МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Л. К. ГОЛОВИНА И. С. ЗУЕВ Г. А. ПЕТРОВ

ПРОЕКТИРОВАНИЕ МИКРОКОНТРОЛЛЕРНОЙ СИСТЕМЫ УПРАВЛЕНИЯ

Учебно-методическое пособие

Санкт-Петербург
Издательство СПбГЭТУ «ЛЭТИ»

2022

УДК 004.382.7(07)
ББК 3973.26я7
Г61

Головина Л. К., Зуев И. С., Петров Г. А.

Г61 Проектирование микроконтроллерной системы управления: учеб.-метод. пособие, СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2022. 44 с.

ISBN 978-5-7629-3076-5

Описана методика проектирования микроконтроллерной системы управления, даны варианты заданий к лабораторным работам и методические указания к их выполнению. Цикл лабораторных работ разработан для поддержки соответствующего курсового проекта, посвященного разработке микропроцессорной системы управления на базе микроконтроллера ADuC812 фирмы «Analog Devices», имеющего центральное процессорное ядро Intel MCS-51. Разработка и отладка программы управления, реализуемой микроконтроллером, выполняется в среде симулятора ADSim812.

Предназначено для студентов СПбГЭТУ «ЛЭТИ» специальности 09.03.01 направления подготовки «Информатика и вычислительная техника» дневной, вечерней и заочной форм обучения. Может быть полезно инженерно-техническим работникам этой сферы деятельности.

УДК 004.382.7(07)
ББК 3973.26я7

Рецензент канд. техн. наук., доцент С. В. Тихов (СПбГУПТД).

Утверждено
редакционно-издательским советом университета
в качестве учебно-методического пособия

ISBN 978-5-7629-3076-5

ВВЕДЕНИЕ

Современный *микроконтроллер* – это размещенная на кристалле сложная цифровая система, в состав которой входит 8-, 16-, или 32-разрядный процессор, внутренняя память программ, широкий набор интерфейсных и периферийных устройств. Эти однокристальные системы ориентированы в первую очередь на выполнение функций *управления* различными объектами (устройствами и процессами). Сфера их применения очень широка: от современной бытовой техники до сложнейших систем управления технологическими процессами и робототехническими комплексами.

За последние пять лет выпуск микроконтроллеров в мире увеличился более чем в три раза и исчисляется миллиардами штук в год.

Несмотря на непрерывное развитие и появление все новых и новых 16- и 32-разрядных микроконтроллеров, основная доля мирового микроконтроллерного рынка и по сей день остается за 8-разрядными устройствами, поскольку именно они являются самыми дешевыми управляющими системами и при этом обеспечивают приемлемую точность в 2,5 десятичных знака.

Среди 8-разрядных микроконтроллеров первенство принадлежит семейству *Intel Micro Control System* (MCS-51) по количеству разновидностей и количеству компаний, выпускающих его модификации. Фактически система MCS-51 стала одним из стандартов 8-битных микроконтроллеров, благодаря своей удачной *расширяемой архитектуре*. Все микроконтроллеры семейства MCS-51 имеют общее ядро MCS-51 с общей системой команд, а управление объектом осуществляется с помощью встроенных в систему *периферийных устройств*. Система команд ядра MCS-51 способна управлять ЛЮБЫМ периферийным устройством, даже тем, которого еще нет в проекте! Фирма «Intel» продала десяткам компаний по всему миру патент на использование ядра MCS-51. Купившая патент компания встраивает в микросхему свое периферийное устройство (устройства) и регистры специальных функций для управления им, резерв места для которых имеет ядро MCS-51, и выпускает микросхему на рынок.

В результате на сегодняшний день существует более 200 модификаций микроконтроллеров семейства MCS-51. Эти модификации включают в себя кристаллы с широчайшим спектром периферии от простых 20-выводныхустройств с одним таймером и 1 К программной памятью до сложней-

ших 100-выводных кристаллов с 10-разрядными АЦП, массивами таймеров/счетчиков, аппаратными умножителями, 64 К программной памяти на кристалле. Каждый год появляются все новые варианты представителей этого семейства с улучшенными характеристиками. Основными направлениями развития являются увеличение быстродействия, снижение напряжения питания и потребления, увеличение объема ОЗУ и FLASH-памяти на кристалле с возможностью внутрисхемного программирования, введение в состав периферии микроконтроллера сложных устройств типа системы управления приводами, CAN и USB интерфейсов и т. п.

Основными производителями клонов 51-го семейства в мире являются фирмы «Philips», «Siemens», «Intel», «Atmel», «Dallas», «Temic», «Oki», «AMD», «MHS», «Gold Star», «Winbond», «Silicon Systems» и ряд других.

Комплекс лабораторных работ (ЛР) «Проектирование микроконтроллерной системы управления» ставит задачу изучения архитектуры микроконтроллера с ядром MCS-51 и представляет единый цикл работ, каждая из которых является частью общей задачи управления определенным объектом микроконтроллерной системы. В процессе управления задействованы основные периферийные устройства микроконтроллера: порты ввода-вывода, таймеры-счетчики, аналого-цифровой преобразователь, последовательный интерфейс, системы прерывания и прямого доступа к памяти.

Базовым микроконтроллером в предлагаемом цикле ЛР является изделие ADuC812 фирмы «Analog Devices» со встроенным аналого-цифровым преобразователем и расширением сегмента внешней оперативной памяти до 16 Мб при 2-байтном внешнем обмене данными.

Цикл лабораторных работ (ЛР) состоит из шести работ:

1. Изучение системы команд MCS-51 и кросс-средств разработки программного обеспечения для однокристального микроконтроллера ADuC812.

2. Реализация логических функций.

3. Изучение режимов работы таймера/счетчика в микроконтроллере.

4. Изучение системы приоритетных прерываний.

5. Изучение работы последовательного порта.

6. Изучение работы аналого-цифрового преобразователя в микроконтроллере.

Цикл ЛР является апробацией отдельных частей курсового проекта [1]. С архитектурой MCS-51 рекомендуется ознакомиться по учебнику [2].

Лабораторная работа 1. ИЗУЧЕНИЕ СИСТЕМЫ КОМАНД MCS-51 И КРОСС-СРЕДСТВ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ОДНОКРИСТАЛЬНОГО МИКРОКОНТРОЛЛЕРА ADuC812

Цель работы:

1. Ознакомление с системой команд однокристального микроконтроллера (ОМК) MCS-51.

2. Получение навыков программирования для ОМК MCS-51.

3. Получение навыков работы с кросс-средствами разработки программного обеспечения для ОМК MCS-51.

1.1. Методические указания

Разработка программного обеспечения с использованием кросс-средств в общем случае включает следующие этапы:

1. Подготовка программы на входном языке с использованием любого текстового редактора. В настоящем цикле лабораторных работ в целях глубокого изучения архитектуры ОМК в качестве входного языка используется язык ассемблера.

2. Трансляция программы в объектный код.

3. Отладка программы с помощью программного отладчика (симулятора). В настоящем цикле лабораторных работ используется полноэкранный отладчик ADSIM812.

4. Загрузка объектного кода в память микроконтроллера.

5. Проверка правильности работы программы при выполнении ее непосредственно микроконтроллером.

В данном цикле лабораторных работ выполняются пункты 1, 2, 3.

Для подготовки исходного текста программы на входном языке может использоваться любой редактор, формирующий текстовый файл в коде ASCII. Использование редактора для ввода и редактирования исходного текста программы на языке ассемблера не имеет никаких особенностей и поэтому не рассматривается.

Для трансляции исходного текста программы в объектный код в данном цикле лабораторных работ используется ассемблер ASM51. На этапе трансляции выявляются и должны быть исправлены синтаксические ошибки в исходном тексте. При разработке достаточно сложных многомодульных программ

используется также редактор связей (линкер), который позволяет включать в исполняемую программу независимо разработанные и библиотечные модули. В настоящем цикле лабораторных работ редактор связей не используется.

Отладчик позволяет моделировать (эмулировать) работу ОМК в режимах покомандного, пофрагментного исполнения программы, останова по условию, инициирования прерывания и позволяет промоделировать практически все возможные варианты работы программы и тем самым убедиться в ее работоспособности либо выявить и исправить семантические ошибки в программе.

Язык ассемблера ASM-51 имеет стандартный формат операторов ассемблера ОМК MCS-51:

[<метка>:] <операция> [<операнд(ы)>] [;<комментарий>].

Здесь и далее в квадратных скобках указываются необязательные элементы синтаксических конструкций.

Поле метки содержит символическое имя адреса ячейки памяти, в которой хранится отмеченная команда или операнд. Метка представляет собой символьную комбинацию, начинающуюся с буквы и содержащую не более 13 символов. Метка должна отделяться от поля операнда двоеточием. Ассемблер TASM различает шрифт, используемый при наборе метки, т. е. «LABEL» и «label» будут интерпретированы как различные метки.

В поле операции записывается мнемоническое обозначение команды ОМК, например, MOV – пересылка, ADD – сложение и т. д.

В поле операндов задаются участвующие в операции операнды. Операторы ассемблера могут иметь ни одного, один или два операнда. В последнем случае операнды разделяются запятой. Операнд может быть задан своим адресом (прямым или косвенным) или непосредственным значением. Прямой адрес может быть задан числом, мнемоническим обозначением или символическим именем, например:

MOV R0,128 – используется мнемоническое обозначение регистра R0 и числовое значение адреса ячейки в памяти данных;

ADD A,SUM – используется мнемоническое обозначение аккумулятора и символическое обозначение адреса операнда в памяти данных.

Признаком косвенной адресации служит префикс @. Например: команда MOV @R0,A пересылает содержимое аккумулятора в ячейку резидентной памяти данных (РПД), адресуемую содержимым регистра R0 (косвенная адресация РПД осуществляется с использованием регистров R0 и R1 текущего банка,

а внешней памяти данных (ВПД) – с использованием регистра-указателя данных DPTR).

Непосредственный операнд представляется числом или символическим именем с обязательным префиксом #. Например, MOV R6,#15; ADDC A,#OP1. Задаваемые числами адреса и непосредственные значения должны содержать суффикс указания системы счисления: B – двоичная, O – восьмеричная, D – десятичная и H – шестнадцатеричная. Число без суффикса по умолчанию считается десятичным. Если шестнадцатеричная константа начинается с буквы, ей должен предшествовать нуль, например, 0A4H, 0D7H.

Для задания начального адреса размещения в памяти и конца программы, присвоения символьным идентификаторам действительных значений, инициализации констант и т. п. используются директивы (псевдокоманды) ассемблера, основные из которых представлены в табл.1.1.

Таблица 1.1

Псевдокоманды ассемблера

Директива	Комментарий
ORG	Задает начальный адрес объектного кода для следующей за директивой программы, например: ORG 100H
EQU	Присваивает символическому имени абсолютное значение, например: SUM EQU 48H
DB	Обеспечивает занесение в память программ байтовой константы или цепочки байтовых констант, например: DB 0BH,77H
DW	Обеспечивает занесение в память программ одной или нескольких двухбайтовых констант, иапример: DW 3504H,10FFH,65ECH
END	Указывает конец программы

Более подробное описание ASM-51 приведено в документации, выдаваемой преподавателем (файл asm51.pdf).

Особенностью ассемблера ASM-51 является необходимость определять адреса, соответствующие мнемоническим обозначениям регистров специальных функций (SFR). Задавать адреса SFR можно непосредственно в программе или включением в программу соответствующих файлов (MOD52 или MOD812), определяющих мнемонические обозначения регистров SFR.

В табл. 1.2 приведены адреса SFR ОМК MCS-51. В программе рекомендуется использовать мнемонические обозначения регистров SFR, однако следует знать, где в системе размещаются соответствующие регистры. Звездочкой отмечены регистры SFR, находящиеся в области булевого сегмента.

В результате ассемблирования создается файл листинга – <имя_файла>.lst, в

котором содержатся указания на ошибки, выявленные в процессе трансляции и, если ошибок нет, объектный файл в абсолютном двоичном формате – <имя_файла>.hex.

Таблица 1.2

Адреса регистров специальных функций

Мнемоническое обозначение	Название регистра	Адрес
* ACC	Аккумулятор	0E0H
* B	Регистр-расширитель аккумулятора	0F0H
* PSW	Слово состояния программы	0D0H
SP	Регистр указатель стека	81H
DPTR	Регистр указатель данных (DPH) (DPL)	83H 82H
* P0	Порт 0	80H
* P1	Порт 1	90H
* P2	Порт 2	0A0H
* P3	Порт 3	0B0H
* IP	Регистр приоритетов	0B8H
* IE	Регистр маски прерываний	0A8H
TMOD	Регистр режима таймера/счетчика	89H
* TCON	Регистр управления/статуса таймера	88H
TH0	Таймер 0 (старший байт)	8CH
TL0	Таймер 0 (младший байт)	8AH
TH1	Таймер 1 (старший байт)	8DH
TL1	Таймер 1 (младший байт)	8BH
* SCON	Регистр управления приемопередатчиком	98H
SBUF	Буфер приемопередатчика	99H
PCON	Регистр управления мощностью	87H

После исправления всех синтаксических ошибок (если таковые имелись) необходимо загрузить полученный объектный файл в отладчик для пошаговой отладки алгоритма. Используемый в данном цикле лабораторных работ полноэкранный отладчик ADSIM812 позволяет:

– загрузить для отладки сгенерированные ассемблером объектные файлы;

– просмотреть на экране дизассемблированный текст загруженной программы, включая адреса и коды команд, содержимое всех регистров ОМК, область памяти программ, резидентной и внешней памяти данных;

– выполнить загружаемую программу по шагам с просмотром результата после каждого шага и в непрерывном режиме с остановом по достижении задаваемых пользователем адресов;

– внести изменения в загружаемую программу в мнемонических обозначениях ассемблера, а также в машинных кодах;

– внести изменения в содержимое регистров, флагов и памяти в командном режиме и в режиме полноэкранного редактирования;

– получить трассировку программы;

– определить время выполнения загружаемой программы и её частей по встроенному счетчику.

Подробное описание отладчика ADSIM812 содержится в [1] и в документации, выдаваемой преподавателем (файл ADuC812.pdf).

1.2. Подготовка к работе

1. Повторить последовательность и назначение этапов подготовки программного обеспечения с использованием кросс-средств. Повторить синтаксис директив ассемблера ASM51, команды и режимы адресации OMK MCS-51.

2. Подготовить программу на языке ассемблера, выполняющую пересылку массива констант (8 чисел) из памяти программ (ПП) в резидентную память данных (РПД) по адресам 20H–27H. В программе необходимо предусмотреть инициализацию указанной области ПП константами с помощью директивы DB.

Считывание данных из ПП в цикле реализуется парой команд:

– MOV A,#0;

– MOVC A,@A+DPTR.

При этом двухбайтовый регистр-указатель DPTR в начале программы должен быть инициализирован командой MOV DPTR,#addr (addr– адрес первой константы), а затем инкрементироваться при каждом проходе цикла (INC DPTR). Пересылку значений в ячейки РПД целесообразно выполнять в цикле командой MOV @R1,A. Предварительно регистр R1 должен инициализироваться константой 20H (MOV R1,#20H), а затем инкрементироваться при каждом проходе цикла (INC R1).

Кроме того, для подсчета числа пересылок необходим счетчик цикла (например, регистр R0). Для организации цикла удобно использовать команду DJNZ R0,M1, которая декрементирует содержимое R0 и в случае его неравенства нулю передает управление на метку M1 (начало цикла).

3. Подготовить программу, копирующую в ячейки РПД с адресами 18H–1DH (регистровый банк 3) содержимое следующих регистров специальных функций в указанной последовательности: DPH, DPL, TH0, TL0, TH1, TL1. Для наблюдаемости процесса следует предварительно записать в эти регист-

ры некоторые отличные от нуля значения. Для выполнения пересылок использовать команду MOV @R0,ad, где ad – адрес источника (см. табл. 1.2). Регистр R0, используемый для косвенной адресации загружаемых ячеек РПД, должен относиться к *первому регистровому банку*. Программу следует демонстрировать в *пошаговом режиме*.

4. Подготовить программу для обработки измерительной информации. Исходные данные хранятся во внешней памяти данных в виде массива (10 чисел). Необходимо найти среднее значение x_{cp}, сравнить его с уставками – двумя константами $Q_{\max}$ и $Q_{\min}$, хранящимися также во внешней памяти данных. По результатам сравнения сформировать признак p и вывести его на линии порта P1:

$$x_{cp} \geq Q_{\max}, \qquad p := 11b$$
$$Q_{\max} > x_{cp} > Q_{\min}, \qquad p := 10b$$
$$x_{cp} \leq Q_{\min}, \qquad p := 00$$

Для демонстрации результатов среднее значение x_{cp} следует запомнить в регистре РПД; следует определить *реальные данные*, при которых сумма десяти чисел больше 255. Эту и последующие программы следует демонстрировать в *автоматическом режиме*, для чего организовать *зацикливание программы* как это делается в программе управления реальным объектом.

1.3. Порядок выполнения работы

1. Подготовить с помощью кросс-средств программы, составленные в соответствии с предыдущим разделом.

2. С помощью кросс-отладчика проверить правильность работы этих программ, записав предварительно исходные данные в заданные разделы памяти, определить время выполнения программ. Продемонстрировать работу программ преподавателю.

Содержание отчета

1. Титульный лист.

2. Задание на лабораторную работу.

3. Общий алгоритм подготовки программного обеспечения.

4. Схемы и описания алгоритмов программ, а также листинги соответствующих хорошо комментированных программ на ассемблере.

5. Оценки времени выполнения программ, полученные с помощью отладчика.

Контрольные вопросы

1. Поясните основные этапы разработки программного обеспечения для ОМК.
2. Какие режимы адресации реализованы в ОМК MCS-51? Какие регистры могут быть использованы для косвенной адресации РПД и ВПД?
3. Какие функциональные возможности имеет отладчик ADSIM812? Перечислите основные режимы работы отладчика.
4. Каким образом в отладчике можно установить точки прерывания?
5. Почему в программе сохранения регистров специальных функций в ВПД нельзя использовать цикл?

Лабораторная работа 2. РЕАЛИЗАЦИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ

Цель работы:

1. Ознакомление с командами логических операций ОМК MCS-51.
2. Изучение способов реализации логических функций в ОМК и их сравнительный анализ.

2.1. Методические указания

Одной из распространенных функций в системах локального управления является реализация логических (булевых) функций. Программная реализация таких функций с использованием ОМК MCS-51 возможна различными способами, в настоящей работе рассмотрено четыре способа.

Особенностью ОМК MCS-51 являются команды, обрабатывающие битовые операнды, что позволяет говорить о составе в его архитектуре «булевого процессора». Перечень таких команд:

CLR C – сброс флага переноса;

CLR bit – сброс флага прямо адресуемого бита;

SETB C – установка переноса;

SETB bit – установка прямо адресуемого бита;

MOV C,bit – пересылка бита в перенос;

MOV bit,C – пересылка переноса в бит.

CPL C – инверсия переноса;

CPL bit – инверсия прямо адресуемого бита;

ANL C,bit – конъюнкция переноса и прямо адресуемого бита;

ANL C,/bit – конъюнкция переноса и прямо адресуемого инверсного бита;

ORL C,bit – дизъюнкция переноса и прямо адресуемого бита;

ORL C,/bit – дизъюнкция переноса и прямо адресуемого инверсного бита;

Бит переноса C выполняет функции «булевого аккумулятора». В качестве прямо адресуемых бит булевого сегмента памяти используются биты ячеек резидентной памяти данных (РПД), размещенных в пространстве адресов с 20H по 2FH включительно. Кроме того, могут адресоваться отдельные биты аккумулятора A, регистра-расширителя B, портов и других регистров специальных функций (см. табл. 1.1). Карты адресов отдельных бит в РПД и блоке регистров специальных функций представлены соответственно на рис. 2.1 и 2.2.

Адреса РПД	D7							D0
00H	Регистровый банк 0							
08H	Регистровый банк 1							
10H	Регистровый банк 2							
18H	Регистровый банк 3							
20H	07	06	05	04	03	02	01	00
21H	0F	0E	0D	0C	0B	0A	09	08
22H	17	16	15	14	13	12	11	10
2EH	77	76	75	74	73	72	71	70
2FH	7F	7E	7D	7C	7B	7A	79	78

Рис. 2.1. Размещение банков регистров и адреса бит общего назначения булевого процессора

Адреса РПД	D7							D0	Имя регистра
0F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
0E0H	E7	E6	E5	E4	E3	E2	E1	E0	A
0D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
0B8H	–	–	–	BC	BB	BA	B9	B8	IP
0B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
0A8H	AF	–	–	AC	AB	AA	A9	A8	IE
0A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	P1
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
80H	87	86	85	84	83	82	81	80	P0

Рис. 2.2. Адреса бит в области SFR

Использование команд битовых операций позволяет, в частности, повысить эффективность программ вычисления булевых функций.

2.2. Подготовка к работе

В соответствии с вариантом задания составить таблицу истинности заданной функции $y = f(x_1, x_2, x_3, x_4)$. Варианты задания приведены в табл.2.1.

Таблица 2.1

Варианты задания логических функций

Вариант	Функция $y = f(x_1, x_2, x_3, x_4)$	Вариант	Функция $y = f(x_1, x_2, x_3, x_4)$
1	$x_1 \,\&\, \overline{x_2} \vee \overline{x_3} \,\&\, x_4$	6	$\overline{x_1} \,\&\, x_2 \,\&\, \overline{x_3} \oplus x_4$
2	$x_1 \vee (x_2 \,\&\, x_3 \,\&\, x_4)$	7	$(x_1 \vee \overline{x_2}) \oplus \overline{x_3} \,\&\, x_4$
3	$\overline{x_1 \vee x_2 \,\&\, x_3 \,\&\, \overline{x_4}}$	8	$(\overline{x_1} \oplus \overline{x_2} \,\&\, x_3) \vee x_4$
4	$(x_1 \vee \overline{x_2} \vee x_3) \,\&\, \overline{x_4}$	9	$(x_1 \vee \overline{x_2} \vee x_3) \,\&\, \overline{x_4}$
5	$(x_1 \oplus x_2) \,\&\, (\overline{x_3} \vee x_4)$	10	$\overline{x_1} \,\&\, x_2 \vee (\overline{x_3} \oplus x_4)$

В архитектуре ОМК MCS-51 возможны четыре метода вычисление логических функций.

1**. Вычисление логической функции с помощью команд байтовых поразрядных логических операций.** Использование этого подхода предполагает в качестве первого шага разделение входных булевых переменных, содержащихся в одном слове, т. е. аргументы булевой функции должны быть помещены в один и тот же разряд (например, младший) различных регистров или ячеек памяти. Это выполняется повторением нужного числа раз команд сдвига и пересылки. При этом для сохранения исходного кода аргументов его следует перед началом преобразования продублировать в некотором регистре (ячейке). Например, аргументы размещены в РПД по адресу 20H (младшая тетрада), при использовании для разделения регистров R0–R3, соответствующий фрагмент программы выглядит следующим образом:

```
MOV A,20H
MOV R0,A
RR A
MOV R1,A
RR A
MOV R2,A
RRA
MOV R3,A
```

Затем над этими регистрами выполняются команды логических операций (ANL, ORL, XRL, CPL), последовательность которых соответствует заданной логической функции, при этом в выделенном разряде формируется значение функции.

2. **Использование команд условных переходов.** При данном способе вычисление функции выполняется на основе разветвленного алгоритма типа дерева решений. Значения булевых аргументов последовательно анализируются командами условного перехода до тех пор, пока не определится значение функции. На рис. 2.3 в качестве примера приведен алгоритм реализации функции $y = x_1 \vee \overline{x_2} \,\&\, x_3$ с использованием команд условных переходов.

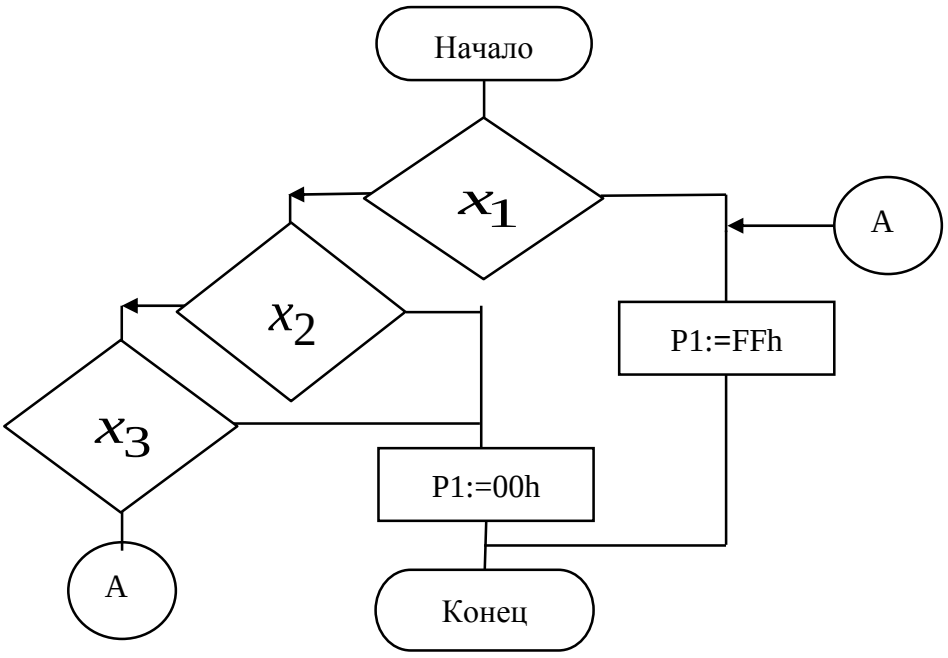

Рис. 2.3. Вычисление логических функций методом использования команд условных переходов

Отметим, что эффективность алгоритма в данном случае может зависеть от порядка проверки переменных с учетом конкретных особенностей реализуемой функции.

3. **Использование команд битовых операций.** При использовании команд битовых операций значение функции вычисляется методом непосредственного доступа к битам – аргументам логической функции. При этом булевы значения аргументов должны размещаться в регистрах либо ячейках РПД, поддерживающих этот режим доступа. На рис. 2.4 представлена блок-схема алгоритма вычисления логических функции способом использования команд битовых операций.

В качестве примера приведен детальный алгоритм вычисления функции $y = x_1 \ \& \ \overline{x}_2 \oplus (\overline{x}_3 \vee x_4)$, где для хранения промежуточных результатов используются младшие разряды аккумулятора и регистра-расширителя B .

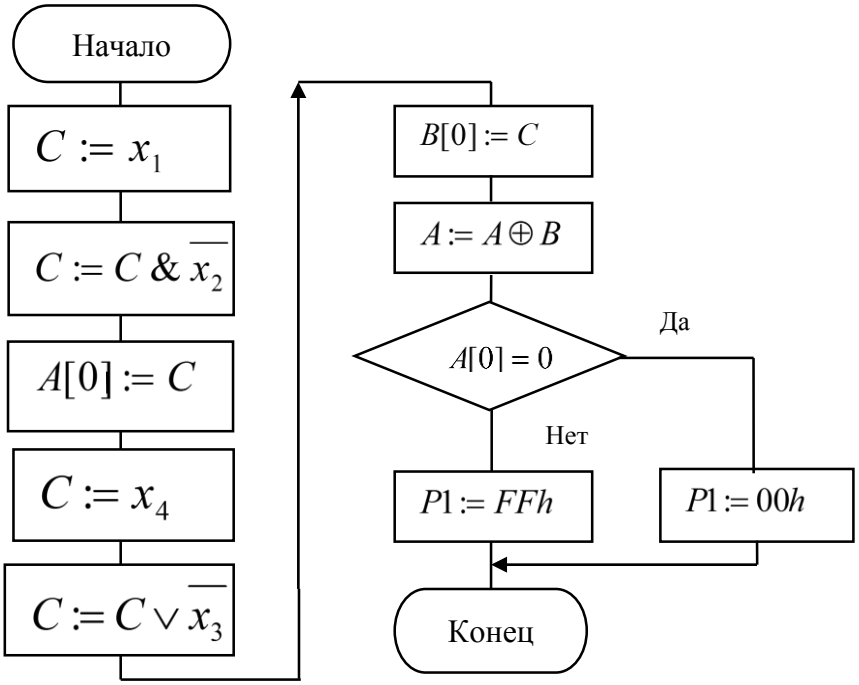

Рис. 2.4. Вычисление логических функции
методом использования команд битовых операций

Так как среди битовых команд отсутствует команда «Исключающее ИЛИ», данная функция выполняется с помощью соответствующей байтовой поразрядной логической команды между регистрами A и B (команда XRL A,B). Отметим, что эта функция может быть также реализована с использованием команд условных переходов.

4. **Табличный способ.** Идея этого способа заключается в использовании вводимой комбинации значений булевых аргументов в качестве адреса ячейки, содержимым которой является соответствующее значение функции. Таблица значений функции должна быть заранее загружена в память системы, начиная с некоторого базового адреса. Текущая комбинация входных значений определяет смещение адреса при обращении к этой таблице. Алгоритм реализации функции в данном случае включает в себя следующие шаги:

– формирование адреса доступа к таблице;

– считывание из таблицы значения функции и формирование массива значений функции;

– формирование массива значений функции для индикации.

Таблица значений функции для индикации может размещаться как в памяти программ так и в РПД или в ВПД. В рамках лабораторных работ целесообразно использовать первый вариант. С практической точки зрения предпочтительным может оказаться последний вариант, так как экономится внутренняя память и ослабляются ограничения на число аргументов функции. Для обращения к таблице в этом случае должна использоваться косвенная адресация через регистр-указатель данных DPTR. Если таблица размещается с адреса 1100H, то в старший байт регистра DPTR (регистр DPH с адресом 83H) необходимо загрузить значение 11H, а в младший (регистр DPL, адрес 82H) – комбинацию входных значений из ячейки ВПД с адресом 20H. Обращение к таблице осуществляется командой MOVX A,@DPTR, в результате выполнения которой в аккумулятор загружается код значения функции, после чего он должен выводиться в порт P1.

2.3. Порядок выполнения работы

1. Подготовить на ассемблере программы, реализующие заданную логическую функцию с использованием описанных методов. Значения аргументов функции целесообразно вводить, например, с линий порта P0.0–P0.3. Результат вывести в порт P1: если функция принимает значение «1», байт индикации имеет значение FFH, в противном случае – 00.

2. С помощью кросс-отладчика проверить правильность этих программ и оценить время вычисления функции различными методами для фиксированной комбинации входных переменных. Выбрать лучший алгоритм для вычисления заданной функции.

Содержание отчета

1. Таблица истинности заданной функции.

2. Схемы и описания алгоритмов вычисления логической функции различными методами и соответствующие листинги хорошо комментированных программ на ассемблере.

3. Оценки затрат памяти программ и данных, а также времени выполнения программы для рассмотренных способов реализации, полученные с помощью кросс-отладчика. Определить лучший алгоритм для вычисления заданной функции.

Контрольные вопросы

1. Перечислите команды битовых операций ОМК.

2. Какой из рассмотренных вариантов характеризуется:

а) минимальным объемом программной памяти;

б) минимальным объемом памяти данных;

в) минимальным временем вычисления функции.

3. Каковы достоинства и недостатки метода, основанного на использовании команд битовых операций, по сравнению с другими методами вычисления логической функции.

Лабораторная работа 3. ИЗУЧЕНИЕ РЕЖИМОВ РАБОТЫ ТАЙМЕРА/СЧЕТЧИКА В МИКРОКОНТРОЛЛЕРЕ

Цель работы:

1. Ознакомление с некоторыми применениями таймера/счетчика в микроконтроллере.

2. Изучение вопросов программирования и применения таймера/счетчика в основных режимах его работы.

3.1. Методические указания

В микроконтроллере MCS-51 в качестве счетчиков и таймеров могут быть использованы два 16-битных таймера/счетчика Т/С0 и Т/С1. В режиме таймера содержимое Т/С инкрементируется в каждом машинном цикле, состоящем из 12 периодов кварцевого резонатора. В режиме счетчика содержимое Т/С инкрементируется под воздействием внешнего сигнала на входах Т0, Т1 при переходе из 1 в 0. Так как для распознавания перехода требуется два машинных цикла, то максимальная частота подсчета входных сигналов составляет 1/24 частоты резонатора. На длительность входных сигналов ограничений сверху нет.

Управление конфигурацией Т/С и взаимодействие его с подсистемой прерывания осуществляется посредством двух регистров:

– регистра режима работы TMOD;

–.регистра управления/состояния TCON.

Назначение отдельных разрядов регистра TMOD приведено на рис. 3.1 и в табл. 3.1, а регистра TCON – на рис. 3.3 и в табл. 3.2.

C/T1				C/T0			
D7	D6	D5	D4	D3	D2	D1	D0
GATE	C/T	M1	M0	GATE	C/T	M1	M0

Рис.3.1. Формат регистра режима TMOD

Биты 7–4 и 3–0 используются для определения режимов работы Т/С1 и Т/С0 соответственно.

Таблица 3.1

Режимы работы таймеров/счетчиков

GATE		Управление блокировкой T/C1 (D7) и T/C0 (D3). Если бит установлен в 1, то T/C1(0) разрешен, пока на входе INT1(0) высокий уровень и бит управления TR1(0) установлен. Если бит сброшен, то T/C1(0) разрешается при установке бита TR в 1
C/T		Бит определения работы T/C в качестве таймера (0) или счетчика (1). Таймер работает от внутреннего источника сигналов синхронизации, а счетчик – от внешних сигналов на входе T1(0)
M1	M0	Определяют режимы работы T/C1 (T/C0)
0	0	Режим 0. Таймер TL работает как 5-битный предделитель, а TH – как 8-битный делитель. Таким образом, разрядность T/C в этом режиме равна 13
0	1	Режим 1. В этом режиме T/C представляет собой 16-битный таймер/счетчик. TH и TL включены последовательно
1	0	Режим 2. 8-битный автоперезагружаемый T/C. TH хранит значение, которое перезагружается в TL каждый раз по переполнению
1	1	Независимая работа TL0 и TH0. Таймер/счетчик 1 останавливается

Режим 0. Работа T/C в этом режиме показана на рис. 3.2, а на примере T/C0.

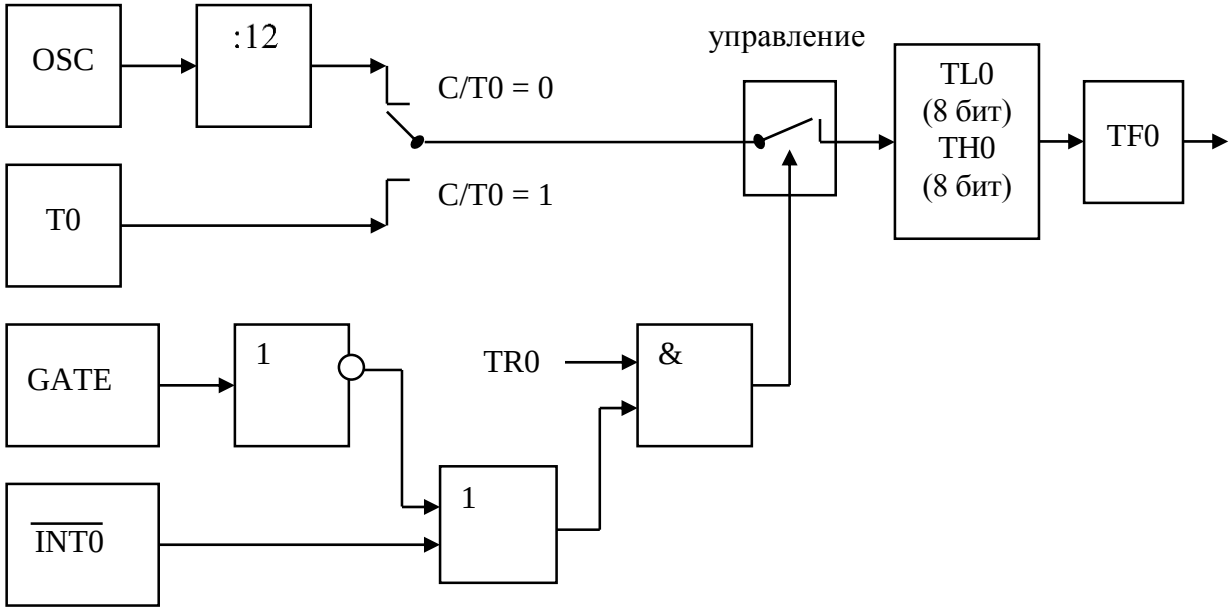

Рис. 3.2, а. Логика работы TC0 в режиме 0

В этом режиме разрядность T/C – 13 бит, при переходе T/C из состояния «все единицы» в состояние «все нули» устанавливается флаг прерывания TF0. Входной синхросигнал таймера разрешен (поступает на вход T/C), когда управляющий бит GATE (блокировка) равен 0, либо на внешний вход запроса прерывания INT0 поступает уровень 1. Необходимо отметить, что установка бита GATE в 1 позволяет использовать таймер для измерения длительности импульсного сигнала, поступающего на вход INT0.

Режим 1. Отличается от режима «0» только тем, что разрядность Т/С равна 16. Работа Т/С0 в этом режиме показана на рис.3.2, б.

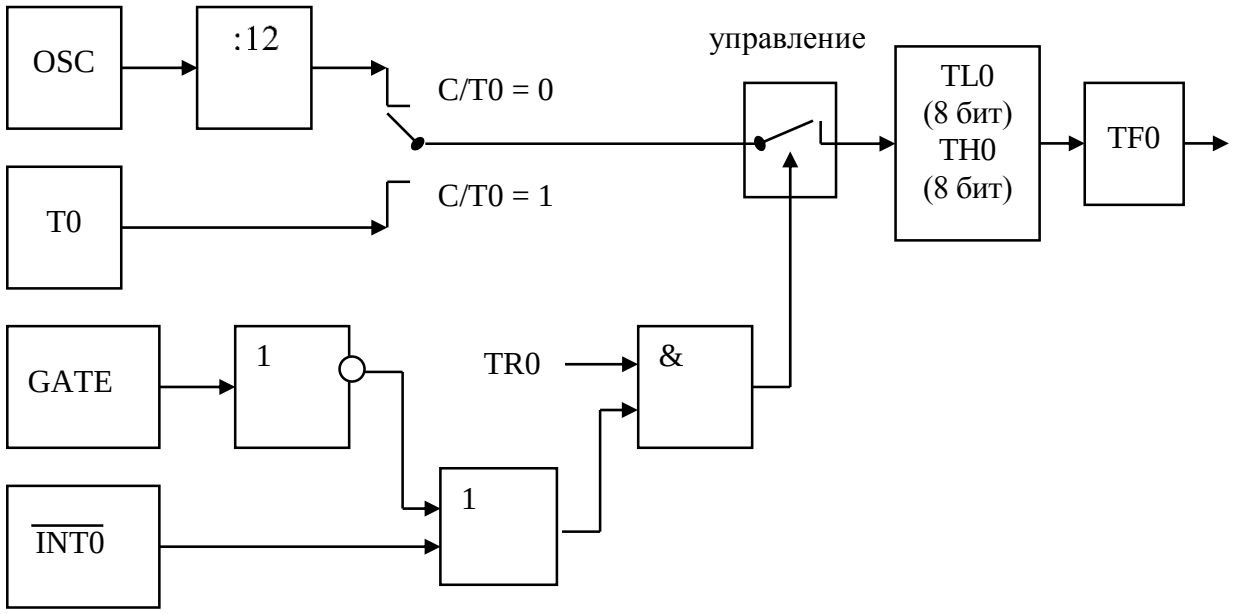

Рис. 3.2, б Логика работы ТС0 в режиме 1

Режим 2. Работа Т/С0 в режиме 2 показана на рис. 3.2, в.

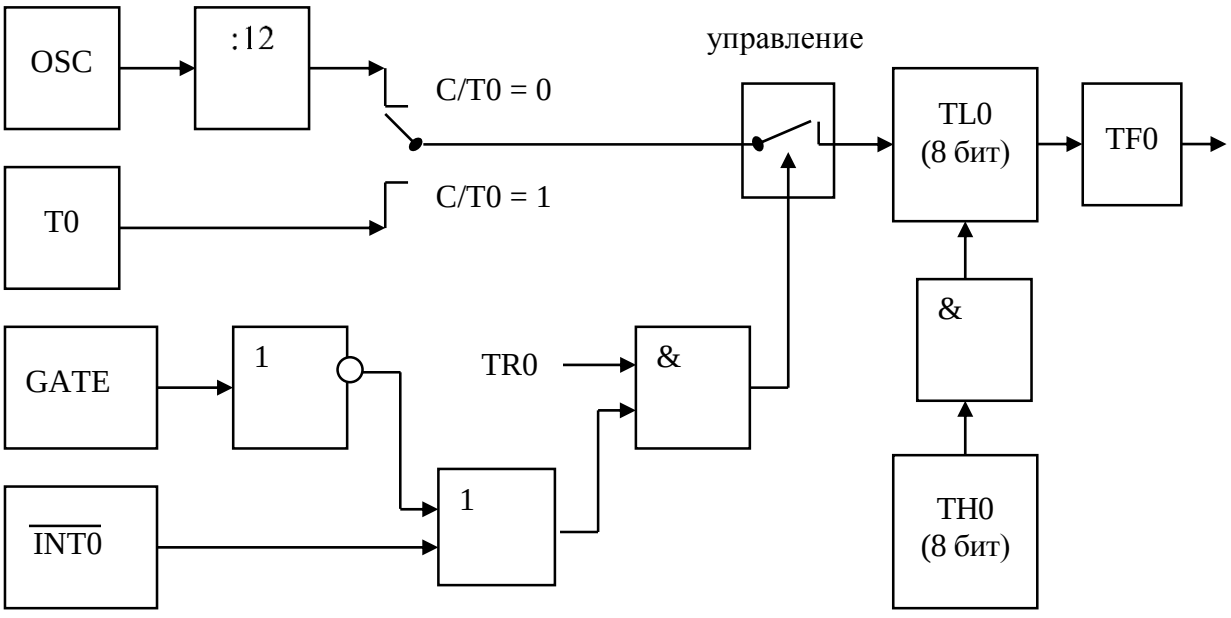

Рис. 3.2, в. Логика работы ТС0 в режиме 2

Переполнение (переход из состояния 11...1 в 00...0) 8-битного счетчика TL0 приводит к формированию флажка TF0 и автоматической загрузке в TL0 байта из регистра TH0 Т/С0, заданного предварительно программным способом.

При перезагрузке содержимое TH0 остается неизменным. В этом режиме Т/С0 и Т/С1 функционируют одинаково.

Режим 3. В режиме 3 Т/С1 и Т/С0 функционируют различно. Т/С1 сохраняет свое текущее содержимое, т. е. это аналогично сбросу управляющего бита TR1 регистра TCON в 0. Работа Т/С0 в режиме 3 показана на рис. 3.2, г. Счетчики TL0 и TH0 работают как два независимых 8-битных счетчика/таймера.

Работу TH0, который может выполнять только функции таймера, определяет только управляющий бит TR1. При этом TH0 использует флаг переполнения TF1. Работу TL0 определяют управляющие биты Т/С0 (C/T, GATE, TR0), входной сигнал INT0 и флаг переполнения TF0.

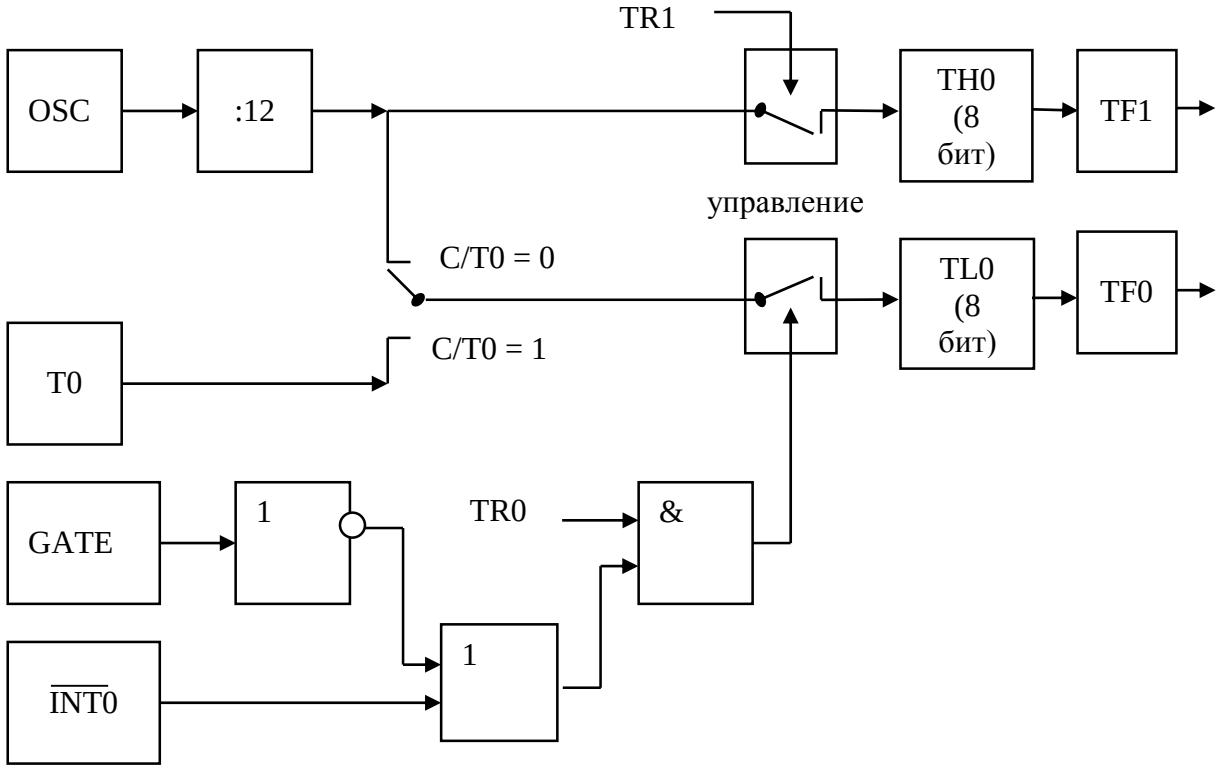

Рис. 3.2, г. Логика работы ТС0 в режиме 3

TCON.7	TCON.6	TCON.5	TCON.4	TCON.3	TCON.2	TCON.1	TCON.0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Рис. 3.3. Формат регистра управления TCON

Обратите внимание, что регистр TCON расположен в булевом пространстве и, соответственно, идентификатор в первой колонке табл. 3.2 является 8-разрядным адресом бита, непосредственно управляемым по командам булевого процессора.

Последующие версии ОМК семейства MCS-51, кроме двух, содержат третий таймер/счетчик Т/С2, расширяющий функциональные возможности ОМК,

который поддерживает режимы захвата, автозагрузки и генератора скорости приема/передачи последовательного канала.

Таблица 3.2

Управление таймерами/счетчиками

Бит	Позиция	Функция
TF1	TCON.7	Флаг переполнения таймера 1. Устанавливается аппаратно при переполнении Т/С1. Сбрасывается при обслуживании прерывания аппаратно
TR1	TCON.6	Бит управления таймера 1. Устанавливается и сбрасывается программой для пуска/останова
TF0	TCON.5	Флаг переполнения таймера. Устанавливается аппаратно, сбрасывается при обработке прерывания
TR0	TCON.4	Бит управления таймера 0. Устанавливается и сбрасывается программой пуска/останова Т/С
IE1	TCON.3	Флаг фронта прерывания 1. Устанавливается аппаратно при обнаружении среза внешнего сигнала прерывания INT1
IT1	TCON.2	Бит управления типом прерывания 1. Устанавливается/сбрасывается программно для спецификации запроса INT1 (срез/низкий уровень)
IE0	TCON.1	Флаг фронта прерывания 0. Устанавливается по срезу сигнала INT0, сбрасывается при обслуживании прерывания
IT0	TCON.0	Бит управления типом прерывания 0. Устанавливается/сбрасывается программно для спецификации запроса INT0 (срез/низкий уровень)

Т/С2 состоит из двух 8-битных регистров – TH2 и TL2. Выбор режимов и управление работой таймера/счетчика осуществляется посредством регистра T2CON. На рис. 3.4 приведен формат регистра T2CON, а в табл. 3.3 приведено назначение его разрядов.

T2CON.7	T2CON.6	T2CON.5	T2CON.4	T2CON.3	T2CON.2	T2CON.1	T2CON.0
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2

Рис. 3.4. Формат регистра управления T2CON

TF2 – флаг переполнения Т/С2. Сбрасывается программным способом. Флаг не устанавливается в режиме генератора приемопередатчика, если TCLK или RCLK установлены в «1».

EXF2 – флаг устанавливается при защелкивании информации в таймерных регистрах (режим захвата) или при перезагрузке, происходящих под воздействием перепада из 1 в 0 на выводе P1.1 в случае разрешения этих режимов (EXEN2 = 1).

Для запросов прерывания от Т/С2 в ОМК выделен один вектор прерываний, поэтому идентификация источника запроса должна выполняться программно, методом анализа состояния шестого и седьмого битов регистра

T2CON, где фиксируются флаги TF2 и EXF2. В подпрограмме обработки прерывания должны быть две соответствующие ветви для каждого из запросов, после выполнения обработки данный запрос должен быть сброшен программно.

Таблица 3.3

Управление таймером/счетчиком С/Т2

T2CON.0	CP/RL2	0 – автозагрузка, 1 – захват
T2CON.1	C/T2	0 – таймер, 1 – счетчик
T2CON.2	TR2	0 – останов СТ2, 1 – программный запуск
T2CON.3	EXEN2	Разрешение захвата/автозагрузки
T2CON.4	TCLK	0 – генератор скорости передачи СТ1, 1 – СТ2 – генератор скорости послед. канала в реж. 1 и 3
T2CON.5	RCLK	0 – генератор скорости приема СТ1, 1 – СТ2 – генератор скорости послед. канала в реж. 1 и 3
T2CON.6	EXF2	Флаг устанавливается при защелкивании информации в таймерных регистрах (режим захвата) или при автозагрузке в случае разрешения этих режимов (EXEN2=1). Сбрасывается программно
T2CON.7	TF2	Флаг переполнения Т/С2. Сбрасывается программно. Флаг не устанавливается если TCLK = 1 или RCLK = 1

Режим захвата выполняется при установке бита EXEN2 = 1 регистра T2CON, разрешающего поступление внешнего сигнала T2EX с вывода порта P1.1 и бита CP/RL2 = 1, задающего режим захвата. В этом режиме 16-разрядный счетчик инкрементируется основной тактовой частотой. При появлении на выводе порта P1.1 перепада из 1 в 0 происходит защелкивание текущего содержимого регистров счетчика TH2 и TL2 во вспомогательные регистры RCAP2H и RCAP2L соответственно. При этом формируется запрос на прерывание (бит EXF2 = 1 в регистре T2CON).

Режим автозагрузки инициируется также перепадом из 1 в 0 внешнего сигнала, поступающего на вывод P1.1. Поэтому для задания этого режима нужно разрешить поступление этого сигнала посредством установки бита EXEN2 = 1 регистра T2CON. Режим автозагрузки задается нулевым значением бита CP/RL2. Предварительно во вспомогательные регистры RCAP2H и RCAP2L записываются данные для загрузки в регистры счетчика TH2 и TL2, которая выполнится при поступлении перепада внешнего сигнала на вывод P1.1. При дальнейшем инкрементировании счетчика и каждом наступлении переполнения произойдет автоматическая перегрузка содержимого вспомогательных регистров в счетчики TH2 и TL2.

3.2. Задания

Вывод результата эксперимента осуществляется через порт P1 контроллера.

1. Разработать программу аппаратно-программной задержки, обеспечивающую наблюдение эффекта «бегущего огня» на индикаторных светодиодах. Вид и частота «бегущего огня» определены вариантом задания (номер бригады) в табл. 3.4.

Совместно с таймером/счетчиком при выполнении лабораторной работы необходимо использовать подпрограмму задержки. Сигнал прерывания TF0 из C/T0 вызывает модуль программной задержки, реализуемой с использованием соответствующего кода, загружаемого в регистр МК и декрементируемого после очередного сигнала прерывания из T/C0. Разрешение прерывания от T/C0 осуществляется установкой бита IE.1 в регистре масок прерывания IE и установкой бита IE.7 для общего разрешения прерывания (см. описание системы прерывания в лабораторной работе 4).

В режиме реального времени реализация задержки только посредством таймера при частоте тактирования микроконтроллера 6 Мгц для 16-битного T/C составляет около 130 мс, что недостаточно для визуального наблюдения за «бегущим огнем», используемым в качестве индикатора результатов эксперимента.

В режиме моделирования процессы в ОМК идут медленнее в сотни раз. Поэтому настройка C/T0 иная, чем в режиме реального времени.

Таблица 3.4

Варианты заданий реализации «бегущего огня»

Вариант задания	Вид «бегущего огня»	Время свечения светодиода/СИД/ в секундах
1	Слева-направо; горит 1 СИД	0,5
2	Слева-направо; горят 2 СИД	1,0
3	Справа-налево; горит 1 СИД	1,5
4	Справа-налево; горят 2 СИД	2,0
5	Слева-направо; горят 4 СИД	0,5
6	Справа-налево; горят 4 СИД	1,0
7	Сходящийся к центру; горят 2 СИД	1,5
8	Расходящийся от центра; горят 2 СИД	2,0

В соответствии с заданием определить длительность и количество вызовов подпрограммы задержки для режимов реального времени и моделирования.

Возврат из подпрограммы программной задержки обеспечивается командой RETI, восстанавливающей содержимое программного счетчика PC из

стека. В стек PC загружается автоматически по аппаратно сформированному коду команды LCALL. В PC при прерывании от T/C0 загружается код адреса 000BH, в котором находится команда перехода к подпрограмме обработки соответствующего прерывания.

При выполнении задания реализовать два варианта разрешения работы таймера/счетчика:

а) разрешение счета обеспечивается битом GATE, сбрасываемым в регистре TMOD;

б) разрешение счета обеспечивается внешним сигналом, поступающим на вход прерывания INT0 (в макете обозначен IO).

2. Изучить особенность работы T/C0 в режиме 2.

Разработать программу подсчета числа внешних сигналов (событий). Источником сигналов является внешний сигнал, поступающий на вход T0, режим работы T/C0 – режим 2. Для наблюдения выполнения программы загрузить в TL0 и TH0 одинаковые коды, например, 0F8H. Результатом выполнения программы является код числа сигналов, выведенный в P1.

Следующая ЛР 4 является развитием ЛР 3, что делает ее грамотно реализованной в системе прерываний. Поэтому допускается готовить и сдавать обе ЛР одновременно, рассматривая их как этапы отладки единой программы.

Содержание отчета

1. Титульный лист.
2. Задание на лабораторную работу.
3. Структурные схемы работы таймера/счетчика для изучаемых режимов.
4. Схемы и описания алгоритмов, а также листинги соответствующих хорошо комментированных программ на ассемблере.

Сделать выводы об эффективности использования таймера/счетчика в рассматриваемых экспериментах.

Лабораторная работа 4. ИЗУЧЕНИЕ СИСТЕМЫ ПРИОРИТЕТНЫХ ПРЕРЫВАНИЙ В МИКРОКОНТРОЛЛЕРЕ MCS-51

Цель работы:

1. Ознакомление с организацией системы прерываний в микроконтроллере.
2. Изучение приемов программирования и принципов построения программ обработки прерываний.

4.1. Методические указания

В микроконтроллере MCS-51 реализована приоритетная система прерываний, структура которой показана на рис. 4.1. Обслуживаются запросы на прерывание, поступающие от следующих источников прерывания:

1. Два внешних источника прерывания (запросы должны поступать от внешних устройств на входы INT0, INT1 микроконтроллера).

2. Таймеры/счетчики 0, 1 (при переполнении) и 2 (при переполнении или автозахвате/автозагрузке).

3. Порт последовательной передачи данных (при передаче или приеме очередного байта информации).

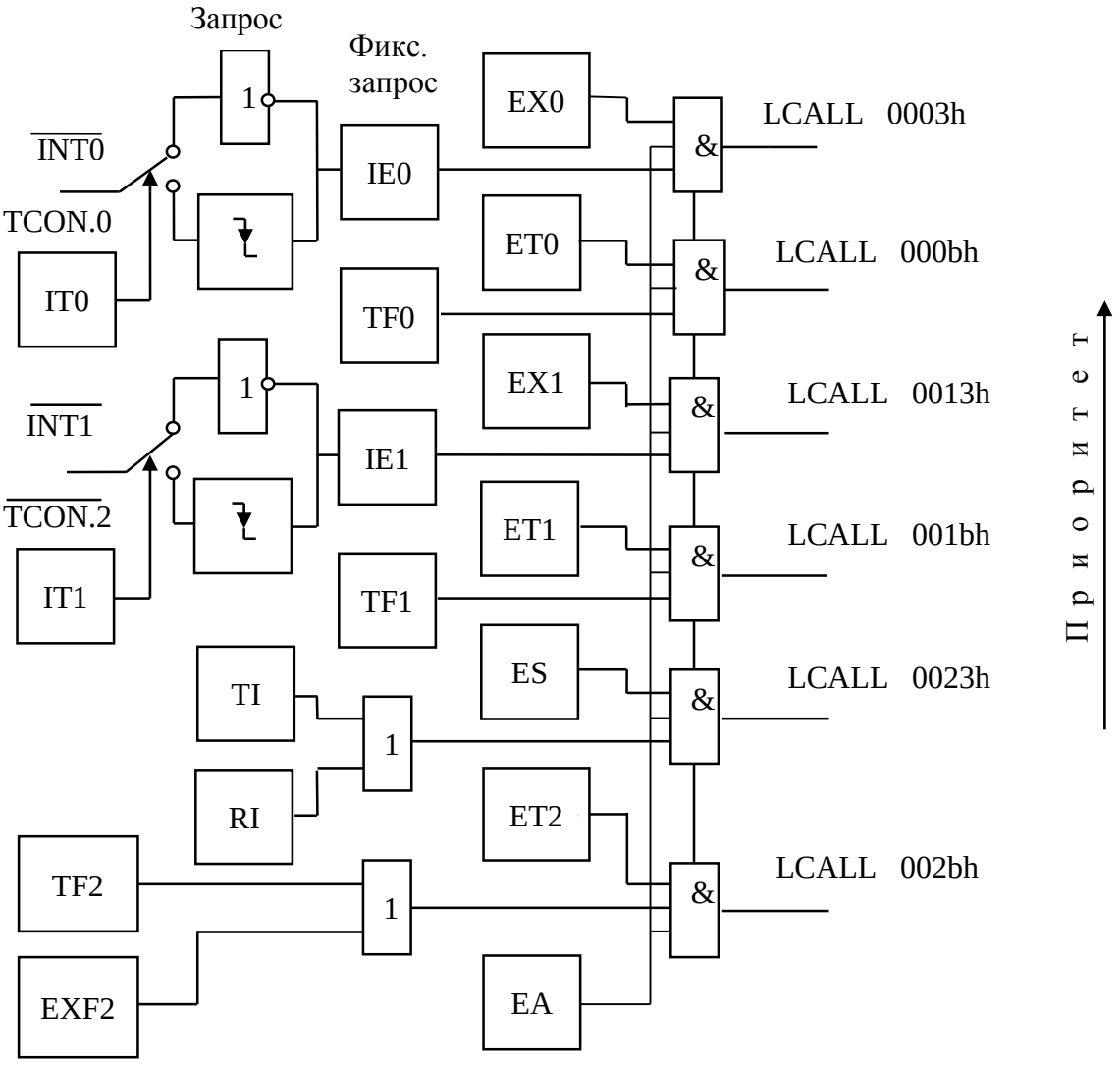

Рис. 4.1. Схема системы прерываний

Внешние прерывания от входов микроконтроллера INT0, INT1 могут быть вызваны или нулевым уровнем на входе, или срезом при переходе сигнала из 1 в 0, что

определяется значением бита управления в регистре TCON. От внешних прерываний устанавливаются флаги IE0, IE1 в регистре TCON, которые инициируют вызов соответствующей подпрограммы обработки запроса прерывания. Сброс этих флагов осуществляется аппаратно только в том случае, если прерывание было вызвано по срезу сигнала. Если прерывание вызвано уровнем сигнала прерывания, то сбросом флага IE управляет соответствующая подпрограмма обслуживания прерывания посредством воздействия на источник прерывания для снятия запроса.

Флаги запросов прерывания TF0, TF1 от таймеров 0 и 1 сбрасываются автоматически при передаче управления подпрограмме обработки прерывания.

Таймер 2 формирует два запроса прерывания с установкой соответствующих флагов в регистре управления таймером T2CON:

TF2 – флаг переполнения T/C2. Сбрасывается программным способом. Флаг не устанавливается в режиме генератора приемопередатчика, если TCLK или RCLK установлены в «1»;

EXF2 – флаг устанавливается при защелкивании информации в таймерных регистрах (режим захвата) или при перезагрузке, происходящих под воздействием перепада из 1 в 0 на выводе P1.1 в случае разрешения этих режимов (EXEN2 = 1).

Для запросов прерывания от T/C2 в ОМК выделен один вектор прерываний, поэтому идентификация источника запроса должна выполняться программно посредством анализа состояния шестого и седьмого битов регистра T2CON, где фиксируются флаги TF2 и EXF2. В подпрограмме обработки прерывания должны быть две соответствующие ветки для каждого из запросов, после выполнения обработки данный запрос должен быть сброшен программно.

Прерывания могут быть запрещены или вызваны пользователем, так как все указанные флаги программно доступны и могут быть установлены/сброшены.

В микроконтроллере имеются два регистра, управляющие разрешением прерываний (IE) и уровнем приоритетов (IP). Форматы этих регистров представлены на рис. 4.2, 4.3, а их описание – в табл. 4.1, 4.2. Чтобы прерывание от данного источника произошло, необходимо, во-первых, установить в 1 соответствующий бит разрешения прерывания в регистре IE и, во-вторых, установить в 1 бит общего разрешения прерываний EA. Система прерываний является двухуровневой, т. е. программа обработки прерывания низшего приоритета (0 в соответствующем бите регистра IP) может быть прервана программой

обработки прерывания высшего приоритета (1 в соответствующем бите регистра IP).

IE.7	IE.6	IE.5	IE.4	IE.3	IE.2	IE.1	IE.0
EA	–	ET2	ES	ET1	EX1	ET0	EX0

Рис. 4.2. Структура регистра разрешения прерываний IE

Таблица 4.1

Назначение битов регистра разрешения прерываний IE

Бит	Позиция	Название и назначение
EA	IE.7	Общее разрешение прерываний. Сбрасывается программно для запрета всех прерываний независимо от состояний остальных битов регистра IE
	IE.6	Не используются
ET2	IE.5	Разрешение прерывания от таймера/счетчика 2 от флага TF2 или EXF2
ES	IE.4	Разрешение прерывания от последовательного порта от флага TI или RI
ET1	IE.3	Разрешение прерывания от таймера/счетчика 1 от флага TF1
EX1	IE.2	Разрешение внешнего прерывания от входа INT1
ET0	IE.1	Разрешение прерывания от таймера/счетчика 0 от флага TF0
EX0	IE.0	Разрешение внешнего прерывания от входа INT0

Если установленный в регистре IP приоритет разных источников прерываний одинаков, то при одновременном возникновении запросов управление получает программа обработки прерывания с меньшим номером соответствующего разряда в регистре IE.

IP.7	IP.6	IP.5	IP.4	IP.3	IP.2	IP.1	IP.0
–	–	PT2	PS	PT1	PX1	PT0	PX0

Рис. 4.3. Структура регистра приоритетов прерываний IP

Таблица 4.2

Назначение битов регистра приоритетов прерываний IP

Бит	Позиция	Название и назначение
	IP.7, IP.6	Не используются
ET2	IP.5	Приоритет прерывания от таймера/счетчика 2 от флага TF2 или EXF2
ES	IP.4	Приоритет прерывания от последовательного порта от флага TI или RI
ET1	IP.3	Приоритет прерывания от таймера/счетчика 1 от флага TF1
EX1	IP.2	Приоритет внешнего прерывания от входа INT1
ET0	IP.1	Приоритет прерывания от таймера/счетчика 0 от флага TF0
EX0	IP.0	Приоритет внешнего прерывания от входа INT0

Идентификатор в первой колонке табл. 4.1 и 4.2 является 8-разрядным адресом бита, непосредственно управляемым по командам булевого процессора. Система прерываний микроконтроллера аппаратно формирует вызов (LCALL) соответствующей подпрограммы обслуживания, если она не заблокирована

одним из следующих условий:

– в данный момент обслуживается запрос прерывания равного или более высокого уровня приоритета;

– текущий машинный цикл не является последним в цикле выполняемой команды;

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

По аппаратно сформированному коду LCALL система прерывания загружает в стек только содержимое счетчика команд PC, а в счетчик команд затем загружается адрес вектора соответствующей подпрограммы обработки прерывания (см. рис. 4.1): 0003H для внешнего прерывания INT0 и 000BH – для прерывания от таймера TF0.

Подпрограммы обслуживания прерываний должны сохранить в стеке все регистры, которые использованы данной программой, командой PUSH . Необходимо, однако, отметить, что при использовании общих регистров R0–R7 для их сохранения достаточно переключить номер банка регистров в регистре PSW. После обработки прерывания сохраненные в стеке регистры должны быть восстановлены командой POP в обратном порядке. В конце программы обработки прерывания требуется использование команды RETI, обеспечивающей правильный возврат из прерывания.

4.2. Задания

1. Построить программу реализации «бегущего огня» согласно лабораторной работе 3 на основе программно-аппаратной задержки. При программировании регистров управления прерыванием обеспечить разрешение прерывания от T/C0 и INT0/. При этом реализовать следующее:

– внешний сигнал прерывания INT0 вызывает программу вывода в порт P1 «решетки», например, кода 55H.

– изучить отличия формирования сигнала прерывания INT0 по фронту и уровню. Замаскировать прерывание INT0, а затем и прерывание от TF0; объяснить поведение программы (отсутствие индикации).

2. Изучить режимы обработки запросов прерываний с различными приоритетами. Назначить сигналу INT0 высший приоритет, а сигналу прерывания TF0 от T/C0 – низший. Для наблюдаемости результата в программе обработке прерывания от TF0 вывести некоторый код в порт P1 и, не выходя из обработки преры-

вания, выполнять бесконечный цикл – этот цикл должен быть прерван по сигналу INT0 с выводом в порт P1 «решетки» 55H. Убедиться, что программа, запускаемая от таймера/счетчика 0, не прерывается, если назначить сигналу INT0 приоритет, равный или меньше приоритета сигнала прерывания TF0 от T/C0.

Содержание отчета

1. Титульный лист.
2. Задание на лабораторную работу.
3. Форматы и назначение управляющих слов, используемых при программировании системы прерывания микроконтроллера.
4. Схемы и описания алгоритмов, межпрограммные переходы при обработке прерываний и листинги хорошо комментированных программ.
5. Выводы по выполненным заданиям.

Лабораторная работа 5. ИЗУЧЕНИЕ РАБОТЫ ПОСЛЕДОВАТЕЛЬНОГО ПОРТА В МИКРОКОНТРОЛЛЕРЕ

Цель работы:

1. Ознакомление с некоторыми применениями канала последовательного ввода/вывода в микроконтроллере.
2. Изучение вопросов программирования и применения канала последовательного ввода/вывода в основных режимах его работы.

5.1 Методические указания

Обмен информацией между удаленными компьютерными устройствами осуществляется побайтно в последовательном коде с помощью высокоскоростного порта последовательной передачи информации, который в MCS-51 может работать в четырех режимах:

1. Синхронный обмен с фиксированной скоростью передачи (сдвиговый регистр).
2. Восьмиразрядный универсальный асинхронный приемопередатчик (УАПП) с переменной скоростью обмена.
3. Девятиразрядный УАПП с фиксированной скоростью обмена.
4. Девятиразрядный УАПП с переменной скоростью обмена.

На рис. 5.1 представлена структурная схема последовательного порта (ПП).

В состав ПП входят каналы передачи и приема информации и устройство управления (УУ) с генератором скорости передачи. Управление работой последовательного порта осуществляется через служебный регистр управления/статуса приемопередатчика SCON и бита SMOD регистра управления мощностью PCON.

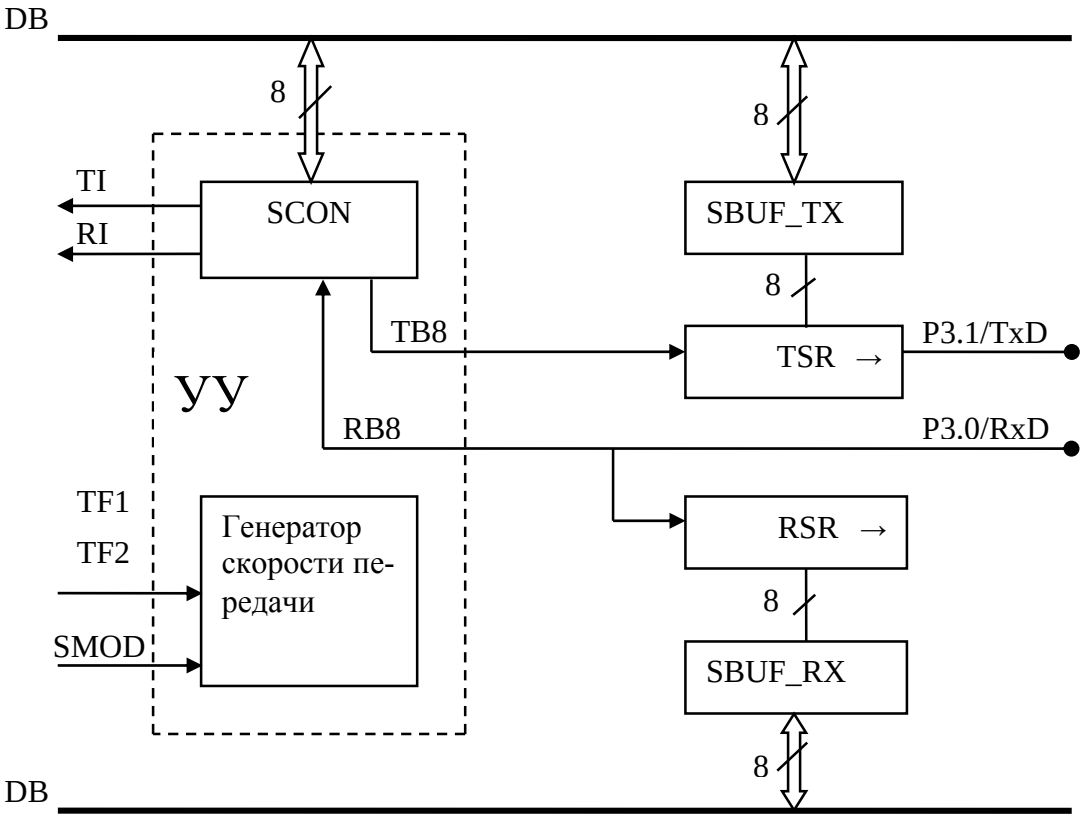

Рис. 5.1 Структурная схема последовательного порта

Основу канала передачи составляет регистр сдвига TSR, выполняющий преобразование передаваемых данных из параллельного кода в последовательный. Для передачи байт данных записывается в буферный регистр передающего канала SBUF_TX и, если регистр TSR завершил преобразование предыдущего байта, данные перезаписываются в регистр сдвига, и под управлением тактовых сигналов осуществляется побитовый сдвиг данных младшими разрядами вперед в линию связи через вывод P3.1/TxD микроконтроллера.

При приеме данных информационные биты поступают с линии связи через вывод P3.0/RxD микроконтроллера на последовательный вход регистра сдвига RSR, входящего в канал приема информации. После завершения приема всех битов, т. е. после завершения преобразования последовательных данных в параллельный код, информационная посылка записывается в буферный регистр канала

приема SBUF_RX и далее может быть считана через шину данных DB процессором. Порт позволяет начинать прием нового байта до того как из приемного регистра считан предыдущий байт. Однако если предыдущий байт еще не считан к моменту, когда закончится прием последующего, предыдущий байт будет потерян.

Задание режимов работы последовательного порта выполняется посредством регистра специального назначения SCON. В нем находятся флаги состояния, кроме того, в него записывается 9-й бит данных (TB8, RB8).

В табл. 5.1 приведены название и назначение битов регистра SCON.

Таб*лица 5.1*

Назначение битов регистра SCON

Бит	Позиция	Название и назначение
SM0 SM1	SCON.7 SCON.6	Флаги выбора режима работы универсального асинхронного приемо-передатчика (УАПП) Устанавливаются/сбрасываются программно: SM0 SM1 Режим работы 0 0 Сдвиговый регистр расширения ввода/вывода. 0 1 8-разрядный УАПП. Переменная скорость передачи. 1 0 9-разрядный УАПП. Фиксированная скорость передачи. 1 1 9-разрядный УАПП. Переменная скорость передачи
SM2	SCON.5	Бит управления режимом УАПП. Устанавливается программно для запрета приема кодов, для которых бит RB8 равен 0
REN	SCON.4	Флаг разрешения/приема. Устанавливается/сбрасывается программно для разрешения/запрета приема данных
TB8	SCON.3	Бит 8 в режиме передачи. Устанавливается/сбрасывается программно для задания состояния 9-го разряда данных в 9-разрядном УАПП
RB8	SCON.2	Бит 8 в режиме приема. Устанавливается/ сбрасывается аппаратно. Определяет состояние 9-го разряда данных в 9-разрядном УАПП
TI	SCON.1	Флаг прерывания от передатчика. Устанавливается аппаратно, когда байт данных передан. Сбрасывается программно после обнаружения прерывания
RI	SCON.0	Флаг прерывания от приемника. Устанавливается аппаратно, когда байт данных принят. Сбрасывается программно после обслуживания прерывания

После приема или передачи очередного байта информации автоматически устанавливается запрос на прерывание в блок обслуживания прерываний.

Последовательный порт является полнодуплексным, т. е. он позволяет одновременно работать на выдачу и на прием информации. Регистры приема/выдачи информации последовательного порта адресуются как один регистр специального назначения SBUF, несмотря на то, что физически они разделены.

Последовательный порт может работать в одном из четырех режимов:

синхронный обмен информацией (режим 0) и три режима асинхронного обмена (режимы 1, 2, 3).

Режим 0. Последовательная информация передается и принимается в синхронном режиме через вывод RxD 8-битными посылками. Через вывод TxD передаются тактовые импульсы. В данном режиме последовательный порт работает как сдвиговый регистр, т. е. информация записывается и считывается последовательно, синхронно с тактовыми импульсами. Скорость обмена фиксирована и составляет 1/12 частоты внешнего тактового генератора.

На рис. 5.2 приведена временная диаграмма работы последовательного порта (ПП) в режиме 0.

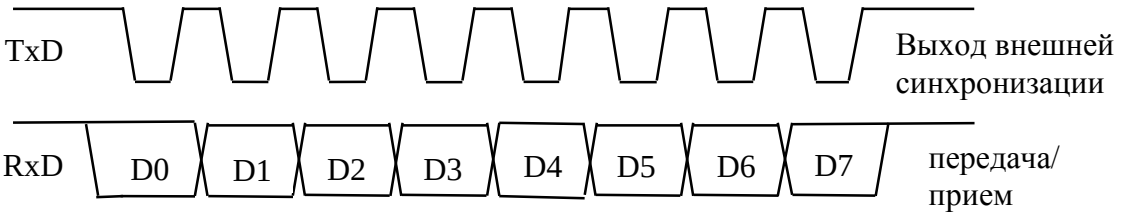

Рис 5.2. Работа ПП в режиме 0

Процесс передачи начинается, когда выполнена любая команда, использующая регистр SBUF в качестве приемника операнда. Команда «запись в SBUF» устанавливает единицу в девятом бите передающего сдвигового регистра и сообщает блоку управления TX о начале передачи. При этом на линию порта P3.1 выводится тактовый сигнал, а на линию порта P3.0 вводятся биты информационной посылки с последовательного выхода регистра TSR. После передачи всех битов информационной посылки в регистре SCON устанавливается в единицу разряд TI – флаг прерывания от передатчика, сигнализирующий процессору о завершении передачи байта данных.

Прием информации начинается при установке разрешения приема – флаг REN = 1 и отсутствии запроса прерывания от приемника – флаг RI = 0 в регистре SCON. При этом формируется разрешение прохождения тактовых сигналов с линии связи от передатчика информации через вывод порта P3.1 в блок синхронизации приемника. Под управлением этих тактовых сигналов выполняется прием информационных битов с линии связи через вывод порта P3.0 в регистр RSR. После приема всех разрядов информационной посылки устанавливается флаг прерывания RI в регистре SCON.

В режимах 1, 2, 3 выполняется асинхронный обмен информацией.

На рис. 5.3 приведен формат кадра асинхронной посылки.

Передача каждого байта начинается со старт-бита, сигнализирующего приемнику о начале посылки, за которым следуют биты данных и, возможно, дополнительный бит P, функциональное назначение которого определяется пользователем. Это может быть контрольный бит паритета или признак адресной информации при организации интерфейса с несколькими приемниками информации. Завершает посылку стоп-бит, гарантирующий паузу между посылками. Старт-бит следующего байта посылается в любой момент после стоп-бита, т. е. между передачами возможны паузы произвольной длительности. Для корректного обмена информацией передатчик и приемник должны быть настроены на одинаковую скорость обмена и на одинаковый формат кадра.

Рис.5.3. Формат кадра асинхронной посылки

Старт-бит, имеющий всегда значение логического 0, обеспечивает простой механизм синхронизации приемника по сигналу от передатчика. Внутренний генератор синхронизации приемника имеет тактовую частоту в 16 раз выше тактовой частоты передатчика и включается при появлении на линии приемника отрицательного фронта старт-бита. Через восемь тактов формируется первый сигнал, по которому осуществляется трехкратное считывание состояния линии приема, и по мажоритарному принципу определяется значение бита. Таким образом, считывание синхронизируется с серединой старт-бита. Если значение бита является ненулевым, то принят не старт-бит, а ложный сигнал, генератор синхронизации приемника выключается, и приемник продолжает следить за состоянием линии в ожидании перепада из единичного состояния в нулевое. Если подтверждается нулевое значение принятого бита, то он фиксируется в регистре сдвига RSR, и приемник через 16 тактов синхросигнала формирует следующий сигнал для приема очередного бита, считывание которого осуществляется также в середине. Процесс продолжается до тех пор, пока не будут приняты все биты кадра. В идеале сигналы считывания располагаются в середине битовых интервалов, что позволяет принимать данные и при незначительном рассогласовании скоростей приемника и передатчика.

Очевидно, что при передаче восьми бит данных, одного контрольного и

одного стоп-бита, предельно допустимое рассогласование скоростей, при котором данные будут распознаны верно, не может превышать 5 %.

Режим 1. В обмене участвуют 10 бит, которые либо передаются (через TxD), либо принимаются (через RxD) в такой последовательности: старт-бит (логический «0»), 8 бит данных (младший бит первым) и стоп-бит (логическая «1»).

Формат информационного кадра в режиме 1 представлен на рис. 5.4.

При приеме стоп-бит заносится в разряд RB8 регистра SCON и может быть проанализирован программно. Если значение этого бита нулевое, то прием информации выполнен не верно, и имеет место ошибка «формата кадра».

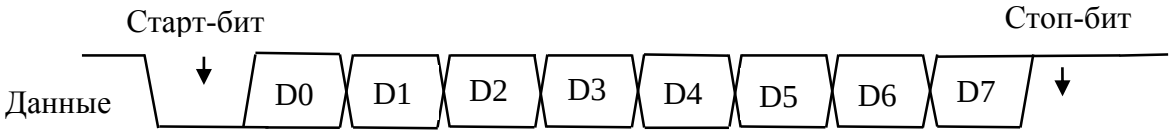

Рис.5.4. Формат кадра ПП в режиме 1

Скорость обмена в этом режиме задается пользователем. Для управления скоростью обмена используется таймер/счетчик 1. Передача начинается, когда выполнена какая-либо команда, использующая регистр SBUF в качестве приемника операнда. После появления сигнала «запись в SBUF», в девятый бит передающего сдвигового регистра загружается «1» (стоп-бит). После передачи в линию всех битов посылки в разряд TI регистра SCON записывается единичный запрос на прерывание от передатчика.

Скорость обмена данными в режиме 1 переменная и задается с помощью таймера/счетчика 1, причем определяется скорость обмена по следующей формуле:

Скорость обмена = (скорость переполнения таймера/счетчика 1) / *n*,

где *n* = 32, если SMOD = 0 (SMOD – старший бит регистра специального назначения PCON); *n* = 16, если SMOD = 1 (двойная скорость приема/передачи).

Например, если таймер/счетчик 1 используется в режиме автоперезагрузки (TMOD.5 = 1, TMOD.4 = 0), он должен быть запущен, а прерывания по переполнению должны быть запрещены (IE.3 = 0). Тогда скорость обмена зависит от значения, перезагружаемого в таймер.

Скорость переполнения таймера равна:

V = (скорость счета таймера) / [256 – (TH1)].

Для обеспечения медленной скорости обмена таймер/счетчик 1 может быть

использован в режиме 1 (TMOD.5 = 0, TMOD.4 = 1) в качестве 16-разрядного счетчика. Тогда должны быть разрешены запросы на прерывание по переполнению (IE.3 = 1), и программным способом должна обеспечиваться перезагрузка таймера. В любом случае, если разряд C/T в регистре TMOD для таймера/счетчика 1 установлен в 0, то скорость счета таймера/счетчика составляет 1/12 частоты внешнего тактового генератора. Если C/T = 1, то скорость счета задается внешним генератором, подключаемым к входу T1 микросхемы.

Режимы 2 и 3. В режимах 2 и 3 обмен информацией выполняется одиннадцатиразрядными посылками, причем в режиме 2 обмен с фиксированной скоростью передачи, равной OSC/32 или OSC/64, а в режиме 3 с переменной скоростью передачи. Одиннадцать бит передаются (через TxD) и принимаются (через RxD) в следующей последовательности: старт-бит (логический «0»), восемь бит данных (младший бит первым), программируемый девятый бит данных и стоп-бит (логическая «1»). Формат кадра представлен на рис. 5.5.

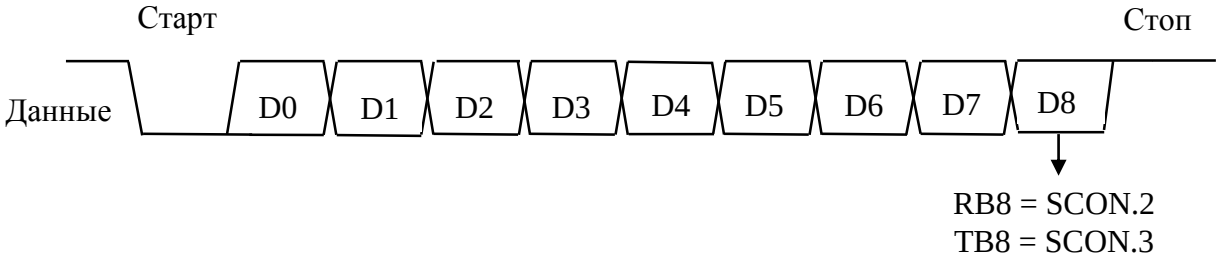

Рис. 5.5 Формат кадра в режимах 2 и 3

В режиме передачи девятый бит данных (размещается в разряде TB8) может иметь значение «0» или «1». В режиме приема девятый бит данных поступает в разряд RB8 регистра SCON. Скорость обмена в режиме 2 составляет 1/64 или 1/32 от частоты внешнего тактового генератора и зависит от состояния бита SMOD (старший бит в регистре специального назначения PCON). Если SMOD = 0 (устанавливается по сбросу микроконтроллера), то скорость обмена составляет 1/64 частоты внешнего тактового генератора. Если SMOD = 1, то скорость обмена составляет 1/32 частоты тактового генератора. В режиме 3 скорость обмена переменная и задается с помощью таймера/счетчика 1 также, как и в режиме 1. Приемная часть является такой же, как в режиме 1. Передающая часть отличается от режима 1 только наличием девятого бита в сдвиговом регистре передачи.

Передача начинается после выполнения какой-либо команды, использующей регистр SBUF в качестве приемника операнда. После поступления сигнала

«запись в SBUF», бит TB8 регистра SCON загружается в девятый разряд сдвигового регистра передачи, и устанавливается запрос на передачу в блоке управления. Процесс передачи, также как в режиме 1, синхронизируется 16-разрядным счетчиком-делителем. Прием начинается после обнаружения перепада из «1» в «0» на входе RxD. Процесс приема данных в режимах 2 и 3 аналогичен приему в режиме 1. Девятый бит данных при приеме записывается в разряд RB8 регистра SCON.

Для целей лабораторной работы организация процесса передачи информации через последовательный канал заключается в программировании передачи в режиме 1 и скорости передачи по частоте переполнения T/C1, настроенному в режим 2. Для отображения процесса передачи в симуляторе следует вывести окно терминала Terminal Window (Serial I/O) во вкладке View. После конфигурации последовательного порта в нем будет указана частота приёма-передачи. Данные, вводимые в этом окне, передаются по ПП в микроконтроллер, передаваемые ПП данные отображаются в окне терминала. Следует учесть, что введенные через ПП данные не будут отображены в окне терминала, но должны появиться в памяти микроконтроллера; для лучшей наблюдаемости процесса можно их сразу выводить в терминал.

Задания

1. Составить программу, обеспечивающую вывод массива символьной информации, хранящегося в ВПД (внешняя память данных) через канал последовательного ввода/вывода с использованием сигналов квитирования (готовность передатчика, готовность приемника). Для формирования сигналов квитирования использовать линии P3.4, P3.5 порта P3. Скорость вывода 1200 бод.

2. Составить программу, обеспечивающую ввод массива символьной информации через канал последовательного ввода/вывода с использованием сигналов квитирования (готовность передатчика, готовность приемника) и запись его в ВПД. Для формирования сигналов квитирования использовать линии порта P3.4, P3.5 порта P3. Скорость ввода 2400 бод.

3. Составить программу, обеспечивающую ввод массива символьной информации через канал последовательного ввода/вывода с использованием сигналов квитирования и механизма прерываний последовательного порта и запись его в ВПД. В качестве фоновой использовать любую циклическую программу.

Три пункта задания рассматриваются как этапы отладки программы ПП. Рекомендуется демонстрация всех пунктов в единой программе.

Содержание отчета

1. Титульный лист.

2. Задание на лабораторную работу.

3. Структурная схема блока последовательного интерфейса и описание системы прерываний, относящейся к последовательному каналу.

4. Схемы и описания алгоритмов программ, а также листинги соответствующих хорошо комментированных программ на ассемблере.

5. Выводы по выполненным заданиям.

Лабораторная работа 6. ИЗУЧЕНИЕ РАБОТЫ АНАЛОГО-ЦИФРОВОГО ПРЕОБРАЗОВАТЕЛЯ В МИКРОКОНТРОЛЛЕРЕ

Цель работы:

1. Ознакомление со структурой блока аналого-цифрового преобразования (АЦП) в микроконтроллере ADuC812.

2. Изучение вопросов программирования и применения блока аналого-цифрового преобразования в основных режимах его работы.

6.1. Методические указания

Блок АЦП в микроконтроллере AduC812 включает в себя восьмиканальный пятимикросекундный преобразователь аналогового сигнала в цифровой с однополярным питанием, многоканальный мультиплексор, устройство выборки-хранения (УВХ), источник опорного напряжения (ИОН), систему калибровок.

Компоненты блока управляются через три интерфейсных SFR-регистра: ADCCON1, ADCCON2, ADCCON3.

Регистр ADCCON1 управляет преобразованием, временем переключения, режимами преобразования и потреблением устройства. На рис. 6.1 представлена структура, а в табл. 6.1 назначение разрядов регистра ADCCON1.

MD1	MD0	CK1	CK0	AQ1	AQ0	T2C	EXC

Рис. 6.1. Управляющий регистр ADCCON1

Адрес ADCCON1 EFH

Значение при включении питания 20H

Битовой адресации нет.

Однократный или повторяющийся режимы преобразования могут выполняться программно или подачей внешнего сигнала запуска преобразования на

контакт 23 (CONVST). Для инициирования повторяющегося процесса преобразования можно использовать сигналы таймера 2. АЦП можно установить в режим передачи данных по каналу прямого доступа к памяти – ПДП (DMA). При этом блок повторяет циклы преобразования и посылает результаты во внешнюю память данных (RAM), минуя процессор.

Таблица 6.1

Содержание регистра ADCCON1

Расположение бит	Мнемоника	Описание
ADCCON1.7 ADCCON1.6	MD1 MD0	Биты выбора режима работы АЦП: MD1 MD0 Режим АЦП 0 0 Дежурный 0 1 Нормальный 1 0 Дежурный, если не выполняется цикл преобразования 1 1 Холостой, если не выполняется цикл преобразования
ADCCON1.5 ADCCON1.4	CK1 CK0	Биты деления тактовой частоты задают коэффициент деления основной частоты микропроцессора для получения тактовой частоты АЦП: * CK1 CK0 Делитель для MCLK 0 0 1 0 1 2 1 0 4 1 1 8
ADCCON1.3 ADCCON1.2	AQ1 AQ0	Биты задержки переключения выбирают время, необходимое для перезарядки УВХ при переключении мультиплексора: ** AQ1 AQ0 Число тактов задержки запуска АЦП 0 0 1 0 1 2 1 0 3 1 1 4
ADCCON1.1	T2C	Бит запуска преобразования от Таймера 2. Если бит установлен, то сигнал переполнения Таймера 2 используется для запуска АЦП
ADCCON1.0	EXC	Бит разрешения внешнего запуска. Если бит установлен, то контакт 23 (CONVST) будет использоваться для приема сигнала запуска АЦП (активный низкий, длительность не менее 100 нс)

* Цикл преобразования АЦП занимает 5 мкс в дополнение к числу тактов переключения, задаваемых битами AQ0, AQ1.

** При импедансе источника входного сигнала менее 8 Ком AQ1 = AQ0 = 0, т.е. AQ = 1. Иначе задержку увеличивают до 2–4 тактов.

Регистр ADCCON2 управляет выбором номера канала и режимами преобразования. На рис. 6.2 представлена структура, а в табл. 6.2 распределение разрядов регистра ADCCON2.

ADCI	DMA	CCONV	SCONV	CS3	CS2	CS1	CS0

Рис. 6.2. Управляющий регистр ADCCON2

Адрес ADCCON2 D8H

Значение при включении питания 00H

Битовая адресация есть.

Таблица 6.2

Содержание регистра ADCCON2

Расположение бит	Бит	Описание
ADCCON2.7	ADCI	Бит прерывания АЦП. Устанавливается аппаратно по окончании однократного цикла преобразования АЦП или по окончании передачи блока в режиме ПДП. ADCI очищается аппаратно при переходе по вектору на процедуру обслуживания прерывания
ADCCON2.6	DMA	Бит разрешения режима ПДП. Устанавливается пользователем для начала операции ПДП со стороны АЦП
ADCCON2.5	CCONV	Бит циклического преобразования. Устанавливается пользователем для задания режима непрерывного циклического преобразования АЦП. В этом режиме АЦП выполняет преобразование в соответствии с типом синхронизации и конфигурацией каналов, выбранными в других SFR
ADCCON2.4	SCONV	Бит запуска однократного преобразования. Устанавливается пользователем для однократного запуска АЦП. Бит сбрасывается автоматически по завершении преобразования
ADCCON2.3 ADCCON2.2 ADCCON2.1 ADCCON2.0	CS3 CS2 CS1 CS0	Биты выбора входных каналов (CS3 CS0). Преобразование будет выполняться для канала, номер которого указан данными битами. В режиме ПДП выбор номера канала осуществляется из ID-канала, записанного во внешней памяти: CS3 CS2 CS1 CS0 0 n2 n1 n0 Номер входного канала (n2n1n0) 1 0 0 0 Температурный сенсор (внутренний) 1 * * * Другие комбинации 1 1 1 1 Останов ПДП

Регистр ADCCON3 дает прикладным программам информацию о занятости АЦП. В табл. 6.3 представлено распределение разрядов регистра ADCCON3.

Адрес ADCCON3 F5H

Значение при включении питания 00H

Битовой адресации нет.

Таблица 6.3

Содержание регистра ADCCON3

Расположение бит	Мнемо–ника	Описание
ADCCON3.7	BUSY	Бит занятости АЦП (только для чтения). Устанавливается на время преобразования или калибровки АЦП. Автоматически очищается по завершению циклов преобразования или калибровки
ADCCON3.6 … ADCCON3.0	RSVD	Биты ADCCON3.6–ADCCON3.0 зарезервированы. Они считываются с нулевым значением и их следует записывать только нулями

После настройки и запуска АЦП выполняется преобразование и запись результата, представляющего собой 12-разрядный код, в регистры ADCDATA. Формат слова результата приведен на рис 6.3.

ADCDATAH

N3	N2	N1	N0	D11	D10	D9	D8

4 разряда – номер канала Старшие 4 разряда результата

ADCDATAL

D7	D6	D5	D4	D3	D2	D1	D0

Младший байт результата

Рис. 6.3. Результат аналого-цифрового преобразования

Выполнение аналого-цифрового преобразования в циклическом режиме с максимальной скоростью может быть реализовано с использованием режима *прямого доступа к памяти* (ПДП). Режим ПДП включается битом разрешения ПДП (ADCCON2.6). Результат каждого преобразования записывается во внешнюю статическую память, минуя микропроцессорное ядро. До включения режима ПДП необходимо разметить внешнюю память, в которую будут записываться данные. Разметка состоит в записи идентификаторов номеров каналов ID (четыре старшие разряда). Пример разметки показан на рис. 6.4. Для каждого преобразования во внешней памяти планируют два байта: в старшей тетраде младшего байта записывается номер преобразуемого канала, а оставшееся место резервируется для последующей записи результата преобразования. Причем старшая тетрада результата записывается сразу после номера канала, а младший байт результата – в следующий байт.

После разметки стартовый адрес начала размеченной таблицы (0 для примера на рис. 6.4) записывается в регистры DMAP, DMAH и DMAL. Три байта

стартового адреса следует записывать в следующем порядке: DMAL, DMAH и DMAP. Конец таблицы ПДП обозначается записью ≪1111≫ в поле выбора канала. Время преобразования устанавливается через SFR ADCCON1. Теперь для запуска ПДП и передачи результатов в последовательные ячейки внешней памяти можно установить бит разрешения (ADCCON2.6, DMA). Окончание ПДП преобразования устанавливается битом прерывания АЦП в ADCCON2.7. По окончанию ПДП внешняя память данных окажется загруженной новыми результатами работы АЦП согласно разметке (см. рис. 6.4).

Адрес	Байт								Содержание
00000AH	1	1	1	1					Команда СТОП ПДП
									Младший байт результата преобразования канала 3
	0	0	1	1					Повторить преобразование канала 3
									Младший байт результата преобразования канала 3
	0	0	1	1					Преобразовать канал 3, старшая тетрада результата
									Младший байт результата преобразования температурного сенсора
	1	0	0	0					Преобразовать температурный сенсор
									Младший байт результата преобразования канала 5
	0	1	0	1					Преобразовать канал 5, старшая тетрада результата
000001H									Младший байт результата преобразования канала 2
000000H	0	0	1	0					Преобразовать канал 2, старшая тетрада результата

Рис. 6.4. Разметка внешней памяти в режиме ПДП

Для моделирования процесса аналогово-цифрового преобразования в симуляторе следует подготовить *файл результатов преобразования* и указать его имя в окне Configutation/ADC/DACConfigutation в качестве входного файла. Пример такого файла можно найти в TEST\ADCDATA.ADC. Файл представляет собой множество 16-битных результатов преобразования, каждый из которых содержит старшую тетраду результата и младший байт результата; 4 старших разряда не используются и содержат нули. Всякий раз, когда происходит преобразование в АЦП-блоке (независимо от способа преобразования и номера канала), реально считывается следующее значение из этого файла и используется как цифровой результат преобразования.

Задания

1. Составить программы, обеспечивающие ввод аналоговой информации последовательно по каждому из восьми каналов (по одному отсчету). Найти максимальное и минимальное значения. Максимальное значение вывести в порты P0 и P1 (P1.0 – P1.3), минимальное – в порты P2 и P1 (P1.4 – P1.7). Запуск АЦП выполнять:

а) программно по окончании предыдущего преобразования в режиме слежения;

б) программно по прерыванию АЦП;

в) внешним сигналом.

Блок АЦП не входит в стандарт микроконтроллера MCS-51, а является расширением. Поэтому для подключения регистров специальных функций АЦП необходимо включить в программу файл MOD812.

2. Составить программу, обеспечивающую ввод аналоговой информации последовательно по трем каналам (15 отсчетов) в режиме прерывания АЦП. Запуск преобразования выполнять с помощью таймера. Обосновать выбор частоты запуска АЦП. Выполнить обработку информации по алгоритму, представленному в п. 4 лабораторной работы 1.

3. Составить программу, обеспечивающую ввод аналоговой информации по одному из каналов (32 отсчета), и запись результата преобразования во внешнюю память данных в режиме прямого доступа.

Содержание отчета

1. Титульный лист.

2. Задание на лабораторную работу.

3. Форматы и назначение управляющих слов, используемых при программировании АЦП, и описание системы прерываний, относящейся к блоку АЦП.

4. Схемы и описания алгоритмов программ, а также листинги соответствующих хорошо комментированных программ на ассемблере.

5. Выводы по выполненным заданиям.

Список использованной литературы

1. Зуев И. С., Петров Г. А. Проектирование микроконтроллерной системы управления: метод. указания к курсовому проекту по дисциплине «Микропроцессорные системы». СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2008.

2. Щелкунов Н. Н., Дианов А. П. Микропроцессорные средства и системы. М.: Радио и связь, 1989.

СОДЕРЖАНИЕ

Головина Людмила Константиновна
Зуев Игорь Станиславович
Петров Геннадий Алексеевич

Проектирование микроконтроллерной системы управления

Учебно-методическое пособие

Редактор О. Р. Крумина

Подписано в печать 00.00.00. Формат 60×84 1/16.
Печать цифровая. Печ. л. 2,75.
Гарнитура «Times New Roman». Тираж 65 экз. Заказ .

Издательство СПбГЭТУ «ЛЭТИ»
197022, С.-Петербург, ул. Проф. Попова, 5Ф