

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Белорусский национальный технический университет

Кафедра «Информационно-измерительная техника
и технологии»

ПРОГРАММИРУЕМЫЕ ЦИФРОВЫЕ УСТРОЙСТВА: МИКРОКОНТРОЛЛЕРЫ

Практикум

для студентов специальностей

1-38 02 01 «Информационно-измерительная техника»,

1-38 02 03 «Техническое обеспечение безопасности»,

1-54 01 02 «Методы и приборы контроля качества
и диагностики состояния объектов»

*Рекомендовано учебно-методическим объединением
по образованию в области приборостроения, учебно-методическим
объединением по образованию в области обеспечения качества*

Минск
БНТУ
2020

УДК [004.318–181.48 + 321.382.049.77] (075.8)

ББК 32.973я7

И85

А в т о р ы:

А. В. Исаев, П. Г. Кривицкий, К. В. Пантелеев

Р е ц е н з е н т ы:

В. Ф. Алексеев, Д. В. Кнотько

И85 Программируемые цифровые устройства: микроконтроллеры : практикум для студентов специальностей 1-38 02 01 «Информационно-измерительная техника», 1-38 02 03 «Техническое обеспечение безопасности», 1-54 01 02 «Методы и приборы контроля качества и диагностики состояния объектов» / А. В. Исаев, П. Г. Кривицкий, К. В. Пантелеев. – Минск: БНТУ, 2020. – 95 с.

ISBN 978-985-583-071-0.

В практикуме содержатся основные характеристики и материалы для программирования и использования микроконтроллеров при решении практических задач, связанных с созданием программируемых электронных приборов. Изложены указания по выполнению лабораторных работ, включающих изучение отладочных сред IDE MPLab и Keil μ Vision, создание простейших проектов для микроконтроллеров семейства Microchip и Intel. Каждая лабораторная работа содержит краткие теоретические сведения, порядок выполнения работы, подробные примеры решения заданий.

УДК [004.318–181.48 + 321.382.049.77] (075.8)

ББК 32.973я7

ISBN 978-985-583-071-0

© Белорусский национальный
технический университет, 2020

Лабораторная работа № 1

ИЗУЧЕНИЕ СРЕДЫ РАЗРАБОТКИ MPLAB. СОЗДАНИЕ ПРОСТЕЙШИХ ПРОЕКТОВ ДЛЯ МИКРОКОНТРОЛЛЕРОВ СЕМЕЙСТВА MICROCHIP

Цель работы. Изучить организацию 8-разрядного RISK микроконтроллера семейства Microchip. Ознакомиться с его системой и форматом команд, способами адресации. Изучить среду разработки MPLab. Составить и проверить выполнение программы для нахождения функции, используя возможности среды MPLab.

1.1. Микроконтроллеры семейства Microchip. Краткие теоретические сведения

1.1.1 Общие сведения. В качестве общих элементов микроконтроллеров семейства Microchip можно выделить следующее:

- гарвардская архитектура – то есть отдельные области памяти для хранения команд (программы) и данных, которые в свою очередь имеют различную разрядность;
- высокопроизводительная RISC архитектура процессора;
- восьмиразрядная шина данных;
- 35 команд, 4 способа адресации.

Все контроллеры семейства Microchip разделяются:

1) по разрядности команд:

- базовое семейство (12-разрядные команды);
- среднее семейство (14-разрядные команды);
- высокоразрядное семейство (16-разрядные команды).

2) по типу применяемой памяти команд:

- микроконтроллеры масочного типа. Программирование микроконтроллеров осуществляется на производственных лентах;
- EPROM (электронный программируемый ROM). Однократно программируемый микроконтроллер;
- в флэш-микросхемах программирование микроконтроллера может производиться в самой схеме.

3) по объему памяти команд;

- 4) по объему памяти данных (количеству регистров общего назначения);

5) по типу, количеству и параметрам встроенной периферии (наличие, количество и разрядность АЦП, ЦАП, таймеров, аппаратных интерфейсов, портов ввода/вывода и др.).

Структуру микроконтроллеров Microchip можно разделить на:

- ядро;
- периферийный модуль;
- модуль специального назначения.

В структуру ядра входят следующие модули:

- основной тактовый генератор;
- логика сброса;
- центральный процессор;
- арифметико-логическое устройство;
- организация памяти;
- организация прерываний;
- система команд.

Периферийные модули позволяют организовать интерфейс связи с внешней схемой, а также позволяют выполнить отсчет времени и интервалов. Периферийный модуль включает следующее:

- таймер;
- модуль захвата;
- модуль сравнения;
- модуль широтно-импульсной модуляции;
- синхронный и последовательный порт;
- основной и ведущий порт;
- источник опорного напряжения;
- компараторы;
- АЦП 8-, 10-разрядные, интегрирующие;
- ЦАП;
- ведомый параллельный порт;
- универсальные порты ввода/вывода.

Модули специального назначения обеспечивают уменьшения стоимости системы, увеличения надежности и гибкости проектирования. Сюда входят:

- биты конфигурации;
- схемы сброса;
- сторожевые таймеры;
- режим энергосбережения;
- интегрированный тактовый генератор;
- модули внутрисхемного программирования.

1.1.2 Организация памяти. Встроенную память микроконтроллера можно разделить на два типа: память программ и память данных. В качестве примера будем рассматривать преимущественно микроконтроллер среднего семейства PIC12F675.

Память программ. Микроконтроллеры PIC12F675 имеют 13-разрядный счетчик команд PC, способный адресовать до $8K \times 14$ слов памяти программ. Физически реализовано в PIC12F675 $1K \times 14$ (0000h–05FFh) памяти программ. Обращение к физически не реализованной памяти программ приводит к адресации памяти в адресном пространстве 0000h–05FFh. Адрес вектора сброса – 0000h. Адрес вектора прерываний – 0004h (рис. 1.1).

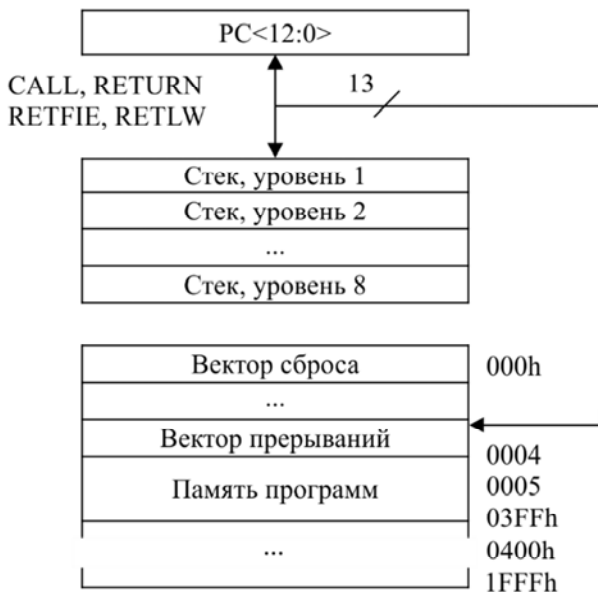


Рис. 1.1. Организация памяти программ и стека в PIC12F675

Память данных. Память данных разделяется на две категории: регистровая память данных и EEPROM-память данных.

Регистровая память данных разделена на два банка, которые содержат регистры общего (GPR) и специального (SFR) назначения. Первые 32 ячейки каждого банка зарезервированы под регистры специального назначения. Эти регистры предназначены для управ-

ления ядром микроконтроллера и периферийными модулями и реализованы как статическое ОЗУ. Организация регистровой памяти данных в PIC12F675 представлена в табл. 1.1.

Таблица 1.1

Организация регистровой памяти данных в PIC12F675

Адрес	Имя	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
1	2	3	4	5	6	7	8	9	10
Банк 0									
00h	INDF	Обращение к регистру, адрес которого указан в FSR							
01h	TMR0	Регистр модуля TMR0							
02h	PCL	Младший байт счетчика команд PC							
03h	STATUS			RP0	-TO	-PD	Z	DC	C
04h	FSR	Регистр адреса при косвенной адресации							
05h	GPIO	-	-	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
0Ah	PCLATH	-	-	-	Буфер старших 5 бит счетчика команд PC				
0Bh	INTCON	GIE	PEIE	TOIE	INTE	GPIE	TOIF	INTF	GPIF
0Ch	PIR1	EEIF	ADIF	-	-	CMIF	-	-	TMR1IF
0Eh	TMR1L	Младший регистр 16-разрядного таймера/счетчика TMR1							
0Fh	TMR1H	Старший регистр 16-разрядного таймера/счетчика TMR1							
10h	T1CON	-	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNCS	TMR1CS	TMR1ON
19h	CMCON	-	GOOUT	-	CINV	CIS	CM2	CM1	CM0
1Eh	ADRESH	8 (левое выравнивание) или 2 (правое выравнивание) старших бита результата АЦП							
1Fh	ADCON0	ADFM	VCFG	-	-	CHS1	CHS0	GO/DONE	ADON
Банк 1									
80h	INDF	Обращение к регистру, адрес которого указан в FSR							
81h	OPTIONREG	-GPPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0
82h	PCL	Младший байт счетчика команд PC							
83h	STATUS	-	-	RP0	-TO	-PD	Z	DC	C
84h	FSR	Регистр адреса при косвенной адресации							
85h	TRISIO	-	-	TRIS5	TRIS4	TRIS3	TRIS2	TRIS1	TRIS0
8Ah	PCLATH	-	-	-	Буфер старших 5 бит счетчика команд PC				
8Bh	INTCON	GIE	PEIE	TOIE	INTE	GPIE	TOIF	INTF	GPIF
8Ch	PIE1	EEIE	ADIE	-	-	CMIE	-	-	TMR1IE

Окончание табл. 1.1

1	2	3	4	5	6	7	8	9	10
8Eh	PCON	–	–	–	–	–	–	-POR	-BOD
90h	OSCCAL	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	–	–
95h	WPU	–	–	WPU5	WPU4	–	WPU2	WPU1	WPU0
96h	IOCB	–	–	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
99h	VRCON	VREN	–	VRR	–	VR3	VR2	VR1	VRO
9Ah	EEDATA	Данные, записываемые в EEPROM память							
9Bh	EEADR	–	Адрес ячейки в EEPROM памяти данных						
9Ch	EECON1	–	–	–	–	WRER	WREN	WR	RD
9Dh	EECON2	Управляющий регистр записи в EEPROM памяти данных							
9Eh	ADRESL	2 или 8 младших бита результата АЦП							
9Fh	ANSEL	–	ADC S2	ADCS 1	ADCS 0	ANS3	ANS2	ANS1	ANS0

Регистры общего назначения имеют адреса с 20h по 5Fh в каждом банке памяти данных. Физически не реализованные регистры читаются как «0». Бит RP0 регистра STATUS (далее STATUS <RP0> или STATUS <5>) предназначен для выбора текущего банка памяти данных: RP0 = 0 – выбран банк «0»; RP0 = 1 – выбран банк «1». В микроконтроллере PIC12F675 регистры общего назначения имеют организацию 64 x 8. Обращение к регистрам можно выполнить прямой или косвенной адресацией (используя регистр FSR).

Ниже (табл. 1.2–1.4) и далее (п. 1.1.3) будут приведены основные сведения по организации регистровой памяти микроконтроллеров PIC12C508, PIC16F84A, PIC12CE673, которые будут необходимы для выполнения индивидуальных заданий к лабораторной работе.

Таблица 1.2

Память данных PIC12C508

Адрес	Имя	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
1	2	3	4	5	6	7	8	9	10
–	TRIS	–	–	Установка направления канала					
–	OPTION	-GPWU	-GPPU	T0CS	T0SE	PSA	PS2	PS1	PS0
00h	INDF	Обращение к регистру, адрес которого записан в FSR							
01h	TMR0	Регистр нулевого таймера (TMR0)							
02h	PCL	Младший байт счетчика команд (PC)							
03h	STATUS	GPWUF	–	0	-TO	-PD	Z	DC	C
04h	FSR	Регистр адреса при косвенной адресации.							

Окончание табл. 1.2

1	2	3	4	5	6	7	8	9	10
05h	OSCCAL	CAL3	CAL2	CAL1	CAL0	–	–	–	–
06h	GPIO	–	–	GP5	GP4	GP3	GP2	GP1	GP0
07h–1Fh	Регистры общего назначения								

Таблица 1.3

Память данных PIC16F84A

Адрес	Имя	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
1	2	3	4	5	6	7	8	9	10
Банк 0									
00h	INDF	Обращение к регистру, адрес которого указан в FSR							
01h	TMRO	Регистр модуля TMRO (8-ми разрядный таймер/счетчик)							
02h	PCL	Младший байт счетчика команд PC							
03h	STATUS	IRP	RP1	RP0	-TO	-PD	Z	DC	C
04h	FSR	Регистр адреса при косвенной адресации							
05h	PORTA	–	–	–	RA4	RA3	RA2	RA1	RA0
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
07h	–	Не реализован							
08h	EEDATA	Регистр данных, записываемых в EEPROM память							
09h	EEADR	Адрес ячейки в EEPROM памяти данных							
0Ah	PCLATH	–	–	–	Буфер старших 5 бит счетчика команд PC				
0Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
0Ch–4Fh	68 регистров общего применения								
Банк 1									
80h	INDF	Обращение к регистру, адрес которого указан в FSR							
81h	OPTION_REG	-RBPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0
82h	PCL	Младший байт счетчика команд PC							
83h	STATUS	IRP	RP1	RPO	-TO	-PD	Z	DC	C
84h	FSR	Регистр адреса при косвенной адресации							
85h	TRISA	Направление данных через порт A							
86h	TRISB	Направление данных через порт B							
87h	–	Не реализован							
88h	EECON1	–	–	–	EEIF	WRE RR	WRE N	WR	RD
89h	EECON2 ¹⁾	Управляющий регистр записи в EEPROM память данных							
8Ah	PCLATH	–	–	–	Буфер старших 5 бит счетчика команд PC				
8Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
8Ch–CFh	Отображается на пространство нулевого банка								

Таблица 1.4

Память данных PIC12CE673

Адрес	Имя	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
1	2	3	4	5	6	7	8	9	10
Банк 0									
00h	INDF	Обращение к регистру, адрес которого указан в FSR							
01h	TMR0	Регистр модуля TMRO (8-ми разрядный таймер/счетчик)							
02h	PCL	Младший байт счетчика команд PC							
03h	STATUS	IRP	RP1	RP0	-TO	-PD	Z	DC	C
04h	FSR	Регистр адреса при косвенной адресации							
05h	GPIO	–	–	GP5	GP4	GP3	GP2	GP1	GP0
0Ah	PCLATH	–	–	–	Буфер старших 5 бит счетчика команд PC				
0Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
0Ch	PIR1	–	ADIF	–	–	–	–	–	–
1Eh	ADRES	Буфер модуля АЦП							
1Fh	ADCON0	ADCS1	ADCS0	–	CHS1	CHS0	GO/DONE	–	ADON
20h–7Fh		Регистры общего применения							
Банк 1									
80h	INDF	Обращение к регистру, адрес которого указан в FSR							
81h	OPTION_REG	-RBPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0
82h	PCL	Младший байт счетчика команд PC							
83h	STATUS	IRP	RP1	RP0	-TO	-PD	Z	DC	C
84h	FSR	Регистр адреса при косвенной адресации							
85h	TRISIO	Направление данных через порт							
8Ah	PCLATH	–	–	–	Буфер старших 5 бит счетчика PC				
8Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
8Ch	PIE1	–	ADIE	–	–	–	–	–	–
8Eh	PCON	–	–	–	–	–	–	-POR	–
8Fh	OSCCAL	CAL3	CAL2	CAL1	CAL0	CALFST	CALSLW	–	–
9Fh	ADCON1	–	–	–	–	–	PCFG2	PCFG1	PCFG0
A0h–BFh		Регистры общего применения							

1.1.3 Регистры управления процессорным ядром. Регистр STATUS. Регистр STATUS содержит флаги состояния АЛУ (табл. 1.5–1.8), флаги причины сброса микроконтроллера и биты управления банками памяти данных.

Таблица 1.5

Биты регистра STATUS микроконтроллера PIC12F675

№ бита	Имя	Описание
1	2	3
Бит 6, 7	–	Зарезервированы
Бит 5	RP0	Выбор банка памяти данных (при прямой адресации)
Бит 4	-TO	Флаг переполнения сторожевого таймера WDT
Бит 3	-PD	Флаг детектора выключения питания: 1 – после сброса POR или выполнения команды CLRWDT; 0 – после выполнения команды SLEEP
Бит 2	Z	Флаг нулевого результата
Бит 1	DC	Флаг десятичного переноса/заема: 1 – был перенос из младшего полубайта; 0 – не было переноса из младшего полубайта
Бит 0	C	Флаг переноса/заема: 1 – был перенос из старшего бита; 0 – не было переноса из старшего бита

Таблица 1.6

Биты регистра STATUS PIC12C508

Бит	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
Имя	GPWUF	–	–	-TO	-PD	Z	DC	C
GPWUF – Бит сброса GPIO: 1 – сброс произошел при выходе из режима SLEEP по изменению сигнала на входе; 0 – другой вид сброса								

Таблица 1.7

Биты регистра STATUS PIC16F84A и PIC12CE673

Бит	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
Имя	IRP	RP1	RP0	-TO	-PD	Z	DC	C
IRP – Зарезервирован, должен поддерживаться равным '0' RP1 – Зарезервирован, должен поддерживаться равным '0' RP0 – Выбор банка памяти данных (используется при прямой адресации): 1 – Банк 1; 0 – Банк 0								

Регистр STATUS может быть адресован любой командой. Если обращение к регистру STATUS выполняется командой, которая воздействует на флаги Z, DC и C, то изменение этих трех битов командой заблокирована. Эти биты сбрасываются или устанавливаются согласно логике ядра микроконтроллера. Команды изменения регистра STATUS также не воздействуют на биты -TO и -PD.

При изменении битов регистра STATUS рекомендуются команды, не влияющие на флаги ALU (SWAPF, MOVWF, BCF и BSF).

Регистр *OPTION_REG*. Регистр «OPTION_REG» (табл. 1.9–1.11) содержит следующие биты управления: предварительный делитель TMR0/WDT, активный фронт внешнего прерывания GP2/INT, таймер TMR0, включение подтягивающих резисторов на входах GPIO.

Таблица 1.9

Биты регистра OPTION_REG PIC12F675

№ бита	Имя	Описание
1	2	3
Бит 7	-GPPU	Бит включения подтягивающих резисторов на входах GPIO
Бит 6	INTEDG	Выбор активного фронта сигнала на входе прерывания INT
Бит 5	TOCS	Выбор тактового сигнала для TMR0
Бит 4	TOSE	Выбор фронта приращения TMR0 при внешнем сигнале: 1 – приращение по заднему фронту на выводе GP2/TOCKI; 0 – приращение по переднему фронту на выводе GP2/TOCKI
Бит 3	PSA	Выбор включения предделителя: 1 – предделитель включен перед сторожевым таймером WDT; 0 – предделитель включен перед таймером TMR0
Бит 2–0	PS2–PS0	Установка коэффициента деления предделителя

Таблица 1.10

Биты регистра OPTION PIC12C508

Бит	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
Имя	-GPWU	-GPPU	TOCS	TOSE	PSA	PS2	PS1	PS0
-GPWU – Разрешение выхода из режима SLEEP по изменению сигнала на входах GP0, GP1, GP3: 1 – запрещено; 0 – разрешено								

Таблица 1.11

Биты регистра OPTION PIC16F84A и PIC12CE673

Бит	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
Имя	-RBPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0
-RBPU: Включение подтягивающих резисторов на входах порта В: 1 – подтягивающие резисторы отключены; 0 – подтягивающие резисторы включены								

Регистр *OSCCAL*. В регистре *OSCCAL* размещаются биты калибровки внутреннего RC генератора 4 МГц (6-разрядная константа загружается в регистр *OSCCAL* для его коррекции) (табл. 1.12).

Таблица 1.12

Биты регистра *OSCCAL_REG* PIC12C508

Бит	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4–0
Имя	CAL3	CAL2	CAL1	CAL0	–
Описание	Биты калибровки внутреннего RC генератора				–

Регистры *PCLATH* и *PCL*. На рис. 1.2 показаны способы загрузки в счетчик команд PC:

- запись в счетчик команд PC происходит при записи значения в регистр *PCL* ($PCLATH <4:0> \rightarrow PCH$);
- запись значения в счетчик команд PC происходит при выполнении команды *CALL* или *GOTO* ($PCLATH <4:3> \rightarrow PCH$).

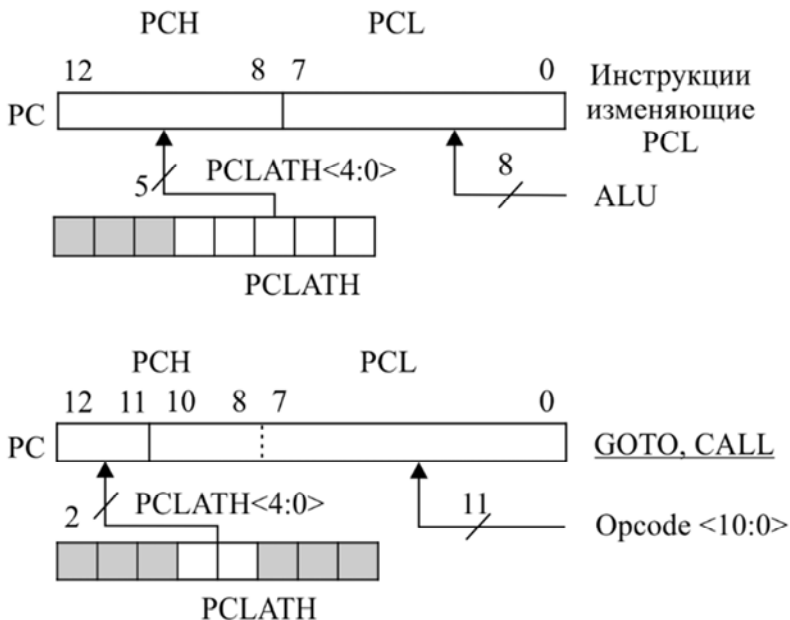


Рис. 1.2. Запись значения в счетчик команд PC

13-разрядный регистр счетчика команд PC указывает адрес выполняемой инструкции. Младший байт счетчика команд PCL доступен для чтения и записи. Старший байт PCN, содержащий <12:8> биты счетчика команд PC, не доступен для чтения и записи. Все операции с регистром PCN происходят через дополнительный регистр PCLATH. При любом виде сброса микроконтроллера счетчик команд PC очищается.

Вычисляемый переход может быть выполнен командой приращения к регистру PCL (например, ADDWF PCL). При выполнении табличного чтения вычисляемым переходом следует следить, чтобы значение PCL не превысило границу блока памяти (256 байт).

Регистры организации косвенной адресации INDF и FSR. Для выполнения косвенной адресации необходимо обратиться к физически не реализованному регистру INDF. Обращение к регистру INDF фактически вызовет действие с регистром, адрес которого указан в FSR. Косвенное чтение регистра INDF (FSR = 0) даст результат 00h. Косвенная запись в регистр INDF не вызовет никаких действий (вызывает воздействия на флаги ALU в регистре STATUS).

Стек. Микроконтроллеры PIC12F675 имеют 8-уровневый 13-разрядный аппаратный стек. Стек не имеет отображения на память программ и память данных, нельзя записать или прочитать данные из стека. Значение счетчика команд заносится в вершину стека при выполнении инструкций перехода на подпрограмму (CALL) или обработке прерываний. Чтение из стека и запись в счетчик команд PC происходит при выполнении инструкций возвращения из подпрограммы или обработки прерываний (RETURN, RETLW, RETFIE), при этом значение регистра PCLATH не изменяется. Стек работает как циклический буфер. После 8 записей в стек, девятая запись заменит первую, а десятая – вторую и так далее.

1.1.4 Дополнительные модули микроконтроллеров Microchip.

Биты конфигурации предназначены для установки режимов работы некоторых модулей микроконтроллера, которые не должны изменяться в процессе выполнения всего программного кода.

Эти режимы настраиваются в ходе программирования микроконтроллера и изменению не подлежат. Слово конфигурации состоит из 14 бит. Биты конфигурации расположены в памяти программ по адресу 2007h, и могут быть установлены в «0» или в «1».

Биты конфигурации PIC12F675 представлены в табл. 1.13.

Таблица 1.13

Биты конфигурации (2007h) PIC12F675

№ бита	Имя	Назначение		
1	2	3		
Бит 13–12	BG1:BG1	Биты калибровки сброса по снижению напряжения питания: 00 – нижний предел калибровки; 11 – верхний предел калибровки		
Бит 11–9	–	Не используется		
Бит 8	-CPD	Бит защиты EEPROM памяти данных		
Бит 7	-CP	Бит защиты памяти программ		
Бит 6	BODEN	Разрешение сброса по снижению напряжения питания		
Бит 5	-MCLRE	Выбора режима работы вывода GP3/-MCLR: 1 – GP3/-MCLR работает как -MCLR; 0 – GP3/-MCLR работает как цифровой канал порта ввода/вывода, -MCLR внутренне подключен к VDD		
Бит 4	-PWRTE	Разрешение работы таймера включения питания		
Бит 3	WDTE	Разрешение работы сторожевого таймера		
Бит 2–0	FOSC2-0	Выбор сигнала тактового генератора		
		111	RC генератор	вывод GP4/OSC2/CLKOUT работает как CLKOUT, RC цепочка подключается к выводу GP5/OSC1/CLKIN
		110	RC генератор	вывод GP4/OSC2/CLKOUT работает как канал порта ввода/вывода, RC цепочка подключается к выводу GP5/OSC1/CLKIN
		101	INTOSC генератор	вывод GP4/OSC2/CLKOUT работает как CLKOUT, вывод GP5/OSC1/CLKIN работает как канал порта ввода/вывода
		100	INTOSC генератор	вывод GP4/OSC2/CLKOUT работает как канал порта ввода/вывода, вывод GP5/OSC1/CLKIN работает как канал порта ввода/вывода
		011	EC генератор	вывод GP4/OSC2/CLKOUT работает как канал порта ввода/вывода, вывод GP5/OSC1/CLKIN работает как CLKIN
		010	HS генератор	резонатор подключается к выводам GP4/OSC2/CLKOUT, GP5/OSC1/CLKIN
		001	XT генератор	резонатор подключается к выводам GP4/OSC2/CLKOUT, GP5/OSC1/CLKIN
		000	LP генератор	резонатор подключается к выводам GP4/OSC2/CLKOUT, GP5/OSC1/CLKIN

Биты конфигурации PIC12C508.

MCLRE – Бит выбора режима работы вывода -MCLR:

1 – MCLR включен;

0 – MCLR подключен к V_{DD} (внутрисхемно).

CP – Бит защиты памяти программ:

1 – защита памяти программ выключена;

0 – защита памяти программ включена.

WDTE – Бит разрешения работы сторожевого таймера:

1 – WDT включен;

0 – WDT выключен.

FOSC1, FOSC0 – Биты выбора режима тактового генератора.

EXTRC – внешняя RC цепочка.

INTRC – внутренняя RC цепочка.

XT, LP – внешний резонатор.

Биты конфигурации PIC16F84A.

PWRTE – Бит разрешения задержки при включении питания:

1 – задержка отключена;

0 – задержка включена.

CP – Бит защиты памяти программ:

1 – защита памяти программ выключена;

0 – защита памяти программ включена.

WDTE – Бит разрешения работы сторожевого таймера:

1 – WDT включен;

0 – WDT выключен.

FOSC1, FOSC0 – Биты выбора режима тактового генератора:

RC – внешняя RC цепочка;

HS, XT, LP – внешний резонатор.

Биты конфигурации PIC12CE673.

PWRTE – Бит разрешения задержки при включении питания:

1 – задержка отключена;

0 – задержка включена.

MCLRE – Бит выбора режима работы вывода -MCLR:

1 – MCLR включен;

0 – MCLR подключен к V_{DD} (внутресхемно).

CP – Бит защиты памяти программ:

1 – защита памяти программ выключена;

0 – защита памяти программ включена.

WDTE – Бит разрешения работы сторожевого таймера:

1 – WDT включен;

0 – WDT выключен.

FOSC2, FOSC0 – Биты выбора режима тактового генератора.

111 – EXTRC, Clockout on OSC2;

110 – EXTRC, OSC2 is I/O;

101 – INTRC, Clockout on OSC2;

100 – INTRC, OSC2 is I/O;

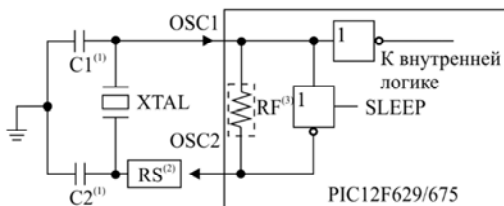
010 – HS Oscillator;

001 – XT Oscillator;

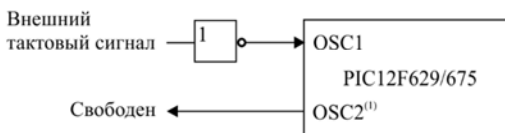
000 – LP Oscillator.

1.1.4.1 Тактовый генератор. Микроконтроллеры PIC12F675 могут работать в одном из восьми режимов тактового генератора. Выбрать режим тактового генератора можно при программировании микроконтроллера в слове конфигурации (FOSC2, FOSC1 и FOSC0).

Кварцевый/керамический резонатор. В режимах тактового генератора XT, LP и HS кварцевый или керамический резонатор подключается к выводам OSC1, OSC2 (рис. 1.3, а). Для микроконтроллеров PIC12F675 нужно использовать резонаторы с параллельным резонансом. В режимах XT, LP и HS микроконтроллер также может работать от OSC1 (рис. 1.3, б).



а



б

Рис. 1.3. Подключение кварцевого/керамического резонатора:

а – и внешнего источника тактового сигнала;

б – в HS, XT и LP режиме тактового генератора

RC генератор. В приложениях, не требующих высокостабильной тактовой частоты, допустимо использовать RC режим генератора, уменьшающий стоимость устройства. Частота RC генератора зависит от напряжения питания, значения сопротивления R_{EXT} , емкости C_{EXT} . Дополнительно частота будет варьироваться в некоторых пределах из-за технологического разброса параметров кристалла.

Внутренний RC генератор 4 МГц. Внутренний тактовый генератор формирует тактовый сигнал с частотой 4 МГц.

В последней ячейке памяти программ сохраняется калибровочная константа для внутреннего RC генератора. Калибровочная константа сохраняется в виде команды RETLW XX (где XX – калибровочное значение) в регистр OSCCAL. Калибровочная константа записывается в PIC12F675 на вывод GP4/OSC2/CLKOUT.

1.1.4.2 Система сброса. Различают следующие виды сбросов:

- сброс по включению питания POR;
- сброс по сигналу -MCLR в нормальном режиме работы;
- сброс по сигналу -MCLR в SLEEP режиме;
- сброс от WDT в нормальном режиме работы;
- сброс по снижению напряжения питания BOR.

Большинство регистров сбрасываются в начальное состояние при всех видах сбросов (кроме сброса WDT в SLEEP режиме). Сброс WDT в SLEEP режиме рассматривается как возобновление нормальной работы и на значение регистров не влияет. Для определения вида сброса можно использовать биты -TO и -PD.

Внешний сброс (-MCLR). На входе -MCLR есть внутренний фильтр, не пропускающий короткие импульсы. Напряжение на выводе -MCLR больше, чем указано в спецификациях, что может привести к сбросу микроконтроллера. Следует отметить, что сброс WDT не управляет выводом -MCLR.

Сброс по включению питания (POR). Интегрированная схема POR удерживает микроконтроллер в состоянии сброса, пока напряжение V_{DD} не достигнет требуемого уровня. Для включения схемы POR необходимо соединить вывод -MCLR с V_{DD} через резистор, не требуя внешней RC цепочки, обычно используемой для сброса. Когда микроконтроллер переходит в режим нормальной работы, рабочие параметры (напряжение питания, частота и т. д.) должны соответствовать номинальным. В случае, если они не удовлетворяют требованиям, микроконтроллер находится в состоянии сброса.

Таймеры включения питания (PWRT) и запуска генератора (OST). Таймер включения питания обеспечивает задержку в 72 миллисекунды (номинальное значение) по сигналу схемы сброса POR или BOR. Таймер включения питания работает от внутреннего RC генератора и удерживает микроконтроллер в состоянии сброса по активному сигналу от PWRT. Задержка PWRT позволяет достигнуть напряжению V_{DD} номинального значения.

Таймер запуска генератора обеспечивает задержку в 1024 такта генератора (вход OSC1) после окончания задержки от PWRT (если она включена). Это гарантирует, что частота кварцевого/керамического резонатора стабилизировалась. Задержка OST включается только в режимах HS, XT и LP тактового генератора после сброса POR или выхода микроконтроллера из режима SLEEP.

Детектор пониженного напряжения питания (BOD). Микроконтроллеры PIC12F675 имеют интегрированную схему сброса по снижению напряжения питания. В случае, если напряжение V_{DD} опускается ниже V_{BOR} на время большее (или равное) T_{BOR} , то происходит сброс по снижению напряжения питания.

При любом виде сброса микроконтроллер находится в состоянии сброса, пока напряжение V_{DD} не будет выше V_{BOR} . После нормализации напряжения питания микроконтроллер находится в состоянии сброса еще 72 миллисекунды, если $\text{-PWRT} = 0$.

Если напряжение питания V_{DD} стало ниже V_{BOR} во время работы таймера по включению питания, микроконтроллер возвращается в состояние сброса BOR, а таймер инициализируется заново. Каждый переход напряжения питания V_{DD} через границу V_{BOR} инициализирует PWRT, создавая задержку в 72 миллисекунды. При включении схемы BOD всегда нужно включать таймер PWRT.

Последовательность удержания микроконтроллера в состоянии сброса. При включении питания выполняется следующая последовательность удержания микроконтроллера в состоянии сброса: сброс POR, задержка PWRT (если она разрешена), задержка OST. Полное время задержки изменяется в зависимости от режима работы тактового генератора и состояния бита -PWRT .

Если сигнал -MCLR удерживается в низком уровне достаточно долго (дольше времени всех задержек), после перехода -MCLR в высокий уровень программа начнет выполняться немедленно, что

может быть полезно при одновременном запуске нескольких микроконтроллеров, работающих параллельно.

1.1.4.3 сторожевой таймер (WDT). Встроенный сторожевой таймер WDT работает от отдельного *RC* генератора, не требующего внешних компонентов. Это позволяет работать сторожевому таймеру WDT при выключенном тактовом генераторе (выводы OSC1, OSC2) в SLEEP режиме. В нормальном режиме работы при переполнении WDT происходит сброс микроконтроллера. Если микроконтроллер находится в SLEEP режиме, переполнение WDT выводит его из режима SLEEP с продолжением нормальной работы.

WDT имеет номинальное время переполнения 18 мс (без предделителя). Если требуется большее время переполнения WDT, необходимо программно подключить предделитель в регистре OPTION_REG с коэффициентом деления от 1:2 до 1:128. С включенным предделителем время переполнения может достигать 2,3 с.

CLRWDТ и SLEEP сбрасывают сторожевой таймер и предделитель, если он подключен к WDT, откладывая сброс устройства.

1.1.4.4 Режим энергосбережения (SLEEP). Переход в режим энергосбережения происходит по команде SLEEP. При переходе в режим SLEEP выполняется следующее:

- сторожевой таймер WDT сбрасывается, но продолжает работать;
- в регистре STATUS бит -PD сбрасывается в «0»;
- бит -TO устанавливается в «1»;
- тактовый генератор выключен;
- порты ввода/вывода остаются в том же состоянии, что и до выполнения команды SLEEP (высокий уровень, низкий уровень, третье состояние).

Для снижения энергопотребления в SLEEP режиме все каналы ввода/вывода должны быть подключены к V_{DD} или V_{SS} . При отсутствии токов из внешней схемы через выводы портов, выходы модуля компараторов и источника опорного напряжения выключены. Выводы, находящиеся в третьем состоянии должны иметь высокий или низкий уровень сигнала, чтобы избежать токов переключения входных буферов. Должны учитываться внутренние подтягивающие резисторы, включенные на входах GPIO. На входе -MCLR должен быть высокий уровень сигнала.

Микроконтроллер выйдет из режима SLEEP по одному из следующих событий:

- внешний сброс по сигналу на входе -MCLR;
- переполнение сторожевого таймера WDT (если он разрешен);
- периферийное прерывание (по сигналу INT, изменение уровня сигнала на входах GPIO и др.).

Внешний сброс по сигналу -MCLR вызывает сброс микроконтроллера. Два других события вызывают продолжение выполнения программы. Биты -TO и -PD в регистре STATUS могут использоваться для определения причины сброса. Бит -PD сбрасывается в «0» при переходе в режим SLEEP. Бит -TO сбрасывается в «0», если произошло переполнение WDT.

При выполнении команды SLEEP происходит предвыборка следующей инструкции (PC + 1). Если прерывание должно вывести микроконтроллер из режима SLEEP, соответствующий бит разрешения прерывания устанавливается в «1». Микроконтроллер выходит из режима SLEEP независимо от состояния бита GIE. Если GIE = 0, выполняется следующая инструкция после SLEEP без перехода по вектору прерываний. Если GIE = 1, исполняется следующая инструкция после SLEEP и происходит переход на подпрограмму обработки прерываний (адрес 0004h). Когда выполнение какой-либо команды при выходе из режима SLEEP нежелательно, необходимо поле команды SLEEP вставить NOP.

При выходе из режима SLEEP сторожевой таймер WDT сбрасывается, независимо от источника «пробуждения».

1.1.5 Система и формат команд. PIC12F675 имеет систему команд аккумуляторного типа, разделенную на три основных группы:

- байт ориентированные команды;
- бит ориентированные команды;
- команды управления и операции с константами.

Каждая команда состоит из одного 14-разрядного слова, разделенного на код операции (OPCODE), определяющий тип команды и один или несколько операндов, определяющие операцию.

Для байт ориентированных команд «*f*» является указателем регистра, а «*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 сохраняе-

ты в регистре, который используется в команде. В бит ориентированных командах «*b*» определяет номер бита участвующего в операции, а «*f*» – указатель регистра, который содержит этот бит.

В командах управления или операциях с константами «*k*» представляет 8 или 11 бит константы или значения литералов.

Система команд включает 35 операций (табл. 1.14). Все команды выполняются за один машинный цикл, кроме команд условия, в которых получен истинный результат и инструкций, изменяющих значение счетчика команд PC. В случае выполнения команды за два машинных цикла, во втором цикле выполняется инструкция NOP. Один машинный цикл состоит из четырех тактов генератора.

Таблица 1.14

Команды микроконтроллеров семейства Microchip

Команда	Операнды	Операция	Флаги	Описание
1	2	3	4	5
ADDLW <i>k</i>	$0 < k < 255$	$(W) + k \rightarrow (W)$	C, DC, Z	Содержимое регистра W складывается с 8-разрядной константой « <i>k</i> ». Результат сохраняется в регистре W
ADDWF <i>f, d</i>	$0 < f < 127$ <i>d</i> [0, 1]	$(W) + (f) \rightarrow$ (<i>dest</i>)	C, DC, Z	Сложить содержимое регистров W и <i>f</i> . Если <i>d</i> = 0, результат сохраняется в регистре W. Если <i>d</i> = 1, результат сохраняется в регистре <i>f</i>
ANDLW <i>k</i>	$0 < k < 255$	$(W). \text{AND. } k \rightarrow$ (W)	Z	Выполняется побитное «И» содержимого регистра W и 8-разрядной константы « <i>k</i> ». Результат сохраняется в регистре W
ANDWF <i>f, d</i>	$0 < f < 127$ <i>d</i> [0, 1]	$(W). \text{AND. } (f) \rightarrow$ (<i>dest</i>)	Z	Выполняется побитное «И» содержимого регистров W и <i>f</i> . Если <i>d</i> =0, результат сохраняется в регистре W. Если <i>d</i> = 1, результат сохраняется в регистре <i>f</i>
BCF <i>f, b</i>	$0 < f < 127$ $0 < b < 7$	$0 \rightarrow (f < b >)$	–	Очистить бит « <i>b</i> » в регистре <i>f</i>
BSF <i>f, b</i>	$0 < f < 127$ $0 < b < 7$	$1 \rightarrow (f < b >)$	–	Установить бит « <i>b</i> » в регистре <i>f</i>

Продолжение табл. 1.14

1	2	3	4	5
BTFSCL f, b	$0 < f < 127$ $0 < b < 7$	Пропустить, если (f) = 0	–	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 цикле 2 выполняется NOP
BTFSCL f, b	$0 < f < 127$ $0 < b < 7$	Пропустить, если (f) = 1	–	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 цикле 2 выполняется NOP
CLRF f	$0 < f < 127$	00h → (f); 1 → Z	Z	Очистить содержимое регистра f и установить флаг Z/
CLRW	–	00h → (W) 1 → Z	Z	Очистить содержимое регистра W и установить флаг Z
CLRWDI	–	00h → WDT, 00h → предделитель WDT; 1 → -TO, 1 → -PD	-TO, -PD	Инструкция CLRWDI сбрасывает WDT и предделитель, если он подключен к WDT. В регистре STATUS устанавливает биты -TO и -PD
CALL k	$0 < k < 2047$	(PC) + 1 → TOS, k → PC <10:0>; (PCLATH <4:3>) → PC <12:11>	–	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 счетчик команд PC <12:11> из PCLATH
COMF f, d	$0 < f < 127$ d [0,1]	(-f) → (dest)	Z	Инvertировать все биты в регистре f. Если d = 0, результат сохраняется в регистре W. Если d = 1, результат сохраняется в регистре f
GOTO k	$0 < k < 2047$	k → PC <10:0>; (PCLATH <4:3>) → PC <12:11>	–	Выполнить безусловный переход. Одиннадцать бит адреса загружаются из кода команды в счетчик команд PC <10:0>. Два старших бита загружаются в счетчик PC <12:11> из регистра PCLATH. Команда GOTO выполняется за два цикла

Продолжение табл. 1.14

1	2	3	4	5
NOP	–	–	–	Нет операции
IORLW k	$0 < k < 255$	(W). OR. k → (W)	Z	Выполняется побитное «ИЛИ» содержимого регистра W и 8-разрядной константы «k». Результат сохраняется в регистре W
IORWF f, d	$0 < f < 127$ d [0,1]	(W). OR. (f) → (dest)	Z	Выполняется побитное «ИЛИ» содержимого регистров W и f. Если d = 0, результат сохраняется в регистре W. Если d = 1, результат сохраняется в регистре f
DECF f, d	$0 < f < 127$ d [0,1]	(f) - 1 → (dest)	Z	Декрементировать содержимое регистра f. Если d = 0, результат сохраняется в регистре W. Если d = 1, результат сохраняется в регистре f
DECFSZ f, d	$0 < f < 127$ d [0,1]	(f) - 1 → (dest); пропустить, если результат равен 0	–	Декрементировать содержимое регистра f. Если d = 0, результат сохраняется в регистре W, если d = 1, то сохраняется в регистре f. Если результат не равен «0», то исполняется следующая инструкция. Если равен «0», то следующая инструкция не выполняется. Во втором цикле выполняется NOP
INCF f, d	$0 < f < 127$ d [0,1]	(f) + 1 → (dest)	Z	Инкрементировать содержимое регистра f. Если d = 0, результат сохраняется в регистре W. Если d = 1, результат сохраняется в регистре f
INCFSZ f, d	$0 < f < 127$ d [0,1]	(f) + 1 → (dest); пропустить, если результат равен 0	–	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

Продолжение табл. 1.14

1	2	3	4	5
MOVWF f, d	$0 < f < 127$ d [0,1]	(f) → (dest)	Z	Содержимое регистра f пересылается в регистр адресата. Если d = 0, значение сохраняется в регистре W. Если d = 1, значение сохраняется в регистре «f».
MOVLW k	$0 < k < 255$	k → (W)	–	Переслать константу «k» в регистр W. В неиспользуемых битах устанавливает «0»
MOVWF f	$0 < f < 127$	(W) → (f)	–	Переслать содержимое регистра W в регистр f
RETFIE	–	TOS → PC; 1 → GIE	–	Возврат из подпрограммы обработки прерываний. Вершина стека TOS загружается в счетчик команд PC. Устанавливается в «1» флаг глобального разрешения прерываний GIE (INTCON<7>). Выполняется за 2 цикла
RETLW k	$0 < k < 255$	k → (W); TOS → PC	–	В регистр W загружается 8-разрядная константа. Вершина стека TOS загружается в счетчик команд PC. Инструкция выполняется за 2 цикла
RETURN	-	TOS → PC	–	Возврат из подпрограммы. Вершина стека TOS загружается в счетчик PC. Инструкция выполняется за 2 цикла
RLF f, d	$0 < f < 127$ d [0,1]	–	C	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 f, d	$0 < f < 127$ d [0,1]	–	C	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 f
SUBLW k	$0 < k < 255$	k – (W) → (W)	C, DC, Z	Вычесть содержимое регистра W из 8-разрядной константы «k». Результат сохраняется в регистре W

1	2	3	4	5
SUBWF f, d	$0 < f < 127$ d [0,1]	(f) – (W) → (dest)	C, DC, Z	Вычесть содержимое регистра W из регистра f. Если d = 0, результат сохраняется в регистре W. Если d = 1, результат сохраняется в регистре f
SWAPF f, d	$0 < f < 127$ d [0,1]	(f <3:0> → (dest <7:4>); (f <7:4> → (dest <3:0>)	–	Поменять местами старший и младший полубайты регистра f. Если d = 0, результат сохраняется в регистре W. Если d = 1, результат сохраняется в регистре f
SLEEP	–	00h → WDT; 1 → -TO; 0 → -PD	-TO, -PD	Сбросить флаг включения питания -PD в «0». Установить флаг переполнения WDT -TO в «1». Очистить таймер WDT и его предделитель. Перевести микроконтроллер в режим SLEEP и выключить тактовый генератор
XORLW k	$0 < k < 255$	(W). XOR. k → (W)	Z	Выполняется побитное «исключающее ИЛИ» содержимого регистра W и 8-разрядной константы «k». Результат сохраняется в регистре W
XORWF t, d	$0 < f < 127$ d [0,1]	(W). XOR. (f) → (dest)	Z	Выполняется побитное «исключающее ИЛИ» содержимого регистров W и f. Если d = 0, результат сохраняется в регистре W. Если d = 1, результат сохраняется в регистре f

Любая команда, которая определяет регистр памяти данных как часть команды, выполняется по принципу «чтение–модификация–запись», т. е. выполняется чтение регистра, изменяются данные, а затем результат сохраняется в регистре назначения (зависит от состояния бита «d»).

1.2. Интегрированная среда разработки IDE MPLab

MPLab – это интегрированная среда разработки (IDE, Integrated Development Environment) для семейства микроконтроллеров PICmicro, позволяющая писать, отлаживать и оптимизировать

программы для разработок, построенных на микроконтроллерах семейства Microchip. На рис. 1.4 представлена структурная схема проекта, разработанного в среде IDE MPLab.

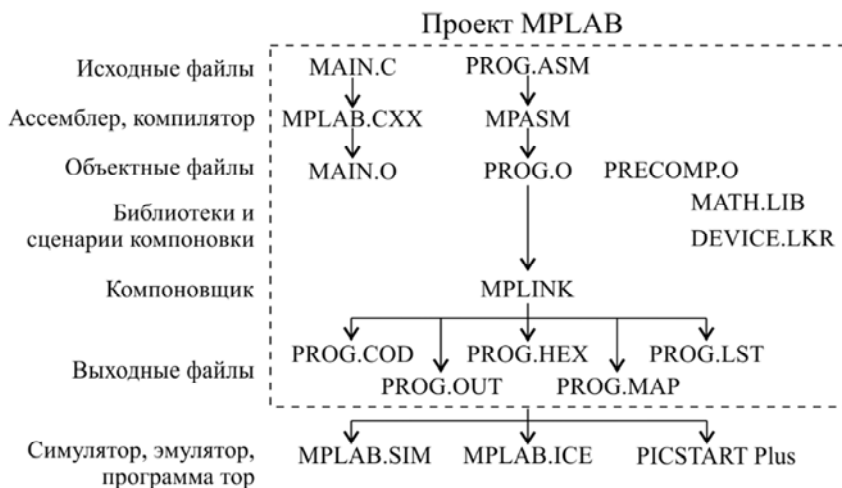


Рис. 1.4. Структурная схема проекта MPLAB

MPLab представляет собой законченную среду разработки, в которой интегрировано несколько инструментальных средств:

- MPLAB менеджер проектов (*project Manager*). Используется для создания проектов и работы со связанными файлами;
- MPLAB редактор (*Editor*). Используется для создания и редактирования текстовых файлов;
- MPLAB-SIM симулятор (*Simulator*). Позволяет моделировать выполнение команд и входные или выходные сигналы контроллера;
- MPLAB-ICE внутрисхемный эмулятор (*Emulator*) позволяет, используя дополнительные аппаратные средства и компьютер, заменять микроконтроллер в разрабатываемом устройстве в режиме реального времени. MPLAB-ICE самый новый эмулятор от Microchip;
- MPASM универсальный ассемблер;
- MPLINK компоновщик (*Linker*). Создает законченное приложение, связывая перемещаемые модули MPASM, MPLAB-C17 и MPLAB-C18;
- MPLIB управляет пользовательскими библиотеками;

- MPLAB-CXX компилятор C. Позволяет включать в проект исходные тексты, написанные на языках C и ассемблер;
- PRO MATE II и PICSTART Plus программаторы;
- PICMASTER и PICMASTER-CE внутрисхемные эмуляторы;
- инструментальные средства сторонних производителей.

1.2.1 Панель управления IDE MPLab. Основная панель управления IDE MPLAB имеет вкладки, рассмотренные ниже.

Меню *Project* – пункты для управления проектом и конфигурацией. Изменение установок созданного проекта *Edit Project*. *Make Project* – компилирование исходных файлов для создания одного hex файла, *Build All* – компилирование всех файлов, *Build Node* – компилирование одного выборочного файла. Меню *Install Language Tool* предназначено для выбора языка программирования и конфигурации.

Меню *Debugger* – после компилирования проекта, разрабатываемое устройство может работать неправильно. Т. е. потребуется отладка кода. Для симулирования работы микроконтроллера возможно использование MPLAB-SIM или внутрисхемный эмулятор MPLAB-ICE, PICMASTER и др. Меню *Debugger* содержит все опции, необходимые при отладке кода с симулятором или эмулятором.

Execute содержит следующие опции:

- *Execute an Opcode* – выполнить введенную Вами команду;
- *Conditional Break* – выполнять программу в непрерывном режиме пока не выполнится введенное условие или не будет нажат *Halt*. При выборе этой опции откроется окно для конфигурирования.

Simulator Stimuls позволяет имитировать внешние сигналы на входных выводах (портах) микроконтроллера.

Center Debug Location позволяет переместить указатель текущей выполняемой команды в центр окна. Работает с окнами исходных файлов, программной памяти и абсолютного листинга.

Breakpoint Settings – конфигурация точек останова (прерывания). Можно устанавливать до 16 точек. Если останов в выбранной точке не происходит, то проверьте установлен ли соответствующий флаг в окне конфигурации (*Options > Development Mode*).

Trace Point Settings – конфигурация трассировки, то есть записи в соответствующем окне последовательности выполняемых команд.

Trigger In/Out Settings u Trigger Out Point Settings возможны при работе с внутрисхемным эмулятором и управляют конфигурацией прерывания по внешним сигналам и др.

Clear All Points позволяет очистить все установленные точки останова и трассировки.

Complex Trigger Settings u Enable Code Coverage предназначено для работы с эмулятором.

Clear Program Memory (Ctrl + Shift + F2) – очистить память.

System Reset (Ctrl + Shift + F3) – системный сброс. Пересбрасывается MPLAB, симулятор или подключенный эмулятор. Затем производится инициализация всего оборудования, как при старте MPLAB.

Power-On-Reset (Ctrl + Shift + F5) – сброс отлаживаемого устройства (симулятор или эмулятор), аналогичный состоянию при подаче напряжения питания.

PICSTART Plus – меню конфигурации и управления фирменным программатором *PICSTART Plus*.

Option – меню для конфигурирования самого MPLAB.

Меню Programmer options. Выбор программатора, его конфигурация и выбор порта для подключения.

Меню Tools. Опции меню «Tools» позволяют запустить файлы в ДОС строке и проверить возможность обмена информацией с внешними устройствами (программаторами и эмуляторами).

Меню Window обеспечивает открытие окон для обзора программной памяти, содержимого стека и регистров. Все опции меню *Window* доступны в режиме симулятора или внутрисхемного эмулятора. В режиме *Editor Only* доступны окна *Listing* и *Symbol list*. Подробнее об возможных опциях меню *Window*:

– *Program Memory* – окно памяти программы;

– *Trace Memory* – окно результатов трассировки. Если счетчик программы (PC) попадает в область, отмеченную для трассировки, то в этом окне сохраняется порядок выполнения команд;

– *EEPROM Memory* – окно просмотра содержимого EEPROM (электрически программируемая область памяти, сохраняется при отключении питания. Присутствует в кристаллах, содержащих в обозначении букву F, например, 16F84);

– *Calibration Data* – окно просмотра калибровочных данных;

– *Calibration Memory* – окно просмотра калибровочной памяти для кристаллов, содержащих таковую (PIC14400, например);

– *Absolute Listing* – окно просмотра абсолютного листинга – полного отчета о компилировании;

– *Map File* – окно просмотра схемы памяти. По умолчанию этот файл при компилировании не генерируется. Чтобы он был доступен, надо вызвать окно редактирования проекта *Project > Edit Project*, подсветить выходной HEX-файл и выбрать его свойства *Node Properties*. В открывшемся окне поставить галочку в графе *ON* для *Cross Reference File* и в графе *DATA* ввести имя файла с именем Вашего проекта и расширением *.map*, например, *myprog.map*. После изменений перекомпилируйте проект;

– *Stack* – просмотр стека. Не стоит забывать, что у некоторых микроконтроллеров глубина стека всего 2 (например, PIC16C505). Это значит, что можно вызывать только одну подпрограмму из подпрограммы. При третьем вызове информация о возврате теряется, и программа будет работать самым неожиданным образом;

– *File Registers* – окно просмотра содержимого регистров общего назначения (RAM). Не забудьте, что вид просмотра можно выбрать из меню, щелкнув в левом верхнем углу окна;

– *Show Symbol List* (Ctrl + F8) – окно списка символов. Все используемые переменные, регистры, метки, константы и др.;

– *Stopwatch* – окно просмотра временных параметров. Показывает тактовую частоту (выставляется в окне конфигурации микроконтроллера) и время, прошедшее с начала выполнения программы или с момента обнуления в тактах и микросекундах. Очень удобно при отладке программы, когда используются счетчики, таймеры, прерывания и все, связанное со временем;

– *Project Window* – окно файла проекта. Включает пути к файлам, подключенные файлы, даты и другая информация;

– *Watch Windows* – меню управления окнами просмотра;

– *Modify* – окно модификации регистров. Можно ввести абсолютный адрес, имя переменной или регистра. Можно выбрать тип модифицируемой памяти и заполнить область.

– *Tile Horizontal, Tile Vertical, Cascade, Iconize All, Arrange Icons* – опции управления открытыми окнами: упорядочить по горизонтали, по вертикали, уложить каскадом, свернуть все;

– *Special Function Registers* – окно просмотра регистров специального назначения (SFR);

– *Open Windows* – открыть загруженное или свернутое окно.

Меню Help. Вызов помощи и справок.

1.2.2 Панель инструментов Toolbar. Существуют следующие наборы кнопок: редактирование, отладка, управление проектом. Для конфигурирования панели инструментов выберите *Option > Environment Setup* и вкладку *General*. Можно выбрать месторасположение панели или, нажав кнопку *Layout*, добавить или убрать кнопки.

Графическое обозначение и назначение кнопок в соответствии с каждой панелью представлены в табл. 1.15.

Таблица 1.15

Графическое обозначение и назначение кнопок

Панель <i>Edit</i>		Панель <i>Debug</i>	
Кнопка	Действие	Кнопка	Действие
1	2	3	4
	создать новый файл		запустить программу в непрерывном режиме
	открыть файл		остановить выполнение программы
	сохранить файл		выполнить очередную команду
	вырезать выделение в буфер		выполнить команду, не опускаясь в подпрограммы
	копировать выделение в буфер		сброс микроконтроллера
	вставить из буфера		модифицировать счетчик программы (PC)
	печать		выполнить введенную команду
	поиск текстовой последовательности		создать новое окно просмотра
	повторить последний поиск		модифицировать регистр, переменную
	поиск и замена текстовой последовательности		установить точку останова
	повторить поиск и замену текстовой последовательности		условная точка останова

1	2	3	4
	вернуться на одно действие назад		установить точки трассировки
	сдвинуть строку вправо на расстояние табуляции		остановить трассировку
	сдвинуть строку влево на расстояние табуляции		очистить все точки останова, трассировки, защелки
	переместиться на строку с номером...		установить защелки
	включить нумерацию строк в файлах		системный сброс
	вызвать справку MPLAB		вызвать описание версии MPLAB
	создать проект		открыть проект
	открыть проект		сохранить проект
	закрыть проект		поиск текстовой последовательности
	сохранить проект		вырезать выделение в буфер
	редактировать проект		копировать выделение в буфер
	компилировать проект		вставить из буфера
	компилировать все исходные файлы		сохранить файл
	компилировать один исходный файл		окно просмотра регистров общего назначения
	установить язык проекта		окно просмотра памяти программы
	вызвать помощь по ассемблеру MPASM		окно просмотра регистров специального назначения
			компилировать проект

1.2.3 Представление чисел. В исходном тексте представлять числа можно по-разному и в разных системах счисления (табл. 1.16).

Таблица 1.16

Способ представления чисел

Формат	Синтаксис	Пример
1	2	3
Десятичный	D «число», число	D «100», 100
Шестнадцатиричный	H «число», 0xчисло	H «f9», 0xAF00
Восьмиричный	O «число»	O «777»
Двоичный	B «число»	B «11110000»
Символьный	«символ», A «символ»	«C», A «C»

1.2.4 Директивы языка MPASM. Директивы ассемблера располагаются в тексте исходного файла для расширения функций ассемблера, определения и управления процессом (табл. 1.17). *Общее правило* – директивы не должны располагаться в первой позиции строки в тексте (иначе будут определены как метка). Арифметические операторы приведены в табл. 1.18.

Таблица 1.17

Директивы языка MPASM

Директива	Описание	Пример
1	2	3
<i>Директивы управления</i>		
CONSTANT	Определение символьной константы	constant cnt = 255
#DEFINE	Определение текстовой последовательности для замены	#define snd portsnd, 1
END	Конец блока программы	end
EQU	Определение константы	temp equ 0xF0
ERROR	Сообщение об ошибке	error "error line"
ERROR LEVEL	Установка типа сообщений об ошибках в файле листинга и файле ошибок	errorlevel 1, -202
INCLUDE	Вставить другой файл источника	Include <addmain.asm>
LIST	Определение формата (см. MPASM Help) или разрешение вывод, а если было предварительно запрещено	list p = 17c42, f = INHX32, r = DEC
MESSG	Создать пользовательское сообщение	messg "see here!"

Окончание табл. 1.17

1	2	3
NOLIST	Запретить вывод	nolist
ORG	Установить начальный адрес программы	org 0x100
PAGE	Вставить страницу в файл листинга	page
PROCESSOR	Установить тип микроконтроллера	processor 16F84
VARIABLE	Определение символьной переменной	variable temp = 0xF0
RADIX	Установить систему счисления по умолчанию для выражения данных	radix dec
SET	Определение константы. Аналогична EQU, но в последствии можно переопределить	temp set b «00110011»
SPACE	Вставить пустые строки в файл листинга	space 3
SUBTITLE	Вставить второй заголовок в файл листинга	subtitle «Main Project»
TITLE	Вставить заголовок в файл листинга	title "Project Of PIC"
#UNDEFINE	Удаление определенной текстовой последовательности	#undefine snd
<i>Условия</i>		
ELSE	Начало блока альтернативного условия	else
ENDIF	Завершение блока условия	endif
ENDW	Завершение цикла ПОКА	endw
IF	Начало блока условия	if version == 100
IFDEF	Выполнить, если определено	ifdef testing
IFNDEF	Выполнить, если не определено	ifndef testing
WHILE	Цикл ПОКА	while i < count
<i>Данные</i>		
CBLOCK	Определение блока констант	cblock 0x20
CONFIG	Описание бит конфигурации	config H «FFFF»
DATA	Создание числовых и текстовых данных	txt data "please", 0x30
DB	Определение байта данных	temp db 0xFF
DE	Определение данных в EEPROM	temp de 0xF0, 0xF1
DT	Определение таблицы	temp dt «text», 0, 0x30
DW	Определение слова (2 байта) данных	temp dw 39, «text»
ENDC	Окончание блока констант	endc
FILL	Заполнение области константой	fill 0x1009, 5
IDLOCS	Определение ID	idlocs H «FFEE»
RES	Резервирование памяти	buffer res 64
<i>Макросы</i>		
ENDM	Окончание макроса	endm
EXITM	Выход из макроса	exitm
EXPAND	Полный текст макроса в файле листинга	expand
LOCAL	Определение локальной переменной в макросе	local leng, tmp
MACRO	Определение макроса	out sym macro temp
NOEXPAND	Не разворачивать макрос	noexpand

Таблица 1.18

Арифметические операторы MASM

Оператор	Описание	Пример
1	2	3
\$	Текущий счетчик программы	goto \$ + 3
(левая скобка	1 + (d * 4)
)	правая скобка	(length + 1) * 255
!	операция «НЕ» (логическая инверсия)	if !(a - b)
~	инверсия	flags = ~ flags
-	отрицательное число (вторая инверсия)	-1 * length
high	выделить старший байт слова	movlw high llasid
low	выделить младший байт слова	movlw low (llasid + 2551)
*	умножение	a = c * b
/	деление	a = b / c
%	модуль	length = total % 16
+	сложение	tot len = length * 8 + 1
-	вычитание	Entry_Son = (Tot - 1) / 8
<<	сдвиг влево	val = flags << 1
>>	сдвиг вправо	val = flags >> 1
>=	больше либо равно	if ent >= num
>	больше	if ent > num
<	меньше	if ent < num
<=	меньше либо равно	if ent <= num
==	равно	if ent = num
!=	не равно	if ent != num
&	поразрядное «И»	flags = flags & err_bit
^	поразрядное «ИСКЛЮЧАЮЩЕЕ ИЛИ»	flags = flags ^ err_bit
	поразрядное «ВКЛЮЧАЮЩЕЕ ИЛИ»	flags = flags err_bit
&&	логическое «И»	if (len = 512) && (b = c)
	логическое «ИЛИ»	if (len = 512) (b = c)
=	установить равному...	entry_index = 0
+=	сложить и установить равному...	entry_index += 1
-=	вычесть и установить равному...	entry_index -= 1
*=	умножить и установить равному...	entry_index *= length
/=	делить и установить равному...	entry_index /= length
%=	модуль и установить равному...	entry_index %= 8
<<=	сдвиг влево и установить равному...	entry_index << 3
>>=	сдвиг вправо и установить равному...	entry_index >> 4
&=	«И» и установить равному...	entry_index &= err_flags
=	«ВКЛЮЧАЮЩЕЕ ИЛИ» и установить равному...	entry_index = err_flags

1	2	3
$\wedge=$	«ИСКЛЮЧАЮЩЕЕ ИЛИ» и установить равному...	entry_index $\wedge=$ err_flags
++	увеличить на 1 (инкремент)	i ++
--	уменьшить на 1 (декремент)	i --

1.3. Создание проекта в среде разработки IDE MPLab

Для создания проекта выполняются следующие шаги.

1) **Выбор папки проекта.** Создайте на диске папку проекта.

Примечание! MPLab не поддерживает длинные и русские имена папок, то есть имя должно состоять из латинских букв. То же самое относится и к названию проекта, и к именам исходных файлов.

2) **Выбор языка инструментальной среды.** Выберите *Project > Install Language Tool* из меню. Выберите *Language Suite: Microchip* и *Tool Name: MPASM*. Если не указан путь к файлу ассемблера mprasmwin.exe, укажите. Установите опцию *Windowed* и нажмите *OK*.

3) **Создание файла проекта.** В MPLAB выберите *Project > New Project*, укажите путь к Вашей папке и введите имя файла проекта, желательно такое же, как и имя основного исходного файла. Возможно указание нескольких папок через точку с запятой, например: *c:\mplab\projects\mpproj\include;c:\include\h;..\sys*. Нажмите *OK*.

4) **Установка конфигурации.** После перечисленных действий откроется окно редактирования проекта. Нажмите кнопку *Change* в пункте *Development Mode*. Откроется окно с вкладками для установки. В дальнейшем это окно можно будет вызвать, выбрав *Options > Development Mode*. Остановимся подробнее:

– *Tools*. Выберите инструментальную среду. Будем использовать симулятор *MPLAB-SIM*, установите признак и выберите тип микроконтроллера.

– *Clock*. Выберите тип генератора тактовых импульсов (кварцевый резонатор, внутренний или внешний RC-генератор, внешняя частота и т. д.) и его частоту.

– *Configuration*. Конфигурация сторожевого таймера и памяти программ. Если Вы разрешаете использование сторожевого таймера, то задайте и коэффициент деления предделителя. А для микроконтроллеров 17-й серии необходимо указать, какая память про-

грамм будет использоваться: внутренняя (*microcontroller*), внешняя (*microprocessor*) или обе (*extended microcontroller*).

– *Pins*. Разрешите или запретите использование вывода внешнего сброса (*MCLR*).

– *Break option*. Конфигурация прерываний и стека при отладке (лучше пока оставить по умолчанию). Нажимаете *OK*.

5) **Конфигурация выходного файла.** Щелкните, чтобы подсветить на файле *<my project>[.hex]*. При этом станет доступно меню *Node Properties*.

6) **Добавление исходного файла.** Чтобы добавить свой исходный файл на ассемблере, нажмите *Add Node*.

Далее нажмите *OK* в окне редактирования проекта. В дальнейшем это окно всегда можно будет вызвать через *Project>Edit Project*.

7) **Открытие исходного файла.** Для редактирования откройте свой исходный файл (*File > Open*) или создайте новый (*File > New*).

В проекте может быть несколько исходных файлов, при редактировании проекта указывать необходимо один, остальные включаются в основной исходный файл директивой ассемблера *include*.

8) **Создание программы.** Напишите исходный текст вашей программы, используя команды выбранного микроконтроллера и директивы пакета MPASM. Программа должна начинаться с директивы *List* с названием выбранного устройства и заканчиваться *end*.

9) **Компиляция программы** (построение проекта). Откомпилируйте свою программу, выбрав команду *Project > Build All*. После завершения процесса будет вызвано окно с генерированной командной строкой, перечнем предупреждений или ошибок (если есть) и результатом компиляции: были ошибки (*build failed*) или нет (*build successful*). Помощь при исправлении ошибок может показать файл-листинг (путь вызова: *Window > Absolute Listing*).

В результате выполнения данного этапа в папке с проектом создается ряд дополнительных файлов (*lst, hex*).

10) **Выполнение программы в пошаговом режиме.** Проконтролируйте правильность выполнения программ, для этого:

– вызовите окно просмотра *Window > Watch Windows* и внесите в него мнемоническое название регистров, используемых в проекте;

– вызовите окно регистров общего назначения *Window > File Registers*;

– вызовите окно памяти программ *Window > Program Memory*;

– проведите сброс вашей программы, выбрав команду *Debug > Reset*. В результате этого подсветится первая команда вашей программы, адрес которой 0000;

– выполните программу в пошаговом режиме, для чего выберите команду *Debug > Run > Step (F7)* или *Debug > Run > Step Over (F8)*. Каждый выбор команды меню будет выполнять одну команду вашей программы. Результат выполнения вашей программы контролируется в соответствующих вызванных панелях: *Watch Windows, File Registers, Program Memory*.

1.4. Индивидуальные задания

1.4.1 Согласно варианту (табл. 1.19) нарисовать подробный алгоритм и написать программу по нахождению значения функции для соответствующего микроконтроллера семейства Microchip.

Таблица 1.19

Варианты индивидуальных заданий

№ варианта	Функция	Микроконтроллер
1	2	3
1	$F = A \oplus \overline{B} + C \wedge B$	PIC12C508
2	$F = A \oplus \overline{B} - \overline{C} \vee A$	PIC16F84
3	$F = A + 0x2F \oplus \overline{B \wedge C}$	PIC12CE673
4	$F = \overline{A \wedge C} \oplus \overline{B} - C$	PIC16F84
5	$F = A - B \vee \overline{C \wedge 0xA7}$	PIC12C508
6	$F = A \vee \overline{B} + C \oplus A$	PIC12CE673
7	$F = A \vee 0xE9 + \overline{C} \oplus B$	PIC16F84
8	$F = A - \overline{B} \oplus C \wedge \overline{B}$	PIC12CE673
9	$F = \overline{A \wedge \overline{B} - \overline{C}} \vee A$	PIC12C508
10	$F = A - 0xDA \vee \overline{B \wedge C}$	PIC16F84
11	$F = A \oplus \overline{C} \wedge \overline{B} + C$	PIC12CE673
12	$F = A - \overline{B} \vee \overline{C} \wedge 0x6D$	PIC12C508
13	$F = \overline{A \vee \overline{B} + \overline{C}} \oplus A$	PIC16F84

1	2	3
14	$F = A - 0x5E \oplus C \vee B$	PIC12CE673
15	$F = A + B \vee C \wedge 0xA$	PIC16F84

1.4.2 Создать проект (см. подраздел 1.3) и написать программу по нахождению значения функции для соответствующего микроконтроллера, используя MPLab.

1.4.3 Проконтролировать правильность выполнения программы в пошаговом режиме с фиксацией результатов в соответствующих панелях среды разработки MPLab.

1.5. Порядок выполнения работы

Порядок выполнения задания рассмотрим на примере.

Задание. Найти значение функции $F = A + \overline{B \wedge C}$, где A, B, C – однобайтные числа. Результат сохранить в область регистров общего назначения по адресу 0×30 . Операнды: $A = 54, B = 75, C = 2A$.

Решение. Составляем алгоритм решения задачи. Параллельно определяем параметры настройки периферийных модулей и расположение переменных в области регистров общего назначения, необходимых для решения задачи. Алгоритм программы представлен на рис. 1.5.

Создаем проект (см. подраздел 1.3) и сохраняем его в рабочую папку своей программы. В проекте открываем созданный ранее файл и вносим туда текст программы. Программа, построенная в соответствии с алгоритмом (рис. 1.5), приведена в табл. 1.20.

Таблица 1.20

Исходный код программы

Операция	Описание
1	2
List p=12f675	Задаем компилятору MPASM тип микроконтроллера, с которым будем работать

1	2
Include <p12f675.inc>	Подключаем файл, в котором содержится информация о регистрах FSR, битах конфигурации и их мнемокодах
<u>CONFIG CP OFF & CPD OFF & IntOSC OSC</u>	
Задаем параметры периферийным модулям: отключаем защиту памяти программ и данных, подключаем встроенный RC-генератор, остальное по умолчанию	
PA equ 0x20	Определяем регистры области POH для хранения переменных и результата
PB equ 0x21	
PC equ 0x22	
REZ equ 0x30	
ORG 000	Директива MPASM определяющая месторасположения следующей команды в области памяти программ
start	Метка начала программы
movlw 0x54	Загрузка значения переменной «А» в соответствующий регистр области POH через аккумулятор
movwf PA	
movlw 0x75	Загрузка значения переменной «В» в соответствующий регистр области POH через аккумулятор
movwf PB	
movlw 0x2A	Загрузка значения переменной «С» в соответствующий регистр области POH через аккумулятор
movwf PC	
movf PB,w	Загрузка в аккумулятор переменной «В»
andwf PC,w	Операция «И» между содержимым аккумулятора и POH хранения переменной «С». Результат на место аккумулятора
movwf PEZ	Пересылка содержимого аккумулятора в регистр общего назначения REZ
comf REZ,w	Инверсия содержимого регистра REZ. Результат операции записывается в аккумулятор
addwf PA,w	Операция сложения содержимого аккумулятора и значения переменной «А». Результат операции на место аккумулятора
movwf PEZ	Пересылка содержимого аккумулятора в POH, на место конечного результата
goto start	Возврат на повтор выполнения программы
end	Директива MPASM, определяющая завершение программы

Компилируем программу и проверяем ее работоспособность при выполнении программы в пошаговом режиме (п. 10 раздела 1.3).



Рис. 1.5. Алгоритм программы нахождения значения функции

1.6. Требования к содержанию отчета

1.6.1 Подробный алгоритм по нахождению значения функции в соответствии с индивидуальным заданием.

1.6.2 Распечатанный файл *.LST.

1.6.3 Результат выполнения программы нахождения функции.

1.6.4 Ответы на контрольные вопросы.

1.7. Контрольные вопросы

1. Гарвардская архитектура микроконтроллера.
2. RISC архитектура микроконтроллера.
3. Организация памяти микроконтроллера семейства Microchip.
4. Регистр специального назначения.
5. Регистр STATUS и его назначение.
6. Биты конфигурации. Назначение.
7. Организация счетчика команд.
8. Состав пакета MPLab. Организация работы.

Лабораторная работа № 2

СОСТАВЛЕНИЕ ПРОСТЕЙШИХ ПРОГРАММ ВВОДА/ВЫВОДА С ИСПОЛЬЗОВАНИЕМ МИКРОКОНТРОЛЛЕРОВ СЕМЕЙСТВА MICROCHIP

Цель работы. Изучить периферийные модули микроконтроллеров семейства Microchip, их характеристики, способы управления. Организовать программу ввода/вывода с использованием временных аппаратных и программных задержек.

2.1. Организация периферийных модулей микроконтроллеров семейства Microchip. Краткие теоретические сведения

2.1.1 Порт ввода/вывода GPIO (на примере PIC12F675). В PIC12F675 реализовано 6 каналов порта ввода/вывода. Все каналы портов мультиплексированы и имеют дополнительными функции периферийных модулей. Когда используется периферийная функция, вывод не может использоваться как канал порта ввода/вывода.

Регистр GPIO представляет собой 6-разрядный порт ввода/вывода. Все каналы GPIO имеют соответствующие биты направления в регистре TRISIO, позволяющие настраивать канал как вход или выход. Запись «1» в TRISIO переводит соответствующий выходной буфер в 3-е состояние. Запись «0» в регистр TRISIO определяет соответствующий канал как выход, содержимое защелки GPIO передается на вывод микроконтроллера. Вывод GP3 может работать только на вход, чтение соответствующего бита TRISIO дает результат «1». Биты регистра TRISIO управляют направлением каналов GPIO, даже когда они используются как аналоговые входы.

Чтение регистра GPIO возвращает состояние на выводах порта, а запись производится в защелку GPIO. Все операции записи в порт выполняются по принципу «чтение–модификация–запись».

Каждый вывод GPIO в PIC12F675 имеет индивидуальный бит разрешения прерываний по изменению уровня сигнала на входах и бит включения внутреннего подтягивающего резистора (кроме GP3).

Индивидуальные биты разрешения прерываний по изменению уровня сигнала на входе располагаются в регистре IOCB. Прерывания по изменению уровня сигнала на входе запрещены после сброса по включению питания POR.

Работа прерывания по изменению уровня сигнала на входе заключается в следующем: сигнал на выводах GPIO сравнивается со значением, сохраненным при последнем чтении GPIO. В случае несовпадения одного из значений, устанавливается флаг и, если разрешено, генерируется прерывание.

Вывод GP0 может работать в одном из следующих режимов:

- канал порта ввода/вывода;
- аналоговый вход АЦП;
- аналоговый вход компаратора.

2.1.2 Организация прерывания (табл. 2.1–2.3). Микроконтроллер PIC12F675 имеет 7 источников прерываний:

- внешнее прерывание GP2/INT;
- переполнение TMR0;
- изменение уровня сигнала на входах GPIO;
- прерывание от компаратора;
- прерывание от модуля АЦП (только в PIC12F675);
- переполнение TMR1;
- завершение цикла записи в EEPROM память данных.

Для организации прерывания в PIC12F675 используется три регистра специального назначения: INTCON, P1E1 и PIR1.

Регистр INTCON. Регистр INTCON (табл. 2.1–2.3) доступен для чтения и записи, содержит биты разрешений и флаги прерываний: переполнение TMR0, изменения уровня сигнала на выводах GPIO, внешний источник прерываний GP2/INT.

Таблица 2.1

Биты регистра INTCON PIC12F675

Бит	Имя	Описание
1	2	3
Бит 7	GIE	Глобальное разрешение прерываний: 1 – разрешены все прерывания; 0 – все прерывания запрещены
Бит 6	PEIE	Разрешение прерываний от периферийных модулей: 1 – прерывание после записи разрешено; 0 – прерывание после записи запрещено
Бит 5	TOIE	Разрешение прерывания по переполнению TMR0: 1 – прерывание разрешено; 0 – прерывание запрещено
Бит 4	INTE	Разрешение внешнего прерывания INT: 1 – прерывание разрешено; 0 – прерывание запрещено

Окончание табл. 2.1

1	2	3
Бит 3	GPIE	Разрешение прерывания по изменению сигнала на входах GPIO: 1 – прерывание разрешено; 0 – прерывание запрещено
Бит 2	TOIF	Флаг прерывания по переполнению TMR0: 1 – прерывание разрешено; 0 – прерывание запрещено
Бит 1	INTF	Флаг внешнего прерывания INT: 1 – произошло переполнение TMR0 (сбрасывается программно); 0 – переполнения TMR0 не было
Бит 0	GPIF	Флаг прерывания по изменению уровня сигнала на входах GP5–GP0: 1 – зафиксировано изменение уровня сигнала на одном из входов GP; 0 – не было изменения уровня сигнала ни на одном из входов RB7–RB4

Таблица 2.2

Биты регистра INTCON PIC12CE673

Бит	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
Имя	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

Таблица 2.3

Биты регистра INTCON PIC16F84A

Бит	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
Имя	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

Регистр P1E1. Регистр P1E1 содержит биты разрешения периферийных прерываний (табл. 2.4–2.5).

Таблица 2.4

Биты регистра P1E1 PIC12F675

Бит	Имя	Описание
1	2	3
Бит 7	EEIE	Разрешение прерывания по завершению цикла записи в EEPROM память данных
Бит 6	ADIE	Разрешение прерываний по завершению преобразования АЦП

1	2	3
Бит 5–4	–	Не используется
Бит 3	CMIE	Разрешение прерывания от модуля компаратора
Бит 2–1	–	Не используется: Читается как «0»
Бит 0	TMR1E	Разрешение прерывания по переполнению TMR1

Таблица 2.5

Биты регистра PIE1 PIC12CE673

Бит	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
Имя	–	ADIE	–	–	–	–	–	–

Регистр PIR1. Регистр PIR1 содержит флаги прерываний периферийных модулей (табл. 2.6–2.7).

Таблица 2.6

Биты регистра PIR1 PIC12F675

1	Имя	Описание
2	3	
Бит 7	EEIF	Флаг прерывания по окончанию записи в EEPROM память данных: 1 – запись в EEPROM память данных завершена (сбрасывается программно); 0 – запись в EEPROM память данных не завершена или не была начата
Бит 6	ADIF	Флаг прерывания от модуля АЦП: 1 – преобразование АЦП завершено (сбрасывается программно); 0 – преобразование АЦП не завершено
Бит 5–4	–	Не используется
Бит 3	CMIF	Разрешение прерывания от модуля компаратора: 1 – изменился уровень сигнала на входе компаратора (сбрасывается программно); 0 – уровень сигнала на входе компаратора не изменялся
Бит 2–1	–	Не используется
Бит 0	TMR1F	Разрешение прерывания по переполнению TMR1

Биты регистра PIR1 PIC12CE673

Бит	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
Имя	–	ADIE	–	–	–	–	–	–

При переходе на подпрограмму обработки прерываний бит GIE аппаратно сбрасывается в «0», запрещая прерывания, адрес возврата из подпрограммы обработки прерываний помещается в стек, а в счетчик команд PC загружается вектор прерывания 0004h. Источник прерываний может быть определен проверкой флагов прерываний, которые должны быть сброшены программно перед разрешением прерываний, чтобы избежать повторного вызова.

Для внешних источников прерываний (сигнал INT, изменения уровня сигнала на входах GPIO) время перехода на подпрограмму обработки прерываний будет составлять 3–4 машинных цикла. Точное время перехода зависит от конкретного случая, оно одинаково для 1- и 2-цикловых команд. Внешнее прерывание с входа GP2/INT может происходить как по переднему фронту сигнала, так и по заднему (бит INTEDG). Когда активный фронт сигнала появляется на входе GP2/INT, бит INTF (INTCON) устанавливается в «1». Прерывание может быть запрещено сбросом бита INTE (INTCON) в «0». Флаг прерывания INTF сбрасывается. Прерывание INT может вывести микроконтроллер из режима SLEEP.

2.1.3 Модуль таймера TMR0. Таймер/счетчик TMR0 имеет следующие особенности:

- 8-разрядный таймер/счетчик;
- возможность чтения и записи текущего значения счетчика;
- 8-разрядный программируемый предделитель;
- внутренний или внешний источник тактового сигнала;
- выбор активного фронта внешнего тактового сигнала;
- прерывания при переполнении (переход от FFh к 00h).

Режим таймера. TMR0 работает от внутреннего тактового сигнала. Приращение TMR0 происходит в каждом машинном цикле (если предделитель отключен). Работа таймера может вызывать прерывания. Прерывания от TMR0 возникают при переполнении счетчика, т. е. при переходе его значения от FFh к 00h. При возник-

новении прерывания устанавливается флаг прерывания. По завершении обработки прерывания данный флаг сбрасывается программно. В SLEEP режиме микроконтроллера модуль TMR0 выключен и не может генерировать прерывания.

Режим счетчика. TMR0 работает от внешнего источника тактового сигнала с входа GP2/T0CKI. Активный фронт внешнего тактового сигнала выбирается битом T0SE в регистре OPTION_REG. Схема модуля содержит в своем составе 8-разрядный счетчик, который может работать как предделитель. Этот предделитель может быть включен как перед TMR0, так и перед WDT в зависимости от состояния бита PSA (OPTION_REG). Коэффициент деления предделителя определяется битами PSA и PS2–PS0 в регистре OPTION_REG. Если предделитель включен перед TMR0, любые команды записи в TMR0 сбрасывают предделитель. Предделитель также очищается при сбросе микроконтроллера. Данный счетчик недоступен для чтения и записи.

Дополнительно к таймеру TMR0 в рассматриваемом микроконтроллере присутствует модуль таймера TMR1, который имеет следующие особенности:

- 16-разрядный таймер/счетчик (с двумя 8-разрядными регистрами TMR1H, TMR1L);
- значение таймера доступно для записи и чтения (оба регистра);
- выбор источника тактового сигнала (внешний/внутренний);
- синхронный и асинхронный режим работы;
- генерация прерываний по переполнению от FFFFh к 0000h;
- выход из режима SLEEP при переполнении;
- вход внешнего включения таймера -T1G (опционально);
- LP генератор (опционально).

2.1.4 Модуль компаратора. PIC12F675 содержит один аналоговый компаратор, входы которого мультиплексированы с каналами ввода/вывода GP0 и GP1. Выход интегрированного источника опорного напряжения может быть подключен на вход компараторов. Вывод GP2 может быть настроен как цифровой выход компаратора. Модулем компаратора управляет регистр CMCON (табл. 2.8).

Работа компаратора выглядит следующим образом – когда аналоговый сигнал на входе V_{IN+} меньше V_{IN-} , на цифровом выходе установлен логический нуль. Если сигнал на входе V_{IN+} больше V_{IN-} , то на цифровом выходе будет установлена логическая единица.

Биты регистра CMCON

Бит	Имя	Описание
1	2	3
Бит7	–	Не используется
Бит6	CMOUT	Выход компаратора: Если CINV = 0 $1 = V_{IN+} > V_{IN-}$; $0 = V_{IN+} < V_{IN-}$. Если CINV = 1 $0 = V_{IN+} > V_{IN-}$; $1 = V_{IN+} < V_{IN-}$.
Бит5	–	Не используется
Бит4	CINV	Инверсный выход компаратора: 1 – инверсный выход; 0 – не инверсный выход
Бит3	CIS	Подключение входов компараторов: <i>Когда CM2–CM0 = 110 или 101</i> 1 – V_{IN-} подключен к CIN+; 0 – V_{IN-} подключен к CIN–
Бит 2–0	CM2–CM 0	Режим работы компаратора

Существует восемь режимов работы модуля компараторов, устанавливаемые битами CM2–CM0. Биты регистра TRISIO управляют направлением каналов ввода/вывода для каждого режима модуля компараторов. Состояние выхода компаратора можно прочитать в регистре CMCON (бит выхода компаратора доступен только для чтение). Вывод компаратора может быть подключен к каналу порта ввода/вывода GP2 выбором соответствующего режима работы.

Модуль компаратора позволяет использовать внутренний источник опорного напряжения, подключаемый к одному из входов компаратора. Биты управления опорным напряжением размещены в регистре VRCON (табл. 2.9). Источник опорного напряжения имеет 32 различных уровня напряжения (по 16 в каждом диапазоне).

Модуль компаратора может организовывать прерывание при изменении уровня сигнала на выходе компаратора. Флаг прерывания от компараторов CMIF сбрасывается программно. Программной установкой бита CMIF в «1» моделируется возникновение прерывания от модуля компараторов.

Таблица 2.9

Биты регистра VRCON

Бит	Имя	Описание
1	2	3
Бит 7	VREN	Включение источника опорного напряжения
Бит 6	–	Не используется
Бит 5	VRR	Диапазон выходного напряжения V_{REF} : 1 – нижний диапазон; 0 – верхний диапазон
Бит 4	–	Не используется
Бит 3–0	VR3–VR0	Выбор выходного напряжения CV_{REF} : Если VRR = 1, то $V_{REF} = (V_R \langle 3:0 \rangle / 24) V_{DD}$ Если VRR = 0, то $V_{REF} = (V_{RE} / 4) + (V_R \langle 3:0 \rangle / 32) V_{DD}$

2.1.5 Модуль АЦП. Модуль АЦП преобразует входной аналоговый сигнал в соответствующий 10-разрядный цифровой код. В PIC12F675 четыре аналоговых канала, мультиплексируемые на одну схему выборки и хранения.

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 опорного напряжения может быть программно выбран с вывода V_{DD} или V_{REF} .

Для управления модулем АЦП предусмотрено два регистра:

- ADCON0 (таблица 2.10–2.11);
- ANSEL (таблица 2.12).

Таблица 2.10

Биты регистра ADCON0 PIC12F675

Бит	Имя	Описание
1	2	3
Бит 7	ADFM	Формат сохранения 10-разрядного результата (выравнивание)
Бит 6	VCFG	Выбор источника опорного напряжения: 1 – вывод V_{REF} ; 0 – напряжение питания V_{DD}

1	2	3
Бит 5–4	–	Не используется
Бит 3–2	CHS1–0	Выбор аналогового канала
Бит 1	GO/-DONE	Бит начала работы модуля АЦП
Бит 0	ADON	Бит включения модуля АЦП

Таблица 2.11

Биты регистра ADCON0 PIC12CE673

Бит	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
Имя	ADCS1	ADCS0	–	CHS1	CHS0	GO/DONE	–	ADON
ADCS1, ADCS0 – Выбор частоты преобразования АЦП; CHS1, CHS0 – Выбор канала преобразования АЦП; GO/DONE – Бит статуса преобразования АЦП; ADON – Бит запуска преобразования АЦП								

Таблица 2.12

Биты регистра ANSEL

Бит	Имя	Описание
1	2	3
Бит 7	–	Не используется
Бит 6–4	ADCS2–0	Выбор источника тактового сигнала: $000 = F_{OSC} / 2$; $001 = F_{OSC} / 8$; $010 = F_{OSC} / 32$; $x11 = F_{RC}$ (отдельный внутренний RC генератор. Максимальная частота 500 кГц); $100 = F_{OSC} / 4$; $101 = F_{OSC} / 16$; $110 = F_{OSC} / 64$
Бит 3–0	ANS3–0	Настройка вывода как аналоговый вход: 0 – цифровой канал порта ввода/вывода или специальные функции; 1 – аналоговый вход

Источник опорного напряжения зависит от состояния бита VCFG (ADCON). Время получения одного бита результата определяется параметром T_{AD} . Для 10-разрядного результата требуется как минимум 11 T_{AD} . Параметры тактового сигнала для АЦП определяются программно (ANSEL). Для получения корректного результата пре-

образования необходимо выбрать источник тактового сигнала АЦП, обеспечивающий время T_{AD} не менее 1 микросекунды.

Для инициализации преобразования необходимо установить в «1» бит GO/-DONE (ADCON0). Когда преобразование завершено, выполняются следующие действия:

- сбрасывается в «0» бит GO/-DONE;
- устанавливается в «1» флаг ADIF (PIR<6>);
- выполняется переход на обработку прерываний.

10-разрядный результат преобразования сохраняется в спаренном 16-разрядном регистре ADRESH:ADRESL. Запись результата преобразования может выполняться с правым или левым выравниванием.

Модуль АЦП может работать в SLEEP режиме микроконтроллера при условии, что источником импульсов преобразования АЦП будет внутренний RC генератор. После завершения преобразования аппаратно сбрасывается бит GO/-DONE в «0», результат преобразования сохраняется в регистрах ADRESH:ADRESL. Если разрешено прерывание от АЦП, то микроконтроллер выйдет из режима SLEEP.

Если был выбран другой источник тактовых импульсов АЦП (не внутренний RC генератор), то выполнение программой инструкции SLEEP прервет процесс преобразования и выключит модуль АЦП.

2.1.6 Память данных EEPROM. EEPROM энергонезависимая память данных, которая доступна для записи/чтения в нормальном режиме работы микроконтроллера. EEPROM память данных не отображается на адресное пространство памяти данных, а доступна через регистры специального назначения.

Для организации доступа к EEPROM памяти данных используются 4 регистра специального назначения (таблица 2.13–2.14): EECON1, EECON2, EEDATA, EEDADR.

Таблица 2.13

Биты регистра EECON1 PIC12F675

Бит	Имя	Назначение
1	2	3
Бит 7–4	–	Не используется
Бит 3	WRERR	Флаг ошибки записи в EEPROM память данных
Бит 2	WREN	Разрешение записи в EEPROM память данных

1	2	3
Бит 1	WR	Инициализировать запись в EEPROM память данных
Бит 0	RD	Инициализировать чтение из EEPROM памяти данных

Таблица 2.14

Регистр EECON1 PIC16F84A

Бит	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
Имя	Не используются			EEIF	WRERR	WREN	WR	RD

Чтение и запись EEPROM памяти выполняется побайтно. В регистре EEDATA сохраняются 8-разрядные данные записи/чтения, а регистр EEADR содержит адрес ячейки EEPROM памяти данных. PIC12F675 содержат 128 байт EEPROM памяти данных (диапазон адресов 00h–7Fh). При установке защиты на доступ к EEPROM памяти данных, программа микроконтроллера имеет возможность выполнить запись/чтение EEPROM памяти данных. После завершения записи в EEPROM память данных возможно генерирование прерывания.

Регистр EECON2 не реализован физически, читается как 00h. Он используется в операциях записи в EEPROM память данных для реализации обязательной последовательности команд.

Чтение из EEPROM памяти данных. Для чтения EEPROM памяти данных необходимо записать адрес в регистр EEADR и установить бит RD (EECON1) в 1. В следующем машинном цикле данные доступны для чтения из регистра EEDATA. Прочитанное значение из EEPROM памяти данных будет храниться в регистре EEDATA до следующего чтения или записи в этот регистр.

Запись в EEPROM память данных. Для записи в EEPROM память данных необходимо записать адрес в регистр EEADR, данные в регистр EEDATA и установить бит WR и WREN (EECON1) в 1. Рекомендуется запрещать прерывания при выполнении записи в EEPROM память.

Чтобы разрешить запись в EEPROM память данных, необходимо перед началом записи установить бит WREN в «1», защищающий от случайной записи.

2.1.7 Конфигурация портов (табл. 2.15).

Таблица 2.15

Конфигурация портов

Адрес бита	Конфигурация				
1	2	3	4	5	6
PCFG	GP4	GP2	GP1	GP0	Vref
000	A	A	A	A	Vdd
001	A	A	Vref	A	GP1
010	D	A	A	A	Vdd
011	D	A	Vref	A	GP1
100	D	D	A	A	Vdd
101	D	D	Vref	A	GP1
110	D	D	D	A	Vdd
111	D	D	D	D	Vdd

2.2. Программная реализация временной задержки

Организовать временные задержки можно следующими способами:

- 1) аппаратный – временные задержки организуются с помощью следящих встроенных аппаратных модулей-таймеров (счетчиков);
- 2) программный – временные задержки организуются с помощью программных модулей-циклов, т. е. зная время выполнения одной команды, можно организовывать определенные временные задержки.

Простейшей временной программной задержкой является использование команды NOP. Зная, что данная команда выполняется за один машинный цикл, и зная время его выполнения, можно организовать определенную задержку из ряда этих команд. При больших значениях параметров (> 10 микросекунд) такой метод не является оптимальным, т. е. получается сильно увеличенный программный код. В этом случае прибегают к программным модулям (табл. 2.16).

Длительность цикла вычисляется по формуле

$$t = M \cdot [(x_{1_2} + 1 + 2) \cdot x_{1_1} + 1 + 2 + x_0 + 2], \quad (1)$$

где $M = 0,25 \cdot f_{\text{сист}}$ – время выполнения цикла команды;

$f_{\text{сист}}$ – тактовая частота ядра.

Таблица 2.16

Программный модуль временной задержки

Команда	Количество циклов	Комментарий
1	2	3
Count 1	–	Метка подпрограммы
movlw x1_1 movwf C1	1 1	Загрузка в регистр C1 константы через аккумулятор. Задает количество циклов
ll nop ... } x1_2	1	Команды пустой операции. Увеличивает длительность цикла
decfsz C1,1	1(2)	Вычитание содержимого регистра C1 и пропуск следующей команды, если результат равен нулю
goto ll	2	Переход на метку
nop ... } x0	1	Дополнительная задержка с помощью пустых операций
return	2	Возврат из подпрограммы

При значении временной задержки больше 2 миллисекунд организуют программный модуль-цикл в цикле (табл. 2.17).

Длительность подпрограммы вычисляется по формуле

$$t = M \cdot \{[(x_{2_2} + 1 + 2) \cdot x_{2_1} + 2 + x_{2_3} + 1 + 2 + 2] \times x_{1_1} + 2 + x_3 + 2 + 2\}. \quad (2)$$

Таблица 2.17

Программный модуль реализации задержки

Команда	Количество циклов	Комментарий
1	2	3
Count 2	–	Метка подпрограммы
movlw x1_1 movwf C1	1 1	Задание параметров внешнего цикла
ll movlw x2_1 movwf C2	1 1	Задание параметров внутреннего цикла

1	2	3
l2 nop ... } x2_2 Nop decfsz C2,1 goto l2	1 ... 1 1(2) 2	Выполнение внутреннего цикла. Время выполнения: $T = M \cdot \{(1 \cdot x_{2_2} + 1 + 2) \cdot x_{2_1}\}$
nop ... } x2_3 nop	1 ... 1	Дополнительная задержка внутреннего цикла
decfsz c1,1 goto l1	1(2) 2	Организация внешнего цикла
nop ... } x3 nop	1 ... 1	Дополнительная временная задержка
return	2	Возврат из подпрограммы

2.3. Индивидуальные задания

2.3.1 Создать проект MPLab (см. подраздел 1.3).

2.3.2 Согласно заданию и варианту (табл. 2.18) построить схему и программный алгоритм работы процессорного ядра.

Таблица 2.18

Варианты задания

№	Микроконтроллер	F, Гц	q	OUT	IN	Дополнительные условия
1	2	3	4	5	6	7
1	P12C508	2000	5	GP4	GP1 (опрос)	Встроенный RC генератор. Организован внешний сброс
2	P16F84	1	2	PB2	PA2 (опрос)	Внешний HS-генератор 4 МГц
3	P12CE673	0,5	10	GP1	GP3 (прерывание)	Внешний RC генератор
4	P16F84	10	5	PA2	INT (прерывание)	Встроенный RC генератор
5	P12C508	500	10	GP0	GP2 (прерывание)	Внешний XT-генератор 4 МГц

1	2	3	4	5	6	7
6	P12CE673	5	5	GP1	INT (прерывание)	Встроенный RC генератор. Организован внешний сброс
7	P16F84	200	2	PB2	PB7 (прерывание)	Встроенный RC генератор. Организован внешний сброс
8	P12CE673	1	5	GP5	GP2 (опрос)	Встроенный RC генератор
9	P12C508	50	10	GP5	GP2 (опрос)	Встроенный RC генератор Организован внешний сброс
10	P16F84	5000	5	PA5	PB5 (опрос)	Внешний HS-генератор 4 МГц. Организован внешний сброс
11	P12CE673	100	3	GP0	GP4 (прерывание)	Встроенный RC генератор. Организован внешний сброс
12	P12C508	0,5	10	GP2	GP0 (опрос)	Встроенный RC генератор. Организован внешний сброс
13	P16F84	500	5	PA1	PB4 (опрос)	Встроенный RC генератор
14	P12CE673	0,1	20	GP4	INT (прерывание)	Встроенный RC генератор. Организован внешний сброс

Задание. Организовать систему, формирующую сигнал с частотой F и коэффициентом заполнения q на выходе «OUT» в течение 5 секунд при формировании сигнала на входе «IN». Управление может производиться как по опросу, так и по прерыванию.

2.3.3 Проконтролировать правильность выполнения программы в пошаговом режиме с фиксацией результатов в соответствующих панелях среды разработки MPLab.

2.3.4 Произвести оценку длины программы в байтах. Определить время в тактах, необходимое на выполнение программы.

2.4. Порядок выполнения работы

Порядок выполнения лабораторной работы рассмотрим на примере.

Задание. Организовать систему, формирующую импульсный сигнал с частотой 1 кГц, коэффициент заполнения 2, в течении 5 секунд после нажатия внешней клавиши. Дополнительные параметры: микроконтроллер PIC12F675; вход – прерывание по изменению

уровня на входе GP0; выход – GP1, WDT – OFF; генератор – встроенный RC (4 МГц), организована цепь внешнего сброса.

Решение. Прежде чем приступить к реализации программной модели, необходимо определиться со схемотехнической частью системы. Для этого рассмотрим общую принципиальную схему устройства (рис. 2.1).

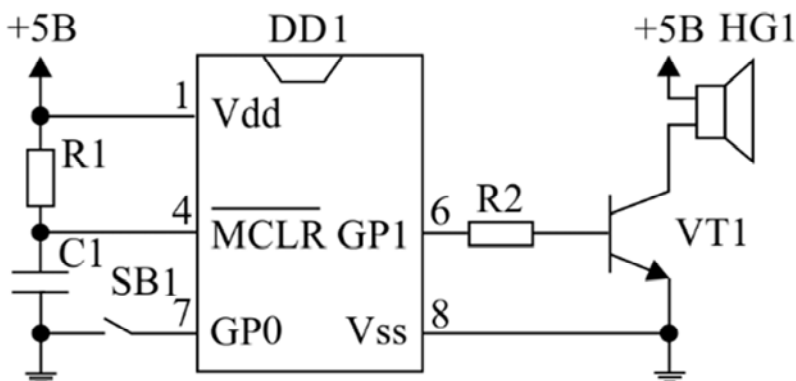


Рис. 2.1. Принципиальная схема устройства

Так как сигнал с частотой 1 кГц представляет собой звуковой сигнал, то для его отслеживания воспользуемся динамической головкой, включенной в коллекторную цепь транзисторного ключа.

Далее изобразим алгоритм работы устройства. Согласно условию, алгоритм работы устройства делится на 2 независимые части:

- режим ожидания внешнего воздействия (основная программа (рис. 2.2, а));
- режим выработки сигнала с заданными параметрами (режим обработки прерывания (рис. 2.2, б)).

Из представленного на рис. 2.2 алгоритма можно заметить, что большую часть программы составляет организация временных задержек. Их организацию реализуем программно.

Согласно выбранной частоте основного генератора (4 МГц) и зная цикл выполнения команды (4 такта), можно определить, что цикл команды равен 1 микросекунде ($n = 4 \cdot 1/4$).

Зная, что частота сигнала f равна 1 кГц, определим период T :

$$T = 1/f = 1 \text{ мс},$$

учитывая, что коэффициент заполнения $q = 2$ следует, что

$$\tau_{\text{сигн}} = \tau_{\text{зад}} = T / 2 = 0,5 \text{ мс} = 500 \text{ мкс}$$



Рис. 2.2. Алгоритм работы устройства:
 а) основная программа; б) обработка прерывания

Из уравнения (1) найдем коэффициенты x_{1_1} и x_{1_2} . В данном случае особая точность не является актуальной, и зная правила вычисления уравнений с двумя неизвестными, зададим $x_{1_2} = 0$. Следовательно x_{1_1} определяется по формуле:

$$t = 1 \cdot (3x_{1_1} + 4) = 500,$$

следовательно,

$$x_{1_1} = 496/3 = 165.$$

Далее следует задать время вывода самого сигнала на выходе системы. Для этого зададим количество выводов временных интервалов полупериодов для обеспечения 5 секунд, соответственно,

$$N = 5/0,0005 = 10\ 000.$$

Для организации такого числа циклов необходимо организовать вложение модулей, задав определенные коэффициенты (табл. 2.19).

Таблица 2.19

Программа организации числа циклов

Программа	Комментарий
1	2
Sign	
movlw n1 movwf C11	Задание выполнения внешнего цикла в количестве n_1
L3 movlw n2 movwf C22	Задание выполнения вложенного цикла в количестве n_2
L4	Подпрограмма формирования импульса
decfsz C22,1 goto L4	Организация вложенного цикла
decfsz C11,1 goto L3	Организация внешнего цикла

Общее время выполнения будет составлять

$$T = n_1 \cdot n_2 \cdot \tau_{\text{сигн}},$$

где $\tau_{\text{сигн}}$ – длительность выводимого импульса.

Учитывая, что n_1 и n_2 не могут быть больше 255, зададим $n_1 = 50$, а $n_2 = 200$.

Программа, соответствующая рассмотренным условиям, приведена в табл. 2.20.

Таблица 2.20

Программная реализация алгоритма работы устройства

Программа	Комментарии
1	2
List p=12f675	Объявление компилятору типа микроконтроллера
#include<p12f675.inc>	Подключает файл, содержащий информацию о регистрах, FSR и их битах, битах конфигурации, их описание
__CONFIG_CP_OFF & CPD_OFF & MCLR_ON & IntOSC_OSC	
Конфигурация определяет параметры программного обеспечения микроконтроллера, настраиваемого при программировании. В данном случае:	
– сторожевой таймер – выключен;	
– сброс на уменьшение напряжения питания – выключено (по умолчанию);	
– защита памяти программ и данных – выключено;	
– внешний сброс MCLR – включено;	
– таймер включения питания – включено (по умолчанию);	
– системный генератор – внутренняя RC-цепь	
C1 equ 0x20 C2 equ 0x21 C3 equ 0x22 C4 equ 0x23 exit equ 0x24	Вводит название для некоторых регистров общего назначения, для удобства обращения к ним
ORG 0000	Вектор сброса
goto main	Переход на основную программу main
ORG 004	Вектор прерывания (адрес начала обработки прерывания)
<i>Подпрограмма генерации сигнала</i>	
Sign	
movlw 50	Задание выполнения цикла в количестве 20000 (80x250), который определит время выполнения программы в 10 секунд
movwf l3	
L3	
movlw 200	
movwf l4	
L4	
call inv	Вызов программы инвертирования сигнала на выходе
call count	Вызов подпрограммы задержки для задания необходимой длительности импульсов
decfsz l4,1	Организация внутреннего цикла (n = 250)
goto l4	

1	2
decfsz 14,1	Организация внешнего цикла (n = 80)
goto 13	
bcf GPIO,1	Сброс выхода для уменьшения энергопотребления
bcf INTCON,0	Сброс сигнала прерывания от изменения уровня на GP
retfie	Выход из подпрограммы с установкой бита разрешения глобального прерывания
<i>Инициализация ядра и периферийных модулей</i>	
main	
bsf STATUS,RP0	Переход в банк № 1
clrf Ansel	Сбрасывает в регистре Ansel биты 0, 1 и 2 в «0» для работы портов микроконтроллера, как цифровые
movlw b»00100000» movwf TRISIO	Определяет работу портов как GP5 – вход, остальные выход. Загрузка производится через аккумулятор
clrf GPIO	Сбрасывает порт GPIO
movlw b»00100000» movwf IOCB movwf WPU	Включает подтягивающий резистор на вход и разрешает прерывание по входу GP1
bcf OPTION_REG,7	Установка общего включения подтягивающих резисторов
bcf STATUS,5	Переход в банк № 0
movlw B»10001000» movwf INTCON	Настраиваемое прерывание по изменению уровня GPIO и глобальное прерывание
clrf exit	Обнуление рабочего регистра
<i>Завершение инициализации. Модуль основной программы.</i>	
Start	Метка программы ожидания
sleep	Переход в режим энергосбережения
goto Start	
<i>Подпрограмма организации временной задержки в 500 мксек</i>	
Count 1	
movlw 165	
movwf 11	
L1 decfsz 11,1	
goto 11	
return .	
<i>Подпрограмма инвертирования сигнала на выходе</i>	
inv	
movlw b»00000010» xorwf exit 1	Инвертирование бита путем операции «исключающего ИЛИ» с маскирующей константой, где «1» содержится в разрядах, подлежащих изменению и загрузка полученного значения в регистр GPIO (на выход микроконтроллера)
movf exit,w	
movwf GPIO	
return	
end.	Директива MPASM завершения проекта

Компилируем программу и проверяем ее работоспособность.

2.5. Требования к содержанию отчета

2.5.1. Алгоритм программы по организации ввода/вывода.

2.5.2. Расчеты для нахождения временных коэффициентов.

2.5.3. Распечатанный файл *.LST, сгенерированный из проекта программой MPASM.

2.5.4. Ответы на контрольные вопросы.

2.6. Контрольные вопросы

1. Порты ввода/вывода в микроконтроллерах Microchip. Организация работы портов ввода/вывода.

2. Система прерываний в микроконтроллерах Microchip и организация их работы.

3. Таймеры в микроконтроллерах Microchip. Организация работы.

4. Аналоговый модуль в микроконтроллерах Microchip.

5. Регистры специального назначения.

Лабораторная работа № 3

ИЗУЧЕНИЕ СРЕДЫ РАЗРАБОТКИ KEIL μ VISION. СОЗДАНИЕ ПРОСТЕЙШИХ ПРОЕКТОВ ДЛЯ МИКРОКОНТРОЛЛЕРОВ СЕМЕЙСТВА MCS51

Цель работы. Изучить организацию 8-ми разрядного CISC микроконтроллера семейства MCS51. Ознакомиться с его системой и форматом команд, способами адресации. Изучить среду разработки Keil μ Vision. Составить и проверить выполнение программы для нахождения функции, используя возможности среды Keil μ Vision.

3.1. Микроконтроллеры семейства MCS51 (8051). ***Краткие теоретические сведения***

3.1.1 Общие характеристики. Микроконтроллеры семейства 8051 имеют следующие аппаратные особенности:

- внутреннее ОЗУ объемом 128 байт;
- четыре двунаправленных побитно настраиваемых восьмиразрядных порта ввода/вывода;
- два 16-разрядных таймера-счетчика;
- встроенный тактовый генератор;
- адресация 64 Кбайт памяти программ и 64 Кбайт памяти данных;
- две линии запросов на прерывание от внешних устройств;
- интерфейс для последовательного обмена информацией с другими микроконтроллерами или персональными компьютерами.

3.1.2 Арифметико-логическое устройство (ALU). 8-битное ALU может выполнять операции сложения, вычитания, умножения и деления; логические операции «И», «ИЛИ», «исключающее ИЛИ», операции циклического сдвига, сброса, и др. Ко входам подключены программно-недоступные регистры T1 и T2, предназначенные для временного хранения операндов, схема десятичной коррекции (DCU) и схема формирования признаков результата операции (PSW).

ALU может оперировать четырьмя типами информационных объектов: булевыми (1 бит), цифровыми (4 бита), байтными (8 бит) и адресными (16 бит). В ALU выполняется 51 операция пересылки или преобразования этих данных. Так как используется 11 режимов адресации (7 для данных и 4 для адресов), то путем комбинирования

операции и режима адресации базовое число команд 111 расширяется до 255 из 256 возможных при однобайтном коде операции.

3.1.3 Организация памяти. Как и у большинства микроконтроллеров, в MCS51 память программ и память данных являются самостоятельными и независимыми друг от друга устройствами, адресуемыми различными командами и управляющими сигналами.

Память программ (ПЗУ). Объем встроенной памяти программ, расположенной на кристалле микроконтроллера 8051 равен 4 Кбайт. При обращении к внешней памяти программ используют 16-разрядный адрес, что обеспечивает им доступ к 64 Кбайт ПЗУ. При выполнении команд переноса данных адресация ячейки памяти может осуществляться с использованием как счетчика команд PC, так и специального двухбайтового регистра-указателя данных DPTR.

Память данных (ОЗУ). Объем расположенной на кристалле памяти данных – 128 байт. Объем внешней памяти данных может достигать 64 Кбайт. Первые 32 байта организованы в четыре банка регистров общего назначения, обозначаемых соответственно банк 0 – банк 3. Каждый из них состоит из восьми регистров R0–R7. В любой момент программе доступен только один банк регистров, номер которого содержится в третьем и четвертом битах слова состояния программы PSW (см. ниже).

Оставшееся адресное пространство может конфигурироваться разработчиком по своему усмотрению: в нем располагаются стек, системные и пользовательские области данных.

Обращение к ячейкам памяти данных возможно двумя способами. Первый способ – прямая адресация ячейки памяти. В этом случае адрес ячейки является операндом соответствующей команды. Второй способ – косвенная адресация с помощью регистров R0 или R1: перед выполнением соответствующей команды в один из них должен быть занесен адрес ячейки, к которой необходимо обратиться.

Для обращения к внешней памяти данных используется только косвенная адресация с помощью регистров R0 и R1 или с помощью 16-разрядного регистра-указателя DPTR.

Регистры специальных функций. К адресному пространству памяти данных примыкает адресное пространство регистров специальных функций SFR (Special Function Register). Регистры SFR (табл. 3.1) управляют работой блоков, входящих в микроконтроллер.

Таблица 3.1

Размещение регистров специальных функций

Адрес	Символ	Наименование
1	2	3
0E0H	*ACC	Аккумулятор
0F0H	*B	Регистр расширитель аккумулятора
0D0H	*PSW	Слово состояния программы
080H	*P0	Порт 0 (SFR P0)
090H	*P1	Порт 1 (SFR P1)
0A0H	*P2	Порт 2 (SFR P2)
0B0H	*P3	Порт 3 (SFR P3)
081H	SP	Регистр указатель стека
083H	DPH	Старший байт регистра указателя данных DPTR
082H	DPL	Младший байт регистра указателя данных DPTR
08CH	TH0	Старший байт таймера «0»
08AH	TL0	Младший байт таймера «0»
08DH	TH1	Старший байт таймера «1»
08BH	TL1	Младший байт таймера «1»
089H	TMOD	Регистр режимов таймеров счетчиков
088H	*TCON	Регистр управления статуса таймеров
0B8H	*IP	Регистр приоритетов
0A8H	*IE	Регистр маски прерывания
087H	PCON	Регистр управления мощностью
098H	*SCON	Регистр управления приемопередатчиком
099H	SBUF	Буфер приемопередатчика
* Регистры, допускающие адресацию отдельных бит при использовании команд операций над битами		

Регистры SFR обеспечивают управление процессорным ядром:

- регистр-указатель стека SP;
- регистр-указатель данных DPTR – используют для фиксации 16-битного адреса в операциях обращения к внешней памяти программ и данных;
- аккумулятор (ACC) является источником операнда и местом фиксации результата при выполнении арифметических, логических операций и ряда операций передачи данных. В распоряжении пользователя имеются 8 регистров общего назначения R0–R7 одного из четырёх возможных банков;

– регистр В используется как источник и как приемник при операциях умножения и деления, обращение к нему, как к регистру SFR, производится аналогично аккумулятору;

– регистр флагов (PSW) (табл. 3.2).

Таблица 3.2

Перечень флагов PSW

Символ	Позиция	Имя и назначение			
1	2	3			
P	PSW.0	Флаг приоритета. Устанавливается и сбрасывается аппаратно в каждом цикле команды и фиксирует нечетное/четное число единичных бит в аккумуляторе			
–	PSW.1	Не используется			
OV	PSW.2	Флаг переполнения. Устанавливается и сбрасывается аппаратно при выполнении арифметических операций			
RS0– RS1	PSW.3– PSW.4	Биты выбора используемого банка регистров:			
		RS0	RS1	Банк	Границы адресов ОЗУ
		0	0	0	00H–07H
		1	0	1	08H–0FH
		0	1	2	10H–17H
		1	1	3	18H–1FH
F0	PSW.5	Флаг пользователя. Может быть установлен, сброшен или проверен программой пользователя			
AC	PSW.6	Флаг вспомогательного переноса. Устанавливается и сбрасывается только аппаратными средствами при выполнении команд сложения/вычитания и сигнализирует о переносе или заеме в бите 3 аккумулятора			
C	PSW.7	Флаг переноса. Устанавливается и сбрасывается как аппаратно, так и программным путем			

3.1.4 Устройство управления и синхронизации. Кварцевый резонатор управляет работой внутреннего генератора, который формирует сигналы синхронизации. Устройство управления (CU) на основе сигналов синхронизации формирует машинный цикл, равный 12 периодам резонатора. Ряд команд выполняется за один машинный цикл. Некоторые команды, оперирующие с 2-байтными словами или связанные с обращением к внешней памяти, выполняются за два цикла, а команды деления и умножения требуют четырех циклов.

Сигналы устройства управления и синхронизации:

- PSEN – разрешение программной памяти;
- ALE – выходной сигнал разрешения фиксации адреса;
- PROG – сигнал программирования;
- EA – блокировка работы с внутренней памятью;
- VPP – напряжение программирования;
- RST – сигнал общего сброса;
- VPD – вывод резервного питания памяти;
- XTAL – входы подключения кварцевого резонатора.

3.1.5 Организация портов ввода вывода. Микроконтроллер 8051 имеет четыре 8-разрядных двунаправленных порта ввода/вывода P0–P3, которые адресуются как регистры специальных функций с возможностью побитной адресации разрядов.

Большинство выводов микроконтроллера используются в качестве линий портов либо для альтернативных функций:

– порты P0 и P2 используются при обращении к внешней памяти. При этом на выходах P0 младший байт адреса внешней памяти мультиплексируется с вводимым/выводимым байтом. Выходы P2 содержат старший байт адреса внешней памяти, если адрес 16-разрядный. При использовании восьмиразрядного адреса портом P2 можно пользоваться для ввода/вывода информации обычным образом. При обращении к внешней памяти в P0 автоматически заносятся «1» во все биты. Информация в P2 при этом остается неизменной;

– порт P3 помимо обычного ввода и вывода используется для формирования и приема управляющих и информационных сигналов.

Альтернативные функции (табл. 3.3) могут быть активированы только, если в соответствующие биты порта P3 занесены «1».

Таблица 3.3

Альтернативные функции порта P3

Вывод порта	Функция
1	2
P3.0	RXD – вход последовательного порта
P3.1	TXD – выход последовательного порта
P3.2	INT0 – внешнее прерывание 0
P3.3	INT1 – внешнее прерывание 1

1	2
P3.4	T0 – вход таймера-счетчика 0
P3.5	T1 – вход таймера-счетчика 1
P3.6	WR – строб записи во внешнюю память данных
P3.7	RD – строб чтения из внешней памяти данных

Каждый из портов содержит регистр-защелку (P0–P3), выходную цепь и входной буфер. Каждый из разрядов регистра-защелки SFR является D-триггером, информация в который заносится с внутренней шины данных микроконтроллера по сигналу «Запись в SFR Px» ($x = 0, 1, 2, 3$) от центрального процессорного элемента (CPU). С прямого выхода D-триггера информация может быть выведена на внутреннюю шину по сигналу «Чтение SFR Px» от CPU, а с вывода микросхемы по сигналу «Чтение выводов Px».

3.1.6 Таймеры/счетчики. В базовых моделях семейства имеются два программируемых 16-битных таймера/счетчика (T/C0 и T/C1), которые могут быть использованы как в качестве таймеров, так и в качестве счетчиков внешних событий. Так как на распознавание периода требуются два машинных цикла, максимальная частота подсчета входных сигналов равна $1/24$ частоты резонатора. На длительность периода входных сигналов ограничений сверху нет.

Для управления режимами работы T/C и для организации их взаимодействия с системой прерываний используются регистры специальных функций: TMOD (табл. 3.4) и TCON (табл. 3.5).

Таблица 3.4

Регистр режима работы таймера/счетчика TMOD

Символ	Позиция	Имя и назначение
1	2	3
GATE	TMOD.7 для T/C1 и TMOD.3 для T/C0	Управление блокировкой. Если бит установлен, то таймер/счетчик «x» разрешен до тех пор, пока на входе «INTx» высокий уровень и бит управления «TRx» установлен. Если бит сброшен, то T/C разрешается

1	2	3		
C/T	TMOD.6 для T/C1 и TMOD.2 для T/C0	Бит выбора режима таймера/счетчика событий. Если бит сброшен, то работает таймер от внутреннего источника сигналов синхронизации, если установлен – счетчик работает от внешних сигналов на входе «Tx»		
M1	TMOD.5 для T/C1 и TMOD.1 для T/C0	Режим работы		
		M1	M0	Описание
		0	0	Таймер ВЕ48. «TLx» работает как 5-битный предделитель
M0	TMOD.4 для T/C1 и TMOD.0 для T/C0	0	1	16 битный таймер/счетчик. «ТНх» и «TLx» включен последовательно
		1	0	8-битный авто перезагружаемый таймер/счетчик. «ТНх» хранит значение, которое должно быть перезагружено в «TLx» каждый раз по переполнению

Таблица 3.5

Регистр управления/статуса таймера TCON

Символ	Позиция	Имя и назначение
1	2	3
TF1	TCON.7	Флаг переполнения таймера «1». Устанавливается аппаратно при переполнении таймера/счетчика. Сбрасывается при обслуживании прерывания аппаратно
TR1	TCON.6	Бит управления таймера «1». Устанавливается/сбрасывается программой для пуска/останова
TF0	TCON.5	Флаг переполнения таймера «0». Устанавливается аппаратно. Сбрасывается при обслуживании прерывания
TR0	TCON.4	Бит управления таймера «0». Устанавливается/сбрасывается программой для пуска/останова таймера/счетчика
IE1	TCON.3	Флаг фронта прерывания «1». Устанавливается аппаратно, когда детектируется срез внешнего сигнала INT1. Сбрасывается при обслуживании прерывания
IT1	TCON.2	Бит управления типом прерывания «1». Устанавливается / сбрасывается программно для спецификации запроса INT1 (срез/низкий уровень)

1	2	3
IE0	TCON.1	Флаг фронта прерывания «0». Устанавливается по срезу сигнала INT0. Сбрасывается при обслуживании прерывания
IT1	TCON.0	Бит управления типом прерывания 0. Устанавливается/сбрасывается программно для спецификации запроса INT0 (срез/низкий уровень)

Как следует из описания управляющих бит TMOD, для обоих T/C режимы работы 0, 1 и 2 одинаковы, а режимы 3 для T/C0 и T/C1 различны. Рассмотрим кратко работу T/C в каждом из режимов.

Режим 0. T/C в режиме 0 представляет собой 8-битный таймер, к входу которого подключен 5-битный предделитель частоты на 32. При переходе из состояния «все единицы» в состояние «все нули» устанавливается флаг прерывания от таймера TF0(1) (возможно организация прерывания). Установка бита GATE в «1» позволяет использовать таймер для измерения длительности импульсного сигнала подаваемого на вход запроса прерывания.

Режим 1. Работа любого T/C такая же, как и в режиме «0», за исключением того, что таймерный регистр имеет разрядность 16 бит.

Режим 2. В этом режиме работа организована таким образом, что переполнение (переход из состояния «все единицы» в состояние, «все нули») 8-битного счетчика TL1 приводит не только к установке флага TF1, но и автоматически перезагружает в TL1 содержимое старшего байта (TH1) таймерного регистра, которое предварительно было задано программным путем. Перегрузка оставляет содержимое TH1 неизменным. В режиме 2 T/C0 и T/C1 работают одинаково.

Режим 3. T/C0 и T/C1 работают по-разному:

- T/C1 сохраняет неизменным свое текущее содержимое.
- T/C0 в режиме 3 TL0 и TH0 функционируют как два независимых 8-битных счетчика. Работу TL0 определяют управляющие биты T/C0 (C/T, GATE TR0), входной сигнал INT0 и флаг переполнения TF0. Работу TH0, который может выполнять только функции таймера (подсчёт машинных циклов микро-ЭВМ), определяет управляющий бит TR1. При этом TH0 использует флаг переполнения TF1. Режим 3 используется в тех случаях, когда требуется наличие дополнительного 8-битного таймера или счетчика событий.

3.1.7 Последовательный порт микроконтроллера 8051. Через универсальный асинхронный приемопередатчик (UART) осуществляются прием и передача информации, представленной последовательным кодом (младшими битами вперед), в полном дуплексном режиме обмена. В состав приемопередатчика (последовательный порт) входят принимающий и передающий сдвигающие регистры, а также специальный буферный регистр (SBUF). Работой последовательного порта управляют два служебных регистра:

- регистр управления/статуса приемопередатчика SCON;
- бит SMOD регистра управления мощностью PCON.

Последовательный порт микроконтроллера 8051 может работать в четырех различных режимах.

Режим 0. Информация и передается, и принимается через вывод входа приемника (RXi, TXi). Принимаются или передаются 8 бит данных. Через вывод выхода передатчика (TXD; выдаются импульсы сдвига, которые сопровождают каждый бит. Частота передачи бита информации равна 1/12 частоты кварцевого резонатора).

Режим 1. В этом режиме передаются через вывод TXD или принимаются через RXD 10 бит информации: старт-бит «0», 8 бит данных и стоп-бит «1» при приеме информации в бит RB8 регистра управления/статуса приемопередатчика SCON. Скорость приема и передачи – величина переменная и задается таймером.

Режим 2. В этом режиме через вывод TXD передаются или через RXD принимаются 11 бит информации: старт-бит, 8 бит данных, программируемый девятый бит и стоп-бит. При передаче девятый бит данных может принимать значение «0» или «1», или, например, для повышения достоверности передачи путем контроля по четности в него может быть помещено значение признака паритета из слова состояния программы (PSW.0). При приеме девятый бит данных помещается в бит RB8 SCON, а стоп-бит, в отличие от режима 1, теряется. Частота приема/передачи выбирается программой и может быть равна либо 1/32, либо 1/64 частоты резонатора в зависимости от управляющего бита SMOD.

Режим 3. Совпадает с режимом 2 во всех деталях, за исключением частоты приема/передачи, которая является величиной переменной и задается таймером. Во всех случаях передача инициализируется инструкцией, в которой данные перемещаются в SBUF. Прием инициализируется при обнаружении перепада из «1» в «0» на входе

приемника. При этом в режиме 0 этот переход должен сопровождаться выполнением условий $R1 = 0$ и $REN = 1$, а для остальных режимов – $REN = 1$. Управление режимом работы приемопередатчика осуществляется через специальный регистр SCON (табл. 3.6).

Таблица 3.6

Биты регистра SCON

Символ	Позиция	Имя и назначение	
1	2	3	
SM0	SCON.7	Биты управления режимом работы приемопередатчика. Устанавливаются/сбрасываются программно	
		SM0	SM1
		0	0
SM1	SCON.6	0	1
		1	0
		1	1
REN	SCON.4	Бит управления режимом приемопередатчика. Устанавливается программно для запрета приема сообщения, в котором девятый бит имеет значение «0». Бит разрешения приема. Устанавливается/сбрасывается программно для разрешения/запрета приема последовательных данных	
TB8	SCON.3	Передача бита 8. Устанавливается/сбрасывается программно для задания девятого передаваемого бита в режиме 9-битового передатчика	
RB8	SCON.2	Прием бита 8. Устанавливается/сбрасывается аппаратно для фиксации девятого принимаемого бита в режиме 9-битового приемника	
T1	SCON.1	Флаг прерывания передатчика. Устанавливается аппаратно при окончании передачи байта. Сбрасывается программно после обслуживания прерывания	
R1	SCON.0	Флаг прерывания приемника. Устанавливается аппаратно	

Регистр SCON содержит не только управляющие биты, определяющие режим работы последовательного порта, но и девятый бит принимаемых или передаваемых данных (RB8 и TB8) и биты прерывания приемопередатчика (R1 и T1).

Скорость (частота) приема/передачи определяется в зависимости от режима. В режиме 0 частота передачи зависит только от резонансной частоты кварцевого резонатора f_{PE3}

$$f = f_{PE3} / 12.$$

За машинный цикл последовательный порт передает бит информации. В режимах 1, 2 и 3 скорость приема/передачи зависит от значения управляющего бита SMOD в регистре PCON (табл. 3.7).

Таблица 3.7

Регистр управления мощностью PCON

Символ	Позиция	Наименование и функция
1	2	3
SMOD	PCON.7	Удвоенная скорость передачи. Если бит установлен в «1», то скорость передачи вдвое больше, чем при SMOD = 0. По сбросу SMOD = 0
–	PCON.6	Не используется
–	PCON.5	Не используется
–	PCON.4	Не используется
GF1 GF0	PCON.3 PCON.2	Флаги общего назначения
PD	PCON.1	Бит пониженной мощности. При установке бита в «1» – переходит в режим пониженной потребляемой мощности
IDL	PCON.0	Бит холостого хода. Если бит установлен в «1», – переход в режим холостого хода

В режиме 2 частота передачи определяется выражением

$$f = 2^{SMOD} \cdot f_{PE3} / 64.$$

Иными словами, при SMOD = 0 частота передачи f равна 1/64 частоты f_{PE3} , а при SMOD = 1 – 1/32 частоты f_{PE3} .

В режимах 1 и 3 в формировании частоты передачи, кроме управляющего бита SMOD, принимает участие таймер 1. Частота f зависит от частоты переполнения f_{OVL1} и определяется следующим образом

$$f = 2^{SMOD} \cdot f_{OVL1} / 32.$$

Прерывание от таймера 1 в этом случае должно быть заблокировано. Таймер может работать как в режиме таймера, так и в режиме счетчика. При этом частота передачи определяется выражением

$$f = 2^{SMOD} f_{PE3} / [32 \cdot 12 \cdot (256 - TH1)].$$

Отметим, что скорости приема и передачи могут различаться.

3.1.8 Система прерываний микроконтроллера 8051. Внешние прерывания INT0 и INT1 могут быть вызваны либо уровнем, либо переходом сигнала из «1» в «0» на входах в зависимости от значений управляющих бит IT0 и IT1 в регистре TCON. От внешних прерываний устанавливаются флаги IE0 и IE1 в регистре TCON, которые инициируют вызов соответствующей программы обслуживания прерывания. Сброс флагов выполняется аппаратно только в том случае, если прерывание было вызвано по переходу/срезу сигнала. Если же прерывание вызвано уровнем входного сигнала, то сбросом флага I должна управлять соответствующая подпрограмма обслуживания прерывания путем воздействия на источник прерывания.

Флаги запросов прерывания RI и TI от таймеров TF0 и TF1 сбрасываются автоматически при передаче управления подпрограмме обслуживания. F RI и TI устанавливаются блоком управления приемопередатчика аппаратно, но сбрасываться должны программно.

В блоке регистров специальных функций есть два регистра, предназначенных для управления режимом прерываний IE и уровнями приоритета IP. В более сложных модификациях микроконтроллеров семейства MCS-51 количество периферийных устройств увеличено, что приводит к необходимости использовать один вектор прерывания для нескольких устройств, либо добавить еще два регистра – режима (маски) и приоритета прерываний (табл. 3.8–3.9).

Таблица 3.8

Биты регистра масок прерывания IE

Символ	Позиция	Имя и назначение
1	2	3
EA	IE.7	Снятие блокировки прерывания. Сбрасывается программно для запрета всех прерываний независимо от состояний IE.4–IE.0
–	IE.6	Не используется

1	2	3
–	IE.5	Не используется
ES	IE.4	Бит разрешения прерывания, от приемопередатчика. Установка/сброс программой для разрешения/запрета прерываний от флагов TI или RI
ET1	IE.3	Бит разрешения прерывания от таймера. Установка/сброс программой для разрешения/запрета прерываний от таймера 1
EX1	IE.2	Бит разрешения внешнего прерывания 1. Установка/сброс программой для разрешения/запрета прерывания 1
ET0	IE.1	Бит разрешения прерывания от таймера 0. Установка/сброс программой для разрешения/запрета прерываний от таймера 0
EX0	IE.0	Бит разрешения внешнего прерывания 0. Установка/сброс программой для разрешения/запрета прерывания 0

Таблица 3.9

Биты регистра приоритетов прерываний IP

Символ	Позиция	Имя и назначение
1	2	3
–	IP.7–IP.5	Не используется
PS	IP.4	Бит приоритета приемопередатчика. Установка/сброс программой для присваивания прерыванию от приемопередатчика высшего/низшего приоритета
PT1	IP.3	Бит приоритета таймера 1. Установка/сброс для присваивания прерыванию от таймера 1 высшего/низшего приоритета
PX1	IP.2	Бит приоритета внешнего прерывания 1. Установка/сброс программой для присваивания высшего/низшего приоритета внешнему прерыванию INT1
PT0	IP.1	Бит приоритета таймера 0. Установка/сброс для присваивания прерыванию от таймера 0 высшего/низшего приоритета
PX0	IP.0	Бит приоритета внешнего прерывания 0. Установка/сброс программой для присваивания высшего/низшего приоритета внешнему прерыванию INTO

Система прерываний формирует аппаратный вызов (LCALL) соответствующей подпрограммы обслуживания, если она не заблокирована одним из следующих условий:

- в данный момент обслуживается запрос прерывания равного или высокого уровня приоритета;
- текущий машинный цикл;

– выполняется команда RETI или любая команда, связанная с обращением к регистрам IE или IP.

По коду LCALL система прерывания помещает в стек только содержимое счетчика команд (PC) и загружает в него адрес вектора соответствующей подпрограммы обслуживания. По адресу вектора должна быть расположена команда безусловной передачи управления (JMP) к начальному адресу подпрограммы обслуживания прерывания. В случае необходимости она должна начинаться командами записи в стек (PUSH) слова состояния программы (PSW), аккумулятора и т. д. и должна заканчиваться командами восстановления из стека (POP). Подпрограммы обслуживания прерывания должны завершаться командой RETI, по которой в счетчик команд перезагружается из стека сохраненный адрес возврата в основную программу. Команда RET также возвращает управление прерванной основной программе.

3.1.9 Система команд. Микро-ЭВМ рассматриваемого семейства имеют архитектуру SISC со стандартным набором команд, который включает в себя 111 основных команд. Их длина – один, два или три байта. Команды выполняются за один или два машинных цикла. Всего микро-ЭВМ выполняют 13 типов команд (табл. 3.10). Первый байт команды всегда содержит код операции (КОП), второй и третий – адреса операндов или их непосредственные значения.

Таблица 3.10

Типы команд микроконтроллера семейства 8051

Тип команды	Первый байт D7–D0	Второй байт D7–D0	Третий байт D7–D0
1	2	3	4
тип 1	коп		–
тип 2	коп	#d	–
тип 3	коп	ad	–
тип 4	коп	bit	–
тип 5	коп	rel	–
тип 6	коп	a7–a0	–
тип 7	коп	ad	#d
тип 8	коп	ad	rel
тип 9	коп	ads	add

1	2	3	4
тип 10	коп	#d	rel
тип 11	коп	bit	rel
тип 12	коп	ad16h	ad16l
тип 13	коп	#d16h	#d16l

Обозначения, принятые в таблице 3.10:

ad – адрес прямоадресуемого байта;

ads – адрес прямо адресуемого байта-источника;

add – адрес прямо адресуемого байта-получателя;

ad11 – 11-разрядный абсолютный адрес перехода;

ad16 – 16-разрядный абсолютный адрес перехода;

rel – относительный адрес перехода;

#d – непосредственный операнд;

#d16 – непосредственный операнд (2 байта);

bit – адрес прямо адресуемого бита;

/bit – инверсия прямо адресуемого бита.

Состав операндов включает в себя операнды четырёх типов: биты, 4-битные цифры, байты и 16-битные слова.

8051 имеет 128 программно-управляемых флагов пользователя. Для адресации битов используется прямой 8-битный адрес. Косвенная адресация битов невозможна.

8-битным операндом может быть ячейка памяти программ или данных, регистры специальных функций и порты ввода/вывода.

4-битные операнды используются только при операциях обмена SWAP и XCHD.

2-байтные операнды – это константы и прямые адреса, для представления которых используются второй и третий байты команды.

Система команд MCS-51 содержит 111 базовых команд, которые по функциональному признаку можно подразделить на следующие:

- пересылки данных;
- арифметических операций;
- логических операций;
- операций над битами;
- передачи управления.

Набор команд MCS-51 поддерживает режимы адресации, рассмотренные ниже.

Прямая адресация. Операнд определяется 8-битным адресом в инструкции. Эта адресация используется только для внутренней памяти данных и регистров SFR.

Косвенная адресация. В этом случае инструкция адресует регистр, содержащий адрес операнда. Данный вид адресации может применяться при обращении как к внутреннему, так и к внешнему ОЗУ. Для указания 8-битных адресов могут использоваться регистры R0 и R1 выбранного регистрового банка или указатель стека SP.

Для 16-битной адресации используется только регистр «указатель данных» (DPTR, Data Pointer).

Регистровая адресация. Применяется для доступа к регистрам R0 + R7 выбранного банка. Выбор одного из четырех регистровых банков осуществляется программированием битов селектора банка (RS1, RS0) в PSW.

Непосредственная адресация. Операнд содержится непосредственно в поле команды вслед за кодом операции и может занимать один или два байта ($data_8$, $data_{16}$).

Индексная адресация. Используется при обращении к памяти программ и только при чтении. В этом режиме осуществляется просмотр таблиц в памяти программ. 16-битовый регистр (DPTR или PC) указывает базовый адрес требуемой таблицы, а аккумулятор указывает на точку входа в нее.

Неявная адресация. Некоторые инструкции используют индивидуальные регистры (например, операции с аккумулятором, DPTR), при этом данные регистры не имеют адреса, указывающего на них; это заложено в код операции.

Команды передачи данных представлены в табл. 3.11.

Таблица 3.11

Команды передачи данных

Название команды	Мнемокод	Операция
1	2	3
Пересылка в аккумулятор из регистра ($n = 0 \dots 7$)	MOV A, Rn	(A) ← (Rn)
Пересылка в аккумулятор прямоадресуемого байта	MOV A, ad	(A) ← (ad)
Пересылка в аккумулятор байта из РПД ($i = 0, 1$)	MOV A, @Ri	(A) ← ((Ri))

Продолжение табл. 3.11

1	2	3
Загрузка константы в аккумулятор	MOV A, #d	(A) ← #d
Пересылка в регистр из аккумулятора	MOV Rn, A	(Rn) ← (A)
Пересылка в регистр прямоадресуемого байта	MOV Rn, ad	(Rn) ← (ad)
Загрузка в регистр константы	MOV Rn, #d	(Rn) ← #d
Пересылка по прямому адресу аккумулятора	MOV ad, A	(ad) ← (A)
Пересылка по прямому адресу регистра	MOV ad, Rn	(ad) ← (Rn)
Пересылка прямоадресуемого байта по прямому адресу	MOV add, ads	(add) ← (ads)
Пересылка байта из РПД по прямому адресу	MOV ad, @Ri	(ad) ← ((Ri))
Пересылка по прямому адресу константы	MOV ad, #d	(ad) ← #d
Пересылка в РПД из аккумулятора	MOV @Ri, A	((Ri)) ← (A)
Пересылка в РПД прямоадресуемого байта	MOV @Ri, ad	((Ri)) ← (ad)
Пересылка в РПД константы	MOV @Ri, #d	((Ri)) ← #d
Загрузка указателя данных	MOV DPTR, #d16	(DPTR) ← #d16
Пересылка в аккумулятор байта из ПП	MOVC A, @A+DPTR	← ((A) + (DPTR))
Пересылка в аккумулятор байта из ПП	MOVC A, @A+PC	(PC) ← (PC) + 1, (A) ← ((A) + (PC))
Пересылка в аккумулятор байта из ВПД	MOVX A, @Ri	(A) ← ((Ri))
Пересылка в аккумулятор байта из расширенной ВПД	MOVX A,@DPTR	(A) ← ((DPTR))
Пересылка в ВПД из аккумулятора	MOVX @Ri, A	((Ri)) ← (A)
Пересылка в расширенную ВПД из аккумулятора	MOVX @DPTR,A	((DPTR)) ← (A)
Загрузка в стек	PUSH ad	(SP)←-(SP)+1,((SP))←-(ad)
Извлечение из стека	POP ad	(ad)←-(SP), (SP)←-(SP)-1
Обмен аккумулятора с регистром	XCH A, Rn	(A) ↔ (Rn)
Обмен аккумулятора с прямоадресуемым байтом	XCH A, ad	(A) ↔ (ad)

Продолжение табл. 3.11

1	2	3
Обмен аккумулятора с байтом из РПД	XCH A, @Ri	$(A) \leftrightarrow ((Ri))$
Обмен младших тетрад аккумулятора и байта РПД	XCHD A, @Ri	$(A0...3) \leftrightarrow ((Ri) 0...3)$
<i>Арифметические операции</i>		
Сложение аккумулятора с регистром ($n = 0...7$)	ADD A, Rn	$(A) \leftarrow (A) + (Rn)$
Сложение аккумулятора с прямоадресуемым байтом	ADD A, ad	$(A) \leftarrow (A) + (ad)$
Сложение аккумулятора с байтом из РПД ($i = 0, 1$)	ADD A, @Ri	$(A) \leftarrow (A) + ((Ri))$
Сложение аккумулятора с константой	ADD A, #d	$(A) \leftarrow (A) + \#d$
Сложение аккумулятора с регистром и переносом	ADDC A, Rn	$(A) \leftarrow (A) + (Rn) + (C)$
Сложение аккумулятора с прямоадресуемым байтом и переносом	ADDC A, ad	$(A) \leftarrow (A) + (ad) + (C)$
Сложение аккумулятора с байтом из РПД и переносом	ADDC A, @Ri	$(A) \leftarrow (A) + ((Ri)) + (C)$
Сложение аккумулятора с константой и переносом	ADDC A, #d	$(A) \leftarrow (A) + \#d + (C)$
Десятичная коррекция аккумулятора	DA A	Если $(A0...3) > 9$ или $((AC) = 1)$, то $(A0...3) \leftarrow (A0...3) + 6$, затем если $(A4...7) > 9$ или $((C) = 1)$, то $(A4...7) \leftarrow (A4...7) + 6$
Вычитание из аккумулятора регистра и заема	SUBB A, Rn	$(A) \leftarrow (A) - (C) - (Rn)$
Вычитание из аккумулятора прямоадресуемого байта и заема	SUBB A, ad	$(A) \leftarrow (A) - (C) - ((ad))$
Вычитание из аккумулятора байта РПД и заема	SUBB A, @Ri	$(A) \leftarrow (A) - (C) - ((Ri))$
Вычитание из аккумулятора константы и заема	SUBB A, d	$(A) \leftarrow (A) - (C) - \#d$
Инкремент аккумулятора	INC A	$(A) \leftarrow (A) + 1$
Инкремент регистра	INC Rn	$(Rn) \leftarrow (Rn) + 1$
Инкремент прямоадресуемого байта	INC ad	$(ad) \leftarrow (ad) + 1$
Инкремент байта в РПД	INC @Ri	$((Ri)) \leftarrow ((Ri)) + 1$

Продолжение табл. 3.11

1	2	3
Инкремент указателя данных	INC DPTR	$(DPTR) \leftarrow (DPTR) + 1$
Декремент аккумулятора	DEC A	$(A) \leftarrow (A) - 1$
Декремент регистра	DEC Rn	$(Rn) \leftarrow (Rn) - 1$
Декремент прямоадресуемого байта	DEC ad	$(ad) \leftarrow (ad) - 1$
Декремент байта в РПД	DEC @Ri	$((Ri)) \leftarrow ((Ri)) - 1$
Умножение аккумулятора на регистр В	MUL AB	$(B)(A) \leftarrow (A)*(B)$
Деление аккумулятора на регистр В	DIV AB	$(B).(A) \leftarrow (A)/(B)$
<i>Логические операции</i>		
Логическое «И» аккумулятора и регистра	ANL A, Rn	$(A) \leftarrow (A) \text{ AND } (Rn)$
Логическое «И» аккумулятора и прямоадресуемого байта	ANL A, ad	$(A) \leftarrow (A) \text{ AND } (ad)$
Логическое «И» аккумулятора и байта из РПД	ANL A, @Ri	$(A) \leftarrow (A) \text{ AND } ((Ri))$
Логическое «И» аккумулятора и константы	ANL A, #d	$(A) \leftarrow (A) \text{ AND } \#d$
Логическое «И» прямоадресуемого байта и аккумулятора	ANL ad, A	$(ad) \leftarrow (ad) \text{ AND } (A)$
Логическое «И» прямоадресуемого байта и константы	ANL ad, #d	$(ad) \leftarrow (ad) \text{ AND } \#d$
Логическое «ИЛИ» аккумулятора и регистра	ORL A, Rn	$(A) \leftarrow (A) \text{ OR } (Rn)$
Логическое «ИЛИ» аккумулятора и прямоадресуемого байта	ORL A, ad	$(A) \leftarrow (A) \text{ OR } (ad)$
Логическое «ИЛИ» аккумулятора и байта из РПД	ORL A, @Ri	$(A) \leftarrow (A) \text{ OR } ((Ri))$
Логическое «ИЛИ» аккумулятора и константы	ORL A, #d	$(A) \leftarrow (A) \text{ OR } \#d$
Логическое «ИЛИ» прямоадресуемого байта и аккумулятора	ORL ad, A	$(ad) \leftarrow (ad) \text{ OR } (A)$
Логическое «ИЛИ» прямоадресуемого байта и константы	ORL ad, #d	$(ad) \leftarrow (ad) \text{ OR } \#d$

Продолжение табл. 3.11

1	2	3
Исключающее «ИЛИ» аккумулятора и регистра	XRL A, Rn	$(A) \leftarrow (A) \text{ XOR } (Rn)$
Исключающее «ИЛИ» аккумулятора и прямоадресуемого байта	XRL A, ad	$(A) \leftarrow (A) \text{ XOR } (ad)$
Исключающее «ИЛИ» аккумулятора и байта из РПД	XRL A, @Ri	$(A) \leftarrow (A) \text{ XOR } ((Ri))$
Исключающее «ИЛИ» аккумулятора и константы	XRL A, #d	$(A) \leftarrow (A) \text{ XOR } \#d$
Исключающее «ИЛИ» прямоадресуемого байта и аккумулятора	XRL ad, A	$(ad) \leftarrow (ad) \text{ XOR } (A)$
Исключающее «ИЛИ» прямоадресуемого байта и константы	XRL ad, #d	$(ad) \leftarrow (ad) \text{ XOR } \#d$
Сброс аккумулятора	CLR A	$(A) \leftarrow 0$
Инверсия аккумулятора	CPL A	$(A) \leftarrow \text{NOT}(A)$
Сдвиг аккумулятора влево циклический	RL A	$(A_{n+1}) \leftarrow (A_n),$ $n = 0 \div 6, (A_0) \leftarrow (A_7)$
Сдвиг аккумулятора влево через перенос	RLC A	$(A_{n+1}) \leftarrow (A_n), n = 0 \div 6$ $(A_0) \leftarrow (C), (C) \leftarrow (A_7)$
Сдвиг аккумулятора вправо циклический	RR A	$(A_n) \leftarrow (A_{n+1}),$ $n = 0 \div 6, (A_7) \leftarrow (A_0)$
Сдвиг аккумулятора вправо через перенос	RRC A	$(A_n) \leftarrow (A_{n+1}), n = 0 \div 6$ $(A_7) \leftarrow (C), (C) \leftarrow (A_0)$
Обмен местами тетрад в аккумуляторе	SWAP A	$(A_{0...3}) \leftrightarrow (A_{4...7})$
<i>Операции с битами</i>		
Сброс переноса	CLR C	$(C) \leftarrow 0$
Сброс бита	CLR bit	$(b) \leftarrow 0$
Установка переноса	SETB C	$(C) \leftarrow 1$
Установка бита	SETB bit	$(b) \leftarrow 1$
Инверсия переноса	CPL C	$(C) \leftarrow \text{NOT}(C)$
Инверсия бита	CPL bit	$(b) \leftarrow \text{NOT}(b)$
Логическое «И» бита и переноса	ANL C, bit	$(C) \leftarrow (C) \text{ AND } (b)$
Логическое «И» инверсии бита и переноса	ANL C, /bit	$(C) \leftarrow (C) \text{ AND } (\text{NOT}(b))$
Логическое «ИЛИ» бита и переноса	ORL C, bit	$(C) \leftarrow (C) \text{ OR } (b)$

Продолжение табл. 3.11

1	2	3
Логическое «ИЛИ» инверсии бита и переноса	ORL C, /bit	$(C) \leftarrow (C) \text{ OR } (\text{NOT}(b))$
Пересылка бита в перенос	MOV C, bit	$(C) \leftarrow (b)$
Пересылка переноса в бит	MOV bit, C	$(b) \leftarrow (C)$
<i>Команды передачи управления</i>		
Переход в полном объеме ПП	LJMP ad16	$(PC) \leftarrow \text{ad16}$
Абсолютный переход внутри страницы в 2 Кб	AJMP ad11	$(PC) \leftarrow (PC) + 2,$ $(PC0-10) \leftarrow \text{ad11}$
Короткий относительный переход внутри страницы в 256 байт	SJMP rel	$(PC) \leftarrow (PC) + 2,$ $(PC) \leftarrow (PC) + \text{rel}$
Косвенный относительный переход	JMP @A+DPTR	$(PC) \leftarrow (A) + (\text{DPTR})$
Переход, если аккумулятор равен нулю	JZ rel	$(PC) \leftarrow (PC) + 2,$ если $(A) = 0,$ то $(PC) \leftarrow (PC) + \text{rel}$
Переход, если аккумулятор не равен нулю	JNZ rel	$(PC) \leftarrow (PC) + 2,$ если $(A) \neq 0,$ то $(PC) \leftarrow (PC) + \text{rel}$
Переход, если перенос равен единице	JC rel	$(PC) \leftarrow (PC) + 2,$ если $(C) = 1,$ то $(PC) \leftarrow (PC) + \text{rel}$
Переход, если перенос равен нулю	JNC rel	$(PC) \leftarrow (PC) + 2,$ если $(C) = 0,$ то $(PC) \leftarrow (PC) + \text{rel}$
Переход, если бит равен единице	JB bit, rel	$(PC) \leftarrow (PC) + 3,$ если $(b)=1,$ то $(PC) \leftarrow (PC) + \text{rel}$
Переход, если бит равен нулю	JNB bit, rel	$(PC) \leftarrow (PC) + 3,$ если $(b)=0,$ то $(PC) \leftarrow (PC) + \text{rel}$
Переход, если бит установлен, с последующим сбросом бита	JBC bit, rel	$(PC) \leftarrow (PC) + 3,$ если $(b) = 1,$ то $(b) \leftarrow 0$ и $(PC) \leftarrow (PC) + \text{rel}$
Декремент регистра и переход, если не нуль	DJNZ Rn, rel	$(PC) \leftarrow (PC) + 2,$ $(Rn) \leftarrow$ $(Rn) - 1,$ если $(Rn) \neq 0,$ то $(PC) \leftarrow (PC) + \text{rel}$
Декремент прямоадресуемого байта и переход, если не нуль	DJNZ ad, rel	$(PC) \leftarrow (PC) + 2,$ $(\text{ad}) \leftarrow$ $(\text{ad}) - 1,$ если $(\text{ad}) \neq 0,$ то $(PC) \leftarrow (PC) + \text{rel}$
Сравнение аккумулятора с прямоадресуемым байтом и переход, если не равно	CJNE A, ad, rel	$(PC) \leftarrow (PC) + 3,$ если $(A) \neq (\text{ad}),$ то $(PC) \leftarrow (PC) + \text{rel},$ если $(A) < (\text{ad}),$ то $(C) \leftarrow 1,$ иначе $(C) \leftarrow 0$

1	2	3
Сравнение аккумулятора с константой и переход, если не равно	CJNE A, #d, rel	$(PC) \leftarrow (PC) + 3$, если $(A) \neq \#d$, то $(PC) \leftarrow (PC) + rel$, если $(A) < \#d$, то $(C) \leftarrow 1$, иначе $(C) \leftarrow 0$
Сравнение регистра с константой и переход, если не равно	CJNE Rn, #d, rel	$(PC) \leftarrow (PC) + 3$, если $(Rn) \neq \#d$, то $(PC) \leftarrow (PC) + rel$, если $(Rn) < \#d$, то $(C) \leftarrow 1$, иначе $(C) \leftarrow 0$
Пустая операция	NOP	$(PC) \leftarrow (PC) + 1$
Сравнение байта в РПД с константой и переход, если не равно	CJNE @Ri, #d, rel	$(PC) \leftarrow (PC) + 3$, если $((Ri)) \neq \#d$, то $(PC) \leftarrow (PC) + rel$, если $((Ri)) < \#d$, то $(C) \leftarrow 1$, иначе $(C) \leftarrow 0$
Длинный вызов подпрограммы	LCALL ad16	$(PC) \leftarrow (PC) + 3$, $(SP) \leftarrow (SP) + 1$, $((SP)) \leftarrow (PC_{0..7})$, $(SP) \leftarrow (SP) + 1$, $((SP)) \leftarrow (PC_{8..15})$, $(PC) \leftarrow ad16$
Абсолютный вызов подпрограммы в пределах страницы в 2 Кб	ACALL ad11	$(PC) \leftarrow (PC) + 2$, $(SP) \leftarrow (SP) + 1$, $((SP)) \leftarrow (PC_{0..7})$, $(SP) \leftarrow (SP) + 1$, $((SP)) \leftarrow (PC_{8..15})$, $(PC_{0..10}) \leftarrow ad11$
Возврат из подпрограммы	RET	$(PC_{8..15}) \leftarrow ((SP))$, $(SP) \leftarrow (SP) - 1$, $(PC_{0..7}) \leftarrow ((SP))$, $(SP) \leftarrow (SP) - 1$
Возврат из подпрограммы Обработки прерывания	RETI	$(PC_{8..15}) \leftarrow ((SP))$, $(SP) \leftarrow (SP) - 1$, $(PC_{0..7}) \leftarrow ((SP))$, $(SP) \leftarrow (SP) - 1$

3.2. Интегрированная среда разработки μ Vision2

Интегрированная среда разработки μ Vision2 представляет собой приложение, содержащее в себе редактор, менеджер проектов и компоновщик. μ Vision2 поддерживает все средства Keil для микроконтроллеров 8051, включая компилятор C, ассемблер, сборщик и преобразователь object/HEX. μ Vision2 помогает контролировать разработку вашего приложения, предоставляя следующие возможности:

- полнофункциональный редактор кода;
- база данных устройств для настройки средств разработки;
- менеджер проектов для создания и управления проектами;

- интегрированный сборщик для компиляции, сборки приложений;
- диалоги для всех средств разработки;
- отладчик с высокоскоростной симуляцией ЦПУ и периферии;
- развитый графический интерфейс;
- ссылки на инструкции к средствам разработки базы данных устройств и пользовательские инструкции.

3.2.1 Панель управления. Группы файлов *File Groups* позволяют сгруппировать связанные друг с другом файлы в проект.

Целевые файлы проекта *Project Targets* позволяют создавать несколько программ из одного проекта. Вам может понадобиться один файл-мишень *Target* для тестирования и другой конечный файл в качестве релиза вашего приложения. Каждый такой файл будет содержать индивидуальный набор инструментов в одном файле проекта.

Чтобы выбрать микроконтроллер для проекта, укажите имя файла-мишени в окне проекта и откройте контекстное меню. Далее используйте команду Выбор устройства *Select device*.

Цепь инструментов хранит связи между *Include* и *Source* файлами. *Build page* окна вывода информации показывает информацию инструментов во время генерации кода.











μVision2 позволяет устанавливать контрольные точки в программе. Для этого используйте кнопки на панели *Edit Toolbar*. После создания программы вы можете включить отладчик с помощью кнопки *Debug*, контрольные точки станут активны в режиме отладки.

μVision2 автоматически конфигурирует симулятор периферийного оборудования под выбранный из библиотеки микроконтроллер. Меню периферийного диалога, вызываемое при помощи команд меню *Debug*, позволяет просматривать и изменять статус встроенных модулей, включая системную и BUS конфигурацию. Все прерывания, включая таймеры, также симулируются. μVision2 симулирует преобразование времени и прерываний встроенного АЦП. В диалоге АЦП можно задать значение напряжения аналоговых вводов.

Окно последовательного порта предоставляет терминальный симулятор для встроенного в чип UART. HEX и ASCII режимы могут быть выбраны из контекстного меню.

3.2.2 Панель инструментов. Панель инструментов предоставляет доступ к командам редактирования текста, управления проектами, настройкам средств разработки, отладке программ, командам управления окнами и справочной системой (табл. 3.12).

Команды панели инструментов

Команда (комбинация клавиш)	Панель инструментов	Описание
1	2	3
<i>Меню Вид</i>		
Status Bar		Панель состояния
File Toolbar		Отобразить или скрыть панель инструментов «File»
Build Toolbar		Отобразить или скрыть панель инструментов компиляции
Debug Toolbar		Панель инструментов отладки
Project Window		Отобразить или скрыть окно проекта
Output Window		Отобразить или скрыть окно вывода
Source Browser		Открыть окно просмотра кода
Disassembly Window		Отобразить или скрыть окно дизассемблера
Watch&Call Stack Window		Отобразить или скрыть окно просмотра и стека вызова
Memory Window		Отобразить или скрыть окно содержимого памяти
Code Coverage Window		Отобразить или скрыть окно кода
Performance Analyzer Window		Отобразить или скрыть окно анализатора исполнения
Symbol Window		Окно символа
Serial Window #1		Отобразить или скрыть окно последовательного порта 1
Toolbox		Отобразить или скрыть панель инструментов
Options...		Параметры редактора
<i>Меню «Project»</i>		
New Project...		Создать новый проект
Open Project...		Открыть существующий проект
Close Project...		Закреть текущий проект
Target Environment		Определить путь к включаемым и библиотечным файлам
Targets, Groups, Files		Подключить целевые файлы, группы файлов и файлы проекта
Select Device for Target		Выбрать процессор

Продолжение табл. 3.12

1	2	3
Options... (Alt + F7)		Изменить настройки целевого устройства, группы или файла
		Настроить текущую цель
		Выбрать текущую цель
Build Target (F7)		Перевести измененные файлы и компилировать приложение
Rebuild Target		Перевести все файлы проекта и компилировать приложение
Translate... (Ctrl + F7)		Компилировать текущий файл
Stop Build		Прекратить процесс компиляции
<i>Меню «Debug»</i>		
Start/Stop Debugging (Ctrl + F5)		Включить или выключить режим отладки μVision2
Go (F5)		Запустить программы до следующей точки прерывания
Step (F11)		Выполнить единичный шаг с заходом в функцию
Step Over (F10)		Выполнить единичный шаг без захода в функцию
Step out of Current Function (Ctrl + F11)		Выполнить единичный шаг с выходом из функции
Stop Running (ESC)		Прекратить выполнение программы
Insert/Remove Breakpoint		Установить точку прерывания на текущую строку
Enable/Disable Breakpoint		Включить/выключить точку прерывания на текущей строке
Disable All Breakpoints		Выключить все точки прерывания в программе
Kill All Breakpoints		Удалить из программы все точки прерывания
Show Next Statement		Показать следующую исполняемую инструкцию
Enable/Disable Trace Recording		Включить запись трассировки для проверки инструкции
View Trace Records		Просмотреть предыдущую выполненную инструкцию
Performance Analyzer...		Открыть диалог настройки анализатора исполнения

1	2	3
Function Editor...		Редактировать функции и INI файл отладчика
<i>Меню «Peripherals»</i>		
Reset CPU		Установить ЦПУ в начальное состояние (сброс)
<i>Меню «Tools»</i>		
Setup PC-Lint...		Настроить PC-Lint
Lint		Запустить активный файл PC-Lint
Lint All C Source files		Запустить в PC-Lint исходный код на языке C
Setup Easy-Case...		Настроить Siemens Easy-Case
Customize Tools Menu...		Добавить программу в меню инструментов

3.2.3 Представление чисел. Способы представления чисел, типы используемых данных, типы памяти и операторы μ Vision2 приведены в табл. 3.13–3.16, соответственно.

Таблица 3.13

Способ представления чисел

Формат	Синтаксис	Пример
1	2	3
Десятичный	D «число», число	D «100», 100
Шестнадцатиричный	H «число», 0xчисло	H «f9», 0xAF00
Восьмиричный	O «число»	O «777»
Двоичный	B «число»	B «11110000»
Символьный	«символ», A»символ»	«C», A «C»

Таблица 3.14

Типы используемых данных

Тип данных	Биты	Байт	Размер Данных
1	2	3	4
bit	1	–	0 to 1
signed char	8	1	–128 to +127
unsigned char	8	1	0 to 255
enum	16	2	–32768 to +32767
signed short	16	2	–32768 to +32767

1	2	3	4
unsigned short	16	2	0 to 65535
signed int	16	2	-32768 to +32767
unsigned int	16	2	0 to 65535
signed long	32	4	-2147483648 to +2147483647
unsigned long	32	4	0 to 4294967295
float	32	4	$\pm 1.175494E-38$ to $3.402823E+38$
sbit	1	-	0 to 1
sfr	8	1	0 to 255
sfr16	16	2	0 to 65535

Таблица 3.15

Типы используемой памяти

Тип памяти	Описание
1	2
code	Данные в памяти программ (64 КВ). Доступ по команде <code>MOVC @A+DPTR</code>
data	Данные, расположенные во встроенной памяти данных, адресуемые непосредственно (128В)
idata	Данные, расположенные во встроенной памяти данных, адресуемые косвенно (256В)
bdata	Доступ к бит адресуемой встроенной памяти данных (бит-данные) (16В)
xdata	Данные, расположенные во внешней памяти данных. Доступны по команде <code>MOVX @DPTR</code>
pdata	Страницы данных во внешней памяти данных (256В). Доступны по команде <code>MOVX @Rn</code>

Таблица 3.16

Операторы C-51

Оператор	Описание	Пример
1	2	3
(левая скобка	$1 + (d * 4)$
)	правая скобка	$(length + 1) * 255$
!	операция "НЕ" (логическая инверсия)	$if ! (a - b)$
~	инверсия	<code>flags = ~ flags</code>
-	отрицательное число (вторая инверсия)	$- 1 * length$
high	выделить старший байт слова	<code>movlw high llasid</code>

1	2	3
low	выделить младший байт слова	movlw low (llasid + 2551)
*	умножение	a = c * b
/	деление	a = b / c
%	модуль	length = total % 16
+	сложение	tot_len = length * 8 + 1
-	вычитание	Entry_Son = (Tot - 1) / 8
<<	сдвиг влево	val = flags << 1
>>	сдвиг вправо	val = flags >> 1
>=	больше либо равно	if ent >= num
>	больше	if ent > num
<	меньше	if ent < num
<=	меньше либо равно	if ent <= num
==	равно	if ent = num
!=	не равно	if ent != num
&	поразрядное "И"	flags = flags & err_bit
^	поразрядное "ИСКЛЮЧАЮЩЕЕ ИЛИ"	flags = flags ^ err_bit
	поразрядное "ВКЛЮЧАЮЩЕЕ ИЛИ"	flags = flags err_bit
&&	логическое "И"	if (len = 512) && (b = c)
	логическое "ИЛИ"	if (len = 512) (b = c)
=	установить равному...	entry_index = 0
+=	сложить и установить равному...	entry_index += 1
-=	вычесть и установить равному...	entry_index -= 1
*=	умножить и установить равному...	entry_index *= length
/=	делить и установить равному...	entry_index /= length
%=	модуль и установить равному...	entry_index %= 8
<<=	сдвиг влево и установить равному...	entry_index << 3
>>=	сдвиг вправо и установить равному...	entry_index >> 4
&=	"И" и установить равному...	entry_index &= err_flags
=	"ВКЛЮЧАЮЩЕЕ ИЛИ" и установить равному...	entry_index = err_flags
^=	"ИСКЛЮЧАЮЩЕЕ ИЛИ" и установить равному...	entry_index ^= err_flags
++	увеличить на 1 (инкремент)	i ++
--	уменьшить на 1 (декремент)	i --

3.3. Порядок создания проекта

Для создания проекта выполняются следующие шаги:

1) **Выбор папки проекта.** Создайте на диске папку проекта. Далее в μ Vision2 выберите *Project > New Project*, укажите путь к созданной папке и введите имя файла проекта. Нажмите *OK*.

2) **Выбор микроконтроллера.** Из списка «*Select Device for Target “...”*» выберите тип используемого микроконтроллера. На вопрос программы о копировании стандартного кода МК 8051 в папку проекта ответьте «Да». В результате в области проекта в группе источников появится новый подключенный файл «STARTUP.A51»

3) **Параметры проекта.** В панели параметров проекта (*Project > Option for Target “...”*) задайте параметры проекта и выходных файлов, состав листинга программы, компиляторы для C51 и ассемблера.

4) **Открытие исходного файла.** Можете открыть исходный файл для редактирования: *File > Open*, или создать новый: *File > New*. Тип файла по умолчанию не определяется.

5) **Создание программы.** Напишите исходный текст вашей программы, используя команды C51-компилятора.

6) **Добавление исходного файла в проект.** Нажав правую клавишу мыши, вызовите контекстное меню, где нажмите на «*Add File to Group...*». В открывшемся окне выберите нужный файл или файлы.

7) **Компиляция.** Откомпилируйте программу, выбрав команду *Project > Build All*. После завершения процесса будет вызвано окно с сгенерированной командной строкой, перечнем предупреждений или ошибок и результатом компиляции: были ошибки (*build failed*) или нет (*build successful*).

В результате выполнения данного этапа в папке с вашим проектом создастся ряд дополнительных файлов, имеющих отношение к вашему проекту (*lst, hex*).

8) **Работа программы в пошаговом режиме.** Проконтролируйте правильность выполнения программ, для этого:

– перейдите в режим отладки: *Start / Stop Debugging*;

– перейдите в режим просмотра и стека вызова *Watch & Call Stack*;

– выполните программу в пошаговом режиме, для этого выберите команду *Debug > Step* (F10) или *Debug > Step Over* (F11). Каждый выбор команды меню будет выполнять одну команду программы. Результат контролируется в соответствующих вызванных панелях;

– вызовите окно последовательного порта 1 (*Serial Window #1*) и проверьте полученный результат, выводимый на последовательный порт. Результат представляется в десятичной системе счисления.

3.4. Индивидуальные задания

3.4.1 Согласно варианту (табл. 3.17) для соответствующего микроконтроллера семейства MCS51 построить подробный алгоритм и написать программу по нахождению значения функции.

Таблица 3.17

Варианты задания

№	Функция	№	Функция
1	2	3	4
1	$F = A \oplus \overline{B + C \wedge B}$	9	$F = \overline{A \wedge B - C \vee A}$
2	$F = A \oplus \overline{B - C \vee A}$	10	$F = A - 5DA2 \vee \overline{B \wedge C}$
3	$F = A + 24A2 \oplus \overline{B \wedge C}$	11	$F = A \oplus \overline{C \wedge B + C}$
4	$F = \overline{A \wedge C \oplus B} - C$	12	$F = A - \overline{B \vee C \wedge 72AB}$
5	$F = A - B \vee \overline{C \wedge 2F3B}$	13	$F = \overline{A \vee B + C} \oplus A$
6	$F = A \vee \overline{B + C} \oplus A$	14	$F = A - 4D3A \oplus \overline{C \vee B}$
7	$F = A \vee \overline{D58A + C} \oplus B$	15	$F = A + B \vee \overline{C \wedge 61DD}$
8	$F = A - \overline{B \oplus C \wedge B}$		

3.4.2 Создать проект и написать программу по нахождению значения функции для соответствующего микроконтроллера.

3.4.3 Проконтролировать правильность выполнения программы в пошаговом режиме с фиксацией результатов в соответствующих панелях μ Vision2.

3.5. Порядок выполнения работы

Выполнение задания рассмотрим на примере.

Задание. Найти значение функции $F = A + \overline{B \wedge C}$, где A, B, C – 2-байтные числа. Результат выведем посредством встроенного последовательного порта. $A = 4354, B = 6275, C = 952A$.

Решение. Составляем алгоритм решения задачи. Параллельно определяем параметры настройки периферийных модулей и распо-

ложение переменных в области регистров общего назначения, необходимых для решения задачи.

Согласно подразделу 3.3 создаем проект и сохраняем его в рабочую папку своей программы. Далее в проекте открываем созданный ранее файл и вносим туда написанную программу. Пример программы приведен в табл. 3.18.

Таблица 3.18

Исходный код программы

Файл «Faction»	
#include <reg51.h>	/* Определяем базовые регистры МК */ /* семейства MCS51 */
#include <stdio.h>	/* Определяем функции ввода вывода */
extern void output (unsigned int);	/*определяем переменную, расположенную*/ /*в другом файле (файл расположен в этой же папке)*/
void main (void)	
{	/* основная программа */
unsigned int PA, PB, PC;	/* определяем переменные, */ /*присутствующие в программе */
SCON = 0x52; /* SCON */	/* настройка последовательного порта */
TMOD = 0x20; /* TMOD */	/* аппаратно (2400 BAUD @12MHZ) */
TCON = 0x69; /* TCON */	
TH1 = 0xf3; /* TH1 */	
while (1) {	
output (~(PA+PB)&PC);	
}	
}	
Файл «Output»	
#include <stdio.h>	
char dummy_buffer [25];	
void output (unsigned int number)	
{	
printf ("\nresult: %d\n\n", number);	
}	

Компилируем программу и проверяем ее работоспособность при пошаговом выполнении программы (подраздел 3.3).

3.6. Требования к содержанию отчета

3.6.1 Подробный алгоритм по нахождению значения функции для соответствующего микроконтроллера.

3.6.2 Распечатанный файл *.LST, сгенерированный из проекта программой Vision2. Файл листинга должен содержать ассемблерную составляющую исходного кода.

3.6.3 Результат выполнения программы по нахождению значения функции.

3.6.4 Ответы на контрольные вопросы.

3.7. Контрольные вопросы

1. CISC архитектура микроконтроллера.
2. Организация памяти микроконтроллеров семейства MCS51.
3. Организация ввода/вывод микроконтроллеров семейства MCS51.
4. Организация последовательного порта микроконтроллеров семейства MCS51.
5. Организация таймеров микроконтроллера семейства MCS51.
6. Организация прерываний микроконтроллеров семейства MCS51.
7. Система и формат команд микроконтроллеров семейства MCS51.
8. Состав пакета Keil μ Vision2. Организация работы.

СПИСОК ЛИТЕРАТУРЫ

1. Белов, А. В. Конструирование устройств на микроконтроллерах. – СПб : Наука и техника, 2005. – 256 с.
2. Катцен, С. Всё, что вам необходимо знать о PIC микроконтроллерах / С. Катцен. – М. : Додека-XXI, 2008. – 656 с.
3. Корабельников, Е. А. Самоучитель по программированию PIC контроллеров для начинающих : руководство по конструированию устройств на микроконтроллерах / Е. А. Корабельников. – М. : Электронные книги, 2008. – 287 с.
4. Магда, Ю. Микроконтроллеры серии 8051. Практический подход / Ю. Магда. – М. : ДМК Пресс, 2008. – 224 с.
5. Предко, М. PIC-микроконтроллеры: архитектура и программирование / М. : ДМК пресс., 2010. – 512 с.
6. Предко, М. Руководство по микроконтроллерам : в 2 т. / М. Предко. – М. : Постмаркет, 2001. – 520 с.
7. Предко, М. Справочник по PIC-микроконтроллерам / М. Предко. – М. : ДМК пресс., 2002. – 488 с.
8. Тавернье, К. PIC-микроконтроллеры. Практика применения / К. Тавернье. – М. : ДМК пресс., 2002. – 273 с.
9. Трамперт, В. AVR-RISC микроконтроллеры / В. Трамперт. – Киев : МК Пресс., 2006. – 464 с.
10. Яценков, В. С. Микроконтроллеры MicroCHIP. Практическое руководство / В. С. Яценков. – М. : Горячая линия – Телеком, 2002. – 296 с.

СОДЕРЖАНИЕ

Лабораторная работа № 1. Изучение среды разработки MPLab. Создание простейших проектов для микроконтроллеров семейства Microchip	3
1.1. Микроконтроллеры семейства Microchip. Краткие теоретические сведения	3
1.2. Интегрированная среда разработки IDE MPLab	25
1.3. Создание проекта в среде разработки IDE MPLab	35
1.4. Индивидуальные задания	37
1.5. Порядок выполнения работы	38
1.6. Требования к содержанию отчета	40
1.7. Контрольные вопросы	40
Лабораторная работа № 2. Составление простейших программ ввода/вывода с использованием микроконтроллеров семейства Microchip	41
2.1. Организация периферийных модулей микроконтроллеров семейства Microchip. Краткие теоретические сведения	41
2.2. Программная реализация временной задержки	52
2.3. Индивидуальные задания	54
2.4. Порядок выполнения работы	55
2.5. Требования к содержанию отчета	61
2.6. Контрольные вопросы	61
Лабораторная работа № 3. Изучение среды разработки KEIL μ Vision. Создание простейших проектов для микроконтроллеров семейства MCS51	62
3.1. Микроконтроллеры семейства MCS51 (8051). Краткие теоретические сведения	62
3.2. Интегрированная среда разработки μ Vision2	83
3.3. Порядок создания проекта	89
3.4. Индивидуальные задания	91
3.5. Порядок выполнения работы	91
3.6. Требования к содержанию отчета	93
3.7. Контрольные вопросы	93
СПИСОК ЛИТЕРАТУРЫ	94

Учебное издание

ИСАЕВ Александр Витальевич
КРИВИЦКИЙ Петр Геннадьевич
ПАНТЕЛЕЕВ Константин Владимирович

ПРОГРАММИРУЕМЫЕ ЦИФРОВЫЕ УСТРОЙСТВА: МИКРОКОНТРОЛЛЕРЫ

Практикум

для студентов специальностей

1-38 02 01 «Информационно-измерительная техника»,

1-38 02 03 «Техническое обеспечение безопасности»,

1-54 01 02 «Методы и приборы контроля качества

и диагностики состояния объектов»

Редактор *Е. О. Германович*
Компьютерная верстка *Е. А. Беспанской*

Подписано в печать 06.02.2020. Формат 60×84 ¹/₁₆. Бумага офсетная. Ризография.

Усл. печ. л. 5,58. Уч.-изд. л. 4,36. Тираж 200. Заказ 569.

Издатель и полиграфическое исполнение: Белорусский национальный технический университет.
Свидетельство о государственной регистрации издателя, изготовителя, распространителя
печатных изданий № 1/173 от 12.02.2014. Пр. Независимости, 65. 220013, г. Минск.