МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)

ОДНОАДРЕСНЫЙ МИКРОКОНТРОЛЛЕР СЕМЕЙСТВА PIC

Рекомендовано редакционно-издательским советом федерального государственного автономного образовательного учреждения высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева» в качестве методических указаний для студентов Самарского университета, обучающихся по основной образовательной программе высшего образования по направлению подготовки 11.03.03 Конструирование и технология электронных средств

Составитель *В.В. Иванов*

С А М А Р А
Издательство Самарского университета
2019

УДК 004(075)
ББК 32.97я7

Составитель ***В.В. Иванов***

Рецензент канд. техн. наук, доц. кафедры радиотехники К.Е. В о р о н о в

Одноадресный микроконтроллер семейства PIC: метод. указания / сост. *В.В. Иванов*. – Самара: Изд-во Самарского университета, 2019. – 40 с.

Предназначены для студентов по направлению подготовки 11.03.03 Конструирование и технология электронных средств при изучении курса «Микропроцессорная техника».

Лабораторные работы охватывают разделы по архитектуре, системам команд, алгоритмам обращения к периферийным устройствам электронно-вычислительных машин на примере микроконтроллера семейства PIC.

Подготовлены на кафедре конструирования и технологии электронных средств.

УДК 004(075)
ББК 32.97я7

СОДЕРЖАНИЕ

Лабораторная работа № 1

КОМАНДЫ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ PIC-контроллера

Цель работы: изучение состава однокристального микроконтроллера с сокращенным набором команд на примере разработки и отладки программ с использованием арифметических операций.

Сведения из теории

Микроконтроллер работает с одноадресными командами (рис. 1.1). Первый операнд всегда находится в рабочем регистре WREG (так в PIC-контроллерах называют аккумулятор). В коде команды есть информация только о втором операнде. Это – либо *при прямой адресации* адрес второго операнда (ffffffff), либо *при непосредственной адресации* сам операнд (kkkkkkkk).

В команде можно адресовать лишь 256 ячеек, поэтому память разбита на банки по 256 регистров. Номер банка необходимо предварительно записать в регистр указателя банков BSR. Максимальное число банков – 16.

Каждая команда состоит из одного 16-разрядного слова, разделенного на 8-ми разрядный код операции (OPCODE) и 8-ми разрядов информации о втором операнде. В таблице 1.1 приведены переменные, входящие в команды.

Таблица 1.1. Параметры команд

Параметр	Символ (буква кодировки)	Диапазон значений
Константа, 1 байт	**k**	от 0 до 0x0FF
Адрес регистра	**Reg (ffffffff)**	от 0 до 0x0FF
Адресуемый бит	**Bit (bbb)**	от 0 до 7

Окончание табл. 1.1

Указатель приёмника результата	**d**	**0** – регистр W, **1** – регистр банка
Бит доступа к памяти	**a**	**0**: Обращение к ОЗУ быстрого доступа (значение регистра BSR игнорируется), **1**: Обращение к ОЗУ с учетом значения регистра BSR(указатель № банка)
Порт назначения	**Port (p)**	определяется типом контроллера

Практически все команды занимают одно слово в памяти программ, кроме четырех команд, занимающих два слова в памяти программ. Эти команды занимают два слова из-за большого числа операндов (команда 32 бита). Во втором слове 4 старших бита всегда равны ‘1’. Если второе слово команды будет выполнено как отдельная команда, то вместо каких-либо операций выполняется пустой цикл NOP.

Команды разделены на четыре основных группы:

- **Байт-ориентированные** команды
- **Бит-ориентированные** команды
- Операции с **константами**
- **Управляющие** команды

Байт-ориентированные команды (табл. 1.2 и рис. 1.1) имеют следующие переменные:

- Регистр (‘f’), к которому выполняется обращение
- Размещение результата команды (‘d’)
- Тип доступа к памяти (‘a’)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE						**d**	**a**	**f**	**f**	**f**	**f**	**f**	**f**	**f**	**f**

Рис. 1.1. Формат байт-ориентированной команды

Для байт-ориентированных команд **'f'** является указателем регистра, **'d'** - указателем адресата результата. Указатель регистра

определяет, какой регистр должен использоваться в команде. Указатель адресата определяет, где будет сохранен результат. Если **'d'=0**, результат сохраняется в регистре **W**. Если **'d'=1**, результат сохраняется в регистре, который используется в команде.

Таблица 1.2. Байт ориентированные команды с регистрами

Мнемоника	**Описание**	**Слово команды** (16 бит)
ADDWF f, d, a	*Сложение WREG и f*	0010 01da ffff ffff
ADDWFC f, d, a	*Сложение WREG, f и бита C*	0010 00da ffff ffff
SUBFWB f, d, a	*Вычитание f из WREG с заемом*	0101 01da ffff ffff
SUBWFB f, d, a	*Вычитание WREG из f с заемом*	0101 10da ffff ffff
SUBWF f, d, a	*Вычитание WREG из f*	0101 11da ffff ffff
SWAPF f, d, a	*Поменять местами полубайты*	0011 10da ffff ffff
INCF f, d, a	*Инкремент f*	0010 10da ffff ffff
DECF f, d, a	*Декремент f*	0000 01da ffff ffff
NEGF f, a	*Негативное значение f*	0110 110a ffff ffff
SETF f	*Установить все биты f*	0110 100a ffff ffff
CLRFf, a	*Очистка f*	0110 101a ffff ffff
MOVF f, 0, a	*Переместить f в WREG*	0101 00da ffff ffff
MOVWF f, a	*Переместить WREG в f*	0110 111a ffff ffff
MOVFF fs, fd	*Переместить fs (источник) в fd(приемник) (два слова)*	1100 ffff ffff ffff 1111 ffff ffff ffff
NOP -	*Нет операции (два варианта кода команды)*	1111 xxxx xxxx xxxx 0000 0000 0000 0000
MULWF f, a	*Умножение WREG и f*	0000 001a ffff ffff

Команды операций с **константами (табл. 1.3)** могут иметь переменные:

- Константа (‘k’), которая будет загружена в регистр
- Номер регистра **nn** косвенного адреса FSR, где расположен адрес загружаемого регистра (в этом случае команда состоит из двух слов)
- Нет переменных (‘-‘)

Таблица 1.3. Операции с константами

Мнемоника	Описание	Слово команды
ADDLW k	*Прибавить константу к WREG*	0000 1111 kkkk kkkk
SUBLW k	*Вычитание WREG из константы*	0000 1000 kkkk kkkk
MULLW k	*Умножение константы на WREG*	0000 1101 kkkk kkkk
MOVLB k	*Установить номер банка k*	0000 0001 0000 kkkk
MOVLW k	*Поместить константу в WREG*	0000 1110 kkkk kkkk
LFSR FSRn,k	*Поместить 12 битный косвенный адрес в FSR* ***nn***	1110 1110 00**nn** kkkk 1111 0000 kkkk kkkk

Система команд ***ортогональна***, то есть во всех командах можно применять все типы адресации**: прямую, косвенную и индексную**.

Косвенная адресация

При косвенной адресации адрес регистра f не включается в в код команды, а расположен в одном из регистров косвенной адресации *FSRn (n=0,1,2)*. В микроконтроллерах PIC18Fxx2 три 12-разрядных регистра косвенной адресации: *FSR0(FSR0H:FSR0L), FSR1(FSR1H:FSR1L), FSR2(FSR2H:FSR2L).* Дополнительно **есть регистры INDF0, INDF1 и INDF2, которые физически не существуют.** Обращение к ним фактически вызовет действие с регистром, адрес которого указан в *FSRn, где n = 0, 1, 2.*

Есть ещё четыре подобных регистра, которые определяют, как изменится FSRn при выполнении косвенной адресации.

- Автодекремент *FSRn* после косвенной адресации (обращение к **POSTDECn**)
- Автоинкремент *FSRn* после косвенной адресации (обращение к **POSTINCn**)
- Автоинкремент *FSRn* перед косвенной адресацией (обращение к **PREINCn**)
- Значение в регистре *WREG* используется как смещение к *FSRn*. После косвенной адресации значение *FSRn* не изменяется (обращение к **PLUSWn**)(индексная адресация).

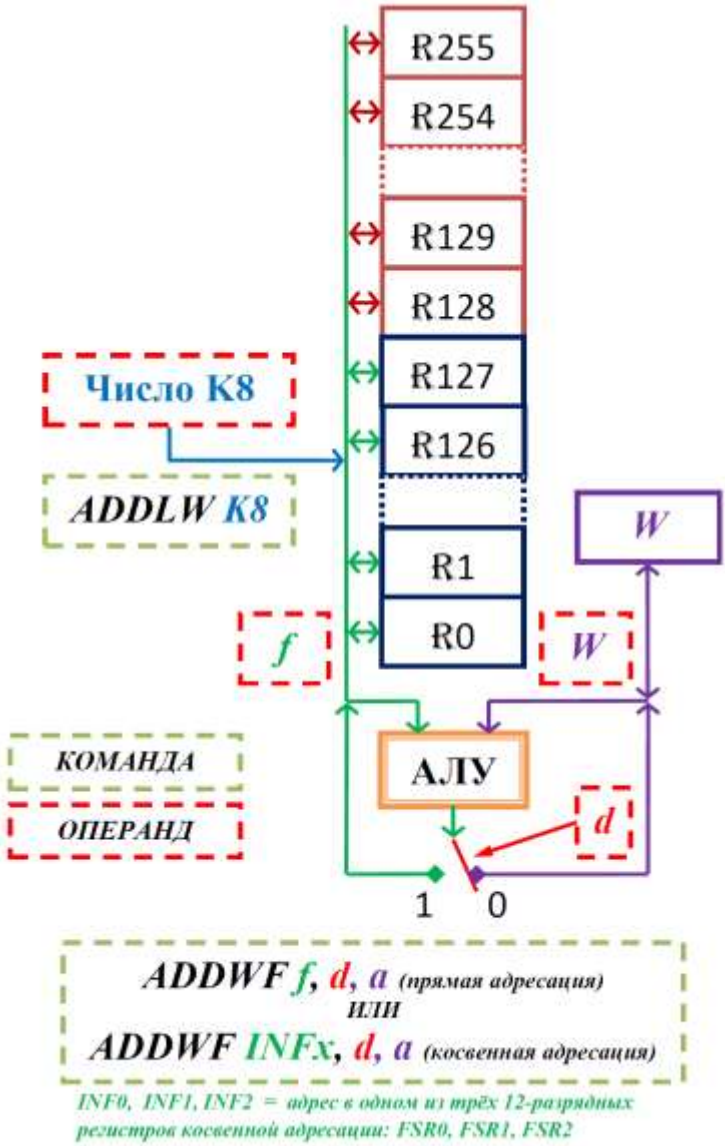


Рис. 1.2. Функциональная схема PIC-контроллера
с точки зрения программиста

Бит ориентированные операции с регистрами

Бит-ориентированные команды (табл. 1.4 и рис. 1.3) используются для ветвления в программах и имеют следующие переменные:

- Регистр (‘**f**’), к которому выполняется обращение
- Номер бита в байте **f** (‘**b** ’)
- Тип доступа к памяти (‘a’)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE				**b**	**b**	**b**	**a**	**f**	**f**	**f**	**f**	**f**	**f**	**f**	**f**

Рис. 1.3. Формат бит-ориентированной команды

Если **a=0**, то используются 128 регистров нулевого банка и 128 регистров специальных функций (банк быстрого доступа).

При **a=1** учитывается BSR (регистр указателя банков).

Таблица 1.4. *Бит* ориентированные операции с регистрами

Мнемоника	**Описание**	**Слово команды** (16 бит)
BCFf, b, a	*Сброс бита **b** в **f***	1001 bbba ffff ffff
BSFf, b, a	*Установка бита **b** в **f***	1000 bbba ffff ffff
BTFSC f, b, a	*Тест бита **b**, пропуст. если ‘0’*	1011 bbba ffff ffff
BTFSSf, b, a	*Тест бита **b**, пропуст. если ‘1’*	1010 bbba ffff ffff
BTGf, b, a	*Инверсия бита **b** в **f***	0111 bbba ffff ffff

ЗАДАНИЯ ДЛЯ ПОДГОТОВКИ

1. Ознакомиться с командами арифметических операций. Рассмотреть действия, выполняемые командами ***ADDWF f, d, a; ADDLW k ; SUBWF f, d, a; SUBLW k; MULLW k ; INCF f, d, a; DECF f, d, a.***

2. Разработать программы на ассемблере (в мнемокодах), используя косвенную адресацию:

1) сложения чисел, находящихся по адресам ADR1 и ADR2 записи результата в регистр с адресом ADR3 **(Программа 1);**

2) вычитания чисел, находящихся по адресу ADR1, из числа, находящегося по адресу ADR2, и записи результата в регистр с адресом ADR3 **(Программа 2);**

3) увеличения на число **NN=0xFF** двухбайтового числа (слова), находящегося в двух регистрах с адресами ADR1+1:ADR1, и записи результата в ячейки ADR2+1:ADR2 **(Программа 3)**. Содержимое регистра ADR1+1 принять равным 0x7E. В программе 3 использовать команду ***ADDWFC f, d, a***;

4) уменьшения на **K=0xEF** двухбайтового числа, находящегося в двух регистрах по адресу ADR1+1:ADR1, и записи результата по адресу ADR2+1:ADR2. **(Программа 4);**

5) инкремента содержимого регистра с адресом ADR2 **(Программа 5);**

6) декремента содержимого регистра с адресом ADR3 **(Программа 6).**

3. Разработать программы №1 – 6, используя прямую адресацию.

Программу рекомендуется начинать следующими начальными установками

```
        title «лабор1»; название программы
        list p=18F4520; тип процессора
        #include<P18F4520.INC>; подключение файла

ADR1  equ 0x400; присвоение ADR1 к РОН (рег. 0x00 банк 4)
ADR2  equ 0x407; присвоение ADR2 к РОН (рег. 0x07 банк 4)
ADR3  equ 0x404; присвоение ADR3 к РОН (рег. 0x04 банк 4)

        org 0x00; адрес следующей команды 0x00
```

Вместо 0x400, 0x407, 0x404 вписать значения для вашего варианта.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить у преподавателя исходные данные для выполнения программ 1...6.

2. Запустите MPLAB-IDE. Создайте новый проект. Выберите в меню **Project>New** для создания нового проекта. Project Name=**LAB1, Project Directory = C:\sem6_MPT\var** (**– номер варианта).

3. Добавьте в проект файл для программы на ассемблере. Выберите в меню **File>Add New File to Project**, создайте файл ****_1.asm** и сохраните в той папке, где был создан проект C:\sem6_MPT\var** (для этого в текстовом окне *Jump to* выбрать *Project directory)*. Не забывайте расширение **.asm**.

4. В меню **Configure>Select device**, выберите тип микроконтроллера PIC18F4520.

5. В меню **Debugger>Select tool** выберите симулятор MPLAB SIM. Появится новая панель управления. Кнопки панели управления слева – направо.

1) Run (F9) – запуск программы.
2) Halt (F5) – остановка программы.
3) Animate – автоматический пошаговый режим (скорость выполнения можно регулировать в Debugger — Settings).
4) Step Into (F7) – шаг низкого уровня (с заходом в подпрограммы в пошаговом режиме).
5) Step Over (F8) – шаг высокого уровня (подпрограмма выполняется в режиме Run).
6) Step Out – выход из подпрограммы (подпрограмма выполняется в режиме Run).
7) Reset – возврат указателя выполнения программы на исходное значение.

6. В окно редактора текста программ скопируйте из проекта **var29** программы с №1 по №6 с косвенной адресацией, затем введите данные из своего варианта.

7. Откомпилируйте проект, щёлкнув по иконке Build All (F10).

8. Просмотрите откомпилирую программу используя меню **View>Disassembler Listing**. Найдите адреса INDF0, INDF1, INDF2, обратите внимание, что код команд занимает два байта.

9. В меню **View** выберите регистры специальных функций (SFR), **Special Function Registers** и **Watch**. В окно Watch добавьте регистры специального назначения WREG, BSR, *FSR0, FSR1, FSR2* ***(кнопка Add SFR)*** и общего назначения, присвоив им название ADR1, ADR2, ADR3. Ввод ***кнопкой Add Symbol***.

10. Провести запуск программы. Результат выполнения программ с №1 по №6 занести в шесть таблиц по шаблону таблицы 1.5. Объяснить полученные результаты.

11. Переделать программы, используя прямую адресацию, и повторить пункты 6-9.

Таблица 1.5. Результаты промежуточных вычислений программой №n

Адрес регистра		Содержимое регистра				
название регистров	конкретное значение ADR**	До выполнения программы	После первого шага	- - - -	После предпоследнего шага	После выполнения программы
ADR1H:ADR1						
ADR2H:ADR2						
ADR3H:ADR3						
рабочий регистр W	FE8h					

СОДЕРЖАНИЕ ОТЧЁТА

1. Блок-схемы алгоритмов программ 1...6.
2. Тексты программ 1...6.
3. Шесть таблиц с исходными данными и результатами выполнения программ.

ПРИЛОЖЕНИЕ 1.1

Таблица П1. Исходные данные к программам 1, 2, 5 и 6

Номер варианта	ADR1 адрес регистра	Число по ADR1	ADR2 адрес регистра	Число по ADR2	Адрес ADR3
1	0x000	43	0x007	25	0x004
2	0x100	93	0x107	18	0x104
3	0x200	29	0x207	56	0x204
4	0x300	0x5E	0x307	37	0x304
5	0x400	38	0x407	84	0x404
6	0x500	57	0x507	0xB0	0x504
7	0x010	0xB2	0x017	0x0A	0x014
8	0x110	0xE3	0x117	0xF4	0x114
9	0x210	22	0x217	23	0x214
10	0x310	0xB2	0x317	61	0x314
11	0x410	0xD7	0x417	0xD7	0x414
12	0x510	0x7F	0x517	0xBC	0x514
13	0x020	0x6D	0x027	15	0x024
14	0x120	94	0x127	0xA5	0x124
15	0x220	30	0x227	76	0x224
16	0x320	0x8C	0x327	91	0x324
17	0x420	84	0x427	0xB2	0x424
18	0x520	0xB0	0x527	0xE3	0x524
19	0x030	0x0A	0x037	22	0x034
20	0x130	0xF4	0x137	0xB2	0x134
21	0x230	0xE3	0x237	0xF4	0x234
22	0x330	22	0x337	23	0x334
23	0x430	0xB2	0x437	61	0x434
24	0x530	0xD7	0x537	0xD7	0x534
25	0x040	0x7F	0x047	0xBC	0x044
26	0x140	0x6D	0x147	15	0x144
27	0x240	94	0x247	0xA5	0x244
28	0x340	30	0x347	76	0x344
29	0x440	0x8C	0x447	91	0x444
30	0x540	84	0x547	0xB2	0x544
31	0x050	0xB0	0x057	0xE3	0x054
32	0x150	0x0A	0x157	22	0x154

ЛАБОРАТОРНАЯ РАБОТА № 2

ПАРАЛЛЕЛЬНЫЕ ПОРТЫ ВВОДА/ВЫВОДА И БИТЫ КОНФИГУРАЦИИ

ЦЕЛЬ РАБОТЫ: изучение команд переходов, параллельных портов ввода/вывода и битов конфигурации PIC-контроллера.

Программная модель параллельного порта

Параллельный порт (рис. 2.1) управляется тремя регистрами: PORTx, LATx, TRISx. Информация с входов приходит в регистр PORTx. Чтение регистра PORTx даёт представление о состоянии входов порта. Регистр – защёлка LATx хранит записанную в PORTx информацию. Информацию можно записать и напрямую в регистр LATx. Регистр TRISx отключает выходы регистра – защёлки LATx от входов порта. Порт работает на выход, если в TRISx записан логический ноль.

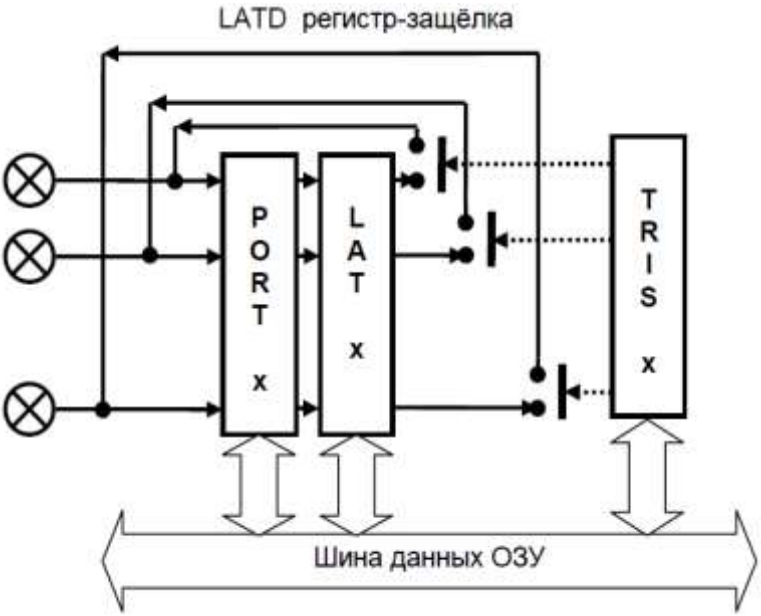


Рис. 2.1. Программная модель параллельного порта

Микроконтроллер PIC18F4520 имеет 5 параллельных портов ввода/вывода. Из них только три имеют 8 контактов ввода/вывода (PortB, PortC, PortD). PortA – 6 выводов, с RA0 по RA5. PortE – 3 вывода, с RE0 по RE2.

Биты конфигурации

В постоянное запоминающее устройства (ПЗУ) кроме программы записываются биты конфигурации. Они устанавливают вид генератора, параметры сброса, защиту записанной в контроллер программы от прочтения и другие свойства микросхемы. Назначение битов и их состояние для данной лабораторной работы приведено в таблице 2.1.

Задача, решаемая в лабораторной работе

Используя учебно-лабораторный стенд, необходимо создать устройство из микроконтроллера, двух кнопок и светодиода. Нажатие на одну кнопку зажигает светодиод, другая кнопка – тушит.

В таблице 2.1 показано к каким выводам PIC-контроллера в зависимости от варианта необходимо подключить кнопки и светодиод.

Таблица 2.1. Варианты подключений

вариант	1	2	3	4	5	6	7	8	9	10
светодиод	RA0	RA1	RA2	RA3	RA4	RA5	RA0	RA1	RA2	RA3
кнопка Вкл	RB0	RB1	RB2	RB3	RB4	RB5	RB6	RC0	RC1	RC2
кнопка Выкл	RB1	RB2	RB3	RB4	RB5	RB6	RB7	RC1	RC2	RC3
вариант	11	12	13	14	15	16	17	18	19	20
светодиод	RA0	RA1	RA2	RA3	RA4	RA5	RA0	RA1	RA2	RA3
кнопка Вкл	RC3	RC4	RC5	RC6	RD0	RD1	RD2	RD3	RD4	RD5
кнопка Выкл	RC4	RC5	RC6	RC7	RD1	RD2	RD3	RD4	RD5	RD6
вариант	21	22	23	24	25	26	27	28	29	30
светодиод	RA0	RA1	RA2	RA3	RA4	RA5	RA0	RA1	RA2	RA3
кнопка Вкл	RD6	RD0	RD1	RD2	RB0	RB1	RB2	RB3	RB4	RB5
кнопка Выкл	RD7	RD1	RD2	RD3	RB1	RB2	RB3	RB4	RB5	RB6

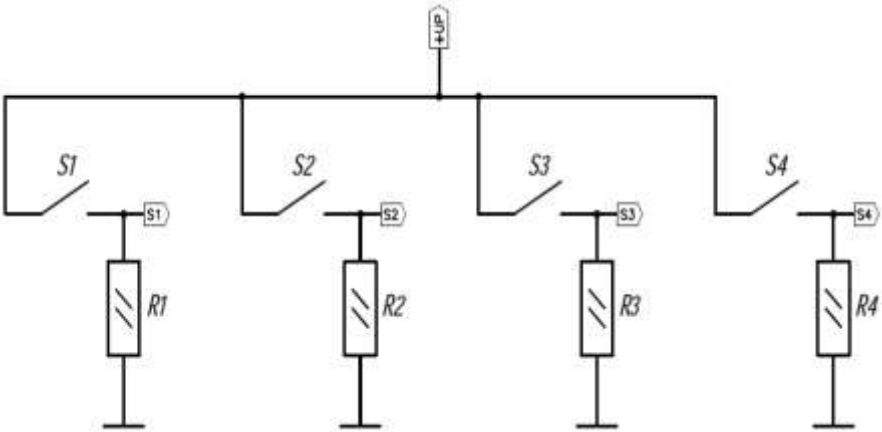


Рис. 2.2. Схема расположения контактов тактовых кнопок

Светодиодные индикаторы HL1 – HL9 подключены через буферные элементы и поэтому имеют высокое входное сопротивление более 47 кОм. Такое решение позволяет подключать светодиоды к контроллеру, не мешая его работе. Индикатор HL9 содержит три светодиода разных цветов.

Принципиальная схема подключения кнопок в учебно-лабораторном стенде показана на рис. 2.2.

Работа стенда в режиме программатора

В составе учебно-лабораторного стенда «КРИСТАЛЛ-22М» есть блок ICD – внутрисхемный отладчик программатор (аналог PicKit2). Для работы с ним в режиме программатора в приложении MPLAB-IDE, применяются следующие управляющие элементы панели инструментов (рис. 2.3).

Рис. 2.3. Управляющие кнопки панели инструментов программатора PicKit2

Назначение кнопок (слева – направо).

1. Запрограммировать МК, записать управляющую программу в МК.
2. Чтение МК, считывание программы в виде HEX кодов.
3. Чтение EEPROM памяти МК.
4. Проверка программирования МК.
5. Стирание МК.
6. Проверка стёртости МК.
7. Высокий уровень на входе сброса, запускает МК.
8. Низкий уровень на входе сброса, сбрасывает МК.
9. Перезапуск ICD в режиме программатора.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. На учебно-лабораторном стенде соберите схему. Подключите светодиод и кнопки к микроконтроллеру.

2. В папке **C:\sem6_MPT\var**** (**– номер варианта) создайте проект ****labIO**, где ** – номер варианта.

3. Добавьте в проект (**File>Add New File to Project**) файл **labIO.asm и запишите в него программу, управляющую светодиодом.

4. В приложении MPLAB-IDE откройте **Configure – Configure bits** (слово конфигурации) и настройте слово конфигурации согласно таблице П2.1 из приложения, чтобы разрешить изменения, необходимо снять флаг *Configuration Bit set in code* (Установка бита конфигурации в коде).

5. Подключите УЛС к компьютеру. Установите в MPLAB-IDE **Programmer – PicKit2**. Появятся девять кнопок (рис.2.3).

6. Скомпилируйте проект (*то есть переведите программу на ассемблере в машинные коды для загрузки в постоянную память программ микроконтроллера*) с помощью горячей клавиши **Ctrl+F10** или нажатием на кнопку «Build All».  При запросе *«Absolute or Relocatable»* выберите ***«Absolute».***

7. Щёлкните по кнопке 9 (перезапуск ICD в режиме программатора).

8. Щёлкните по кнопке 1 (программирование микроконтроллера, записать программу в МК).

9. Щёлкните по кнопке 7 (высокий уровень на входе сброса МК, приводит к запуску МК).

10. Проверти правильность работы программы, нажимая на кнопки УЛС.

11. Результат покажите преподавателю.

СОДЕРЖАНИЕ ОТЧЁТА

Отчёт должен содержать:

1. Рисунок программной модели параллельного порта.
2. Принципиальная схема устройства.
3. Блок-схема алгоритма программы.
4. Программа микроконтроллера.

ПРИЛОЖЕНИЕ 2.1

Таблица П2.1. Установки битов конфигурации

	назначение битов	**установить состояние**
	Oscillator (вид генератора)	HS(кварц)
	Fail - Safe Clock Monitor Enable (монитор тактовой частоты)	Disabled (отключен)
	Internal External Switch Over Mode (подключение внутреннего RC генератора, если пропадает внешнее тактирование от кварцевого генератора)	Disabled (отключен)
	Power Up Timer (Таймер сброса при включении питания)	Enabled (включен)
	Brown Out Detect (детектор пониженного напряжения питания)	Enabled in hardware, SBOREN disabled (аппаратно включен, управление программно - отключен)
	Brown Out Voltage (уровень напряжения сброса для детектора питания)	2,7V
	Watchdog Timer (сторожевой таймер)	Disabled-Controlled by SWDTEN bit
	Watchdog Postscaler (постделитель сторожевого таймера)	1:32768
	CCP2 Mux (выбор линии порта для модуля CCP2)	RC1
	PORTB A/D Enabled (выбор порта по умолчанию, аналоговый или цифровой)	PORTB<4:0> configured as digital I/O on RESET (порт B разряды 0 – 4, цифровые при сбросе)
	Low Power Timer1 Osc enabled (низкопотребляемый режим работы таймера 1)	Disabled
	Master Clear Enabled (включение внешнего сброса)	MCLR Enabled, RE3 – Disabled
	Stack Overflow Reset(сброс при переполнении стека)	Disabled

Продолжение табл. П2.1

	Low Voltage program (режим низковольтного программирования)	Disabled
	назначение битов	**установить состояние**
	Extended Instruction Set Enabled bit (выбор расширенного набора команд)	Disabled
	Code Protect 00800-01FFF (защита памяти программ от чтения программатором)	Disabled
	Code Protect 02000-03FFF	Disabled
	Code Protect 04000-05FFF	Disabled
	Code Protect 06000-07FFF	Disabled
	Code Protect Boot (защита слова конфигурации от чтения программатором)	Disabled
	Data EEPROM Code Protect (Защита EEPROM от чтения программатором)	Disabled
	Table Write Protect 00800-01FFF (Защита памяти программ от табличной записи)	Disabled
	Table Write Protect 02000-03FFF	Disabled
	Table Write Protect 04000-05FFF	Disabled
	Table Write Protect 06000-07FFF	Disabled
	Config. Write Protect (защита слова конфигурации от записи)	Disabled
	Table Write Protect Boot (Защита памяти программ от табличной записи)	Disabled
	Data EEPROM Write Protect (защита EEPROM памяти данных от записи)	Disabled

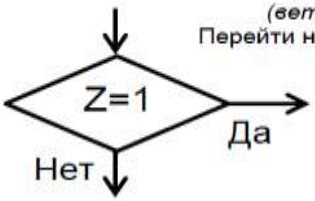
Окончание табл. П2.1

	Table Write Protect 00800-01FFF (Защита памяти программ от табличного чтения)	Disabled
	Table Write Protect 02000-03FFF	Disabled
	Table Write Protect 04000-05FFF	Disabled
	Table Write Protect 06000-07FFF	Disabled
	Table Write Protect Boot (Защита начальной области памяти программ от табличного чтения)	Disabled

ПРИЛОЖЕНИЕ 2.2

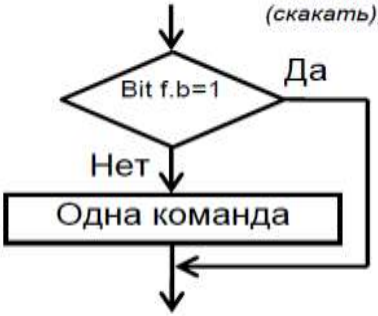
КОМАНДЫ ПЕРЕХОДОВ

BZ n
(**B**ranch if **Z**ero)
(ветвь, если ноль)
Перейти на метку **n**, если ноль

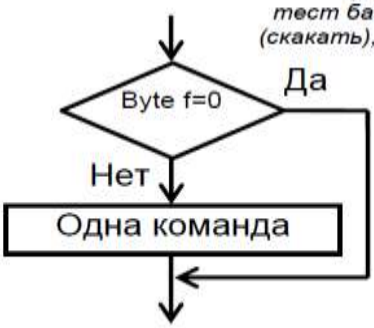


Смещение относительно PC+2
на **nnnn nnnn** слов
(от -128 до127 слов)

BTFSS f, b, a
(**B**it **T**est **f**, **S**kip if **S**et)
тест бита b, пропустить (скакать), если он установлен



TSTFSZ f, a
(**T**est **f**, **S**kip if **0**)
*тест байта **f**, пропустить (скакать), если он равен нулю*



Операции относительного перехода по флагам на метку n

Флаг	*условие*	*команда*
Z	равно нулю	**BZ n**
	неравно нулю	**BNZ n**
C	перенос	**BC n**
	нет переноса	**BNC n**
N	отрицательный результат	**BN n**
	положительный результат	**BNN n**
OV	переполнение	**BOV n**
	нет переполнения	**BNOV n**
безусловный переход	нет условий	**BRA n , (GOTO n)**

Операции с пропуском одной команды

действие	*условие*	*команда*
сравнение f с W	равно	**CPFSEQ f, a**
	больше	**CPFSGT f, a**
	меньше	**CPFSLT f, a**
декремент f	равно нулю	**DECFSZ f,d,a**
	неравно нулю	**DCFSNZ f,d,a**
инкремент f	равно нулю	**INCFSZ f,d,a**
	неравно нулю	**INFSNZ f,d,a**
тест байта f	равно нулю	**TSTFSZ f, a**
тест бита b в байте f	равно нулю	**BTFSC f, b, a**
	равно единице	**BTFSS f, b, a**

BTFSS f, b, a (**B**it **T**est **f**, **S**kip if **S**et) *(тест бита, пропустить(скакать), если он установлен)*

BTFSC f, b, a (**B**it **T**est **f**, **S**kip if **C**lear) *(тест бита, пропустить(скакать), если он очищен)*

ЛАБОРАТОРНАЯ РАБОТА № 3

УПРАВЛЕНИЕ СВЕТОДИОДНЫМ ДИСПЛЕЕМ И ОПРОС КЛАВИАТУРЫ

ЦЕЛЬ РАБОТЫ: изучение параллельных портов ввода/вывода на примере программ динамического управления светодиодным дисплеем и опроса матричной клавиатурой.

Описание динамического управления светодиодным дисплеем

Динамическая индикация основана на инерционности человеческого глаза. Он воспринимает вспыхивающий десятки раз в секунду свет как непрерывное свечение. Это позволяет сократить число проводов, управляющих дисплеем.

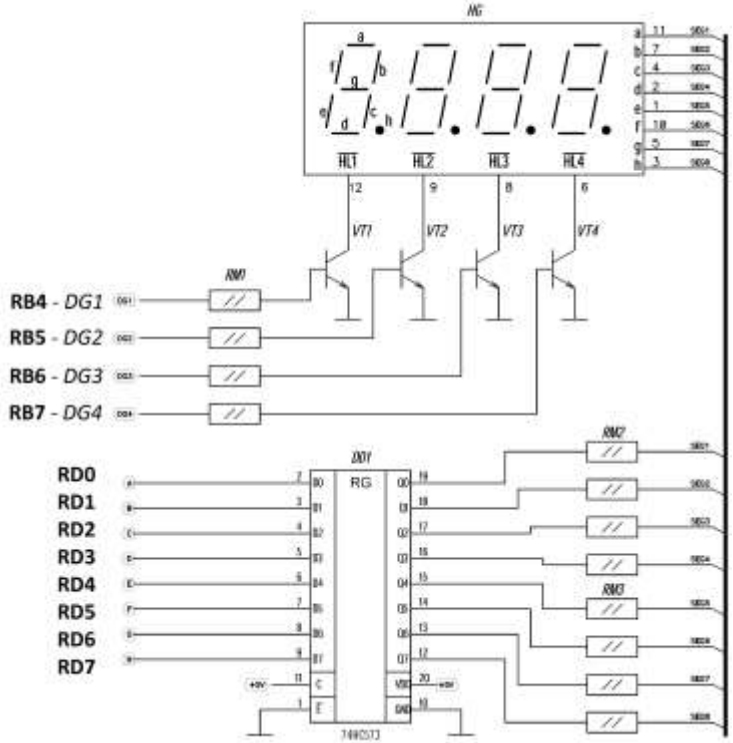


Рис. 3.1. Подключение к МК четырёхэлементного светодиодного дисплея

Семисегментные индикаторы HL1-HL4 подключаются к общей шине (рис. 3.1). То есть одноименные выводы индикаторов соединены между собой. В порядке очерёдности из регистра PORTD по шине передаются сигналы для управления сегментами очередного индикатора. Транзисторы VT1-VT4 подключают тот индикатор, код которого поступает с RD0-RD7. Транзисторами управляют четыре старшие бита регистра LATB через выводы RB4-RB7 порта B.

Резисторы, подключенные к светодиодам сегментов и к базам транзисторов, служат для ограничения тока через светодиоды и базы транзисторов, так как рабочие напряжения на базе открытого транзистора и светодиоде в несколько раз меньше уровня логической единицы.

Матричная клавиатура

Матричная клавиатура (рис. 3.2) содержит четыре строки (SX0-SY3) и четыре столбца (SY0-SY3). На столбцы по очереди подаётся опрашивающее напряжение с выводов RB4-RB7 порта B. Опрос производится циклически записью в регистр LATB последовательности двоичных чисел 0b00010000, 0b00100000, 0b01000000 и 0b10000000. К выходным контактам микроконтроллера

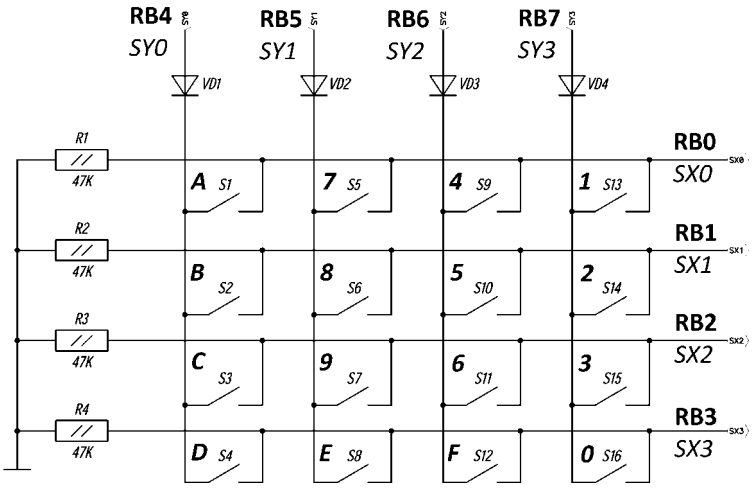


Рис. 3.2. Подключение к МК матричной клавиатуры

с помощью регистра TRISB подключены только четыре старших разряда LATB. Со строк считывается информация о нажатых клавишах. Строки подключены к входам RB0-RB3 порта B. Они опрашиваются чтением регистра PORTB.Диоды VD1-VD4 нужны для предотвращения замыкания выходов RB4-RB7 между собой при одновременном нажатии кнопок из одной строки. Резисторы R1-R4 фиксируют уровень логического нуля.

Работа стенда в режиме программатора

В составе учебно-лабораторного стенда «КРИСТАЛЛ-22М» есть блок ICD – внутрисхемный отладчик программатор (аналог PicKit2). Для работы с ним в режиме программатора в приложении MPLAB-IDE, применяются следующие управляющие элементы панели инструментов (рис. 3.3).

Рис. 3.3. Управляющие кнопки панели инструментов программатора PicKit2

Назначение кнопок (слева - направо).

1. Запрограммировать МК, записать управляющую программу в МК.
2. Чтение МК, считывание программы в виде HEX кодов.
3. Чтение EEPROM памяти МК.
4. Проверка программирования МК.
5. Стирание МК.
6. Проверка стёртости МК.
7. Высокий уровень на входе сброса, запускает МК.
8. Низкий уровень на входе сброса, сбрасывает МК.
9. Перезапуск ICD в режиме программатора.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Подключить индикаторы и клавиатуру к микроконтроллеру (рис. 3.1 и рис. 3.2).

2. В папке ***C:\sem6_MPT\var**** (**– номер варианта) создать проект ***№lab3dyn7segm,*** где № – номер варианта

3. Добавьте в проект (**File>Add New File to Project**) файл ***№lab3.asm*** и скопируйте в него программу из файла 29lab3.asm. Программа выводит на сегменты hgfedcba индикатора 1, 2, 3 и 4 код, прочитанный соответственно с 1, 2, 3 и 4 столбца клавиатуры.

4. В приложении MPLAB-IDE откройте **Configure – Configure bits** (слово конфигурации).

5. Настройте слово конфигурации согласно таблице П2.1 из приложения 2.1 к лабораторной работе № 2.

6. Подключите УЛС (учебно-лабораторный стенд) к компьютеру.

7. В MPLAB-IDE установите **Programmer – PicKit2**. Появятся кнопки (рис. 3.3).

8. Создайте HEX файл для загрузки в МК. Для этого нажмите **Ctrl+F10** (Build All – *построить все*).

9. Щёлкните по кнопке 9 (перезапуск ICD в режиме программатора).

10. Щёлкните по кнопке 1 (программирование микроконтроллера, записать программу в МК).

11. Щёлкните по кнопке 7 (высокий уровень на входе сброса МК, приводит к запуску МК).

12. Нажать по очереди на кнопки клавиатуры. Коды всех кнопок записать в таблицу 3.1. Соответствие кода сегментам на рис.3.1.

13. На основе 29lab3.asm создать программу, которая выводит на 1 и 2 индикаторы десятичный номер варианта, на 3 и 4 – шестнадцатеричный.

14. Запрограммировать контроллер, запустить и результат показать преподавателю.

Таблица 3.1. Результаты лабораторной работы

клавиша	код клавиши
0	hgfedcba
1	
2	
3	
4	
5	
6	
7	
8	
9	
A	
B	
C	
D	
E	
F	

СОДЕРЖАНИЕ ОТЧЁТА

Отчёт должен содержать:

1. Блок-схема алгоритма программы.
2. Программа микроконтроллера.
3. Таблица 3.1.

ЛАБОРАТОРНАЯ РАБОТА № 4

ТАЙМЕР TMR0 И КОНТРОЛЛЕР ПРЕРЫВАНИЙ

ЦЕЛЬ РАБОТЫ: изучение параллельных портов ввода/вывод, контроллера прерываний, таймера TMR0 и программ динамического управления светодиодным дисплеем и матричной клавиатурой.

Сведения из теории

Прерывания

Прерывание – это сигнал процессору от периферийного устройства о том, что произошло событие, требующее немедленного внимания.

Источнику прерываний можно назначить высокий или низкий приоритет. При возникновении прерывания с высоким приоритетом происходит переход по вектору 000008h, а при возникновении прерывания с низким приоритетом – 000018h. Прерывание с высоким приоритетом приостанавливает обработку прерывания с низким приоритетом.

Каждому источнику прерывания соответствует три управляющих бита.

- **Флаг прерывания** (условие прерывания выполнено).
- **Бит разрешения прерывания.**
- **Бит приоритета** (выбор низкого или высокого приоритета прерывания).

Десять регистров специального назначения управляют прерываниями.

- **RCON** (включение приоритетной системы прерываний).
- **INTCON** (запрещение всех прерываний, флаги и разрешения TMR0, INT0, RB).
- **INTCON2** (подтягивающие регистры, фронты INT, приоритеты TMR0, RB).
- I**NTCON3** (управление внешними прерываниями INT1 и INT2).

- **PIR1, PIR2** (Флаги периферийных прерывания).
- **PIE1, PIE2** (Биты разрешения периферийных прерываний).
- **IPR1, IPR2** (Биты приоритета периферийных прерываний).

При переходе к обработке прерывания адрес возврата сохраняется в стеке и запрещаются все прерывания, используемого им приоритета. В это же время в скрытых регистрах WS, STATUSS, BSRS сохраняется последнее значение WREG, STATUS, BSR.

Команда возврата ***RETFIE s*** после возврата разрешает прерывания. Если s=1, при возврате загружаются WREG, STATUS, BSR из скрытых регистров WS, STATUSS, BSRS. Если s=0, этого не происходит.

Первым по значимости регистром управления прерываниями является регистр INTCON (таблица 4.1).

Таблица 4.1. Регистр управления прерываниями INTCON

бит 7	бит 6	бит 5	бит 4	бит3	бит 2	бит 1	бит0
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF

Бит 7. **GIE/GIEH:** бит глобального разрешения прерываний.
IPEN=0
1=разрешены все немаскированные прерывания,
0= все прерывания запрещены.

IPEN=1
1= разрешены прерывания с высоким приоритетом,
0= запрещены прерывания с высоким приоритетом.

Бит 6. **PEIE/GIEL**: бит разрешения периферийных прерываний.
IPEN=0
1=разрешены все периферийные немаскированные прерывания,
0= запрещены все периферийные прерывания.
IPEN=1
1= разрешены прерывания с низким приоритетом,
0= запрещены прерывания с низким приоритетом.

Бит 5. **TMR0IE**: разрешение прерывания по переполнению TMR0.
1=разрешено прерывание по переполнению TMR0,
0=запрещено прерывание по переполнению TMR0.

Бит 4. **INT0IE**: разрешение внешнего прерывания INT0.
1=разрешено внешнее прерывание INT0,
0=запрещено внешнее прерывание INT0.

Бит 3. RBIE: разрешение прерывания по изменению RB7-RB4.
1=разрешено прерывание по изменению RB7-RB4,
0=запрещено прерывание по изменению RB7-RB4.

Бит 2. **TMR0IF**: флаг прерывания по переполнению TMR0.
1=произошло переполнение TMR0,
0=переполнения TMR0 не было.

Бит 1. **INT0IF**: флаг внешнего прерывания INT0.
1=выполнено условие внешнего прерывания INT0,
0=условие внешнего прерывания INT0 не выполнено.

Бит 0. **RBIF**: флаг прерывания по изменению RB7-RB4.
1=зафиксировано изменение на входах RB7-RB4,
0=уровень сигнала на входах RB7-RB4 не изменялся.

Таймер TMR0

Таймер управляется регистром **T0CON** (таблица 4.2).

Таблица 4.2. Регистр управления таймером T0CON

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0

Бит 7. **TMR0ON:** бит разрешения работы TMR0.
1=включен,
0=выключен.

Бит 6. **T08BIT:** режим работы.
1=8 разрядов,
0=16 разрядов.

Бит 5. **T0CS:** выбор источника.
1=сигнал с RA4,
0=внутренний Fosc/4.

Бит 4. **T0SE:** выбор фронта.
1=задний,
0=передний.

Бит 3. **PSA:** подключение предделителя.
1= предделитель не назначен. Тактовый вход минует делитель.
0 = вход счётчика подключен к выходу предварительного делителя.

Бит 2-0. **T0PS2:T0PS0:** коэффициент деления.
111 = 1:256
110 = 1:128
101 = 1:64
100 = 1:32
011 = 1:16
010 = 1:8
001 = 1:4
000 = 1:2

Импульсы, пришедшие на таймер, подсчитывается счётчиком **TMR0**. При **T08BIT =0** счётчик – 16-разрядный. 8-разрядный, если **T08BIT =1**.

Функциональная схема таймера **TMR0** в 8-ми разрядном режиме и выключенной приоритетной системы прерываний представлена на рисунке 4.1. Сброс бита **IPEN** (RCON,7) выключает приоритетную систему прерываний.

Бит **TMR0ON** подключает импульсы к счётчику.

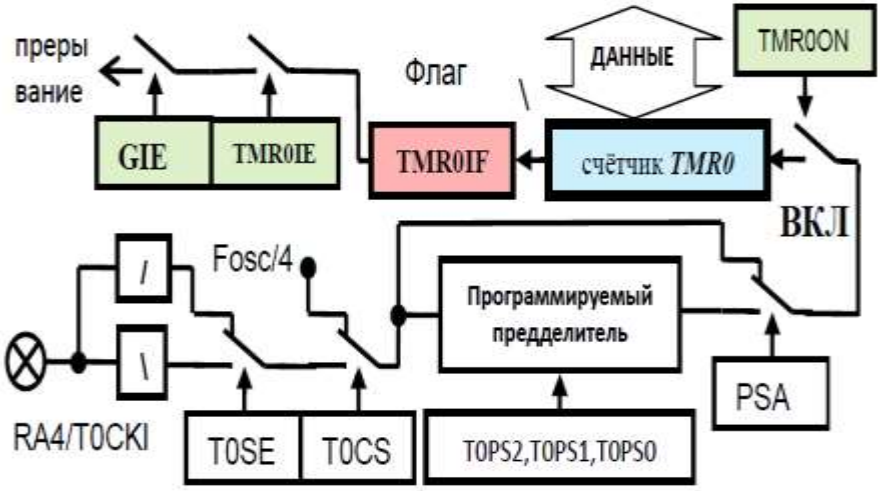


Рис. 4.1. Функциональная схема таймера TMR0 в 8-ми разрядном режиме

В режиме таймера **(T0CS=0)** приращение происходит на каждом машинном цикле, то есть с частотой Fosc/4. Эту частоту можно уменьшить предделителем. Он подключается сбросом бита PSA. Биты **T0PS2: T0PS1:T0PS0** определяют коэффициент деления предделителя Кд. В лабораторной работе Кд=8.

Счётчик **TMR0** в 8-ми разрядном режиме делит входную частоту на 256. Флаг **TMR0IF** устанавливается с частотой Fф=Fosc/(4*256*Кд). Его необходимо сбрасывать вначале подпрограммы обслуживания прерывания по флагу **TMR0IF**. Разрешение этого прерывания осуществляется битом **TMR0IE**. Необходимо также разрешить все прерывания битом **GIE**.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Подключить индикаторы и клавиатуру к микроконтроллеру (рис. 3.1 и рис. 3.2 из лабораторной работы № 3).

2. В папке ***C:\sem6_MPT\var**** (** - номер варианта) создать проект ***№lab4int,*** где № – номер варианта.

3. Добавьте в проект (**File>Add New File to Project**) файл ***№lab4.asm*** и скопируйте в него программу из файла 29lab4.asm.

4. В приложении MPLAB-IDE откройте **Configure – Configure bits** (слово конфигурации).

5. Настройте слово конфигурации согласно таблице П2.1 из приложения 2.1.

6. Подключите УЛС (учебно-лабораторный стенд) к компьютеру.

7. В MPLAB-IDE установите **Programmer – PicKit2**. Появятся кнопки (рис. 4.2) (назначение кнопок описано в лабораторной работе № 3).

8. Создайте HEX файл для загрузки в МК. Для этого нажмите **Ctrl+F10** (Build All – *построить все*).

9. Щёлкните по кнопке 9 (перезапуск ICD *(внутрисхемный отладчик - программатор)* в режим программатора).

10. Щёлкните по кнопке 1 (программирование микроконтроллера, записать программу в МК).

11. Щёлкните по кнопке 7 (высокий уровень на входе сброса МК, приводит к запуску МК).

12. Нажать по очереди на кнопки клавиатуры. Убедиться в работоспособности программы.

13. На основе 29lab4.asm создать программу, которая при нажатии одной клавиши, выводит на первый индикатор шестнадцатеричный код номера клавиши, а на последние два – номер варианта. Номер клавиши равен номеру варианта по модулю 16.

14. Запрограммировать контроллер, запустить и результат показать преподавателю.

Рис. 4.2. Управляющие кнопки панели инструментов программатора PicKit2

СОДЕРЖАНИЕ ОТЧЁТА

Отчёт должен содержать:

1. Блок-схему алгоритма основной программы.
2. Блок-схему алгоритма подпрограммы обработки прерывания.

ПРИЛОЖЕНИЕ 4.1

БЛОК-СХЕМА И ТЕКСТ ОСНОВНОГО ЦИКЛА ПРОГРАММЫ *№ lab4int*


```
                    ОСНОВНОЙ ЦИКЛ
                    (набросок программы)
ML0:    ;тест нажатой клавиши
        BTFSS  f, b, a

        BRA n

        movlw b'XXXXXXXXX'
        movwf A4, 0
        movlw b'XXXXXXXXX'
        movwf A2, 0
        movlw b'XXXXXXXXX'
        movwf A1, 0
        bra ML0

n:      movlw b'01000000'
        movwf A1, 0
        movwf A2, 0
        movwf A3, 0
        movwf A4, 0
        bra ML0
```

В программу кроме основного цикла входят начальные установки и подпрограмма обработки прерываний.

Команда **BTFSS f, b, a** тестирует бит с номером **b** в регистре с номером (или названием) **f**. Если бит равен нулю, то выполняется следующая команда. В противном случае эта команда пропускается.

ЛАБОРАТОРНАЯ РАБОТА № 5

ОТЛАДКА ПРОГРАММЫ С ПОМОЩЬЮ СИМУЛЯТОРА И РЕЖИМА АНИМАЦИИ

ЦЕЛЬ РАБОТЫ: изучение отладки программы с помощью симулятора и режима анимации.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. В папке ***C:\sem6_MPT\var**** (**– номер варианта) откройте проект ***№lab5int,*** где № – номер варианта.

2. Командой **Debugger – Select Tool – MPLAB SIM** подключите симулятор.

3. В меню **View – Watch** выберите следующие столбцы для отображения (щёлкнув правой кнопкой мыши по закладке): ***Symbol name, Value, Decimal, Binary***.

4. В окно **Watch** вставить строки ***TRISB, TRISD, PORTB, PORTD, TMR0L, TMR0H, T0CON, INTCON, INTCON2, RCON, WREG, ADCON1, CDYN, A1 – A4, K1 – K4.***

5. Выберите **Debugger** – **Stimulus-New Workbook.**

6. В окне **Stimulus** выберите вкладку **Asynch** (асинхронный ввод). В столбце **Pin/SFR** добавьте выводы ***RB3-RB0***. В столбце **Action** (условие срабатывания вывода) выберите **Toggle.** (Триггер – установка контакта порта, настроенного на ввод, в логическую 1 и после следующего нажатия в 0 и т.д.).

7. Добавьте секундомер и проект (**Debugger – Stopwatch**).

8. Установите частоту кварца 10 МГц в диалоговом окне свойств симулятора, меню **Debugger – Settings,** вкладка **Osc/Trace.**

9. Во вкладке **Animation** меню **Debugger – Settings** установите скорость работы анимации на максимум (fastest) и выберите Обновление регистров в режиме запуска программы (**Enable Realtime watch updates**).

10. Выберите **View** – **Simulator Logic Analyzer.**

11. Нажатием на кнопку Каналы (**Channels**) откройте окно конфигурации каналов **Configure Channels**,. Добавьте ***RB0-RB7 и RD0-RD7*** в анализатор логических сигналов симулятора.

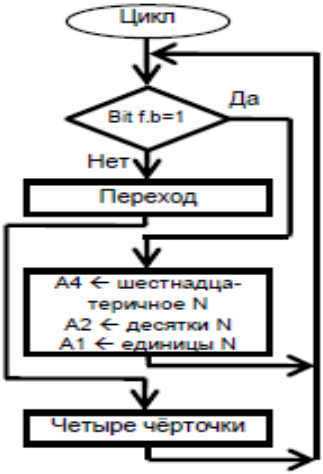
12. Настройте слово конфигурации. Сверните окно.
13. Нажмите на кнопку **Animate (меню Debugger),** чтобы перейти в автоматический пошаговый режим.
14. Ставим точку останова (B).
15. Запускаем RUN.

СОДЕРЖАНИЕ ОТЧЁТА

Отчёт должен содержать:

1. Блок-схему алгоритма основной программы.
2. Блок-схему алгоритма подпрограммы обработки прерывания.
3. Эпюры напряжений на контактах клавиатуры и дисплея.

БЛОК-СХЕМА ОСНОВНОГО ЦИКЛА ПРОГРАММЫ *№ lab4int*




```
                ОСНОВНОЙ ЦИКЛ
                (набросок программы)
ML0:    ;тест нажатой клавиши
        BTFSS  f, b, a

        BRA n

        movlw b'XXXXXXXXX'
        movwf A4, 0
        movlw b'XXXXXXXXX'
        movwf A2, 0
        movlw b'XXXXXXXXX'
        movwf A1, 0
        bra ML0

n:      movlw b'01000000'
        movwf A1, 0
        movwf A2, 0
        movwf A3, 0
        movwf A4, 0
        bra ML0
```

В программу кроме основного цикла входят начальные установки и подпрограмма обработки прерываний.

Методические материалы

ОДНОАДРЕСНЫЙ МИКРОКОНТРОЛЛЕР СЕМЕЙСТВА PIC

Методические указания

Составитель ***Иванов Владимир Васильевич***

Редактор И.П. Ведмидская
Компьютерная вёрстка И.П. Ведмидской

Подписано в печать 19.08.2019. Формат 60x84 1/16.
Бумага офсетная. Печ. л. 2,5.
Тираж 25 экз. Заказ . Арт. 73(P1M)/2019.

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С. П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)
443086 САМАРА, МОСКОВСКОЕ ШОССЕ, 34.

Изд-во Самарского университета.
443086 Самара, Московское шоссе, 34.