Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ
И РАДИОЭЛЕКТРОНИКИ

Кафедра электронной техники и технологии

ЛАБОРАТОРНЫЕ РАБОТЫ
по дисциплине

«МИКРОПРОЦЕССОРЫ В СРЕДСТВАХ МЕДИЦИНСКОЙ ЭЛЕКТРОНИКИ

для студентов специальности
«Медицинская электроника»

Минск 2002

УДК 621.396.66 (075.8)
ББК 52 я 73
Л 12

Авторы: А.Н. Осипов, В.М. Бондарик, С.К. Дик, С.О. Рудаков

Л 12 Лабораторные работы по дисциплине "Микропроцессоры в средствах медицинской электроники" для студентов специальности "Медицинская электроника" / А.Н. Осипов, В.М. Бондарик, С.К. Дик, С.О. Рудаков. - Мн.: БГУИР, 2002. – 37 с.: ил.

ISBN 985-444-376-0.

Лабораторные работы составлены в соответствии с программой дисциплины "Микропроцессоры в средствах медицинской электроники" и включают изучение принципов реализации средств медицинской электроники на основе микроконтроллеров серии.

Предназначены для закрепления и углубления теоретических знаний, полученных на лекциях и в процессе самостоятельного изучения дисциплин, приобретения практических навыков в области применения микроконтроллеров в средствах медицинской электроники.

УДК 621.396.66 (075.8)
ББК 52 я 73

ISBN 985-444-376-0

Лабораторная работа № 1

ВНУТРЕННЯЯ СТРУКТУРА МИКРОКОНТРОЛЛЕРА СЕРИИ 8051

1. Цель работы

Изучение структурной организации контроллера, способов адресации, назначения и функционирования блоков обработки биомедицинской информации. Выработка навыков реализации команд микропроцессорного устройства для обработки биомедицинской информации.

2. Теоретические сведения

Основу **структурной схемы микроконтроллера** образует внутренняя двунаправленная 8-битная шина, которая связывает между собой все основные узлы и устройства: резидентную память программ и данных, арифметико-логическое устройство (АЛУ), блок регистров специальных функций, устройство управления и порты ввода/вывода (рис. 1.1).

8-битное ***АЛУ*** может выполнять:

- арифметические операции сложения, вычитания, умножения и деления;
- логические операции И, ИЛИ, исключающее ИЛИ;
- операции циклического сдвига, сброса, инвертирования и т.п.

В АЛУ имеются программно-недоступные регистры Т1 и Т2, предназначенные для временного хранения операндов, схема десятичной коррекции и схема формирования признаков.

Важной особенностью АЛУ является его способность оперировать не только байтами, но и битами. Отдельно программно-доступные биты могут быть установлены, сброшены, инвертированы, переданы, проверены и использованы в логических операциях. Для управления объектами часто применяются алгоритмы, содержащие операции над входными и выходными булевскими переменными (истина/ложь), реализация которых средствами обычных микропроцессоров сопряжена с определёнными трудностями.

Таким образом, АЛУ может оперировать четырьмя типами информационных объектов: булевое (1 бит), цифровыми (4 бита), байтными (8 бит) и адресными (16 бит). В АЛУ выполняется 51 различная операция пересылки или преобразования этих данных. Так как используется 11 режимов адресации (7

для данных и 4 для адресов), то путём комбинирования «операция/режим адресации» базовое число команд 111 расширяется до 255 из 256 возможных при однобайтном коде операции.

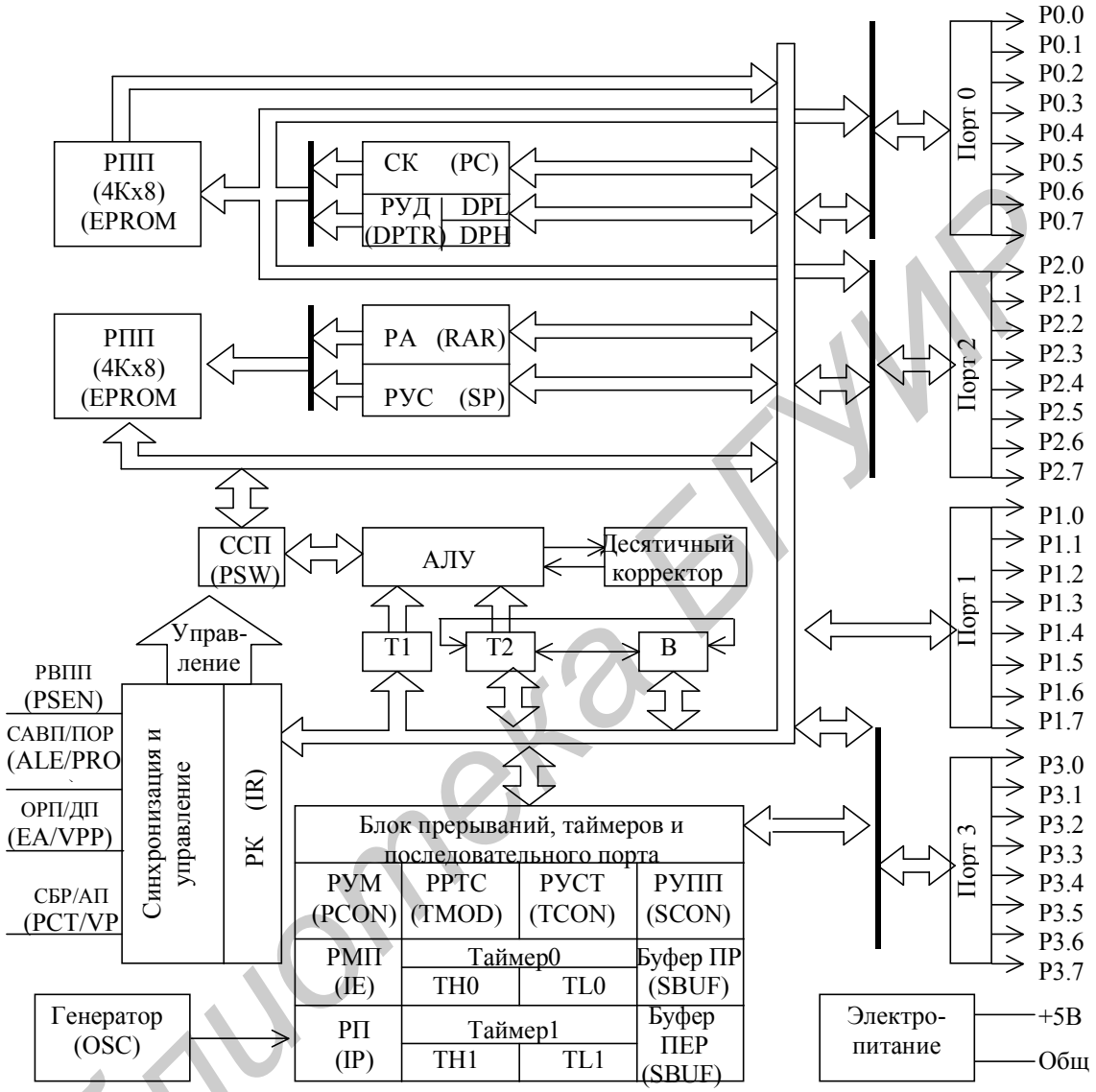


Рис. 1.1. Структурная схема микроконтроллера КМ1816ВЕ751

Резидентная память. Память программ и память данных, размещённые на кристалле микроконтроллера, физически и логически разделены, имеют различные механизмы адресации, работают под управлением различных сигналов и выполняют различные функции.

Память программ (ПЗУ или Flash) имеет ёмкость 4 Кб и предназначена для хранения команд, констант, управляющих слов инициализации, таблиц перекодировки входных и выходных переменных и т.п. Резидентная память программ (РПП) имеет 16-битную шину адреса, через которую обеспечивается доступ из счётчика команд (СК) или из регистра - указателя данных (РУД). По-

следний выполняет функции базового регистра при косвенных переходах по программе или используется в командах, оперирующих таблицами.

Память данных (ОЗУ) предназначена для хранения переменных в процессе выполнения прикладной программы, адресуется одним байтом и имеет ёмкость 128 байт. Кроме того, к адресному пространству резидентной памяти данных (РПД) примыкают адреса регистров специальных функций (РСФ), которые перечислены в табл. 1.1

Таблица 1.1

Блок регистров специальных функций

Символ	Наименование	Адрес
* ACC	Аккумулятор	0E0H
* B	Регистр - расширитель аккумулятора	0F0H
* PSW	Слово состояния программы	0D0H
SP	Регистр - указатель стека	81H
DPTR	Регистр - указатель данных (DPH)	83H
	Регистр - указатель данных (DPL)	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 таймер/счётчик	89H
* TCON	Регистр управления/статуса счётчика	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 приёмопередатчиком	98H
SBUF	Буфер приёмопередатчика	99H
PCON	Регистр управления мощностью	87H

* - регистры, допускающие адресацию отдельных бит.

РПП и РПД могут быть расширены до 64 Кб путём подключения внешних БИС.

Аккумулятор и слово состояния программы (ССП). Аккумулятор является источником операнда и местом фиксации результата при выполнении арифметических, логических операций и ряда операций передачи данных. Кроме того, только с использованием аккумулятора могут быть выполнены операции сдвигов, проверка на нуль, формирование флага паритета и т.п.

При выполнении многих команд в АЛУ формируется ряд признаков операций (флагов), которые фиксируются в регистре ССП. ***Флаг переноса*** (C) устанавливается и сбрасывается аппаратными средствами или программой при выполнении арифметических и логических операций. ***Флаг вспомогательного переноса*** (AC) устанавливается и сбрасывается только аппаратными средствами при выполнении команд сложения и вычитания и сигнализирует о переносе или заёме в бите 3 (старшем). ***Флаг 0*** (F0) – флаг состояния, определяемый пользователем. ***Выбор банка регистров*** (RS0, RS1) устанавливается и сбрасывается программой для выбора рабочего банка регистров. ***Флаг переполнения*** (OV) устанавливается и сбрасывается аппаратно при выполнении арифметических операций. **Флаг паритета** (P) устанавливается и сбрасывается аппаратно в каждом цикле команды и фиксирует нечётное/чётное число единичных бит в аккумуляторе, т.е. выполняет контроль по чётности.

Наиболее «активным» флагом ССП является флаг переноса, который принимает участие и модифицируется в процессе выполнения множества операций, включая сложение, вычитание и сдвиги. Кроме того, флаг переноса (C) выполняет функции «булевого аккумулятора» в командах, манипулирующих с битами. Флаг переполнения (OV) фиксирует арифметическое переполнение при операциях над целыми числами со знаком и делает возможным использование арифметики в дополнительных кодах. АЛУ не управляет флагами селекции банка регистров (RS0, RS1), и их значение полностью определяется прикладной программой и используется для выбора одного из четырёх регистровых банков.

Регистры-указатели. Содержимое 8-битного ***регистра - указателя стека*** (РУС) инкрементируется прежде, чем данные запоминаются в стеке в ходе выполнения команд PUSH и CALL. Содержимое РУС декрементируется после выполнения команд POP и RET. Подобный способ адресации элементов стека называют предынкрементным/постдекрементным. РУС может адресовать

любую область РПД. В процессе инициализации контроллера после сигнала сброса в РУС автоматически загружается код 07Н.

Двухбайтный ***регистр - указатель данных*** (РУД) обычно используется для фиксации 16-битного адреса в операциях с обращением к внешней памяти. Командами микроконтроллера регистр-указатель данных может быть использован или как 16-битный регистр, или как два независимых 8-битных регистра (DPH и DPL).

Таймер/счётчик. В составе средств контроллера имеются регистровые пары с символическими именами TH0, TL0 и TH1, TL1, на основе которых функционируют два независимых программно-управляемых 16-битных таймера/счётчика событий.

Регистры специальных функций. Регистры с символическими именами IP, EE, TMOD, TCON, SCON и PCON используются для фиксации и программного изменения управляющих бит и бит состояния схемы прерывания, таймера/счётчика, приёмопередатчика последовательного порта и для управления мощностью электропитания.

Большинство команд контроллера выполняется за один машинный цикл. Некоторые команды, оперирующие двухбайтными словами или связанные с обращением к внешней памяти, выполняются за два машинных цикла. Только команды деления и умножения требуют четырёх машинных циклов.

Система команд контроллера содержит 111 базовых команд, которые удобно разделить по функциональным признакам на пять групп:

1) команды передачи данных;
2) арифметических операций;
3) логических операций;
4) передачи управления;
5) операций с битами.

Состав операндов контроллера можно разделить на четыре типа:

а) биты;

б) 4-битные цифры;

в) байты;

г) 16-битные слова.

Микроконтроллер имеет 128 программно-управляемых флагов пользователя. Имеется также возможность адресации отдельных бит блока регистров специальных функций и портов. Для адресации бит используется прямой 8-битный адрес. Косвенная адресация бита невозможна. Четырёхбитные операнды используются только при операциях обмена (команды SWAP и XCHD). Восьмибитным операндом может быть ячейка памяти программ или данных (резидентной или внешней), константа (непосредственный операнд), регистры специальных функций (РСФ), а также порт ввода/вывода. Порты и РСФ адресуются только прямым способом. Байты памяти могут адресоваться также и косвенным образом через адресные регистры (R0, Rl, DPTR и PC). Двухбайтные операнды - это константы и прямые адреса, для представления которых используются второй и третий байт команды.

Существуют следующие **способы адресации** операндов-источников:

1) регистровая адресация;
2) прямая;
3) косвенно-регистровая;
4) непосредственная;
5) косвенная адресация по сумме базового и индексного регистра.

Первые три способа используются также для адресации операнда назначения. Указанные пять способов адресации, используемые в различных сочетаниях, обеспечивают 21 режим адресации. Многие команды содержат поля «приемник» и «источник», которые определяют тип данных, метод адресации и участвующие операнды. Для команд, не выполняющих операции перезаписи, операнд назначения является операндом-источником.

Регистровая адресация. Используется для обращения к восьми рабочим регистрам выбранного банка рабочих регистров (эти же регистры могут быть выбраны с помощью прямой адресации и косвенно-регистровой адресации как обычные ячейки внутреннего ОЗУ данных). Регистровая адресация используется также для обращения к регистрам A, B, AB (сдвоенному регистру), DPTR и к флагу переноса C. Использование регистровой адресации позволяет получать двухбайтовый эквивалент трёхбайтовых команд прямой адресации.

Прямая адресация. Используется для обращения к ячейкам внутренней памяти (ОЗУ) данных (0-127), и к регистрам специального назначения. Прямая побитовая адресация используется для обращения к отдельно адресуемым 128 битам, расположенным в ячейках с адресами 20H-2FH и к отдельно адресуе-

мым битам регистров специального назначения. Старший бит байта кода прямого адреса выбирает одну из двух групп отдельно адресуемых бит, расположенных в ОЗУ или регистрах специального назначения. Прямо адресуемые биты с адресами 0-127 (00H-7FH) расположены в блоке из 16 ячеек внутреннего ОЗУ, имеющих адреса 20H-2FH. Указанные ячейки последовательно пронумерованы от младшего бита младшего байта до старшего бита старшего байта. Отдельно адресуемые биты в регистрах специального назначения пронумерованы следующим образом: пять старших разрядов адреса совпадают с пятью старшими разрядами адреса самого регистра, а три младших - определяют местоположение отдельного бита внутри регистра.

Косвенно-регистровая адресация. Используется для обращения к ячейкам внутреннего ОЗУ данных. В качестве регистров-указателей используются регистры R0, R1 выбранного банка регистров. В командах PUSH и POP используется содержимое указателя стека (SP).

Косвенно-регистровая адресация используется также для обращения к внешней памяти данных. В этом случае с помощью регистров-указателей R0 и R1 (выбранного банка рабочих регистров) выбирается ячейка из блока в 256 байт внешней памяти данных. Номер блока предварительно задаётся содержимым порта P2. 16-разрядный указатель данных DPTR может быть использован для обращения к любой ячейке адресного пространства внешней памяти данных объёмом до 64 Кбайт.

Непосредственная адресация позволяет выбрать из адресного пространства памяти программ константы, явно указанные в команде.

Косвенно-регистровая адресация по сумме: базовый регистр плюс индексный регистр (содержимое аккумулятора A) - упрощает просмотр таблиц, зашитых в памяти программ. Любой байт из таблицы может быть выбран по адресу, определяемому суммой содержимого DPTR или PC и содержимого A.

Флаги результата. Слово состояния программы (PSW) включает в себя четыре флага: C - перенос, AC - вспомогательный перенос, OV - переполнение и P - паритет. Флаг паритета напрямую зависит от текущего значения аккумулятора. Если число единичных бит аккумулятора нечётное, то флаг P устанавливается, а если чётное - сбрасывается. Флаг AC устанавливается в случае, если при выполнении операции сложения/вычитания между тетрадами байта возник перенос/заём. Флаг C устанавливается, если в старшем бите результата возникает перенос или заём. При выполнении операций умножения и деления

флаг C сбрасывается. Флаг OV устанавливается, если результат операции сложения/вычитания не укладывается в семи битах и старший, восьмой бит результата не может интерпретироваться как знаковый. При выполнении операции деления флаг OV сбрасывается, а в случае деления на нуль устанавливается. При умножении флаг OV устанавливается, если результат больше 255.

Система команд микроконтроллера приведена в табл. 1.2.

Таблица 1.2

Система команд микроконтроллера серии 8051

Название команды	Мнемокод	Операция
1	2	3
Команды пересылки данных		
Пересылка в аккумулятор из регистра (n=0-7). *Регистровая адресация*	MOV A,Rn	(A)←(Rn)
Пересылка в аккумулятор прямо адресуемого байта. *Прямая адресация*	MOV A,ad	(A)←(ad)
Пересылка в аккумулятор байта из РПД (i=0,l). *Косвенно-регистровая адресация*	MOV A,@Ri	(A)←((Ri))
Загрузка в аккумулятор константы. *Непосредственная адресация*	MOV A,#d	(A)←#d
Пересылка в ПД из аккумулятора	MOV @Ri,A	((Ri))←(A)
Загрузка указателя данных	MOV DPTR,#d	(DPTR)←#d
Пересылка в аккумулятор байта из ПД	MOVX A,@Ri	(A)←((Ri))
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 младшей тетрады аккумулятора с младшей тетрадой байта РПД	XCHD A,@Ri	$(A_{0-3})\leftrightarrow$ $\leftrightarrow ((Ri)_{0-3})$
Команды арифметических операций		
Сложение аккумулятора с регистром (n=0,7)	ADD A,Rn	(A)←(A)+(Rn)
Сложение аккумулятора с константой	ADD A,#d	(A)←(A)+#d
Сложение аккумулятора с регистром и переносом	ADD C A,Rn	(A)←(A)+ +(Rn)+C

Продолжение табл. 1.2

1	2	3
Сложение аккумулятора с байтом из РПД (i=0,1)	ADD A,@Ri	(A)←(A)+((Ri))
Десятичная коррекция аккумулятора	DA A	Если $(A_{0-3})>9\cup$ $\cup((AC)=1)$, то $(A_{0-3})\leftarrow(A_{0-3})+$ +6, затем, если (At-7)>9∪ ∪((C)=l), то $(A_{4-7})\leftarrow$ $\leftarrow(A_{4-7})+6$
Вычитание из аккумулятора регистра и заёма	SUBB A,Rn	(A)←(A)-(C)- -(Rn)
Инкремент аккумулятора	INC A	(A)←(A)+1
Декремент аккумулятора	DEC A	(A)←(A)-1
Умножение аккумулятора на регистр B	MUL A B	(B)(A)←(A)× ×(B)
Деление аккумулятора на регистр B	DIV A B	(B)(A)← ← (A)/(B)
Группа команд логических операций		
Логическое И аккумулятора и регистра	ANL A,Rn	(A)←(A)∪(Rn)
Логическое И аккумулятора и байта из ПД	ANL A,@Ri	(A)←(A)∪(Ri))
Логическое И аккумулятора и константы	ANL A,#d	(A)←(A)∪#d
Логическое ИЛИ аккумулятора и регистра	ORL A,Rn	(A)←(A)∩(Rn)
Логическое ИЛИ аккумулятора и байта из ПД	ORL A,@Ri	(A)←(A)∩((Ri))
Логическое ИЛИ аккумулятора и константы	ORL A,#d	(A)←(A)∩#d
Исключающее ИЛИ аккумулятора и регистра	XRL A,Rn	(A)←(A)∇(Rn)
Исключающее ИЛИ аккумулятора и байта из ПД	XRL A,@Ri	(A)←(A)∇((Ri))
Исключающее ИЛИ аккумулятора и константы	XRL A,#d	(A)←(A)∇#d
Сброс аккумулятора	CLR A	(A)←0
Инверсия аккумулятора	CPL A	(A)←($\overline{A}$)
Сдвиг аккумулятора влево циклический	RLA	(An+i)←(A), n=0-6 $(A_0)\leftarrow(A_7)$
Сдвиг аккумулятора влево через перенос	RLCA	(An+i)←(A), n=0-6 $(A_0)\leftarrow$(C), (C)←(A_7)

Продолжение табл. 1.2

1	2	3
Сдвиг аккумулятора вправо циклический	RRA	(An)←(An+i), n=0-6 (A_7)←(A_0)
Сдвиг аккумулятора вправо через перенос	RRCA	(An)←(An+i), n=0-6 (A_7)←(C), (C)←(A_0)
Обмен местами тетрад в аккумуляторе	SWAP A	(A_{0-3})←(A_{4-7})
Группа команд операций с битами		
Сброс переноса	CLR C	(C)←0
Сброс бита	CLR bit	(b)←0
Установка переноса	SETB C	(C)←1
Инверсия переноса	CPL C	(C)←($\overline{C}$)
Пересылка бита в перенос	MOV C, bit	(C)←(b)
Пересылка переноса в бит	MOC bit,C	(b)←(C)
Переход, если аккумулятор равен 0	JZ rel	(PC)←(PC)+2, если (A)=0, то (PC)←(PC)+rel
Переход, если аккумулятор не равен 0	JNZ rel	(PC)←(PC)+2, если (A)≠0, то (PC)←(PC)+rel
Переход, если перенос равен 1	JC rel	(PC)←(PC)+2, если (C)=1, то (PC)←(PC)+rel
Переход, если перенос равен 0	JNC rel	(PC)←(PC)+2, если (C)=0, то (PC)←(PC)+rel
Переход, если бит равен 1	JB bit,rel	(PC)←(PC)+3, если (b)=l, то (PC)←(PC)+rel
Переход, если бит установлен, с последующим сбросом бита	JBC bit,rel	(PC)←(PC)+3, если (b)=1, то (b)←0 и (PC)←(PC)+rel
Декремент регистра и переход, если он не равен 0	DJTNZ Rn,rel	(PC)←(PC)+2, (Rn)←(Rn)-1, если (Rn)≠0, то (PC)←(PC)+rel

3. Описание программ, используемых в лабораторной работе

В лабораторной работе используются программы: компилятор ***µVision***, отладчик ***dScope*** и обучающая оболочка ***kit.exe***. Рассмотрим работу с этими программами.

µVision – компилятор, предназначенный для создания исполнительных файлов микроконтроллеров серий 8051, 251 и 166. Исполнительные файлы имеют шестнадцатеричный формат.

Проекты управления. Компилятор ***µVision*** управляет программными проектами. С целью определения предупреждений и ошибок имеется возможность компилирования отдельных исходных файлов. Набор инструментальных средств определяет компилятор, ассемблер, компоновщик и другие инструментальные средства, включенные в среду.

Поддержка файла. В компиляторе ***µVision*** для компиляции файла в активном окне и (или) проверки индивидуальных исходных файлов необходимо выбрать команду *Compile File* в меню *Project*. Для быстрой компиляции активного исходного файла можно использовать специальную кнопку на области инструмента.

Окно сообщений *Project Status* отображает состояние трансляции. Если нет ошибки или предупреждения, в данном окне указывается, что трансляция была успешной.

Проектная поддержка файлов. Часто программы состоят из многих исходных файлов. Одни из них могут требовать трансляции, используя компилятор, другие - используя ассемблер, третьи могут в процессе формирования исполнительной программы требовать специальной трансляции.

Чтобы устранить запутанность обслуживания проекта, ***µVision*** включает ***Менеджер проекта***, который:

– обслуживает проектный файл, сохраняющий всю информацию о проекте;

– формирует список всех исходных файлов, требуемых для формирования конечного файла;

– сохраняет настройки инструментальных средств;

– проверяет зависимость исходных файлов и определяет, какие из них являются текущими и какие должны быть восстановлены;

– компилирует и транслирует исходные файлы в объектные файлы;

– объединяет объектные файлы в библиотечный модуль;

– конвертирует созданные объектные модули в *Intel* шестнадцатеричные файлы.

Проектные файлы имеют вид *.PRJ.

Создание нового проектного файла. Для создания нового проектного файла необходимо выбрать команду *New Project* от меню *Project,* которая отображает диалоговое окно *Create New Project*. Затем необходимо выбрать диск и каталог, в который будет размещаться новый проектный файл, ввести имя проекта в текстовом поле *file name* и нажать кнопку *OK*.

Открытие проектного файла. Для открытия существующего проектного файла необходимо выбрать команду *Open Project* в меню *Project*, которая отображает диалоговое окно *Open Project.* Затем необходимо выбрать диск, каталог, проектное имя файла из списка *file name* и нажать OK.

Редактирование проектного файла. Для редактирования свойств текущего проекта необходимо использовать команду *Edit Project* из меню *Project*, которая открывает диалоговое окно *Project*. В данном окне перечислены проектные исходные файлы и опции.

Закрытие проектного файла. Для закрытия текущего проектного файла применяют команду *Close Project* из меню *Project*. При закрытии проектного файла удаляются опции и проектный список файла из памяти. Опции набора инструментальных средств восстанавливаются к значениям, сохраненным по умолчанию в файле *µVision.INI*.

Редактирование файлов. Первичная функция ***µVision*** должна обеспечить создание и изменение исходных файлов. При чтении исходного файла в ***µVision*** он отображается в окне редактирования. Встроенный редактор функции позволяет делать изменения в тексте в окне редактирования и сохранять содержание окна редактирования на диске.

Редактор эксплуатационных режимов

Режим только для чтения. Файлы, которые имеют атрибут «только для чтения», или файлы, которые открываются только для чтения, не могут изменяться ***µVision***. Чтобы сделать изменения в файле «только для чтения», необходимо сохранить файл по команде *Save As* с другим именем. Можно также копировать текст файла в буфер обмена и вставить его содержимое в новое ок-

но редактирования. Если файл «только для чтения», то ***µVision*** отображает индикатор *RDONLY* в строке состояния.

Шестнадцатеричный режим редактирования. Часто необходимо вставлять в файл расширенные символы ASCII. Шестнадцатеричный режим редактирования позволяет редактировать файл, используя смешанное отображение ASCII и шестнадцатеричного формата. Шестнадцатеричные данные отображаются на левой стороне окна, а эквивалентные символы ASCII - на правой стороне окна. В шестнадцатеричном режиме редактирования работают большинство редакторов функций. Для перемещения из области символов ASCII к области шестнадцатеричных данных и обратно необходимо использовать клавишу табуляции.

Режим вставки и режим замены. Редактор в ***µVision*** может работать в режиме вставки и режиме замены. Текущий режим отображается в строке состояния.

Отладчик *dScope*

dScope - программный отладчик, который моделирует аппаратные средства микроконтроллеров серии MCS 51, MCS 251 и 80C166 и может выполнять все машинные команды. В нем моделирование интегрированных периферийных устройств осуществляется посредством загружаемых драйверов.

Оконный интерфейс и операции. Программный отладчик ***dScope*** делит экран на несколько пользовательских областей с перестраиваемой конфигурацией (рис. 1.2). Каждое окно может быть показано или скрыто. Выбрать окно можно, щелкая указателем в пределах окна или табулируя через список окон, используя *Ctrl+Tab*.

Универсальное окно и строка главного меню. Окно *Mainframe* (главное окно) - "родитель" всех других окон ***dScope***. Оно содержит главное меню, инструментальную панель с ярлыками для показа и скрытия индивидуальных "дочерних" окон. Строка состояния расположена внизу главного окна. Его цель - дать короткий комментарий для каждого раздела меню.

Окно отладки. Окно *DEBUG* одновременно отображает исходный текст загруженной программы и команды ассемблера. Для изменения режима визуального отображения применяется меню *Commands*. Текст может быть пролистан вверх, вниз или горизонтально посредством полосы прокрутки, клавиш *PgUp*, *PgDn* или курсора.

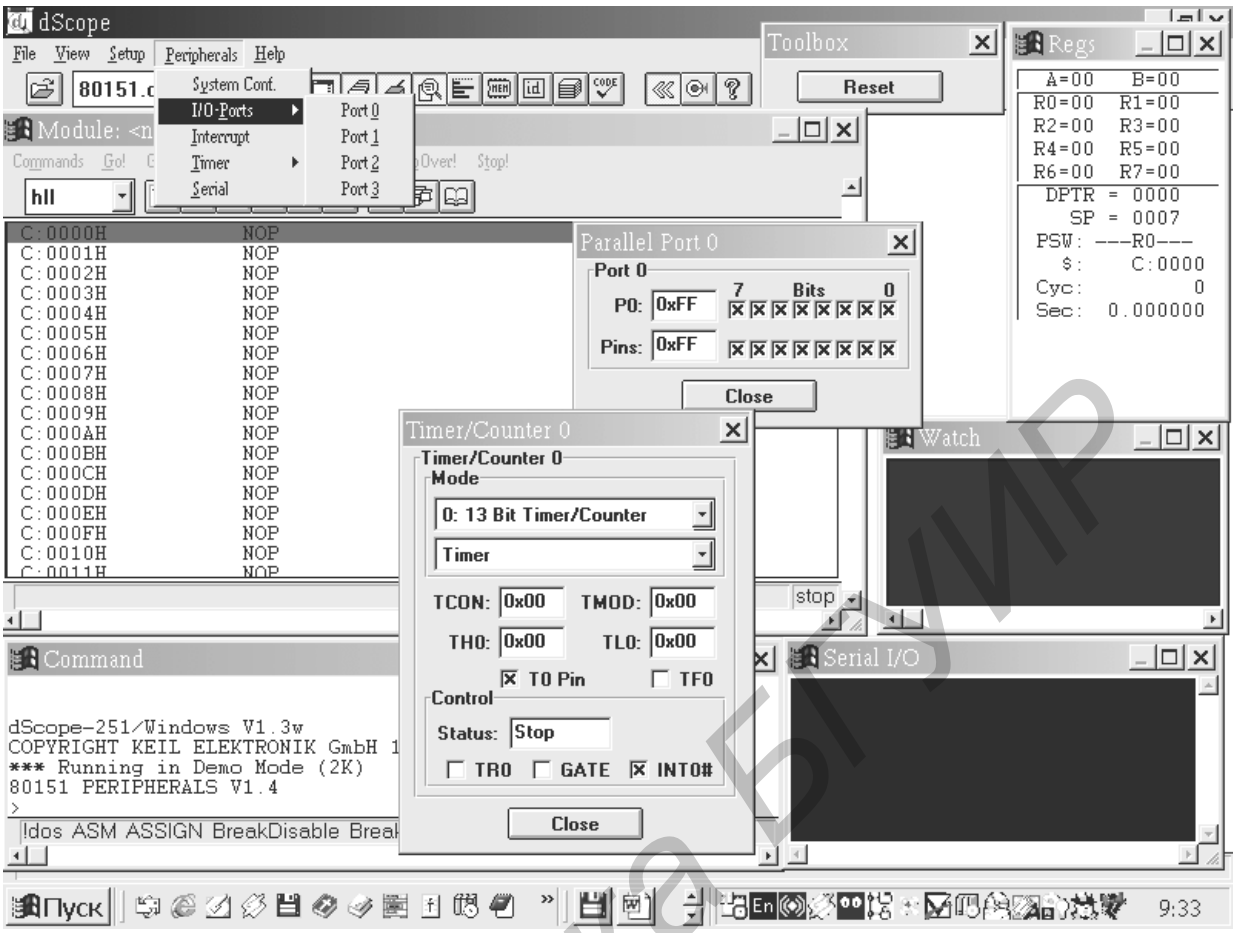


Рис. 1.2. Окна программы ***dScope***

Меню *Commands* также содержит другие команды, имеющие дело с регистрацией следа, исходным выбором модуля, текстовым поиском и другими функциями. Пункты меню сохранения используются для запуска и остановки выполнения программы пользователя. Они не имеют опускающегося меню и дублированы в области диалога справа от окна *DEBUG*. Возможно выполнение отдельных шагов программы различными способами, например, вводя *Alt+S,* или выбирая *StepInto* из строки меню, или нажимая кнопку *StepInto* в области диалога.

Строка состояния окна *DEBUG* показывает текущий режим визуального отображения (*HLL/MIXED/ASM*) и *(RUN / STOP)* состояние программ моделирования. В строке состояния показывается краткое описание команды.

Окно COMMAND (команд) применяется для ввода командной строки. В этом окне также происходит вывод большинства команд. Преимущество окна *COMMAND*: командные строки могут быть помещены в файлы на диске, которые читаются командой *INCLUDE*.

Окно REGISTER (регистр) показывает различные значения ЦЕНТРАЛЬНОГО ПРОЦЕССОРА, текущего счетчика программы и выполненные циклы плюс дает информацию о времени. Размещение регистра в окне REGISTER зависит от типа моделируемого ЦЕНТРАЛЬНОГО ПРОЦЕССОРА.

В ***окне WATCH (наблюдение)*** размещаются значения составных типов (структуры, объединения, массивы) или простые скалярные значения. Элементы, которые будут отображаться, определяются и удаляются командами в окне *COMMAND* или определенным диалогом. Окно *WATCH*, как правило, исчезает после выполнения программы пользователя.

Окно SERIAL (последовательное) эмулирует последовательный терминал, соединенный с MCS 51/251 (80C166) специальными функциональными регистраторами или параллельным портом. Оно поддерживает буфер хронологий, который позволяет рассматривать до 8 Кб данных последовательного вывода.

Окно MEMORY (памяти) используется для отображения диапазона памяти в шестнадцатеричных байтах. Диапазон может быть длиной до 64 Кб.

Окно Symbol Browser (броузер) показывает законченную символическую информацию об отладке загруженной программы пользователя. Эта информация включает общие символы, функции с местными символами и информацией о номере строки. Вывод может быть ограничен символами или определенным типом памяти. Броузер также включает простой, но мощный регулярный механизм поиска выражения. *Symbol Browser* поддерживает перетаскивание символов в другие окна ***dScope***, например окно *COMMAND*, и к входным полям всех диалогов.

Окно PERFORMANCE ANALYZER (анализатора выполнения) отображает символическое имя или числовой адрес до 255 определенных диапазонов. Окно дает быстрый краткий обзор ПРОЦЕССОРНОГО ВРЕМЕНИ, используемого для каждого диапазона.

Окно CALLSTACK (списка последовательных вызовов) показывает текущее функциональное положение запроса. Если выбрана строка дисплея, на которой содержится оператор, вызывающий некую функцию, то на основании окна *CALLSTACK* показывается информация об этой функции.

Окно CODE COVERAGE (охвата кода) показывает функции, основанные на выбранном пользователем модуле вместе с числом команд каждой функции, и оценивает процент выполненных команд. Дополнительные особен-

ности включают динамическую модификацию при выполнении программы пользователя и синхронизирование окна *DEBUG* в следующей выполненной или невыполненной команде.

Окно TOOLBOX (комплекта инструментов) содержит определяемые пользователем кнопки, которые представляют команды. Может быть определено до 15 кнопок. Первая кнопка, помеченная “Сброс”, заранее определена и не может быть удалена.

dScope может осуществлять модификацию значений портов ввода/вывода в ходе выполнения программы и контроль за состояниями таймеров (см. рис. 1.2). Эта функция доступна только после определения драйвера микроконтроллера.

Формат сообщения об ошибках. При обнаружении ошибки ***dScope*** выпускает пронумерованное сообщение об ошибках. Пункт, в котором была найдена ошибка, может быть отмечен в зависимости от ее контекста. Например:

acc + r0 + r500
ERROR 125: symbol or line not found
load c:\objs\measure
ERROR 109: file does not exist

4. Порядок выполнения работы

1. Запустить программный модуль с помощью ярлыка, находящегося на рабочем столе (программа ***Kit.exe***).

2. Прочитать краткие теоретические сведения, для чего необходимо нажать клавишу *F1*.

3. Получить задание. Задание выдаётся с помощью кнопки меню *Задание → Выдать*.

4. Запустить среду ***µVision***, нажав кнопку *Запуск → Среда разработки*.

5. Создать новый проект *Project->New Progec...*, затем в появившемся окне нажать кнопку *Save*.

6. Создать новый файл -*File->New...*, написать программу в соответствии с индивидуальным заданием и затем сохранить ее.

7. Войти в меню *Project->Edit progect...*, нажав кнопку *Add*, добавить свой файл к проекту.

8. При выборе *Progect->Make: Build project* компилятор создаст файл вашей программы в *HEX* коде (в случае отсутствия ошибок).

9. Запустить отладчик, нажав клавишу меню *RUN→dScope Debugger...*

10. Выбрать меню *File->Load CPU driver...*, выбрать 8051.

11. Загрузить вашу программу в формате *HEX*. Для этого в меню *File* необходимо нажать *Load object file…*, выбрать в падающем меню «Показывать файлы с расширением *HEX»*, выбрать ваш файл, нажать ОК.

12. Определить в соответствии с заданием необходимые отладочные окна.

13. Командой *GOInto!* в окне *Module* произвести пошаговый проход программы. (Возврат к началу - кнопка *Reset* в окне *Toolbar*). Скорректировать ошибки.

14. Командой *GO!* произвести эмуляцию работы программы в автоматическом и пошаговом режимах. Записать состояние портов, таймеров и регистров, указанных преподавателем, используя соответствующие окна.

15. Показать преподавателю результат работы.

16. Занести текст программы в отчет.

5. Содержание отчета

1. Цель работы.

2. Листинг программы вычисления заданной функции.

3. Таблица состояний портов, таймеров, регистров.

4. Выводы.

6. Индивидуальные задания

В порт P0 микроконтроллера поступает установившаяся комбинация сигналов с трех датчиков. Каждый датчик использует определенные биты портов (табл. 1.3).

Требуется:

1. Считать информацию каждого датчика с порта P0.

2. Сохранить в памяти информацию с каждого датчика.

3. Произвести вычисление по формуле Dat1*(Dat2+Dat3)*6 и сохранить результат в памяти.

4. Вывести результат на три семисегментных ЖКИ, причем каждый разряд ЖКИ - это соответственно порты: P1 - для младшего, P2 - для среднего, P3 - для старшего разряда.

Таблица 1.3

Варианты индивидуальных заданий

Вариант	Содержание
1	Первый датчик (Dat1) – 5-, 6-, 7-й; второй датчик (Dat2) – 3-, 4-й; третий датчик (Dat3) – 0-, 1-, 2-й биты
2	Первый датчик (Dat1) - 0,- 1-й; второй датчик (Dat2) – 2-, 3-, 4-й; третий датчик (Dat3) – 5-, 6-, 7-й биты
3	Первый датчик (Dat1) – 0-, 1-, 2-й; второй датчик (Dat2) – 3-, 4-, 5-й; третий датчик (Dat3) – 6-, 7-й биты
4	Первый датчик (Dat1) – 0-, 1-, 2-й; второй датчик (Dat2) – 3-, 4-й; третий датчик (Dat3) – 5-, 6-, 7-й биты

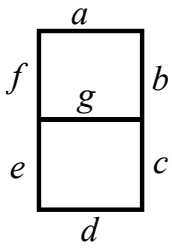
Рекомендуется начать программу со следующих строк:

```
Datchik DATA 30h; байт с порта P0
Dat1 DATA 31h; первый датчик
Dat2 DATA 32h; второй датчик
Dat3 DATA 33h; третий датчик
Rez DATA 34h; результат
Ind1 DATA 35h; первый разряд индикатора
Ind2 DATA 36h; второй разряд индикатора
Ind3 DATA 37h; третий разряд индикатора
ORG 00 ; стартовый адрес
MOV SP,#40h; начало стека
```

Примечание. Для преобразования цифры в семисегментный код рекомендуется использовать следующую процедуру:

– значение цифры передается в аккумулятор A;
– значение кода для семисегментного возвращается в аккумулятор A.

```
Led_TABLE:                  ; Перевод десятичных чисел
INC   A                     ; в семисегментный код
MOVC      A,@A+PC   ;
RET               ;
;Сегменты         gfedcba
```




```
DB    00111111b  ; 0
DB    00000110b  ; 1
DB    01011011b  ; 2
DB    01001111b  ; 3
DB    01100110b  ; 4
DB    01101101b  ; 5
DB    01111101b  ; 6
DB    00000111b  ; 7
DB    01111111b  ; 8
DB    01101111b  ; 9
```

7. Контрольные вопросы

1. Назначение и состав регистров микроконтроллера 8051.
2. Какие флаги состояния используются в микроконтроллере?
3. Перечислите способы адресации и дайте им краткую характеристику.
4. Команды передачи данных.
5. Команды арифметических операций.
6. Команды логических операций.
7. Команды операций с битами.
8. Команды передачи управления.
9. Какие окна существуют в отладчике программ и как они отображаются? Последовательность написания и отладки программ в данном лабораторном модуле.

Лабораторная работа №2

ПРИНЦИПЫ РЕАЛИЗАЦИИ ФУНКЦИЙ УПРАВЛЕНИЯ В СРЕДСТВАХ МЕДИЦИНСКОЙ ЭЛЕКТРОНИКИ

1. Цель работы

Изучить принципы составления рабочей программы, реализующей функции управления средствами медицинской электроники. Выработать практические навыки программирования на языке *Ассемблер* микроконтроллера серии 8051. Изучить функции управления, обращения к внешним устройствам.

2. Краткие теоретические сведения

Задачи управления средств медицинской электроники. Структурная организация, набор команд и аппаратно-программные средства ввода/вывода информации микроконтроллеров хорошо приспособлены для решения задач управления и регулирования средств медицинской электроники. Микроконтроллеры не являются машинами классического фон-неймановского типа, так как физическое и логическое разделение памяти программ и памяти данных исключает возможность модификации или замены прикладных программ микроконтроллеров во время работы, что сильно затрудняет их использование в качестве универсальных средств обработки данных. Для решения задач управления средствами медицинской электроники микроконтроллер может быть использован как средство вычисления и индикации биоэлектрической информации или как устройство управления медицинским аппаратом. Большинство современных медицинских аппаратов снабжено микропрограммным управлением режимов работы. Это может быть отсчет времени процедуры, сигнализация превышения допустимого содержания вредных веществ в воздухе или в крови, индикация данных прибора и т.п. Все операции, которые производятся микроконтроллером, можно изменять посредством перепрограммирования. Перепрограммирование подразумевает занесение в память программ микроконтроллера программы для решения поставленной задачи и последующее ее исполнение.

Данные, получаемые с датчиков и приборов медицинской электроники, в большинстве случаев характеризуются малой частотой сигнала, за исключением некоторых специальных исследований (ультразвуковое исследование). Такой характер биомедицинской информации позволяет избежать при создании

медицинской электронной аппаратуры громоздкости высокоскоростных компьютеризованных систем и воспользоваться более экономичными микроконтроллерами. Наиболее перспективным направлением медицинской техники являются системы с обратной связью (ОС). Такие системы наряду с терапевтическим каналом содержат контур ОС, по которому с целью оптимизации воздействия на организм в реальном масштабе времени передается диагностическая информация. Проектирование систем с ОС базируется на моделях и методах оптимального управления.

Структура одноканальной системы с биотехнической обратной связью (БТОС) показана на рис. 2.1. Управляющее воздействие u(t) формируется управляющим устройством на основе сравнения действительного x(t) состояния объекта с заданным z(t).

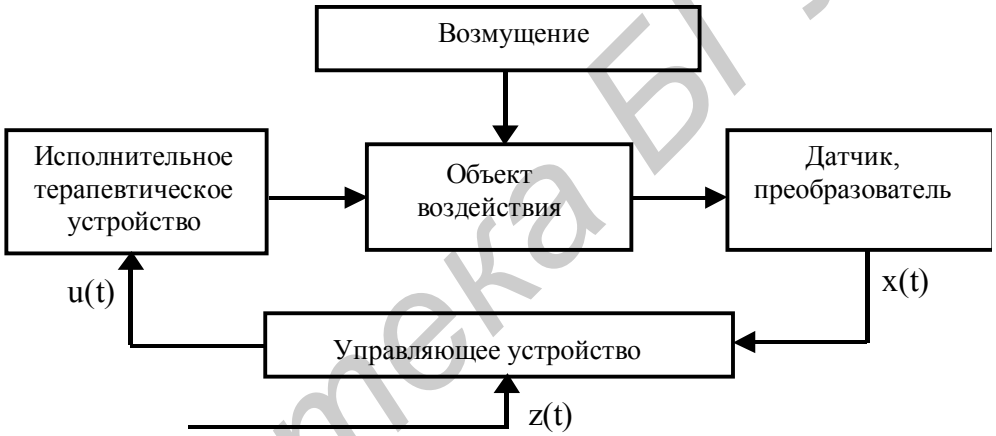


Рис. 2.1. Структура одноканальной медицинской системы с биологической обратной связью

Обычно управляющее устройство в подобных системах вырабатывает сигнал управления u(t), являющийся функцией меры ошибки:

$$u(t) = u(H[z(t), x(t)]),$$

где H [z(t), x(t)] = z(t) - x(t) - мера ошибки, характеризующая разность между действительным и желаемым значениями параметра.

Мгновенное значение сигнала управления u(t) является функцией мгновенных значений выходного x(t) и входного z(t) сигналов. В системах оптимального управления учитываются рассогласования с учетом прогноза характера измерения параметра x(t). Чтобы предсказанное управление было допустимым, необходимо учитывать интегральную меру ошибки J на интервале управления в будущем:

$$J = \int_{t}^{t+T} H[z(\tau), x(\tau)]\, d\tau .$$

Очевидно, что критерий ошибки J следует минимизировать. В оптимальных системах управления прогнозируется управляющий вход u(t) в будущем на интервале $t < \tau < t+T$ и управляющее устройство отрабатывает алгоритмы, направленные на минимизацию критерия ошибки J. Алгоритмы функционирования управляющего устройства реализуют определенные законы управления, которые задаются с помощью дифференциальных уравнений во временной области. Законы управления описывают динамику изменения состояния объекта. Существует множество способов описания динамики изменения состояния объекта, из которых можно выделить два основных:

1) математическое описание процесса, основанное на физических законах;

2) математическое описание процесса, основанное на информации его экспериментального исследования.

Динамика состояния объекта. Динамику состояния любого технического объекта можно описать с помощью четырех групп переменных (рис. 2.2). Сигналы управления u(t) являются входными сигналами динамического процесса или независимыми переменными. Они генерируются системами, внешними по отношению к объекту, в частности управляющим устройством, и влияют на его поведение. Состояние самого объекта воздействия характеризуется переменными состояниями x(t). Выбор сигналов состояния неоднозначен и**,** как правило, определяется возможностью их измерения.

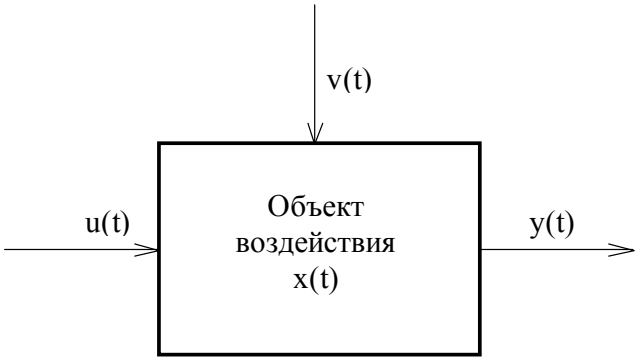


Рис.2.2. Характеристика объекта воздействия

Сигналы с выходов датчиков y(t), обеспечивающих текущий контроль объекта, называют наблюдательными переменными. Возмущениями, дейст-

вующими на объект v(t), могут служить температура окружающей среды, влажность, вибрации и т.д.

Состояние объекта описывается в большинстве случаев обыкновенными дифференциальными уравнениями или дифференциальными уравнениями в частных производных.

В любой момент времени t состояние объекта является функцией начального состояния x(t) и вектора входного сигнала и описывается уравнением состояния динамического процесса:

$$\frac{\partial x(t)}{\partial t} = f\left[x(t), u(t), t\right].$$

Вектор выходного сигнала определяется соотношением

$$y(t) = g\left[x(t), u(t), v(t), t\right].$$

Эти выражения применимы в большинстве задач автоматического управления в медицинских электронных системах.

Спектр использования микроконтроллеров, как недорогих устройств обработки медицинской информации, распространяется почти на все области медицины. Состав и структура контроллера позволяет эффективно использовать его в качестве ядра управляющей системы.

Таймер/счётчик (T/C)

В составе средств MK51 имеются регистровые пары с символическими именами TH0, TL0 и TH1, TL1, на основе которых функционируют два независимых программно-управляемых 16-битных таймера/счётчика событий. При работе в качестве таймера содержимое T/C инкрементируется в каждом машинном цикле, т.е. через каждые 12 тактов. При работе в качестве счётчика T/C инкрементируется под воздействием перехода из 1 в 0 внешнего сигнала, подаваемого на соответствующий (T0, T1) вход микросхемы. Содержимое счетчика будет увеличено на 1 в том случае, если в предыдущем цикле был считан входной сигнал высокого уровня (1), а в следующем - сигнал низкого уровня (0). Новое (инкрементированное) значение счетчика будет сформировано в цикле, следующем за тем, в котором был обнаружен переход сигнала из 1 в 0. Так как на распознавание перехода требуется два машинных цикла, то максимальная частота подсчета входных сигналов равна 1/24 тактовой частоты. На длительность периода входных сигналов ограничений сверху нет. Для гаранти-

рованного прочтения входного считываемого сигнала он должен удерживать значение 1 как минимум в течение одного машинного цикла.

Для управления режимами работы Т/С и организации взаимодействия таймеров с системой прерывания используются два регистра специальных функций TMOD и TCON (табл. 2.1 и 2.2). Как следует из описания управляющих бит TMOD, для обоих Т/С режимы работы 0, 1 и 2 одинаковы. Режимы 3 для Т/С0 и Т/С1 различны.

Таблица 2.1

Регистр режима работы таймера/счётчика (TMOD)

Символ	Позиция	Имя и назначение
Gate	TMOD.7 для Т/С1; TMOD.3 для Т/С0	*Разрешает управлять таймером от внешнего вывода ($\overline{INTx}$). Если бит установлен, то разрешено; в противном случае – запрещено*
С/Т	TMOD.6 для Т/С1; TMOD.2 для Т/С0	Определяет работу в качестве счетчика (бит установлен) или таймера (бит сброшен)
Ml	TMOD.5 для Т/С1; TMOD.1 для Т/С0	Определяет режим работы (см. примечание)
M0	TMOD.4 для Т/С1; TMOD.0 для Т/С0	Определяет режим работы (см. примечание)

Примечание:

Ml	M0	Режим работы
0	0	Режим 0
0	1	Режим 1
1	0	Режим 2
1	1	Режим 3

Режим 0. В этом режиме таймерный регистр имеет разрядность 13 бит. При переходе из состояния «все единицы» в состояние «все нули» устанавливается флаг прерывания от таймера TF1. Входной синхросигнал таймера 1 разрешен (поступает на вход Т/С), когда управляющий бит TR1 установлен и либо управляющий бит Gate (блокировка) равен 0, либо на внешний вывод запроса прерывания INT1 поступает уровень 1. Установка бита Gate позволяет также использовать таймер для измерения длительности импульсного сигнала, подаваемого на вход запроса прерывания.

Таблица 2.2

Регистр управления/статуса таймера (TCON)

Символ	Позиция	Имя и назначение
TF1	TCON.7	Флаг переполнения таймера 1. Устанавливается аппаратно при переполнении таймера/счётчика. 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 Т/С
IE1	TCON.3	Флаг фронта прерывания 1. Устанавливается аппаратно, когда детектируется срез внешнего сигнала ЗПР1 (INT1). Сбрасывается при обслуживании прерывания
IT1	TCON.2	Бит управления типом прерывания 1. Устанавливается/сбрасывается программно для спецификации запроса ЗПР1 (перепад/низкий уровень)
IE0	TCON.1	Флаг фронта прерывания 0. Устанавливается по срезу сигнала ЗПР0. Сбрасывается при обслуживании прерывания
IT0	TCON.0	Бит управления типом прерывания 0. Устанавливается/сбрасывается программно для спецификации запроса ЗПР0 (перепад/низкий уровень)

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

Режим 2. В данном режиме работа организована таким образом, что переполнение (переход из состояния «все единицы» в состояние «все нули») 8-битного счётчика TL1 не только приводит к установке флага TF1, но и автоматически перегружает в TL1 содержимое старшего байта (TH1) таймерного регистра, которое предварительно было задано программным путем. Перезагрузка оставляет содержимое TH1 неизменным. В режиме 2 Т/С0 и Т/С1 работают совершенно одинаково.

Режим 3. В данном режиме Т/С0 и Т/С1 работают по-разному. Т/С1 сохраняет неизменным свое текущее содержимое, так же как и при сбросе управляющего бита TR1. TL0 и TH0 функционируют как два независимых 8-битных счётчика. Работу TL0 определяют управляющие биты Т/С0 (C/T, Gate, TR0),

входной сигнал INT0 и флаг переполнения TF0. Работу TH0, который может выполнять только функции таймера (подсчёт машинных циклов МК), 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 счётчика событий. Можно считать, что в режиме 3 микроконтроллер имеет в своём составе три таймера/счётчика. В том случае, если T/C0 используется в режиме 3, T/C1 может быть включен, выключен или переведен в свой собственный режим 3, он также может быть использован последовательным портом в качестве генератора частоты передачи или в любом применении, не требующем прерывания.

Буфер последовательного порта. Регистр с символическим именем SBUF представляет собой два независимых регистра - буфер приемника и буфер передатчика. Загрузка байта в SBUF немедленно вызывает начало процесса передачи через последовательный порт. Когда байт считывается из SBUF, это значит, что его источником является приемник последовательного порта.

Регистры специальных функций. Регистры с символическими именами IP, EE, TMOD, TCON, SCON и PCON используются для фиксации и программного изменения управляющих бит и бит состояния схемы прерывания, таймера/счётчика, приёмопередатчика последовательного порта и для управления мощностью электропитания микроконтроллера.

Функции обращения к внешним устройствам. Порты ввода/вывода информации. Все четыре порта контроллера предназначены для ввода или вывода информации побайтно. Каждый порт содержит управляемые регистр-защелку, входной буфер и выходной драйвер.

Выходные драйверы портов 0 и 2, а также входной буфер порта 0 используются при обращении к внешней памяти (ВП). При этом через порт 0 в режиме временного мультиплексирования сначала выводится младший байт адреса ВП, а затем выдается или принимается байт данных. Через порт 2 выводится старший байт адреса в тех случаях, когда разрядность адреса равна 16 бит. Все выводы порта 3 могут быть использованы для реализации альтернативных функций, перечисленных в табл. 2.3. Альтернативные функции могут быть задействованы путем записи 1 в соответствующие биты регистра-защелки (P3.0-P3.7) порта 3. Порт 0 является *двунаправленным*, а порты 1, 2 и 3 - *квазидвунаправленными*.

Таблица 2.3

Альтернативные функции порта 3

Символ	Позиция	Имя и назначение
$\overline{RD}$	P3.7	Чтение. Активный сигнал низкого уровня формируется аппаратно при обращении к ВПД
$\overline{WR}$	P3.6	Запись. Активный сигнал низкого уровня формируется аппаратно при обращении к ВПД
T1	P3.5	Вход таймера/счетчика 1 или тест-вход
T0	P3.4	Вход таймера/счетчика 0 или тест-вход
$\overline{INT}$1	P3.3	Вход запроса прерывания 1. Воспринимается сигнал низкого уровня или срез
$\overline{INT}$0	P3.2	Вход запроса прерывания 0. Воспринимается сигнал низкого уровня или срез
TXD	P3.1	Выход передатчика последовательного порта в режиме универсального асинхронного приемо-передатчика (УАПП). Выход синхронизации в режиме сдвигающего регистра
RXD	P3.0	Вход приемника последовательного порта в режиме УАПП. Ввод/вывод данных в режиме сдвигающего регистра

Особенности работы портов. Обращение к портам ввода/вывода возможно с использованием команд, оперирующих байтом, отдельным битом и произвольной комбинацией бит. При этом в тех случаях, когда порт является одновременно операндом и местом назначения результата, устройство управления автоматически реализует специальный режим, который называется "*чтение—модификация—запись*" (табл. 2.4). Этот режим обращения предполагает ввод сигналов не с внешних выводов порта, а из его регистра-защелки, что позволяет исключить неправильное считывание ранее выведенной информации.

Последние три команды (см. табл. 2.4) в приведенном списке также являются командами "*чтение—модификация—запись*", так как по этим командам сначала считывается байт из порта, а затем записывается новый байт в регистр-защелку.

Таблица 2.4

Команды, использующие режим обращения "чтение—модификация—запись"

Имя команды	Назначение	Пример использования
ANL	Логическое И	ANL P1, A;
ORL	Логическое ИЛИ	ORL P2, A;
XRL	Исключающее ИЛИ	XRL P3, A;
JBC	Переход, если в адресуемом бите единица, и последующий сброс бита	JBC P1.1, LABEL;
CPL	Инверсия бита	CPL P3.3;
INC	Инкремент порта	INC P2;
DEC	Декремент порта	DEC P2;
DJNZ	Декремент порта и переход, если его содержимое не равно нулю	DJNZ P3, LABEL;
MOV PX.Y, C	Передача бита переноса в бит Y порта X	MOV P1.2, C;
SET PX.Y	Установка бита Y порта X	SET P2.4;
CLR PX.Y	Сброс бита Y порта X	CLR P0.1;

Временные диаграммы, иллюстрирующие процесс выполнения операций ввода/вывода информации через порты, приведены на рис. 2.3.

Механизм прерываний в микроконтроллере позволяет автоматически реагировать на внешние и внутренние события. Предусмотрено пять источников прерываний: переполнение таймеров/счетчиков C/T0 и C/T1, завершение последовательного обмена, прерывание от внешнего источника INT0 и INT1.

Каждое из внешних прерываний INT0, INT1 может быть активизировано по уровню ("0") или по фронту (переход из "1" в "0") сигналов на выводах порта P3.2, P3.3, что определяется состоянием бит IT0 и IT1 регистра TCON. При поступлении запроса внешнего прерывания INTx устанавливается флаг IEx регистра TCON, что вызывает соответствующее прерывание. Очистка флага IEx производится следующим образом: при прерывании по фронту IEx сбрасывается аппаратно (автоматически внутренними средствами МЭВМ) при обращении к соответствующей подпрограмме обработки прерывания; при прерывании по уровню флаг очищается при снятии запроса внешнего прерывания, т.е. в IEx отслеживается состояние вывода INTx.

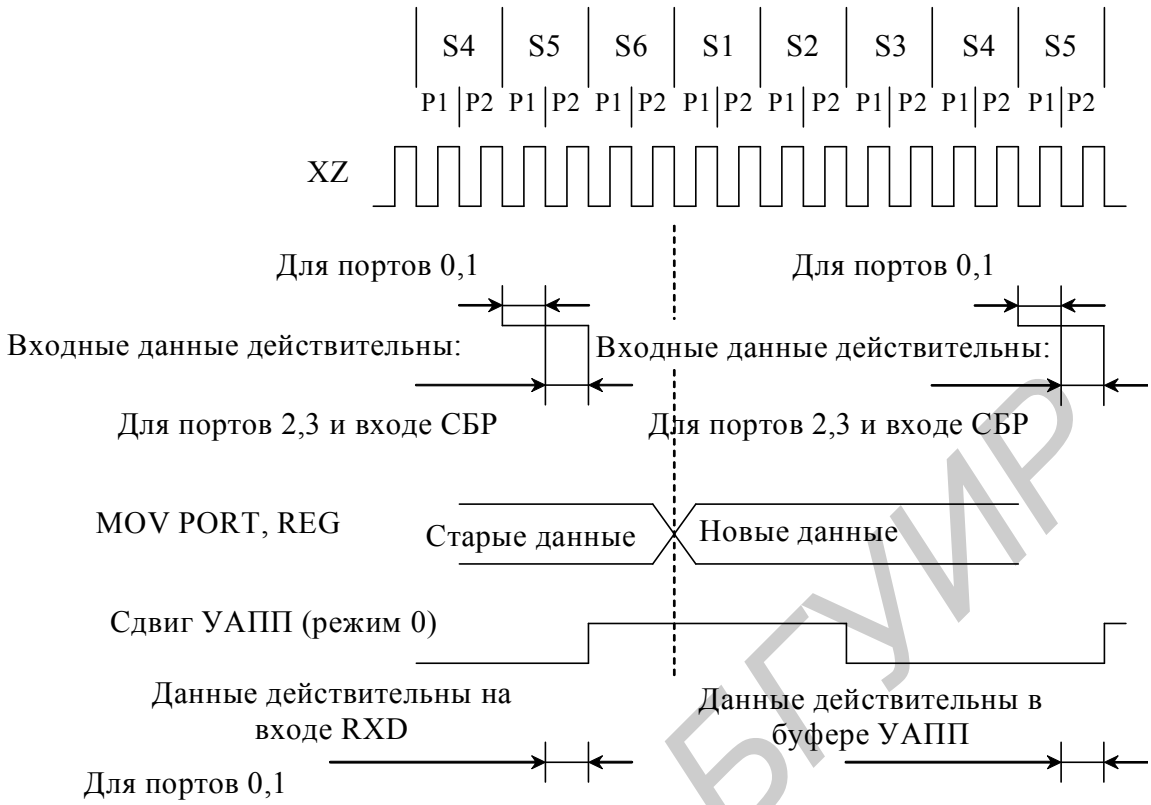


Рис. 2.3. Временные диаграммы, иллюстрирующие процесс выполнения операций ввода/вывода информации через порты микроконтроллера

Прерывания от таймеров/счетчиков вызываются установкой флагов TF0 и TF1 регистра TCON, которая осуществляется при переполнении соответствующих регистров таймеров/счетчиков (за исключением режима 3). Очистка флагов TF0 и TF1 производится внутренней аппаратурой МЭВМ при переходе к подпрограмме обслуживания прерывания.

Прерывание от последовательного порта вызывается установкой флага прерывания приемника RI или флага прерывания передатчика TI в регистре SCON. В отличие от всех остальных флагов, RI и TI сбрасываются только программным путем, обычно в пределах подпрограммы обработки прерывания, где определяется, какому из флагов RI илиTI соответствует прерывание.

Каждый из перечисленных источников прерываний может быть индивидуально разрешен или запрещен установкой или сбросом соответствующего бита в регистре разрешения прерываний IE. Регистр IE содержит также бит EA, сброс которого в “0” запрещает сразу все прерывания. Поэтому необходимым условием прерывания является его разрешение в регистре IE.

Регистр приоритетов прерываний (IP) предназначен для установки уровня приоритета прерывания. Назначение бит регистра следующее: 1-й разряд (PX0) – установка приоритета прерывания от внешнего источника

INT0, 2-й разряд (PT0) - установка приоритета прерывания от таймера/счетчика T/C0), 3-й разряд (PX1) - установка приоритета прерывания от внешнего источника INT1, 4-й разряд (PT1) - установка приоритета прерывания от таймер/счетчика T/C0), 5-й разряд (PS) - установка приоритета прерывания от последовательного порта, 6-, 7-, 8-й - резервные разряды. Наличие в разряде IP «1» устанавливает для соответствующего источника высокий уровень приоритета

Регистр разрешения прерываний (IE) предназначен для разрешения или запрещения прерываний от соответствующих источников.

Назначение разрядов регистра IE:

EA – управление всеми источниками прерываний одновременно. Если EA = 0, то прерывания запрещены. Если EA = 1, то прерывания могут быть разрешены индивидуальными разрешениями EX0, ET0, EX1, ET1, ES;

X – резервный разряд;

ES – управление прерыванием от последовательного порта: ES = 1 – разрешение, ES = 0 – запрещение;

ET – управление прерыванием от T/C1: ET1 = 1 – разрешение, ET1 = 0 – запрещение;

EX – управление прерыванием от внешнего источника $\overline{\text{INT1}}$: EX = 1 – разрешение, EX = 0 – запрещение;

ET – управление прерыванием от T/C0: ET0 = 1 – разрешение, ET0 = 0 – запрещение;

EX – управление прерыванием от внешнего источника INT0: EX0 = 1 – разрешение, EX0 = 0 – запрещение.

Все биты, которые вызывают прерывания (IE0, IE1, TF0, TF1, RI, TI), могут быть программно установлены или сброшены с тем же результатом, что и в случае их аппаратной установки или сброса. Прерывания могут программно вызываться или программно ликвидироваться (ожидающие обслуживания прерывания). Кроме того, прерывания по INT0, INT1 могут вызываться программной установкой P3.2=0 и P3.3=0, как показано в приведенном примере:

```
MAIN:  MOV IE, #00000101B  ; разрешение прерывания INT0, INT1.
       MOV IP, #04H        ; присвоение INT1 старшего приоритета.
       SET EA              ; общее разрешение прерывания.
       MOV IE, #00000101   ; имитация внешних прерываний.
```

```
SUBR:  ORG013H          ; переход к подпрограмме обслуживания
                          INT1.
```

3. Порядок выполнения работы

Внимание! При выполнении лабораторной работы должен использоваться программный модуль ***Kit.exe***. Описание порядка работы с данным модулем приведено в лабораторной работе № 1.

1. Запустить программный модуль с помощью ярлыка, находящегося на рабочем столе (программа ***Kit.exe***).
2. Прочитать краткие теоретические сведения (нажать клавишу *F1*).
3. Получить задание с помощью кнопки меню *Задание → Выдать*.
4. Запустить среду ***µVision***, нажав кнопку *Запуск →Среда разработки*.
5. Создать новый проект *Project->New Progec...*, затем в появившемся окне нажать кнопку *Save*.
6. Создать новый файл -*File->New...*, написать программу в соответствии с заданием и сохранить её.
7. Войти в меню *Project->Edit progect...* и, нажав кнопку *Add*, добавить свой файл к проекту.
8. Выбрав *Progect->Make: Build project*, компилятор создаст файл в HEX коде вашей программы (в случае отсутствия ошибок).
9. Запустить отладчик, нажав клавишу меню *RUN→dScope Debugger...*
10. Выбрать меню File->Load CPU driver..., выбрать 8051.
11. Загрузить вашу программу в формате HEX. Для этого в меню *File* нажать *Load object file…*, выбрать в падающем меню: показывать файлы с расширением HEX, выбрать ваш файл, нажать ОК.
12. Определить в соответствии с заданием необходимые отладочные окна.
13. Командой *GOInto!* в окне *Module* произвести пошаговый проход программы. (Возврат к началу кнопка *Reset* в окне *Toolbar*). Откорректировать ошибки.
14. Командой *GO!* произвести эмуляцию работы программы в автоматическом и пошаговом режимах. Записать состояние портов, таймеров и регистров, указанных преподавателем, используя соответствующие окна.
15. Показать преподавателю результат работы.

16. Занести текст программы в отчет.

4. Содержание отчета

1. Цель работы.
2. Краткие теоретические сведения.
3. Листинг программы вычисления заданной функции.
4. Таблица состояний портов, таймеров и регистров.
5. Выводы.

5. Индивидуальные задания

Задание № 1

В порт B поступают сигналы с датчиков. Первый датчик (B.0) сигнализирует о наполнении емкости водой, второй (B.1) - об отсутствии воды. Посредством сигнала с порта A можно управлять насосом (A.0). Индикация работы осуществляется светодиодом, подключенным к порту A.1 (светодиод должен мигать).

Задание № 2

Написать программу обслуживания 16 кнопочной сеточной клавиатуры. Клавиатура представляет собой сетку из четырёх строк по четыре кнопки в каждой и работает следующим образом. Порт B.0 подключен параллельно к первым входам кнопок первой строки , порт B.1 аналогично к кнопкам второй строки и т.д., соответственно происходит подключение для B.2 и B.3. Порт B.4 подключен параллельно ко вторым входам кнопок первого столбца, порт B.5 аналогично подключен к кнопкам второго столбца и т.д. Для того чтобы проверить нажатие кнопки, необходимо подать логический нуль на столбец (логическую единицу - на остальные столбцы), считать значение с соответствующей строки. Например, для того чтобы проверить, не нажата ли кнопка 7 (третий столбец, вторая строка), требуется подать логический ноль на порт B.6 и считать значение с порта B.1, если это значение '0' - то кнопка нажата. Требуется просканировать все клавиши и номер нажатой клавиши выдать в двоичном виде на порты A.0-A.3.

6. Контрольные вопросы

1. Что такое функция меры ошибок?
2. Режимы работы таймеров/счетчиков.
3. Назначение бит регистра TMOD.
4. Назначение регистра TCON.

5. Альтернативные функции порта 3.

6. Источник прерываний.

7. Назначение бит регистра прерываний IE.

Литература

1. Однокристальные микроЭВМ / А.В. Боборыкин, Г.В. Липовецкий, Г.В. Литвинский и др. - М.: МИКАП, 1994. – 400 с.

2. Проектирование цифровых устройств на однокристальных микроконтроллерах / В.В. Сташин, А.В. Урусов и др. - М.: Энергия, 1996.

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