции традиционных устройств РЗА и обладающих широкими информационными свойствами и сервисными возможностями, существенно повышающими надежность и эффективность функционирования технических средств автоматического управления электроэнергетическими установками [1].

Для обучения студентов цифровой технике и программированию микроконтроллеров на кафедре имеется учебный микропроцессорный комплект (УМК-7). Помимо обучения языку ассемблер на примере контроллера PIC16F877A, студенты знакомятся с внутренней и внешней структурой современных микроконтроллеров, применяемых в оборудовании.

До дня проведения работы студенты должны к ней подготовиться: прочитать описание лабораторной работы и составить программы для своего варианта задания.

Отчет о лабораторной работе должен содержать титульный лист, задание, текст программы, заполненную таблицу результатов, рисунки с копиями экрана.

За оформленную выполненную работу студенты группы или студент, индивидуально выполнивший работу, получает 50%. Остальные 50% получаются при защите лабораторной работы, которая выполняется письменно или устно.

Оборудование и программное обеспечение для проведения лабораторных работ: Windows 98 или выше, среда MPLAB, комплект УМК-7.

Система команд PIC16C87X включает в себя байт-ориентированные команды, бит-ориентированные, операции с константами и команды передачи управления.

Для байт-ориентированных команд буква f обозначает собой регистр, с которым производится действие; буква d определяет, куда положить результат. Если d =0 или w, то результат будет помещен в аккумулятор, при d=1 или f результат будет помещен в регистр с адресом f, упомянутом в команде.

Для бит-ориентированных команд буква b обозначает номер бита, участвующего в команде, а f -это регистр, в котором этот бит расположен.

Для операций с константами и команд передачи управления k обозначает восьми или одиннадцати битную константу.

Все команды выполняются в течение одного командного цикла. В двух случаях исполнение команды занимает два командных цикла: проверка условия и переход, либо изменение программного счетчика, как результат выполнения команды GOTO и CALL. Один командный цикл состоит из четырех тактов генератора. Таким образом, для генератора с частотой 20 МГц время исполнения командного цикла будет 0,2 мкс.

MPLAB IDE (*Integrated Development Environment)* –интегрированная среда разработки, представляет собой программный продукт, работающий под управлением операционной системы WINDOWS и предназначенный для написания, отладки и оптимизации программ для Microchip PIC контроллеров. MPLAB IDE включает в себя:

- MPLAB Project Manager– менеджер проекта;
- MPLAB Editor – текстовый редактор;
- MPLAB-SIM Simulator – симулятор;
- MPASM – универсальный Ассемблер, а так же MPLINK (Линковщик) и MPLIB (Библиотекарь).

Для контроллеров семейства PIC17, PIC18, а так же dsPIC, в среде MPLAB IDE есть возможность работать с компиляторами языка Си –более высокого уровня, по сравнению с ассемблером.

Ряд аппаратных средств, работающих под управлением MPLAB IDE, приведен ниже:

- MLPAB-ICE Emulator – эмулятор;
- программаторы PICSTART Plus и PRO MATE 2;
- инструментальные средства третьих лиц – большое количество других компаний делают инструментальные средства разработки, работающие с MPLAB.

MPLAB-ICD (*In Circuit Debugging)* - внутрисхемный отладчик, позволяет выполнять внутрисхемную отладку программы в реальной схеме, используя генератор и периферию отлаживаемого контроллера. Для отладки используются только два вывода контроллера, работа идёт по *ICSPTM* интерфейсу.

Используя *ICD*, можно отлаживать аппаратно-зависимые участки когда, которые трудно, а порой невозможно воспроизвести в симуляторе, например: работа с АЦП, измерение временных параметров входного сигнала, организация обратной связи с управляемым объектом, отладка интерфейсов USART, SPI, I2C и т.п.

MPLAB-ICD работает под управлением универсальной программной среды MPLAB и обладает следующими возможностями:

- отладка в режиме реального времени и пошаговая отладка;
- связь с компьютером по RS-232;

- одна задаваемая точка останова;

- просмотр и модификация содержимого управляющих регистров, RAM и EEPROM;

- внутрисхемная отладка и встроенная система программирования *PIC*-контроллеров серии PIC16F87x*;*

- работа от источника питания отлаживаемой конструкции в диапазоне от 3,0 до 5,5 В;

- диапазон рабочих частот от 32 кГц до 20 МГц.

1 Лабораторная работа № 1. Системы счислений. Память программ и память данных PIC16F87x

Цель работы: изучение систем счисления и организации памяти программ и памяти данных.

1.1 Системы счисления

В микроконтроллере данные и промежуточные результаты представлены в двоичной системе счисления. Все регистры памяти в микроконтроллере нумеруются в шестнадцатеричной системе. Системы счислений, применяющиеся в цифровой технике, представлены в приложении А. Там же приведен алгоритм перевода чисел из двоичной системы в шестнадцатеричную и наоборот [2].

В десятичной системе прибавление к цифре 9 единицы дает в результате число 10, то есть 9+1=10. Аналогичный результат и в шестнадцатеричной системе F+1=10 или 2F+1=30. То есть в младший разряд записываем цифру 0, в старший добавляем 1. В двоичной системе 1+1=10.

Задание 1. Сложите числа в двоичной системе. Представьте слагаемые и результаты вычислений в шестнадцатеричной, десятичной и в двоично-десятичной системах счислений.

Т а б л и ц а 1.1 – Варианты к заданию 1

Вариант	1	2	3	4	5	6
Число А	00110101	01101101	00111001	01100101	01001001	01110100
Число В	00110001	01101001	00101001	00100101	01000001	00111101

Вариант	7	8	9	10	11	12
Число А	00110001	01101001	00101001	00100101	01000001	01110111
Число В	00110101	01101101	00111001	01100101	01001001	00110111

Вариант	13	14	15	16	17	18
Число А	00110101	00111001	01101001	00110101	01010001	01011101
Число В	00110001	01101101	00111001	01100101	01001001	00110010

1.2 Память программ и память данных PIC16F87x

В МК этого семейства существует память программ и память данных.

Память программ и память данных имеют раздельные шины адреса и данных, что позволяет выполнять параллельный доступ. МК PIC16F877/876 имеют 13-разрядный регистр адреса, способный адресовать 8к X 14 слов Flash памяти программ. Адрес вектора сброса 0000h, адрес вектора прерываний – 0004h. Память программ состоит из 4-х страниц, нумеруемых 0, 1, 2, 3, по 2048 14-разрядных слов.

Память данных, реализованная как EEPROM, состоит из четырех банков, нумеруемых 0, 1, 2 и 3 (приложение Б). Информация в банках хранится в регистрах, состоящих из 8 разрядов. Они делятся на регистры общего назначения (РОН), которые может использовать программист для хранения констант, и к которым можно обратиться прямой или косвенной адресацией, и регистры специального назначения (РСН), выполняющие управление функции ядра и периферийными модулями микроконтроллера . Банки имеют нумерацию в двоичной системе: 00 – банк ‘0’, 01 – банк ‘1’, 10 – банк 2, 11 – банк 3.

В каждом банке находится 128 регистров.

Адреса регистров нумеруются в шестнадцатеричной системе, начиная с адреса 00h, который находится в нулевом банке и кончая адресом 1FFh, который находится в 3-м банке. К РСН обычно обращаются по его имени. Программист, используя РОН для хранения констант или промежуточных данных, присваивает им имена, названия которых позволяют легче реализовывать разработанный алгоритм.

Задание 1. Определите имена регистров и банка по двоичному адресу.

Т а б л и ц а 1.2 - Варианты к заданию 1

Вариант	1	2	3	4	5	6
Регистр А	00000010	00001001	00001010	10000010	10000101	10011111
Регистр В	00101000	00110101	00111000	01010101	01100001	11101110

Вариант	7	8	9	10	11	12
Регистр А	10010010	10010011	10011000	00011110	00011110	10011001
Регистр В	10100010	10100010	10110101	11100101	11101110	00011111

Вариант	13	14	15	16	17	18
Регистр А	10010010	10010011	10011000	00011110	00011110	10001110
Регистр В	10100010	10100010	10110101	11100101	11101110	01111111

Задание 2. По имени регистра и банка определите его адрес.

Т а б л и ц а 1.3 - Варианты к заданию 2

Вариант	1	2	3	4	5	6
Банк	0	1	1	1	1	2
Имя	PCL	FSR	INTCON	STATUS	TRISB	INTCON

Вариант	7	8	9	10	11	12
Банк	1	0	1	0	1	1
Имя	PCL	TMR1	INTCON	STATUS	TRISD	TRISB

Вариант	13	14	15	16	17	18
Банк	1	0	1	0	1	1
Имя	PCL	PIR1	PIE2	PIR2	PIE1	FSR

1.3 Оформление отчета по лабораторной работе

Отчет выполняется на одного выполняющего. В конце приводятся ответ на вопросы. За сданный отчет студент получает 50%. Остальные проценты дополняются после защиты лабораторной работы устно или письменно.

Контрольные вопросы

1. Что такое основание системы счисления?
2. Какие цифры имеются в двоичной системе?
3. Какие цифры имеются в шестнадцатеричной системе?
4. Сколько банков в микроконтроллере PIC16F877A?
5. В какой системе счисления записываются адреса регистров?
6. Сколько страниц в памяти программ микроконтроллера PIC16F877A?

2 Лабораторная работа № 2. Написание и организация программ на Assembler для PIC16F87x. Изучение среды MPLAB

Цель работы: изучить правила написания программ и создания проекта в среде MPLAB.

2.1 Краткие теоретические сведения и задания для внеаудиторной подготовки

У микроконтроллеров имеются выводы для подключения внешних устройств. Выводы объединены в функциональные группы-регистры PCH: PORTA, PORTB, PORTC, PORTD, PORTE. В процессе выполнения программы на выводы порта либо будет подаваться напряжение от ЦПУ, либо оно будет поступать с внешнего источника. Содержание регистра порта, а также нумерация разрядов порта показана на рисунке 2.1. Наличие напряжения будем обозначать цифрой 1, отсутствие - цифрой 0.

Все выводы регистра нумеруются, начиная с нуля, справа налево, и называются разрядами или битами. Отдельные выводы портов могут быть настроены на ввод или вывод сигнала определенного уровня. К выводам порта могут быть подключены цепи к приборам и устройствам, управляющим технологическим процессом, или идущие к датчикам, с которых вводится

аналоговая или цифровая информация. На рисунке 2.1 представлена схема регистра.

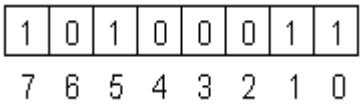


Рисунок 2.1 – Схема 8-разрядного регистра

Представленный на рисунке 2.1 набор нулей и единиц формально считают числом в двоичной системе счисления. Положение цифры в числе называют разрядом.

Настройка отдельных бит в регистрах PORTA, PORTB, PORTC, PORTD, PORTE на ввод или вывод осуществляется через соответствующие биты регистров TRISA, TRISB, TRISC, TRISD, TRISE. Если, например, в регистре TRISC записаны все нули в первый полубайт, а во второй полубайт записаны единицы (подана энергия), то в PORTC биты первого полубайта работают на вывод информации из МК, биты второго полубайта работают на ввод. Если в PORTC будет подана энергия из ЦПУ на все 8 разрядов, то на выходе PORTC появится энергия только в разрядах 0-3. Выход от защелок бит 4-7 будет заперт [Л.11].

В любой регистр нельзя непосредственно записать какое - либо число (константу). Сначала константу по инструкции MOVLW записывают в регистр-аккумулятор, затем по инструкции MOVWF пересылают ее в требуемый регистр.

При включении МК выполнение программы начинается с адреса h’00’ памяти программ.

При выборе реального подключаемого оборудования к контактам микроконтроллера необходимо руководствоваться электрическими характеристиками конкретного микроконтроллера [Л.11].

2.2 Задания для внеаудиторной подготовки

2.2.1 Познакомьтесь с описанием УМК-7 и MPLAB по приложению В.

2.2.2 Познакомьтесь по приложению Г с битами RP1 и RP0 (разряды 6 и 5) регистра STATUS.

2.2.3 Познакомьтесь по приложению Д и [Л.11] с инструкциями, использованными в приведенной ниже программе: CLRF, BSF, BCF, MOVLW, MOVWF, MOVF, GOTO, CLRW.

2.2.4 Определите в каких банках находятся регистры PORTA, PORTB, PORTC, PORTD, PORTE, TRISA, TRISB, TRISC, TRISD, TRISE по приложению Б.

2.2.5 Изучите приведенный ниже образец программы в качестве аналога для написания программы в соответствии с Вашим вариантом.

2.3 Создание проекта программы на Assembler средствами MPLab

2.3.1 Исходные данные и текст программы.

Цель: создать программу, поясняющую функции регистров TRIS по настройке разрядов соответствующих портов на ввод или вывод. Программа должна содержать разделы, присущие полноценной программе:

- присвоение имен определенным РОН;
- указания по записи программы в память программ;
- настройка необходимых регистров;
- рабочая часть программы.

Нижеприведенный текст программы реализует поставленную цель. Для наглядности изображения программы целесообразно придерживаться рекомендованного синтаксиса. Заголовок программы записывается через 5 отступов (одно нажатие Tab). После точки с запятой в программе печатаются комментарии, которые пропускаются компилятором. В данном тексте они созданы для пояснения назначения отдельных разделов программы и правил написания инструкций (команд).

Include<p16F877A.inc>; подключение библиотеки символьных имен РСН
;*В этой части программы отдельным РОН присваиваются имена, имена*
;*пишут без отступа, далее пробел Tab, пишем EQU(указатель EQU от equal*
;*(англ.) – одинаковый, равносильный*), *один пробел и адрес РОН*
;*если нет необходимости в создании именованных РОН,то этой части нет*
Con EQU h'25'
Con1 EQU h'26'
;*В этой части пишутся указания по записи созданной программы в память*
;*программ*
org h'00'; следующая инструкция (NOP) будет записана в память
;программ по адресу h'00'.
nop; пустая инструкция будет записана по адресу h'00' памяти программ
nop; инструкция будет записана по адресу h'01' памяти программ
nop; инструкция будет записана по адресу h'02' памяти программ
;Регистр памяти с адресом h'04' зарезервирован для записи инструкции вызова
;программы, которая запускается для обработки появившегося прерывания
org h'05'; следующая инструкция будет записана начиная с адреса h'05'
;*Инструкции для настройки МК*
CLRF STATUS; очищаем 5,6,7 бит регистра **Status** и выбираем нулевой
;банк. Между инструкцией CLRF и именем регистра один пробел. Сохранить
;окно PrtSc в Word-файле после запуска проекта в режиме симулятора
BSF STATUS,5; записав в 5-й разряд '1' (без пробела после инструкции),
;выбираем первый банк
MOVLW B'11110000'; пересылаем в аккумулятор W число B'11110000',
;соответствующее заданию по настройке выходов порта PORTC
;*Настраиваем регистр TRISC в режим, разрешающий вывод энергии из*

;защелок PORTC на выход в тех разрядах, которые соответствуют заданию

MOVWF TRISC; настраиваем биты 0-3 PORTC на вывод, а биты 4-7 на ввод

BCF STATUS,5; возвращаемся в нулевой банк, в нем будет работать

;рабочая часть

; *Рабочая часть программы*

NACH; метка, на которую возвращаемся после выполнения инструкции ;GOTO

MOVLW B'11111111'; пересылаем в аккумулятор W число B'11111111'

MOVWF PORTC; в окне Watch в 0-3 разрядах появляются единицы, а в ; разрядах 4-7 – нули. Сохранить окно PrtSc в Word-файле

BCF PORTC,0; в разряде 0 - 0.

; сохранить окно PrtSc в Word-файле

BSF PORTC,0; заносим 1 в разряд 0

MOVLW B'10000001';засылаем второе число по заданию в аккумулятор

MOVWF Con; засылаем эту константу в РОН с именем Con

CLRW ;очистка аккумулятора, проверьте в окне Watch

MOVF Con,w; засылаем константу из регистра Con в аккумулятор

; сохранить окно PrtSc в Word-файле

MOVWF PORTC;пересылаем эту константу в PORTC

; сохранить окно PrtSc в Word-файле

GOTO NACH; переход на метку **NACH** для повторения.

END; конец программы

На основе образца подготовьте программу для своего варианта.

Таблица 2.1 – Варианты заданий

Вариант	1	2	3	4	5	6
TRISC	11111000	11110000	11100000	11000000	10000000	00000000
Бит измен-й	1	2	3	4	5	6
2-е число	10000001	10000010	10000011	10000100	10000101	00000110

Таблица 2.2 – Варианты заданий

Вариант	7	8	9	10	11	12
TRISC	00111111	00011111	00001111	00000111	00000011	00000001
Бит измен-й	7	6	5	4	5	6
2-е число	10000001	11000010	11100011	11110100	11111001	11111100

2.3.2 Создание проекта.

2.3.2.1 Если Вы работаете в режиме MPLAB с записью программы в память микроконтроллера и запускаете программу на УМК-7, то убедитесь в наличии электропитания на УМК-7 (соедините шлейфом шину тумблеров с лампочкой нагрева термометра). Проверьте надежность соединения схемы электропитания и УМК-7 с COM-портом системного блока. Нарушения соединений может привести к выходу из строя COM-порта. Работа

программы, реализующей поставленную цель в данной лабораторной работе, может быть просмотрена при использовании MPLAB в режиме симулятора, т.к. нет необходимости в обработке внешних сигналов или выдачи сигналов на какие-то внешние устройства. Данное упражнение можно написать и отладить на ноутбуке, инсталировав на нем MPLAB. Для написания и отладки программы в режиме симулятора создайте папку на компьютере, на котором Вы работаете в разрешенной на этом компьютере папке. Название папки пишите латинскими буквами (по имени Вашей подгруппы, если Вы работаете в лаборатории университета). Если Вы хотите созданный и отлаженный проект на собственном компьютере запустить в лаборатории университета, то создайте такую же папку на флэшке, в которой Вы сохраните копию созданного проекта.

2.3.2.2 Запустите MPLAB.Откройте новый лист: File>New. В появившемся диалоговом окне набираем текст программы или скопируем его в окно. После копирования обязательно отредактируйте. Перепишите заголовок в редакторе MPLab и удалите старое написание, затем проверьте правильность соблюдения пробелов и отделения текста комментарий точкой с запятой.

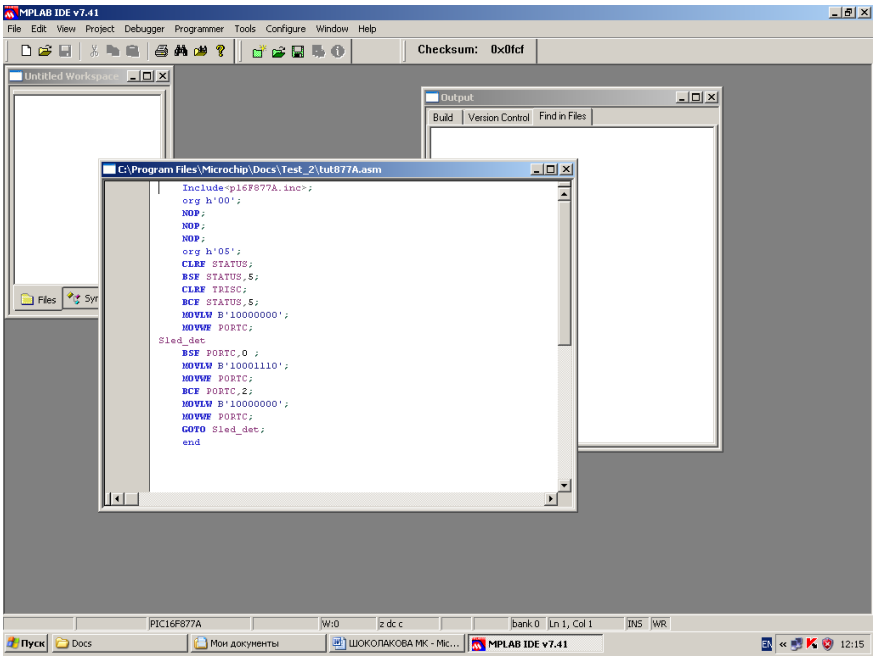


Рисунок 2.2 – Окно для ввода текста программы

Сохраняем файл с помощью File\ Save as (тип файла –Assembly Source Files (*.asm) в ранее созданной Вами папке (пример:Lr2_1. asm). Номер варианта после подчеркивания.

2.3.2.3 Запускаем мастера по созданию проекта : Project>Project Wizzard

2.3.2.4 В появившемся диалоговом окне нажать кнопку «Далее».

В следующем окне указать тип микроконтроллера PIC16F877A и нажать кнопку «Далее».

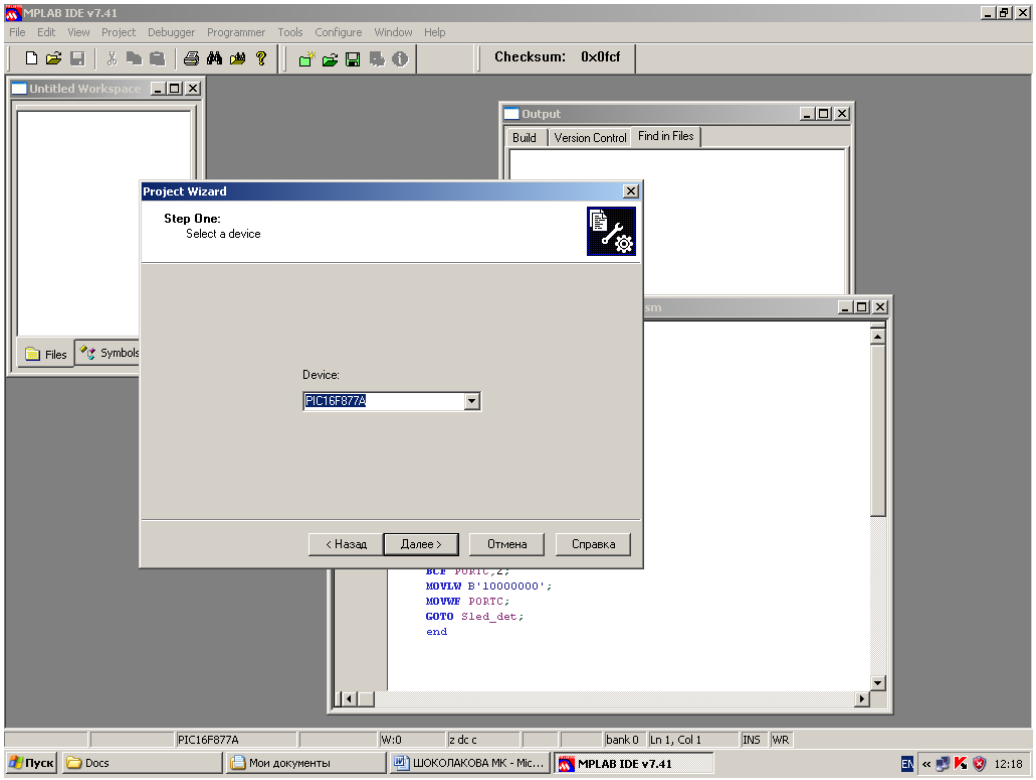


Рисунок 2.3 – Окно выбора типа микроконтроллера

2.3.2.5 В следующем открывшемся окне, представленным на рисунке 2.4, нажать кнопку «Далее», не изменяя НИКАКИХ параметров.

2.3.2.6 После нажатия «Далее» открывается новое окно (рисунок 2.5). В нем заполните поле Project name (ввести имя проекта, например, Lr21. asm) и в поле Project Directory через Brows указать (не открывать) папку, в которой находится сохраненный Вами файл Lr21. asm.

2.3.2.7 Нажимаем кнопку «Далее». Открывается окно, представленное на рисунке 2.6. В нем с помощью кнопки «Add», добавить с левого окна в правое:

- сохраненный файл Lr21.asm;

- файл P16F877A.INC из Мой компьютер\System C:\Program Files\Microchip\MPASM Suite\ P16F877A.INC с библиотекой инструкций Assembler для P16F877A . Этот файл иногда может находится в папке, где находится файл Lr21.asm. Оба файла отметить значком «флажок».

2.3.2.8 Нажимаем кнопку «Далее» и в следующем открывшемся окне кнопку «Готово». На рисунке 2.8 представлено окно с созданным мастером проектом.

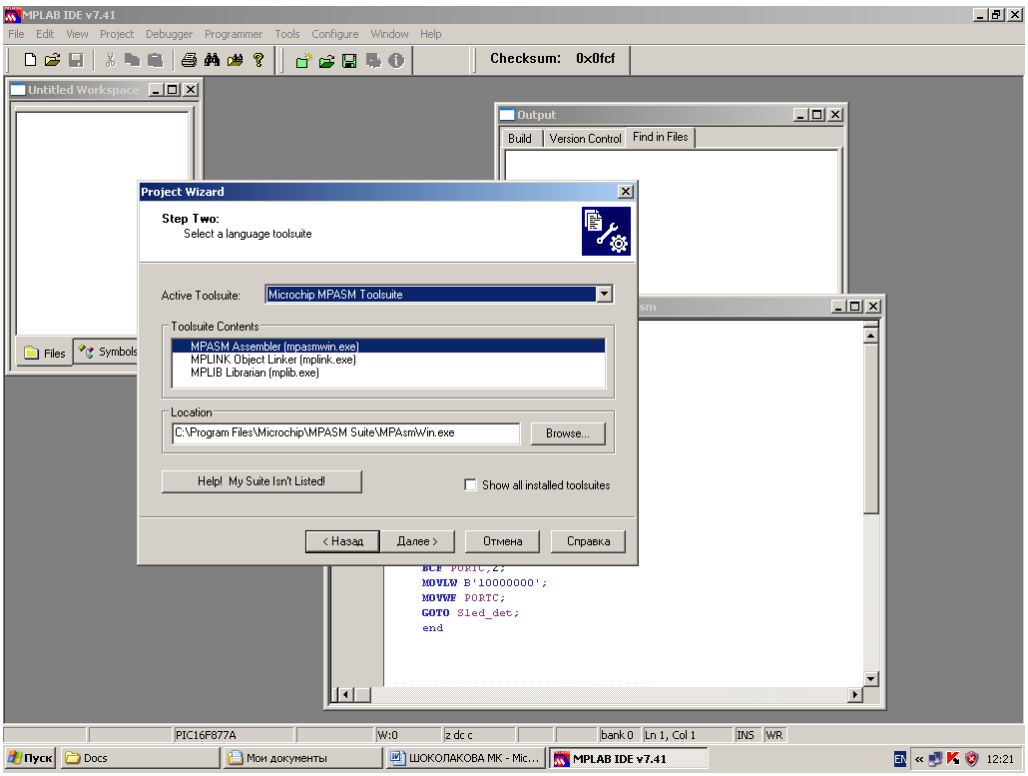


Рисунок 2.4 – В этом окне ничего не вводится

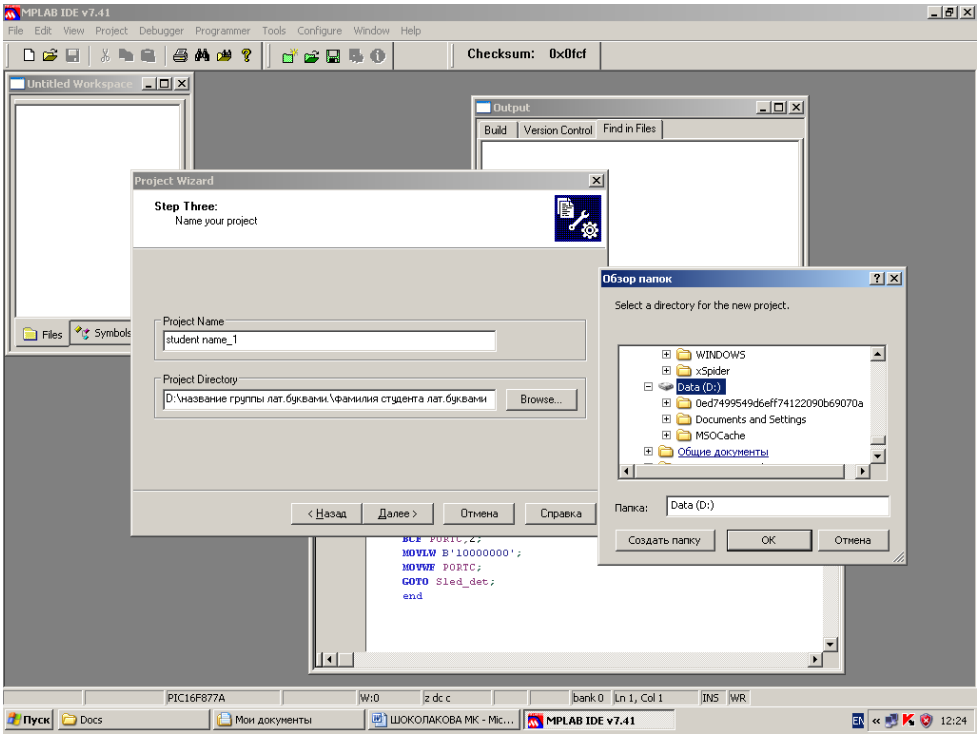


Рисунок 2.5 – Окно выбора папки, где сохраняется проект

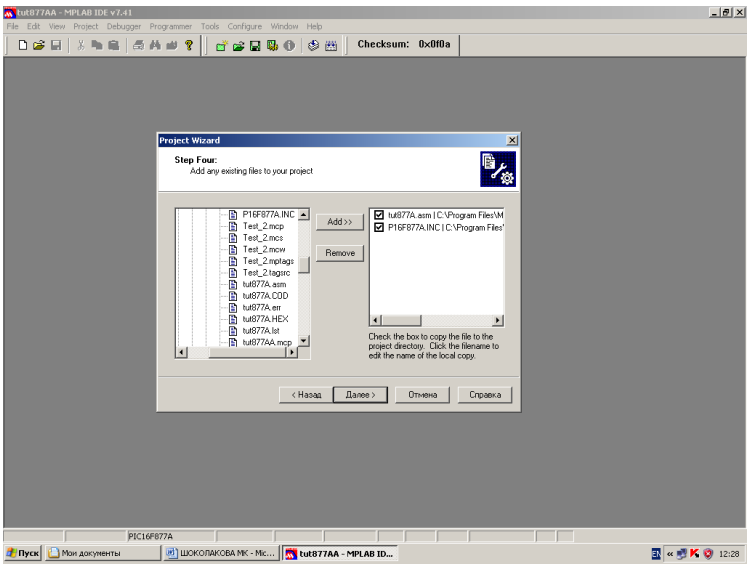


Рисунок 2.6 – Окно выбора библиотеки инструкций и файла проекта

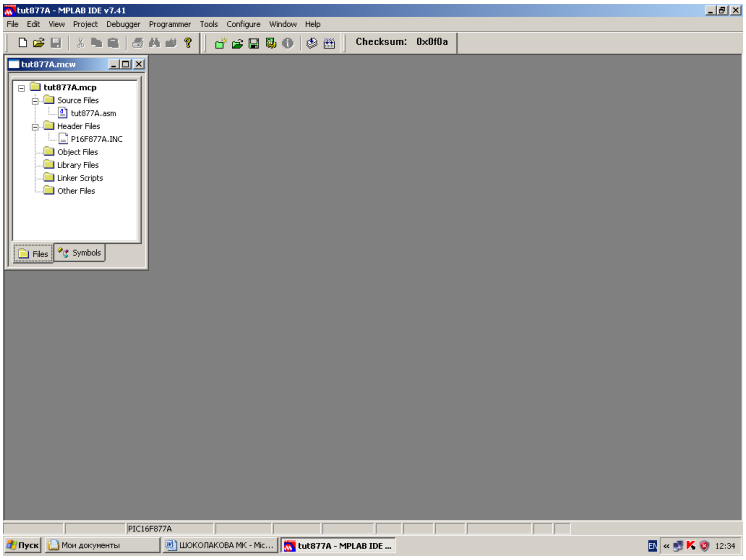


Рисунок 2.7 – Окно с созданным проектом

2.3.2.9 Производим настройки для запуска проекта. Из меню Configurue выбираем: Configurue>Select Device. Появляется окно, представленное на рисунке 2.8 с параметрами МК PIC16F877A. Подтверждаем их OK.

2.3.2.10 Далее выбираем: Configurue>Configuration Bits. В открывшемся диалоговом окне, представленном на рисунке 2.9, выставляем параметры согласно рисунку (Hs, Off, Off, Off, Disabled, Write Protection off, Off, Off). После ввода всех параметров закрываем данное окно.

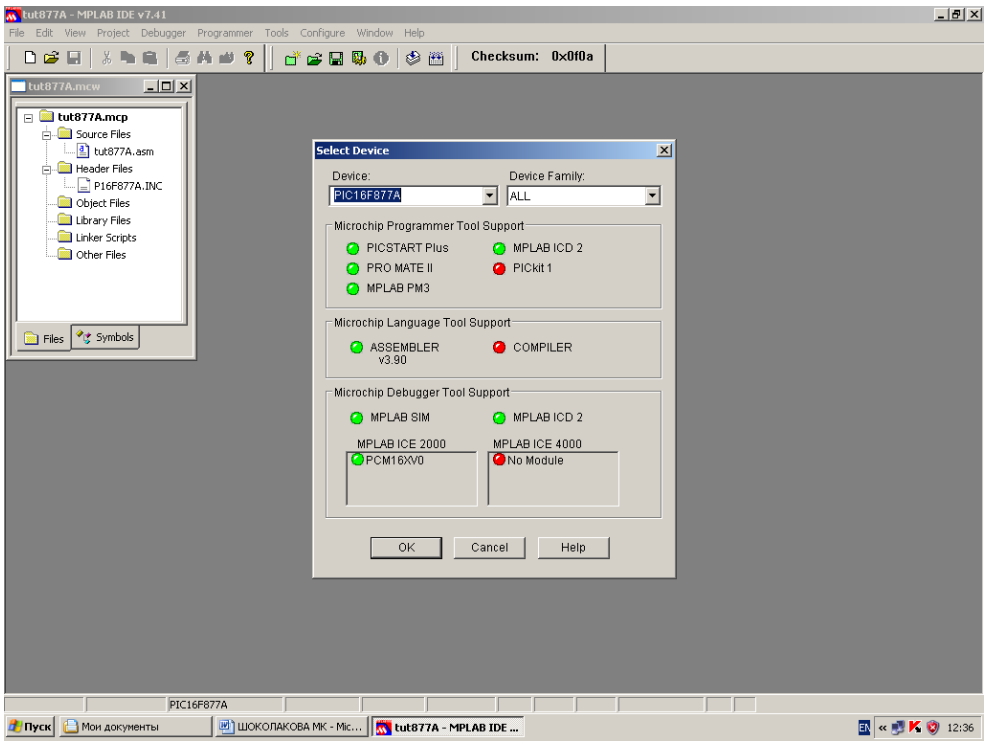


Рисунок 2.8 – Окно с параметрами для выбранного микроконтроллера

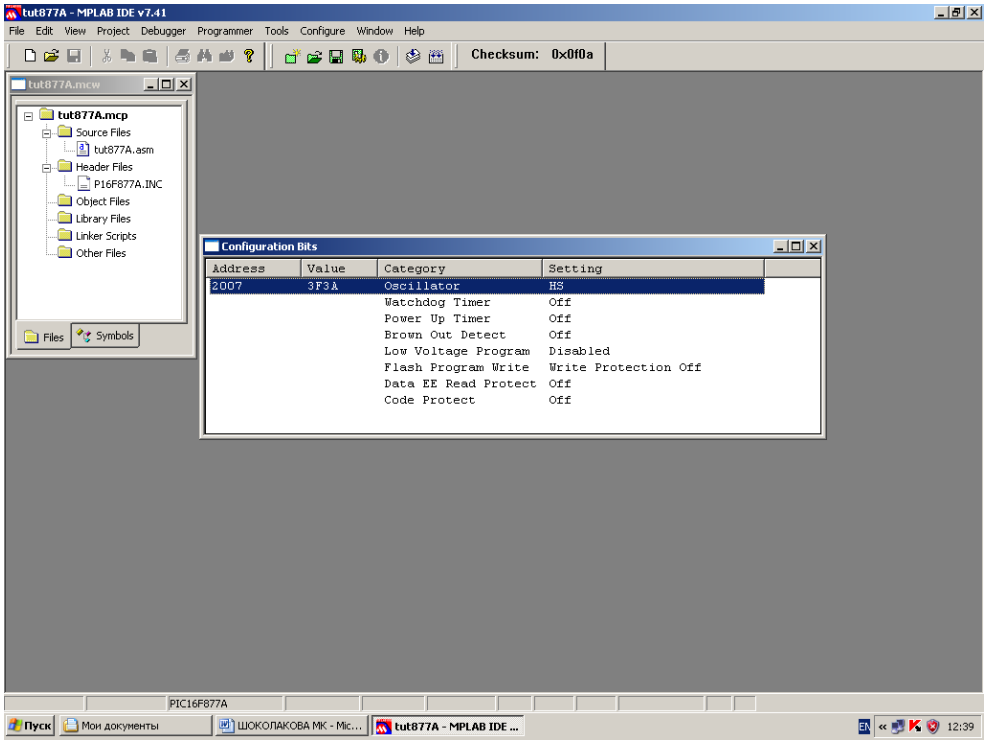


Рисунок 2.9 – Окно настройки параметров

2.3.2.11 Если проект будем выполнять в режиме симулятора, то выбираем: Debugger>Select Tool>MPLAB Sim.

2.3.2.12 Компилируем проект : Project>Make.

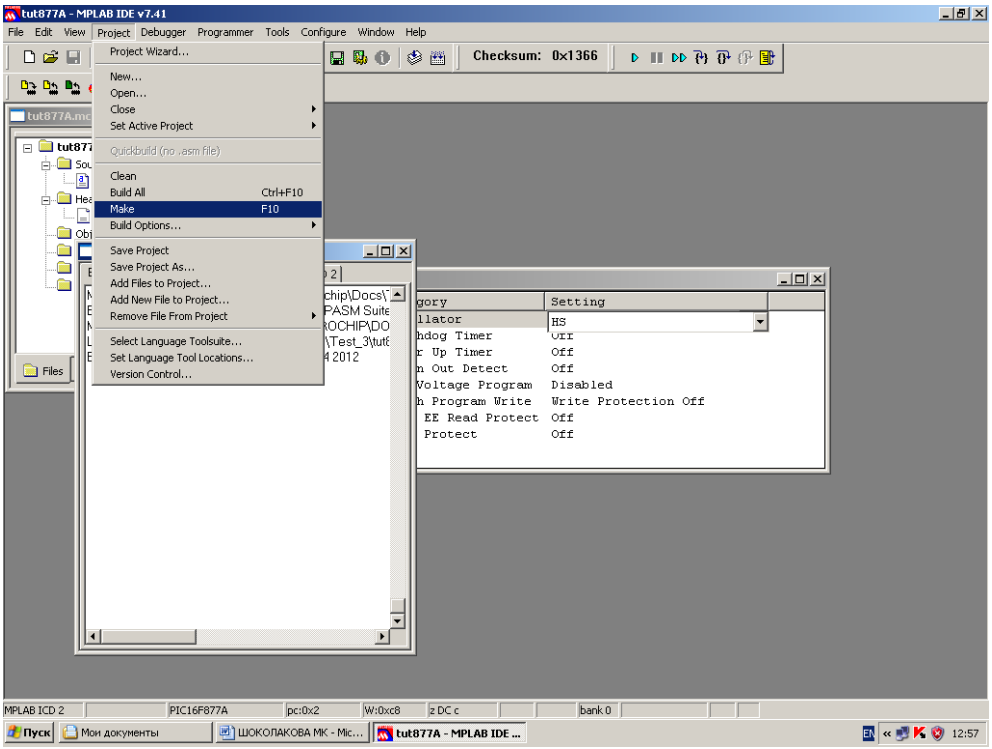


Рисунок 2.10 – Окно после компиляции

Если проект выполнен без ошибок, то в окне сообщений компилятора не будет Error. Если будут строки с Error, то делаем двойной щелчок по этой строке. Указатель устанавливается в ошибочном месте в тексте программы. Устраняем ошибку.

2.3.2.13 Создаем окно наблюдения : View>Watch (рисунок 2.11). С помощью кнопки «Add SFR» добавляем в окно наблюдения регистры, состояние которых мы хотим увидеть по ходу выполнения программы. В данной программе мы хотим увидеть состояние PCH :STATUS,TRISC, PORTC, WREG. Набираем начальную букву регистра в левом верхнем окне окна Watch, выбираем из открывшегося списка наименование нужного регистра и нажимаем кнопку «Add SFR». Чтобы увидеть состояние POH, набираем начальную букву регистра в правом верхнем окне окна Watch, выбираем из открывшегося списка наименование нужного регистра (Con) и нажимаем кнопку «Add Symbol». Для каждого регистра назначаем систему счисления, в которой будет выводиться состояние регистра, преобразованное в значение числа: двоичная (b), шестнадцатеричная (h), десятичная (d). Для этого в окне Watch щелкаем правой клавишей по нужному регистру, выбираем Свойства и в открывшемся окне выбираем систему счисления при отображении. На рисунке 2.12 представлено созданное окно наблюдения.

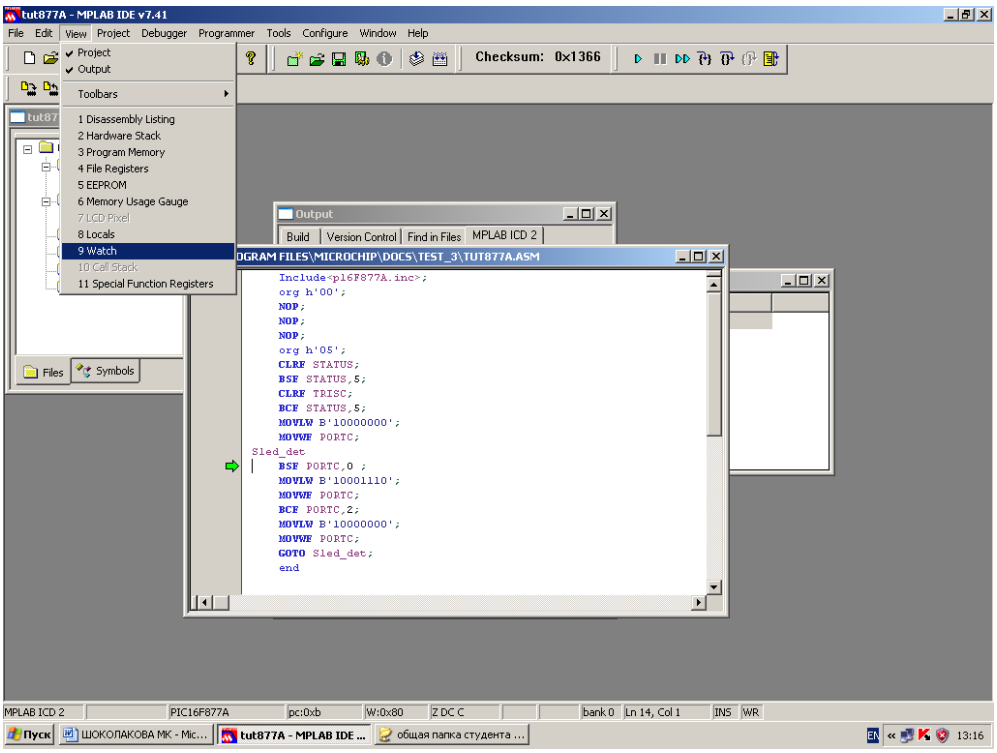


Рисунок 2.11 – Окно создания окна наблюдения

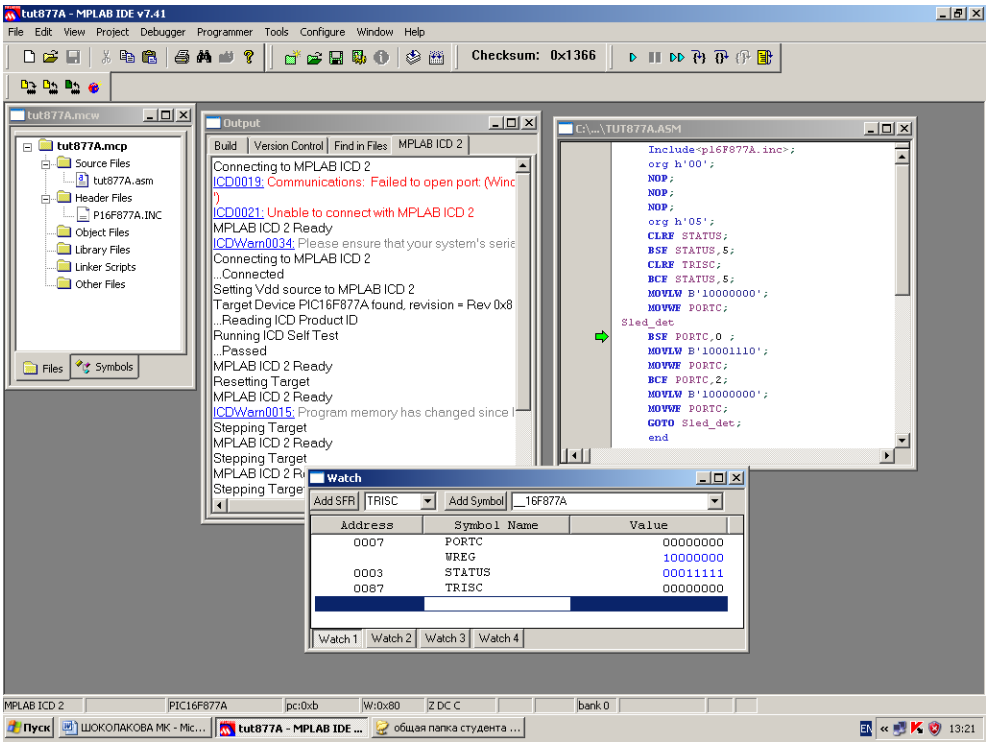


Рисунок 2.12 – Окно с созданным окном наблюдения

Сохраняем созданный проект Project>Save.

Вы заметили, что после выполнения Project>Make, на тексте програм-мы появился зеленый указатель. Симулятор ожидает команду продолжения операций. Прокрутите исполнение программы в пошажном режиме, нажимая

на F7, или на значок Into на верхней ленте, или в меню Debugger. Если что-то не получилось, исправьте ошибки и снова сделайте Project>Make. Когда программа заработает правильно, сохраните ее в папке на компьютере и в папке на флэшке: Project>Save и Project>Save As. Имя проекта латинскими буквами, а расширение MPlab присвоит автоматически (.mcp).

Запуск созданного проекта в MPlab на другом компьютере выполняется в следующей последовательности. Вызовите Project>Open с флэшки Ваш проект. Сделайте Project>Make. Появится зеленый маркер в тексте программы. Можете работать дальше.

2.3.2.14 В режиме симулятора невозможно отлаживать аппаратно-зависимые участки кода, которые трудно, а порой невозможно воспроизвести в симуляторе, например: работа с АЦП, измерение временных параметров входного сигнала, организация обратной связи с управляемым объектом, отладка интерфейсов *USART, SPI, I2C* и т.п.

Если Вы хотите записать созданную программу в память микроконтроллера и запустить программу на УМК-7, то надо продолжить дальнейшую настройку для запуска проекта в режиме MPLAB ICD2.

Установите флэшку с Вашим проектом на компьютере, подключенным к УМК-7. На данном компьютере вызываем из этой папки созданный проект. Вызовите Project>Open с флэшки Ваш проект. Сделайте Project>Make.

Выбираем: Debugger>Select Tool>MPLAB ICD2 (рисунок 2.13).

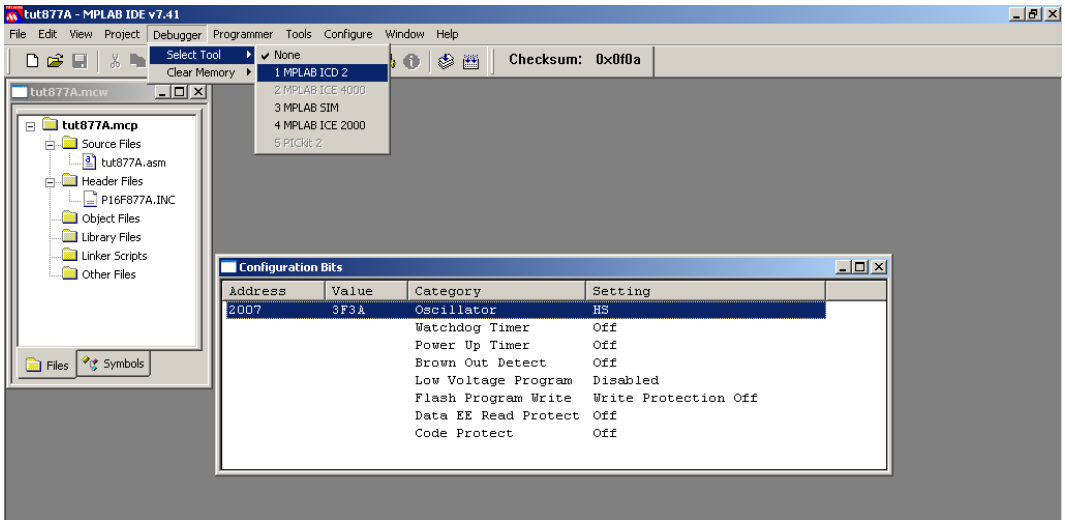


Рисунок 2.13 – Окно выбора режима MPLAB ICD2

2.3.2.15 Настраиваем соединение УМК-7 с компьютером через порт COM-1: Debugger>Settings>Communication>COM1. Далее OK.

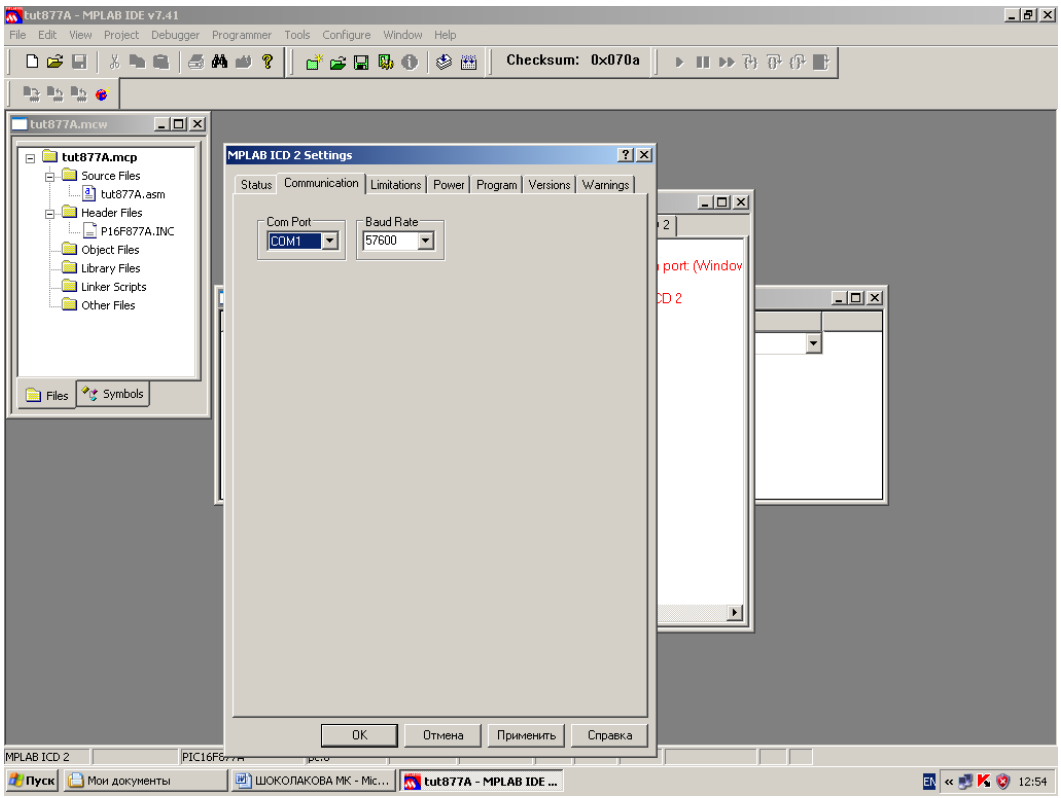


Рисунок 2.14 – Окно выбора параметров соединения

2.3.2.16 Записываем созданную программу в память программ микроконтроллера: Debugger>Program.

2.3.2.17 Для покомандного исполнения программы нажимаем F7 или Step Into (вверху, справа). Для выполнения созданной программы в автоматическом режиме: Debugger>Run. Остановка: Debugger>Halt. Возвращение в начальное состояние Debugger>Reset. Эти же операции можно выполнить нажатием соответствующих кнопок на панели.

2.4 Оформление отчета по лабораторной работе

Отчет выполняется на группу. В Word-файле сохраняется текст программы и через PrtSc - характерное состояние регистров в окне при пошаговом выполнении программы. В текстах программ обычно указаны места, в которых надо сохранять окна наблюдения. Отчет в формате Word сохраняется в общей папке компьютерного класса. Защита производится на следующем занятии.

2.5 Выводы

2.5.1Режим управления технологическим оборудованием через регистр PORTC (или другой порт), или режим ввода информации в разряды PORTC (или другой порт) устанавливается через настройку соответствующего регистра TRIS.

2.5.2 При использовании инструкции CLRF для очистки регистра STATUS обнуляются биты 5-7.

2.5.3 При всех изменениях в написании программы по ходу ее отладки необходимо заново компилировать проект Project>Make и сохранять Project>Save. Если проект выполняется в режиме MPLAB ICD2, то необходимо заново записать его в память программ Debugger>Program.

Контрольные вопросы

1. Что выполняет инструкция CLRF STATUS?
2. Что выполняет инструкция MOVLW B’00001111’?
3. Что выполняет инструкция MOVWF PORTC?
4. Что выполняет инструкция MOVF PORTC?
5. Что выполняет инструкция CLRW?
6. С какой целью выполняется инструкция MOVWF TRISD?
7. С какой целью выполняется инструкция STATUS, 5?
8. С какой целью выполняется инструкция BCF STATUS, 5?
9. Что выполняет инструкция GOTO METKA?
10. Что такое адрес регистра и содержимое регистра?
11. Назначение символа «;» в тексте программы.
12. В каких банках находятся регистры PORTC и TRISC?
13. Назначение регистра TRISC.
14. Какие из рассмотренных инструкций относятся к байт ориентированным?
15. Какое напряжение питания МК?
16. Какое напряжение на выходных контактах PORTC?
17. Какие из рассмотренных инструкций относятся к байт ориентированным?
18. Какие из рассмотренных инструкций относятся к бит ориентированным?
19. Какие из рассмотренных инструкций относятся к инструкциям управления и операциям с константами?

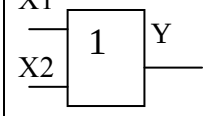
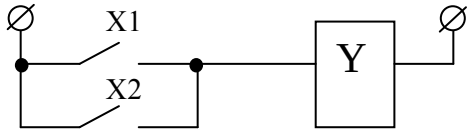
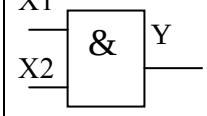
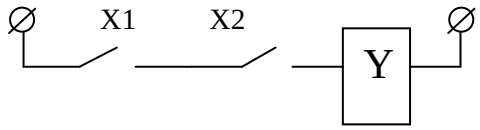
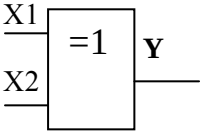
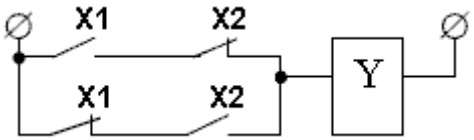
3 Лабораторная работа №3. Сложение, вычитание. Деление, умножение. Логика

Цель работы: изучение инструкций ADDWF, ANDWF, IORWF, XORWF, SUBWF, RRF, RLF, а также изменения содержания бит в регистре STATUS от результата выполнения предыдущей инструкции. Выполнение сложения, вычитания, умножения, деления и применение логических функций.

3.1 Краткие теоретические сведения и задания для внеаудиторной подготовки

В таблице 3.1 представлены логические функции, их таблицы истинности, условные обозначения и интерпретация на электрической схеме. Все логические операции выполняются поразрядно.

Таблица 3.1 – Логические функции Y=f(X1,X2)

Инструкция	Таблица истинности			Условное обозначение	Интерпретация на электрической схеме
	X1	X2	Y		
IORWF Функция OR (ИЛИ)	0 0 1 1	0 1 0 1	0 1 1 1		 При замыкании любого контакта в цепи будет ток
ANDWF Функция AND (И)	0 0 1 1	0 1 0 1	0 0 0 1		 Ток в цепи будет только при замыкании обоих контактов
XORWF Функция XOR (Исключи- тельное ИЛИ)	0 0 1 1	0 1 0 1	0 1 1 0		 Ток в цепи будет только тогда, когда переключатели X1 и X2 будут в разном положении

Логическая функция ИЛИ (OR) может применяться для включения оборудования любой из двух кнопок, расположенными в разных местах помещения.

Логическая функция И (AND) может применяться в том случае, если оборудование должно включаться, при срабатывании (включении) обязательно двух контактов. Например, освещение в помещении автоматически включается при низкой освещенности и наличия в нем людей.

Логическая функция Исключающее ИЛИ (XOR) может применяться, например, для управления освещением длинного тоннеля. В разных концах тоннеля устанавливаются переключатели X1 и X2, имеющие нормально открытый и нормально закрытый контакты. С любого конца тоннеля, изменив положение переключателя, можно включить освещение, если оно было выключено или выключить – если оно было включено.

3.2 Исходные данные и выполнение программы

Изучите формат инструкций **ADDWF, SUBWF, ANDWF, IORWF, XORWF, RRF, RLF, BTFSS и BTFSC** по приложению Д. Изучите текст программы данной лабораторной работы, приведенный ниже. Программа легче читается, если в инструкциях вместо адреса регистра писать его символьное имя. В лабораторной работе №2 мы уже писали в инструкциях вместо адресов РСН их имена: PORTC, TRISC, STATUS, W и присваивали символьные имена РОН (R1 EQU h’21’). Указатель EQU от equal (англ.) – одинаковый, равносильный.

В микроконтроллерах PIC16F87x операция вычитания выполняется как сложение в дополнительном коде. Ниже приведены простые примеры перевода вычитаемого в дополнительный код.

А) 8-10=-2

В двоичном виде: 00001000 – 00001010

Запишем 00001010 в инверсном коде: 11110101. Для получения дополнительного кода добавим 1. Получим 11110110.

00001000
+ 11110110
= 11111110

Результат выражен в дополнительном коде. Отнимем 1 от результата: 11111110 – 1=11111101. Инвертируем результат: 00000010=2.

Б) 10-8=2

Запишем 00001000 в дополнительном коде, произведя инвертирование и потом добавление 1: 11110111 + 1 = 11111000.

11111010 + 11111000 = 00000010 = 2 , при этом отметим, что произошел перенос в 8-й разряд.

Задание: создать программу, позволяющую проверить результат применения инструкций логических операций и отследить реакцию содержимого разрядов 0-2 в регистре STATUS после выполнения предшествующей операции. Появление в этих разрядах регистра STATUS флагов (1) или их отсутствие (0) позволяет осуществлять условные переходы при использовании инструкций BTFSS и BTFSC.

Текст программы лабораторной работы.

```
        Include <p16F877A.inc>; в этом файле описаны символьные имена РСН
R1   EQU     h'21'; R1 - символьное имя регистра с адресом h’21’
R2   EQU     h'22'; R2 - символьное имя регистра с адресом h’22’
R3   EQU     h'23'; R3 - имя регистра с адресом h’23’
REZ  EQU     h'24'; REZ - имя регистра для записи результатов
;            Инструкции для настройки МК
        ORG  h'00'; следующая инструкция NOP будет записана по адресу h’00’
        NOP; пишется для настройки отладчика
```

```
        NOP
        NOP
        ORG     h'05'; следующая инструкция CLRF запишется по адресу h’05’
        CLRF  STATUS; очищаем регистр от мусора, выбираем нулевой банк
        ; сохранить окно
        BSF     STATUS, 5; переходим в первый банк в нём регистр TRISC
        CLRF    TRISC; настраиваем все биты PORTC на вывод данных
        BCF  STATUS,  5; возвращаемся в нулевой банк
;               Рабочая часть программы. Ввод данных
        MOVLW   D'240'; запись константы  в  аккумулятор W
        MOVWF   R1; запись содержимого W в регистр R1
        MOVLW   D'130'; запись константы в аккумулятор W
        MOVWF   R2; запись содержимого W в регистр R2
        MOVLW   D'5'; запись константы в W
        MOVWF   R3; запись содержимого W в регистр R3
        ; сохранить окно
;               Выполнение вычислений
        MOVF   R1, W; запись константы из R1 в регистр W
        SUBWF   R2, W; вычитание W=R2-W=R2- R1
        MOVWF   REZ; переслать содержимое W в регистр REZ
        ; сохранить окно
        MOVF   R2, W; запись константы из R2 в регистр W
        ADDWF  R1, W;  W=R2+W=R1+R2
        MOVWF   REZ
        ; сохранить окно
        MOVF   R2, W; копируем содержимое R2 в W
        SUBWF   R1, W; вычитание W=R1-W=R1-R2. Результат оставляем в W
        MOVWF   REZ; переслать содержимое W в регистр REZ
        ; сохранить окно
        MOVF   R2, W; запись константы в регистр W из R2
        ADDWF R3, W; W=R3+W=R2+R3
        MOVWF   REZ
        ; сохранить окно
        MOVF   R2, W; запись константы в регистр W из R2
        SUBWF R2, W
        MOVWF   REZ
        ; сохранить окно
        MOVF   R2, W; запись константы в регистр W из R2
        ANDWF   R1, W; операция W=R1 AND W(R2), результат в W
        MOVWF   REZ
        ; сохранить окно
        MOVF   R2, W; запись константы в регистр W из R2
        IORWF   R1, W; операция W=R1 OR W(R2), результат в W
        MOVWF   REZ
```

```
; сохранить окно
MOVF   R2, W; запись константы в регистр W из R2
XORWF   R1, W; операция W=R1 XOR W(R2), результат в W
MOVWF   REZ
; сохранить окно
RRF R1,w;деление на 2 без очистки бита C регистра STATUS. После
;выполнения сдвига вправо сдвигаемый бит 0 из регистра R1 переместился в
;бит C регистра STATUS, а единица из бита C регистра STATUS добавилась
;слева к значению регистра R1 и сохранилась в аккумуляторе W. Значение
;неверно.
MOVWF   REZ
; сохранить окно
RLF R1,w; 24. 2=480, произошло переполнение регистра, в нем
;осталось 480-256=224, в бите C регистра STATUS появилась 1, чтобы
;операция деления выполнилась верно, надо его обнулить
MOVWF   REZ
; сохранить окно
BCF STATUS,C
RRF R1,w
MOVWF   REZ
; сохранить окно
END; конец программы.
```

Данные для варианта возьмите из таблицы 3.2.

Т а б л и ц а 3.2 – Варианты задания

Вариант	Числа R1, R2, R3	Вариант	Числа R1, R2, R3
1	81h, 12h, 1h	6	86h, 67h, 6h
2	82h, 23h, 2h	7	87h, 68h, 7h
3	83h, 34h, 3h	8	88h, 69h, 8h
4	84h, 45h, 4h	9	89h, 6Ah, 9h
5	85h, 56h, 5h	10	8Ah, 6Bh, Ah

3.3 Оформление отчета по лабораторной работе

Отчет выполняется на группу по результатам выполнения лабораторной работы в режиме MPlabSim. В Word-файле сохраняется текст программы и через PrtSc - характерное состояние регистров в окне при пошаговом выполнении программы. В этот же файл включается таблица 3.3, с данными, соответствующими варианту задания, которые записываются при пошаговом выполнении программы в режиме MPlabSim. В выводах отметьте, как изменялось состояние бит в разрядах 0-2 регистра STATUS в зависимости от результата предыдущей операции.

Т а б л и ц а 3.3 - Пример записи результатов работы команд

Числа на входе		Действие	Результат		Значение в Status Форма В
Форма В или D	Форма В или D		REZ(D)	Wreg(B)	
		CLRF STATUS			00011111
R1=D’240’	R2=D’130’	R2-R1=130-240	D’146’	-	00011010
R1=D’240’	R2=D’130’	R1+R2=240+130	D’114’	-	00011001
R1=D’240’	R2=D’130’	R1-R2=240-130	D’110’	-	00011001
R2=D’130’	R3=D’5’	R2+R3=130+5	D’135’	-	00011000
R2=D’130’	R2=D’130’	R2-R2=130-130	D’0’	-	00011111
R1=b’11110000’	R2=b’10000010’	R1 AND R2	-	10000000	00011011
R1=b’11110000’	R2=b’10000010’	R1 OR R2	-	11110010	00011011
R1=b’11110000’	R2=b’10000010’	R1 XOR R2	-	01110010	00011011
R1=b’11110000’	R1/2	RRF	248	11111000	00011010
R1=b’11110000’	R1*2	RLF	224	11100000	00011011
		BCF STATUS,C			00011010
R1=b’11110000’	R1/2	RRF	120	01111000	00011010

Создайте окно наблюдения для всех применяемых в программе регистров в необходимом формате чисел. Результаты работы всех операций с числами запишите в таблицу 3.2.

3.4 Выводы

3.4.1 Для получения правильного применения инструкций RRF и RLF перед ее применением необходимо обнулить бит С в регистре STATUS.

3.4.2 Деление на 2 нечетных чисел выполняется с погрешностью.

3.4.3 Если операция суммирования заканчивается с переполнением, то в регистре STATUS бит 0=1. Если операция заканчивается с отрицательным результатом, то в регистре STATUS бит 1=1, а бит 0=0; сложение без переполнения – биты 0-2=0; результат операции нулевой – бит 2 (Z)=1.

Контрольные вопросы

1. Как определить с регистрами какого банка работает программа?

2. Что выполняет инструкция ADDWF R2,F и в какой регистр помещается результат ее выполнения?

3.Что выполнят инструкция SUBWF R2,F и в какой регистр помещается результат ее выполнения?

4.Что выполняет инструкция ANDWF R1,w и в какой регистр помещается результат ее выполнения?

5. Что выполняет инструкция IORWF R1,f и в какой регистр помещается результат ее выполнения?

6. Что выполняет инструкция XORWF R2,W и в какой регистр помещается результат ее выполнения?

7. Приведите инструкции установки и сброса бита.

8. Назначение директивы ORG h’05’.

9. Приведите таблицы истинности логических операций.

10. Назовите знакомые вам устройства, в которых применяются МК.

11. В каком банке находятся созданные в программе регистры R1, R2 и REZ.

12. Какой результат мы увидим при сложении 230+60 в МК?

13. Какой результат мы увидим при вычитании 130-135 в МК?

14. Что надо делать, чтобы операции деления на 2 и умножения на 2 выполнялись верно?

4 Лабораторная работа № 4. Вложенные таймеры

Цель работы: изучение способов создания задержек выполнения следующей инструкции, в соответствии с реализацией управляющего алгоритма. Например, такие задержки нужны в цикле АПВ, в реализации алгоритмов АЧР и т.д. Реализацию пауз между следующими друг за другом инструкциями можно осуществить с помощью таймеров на основе вложенных циклов или таймеров, реализованных в микроконтроллере как периферийные модули микроконтроллера: TMR0, TMR1 и TMR2.

Изучите работу таймеров TMR0, TMR1 и TMR2 и назначение регистра PCL по [11] и приложению Г, а также инструкции DECF и DECFSZ.

В МК PIC16F87x выполнение инструкции осуществляется за четыре такта. В зависимости от времени выполнения одного такта при циклическом исполнении набора инструкций может быть создана временная задержка. В 8-разрядный регистр, в который вносится константа, определяющая число циклов, может быть записано максимально число 255. Если применяются вложенные циклы, то каждый последующий внешний цикл предполагает выполнение внутреннего цикла и, следовательно, существенного увеличения времени работы таймера и создания паузы перед выполнением следующей инструкции в программе.

Если тактовая частота генератора равна 20 МГц, то время выполнения одного такта равно 0,05 мкс. Время выполнения машинного цикла, состоящего из четырех тактов равно 0,2 мкс.

Фрагмент одного внутреннего цикла таймера в приведенной ниже программе выполняет команду вычитания и команду BTFSS (проверки появления 1 во 2-м бите регистра STATUS, сообщающей, что результат вычитания нулевой) за два цикла и команду безусловного перехода за 2 цикла, т.е. в сумме за 4 цикла. Максимально во внутреннем цикле может быть выполнено 255 подциклов (предельное значение десятичного числа, которое

может быть введено в 8-ми разрядный регистр). При вычитании 255-й раз условие STATUS(Z)=1 выполняется, пропускается команда GOTO и происходит выход из цикла. Общее время выполнения внутреннего цикла:

$$255 \cdot 4 \cdot 0{,}2 = 204 \text{ мкс.}$$

При организации трех внешних циклов происходит повторение всех внутренних циклов:

$$3 \cdot 204 = 612 \text{ мкс.}$$

На основании приведенного примера, подбором числа повторений в каждом цикле, можно рассчитать общее время паузы, созданной таймером. На основании приведенного примера, взяв число повторений в каждом цикле из задания, можно рассчитать общее время паузы, созданной таймером.

4.1 Исходные данные и выполнение программы

Будем выполнять программу в режиме MPlabSim. Используем текст программы, приведенный ниже. В окно наблюдения включите PCH STATUS, TRISC, Wreg, PORTC, PCL, а также POH Sch_in, Sch_out, Sch3. В пошаговом режиме перед запуском таймера CALL Timer сохраните в Word-файле состояние окна наблюдения и далее сохраняйте эти окна после завершения каждого цикла. Сравните состояние регистров STATUS, Wreg и PCL перед включением таймера и после выхода из него. Определите разницу в значениях разрядов. Измените в программе значения количества выполняемых циклов в соответствии с вариантами из таблицы 4.1. На основании приведенного выше расчета времени паузы, подсчитайте задержку, создаваемую таймером для Вашего варианта.

Таблица 4.1 - Задания по количеству операций в каждом цикле

Наимен.цикла	Вариант 1	Вариант 2	Вариант 3
Sch_in	255	255	255
Sch_out	255	255	255
Sch3	2	4	6

Текст программы лабораторной работы № 4.

```
    Include<p16F877A.inc>
Sch_in   EQU  H'22'; счетчик внутреннего цикла задержки.
Sch_out EQU  H'23' ; счетчик внешнего цикла задержки.
Sch3     EQU  H'24' ; счетчик третьего цикла задержки.
    ORG h'00'
    NOP
    NOP
    NOP
    ORG h'05'
    CLRF STATUS
```

```
        BSF STATUS,5
        CLRF TRISC
        BCF STATUS,5
        MOVLW   B'10000000'
        MOVWF   PORTC
        MOVLW d'2'; задаем количество повторения третьего цикла
;таймера равное 2 только для отладки. Для выполнения на микроконтроллере
;в режиме Run используйте задание по варианту из таблицы 4
; сохранить окно в отладочном режиме
        CALL Timer
        MOVLW B'00000001'
        MOVWF PORTC; по загоранию лампы Л-0 на УМК-7 фиксируем
;секундомер для определения времени работы таймера в режиме Run
    ; сохранить окно в отладочном режиме
        GOTO $; разделитель основной программы и подпрограмм.
    Timer; имя подпрограммы с двумя вложенными циклами.
        MOVWF  Sch3; значение W является аргументом для таймера.
    M3  ;метка третьего цикла
        MOVLW   D'2'; для отладки 2, а для варианта-по заданию
        MOVWF  Sch_out; устанавливаем значение внешнего счетчика.
    M_out ; метка внешнего счетчика.
        MOVLW   D'2';      для отладки 2, а для варианта-по заданию
        MOVWF   Sch_in; устанавливаем значение внутреннего счетчика.
    M_in ;метка внутреннего счетчика.
        DECF Sch_in,F; уменьшаем значение счетчика Sch_in на 1
    ; сохранить окно в отладочном режиме, когда Sch_in=0
        BTFSS STATUS,Z; если счетчик Sch_in обнулился,
  ;Z=1,пропускаем GOTO.
       GOTO   M_in; срабатывает только при Z=0.
       DECF   Sch_out,F; уменьшаем значение счетчика Sch_out  на 1.
       BTFSS   STATUS, Z; если счётчик обнулился, пропускаем GOTO.
        GOTO   M_out; инструкция срабатывает только при Z=0.
       DECF   Sch3,F; уменьшаем значение счетчика Sch3 на 1,
        BTFSS  STATUS,Z ; если счетчик Sch_in обнулился, Z=1,
;пропускаем GOTO.
       GOTO   M3; срабатывает только при Z=0.
        RETURN; конец подпрограммы Timer.
        END; конец текста всей программы.
```

4.2 Оформление отчета по лабораторной работе

Отчет оформляется на группу. В созданный Word-файл копируется текст программы и окна по ходу выполнения программы в режиме MPLabSim (перед запуском таймера и после его отработки после вывода константы в

PORTC). Рассчитывается время паузы, созданной таймером при использовании заданных по варианту значений количества повторений в каждом цикле и тактовой частоты 20 МГц.

4. 3 Выводы

4.3.1 Применение встроенных таймеров позволяет создать в 8-разрядных микроконтроллерах необходимые расчетные временные задержки.

4.3.2 В программе может быть создано несколько подпрограмм с разными временными задержками.

Контрольные вопросы

1. Что выполняет инструкция GOTO METKA?
2. Назначение символа «;» в тексте программы?
3. Какое напряжение питания МК?
4. Какое напряжение в PORTC?
5. Поясните инструкцию условного перехода BTFSC STATUS, Z
6. Поясните инструкцию условного перехода BTFSS STATUS, Z
7. Поясните инструкцию DECF Sch_out,F
8. Поясните инструкцию INCF R,f
9. Поясните инструкцию GOTO $
10. Поясните инструкцию RETURN
11. Что такое машинный цикл и как определить время его выполнения?
12. Сколько машинных циклов требуется для выполнения команд в подпрограмме Timer в отладочном режиме при задании всем счетчикам значения 3?
13. Как вызывается подпрограмма на выполнение?
14. Какой адрес появляется в счетчике команд в регистре PCL после выполнения подпрограммы?

5 Лабораторная работа № 5. Таймер TMR1

5.1 Краткие теоретические сведения

Таймер TMR1 входит в состав периферийных устройств микроконтроллера PIC16F877A. TMR1 - 16-разрядный таймер/счетчик, состоящий из двух 8-разрядных регистров (TMR1H и TMR1L), доступных для чтения и записи. Счет выполняется в спаренных регистрах (TMR1H:TMR1L), инкрементируется их значение от 0000h до FFFFh. Если перед включением TMR1 в эти регистры будут записаны некоторые числа, то после включения TMR1 единица добавляется вначале к значению, записанному в регистр TMR1L. Включается TMR1 установкой бита TMR1ON в 1 (регистр T1CON<0>).

При переполнении регистров они будут снова равны 0000h. При переполнении счетчика устанавливается в 1 бит флага прерывания TMR1IF в регистре PIR1<0> вне зависимости от состояния бита разрешения/запрещения прерываний. Флаг прерывания снимается в программе обработки прерывания или по ходу выполнения программы.

Запрет прерывания, (при котором не будет вызвана программа обработки прерывания , название которой указывается в адресе 04h) может быть выполнен:

- установкой бита глобального запрещения/разрешения прерываний GIE=0 (регистр INTCON<7>);
- установкой бита запрещения/разрешения прерываний от периферийных устройств PEIE=0 (регистр INTCON<6>);
- установкой бита запрещения/разрешения периферийных прерываний TMR1IE=0 (регистр PIE1<0>).

Если прерывания будут разрешены (соответствующие биты равны 1), то после переполнения счетчика запустится подпрограмма обработки этого прерывания, название которой записывается в память программ по адресу 04h. (разумеется, можно в подпрограмме обработки прерывания, анализируя флаги прерывания, выявить причину появления прерывания и правильно определить реакцию программы на это прерывание).

TMR1 может работать в режимах: режим таймера, режим счетчика. В лабораторных работах рассмотрен только режим таймера.

Управляющие биты TMR1 находятся в регистре T1CON.

Если бит TMR1CS=0 (регистр T1CON<1>), то выбирается внутренний источник тактовых импульсов Fosc/4. TMR1 инкрементируется при каждом машинном цикле, т.е. вначале заполняется регистр TMR1L и после его переполнения добавляется 1 в регистр TMR1H. Регистр TMR1L обнуляется и опять инкрементируется. При переполнении регистра TMR1H появляется прерывание, а бит TMR1IF в регистре PIR1<0> становится равным 1 (появляется «флаг»). Этот флаг необходимо программно снимать, чтобы различить следующее переполнение. Если TMR1 не выключен, то он автоматически начнет заполняться до нового переполнения. Запускается TMR1 установкой бита TMR1ON в единицу в регистре T1CON<0>.

Регистры TMR1H и TMR1L не сбрасываются в 00h при сбросе по включению питания и других видах сброса, кроме сброса по сигналу триггера специальных событий модуля CCP1 и CCP2.

Предделитель TMR1 предназначается для замедления заполнения счетчика в соответствии со своей настройкой. Коэффициент деления предделителя (биты 5-4 в регистре T1CON<5:4>) принимает следующие значения:

11 = 1:8;
10 = 1:4;
01 = 1:2;
00 = 1:1.

Предделитель очищается при записи чисел в регистр TMR1L или TMR1H.

5.2 Исходные данные и выполнение программы

Соедините разряды PORTC и разъемы сигнальных ламп также, как и в лабораторной работе №2. Запишите текст программы. Для отладки ее в режиме MPlabSim в целях сокращения циклов заполнения регистров TMR1N и TMR1L и появления флага их переполнения в регистре PIR1(бит TMR1IF) уберите точку с запятой перед строками, в которых вносятся в аккумулятор и затем в регистры TMR1H и TMR1L временные числа. Точки с запятой перед строками, где вносятся изменения в настройки предделителя, не убирайте. В окно наблюдения включите PCH STATUS, W, PORTC, PCL, TMR1H, TMR1L, T1CON, PIR1, PIE1 и INTCON. В пошаговом режиме сохраните в Word-файле состояние окна наблюдения перед запуском Timer1 и после окончания работы. Сравните состояние регистров STATUS, W и PCL перед включением таймера и после выхода из него. Определите разницу в значениях разрядов. Восстановите в программе точки с запятой в строках, где вносились в регистры TMR1H и TMR1L временные числа. Снимите со строк изменения предделителя точки с запятой и внесите значение числа П из задания. Если есть возможность проверки работы программы на УМК-7, то действуйте в соответствии с указаниями в предыдущем упражнении. Запустите программу в режиме Run. При загорании всех лампы в разряде 7 включите секундомер, а при загорании всех ламп зафиксируйте время. Запишите время в отчете.

Таблица 5.1 - Задания по изменению времени работы TMR1

Параметр	Вариант 1	Вариант 2	Вариант 3
Предделитель	00000000	00100000	00110000

Текст программы лабораторной работы № 5.

```
Include<p16F877A.inc>
ORG h'00'
NOP
NOP
NOP
ORG h'05'
CLRF STATUS
BSF STATUS,5
CLRF TRISC
CLRF PIE1;запрещаем периферийные прерывания
BCF STATUS,5
CLRF T1CON; установка TMR1 в режим ожидания
;тактирование TMR1от внутреннего генератора, значение
;предделителя частоты 1:1
```

```
CLRF TMR1H; очищаем старший регистр-счетчик TMR1
CLRF TMR1L; очищаем младший регистр-счетчик TMR1
CLRF INTCON; запрещаем все прерывания и прерывания от TMR1
MOVLW b'00010000'
MOVWF T1CON;устанавливаем предделитель частоты 1:2
;MOVLW b'П';устанавливаем предделитель частоты по заданию
;MOVWF T1CON
MOVLW B'10000000'
MOVWF PORTC;включаем секундомер
CALL Timer1
MOVLW B'11111111'
MOVWF PORTC; засекаем время работы таймера
GOTO $; разделитель основной программы и подпрограмм.
Timer1
MOVLW D'255'; для отладки TMR1, при добавлении 1 регистр
;переполнится, появится флаг прерывания, оба регистра  обнулятся
MOVWF TMR1H;для отладки TMR1
MOVLW D'254';для отладки TMR1
MOVWF TMR1L;для отладки TMR1
BCF PIR1,TMR1IF; сброс флага переполнения таймера
BSF T1CON,TMR1ON; пуск таймера TMR1
;следующие три строки это циклы работы таймера TMR1
M_TRM1;
BTFSS PIR1,TMR1IF; проверка флага переполнения таймера
GOTO M_TRM1; если флага нет,то цикл TMR1 продолжается
BCF T1CON,TMR1ON; останов таймера TMR1
RETURN; конец подпрограммы Timer.
END; конец текста всей программы.
```

5.3 Оформление отчета по лабораторной работе

Отчет оформляется на группу. В созданный Word-файл копируется текст программы и окна по ходу выполнения программы в режиме MPLabSim перед включением TMR1 и после возвращения в основную программу. Рассчитывается время паузы, созданной таймером при использовании заданных по варианту значений предделителя.

Контрольные вопросы

1. Что будет, если в подпрограмме не остановить работу TMR1?
2. Как влияет установка предделителя на общее время работы таймера?
3. Как изменяется значение в регистре PCL при входе в подпрограмму Timer1 и при выходе из нее?

4. Где сохраняется значение счетчика регистра, позволяющее при завершении подпрограммы вернутся к выполнению следующей инструкции?

5. В каком регистре и в каком порядковом разряде этого регистра появляется флаг переполнения таймера Timer1?

6. В каком регистре и в каком порядковом разряде этого регистра нужно установить 1 для включения таймера Timer1?

7. В каком регистре и в каком порядковом разряде этого регистра нужно ввести 1 для остановки таймера Timer1?

8. При чтении какой инструкции процессор определяет конец подпрограммы?

9. Назовите сдвоенные регистры Timer1, в которых производится счет.

6 Лабораторная работа № 6. Обработка прерываний от периферийных модулей (TMR1)

6.1 Краткие теоретические сведения

В соответствии со структурной схемой организации прерываний, прерывания от периферийных модулей (таймеров, АЦП, CCP, MSSP, USART, PSP, BOD) отнесены к группе прерываний от периферийных модулей. В упражнении рассмотрено прерывание от переполнения TMR1.

При разрешенных прерываниях, в случае их возникновения, осуществляется переход на регистр 04h в памяти программ , где находится указание о безусловном переходе на подпрограмму обработки прерывания. Сама программа отделяется от основной программы через GOTO $, а заканчивается командой RETFIE. При переходе на подпрограмму бит GIE в регистре INTCON<7> автоматически сбрасывается в 0. В теле этой подпрограммы снимается флаг, сообщающий о возникновении данного прерывания, что исключает повторную обработку прерывания, и выполняются действия, которые необходимо выполнить при появлении данного прерывания. После выполнения инструкции RETFIE восстанавливается автоматически разрешение прерываний и основная программа продолжает выполняться с адреса, перед которым произошло прерывание. Это осуществляется благодаря запоминанию в стеке адреса последней выполненной инструкции.

6.2 Исходные данные и выполнение программы

Таблица 6.1 - Задания по изменению времени работы TMR1

Вариант	1	2	3
Предделитель	00010000	00100000	00110000

В соответствии с этим заданием по варианту 1 коэффициент предделителя 1:2, по варианту 2 – 1:4 и по варианту 3 – 1:8. Для наблюдения

за работой программы в режиме симулятора в окно наблюдения включите туда регистры STATUS, TRISC, Wreg, PORTC, INTCON, PIE1, PIR1, T1CON, TMR1H, TMR1L, PCL. Снимите окна в начальный момент, в момент запуска подпрограммы Timer, программы обработки прерывания PRER и в момент возвращения из программы обработки прерывания в основную программу и вывода на лампы сигнала об окончании процесса. После запуска в программе Timer таймера TMR1 посчитайте за сколько нажатий на F7 при пошаговом исполнении программы заполнится регистр TMR1L и произойдет добавление единицы в регистр TMR1H, что вызовет его переполнение и появление прерывания, а значит и запуск программы обработки прерывания и вывод на разряды 0-4 PORTC энергии. Нажмите рестарт и возвратитесь в исходное состояние программы в режиме симулятора. При режиме MPLab ICD нажмите Halt, а затем рестарт. Проанализируйте изменения в регистрах, связанных с работой TMR1 и состояние счетчика команд PCL.

Текст программы лабораторной работы № 6.

```
Include<p16F877A.inc>
ORG h'00'
 GOTO GLAV; при запуске программы сразу происходит переход на
;выполнение основной программы с пропуском инструкции перехода
; на подпрограмму обработки прерывания Prer
NOP
NOP
NOP
ORG h'04'
GOTO Prer
NOP
GLAV
NOP
NOP
CLRF STATUS
BSF STATUS,5
CLRF TRISC
MOVLW b'00000001'
MOVWF PIE1;разрешаем прерывание от переполнения TMR1
BCF STATUS,5
CLRF PORTC
CLRF TMR1H; очищаем старший регистр-счетчик TMR1
CLRF TMR1L; очищаем младший регистр-счетчик TMR1
MOVLW b'11000000'
MOVWF INTCON; разрешаем все прерывания и прерывания от периферийных модулей
MOVLW b'00000000'
MOVWF T1CON;устанавливаем предделитель частоты 1:1
```

```
MOVLW B'00001000'
MOVWF PORTC; в 3-м разряде появляется единица
MOVLW D'255'; для отладки TMR1, при добавлении 1 регистр
;переполнится, появится флаг прерывания, оба регистра  обнулятся
MOVWF TMR1H
MOVLW D'250';для отладки TMR1
MOVWF TMR1L
BSF T1CON,TMR1ON; пуск таймера TMR1
GOTO $; разделитель основной программы и подпрограмм.
Prer; имя подпрограммы обработки прерывания, вызванного переполнением
;счетчика TMR1, после ее отработки в разрядах 0-5 PORTC единицы
BCF PIR1,TMR1IF; сброс флага переполнения таймера
MOVLW b'00111111'
MOVWF PORTC; в PORTC единицы в разрядах 0-5
BCF T1CON,TMR1ON; останов таймера TMR1
RETFIE
end; конец текста всей программы
```

6.3 Оформление отчета по лабораторной работе

Отчет оформляется на группу. В созданный Word-файл копируется текст программы и окна по ходу выполнения программы в режиме MPLabSim.

Контрольные вопросы

1. После выполнения инструкции RETFIE куда возвращается программа?
2. В каком состоянии находятся биты разрешения прерываний в регистрах INTCON и PIE1 после выполнения инструкции RETFIE ?
3. Как изменяется значение в регистре PCL при входе в подпрограмму обработки прерывания и при выходе из нее?
4. Где сохраняется значение счетчика регистра, позволяющее при завершении подпрограммы вернутся к выполнению следующей инструкции?
5. Какие прерывания относятся к внешним?
6. Сколько регистров в стеке PIC16F877A?
7. Как заполняются регистры стека при появлении разрешенных прерываний?
8. На какой адрес в памяти программ переходит счетчик при появлении разрешенного прерывания?
9. В каком регистре и в каком разряде запрещаются все прерывания?
10. Что надо ввести в разряд разрешения/запрещения прерываний, чтобы разрешить/запретить прерывания?

7 Лабораторная работа № 7. Дистанционное включение высоковольтного выключателя

Цель работы: изучить организацию программы, написанную на Assembler для микроконтроллера PIC16F877A, реализующую алгоритм телеуправления высоковольтным выключателем.

Включение высоковольтных выключателей на необслуживаемых подстанциях (ПС) осуществляется с использованием систем телеметрии дистанционно с диспетчерских пунктов (ДП). Состояние коммутационного оборудования ПС отображается на диспетчерском щите и дисплеях автоматизированной системы управления (SCADA) на ДП. Сигнал на включение (отключение) формируется на пульте управления на диспетчерском пункте (ДП), передается по каналам связи на приемное устройство, расположенное на ПС, и затем поступает в цепь управления включением (отключением) выключателя. Телеуправляемое коммутационное оборудование при включении (отключении) формирует сигнал, который передается обратно на ДП, подтверждая состояние выключателя после выполнения операции. Если операция не будет выполнена, то на ДП не придет сигнал подтверждения выполнения операции. При передачи сигнала по каналам связи и преобразования его в аппаратуре связи и управления происходят задержки, которые могут быть вычислены для нормального процесса. В случае, если сигнал подтверждения выполнения операции не придет обратно с допустимой задержкой, то выключатель остается гореть прежним цветом, но начинает мигать, привлекая внимание оперативного дежурного, который далее действует по регламентирующей инструкции.

7.1 Общее описание лабораторной работы

На рисунке 7.1 приведена схема соединений на УМК-7, позволяющая имитировать описанный выше алгоритм. Сигнал на включение выключателя подается с первого бита регистра PORTD. Проводник соединяет этот бит через тумблер S0 с шиной. Он сразу включен. Другой проводник через тумблер S1 соединен с битом 1 регистра PORTB. Третий проводник через тумблер S2 соединен с битом 0 (RB0) PORTB. PORTC нужно соединить с сигнальными лампами двумя проводами по заданию для сигнализации включенного и отключенного состояния выключателя. Первоначально тумблеры S1 и S2 отключены. В этом режиме при включении программы в режиме MPLab горит лампа отключенного положения выключателя (в соответствии с заданием). Включение выключателя производится тумблером S1. При поступлении напряжения на бит RB1 запускается подпрограмма ожидания поступления сигнала подтверждения включения выключателя с ПС. Если этот сигнал не придет, то лампа сигнализации отключенного положения выключателя начнет мигать с частотой, определяемой таймером Timer1. Сигнал подтверждения включения выключателя имитируется включением

тумблера S2. Поступая на бит RB0, он вызывает внешнее прерывание и запуск программы обработки прерывания, которая останавливает работу счетчика и переводит состояние выключателя во включенное положение (на щите загорается сигнализация включенного положения выключателя).

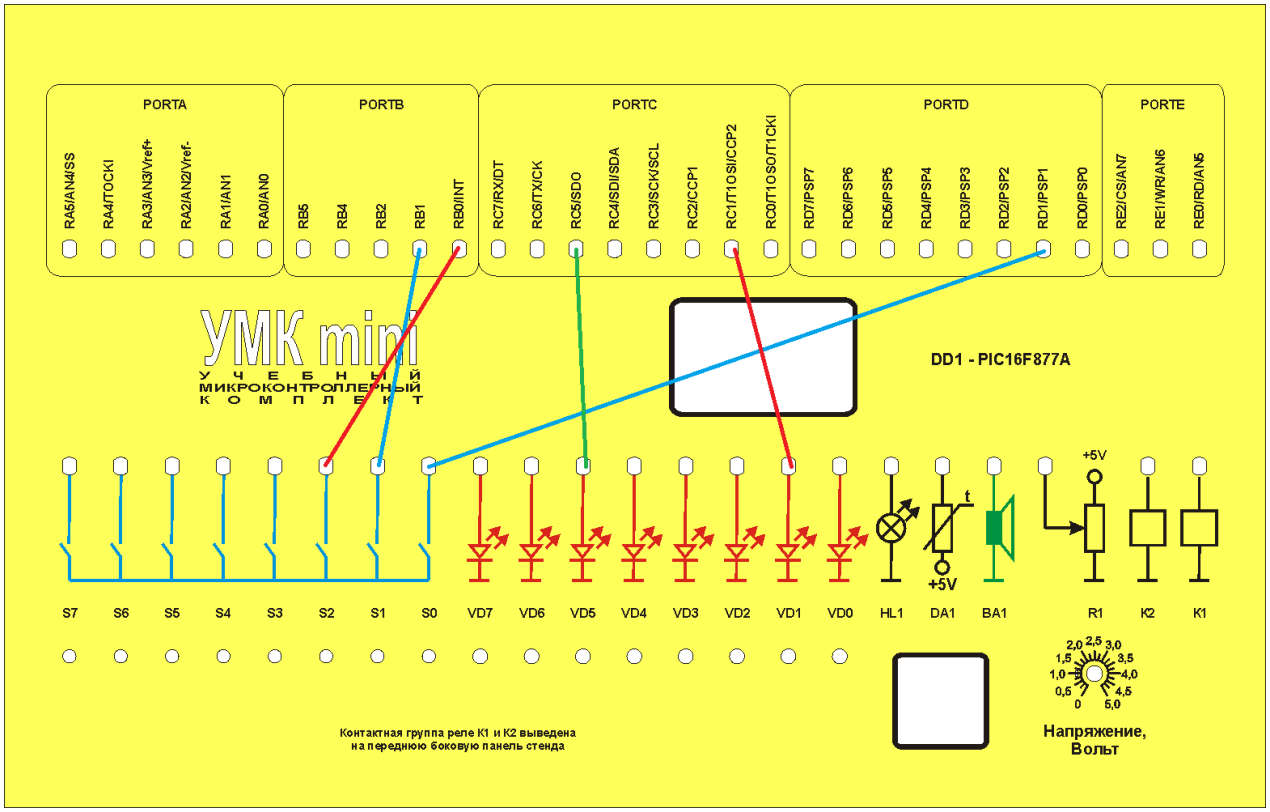


Рисунок 7.1 – Схема соединений на УМК-7 для лабораторной работы № 7

7.2 Исходные данные и выполнение программы

Подготовка текста программы с исходными данными, взятыми из таблицы 7.1, производится в режиме симулятора. На УМК-7 собирается схема соединений по варианту, которая отличается номерами ламп, сигнализирующими включение и отключение выключателя. На компьютере, подключенном к УМК-7, вызовите программу включения выключателя vclon из списка готовых проектов и измените значения констант, соответствующих включению и выключению выключателя, времени допустимой задержки ожидания прихода сигнала подтверждения выключателя, и времени пауз при мигании отключенного выключателя. Выполните компиляцию проекта с изменениями (Project>Make). Настраиваем работу УМК-7 в режиме MPLAB ICD2: Debugger>Select Tool>MPLAB ICD2. Настраиваем соединение УМК с компьютером: Debugger>Settings>Communication>COM1. Далее OK.
Записываем созданную программу в память программ: Debugger>Program.
Создаем окно наблюдения (если оно отсутствовало в базовом проекте) с набором регистров, как в приведенных ниже окнах. После запуска программы

при отключенных тумблерах S1 и S2 программа ждет сигнала на включение выключателя. Включите тумблер S1, и не включайте тумблер S2. С установленной задержкой замигает выключатель отключенного положения выключателя, т.к. в допустимое время не пришел сигнал подтверждения включения выключателя, имитируемый включением тумблера ST2. Запомните, примерно, продолжительность задержки между включением тумблера и зажиганием лампы отключенного положения в режиме мигания. Нажмите Halt. Сохраните окно через PrtSc. Нажмите Reset. Программа вернется в исходное состояние. При повторном пуске Вы должны включить тумблер ST2 за время меньшее, чем допустимая задержка. Нажмите Run, включите тумблер S2. Программа обработки прерывания должна остановить работу подпрограммы ожидания сигнала подтверждения включения выключателя и выключатель на щите будет переведен во включенное положение. Нажмите Halt и сохраните окно через PrtSc, в котором видно, что бит включения выключателя в PORTC равен 1.

Таблица 7.1- Задания к лабораторной работы № 7

Параметры	Вариант 1	Вариант 2	Вариант 3
Номер бита включенного	2	3	4
Номер бита отключенного	6	4	2
Sch3 задержки	60	90	110
Sch3 мигания	5	10	20

На рисунке 7.2 представлено окно с включенным выключателем, а на рисунке 7.3 - окно с отключенным выключателем.

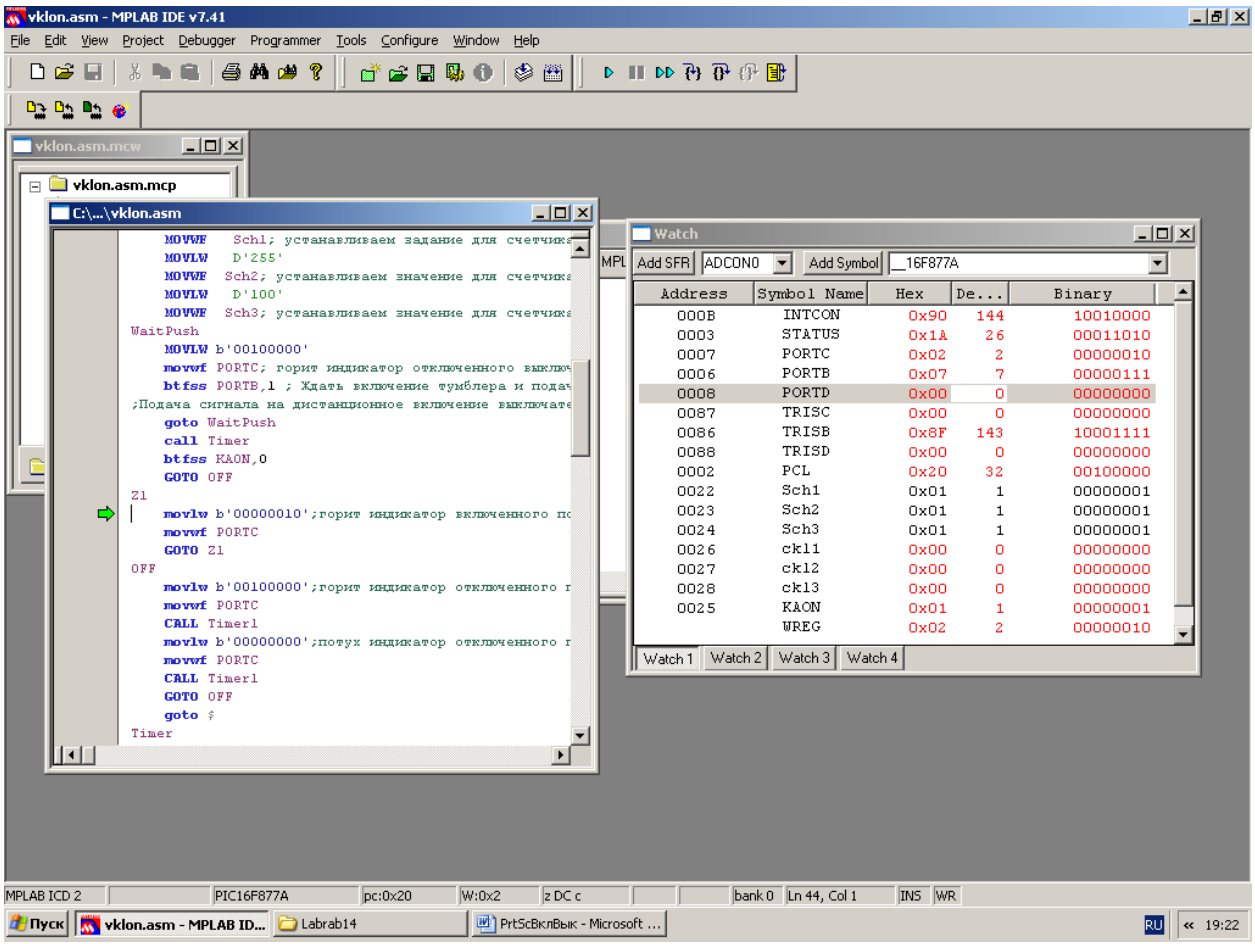


Рисунок 7.2 - Выключатель включился, т.к. сигнал включения конечного выключателя пришел с допустимой задержкой времени

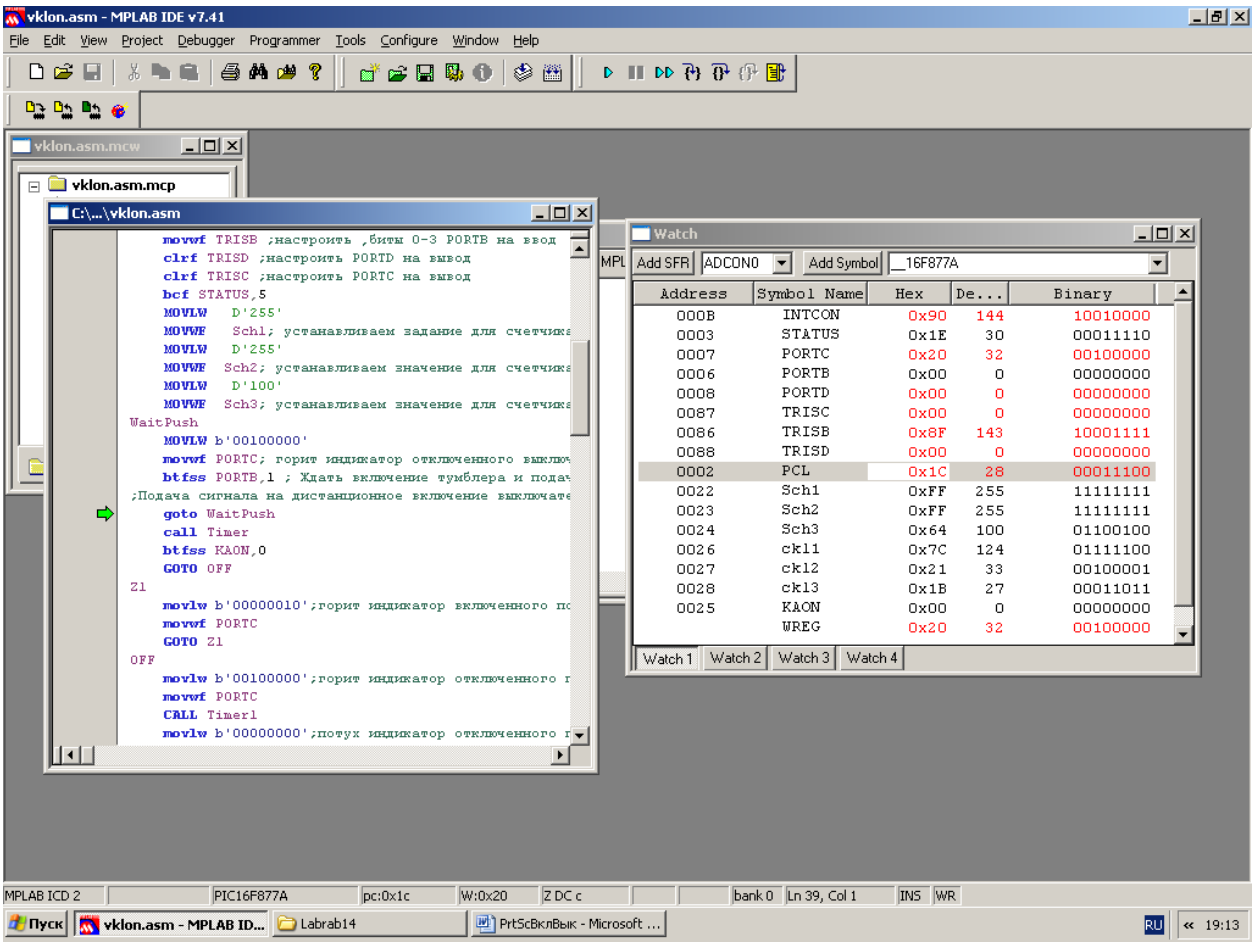


Рисунок 7.3 - Выключатель отключен. Сигнал на включение не подавался

Текст программы лабораторной работы № 7.

```
include "p16f877A.inc"
Sch1EQU H'22'; регистр хранения числа циклов в первом цикле
Sch2EQU H'23' ; регистр хранения числа циклов во втором цикле
Sch3EQU H'24' ; регистр хранения числа циклов в третьем цикле
KAON    EQU H'25';фиксация включения
ckl1 EQU h'26'
ckl2 EQU h'27'
ckl3 EQU h'28'
      org h'00'
      GOTO Nach
      org h'04'
      GOTO Prer
Nach
      clrf STATUS
      clrf PORTC ; Очистка регистра порта C
      clrf PORTB ; Очистка регистра порта B
      CLRF KAON;очистка регистра фиксации включения
      MOVLW b'10010000'
```

```
        MOVWF INTCON; разрешаем все прерывания и внешние прерывания
        bsf STATUS,5
        movlw B'00000111'
        movwf TRISA;биты 0-2 порта A на ввод
        movlw B'00001111'
        movwf TRISB ;настроить ,биты 0-3 PORTB на ввод
        clrf TRISD ;настроить PORTD на вывод
        clrf TRISC ;настроить PORTC на вывод
        bcf STATUS,5
        MOVLW   D'255'
        MOVWF   Sch1; устанавливаем задание для счетчика 1
        MOVLW   D'255'
        MOVWF  Sch2; устанавливаем значение для счетчика 2
        MOVLW   D'100'
        MOVWF  Sch3; устанавливаем значение для счетчика 3
WaitPush
        MOVLW b'00100000'
        movwf PORTC; горит индикатор отключенного выключателя
        btfss PORTB,1 ; Ждать включение тумблера и подачи 1 в RB1
;Подача сигнала на дистанционное включение выключателя
        goto WaitPush
        call Timer
        btfss KAON,0
        GOTO OFF
Z1
        movlw b'00000010';горит индикатор включенного положения
;выключателя
        movwf PORTC
        GOTO Z1
OFF
        movlw b'00100000';горит индикатор отключенного положения
;выключателя
        movwf PORTC
        CALL Timer1
        movlw b'00000000';потух индикатор отключенного положения
;выключателя
        movwf PORTC
        CALL Timer1
        GOTO OFF
        goto $
Timer
        MOVF Sch3,w;
        MOVWF  ckl3; значение W является аргументом для 3-го цикла таймера.
N3   MOVF Sch2,w;
```

```
      MOVWF  ckl2; значение W является аргументом для 2-го цикла таймера.
N2    MOVF Sch1,w;
      MOVWF  ckl1; значение W является аргументом для 1-го цикла таймера.
N1    DECFSZ ckl1,F; уменьшаем значение переменной ckl1 на 1 для счетч.1
      GOTO N1; при нулевом результате выполняем следующую инструкцию
      DECFSZ ckl2,F; уменьшаем значение переменной ckl2 на 1 для счетч.2
      GOTO N2; если счётчик обнулился, пропускаем GOTO.
      DECFSZ ckl3,F; уменьшаем значение переменной ckl3 на 1 для счетч.3
      GOTO N3; если счётчик обнулился, выходим из программы.
      RETURN; конец подпрограммы Timer2.
Prer
      BCF INTCON,1 ;снят флаг внешнего прерывания
;устанавливаем задания для Timer, которые вызовут его остановку и переход к
;выполнению следующей инструкции
      MOVLW   D'1'
      MOVWF   Sch1; устанавливаем задание для счетчика 1
      MOVLW   D'1'
      MOVWF  Sch2; устанавливаем значение для счетчика 2
      MOVLW   D'1'
      MOVWF  Sch3; устанавливаем значение для счетчика 3
      MOVLW b'00000001'
      MOVWF KAON;пришел сигнал о включении выключателя
      RETFIE
Timer1
      MOVLW D'15';
      MOVWF  ckl3; значение W является аргументом для 3-го цикла таймера.
N6    MOVLW D'255';
      MOVWF  ckl2; значение W является аргументом для 2-го цикла таймера.
N5    MOVLW D'255';
      MOVWF  ckl1; значение W является аргументом для 1-го цикла таймера.
N4    DECFSZ ckl1,F; уменьшаем значение переменной ckl1 на 1 для счетч.1
      GOTO N4; при нулевом результате выполняем следующую инструкцию
      DECFSZ ckl2,F; уменьшаем значение переменной ckl2 на 1 для счетч.2
      GOTO N5; если счётчик обнулился, пропускаем GOTO.
      DECFSZ ckl3,F; уменьшаем значение переменной ckl3 на 1 для счетч.3
      GOTO N6; если счётчик обнулился, выходим из программы.
      RETURN; конец подпрограммы Timer1.
      End
```

7.3 Оформление отчета по лабораторной работе

Отчет оформляется на группу. В созданный Word-файл копируется текст программы и два окна: при не включившемся выключателе и при включившемся.

Контрольные вопросы

1. На какой бит поступает внешнее прерывание?
2. В каком регистре и какие биты разрешают внешнее прерывание?
3. В каком состоянии находится бит GIE в регистре INTCON перед началом работы программы обработки прерывания?
4. В каком состоянии находится бит GIE в регистре INTCON после выполнения команды RETFIE?
5. Какие прерывания относятся к внешним?
6. Что происходит при возникновении разрешенного прерывания?
7. Назовите причины появления задержки отработки сигнала включения выключателя и сигнала подтверждения включения выключателя на удаленной ПС.
8. Почему программа, содержащая под программу обработки прерывания, начинается с безусловного перехода на адрес больше пятого?

8 Лабораторная работа № 8. Преобразование аналоговых сигналов. АЦП

Цель работы: изучение операций преобразования аналоговых сигналов в двоичное значение с помощью АЦП микроконтроллера, применение таймера TMR0. Процесс преобразования аналогового значения (напряжения, силы тока) в двоичное значение происходит в цифровых измерительных приборах, электросчетчиках, терминалах РЗ и ПА и т.д.

8.1 Теоретические сведения

Модуль аналого-цифрового преобразователя (АЦП) имеет восемь каналов у 40/44-выводных микросхем. Входной аналоговый сигнал через коммутатор каналов заряжает внутренний конденсатор АЦП C_{hold}. Модуль АЦП преобразует напряжение, удерживаемое на конденсаторе C_{hold} в соответствующий 10-разрядный цифровой код методом последовательного приближения. Источник верхнего и нижнего опорного напряжения может быть программно выбран с выводов V_{DO}, V_{SS}, RA2 или RA3.

Допускается работа модуля АЦП в SLEEP режиме микроконтроллера, при этом в качестве источника тактовых сигналов должен быть выбран RC генератор.

Для управления АЦП в микроконтроллере используется 4 регистра:

- регистр результата ADRESH (старший бит);
- регистр результата ADRESL (младший бит);
- регистр управления ADCON0;
- регистр управления ADCON1.

Регистр ADCON0 используется для настройки работы модуля АЦП, а с помощью регистра ADCON1 устанавливается какие входы микроконтроллера будут использоваться модулем АЦП и в каком режиме (аналоговый вход или цифровой порт ввода/вывода).

После включения и конфигурации АЦП выбирается рабочий аналоговый канал. Соответствующие биты TRIS аналоговых каналов должны настраивать порт ввода/вывода на вход. Перед началом преобразования необходимо выдержать временную паузу, расчет которой приведен в [11].

Рекомендованная последовательность действий для работы АЦП:

1) Настроить модуль АЦП:

- настроить выводы как аналоговые входы, входы V_{REF} или цифровые каналы ввода/вывода (ADCON1);
- выбрать входной канал АЦП (ADCON0);
- выбрать источник тактовых импульсов для АЦП (ADCON0);
- включить модуль АЦП (ADCON0).

2) Настроить прерывание от модуля АЦП (если необходимо):

- сбросить бит ADIF в 0 (регистр PIR1<6>);
- установить бит ADIE в 1(регистр PIE1<6>);
- установить бит PEIE в 1(регистр INTCON<6>);
- установить бит GIE в 1(регистр INTCON<7>).

3) Выдержать паузу, необходимую для зарядки конденсатора C_{HOLD}. Пауза обеспечивается с помощью таймера TMR0.

4) Начать аналого-цифровое преобразование:

- установить бит GO/-DONE в 1(ADCON0<2>).

5)Ожидать окончания преобразования :

- ожидать пока бит GO/-DONE не будет сброшен в 0; ИЛИ
- ожидать прерывание по окончанию преобразования.

6) Считать результат преобразования из регистров ADRESH:ADRESL, сбросить бит ADIF в 0 (регистр PIR1<6>), если это необходимо.

7)Для следующего преобразования необходимо выполнить шаги начиная с пункта 1 или 2. Время преобразования одного бита определяется как время T_{AD}. Минимальное время ожидания перед следующим преобразованием должно составлять не менее 2 T_{AD}.

8.2 Задания для внеаудиторной подготовки

Изучите назначение регистров ADCON0, ADCON1, ADRESH, ADRESL, INTCON, PIR1, PIE1, TRISA, PORTA, TRISE, PORTE, TMR0 (приложения Ж и И).

8.3 Исходные данные и задание

На рисунке 6.1 показана схема соединения разъемов и бит регистров для выполнения лабораторной работы. Аналоговый сигнал подается с выхода

делителя на бит RA0 PORTA. Сигнал внешнего прерывания на бит RB0 PORTB подается от бита 1 PORTD через схему с двумя тумблерами, один из которых включен при запуске программы, а другой выключен. Он включается для подачи сигнала прерывания на бит RB0, в результате чего происходит выход из цикла и установка следующего значения для обработки в АЦП.

Вывод двоичного значения сигнала после преобразования аналогового сигнала в АЦП из PORTC осуществляется на лампы 0-7 (рисунок 8.1).

Биты PORTC и RORTD настраиваются на вывод, биты 0-2 PORTA и биты 0-3 PORTB - на ввод.

АЦП настраивается на работу от основного генератора частоты с пределителем Fosc/8.

TMR0 запускается с предделителем 1:256. Время его работы обеспечивает зарядку конденсатора в АЦП до номинального уровня. В окне наблюдения выведите все регистры, состояние которых меняется в процессе выполнения программы.

8.4 Выполнение упражнения

Установите соединения в соответствии со схемой на рисунке 8.1. Откройте готовый проект АЦП: Project>Open>acp. При необходимости восстановите соединение с УМК-7: Debugger>Connect. Запишите программу на МК: Debugger>Program. Установите задатчик напряжения в требуемое значение. Запустите программу нажав на Run. Высветится двоичное число на индикаторах. Нажмите Halt. После остановки программы сохраните окно наблюдения с выведенными значениями регистров на момент остановки. Эти значения соответствуют состоянию, когда программа ждет внешнего прерывания. В PORTC выведено значение в двоичном виде преобразованного значения напряжения. Сравните значение по индикаторам со значением в PORTC. Некоторые лампочки могут не гореть. Сохраните окно в файле Word. Нажмите на Reset и верните программу в исходное состояние. Установите новое значение напряжения и повторите измерение. На основании нескольких измерений постройте характеристику, связывающую значение напряжения и значение числа в регистре PORTC (в десятичном виде).

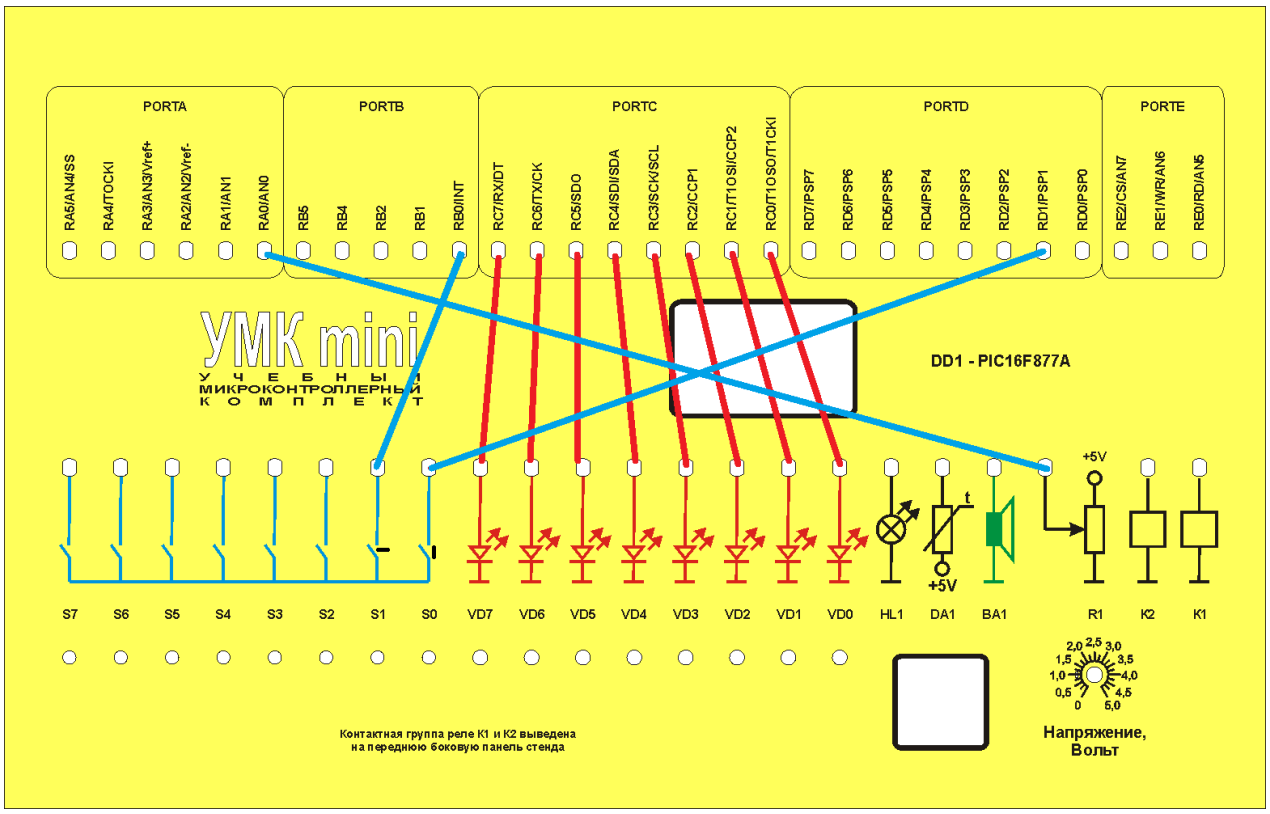


Рисунок 8.1 – Схема соединений в лабораторной работе № 8

Таблица 8.1 - Задания к лабораторной работе № 8

Номер вар-та	1	2	3	4	5
Положение задатчика	4,5 3,5 2,5	4,0 3,0 2,0	3,5 2,5 1,5	3,0 2,0 1,0	2,5 1,5 0,5

Текст программы лабораторной работы № 8.

```
include "p16F877A.inc"
org h'00'
nop
nop
nop
org h'05'
CLRF STATUS
CLRF PORTC ; Очистка регистра PORTC
movlw B'01000001' ; Настройка АЦП. Частота Fosc/8
movwf ADCON0
bsf STATUS,5
movlw B'00000111'
movwf TRISA;биты 0-2 PORTA на ввод
```

```
        movlw B'00001111'
        movwf TRISB ;настроить ,биты 0-3 PORTB на ввод
        clrf TRISD ;настроить PORTD на вывод
        clrf TRISC ;настроить PORTC на вывод
        movlw B'10000111' ;Установка предделителя перед TMR0
        movwf OPTION_REG ; коэффициент деления 1:256
        movlw B'00001110' ;эта константа вводится в регистр ADCON1
;для настройки АЦП – левое выравнивание и выбор бита RA0 в PORTA
;для ввода аналогового сигнала
        movwf ADCON1
        bcf STATUS,5
        movlw B'00000111'
        movwf PORTD; подаем энергию в PORTD
Main
        btfss INTCON,T0IF ; Ждать переполнения TMR0 – идет зарядка
        ;конденсатора C_HOLD
        goto Main
        bcf INTCON,T0IF ;Сбросить флаг прерывания от TMR0
        bsf ADCON0,GO ; Запуск АЦП
Wait
        btfss PIR1,ADIF ; Ждать окончания преобразования
        goto Wait
        movf ADRESH,W ;Вывод результата преобразования
        movwf PORTC ; на светодиоды порта C
WaitPush; после загорания светодиодов нажать Halt, снять PrtSc, затем
;нажать Reset, установить новое напряжение и запустить Run
        btfss PORTB,0 ; цикл организован для остановки после вывода
;результата измерения в PORTC
        goto WaitPush
        end
```

8.5 Оформление отчета по лабораторной работе

Отчет оформляется на группу. На основании измерений постройте характеристику, связывающую значение напряжения и значение числа в регистре PORTC (в десятичном виде).

Контрольные вопросы

1. Сколько каналов ввода аналоговых значений имеет микроконтроллер PIC16F877A?
2. Каким методом преобразуется аналоговое значение в цифровое в модуле АЦП микроконтроллера PIC16F877A?

3. Какие регистры используются для управления АЦП в микроконтроллере PIC16F877A?

4. Для чего нужна пауза перед подключением канала АЦП в микроконтроллере PIC16F877A?

5. Сколько должно быть T_{AD} для 10-разрядного преобразования в микроконтроллере PIC16F877A?

6. В каких регистрах сохраняется результат 10-разрядного преобразования в АЦП в микроконтроллере PIC16F877A?

7. Почему нельзя отлаживать программу в режиме MPLabSim?

8. Поясните «левое» и «правое» выравнивание в регистрах ADRESH и ADRESL.

9. Как создается ожидание окончания преобразования в АЦП?

9 Лабораторная работа № 9. АПВ ВЛ

Цель работы: исследовать работу программы, реализующей алгоритм автоматического повторного включения выключателя высоковольтной линии после его отключения РЗ на основе использования моделирующих возможностей стенда УМК-7.

9.1 Теоретические сведения и выполнение программы

Программа, реализующая упрощенный вариант работы АПВ ВЛ в нормальном режиме ожидает поступление внешнего сигнала, сигнализирующего об отключении выключателя ВЛ. Сигнал отключения выключателя имитируется на УМК-7 подачей единичного сигнала с помощью кратковременного включения тумблера S1 на рисунке 9.1. Энергия подается на бит RB1 PORTB, что вызывает запуск программы Timer2, которая с выдержкой времени в соответствии с заданием подает в определенный разряд PORTC на определенное время энергию. Этот сигнал по телемеханике передается на ПС. Номер разряда выбирается по заданию. Время паузы АПВ, создаваемой таймером Timer2, необходимо рассчитать, исходя из тактовой частоты 20 МГц. Время таймера Timer1определяет задержку, необходимую для включения выключателя и исчезновения сигнала отключения выключателя, пришедшего по телемеханике. В процессе выполнения программы снять окно PrtSc при успешном и неуспешном АПВ.

Создание паузы между выполнением следующими друг за другом инструкциями в программе может быть выполнено запуском в этом промежутке подпрограммы таймера. В микроконтроллере PIC16F877A имеются три модуля таймеров: TMR0, TMR1 и TMR2. В данной программе используется таймер, создаваемый программно, с использованием вложенных циклов. Изменением числа вложенных циклов можно создать довольно широкий спектр временных задержек.

Таблица 9.1- Варианты задания для лабораторной работы № 9

№ вар.	1	2	3	4	5	6	7	8	9
Тапв, сек	0,21	0,41	0,52	0,62	0,78	0,88	0,93	0,98	1,04
№ бит вкл.	1	2	3	4	5	6	7	0	1
№ бит «успешно»	1	1	1	1	0	0	0	0	3
№ бит «неуспешно»	3	4	5	7	3	4	5	7	1

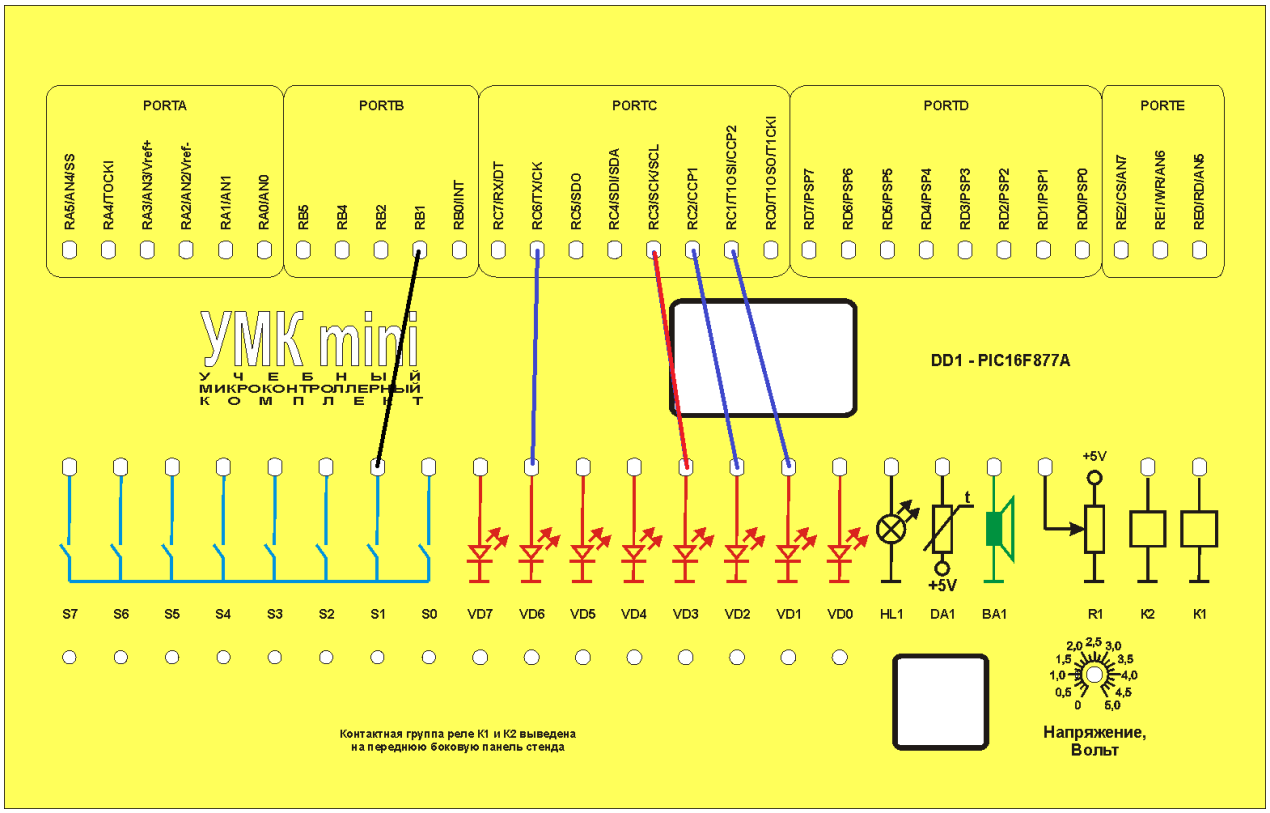


Рисунок 9.1 - Схема соединений для лабораторной работы № 9. Включение выключателя на бит 3. Свой вариант установить по заданию

9.2 Пример расчета времени паузы АПВ

Задание: разработать программу, реализующую упрощенный вариант работы однократного АПВ ВЛ со следующими исходными данными:

- время паузы 0,1 сек (100000 мкс);
- сигнал на включение выключателя ВЛ выдается с 3-го бита PORTC;
- сигнал отключения выключателя имитировать подачей энергии на бит 1 PORTB;
- задержку времени, необходимую для запуска схемы включения выключателя принять равную 200 подциклам Timer1.

В лабораторной работе № 4 было подсчитано, что на первом цикле мы можем создать максимально задержку в 205 мкс. Определение числа циклов и подциклов выполняем последовательно. Вначале проверим, сколько раз надо повторить первый цикл, чтобы создать требуемую задержку

$$N_{второго\ внешнего\ цикла} = 100000/205=487.$$

Число подциклов второго внешнего цикла максимально 255. Подсчитаем, примерно, время задержки, которая получится, если на втором цикле прокрутится 255 раз первый цикл:

$$255 \cdot 205 = 52275 \text{ мкс.}$$

Определим, примерно, число подциклов третьего цикла, в котором на каждом цикле будет прокручиваться полных два нижних цикла:

$$100000/52275 = 2.$$

Приведенный расчет дает нам возможность рассчитывать примерное число подциклов третьего цикла для создания определенной паузы. Предположим, что нам задана выдержка 1,14 сек. Тогда приближенное количество подциклов третьего цикла определится по формуле:

$$1140/\ 52=22.$$

Текст программы лабораторной работы № 9.

```
include "p16f877A.inc"
Sch_in     EQU  H'22'; счетчик внутреннего цикла задержки.
Sch_out   EQU  H'24' ; счетчик внешнего цикла задержки.
Sch3EQU  H'23' ; счетчик третьего цикла задержки.
apv  EQU h'25' ;кратность АПВ
colpovtor EQU h'26'; счетчик АПВ
      org h'00'
      nop
      nop
      nop
      org h'05'
      CLRF STATUS
      CLRF PORTC ; очистка регистра PORTC
      BSF STATUS,5
      MOVLW B'00001111'
      MOVWF TRISB ;настроить биты 0-3 PORTB на ввод
      CLRF TRISC; настроить PORTC на вывод
      BCF STATUS,5
```

```
        MOVLW b'00000001'
        MOVWF apv ;АПВ однократное, настройка
        MOVLW b'01000000'
      MOVWF PORTC ;АПВ включено
      CLRF colpovtor
Nach
WaitPush
      BTFSS PORTB,1 ; Ждать включение тумблера S1 и подачи 1 в RB1
      CALL  Ustan
      MOVF colpovtor,w
      SUBWF apv,w; нулевой результат означает, что выключатель
;отключился РЗ повторно, АПВ неуспешно
      BTFSS STATUS,Z
      CALL Timer2
M1   MOVLW b'00000100'
      MOVWF PORTC
      ; отключение сигнализации "Сработала АПВ Л-113"
      ;включение сигнализации "АПВ Л-113 неуспешно"
      ;АПВ отключено
      GOTO M1;остановить программу (Halt-Reset), включить(Run) после
включения выключателя
      GOTO $
Timer1
      MOVLW D'1';для задания 200. Это время длительности сигнала на
;на включение выключателя
      MOVWF Sch_in; устанавливаем значение внутреннего счетчика.
N_in
      DECF Sch_in,F; уменьшаем значение счетчика Sch_in на 1.
      BTFSS STATUS,Z; если счетчик Sch_in обнулился, пропускаем GOTO.
      GOTO N_in; срабатывает только при Z=0
      RETURN
Timer2
      MOVLW d'1'; задание 2
      MOVWF  Sch3; значение W является аргументом для таймера.
N3
      MOVLW   D'1';для задания 255
      MOVWF  Sch_out; устанавливаем значение внешнего счетчика.
N_out; метка внешнего счетчика.
      MOVLW   D'1';для задания 255
      MOVWF   Sch_in; устанавливаем значение внутреннего счетчика.
Nin; метка внутреннего счетчика.
      DECFSZ Sch_in,F; уменьшаем значение счетчика Sch_in на 1.
      GOTO Nin; срабатывает только при ненулевом результате.
      DECFSZ Sch_out,F; уменьшаем значение счетчика Sch_out  на 1.
```

```
    GOTO N_out;  срабатывает только при ненулевом результате.
    DECFSZ Sch3,F; уменьшаем значение счетчика Sch3 на 1,
    GOTO N3;  срабатывает только при ненулевом результате.
    BSF PORTC,3; послан сигнал на включение выключателя Л-113
    BSF PORTC,1; включение сигнализации "Сработала АПВ Л-113"
    CALL Timer1; выдержка на отработку включения выключателя Л-113
    BCF PORTC,3; снят сигнал на включение выключателя Л-113
    INCF colpovtor,1
    GOTO Nach
    RETURN; конец подпрограммы Timer2.
Ustan
    CLRF colpovtor
    GOTO WaitPush
    RETURN
    End
```

9.2 Оформление отчета по лабораторной работе

Отчет оформляется на группу. В отчет включаются два окна: с успешным и неуспешным АПВ.

Контрольные вопросы

1. Для чего применяется АПВ при отключении выключателей ВЛ?
2. Поясните работу программы при отключенном тумблере, имитирующим отключение выключателя.
3. В реальных условиях какое устройство выполняет функции тумблера?
4. В каких случаях бит Z в регистре STATUS равен 1?
5. Как работает инструкция BTFSC?
6. Почему в программе после неуспешного АПВ организован цикл, из которого можно выйти только нажатием кнопки Halt?
7. Поясните работу программы при успешном АПВ.
8. Как выполняется инструкция DESFSZ?

10 Лабораторная работа № 10. Автоматическое регулирование напряжения

Цель работы: исследовать работу программы, имитирующей на УМК-7 работу регулятора напряжения типа АРПН. Регулятор поддерживает автоматически напряжение в заданном диапазоне.

10.1 Выполнение упражнения

Установите соединения в соответствии со схемой на рисунке 10.1. Функции двигателя, перемещающего анцапфы в сторону увеличения или уменьшения напряжения, осуществляем вручную с помощью корректора. Если загорается лампа, соответствующая разряду из PORTC на «прибавить», то поворачиваем корректор в сторону увеличения напряжения. Если загорается лампа, соответствующая разряду из PORTC на «убавить», то поворачиваем корректор в сторону уменьшения напряжения. Когда установленное напряжение Utek будет в диапазоне Umin<Utek<Umax, то цель достигнута. Значения Umin и Umax, исходное значение корректора Uкор и количество циклов Sch_in для таймера выбираются из таблицы 10.2 в соответствии с номером задания. Для определения направления изменения значения корректора от первоначально заданного необходимо пользоваться таблицей 10.1. В ней приведены значения в двоичном виде преобразованных положений корректора. Если исходное значение корректора меньше диапазона Umin>Uкор<Umax, то на первом шаге надо увеличить это значение, если оно больше Umin<Uкор>Umax, то надо его уменьшить.

Таблица 10.1 – Двоичные значения корректора

Положение корректора	Двоичное значение после АЦП
4,5	11110101
4,0	11011100
3,5	10111110
3,0	10011001
2,5	10000000
2,0	01100100
1,5	00111111
1,0	00100011
0,5	00001000

Таблица 10.2 - Исходные данные для лабораторной работы № 10

№ варианта	Umin	Umax	Sch_in	Uкор	(+)	(-)
1	10011001	10111110	100	2,5	4	2
2	00100011	00111111	110	2,0	2	3
3	10000000	10011001	130	2,0	5	4

Откройте на компьютере, подключенном к УМК-7, готовый проект. При необходимости восстановите соединение с УМК-7: Debugger>Connect. В соответствии с заданием измените значения переменных Umin, Umax и количество циклов Sch_in. Скомпилируйте проект Project>Make. Создайте

окно наблюдения. Окно наблюдения включает регистры Umin, Utek, Umax, UMENSH, Sch_in и PORTC. Запишите программу на МК: Debugger>Program. Установите корректор напряжения на требуемое значение. Запустите программу, нажав на Run. Засветится на индикаторах лампа, соответствующая указанию «прибавить» или «убавить». Нажмите Halt. Снимите окно и сохраните. Запустите программу опять: Reset, далее Run. Медленно поворачивая корректор в сторону, соответствующую указанию пока не погаснет лампа. Нажмите Halt. Снимите окно и сохраните. Вы установили напряжение в соответствии с заданием. На рисунках 10.2 и 10.3 приведены PrtSc для данного примера.

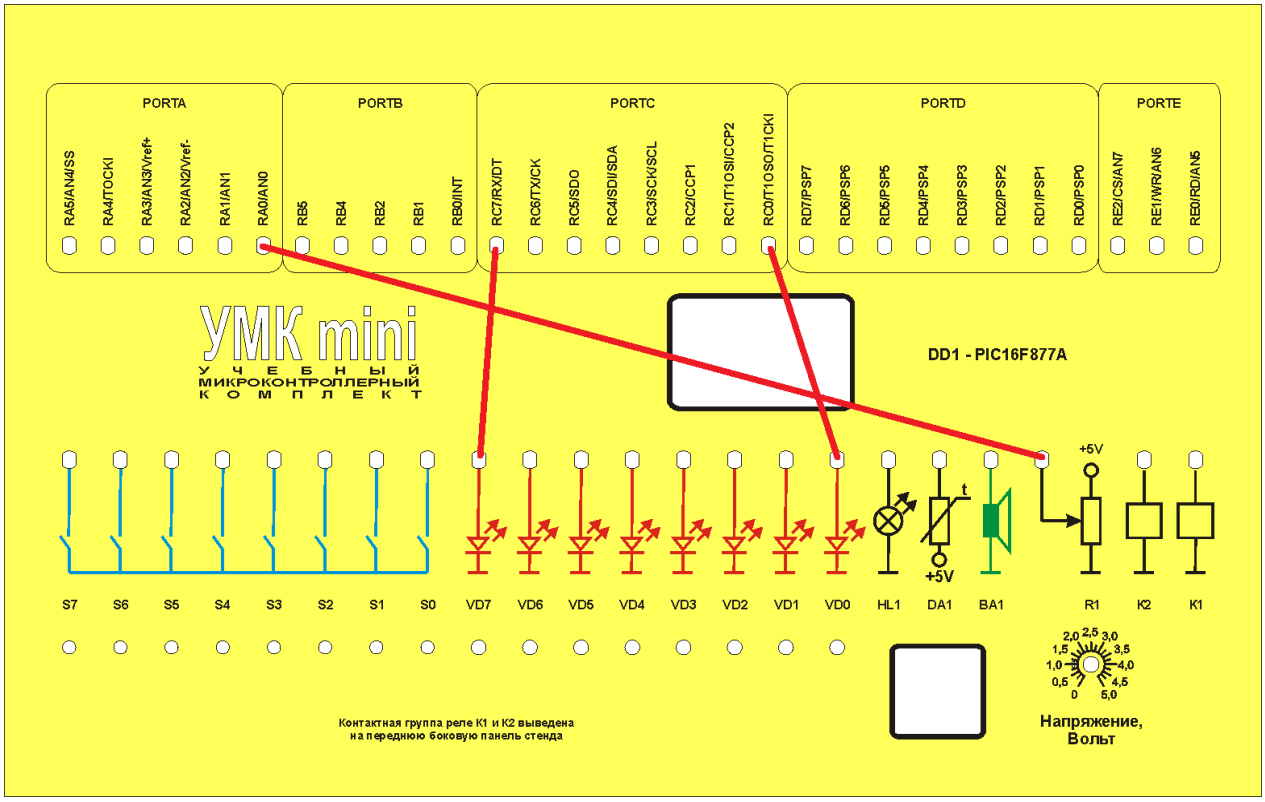


Рисунок 10.1 – Схема соединений на УМК-7 для лабораторной работы № 10

Текст программы лабораторной работы № 10.

```
      include "p16F877A.inc"
Umin            EQU h'40'
Umax            EQU h'41
Utek          QU h'42'
UMENSH      EQU h'43'
Sch_in    EQU  h'44'; счетчик внутреннего цикла задержки.
      org h'00'
      nop
```

```
        nop
        nop
        org h'05'
        clrf STATUS
        clrf  PORTC ; Очистка регистра порта C
          movlw  B'10111110' ;значение Umax по заданию
        movwf  Umax
        movlw  B'10011001' ;значение Umin по заданию
        movwf  Umin
        movlw  B'01000001' ; Включение АЦП. Частота Fosc/8
        movwf  ADCON0
        bsf  STATUS,5
        movlw B'00000111'
        movwf  TRISA;биты 0-2 порта A на ввод
        ;movlw  B'00001111'
        ;movwf  TRISB ;настроить ,биты 0-3 PORTB на ввод
        ;clrf  TRISD ;настроить PORTD на вывод
        clrf  TRISC ;настроить PORTC на вывод
        movlw B'10000111' ;Установка предделителя перед TMR0
        movwf OPTION_REG ; Запуск TMR0 с коэффициентом деления 1:256
        movlw B'00001110'
        movwf  ADCON1 ; Настройка АЦП – левое выравнивание ,аналоговый
;канал RA0
        bcf STATUS,5
        ;movlw  B'00000111'
        ;movwf  PORTD
Main
        btfss INTCON,T0IF ; Ждать переполнения TMR0
        goto Main
        bcf INTCON,T0IF ;Сбросить флаг прерывания от TMR0
        bsf ADCON0,GO ; Запуск АЦП
Wait
        btfss PIR1,ADIF ; Ждать окончания преобразования, когда ADIF =1
        goto Wait
        movf ADRESH,W ;Вывод результата преобразования
        bcf STATUS,C
        movwf Utek
        movf Umax,W
        movwf UMENSH
        movf Utek,W
        subwf UMENSH,f; Umax - Utek
        btfss STATUS,C; Если C=0, то Utek больше Umax-надо уменьшить
;напряжение, если C=1, то надо проверить Umin - Utek
        call UMEN
```

```
        movf Umin,W
        movwf UMENSH
        movf Utek,W
        subwf UMENSH,f; Umin - Utek
        btfss STATUS,C; Если C=0, то Utek больше Umin, напряжение
;находится в допустимом диапазоне отклонений и его менять не надо
        call NORMA
        call UVEL
        goto$
UMEN
        movlw b'00000001'
        movwf PORTC
        call Timer
        goto Main
        return
UVEL
        movlw b'10000000'
        movwf PORTC
        call Timer
        goto Main
        return
NORMA
        movlw b'00000000'
        movwf PORTC
        call Timer
        goto Main
        return
Timer
        movlw D'2';     только для отладки, далее из задания
        movwf  Sch_in; устанавливаем значение внутреннего счетчика.
N_in
        decf Sch_in,F; уменьшаем значение счетчика Sch_in на 1.
        btfss STATUS,Z; если счетчик Sch_in обнулился, пропускаем GOTO.
        goto N_in; срабатывает только при Z=0
        return
        end
```

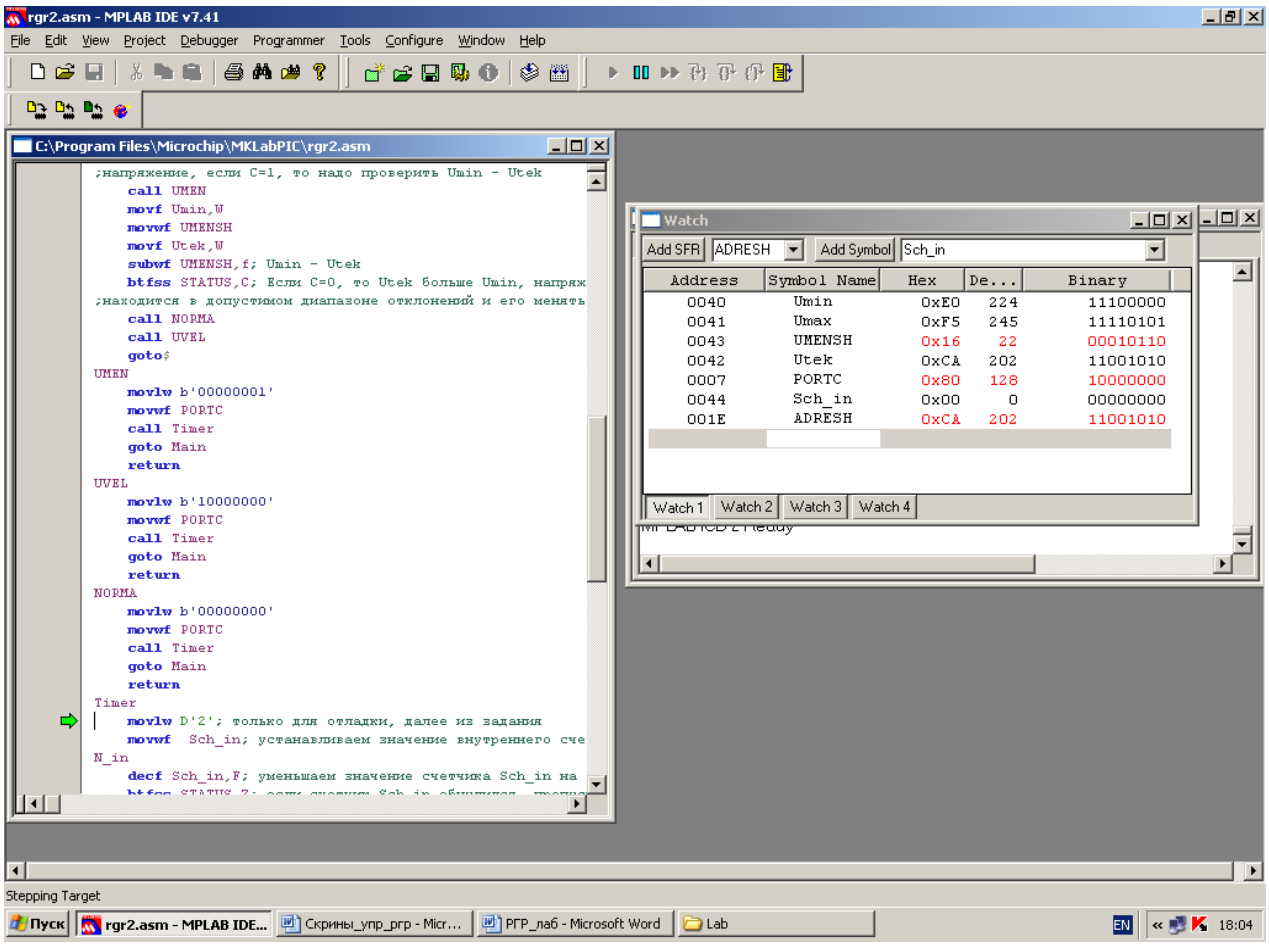


Рисунок 10.2 – Включение регулятора в сторону увеличения напряжения на бит 7 в соответствии с заданием

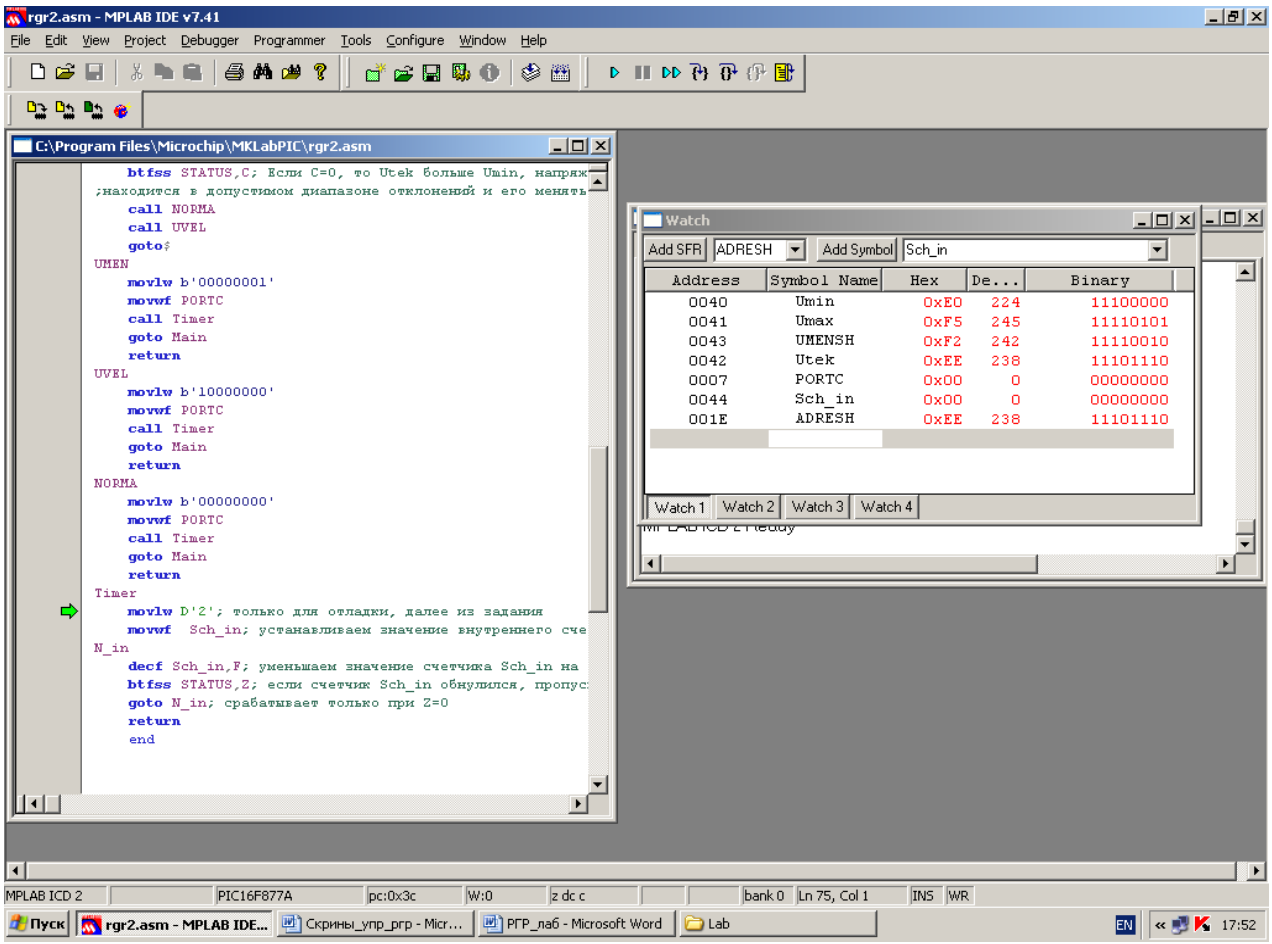


Рисунок 10.3 – Напряжение введено в допустимый диапазон. Сигналы на регулятор напряжения на PORTC отсутствуют

10.2 Оформление отчета по лабораторной работе

Отчет оформляется на группу. В отчет включаются два окна: стартовое и, когда напряжение находится в допустимом интервале напряжения.

Контрольные вопросы

1. Как определяется, что текущее положение корректора, соответствует напряжению, значение которого меньше минимального по заданию ?
2. Как определяется, что текущее положение корректора, соответствует напряжению, значение которого больше минимального по заданию ?
3. Как определяется, что текущее положение корректора, соответствует напряжению, значение которого больше максимального по заданию ?
4. В каких случаях бит Z в регистре STATUS равен 1?
5. Как можно реализовать цикл ожидания без использования инструкции BTFSS?
6. В каких случаях бит C в регистре STATUS равен 0?
7. Для чего используется таймер TMR0?

8. Как определяется окончание преобразования в АЦП и возможность считывания результата преобразования, выводимого в регистр ADRESH?

9. Что должна обеспечить выдержка времени, создаваемая TMR0?

Приложение А
Системы счислений

Количество цифр (символов) применяемых в системе называют ее основанием. Минимальный объем информации, который можно записать на носителе информации называют **бит**. Восемь носителей информации объединили в одну ячейку памяти, и назвали **байт.**

Т а б л и ц а А.1 - Запись чисел в различных системах счислений

Десятичная система	Двоичная система	Двоично-десятичная система	Шестнадцатеричная система
0	0000	0000	0
1	0001	0001	1
2	0010	0010	2
3	0011	0011	3
4	0100	0100	4
5	0101	0101	5
6	0110	0110	6
7	0111	0111	7
8	1000	1000	8
9	1001	1001	9
10	1010	0001 0000	A
11	1011	0001 0001	B
12	1100	0001 0010	C
13	1101	0001 0011	D
14	1110	0001 0100	E
15	1111	0001 0101	F
16	00010000	0001 0110	10
17	00010001	0001 0111	11
18	00010010	0001 1000	12
19	00010011	0001 1001	13
20	00010100	0010 0000	14

Примеры записи чисел в программе:

- D'07' – десятичное число;
- B'0111' – двоичное число;
- H'07'или 0x07 или 07h – шестнадцатеричное число.

Алгоритм перевода чисел из двоичной системы в шестнадцатеричную систему: сначала двоичное число разбиваем на четверки чисел справа налево, затем для каждой четверки записываем его эквивалент из приведенной таблицы А1. П р и м е р. $0101\ 1011\ 1101_2 = 5BD_{16}$.

Обратный перевод чисел очевиден, необходимо только аккуратно писать именно четверки двоичных чисел, дописывая при необходимости нули слева: $1_{16} = 0001_2$, $2_{16} = 0010_2$. П р и м е р. $415C_{16} == 0100\ 0001\ 0101\ 1100_2$.

Приложение Б

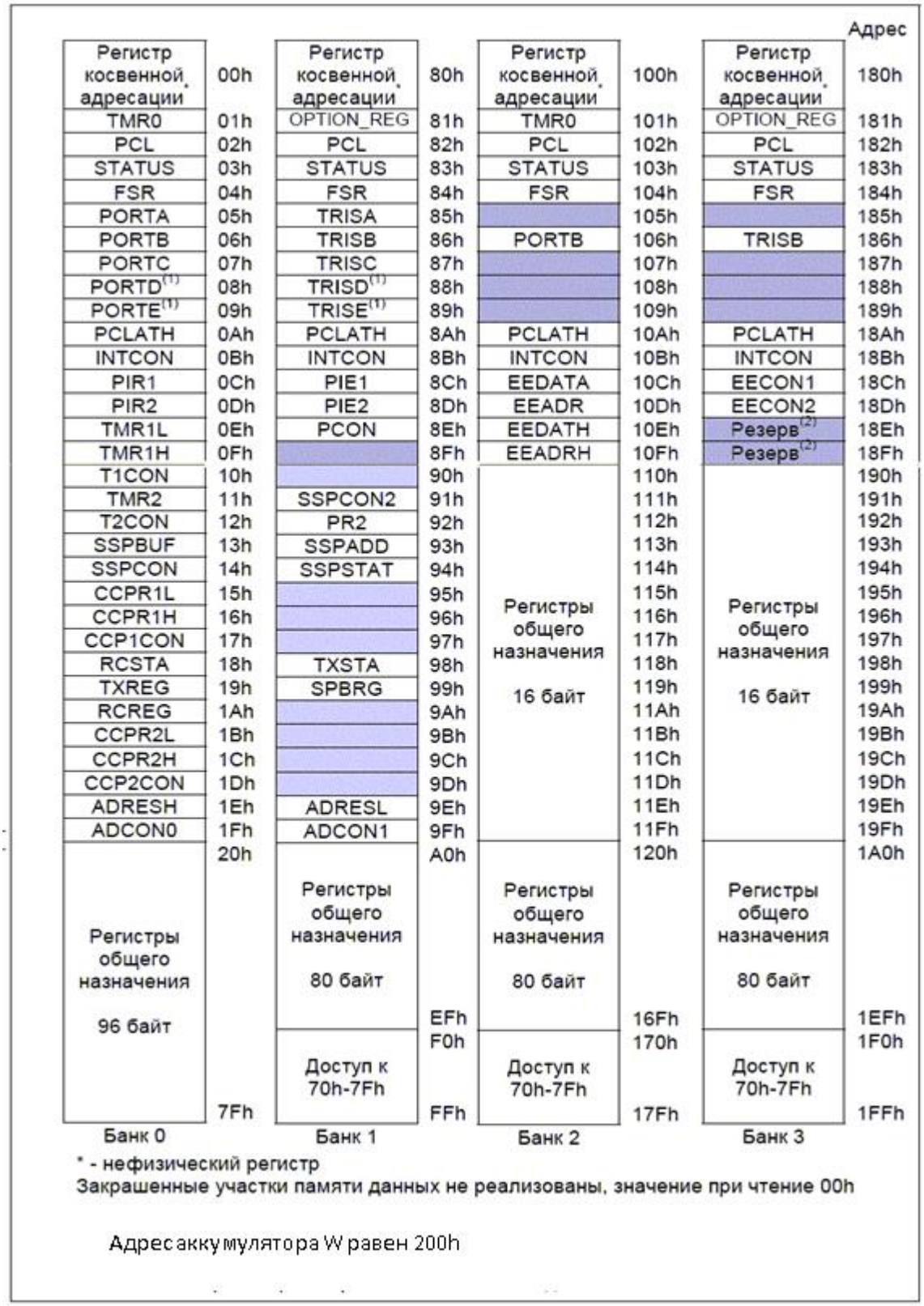


Рисунок Б.1 – Карта памяти данных микроконтроллеров PIC16F87x

Приложение В

Описание лабораторного комплекса УМК-7

Лабораторный комплекс УМК-7 создан для подготовки специалистов согласно современным требованиям, диктуемых развитием технологий в области: применения микропроцессоров и микроконтроллеров в системах (АР, СОИ, ДУ) технологических процессов. Помимо обучения языку ассемблер на примере легко программируемого контроллера PIC16F877A, студент ознакомится с внутренней и внешней структурой современных микроконтроллеров. Таким образом, комплект предназначен для изучения архитектурных и программных возможностей микропроцессоров.
Структурная схема УМК-7 представлена на рисунке В1.

Устройство программирования микроконтроллера, предназначено для ввода программы в микроконтроллер с ПК.

Выводы микроконтроллера непосредственно соединены с внешними разъёмами лабораторного комплекса.

Клеммные соединения (внешние разъёмы), предназначены для соединения выводов микроконтроллера с внешними устройствами.

К внешним устройствам относятся: внешние тумблеры, светодиоды, датчик температуры с нагревательным элементом, устройство звуковой сигнализации, источник регулируемого напряжения для АЦП, а также два внешних реле для управления внешними цепями до 12 Вольт и током до 0,5 Ампер.

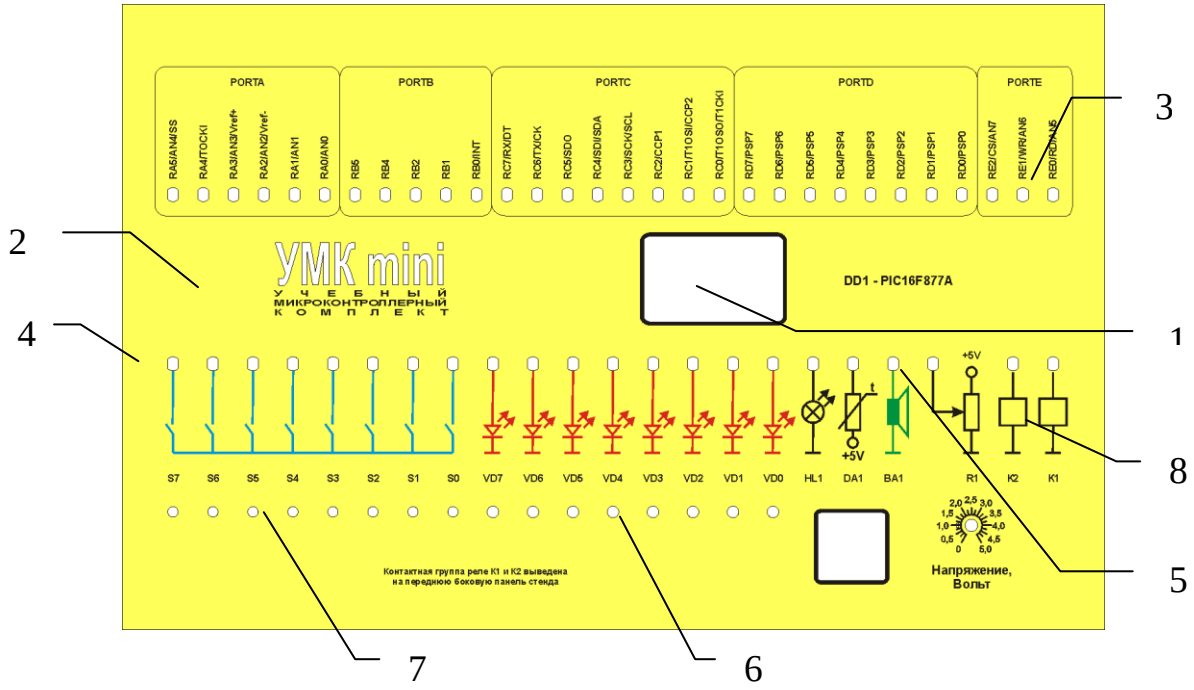


Рисунок В.1 - Передняя панель стенда

1 - микроконтроллер PIC16F877A;

2 – модуль MPLAB-ICD;

3 - ряд клеммных соединений (выводы PIC16F877A);

4 - ряд клеммных соединений (выводы внешних устройств);

5 - аналоговый выход (ограничение по току до 1 мА);

6 - световая сигнализация;

7 – тумблеры с выходом 0 или 5 Вольт (ограничение по току до 1 мА, верхнее положение тумблера соответствует наличию на выходе напряжения 5 Вольт);

8 – внешние реле К1 и К2.

Для того чтобы скоммутировать контроллер с внешними устройствами соедините нужные клеммы из ряда клеммных соединений контроллера с клеммами из ряда клеммных соединений внешних устройств, при помощи перемычек.

MPLAB-ICD – отладочный комплект для микроконтроллеров серии PIC16F87X. Используя возможность внутрисхемной отладки (ICD), встроенную в кристаллы PIC16F87X, а также протокол внутрисхемного последовательного программирования фирмы Microchip (ICSPTM), MPLAB-ICD – является программатором и внутрисхемным отладчиком одновременно. Он работает под управлением Интегрированной Среды разработки MPLAB IDE, подключается к Отлаживаемому Устройству и работает как микроконтроллер PIC16F87X.

MPLAB-ICD специально предназначен для помощи при ознакомлении и отладке кода в составе лаборатории.

MPLAB-ICD обеспечивает:

- пошаговое выполнение кода в реальном масштабе времени;
- точки останова (break points);
- внутрисхемная отладка;
- встроенное программирование;
- диапазон рабочих напряжений от 3.0V до 5.5V;
- рабочие частоты от 32 кГц до 20 МГц;
- интерфейс пользователя MPLAB;
- совместимость с Windows XXXX;
- RS-232 Интерфейс.

В целом на лабораторном комплексе УМК-7 проводятся следующие работы:

- изучение возможностей программной среды MPLAB;
- приобретение начальных навыков программирования, компиляции и выполнения простых программ для микроконтроллера PIC16F877A;
- ознакомление с внутренней и внешней структурой микроконтроллера PIC16F877A;

- освоение принципов соединения выводов микроконтроллера с внешними устройствами;
- приобретение навыков записи программы в микроконтроллер PIC 16F877A;
- исследование выполнения отдельных команд, простых программ;
- изучение операторов установки битов и операций сдвига;
- приобретение навыков работы с числами в шестнадцатеричном коде;
- изучение команд логических операций и переходов;
- знакомство с регистрами, изучение команд управления, байтовых логических операций, программных способов маскирования данных, использования косвенной адресации;
- закрепление навыков маскирования данных и организации условных переходов;
- закрепление навыков инициализации портов;
- исследование особенностей записи и обращения к подпрограммам;
- изучение методов использования стека при создании программ;
- закрепление навыков программирования вывода;
- изучение таймерных функций процессора, режимов работы таймера;
- обучение способам организации прерываний, инициализация различных видов прерываний;
- изучение принципов аналогово-цифрового преобразования, получение навыков сбора схемы с аналоговыми сигналами.

Приложение Г

Регистр Status

В таблице Г.1 показаны расположение и имена бит регистра Status.

Т а б л и ц а Г.1 – Биты регистра Status

Номера бит	7	6	5	4	3	2	1	0
Имена бит	IRP	RP1	RP0	-TO	-PD	Z	DC	C

В таблице Г.1 приняты следующие имена:

- IRP - бит выбора банка при косвенной адресации (IRP=1 - выбор банков 2 или 3, IRP=0 - выбор банка 0 или 1);

- RP1, RP0 - биты выбора банка при непосредственной адресации (т.е. разряды 5 и 6). Их значения для выбора банков показаны в таблице А.2;

- TO - флаг переполнения сторожевого таймера;

- PD - флаг включения питания;

- Z - флаг нулевого результата. Записывается “1” при нулевом результате арифметической или логической операции;

- DC - флаг десятичного переноса или заема. Записывается “1”, если был перенос из младшего полубайта регистра в старший полубайт, актуально при выполнении команд сложения и вычитания в двоично-десятичной системе;

- C - флаг переноса или заема. Записывается “1”, если был перенос из старшего бита регистра для команд сложения и вычитания. Вычитание выполняется с помощью сложения уменьшаемого и вычитаемого, которое представлено в дополнительном коде.

Т а б л и ц а Г.2 – Выбор номера банка при непосредственной адресации

RP1(первый бит номера банка)	RP0 (нулевой бит номера банка)	Номер банка (десятичная система)
0	0	0
0	1	1
1	0	2
1	1	3
Примечание: RP1 и RP0 образуют разряды двоичного числа		

К отдельным битам регистра можно обращаться по имени или по номеру. Инструкция BTFSC STATUS,Z эквивалентна инструкции BTFSC STATUS,2.

Т а б л и ц а Г.3 – Значения регистра Status после команды CLRF

IRP	RP1	RP0	-TO	-PD	Z	DC	C
0	0	0	1	1	1	1 или 0	1 или 0

Приложение Д

Описание инструкций МК PIC16F87x

Константа в инструкциях представлена символом k. В описании инструкций указаны флаги, которые могут измениться при ее выполнении. Параметр d (dest) в инструкциях определяет, в какой регистр записывается результат. Если d=0 или отсутствует, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f **(**от слова file**)**. В программах для параметра d вместо 0 можно писать W, вместо 1 можно писать f.

Вместо адреса регистра в программе более удобно писать его символьное имя. Например, если регистр по адресу h'21' имеет имя R1, тогда инструкция, описанная как ADDWF f, d, в программе может быть записана двумя способами: ADDWF h'21' , W или ADDWF R1, W.

Директива ORG h'xx' – это указатель для ассемблера, что код, следующий за этим выражением, будет записан в память программ по этому адресу.

ADDLW - Сложить k с W.
Синтаксис: [*label*] ADDLW k.
Операнды: 0≤k≤255.
Операция: (W) + k → (W).
Изменяемые флаги: C, DC, Z.
Описание: Содержимое регистра W складывается с 8-разрядной константой **k**. Результат сохраняется в регистре W.
ADDWF - Сложение W и f.
Синтаксис: [*label*] ADDWF f, d.
Операнды: 0≤f≤127.
Операция: (W) + (f) → (dest).
Изменяемые флаги: C, DC, Z.
Описание: Сложить содержимое регистров W и f. Если d=0, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f.
ANDLW - Побитное И константы k и W.
Синтаксис: [*label*] ANDLW k.
Операнды: 0≤k≤255.
Операция: (W) .AND. k → (W).
Изменяемые флаги: Z.
Описание: выполняется побитное И содержимого регистра W и 8-разрядной константы k. Результат сохраняется в регистре W.
ANDWF – Побитное И W и f.
Синтаксис: [*label*] ANDWF f, d.
Операнды: 0≤f≤127.
Операция: (W) .AND. (f) → (dest).
Изменяемые флаги: Z.

Описание: выполняется побитное И содержимого регистра W и f. Результат сохраняется в регистре W. Если d=0, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f.
BCF - очистить бит b в регистре f.
Синтаксис: [*label*] BCF f, b.
Операнды: 0≤f≤127; 0≤b≤7.
Операция: 0 → (f<b>).
Изменяемые флаги: нет.
Описание: очистить бит b в регистре f.
BSF - Установить бит b в регистре f.
Синтаксис: [*label*] BSF f, b.
Операнды: 0≤f≤127; 0≤b≤7.
Операция: 1→f<b>.
Изменяемые флаги: нет.
Описание: установить бит b=1 в регистре f.
BTFSC - Проверить бит b в регистре f, пропустить следующую инструкцию, если b=0.
Синтаксис: [*label*] BTFSC f, b.
Операнды: 0≤f≤127; 0≤b≤7.
Изменяемые флаги: нет.
Описание: если бит b в регистре f равен 1, то выполняется следующая инструкция. Если бит b в регистре f равен 0, то следующая инструкция не выполняется, команда выполняется за два цикла. Во втором цикле выполняется пустая инструкция NOP.
BTFSS - Проверить бит b в регистре f**,** пропустить следующую инструкцию, если b = 1.
Синтаксис: [*label*] BTFSS f, b.
Операнды: 0≤f≤127; 0≤b≤7.
Изменяемые флаги: нет.
Описание: если бит b в регистре f равен 0, то выполняется следующая инструкция. Если бит b в регистре f равен 1, то следующая инструкция не выполняется, команда выполняется за два цикла. Во втором цикле выполняется пустая инструкция NOP.
CALL - Вызов подпрограммы.
Синтаксис: [*label*] CALL k.
Операнды: 0≤k≤2047.
Операция: (PC) + 1 →TOS, k→PC<10:0>, (PCLATH<4:3>)→PC<12:11>.
Изменяемые флаги: нет.
Описание: вызов подпрограммы. Адрес следующей инструкции (PC+1) помещается в вершину стека. Одиннадцать бит адреса загружаются из кода команды в счетчик команд PC<10 :0>. Два старших бита загружаются в счет-

чик команд PC<12:11> из регистра PCLATH. Команда CALL выполняется за два цикла.
CLRF - Очистить f.
Синтаксис: [*label*] CLRF f.
Операнды: 0≤f≤127.
Операция: 00h → (f); 1 →Z.
Изменяемые флаги: Z.
Описание: очистить содержимое регистра f и установить флаг Z=1 в регистре STATUS<2>.
CLRW - Очистить W.
Синтаксис: [*label*] CLRW.
Операнды: нет.
Операция: 00h → (W); 1 →Z.
Изменяемые флаги: Z.
Описание: очистить содержимое регистра W и установить флаг Z=1 в регистре STATUS<2>.
CLRWDT - Очистить WDT.
Синтаксис: [*label*] CLRWDT.
Операнды: нет.
Операция: 00h → (WDT), 00h → предделитель WDT, 1 →-TO, 1 →-PD.
Изменяемые флаги: -TO, -PD.
Описание: инструкция CLRWDT сбрасывает WDT и предделитель, если он подключен к WDT. В регистре STATUS устанавливаются в единицу биты –TO и –PD.
COMF - Инвертировать f.
Синтаксис: [*label*] COMF f, d.
Операнды: 0≤f≤127.
Операция: (-f) → (dest).
Изменяемые флаги: Z.
Описание: инвертировать все биты в регистре **f**. Если d=0, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f.
DECF - Вычесть 1 из f.
Синтаксис: [*label*] DECF f, d.
Операнды: 0≤f≤127.
Операция: (f) -1 → (dest).
Изменяемые флаги: Z.
Описание: декрементировать содержимое регистра f. Если d=0, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f.
DECFSZ - Вычесть 1 из f и пропустить следующую инструкцию, если результат равен 0.
Синтаксис: [*label*] DECFSZ f, d.

Операнды: 0≤f≤127.
Операция: (f) -1 → (dest); пропустить следующую инструкцию, если результат равен 0.
Изменяемые флаги: нет.
Описание: декрементировать содержимое регистра f. Если d=0, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f. Если результат не равен 0, то исполняется следующая инструкция. Если результат равен 0, то следующая инструкция не выполняется, команда выполняется за два цикла. Во втором цикле выполняется NOP.
GOTO - Безусловный переход.
Синтаксис: GOTO k.
Операнды: 0≤k≤2047.
Операция: k →PC<10:0>, (PCLATH<4:3) →PC<12:11>.
Изменяемые флаги: нет.
Описание: выполнить безусловный переход. Одиннадцать бит адреса загружаются из кода команды в счетчик команд PC<10:0>. Два старших бита загружаются в счетчик команд PC<12:11> из регистра PCLATH. Команда GOTO выполняется за два цикла.
INCF - Прибавить 1 к f.
Синтаксис: [*label*] INCF f, d.
Операнды: 0≤f≤127.
Операция: (f) + 1 → (dest).
Изменяемые флаги: Z.
Описание: инкрементировать содержимое регистра f. Если d=0, результат сохраняется в регистре W. Если d = 1, результат сохраняется в регистре f.
INCFSZ - Прибавить 1 к f и пропустить следующую инструкцию, если результат равен 0 (при переполнении)
Синтаксис: [*label*] INCFSZ f, d.
Операнды: 0≤f≤127.
Операция: (f) + 1 → (dest).
Изменяемые флаги: нет.
Описание: инкрементировать содержимое регистра f. Если d=0, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f.
Если результат не равен 0, то исполняется следующая инструкция. Если результат равен 0, то следующая инструкция не выполняется, команда выполняется за два цикла. Во втором цикле выполняется NOP.
IORLW - Побитное 'ИЛИ' константы k и W.
Синтаксис: [*label*] IORLW k.
Операнды: 0≤k≤255.
Операция (W).OR.(k) → (W).
Изменяемые флаги: Z.

Описание: выполняется побитное 'ИЛИ' содержимого регистра W и 8-разрядной константы k. Результат сохраняется в регистре W.
IORWF - Побитное 'ИЛИ' W и f.
Синтаксис: [*label*] IORWF f, d.
Операнды: 0≤f≤ 127.
Операция: (W).OR.(f) → (dest).
Изменяемые флаги: Z.
Описание: выполняется побитное ИЛИ содержимого регистров W и f. Если d=0, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f.
MOVF - Переслать f.
Синтаксис: [*label*] MOVF f, d.
Операнды: 0≤f≤127.
Операция: (f) → (dest).
Изменяемые флаги: Z.
Описание: содержимое регистра f пересылается в регистр адресата, определяемого значением d. Если d=0, то результат сохраняется в W. Если d=1, то значение сохраняется в регистре f. Используется для проверки содержимого регистра f на ноль.
MOVLW - Переслать k в W.
Синтаксис: [*label*] MOVLW k.
Операнды: 0≤k≤255.
Операция: k→(W).
Изменяемые флаги: нет.
Описание: в неиспользуемых битах ассемблер устанавливает 0.
MOVWF - Переслать W в f.
Синтаксис: [*label*] MOVWF f.
Операнды: 0≤f≤127.
Операция: (W) → (f).
Изменяемые флаги: нет.
Описание: переслать содержимое регистра W в регистр f .
NOP - Нет операции.
RETFIE – Возврат из подпрограммы с разрешением прерывания.
Операнды: нет.
Операция: TOS→PC, 1→GIE в регистре INTCON.
Изменяемые флаги: нет.
Описание: возврат из подпрограммы обработки прерываний. Вершина стека TOS загружается в счетчик команд PC. Устанавливается в '1' флаг глобального разрешения прерываний GIE (INTCON<7>). Инструкция выполняется за два цикла.
RETURN - Возврат из подпрограммы.

Синтаксис: [*label*] RETURN.
Операнды: нет.
Операция: TOS → PC.
Изменяемые флаги: нет.
Описание: возврат из подпрограммы. Вершина стека TOS загружается в счетчик PC. Инструкция выполняется за 2 цикла.
RLF - Циклический сдвиг регистра **f** влево через бит C регистра **Status.**
Синтаксис: [*label*] RLF f, d.
Операнды: 0≤f≤127.
Изменяемые флаги: C.
Описание: выполняется циклический сдвиг влево содержимого регистра f через бит C регистра Status. Если d=0, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f.

RRF - Циклический сдвиг регистра f вправо через бит C регистра status.
Синтаксис: [*label*] RRF f, d.
Операнды: 0≤f≤127.
Изменяемые флаги: C.
Описание: выполняется циклический сдвиг вправо содержимого регистра f через бит C регистра Status. Если d=0, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f.

SLEEP Перейти в режим SLEEP.
Синтаксис: [*label*] SLEEP.
Операнды: нет.
Операция: 00h → WDT, 00h → предделитель WDT, 1 → -TO, 0 → PD в регистре STATUS<4:3>.
Изменяемые флаги: -TO, -PD.
Описание: сбросить флаг включения питания -PD в '0'. Установить флаг -TO переполнения WDT в '1'. Очистить таймер WDT и его предделитель. Перевести микроконтроллер в режим SLEEP и выключить тактовый генератор.
SUBLW - Вычесть W из k.
Синтаксис: [*label*] SUBLW k.

Операнды: 0≤k≤255.
Операция: k - (W) → (W).
Изменяемые флаги: C, DC, Z.
Описание: вычесть содержимое регистра W из 8-разрядной константы k. Результат сохраняется в регистре W.
SUBWF - Вычесть W из f.
Синтаксис: [*label*] SUBWF f, d.
Операнды: 0≤f≤127; d∈ [0,1].
Операция: (f) –(W) → (dest).
Изменяемые флаги: C, DC, Z.
Описание: вычесть содержимое регистра W из регистра f. Если d=0, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f.
SWAPF - Поменять местами полубайты в регистре f.
Синтаксис: [*label*] SWAPF f, d.
Операнды: 0≤f≤127.
Операция: (f<3:0>) → (dest<7:4>), (f<7:4>) → (dest<3:0>).
Изменяемые флаги: нет.
Описание: поменять местами старший и младший полубайты регистра f. Если d=0, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f.
XORLW- Побитное исключающее ИЛИ k и W.
Синтаксис: [*label*] XORLW k.
Операнды: 0≤k≤255.
Операция: (W).XOR.k → (W).
Изменяемые флаги: Z.
Описание: выполняется побитное исключающее ИЛИ содержимого регистра Wи 8-разрядной константы k. Результат сохраняется в регистре W.
XORWF- Побитное исключающее ИЛИ W и f.
Синтаксис: [*label*] XORWF f, d.
Операнды: 0≤f≤127.
Операция: (W).XOR.(f) → (dest).
Изменяемые флаги: Z.
Описание: выполняется побитное исключающее ИЛИ содержимого регистра W и f. Если d=0, результат сохраняется в регистре W. Если d=1, результат сохраняется в регистре f.

Приложение Е

Модуль таймера TMR1

TMR1 - 16-разрядный таймер/счетчик, состоящий из двух 8-разрядных регистров (TMR1H и TMR1L), доступных для чтения и записи. Счет выполняется в спаренных регистрах (TMR1H:TMR1L), инкрементируется их значение от 0000h до FFFFh. При добавлении ещё единицы будет переполнение регистров и в счетчиках будет снова 0000h. При переполнении счетчика устанавливается в 1 бит флага прерывания TMR1IF в регистре PIR1<0>. Само прерывание можно разрешить/запретить установкой/сбросом бита TMR1IE в регистре P1E1<0>.

TMR1 может работать в режимах: режим таймера, режим счетчика.

Включается TMR1 установкой бита TMR1ON в 1 (T1CON<0>).

Битом TMR1CS (T1CON<1>) выбирается источник тактовых импульсов.

TMR1 инкрементируется при каждом машинном цикле.

Когда включен генератор тактовых импульсов (T1OSCEN=1), выводы RC1/T1OSI/CCP2 и PC0/T1OSO/TICK1 настроены как входы. Значение битов TRISC<1:0> игнорируется, а чтение данных с этих выводов дает результат 0.

Управляющие биты TMR1 находятся в регистре T1CON.

Регистры TMR1H и TMR1L не сбрасываются в 00h при сбросе по включению питания и других видах сброса.

Предделитель TMR1 очищается при записи чисел в регистр TMR1L или TMR1H.

Регистр PIE1 (адрес 8Ch) доступен для чтения и записи, содержит биты разрешения периферийных прерываний. Чтобы разрешить периферийные прерывания необходимо установить в '1' бит PEIE (INTCON<6>).

Регистр INTCON (адрес 0Bh, 8Bh, 10Bh или 18Bh) доступен для чтения и записи, содержит биты разрешений и флаги прерываний:

- переполнение TMR1;
- изменения уровня сигнала на выводах PORTB;
- внешний источник прерываний RB0/INT.

Регистр PIR1 доступен для чтения и записи, он содержит флаги прерываний периферийных модулей.

Флаги прерываний устанавливаются при возникновении условий прерываний вне зависимости от соответствующих битов разрешения и бита общего разрешения прерываний GIE (INTCON<7>). Программное обеспечение пользователя должно сбрасывать соответствующие флаги при обработке прерываний от периферийных модулей.

Приложение Ж

Таймер TMR0

Таймер TMR0 – 8-разрядный таймер/счетчик с предделителем. Чем больше коэффициент предделителя, тем медленнее заполняется счетчик таймера TMR0. После переполнения счетчика (значение в регистре TMR0 равно FFh) происходит прерывание, в результате чего в регистре INTCON бит TOIF (бит 2) устанавливается в 1. Само прерывание может быть разрешено/запрещено установкой/сбросом бита TOIE (INTCON<5>). При запрете прерывания программа не останавливается, но флаг прерывания от TMR0 TOIF (INTCON<2>) появляется, и он должен быть сброшен в подпрограмме обработки прерывания или в основной программе. Для включения таймера и получения прерывания от него при переполнении счетчика таймера используются регистры OPTION_REG и INTCON. Связь регистров и бит с TMR0 показана в таблице Ж.1.

Таблица Ж.1- Связь регистров и бит с TMR0

Адрес	Имя	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
01,101	TMR0	Регистр таймера TMR0							
0B,8B, 10B,18B	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
81,181	OPTION_ REG	RBPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0

Затененные биты не используются.

1 Регистр OPTION_REG

Регистр OPTION_REG доступен для чтения и для записи. Он содержит биты: конфигурации предделителя (PSC) для TMR0/WDT, внешнего прерывания INT и состояния выходов порта B. Описание битов регистра OPTION_REG показано в таблице Ж.2.

Таблица Ж.2 - Регистр OPTION_REG (адреса 81h, 181h)

№ бита	7	6	5	4	3	2	1	0
Имя бита	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
после сброса	1	1	1	1	1	1	1	1
доступность	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Обозначения: R – читаемый разряд, W – записываемый разряд.								

Назначение битов регистра OPTION_REG приведено ниже.

Бит 7: -RBPU: включение подтягивающих резисторов на PORTB.

0 = подтягивающий резистор включен.

1 = подтягивающий резистор выключен.

Примечание: если используется низкий уровень напряжения при программировании по входу PGM и выводы PORTB подтянуты к высокому уровню, то 3-ий бит в регистре TRISB необходимо сбросить, чтобы вход RB3 не был подтянут к высокому уровню. Это необходимо для правильного программирования устройства.

Бит 6: INTEDG: выбор активного фронта сигнала прерывания на входе RB0/INT регистра PORTB.

0 = прерывание по заднему фронту.

1 = прерывания по переднему фронту.

Бит 5: TOCS: выбор источника тактового сигнала для TIMER0.

1 = тактовый сигнал с входа RA4/T0CKI.

0 = внутренний источник тактового сигнала (CLKOUT).

Бит 4: T0SE: выбор фронта приращения TMR0 при внешнем тактовом сигнале.

1 = приращение по заднему фронту сигнала на T0CKI.

0 = приращение по переднему фронту сигнала на T0CKI.

Бит 3: PSA: выбор включения предделителя.

1 = предделитель включен перед сторожевым таймером WDT.

0 = предделитель включен перед таймером TMR0.

Биты 2-0: PS2 - PS0: выбор коэффициента предделителя. Чем больше коэффициент предделителя, тем больше время паузы, обеспечиваемой таймером. Коэффициенты деления предделителя показаны в таблице Ж.3.

Таблица Ж.3- Коэффициенты деления в зависимости от значений PS2- PS0

Значение			TMR0 (PSA = 0)	WDT (PSA = 1)
PS2	PS1	PS0		
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

Установка коэффициента деления предделителя 1:1 для TMR0 соответствует переключению предделителя на сторожевой таймер.

2 Регистр INTCON

Регистр INTCON доступен для чтения и для записи. Он содержит биты масок прерываний и флаги прерываний. Флаги прерываний должны сбрасываться программно. Регистр показан в таблице Ж.4.

Таблица Ж.4 - Регистр INTCON (адреса 0Bh, 8Bh, 10Bh, 18Bh)

№ бита	7	6	5	4	3	2	1	0
Имя бита	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
после сброса	0	0	0	0	0	0	0	0
доступность	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Обозначения: R – читаемый разряд, W – записываемый разряд.								

Назначение битов регистра INTCON приведено ниже.
Бит 7: GIE – Общее (глобальное) управление прерываниями.
1=все немаскируемые прерывания разрешены.
0=все прерывания запрещены.
Примечание: если происходит прерывание, то бит GIE сбрасывается. По команде выхода из подпрограммы (RETFIE) этот бит устанавливается.
Бит 6: PEIE – маска прерываний от периферийных устройств.
1=все немаскируемые прерывания периферийных устройств разрешены.
0=все прерывания периферийных устройств запрещены.
Бит 5: T0IE – маска прерывания по переполнению TMR0.
1=прерывание TMR0 разрешено.
0= прерывание TMR0 запрещено.
Бит 4: INTE – маска внешнего прерывания по входу RB0/INT.
1=прерывание по входу RB0/INT разрешено.
0=прерывание по входу RB0/INT запрещено.
Бит 3: RBIE – маска прерывания по изменению состояния на входах RB7: RB4 PORTB.
1=прерывание по изменению уровня сигнала на входах RB7: RB4 PORTB разрешено.
0=прерывание по изменению уровня сигнала на входах RB7: RB4 PORTB запрещено.
Бит 2: T0IF – флаг прерывания при переполнении TMR0.
1=устанавливается, если регистр TMR0 переполнен (очищается программно).

0=если регистр TMR0 не переполнен.
Бит 1: INTF – флаг внешнего прерывания по входу RB0/INT.
1=устанавливается, если происходит прерывание по входу RB0/INT.
0=если прерывание по входу RB0/INT не произошло.
Бит 0: RBIF – флаг прерывания по изменению уровня сигнала на входах RB7: RB4 PORTB.
1=устанавливается, если изменился уровень сигнала на одном из входов RB7:RB4 PORTB (очищается программно).
0=если уровень сигнала на входах RB7:RB4 не изменился.

Приложение И

Модуль АЦП

Модуль аналого-цифрового преобразователя (АЦП) имеет восемь каналов у 40/44-выводных микросхем.

Входной аналоговый сигнал через коммутатор каналов заряжает внутренний конденсатор АЦП C_{hold}. Модуль АЦП преобразует напряжение, удерживаемое на конденсаторе C_{hold} в соответствующий 10-разрядный цифровой код методом последовательного приближения. Источник верхнего и нижнего опорного напряжения может быть программно выбран с выводов V_{DO}, V_{SS}, RA2 или RA3.

Допускается работа модуля АЦП в SLEEP режиме микроконтроллера, при этом в качестве источника тактовых сигналов должен быть выбран RC генератор.

Для управления АЦП в микроконтроллере используется 4 регистра.

- регистр результата ADRESH (старший бит);
- регистр результата ADRESL (младший бит);
- регистр управления ADCON0;
- регистр управления ADCON1.

Регистр ADCON0 используется для настройки работы модуля АЦП, а с помощью регистра ADCON1 устанавливается какие входы микроконтроллера будут использоваться модулем АЦП и в каком режиме (аналоговый вход или цифровой порт ввода/вывода).

Таблица И.1- Регистр ADCON0 (адрес 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/-DONE	-	ADON
Бит 7							Бит 0

Биты 7-6: ADCS1: ADCS0: выбор источника тактового сигнала.

Время получения одного бита результата определяется параметром T_{AD}. Для 10-разрядного результата требуется как минимум 12 T_{AD}. Параметры тактового сигнала для АЦП определяются программно, T_{AD} может принимать следующие значения:

00= $2T_{OSC}$;
01= $8T_{OSC}$;
10= $32T_{OSC}$;
11= время такта внутреннего генератора RC модуля АЦП.

Биты 5-3: CHS2: CHS0: Выбор аналогового сигнала:
000 = канал 0, (RA0/AN0);
001 = канал 1, (RA1/AN1);
010 = канал 2, (RA2/AN2);

011 = канал 3, (RA3/AN3);
100 = канал 4, (RA4/AN4);
101 = канал 5, (RA5/AN5);
110 = канал 6, (RA6/AN6);
111 = канал 7, (RA7/AN7).

Бит 2: GO/-DONE: бит статуса модуля АЦП. Бит ADON = 1.

1= модуль АЦП выполняет преобразование (установка бита вызывает начало преобразования).

0=состояние ожидания (аппаратно сбрасывается по завершению преобразования).

Бит 1 Не используется: читается как 0.

Бит 0 ADON: бит включения модуля АЦП.

1= модуль АЦП включен.

0=модуль АЦП выключен и не потребляет тока.

Сброс бита GO/-DONE в 0 во время преобразования в регистре ADCON0 приведет к прекращению преобразования. При этом регистры результата (ADRESH-ADRESL) не изменит своего содержимого. После досрочного завершения преобразования необходимо обеспечить временную задержку $2T_{AD}$. Выдержав требуемую паузу, можно начать новое преобразование установкой бита GO/-DONE в 1.

На рисунке И.1 показана последовательность получения результата после установки бита GO/-DONE в 1.

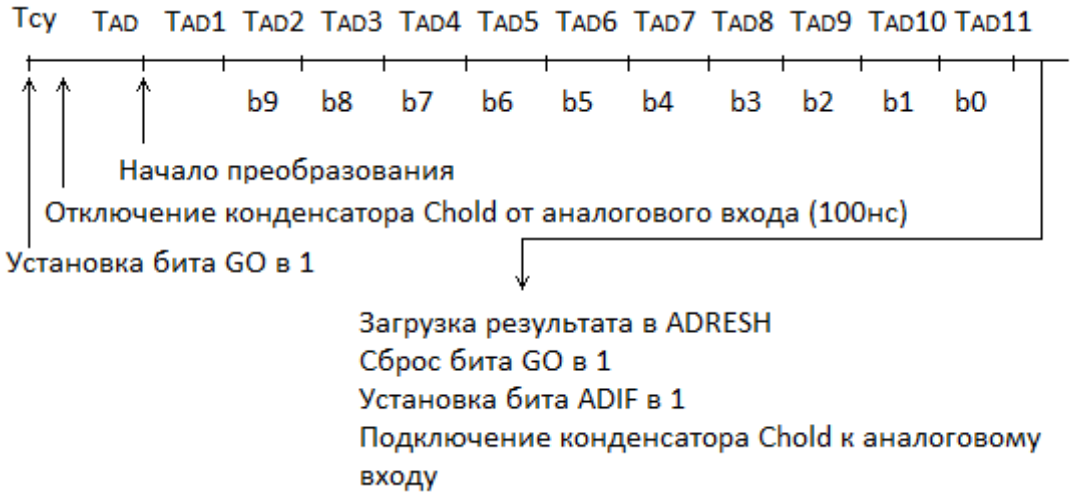


Рисунок И.1 - Последовательность получения результата после установки бита GO/-DONE в 1

Таблица И.2 - Регистр ADCON1 (адрес 9Fh)

R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0
Бит 7							Бит 0

Бит 7: ADFM: формат сохранения 10-разрядного результата.
1= правое выравнивание, 6 старших бит ADRESH читаются как 0.
0= левое выравнивание, 6 младших бит ADRESL читаются как 0.
Биты 6-4: не используются, читаются как 0.
Биты 3-0: PCFG3:PCFG0: управляющие биты настройки каналов АЦП.

Таблица И.3 - Настройка разрядов в регистрах PORTA и PORTE и входов опорного напряжения

PCGF3: PCGF0	Каналы АЦП Настройка разряда в регистрах при установке комбинации								V_{REF-}	V_{REF+}	Кан./ V_{REF} (1)
	AN7 RE2	AN6 RE1	AN5 RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0			
0000	A	A	A	A	A	A	A	A	V_{DD}	V_{SS}	8/0
0001	A	A	A	A	V_{REF+}	A	A	A	RA3	V_{SS}	7/1
0010	D	D	D	A	A	A	A	A	V_{DD}	V_{SS}	5/0
0011	D	D	D	A	V_{REF+}	A	A	A	RA3	V_{SS}	4/1
0100	D	D	D	D	A	D	A	A	V_{DD}	V_{SS}	3/0
0101	D	D	D	D	V_{REF+}	D	A	A	RA3	V_{SS}	2/1
011x	D	D	D	D	D	D	D	D	V_{DD}	V_{SS}	0/0
1000	A	A	A	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	V_{DD}	V_{SS}	6/0
1010	D	D	A	A	V_{REF+}	A	A	A	RA3	V_{SS}	5/1
1011	D	D	A	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2	4/2
1100	D	D	D	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2	3/2
1101	D	D	D	D	V_{REF+}	V_{REF-}	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	V_{DD}	V_{SS}	1/0
1111	D	D	D	D	V_{REF+}	V_{REF-}	D	A	RA3	RA2	1/2

A – аналоговый вход, D – цифровой канал ввода/вывода, V_{REF+} - вход положительного опорного напряжения, V_{REF-} - вход отрицательного опорного напряжения.

В регистре ADRESH:ADRESL сохраняется 10-разрядный результат аналого-цифрового преобразования.

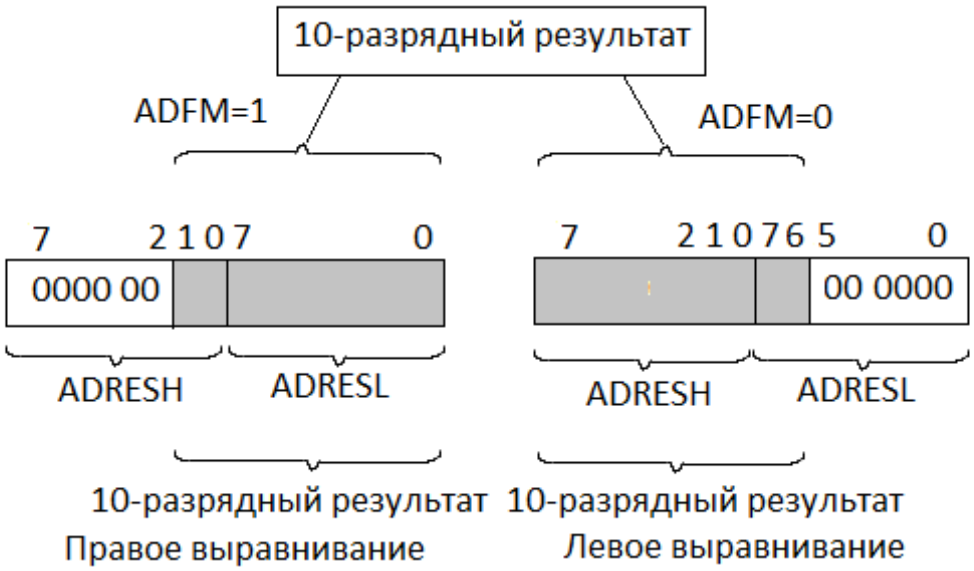


Рисунок И.4- Размещение результата двоичного преобразования в регистрах ADRESH:ADRESL

Когда преобразование завершено, результат преобразования записывается в регистр ADRESH:ADRESL, после чего сбрасывается флаг GO/-DONE (регистр ADCON0<2>) и устанавливается флаг прерывания ADIF в регистре PIR1.

После включения и конфигурации АЦП выбирается рабочий аналоговый канал. Соответствующие биты TRIS аналоговых каналов должны настраивать порт ввода/вывода на вход. Перед началом преобразования необходимо выдержать временную паузу, расчет которой приведен в [11].

Список литературы

1 Погребинский М.П. Микропроцессорные системы управления электротехническими установками. –М.: МЭИ, 2003.

2 Информатика. Базовый курс: Учебное пособие для вузов под ред. Симоновича С.В. - СПб.: Питер, 2003.

3 Сайт в Internet www.microchip.ru.

4 Катцен С. PIC микроконтроллеры. Все, что необходимо вам знать. – М.: Додека, 2008.

5 Копесбаева А.А. Микропроцессорные комплексы в системах управления: Учебное пособие. -Алматы, АИЭС, 2010.

6 Кохц Дитер. Измерение, управление с помощью PIC-контроллеров.- Киев: Наукова думка. 2007.

7 Заец Н.И. Радиолюбительские конструкции на PIC. –М.: Солон, 2003.

8 Яценков В.С. Микроконтроллеры Microchip. Практическое руководство. – 2 –е изд. исп. и допол. – М.: Горячая линия – Телеком, 2005.

9 Фрунзе А.В., Фрунзе М.А. Микроконтроллеры? Это же просто. - М.: ООО ИД СКИМЕН, 2003.

10 Дьяков А.Ф., Овчаренко Н.И. Микропроцессорная релейная защита и автоматика электроэнергетических систем.- М.: Издательство МЭИ, 2000.

11 Однокристальные 8-разрядные FLASH CMOS микроконтроллеры компании Microchip Technology Incorporatid. ООО «Микро-Чип».- Москва, 2002.

12 Новиков Ю.В. Введение в цифровую электронику: видеокурс. — М.: ИНТУИТ.РУ «Интернет-Университет Информационных Технологий», 2010.

13 Ю. А. Шпак Программирование на языке С для AVR и PIC микроконтроллеров - Издательство Корона-Век, МК-Пресс, 2011.

Св. план 2015 г., поз. 24

Акименков Михаил Вениаминович

ОСНОВЫ ЦИФРОВОЙ ТЕХНИКИ
(ОСНОВЫ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ)

Методические указания
к выполнению лабораторных работ
для студентов специальностей 5В071800 – Электроэнергетика и
5В081200-Энергообеспечение сельского хозяйства

Редактор Голева Н.М.
Специалист по стандартизации Молдабекова Н.К.

Подписано в печать _____ Формат 60х84 1/16
Тираж __ экз. Бумага типографская №1
Объем __ уч.- изд. л. Заказ ___ Цена_____ тг.

Копировально-множительное бюро
некоммерческого акционерного общества
«Алматинский университет энергетики и связи»
050013, Алматы, Байтурсынова, 126