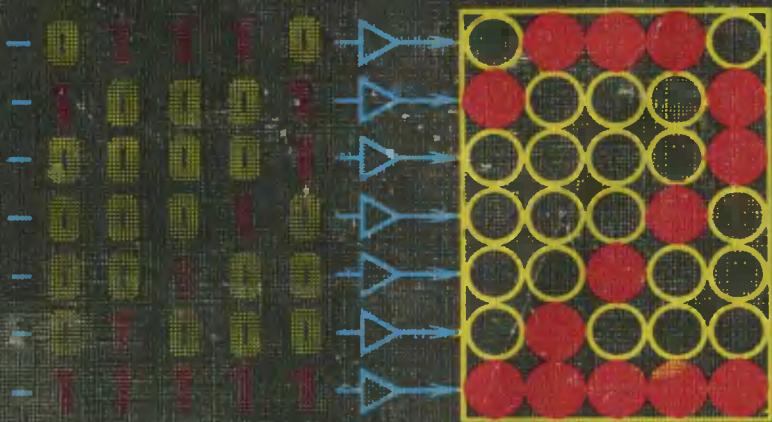


В.В.Сташин А.В.Урусов О.Ф.Малогонцева

Проектирование цифровых устройств на однокристальных микроконтроллерах



КМ1816ВЕ4 и КМ1816ВЕ51

- Особенности системы команд, особенности применения.
- Формализованная методика разработки прикладного программного обеспечения для систем и устройств логического управления объектами и технологическими процессами.
- Многочисленные примеры микроконтроллерной реализации типовых процедур управления.

Энергоатомиздат

В.В.Сташин А.В.Урусов О.Ф.Мологонцева

Проектирование цифровых устройств на однокристальных микроконтроллерах



Москва Энергоатомиздат 1990

ББК 32.96:32.97

С 78

УДК 681.58:681.32

Р е ц е н з е н т И.И. Шагурин

Сташин В.В. и др.

С 78 Проектирование цифровых устройств на однокристальных микроконтроллерах/ В.В. Сташин, А.В. Урусов, О.Ф. Мологонцева. — М.: Энергоатомиздат, 1990. — 224 с.

ISBN 5-283-01543-2

Описаны структуры и системы команд однокристальных микроконтроллеров (МК) серии КМ1816. Приведены многочисленные примеры программ, реализующих типовые процедуры управления объектами и обработки данных. Даны рекомендации по выбору вариантов схем цифровых устройств. Изложена методика проектирования цифровых управляющих устройств на базе однокристальных МК.

Для инженеров-проектировщиков средств и систем автоматики, вычислительной техники и робототехники.

С 240202000-373
051 (01) 90 240-89

ББК 32.96:32.97

ISBN 5-283-01543-2

© Авторы, 1990

Предисловие

Развитие микроэлектроники и широкое применение ее изделий в промышленном производстве, в устройствах и системах управления самыми разнообразными объектами и процессами является в настоящее время одним из основных направлений научно-технического прогресса.

Использование микроэлектронных средств в изделиях производственного и культурно-бытового назначения не только приводит к повышению технико-экономических показателей изделий (стоимости, надежности, потребляемой мощности, габаритных размеров) и позволяет многократно сократить сроки разработки и отодвинуть сроки "морального старения" изделий, но и придает им принципиально новые потребительские качества (расширенные функциональные возможности, модифицируемость, адаптивность и т.д.).

За последние годы в микроэлектронике бурное развитие получило направление, связанное с выпуском однокристальных микроконтроллеров, которые предназначены для "интеллектуализации" оборудования различного назначения. Однокристальные (однокорпусные) микроконтроллеры представляют собой приборы, конструктивно выполненные в виде БИС и включающие в себя все составные части "голой" микроЭВМ: микропроцессор, память программ и память данных, а также программируемые интерфейсные схемы для связи с внешней средой. Использование микроконтроллеров в системах управления обеспечивает достижение исключительно высоких показателей эффективности при столь низкой стоимости (во многих применениях система может состоять только из одной БИС микроконтроллера), что микроконтроллерам, видимо, нет разумной альтернативной элементной базы для построения управляющих и/или регулирующих систем. К настоящему времени более двух третей мирового рынка микропроцессорных средств составляют именно однокристальные микроконтроллеры.

Отечественная микроэлектронная промышленность освоила широкомасштабный выпуск однокристальных микроконтроллеров, по существу представляющих собой особый класс вычислительной техники. К этому классу можно отнести: 4-битные микроконтроллеры серий 1814, 1820, 1829 и 1013; 8-битные микроконтроллеры серии 1816; микроконтроллеры сигнальные (аналоговые микропроцессоры) серии 1813.

Часто в публикациях микроконтроллеры называют однокристальными микроЭВМ. По нашему мнению, для этих приборов в большей степени подходит название МИКРОКОНТРОЛЛЕР, так как незначительная емкость памяти, физическое и логическое разделение памяти программ (ПЗУ) и памяти данных (ОЗУ), упрощенная и ориентированная на задачи управления система команд, примитивные методы адресации команд и данных, а также специфическая организация ввода/вывода информации предопределяют область их использования в качестве специализированных вычислителей, включенных в контур управления объектом или процессом. Структурная организация, набор команд и аппаратурно-программные средства ввода/вывода информации микроконтроллеров лучше всего приспособлены для решения задач управления и регулирования в приборах, устройствах и системах автоматики, а не для решения задач обработки данных. Микроконтроллеры не являются машинами классического "фон-неймановского" типа, так как физическая и логическая разделенность памяти программ и памяти данных исключает возможность модификации и/или замены (перегрузки) прикладных программ микроконтроллеров во время работы, что сильно затрудняет их использование в качестве универсальных средств обработки данных. Исходя из этих соображений, мы полагаем, что для всех приборов, отличающихся перечисленными признаками, наиболее точным является наименование МИКРОКОНТРОЛЛЕР (МК).

В данной книге в качестве базовых выбраны однокристальные микроконтроллеры, принадлежащие серии 1816. Ко времени написания книги в состав этой серии входят два типа микроконтроллеров: КМ1816ВЕ48 и КМ1816ВЕ51, обозначаемые далее для краткости МК48 и МК51 соответственно.

Предлагаемая вниманию читателей книга является первым книжным изданием на русском языке, в котором систематически излагаются вопросы проектирования цифровых устройств и систем на основе однокристальных 8-битных микроконтроллеров. В книге дается описание структурной организации и систем команд МК48 и МК51; приводятся многочисленные примеры программ, реализующих типовые процедуры управления объектами; излагается формализованная методика разработки цифровых устройств на основе микроконтроллеров и рассматриваются примеры процесса проектирования. Основное внимание в книге уделено обобщению опыта разработки цифровых устройств на микроконтроллерах и формализованной методике, регламентирующей последовательность действий при разработке прикладного программного обеспечения на самом сложном и слабоформализуемом этапе работы — "от постановки задачи к исходной программе".

Авторы выражают глубокую признательность рецензенту книги профессору И.И. Шагурину и научному редактору профессору А.Д. Иванникову за ценные советы и замечания, которые в большой степени способствовали улучшению материала.

Авторы будут благодарны всем читателям, которые сочтут возможным прислать свои замечания и пожелания по адресу: 113114, Москва, М-114, Шлюзовая наб., 10, Энергоатомиздат.

Авторы

Список сокращений, символических имен и аббревиатур

1. Русская нотация	
АЛУ	— арифметико-логическое устройство
БИС	— большая интегральная схема
ВК	— выбор корпуса (см. СЕ, англ.)
ВПД	— внешняя память данных
ВПП	— внешняя память программ
ВХПР	— вход приемника УАПП (см. RXD)
ВЫХПЕР	— выход передатчика УАПП (см. TXD)
ЭП	— управляющий сигнал записи (см. WR)
ЗПР	— запрос прерывания (см. INT)
КОП	— код операции (поле в теле команды)
МК	— микроконтроллеры серии 1816: КМ1816ВЕ48 и КМ1816ВЕ51. Для краткости: МК48 и МК51
ОЗУ	— оперативное запоминающее устройство (см. RAM)
ОРПП	— управляющий сигнал отключения резидентной памяти программ см. EA, англ.)
Порт BUS	— порт-расширитель МК48 (см. DBX)
ПП	— память программ
ПРОГ	— управляющий сигнал для программирования РПП (см. PROG)
РА	— регистр адреса (см. RAR)
РВВ	— БИС расширителя ввода/вывода
РВПП	— управляющий сигнал разрешения внешней памяти программ (см. PSEN)
РК	— регистр команд (см. IR)
РМП	— регистр маски прерывания (см. IE)
РП	— регистр приоритетов (см. IP)
РПД	— резидентная память данных
РПП	— резидентная память программ
РРТС	— регистр режима таймера/счетчика (см. TMOD)
РСФ	— регистры специальных функций (PSW, TMOD, TCON, SCON, PCON, IE, IP)
РУД	— регистр-указатель данных (см. DPTR)
РУМ	— регистр управления мощностью (см. PCON)
РУПП	— регистр управления последовательного порта (см. SCON)
РУС	— регистр-указатель стека (см. SP)
РУСТ	— регистр управления состояния таймера (см. TCON)
САВП	— управляющий сигнал строба адреса внешней памяти (см. ALE)
СБР	— управляющий сигналброса (см. RST)
СК	— счетчик команд (см. PC, англ.)
ССП	— слово состояния программы (см. PSW)
СТБ	— стробирующий сигнал (см. STB)
Т/С	— таймер/счетчик событий (см. TCNT)
УАПП	— универсальный асинхронный приемопередатчик (последовательный порт МК51)

- управляющий сигнал чтения (см. RD)
- управляющий сигнал пошагового (покомандного) режима работы (см. SS)

2. Английская нотация

A	- регистр-аккумулятор
AC	- вспомогательный перенос (Auxiliary Carry flag in PSW)
ACC	- символическое имя регистра A
ad	- прямой 8-битный адрес байта РПД (0–127), порта или РСФ
add	- прямой 8-битный адрес назначения
ads	- прямой 8-битный адрес источника
ad11	- прямой 11-битный адрес передачи управления
ad16	- прямой 16-битный адрес передачи управления
ad 16h	- старший байт прямого 16-битного адреса
ad16l	- младший байт прямого 16-битного адреса
ALE	- Address Latch Enable (см. CABII)
B	- регистр-расширитель аккумулятора
Bb	- обобщенное имя бита в аккумуляторе ($b = 0 \div 7$) МК 48
bbb	- 3-битное поле в КОП (МК48), специфицирующее бит аккумулятора
bit	- прямой 8-битный адрес бита (MK51)
BS	- переключатель банка регистров в MK48 (Bank Switch)
BUS	- порт-расширитель MK48
C	- флаг переноса
CE (CS)	- Chip Enable (Chip Select) (см. ВК, рус.)
CLK	- синхросигнал (Clock)
C/T	- управляющий бит выбора режима таймера/счетчика (Timer or Counter selector in TMOD)
#d	- 8-битный непосредственный операнд (константа)
#d16	- 16-битный непосредственный операнд (константа)
#d16h	- старший байт 16-битного непосредственного операнда
#d16l	- младший байт 16-битного непосредственного операнда
DBF	- выбор банка памяти в MK48 (Detect Bank Flag)
DB,X	- Data Bus; X = 0 \div 7 (см. Порт BUS)
DPH	- Data Pointer High (старший байт РУД)
DPL	- Data Pointer Low (младший байт РУД)
DPTR	- Data Pointer (см. РУД)
EA	- управляющий бит снятия блокировки всех прерываний (Enable All control bit in IE)
EA/VPP	- External Address/Voltage Power Programming (см. ОРПП)
EPROM	- Erasable Programmable Read Only Memory (см. РПП)
ES	- управляющий бит разрешения прерывания от УАПП (Enable Serial port control bit in IE)
ET	- управляющий бит разрешения прерывания от таймера (Enable Timer control bit in IE)
EX	- управляющий бит разрешения внешнего прерывания (Enable External interrupt control bit in IE)
F0, F1	- флаги, специфицируемые пользователем
GATE	- бит управления блокировкой Т/С (Gating control bit in TMOD)
GF0, GF1	- флаги пользователя (General Flags in PCON) в MK51
i	- бит в КОП, определяющий регистр косвенного адреса: i = 0,1 (R0, R1)
IDL	- управляющий бит холостого хода (Idle mode in PCON)
IE	- 1) Interrupt Enable register (см. РМП) 2) флаг внешнего прерывания, установленный по спаду с сигнала ЗПР (Interrupt Edge flag in TCON)

IP	- Interrupt Priority control register (см. РПГ)
INT	- Interrupt (см. ЗПР)
IR	- Instruction Register (см. РК, рус.)
IT	- управляющий бит выбора типа (уровень/спад) сигнала ЗПР (Interrupt Type control bit in TCON)
M0, M1	- управляющие биты выбора режима работы Т/С (Operating Mode in TMOD)
OV	- флаг переполнения (Overflow flag in PSW)
P	- флаг паритета (Parity flag in PSW)
PC	- Program Counter (см. СК, рус.)
PCON	- Power Control register (см. РУМ)
PD	- бит управления мощностью потребления (Power Done in PCON)
Pp	- обобщенное имя порта ввода/вывода в MK48 (p = 1,2 или 4–7)
pp	- 2-битное поле в коде операции, определяющее порт MK48: 00 – P4; 01 – P5, P1; 10 – P6, P2; 11 – P7
PROG	- Programming EPROM (см. ПРОГ)
PS	- управляющий бит приоритета УАПП (Serial port Priority control bit in IP)
PSEN	- Program Store Enable (см. РВПП)
PSW	- Program Status Word (см. ССП)
PT	- управляющий бит приоритета таймера (Timer Priority control bit in IP)
PX	- управляющий бит приоритета внешнего прерывания (External interrupt Priority control bit in IP)
PX,Y	- символическое имя бита Y порта X; Y = 0 \div 7 для MK48: X = 1, 2, BUS для MK51: X = 0, 1, 2, 3
RAM	- Random Access Memory (см. ОЗУ)
RAR	- RAM Address Register (см. РА, рус.)
RB8	- девятый (bit 8) принятый бит (Receive Bit 8 in SCON)
RD	- Read (см. ЧТ)
rel	- 8-битный относительный адрес передачи управления ($-127 \div +128$) в MK51
REN	- управляющий бит разрешения приема в УАПП (Receiver Enable control bit in SCON)
RI	- флаг прерывания от приемника (Received Interrupt flag in SCON)
Ri	- обобщенное имя регистра косвенного адреса (R0 или R1)
Rn	- обобщенное имя рабочего регистра (n = 0 \div 7)
rrr	- 3-битное поле в коде операции, определяющее регистр общего назначения (R0–R7)
RS	- управляющий бит выбора банка регистров (Register bank Select in PSW)
RST	- Reset (см. СБР)
RST/VPD	- Reset/Voltage Power Done (см. СБР)
RXD	- Receive Data pin (см. ВХПР)
Sn	- состояние устройства управления MK51, n = 1 \div 6 (State)
SCON	- Serial port Control/status register (см. РУПП)
SM0, SM1,	- управляющие биты режима работы УАПП (Serial port Mode control bits in SCON)
SM2	- управляющий бит двойной скорости передачи (Double Baud rate in PCON)
SMOD	- управляющий бит временных диаграмм и схемах, привязывающие сигналы к состояниям S устройства управления и фазам Р синхросигналов (X = 1 \div 6; Y = 1,2) для MK51
SXPY	- пояснения на временных диаграммах и схемах, привязывающие сигналы к состояниям S устройства управления и фазам Р синхросигналов (X = 1 \div 6; Y = 1,2) для MK51
SP	- Stack Pointer (см. РУС)
SS	- Single Step (см. ШАГ)
STB	- Strobe (см. СТБ)

- | | |
|--------|--|
| T0, T1 | - тест-входы МК |
| TB8 | - девятый (bit 8) передаваемый бит (Transmit Bit8 in SCON) |
| TCNT | - Timer/Counter events (см. Т/C, рус.) |
| TOON | - Timer/counter Control/status register (см. РУСТ) |
| TF | - флаг переполнения таймера (Timer overflow Flag in TCON) |
| TH | - старший байт таймера (Timer High byte) |
| TI | - флаг прерывания от передатчика (Transmit Interrupt flag in SCON) |
| TL | - младший байт таймера (Timer Low byte) |
| TMOD | - Timer/counter Mode register (см. РРТС, рус.) |
| TR | - управляющий бит пуска таймера (Timer Run control bit in TCON) |
| TXD | - Transmit Date (см. ВЫХПЕР) |
| WR | - Write (см. ЗП) |

3. Специальные символы

- | | |
|---------|--|
| ← | - оператор присваивания (замещения) |
| ↔ | - оператор взаимного обмена |
| Λ, ∨, ∄ | - операторы логических операций: И (конъюнкция), ИЛИ (дизъюнкция), исключающее ИЛИ |
| @ | - префикс косвенной адресации |
| # | - префикс непосредственного операнда |
| (Y) | - содержимое регистра или ячейки памяти с именем Y |
| ((Y)) | - содержимое ячейки памяти, адресуемой содержимым Y (косвенная адресация) |
| AND | - текущее содержимое счетчика команд МК |
| AND | - логическая операция "конъюнкция" при ассемблировании |
| B | - суффикс двоичного кода (Binary code) |
| H | - суффикс шестнадцатеричного кода (Hexadecimal code) |
| HIGH | - логическая операция выделения старшего байта из d76 при ассемблировании |
| LOW | - логическая операция выделения младшего байта из d16 при ассемблировании |
| NOT | - логическая операция "инверсия" при ассемблировании |
| OR | - логическая операция "дизъюнкция" при ассемблировании |

Особенности проектирования микроконтроллерных устройств управления объектами

1.1.

Введение в предметную область

Простейший в серии 1816 микроконтроллер MK48 имеет на кристалле (в корпусе БИС) следующие аппаратурные средства: процессор разрядностью 1 байт; стираемое программируемое ПЗУ программ емкостью 1 Кбайт; ОЗУ данных емкостью 64 байта; программируемый 8-битный таймер/счетчик; программируемые схемы ввода/вывода (27 линий); блок векторного прерывания от двух источников; генератор; схему синхронизации и управления.

Значительно более сложный и развитый MK51 имеет в своем составе такие аппаратурные средства: процессор, в состав которого входят 1-байтное АЛУ и схемы аппаратурной реализации команд умножения и деления; стираемое ПЗУ программ емкостью 4 Кбайта, ОЗУ данных емкостью 128 байт; два 16-битных таймера/счетчика; программируемые схемы ввода/вывода (32 линии); блок двухуровневого векторного прерывания от пяти источников; асинхронный канал дуплексного последовательного ввода/вывода информации со скоростью до 375 кбит/с; генератор, схему синхронизации и управления.

Структуры микроконтроллеров серии 1816 и их системы команд таковы, что в случае необходимости функционально-логические возможности контроллеров могут быть расширены. С использованием внешних дополнительных БИС постоянной и оперативной памяти адресное пространство МК может быть значительно расширено, а путем подключения различных интерфейсных БИС число линий связи МК с объектом управления может быть увеличено практически без ограничений.

Микроконтроллеры серии 1816 требуют одного источника электропитания напряжением +5 В ± 10%, рассеивают мощность около 1,5 Вт и работают в диапазоне температур от 0 до 70 °C. По входам и выходам МК серии 1816 электрически совместимы с интегральными схемами ТТЛ. MK48 и MK51 имеют различные системы команд и, следовательно, не обладают свойством программной совместимости на уровне объектных кодов. При этом они программно совместимы по принципу "снизу вверх" (MK48 → MK51) на уровне исходных текстов программ, написанных на языке ассемблера.

Таблица 1.1. Семейство МК серии 1816

Тип МК	Емкость резидентной памяти программ, Кбайт	Емкость резидентной памяти данных, байт	Частота синхронизации, МГц
KM1816BE48	СППЗУ 1	64	6
KM1816BE49	ПЗУ 2	128	11
KM1816BE35	—	64	11/6
KM1816BE51	СППЗУ 4	128	12
KM1816BE31	—	128	12

Микроконтроллер MK48 может работать в диапазоне частот синхронизации от 1 до 6 МГц, а минимальное время выполнения команды составляет 2,5 мкс. Микроконтроллер MK51 может работать в диапазоне частот от 1,2 до 12 МГц, при этом минимальный цикл выполнения команды равен 1 мкс, а быстродействие равно одному миллиону коротких операций в секунду.

Из такой краткой характеристики однокристальных МК серии 1816 видно, что эти приборы обладают значительными функционально-логическими возможностями и представляют собой эффективное средство компьютеризации (автоматизации на основе применения средств и методов обработки данных и цифрового управления) разнообразных объектов и процессов.

Семейство МК серии 1816 имеет в своем составе различные модификации, отличающиеся друг от друга признаками, которые перечислены в табл. 1.1.

Анализ основных признаков МК серии 1816 показывает, что MK48 и MK51 целесообразно использовать на этапе опытно-конструкторской разработки и отладки систем, а также в малосерийных изделиях. Микроконтроллер MK49 имеет масочное ПЗУ программ, и поэтому его следует применять в крупносерийных изделиях. Микроконтроллеры, в которых нет резидентной памяти программ, используют, как правило, не в конечных изделиях, а в автономных отладочных устройствах и многофункциональных программируемых контроллерах, где в качестве памяти программ и данных используются внешние БИС и имеются средства загрузки программ.

1.2.

Основные положения

В устройствах управления объектами (контроллерах) на основе МК аппаратурные средства и программное обеспечение существуют в форме неделимого аппаратурно-программного комплекса. При проектировании контроллеров приходится решать одну из самых сложных задач разработки, а именно задачу оптимального распределения функций контроллера между аппаратурными средствами и программным обеспечением. Решение этой задачи осложняется тем, что взаимосвязь и взаимовлияние аппаратурных средств и программного обеспечения в микропроцессорной технике претерпевают динамичные изменения. Если в начале разви-

тия МП-техники определяющим было правило, в соответствии с которым аппаратурные средства обеспечивают производительность, а программное обеспечение – дешевизну изделия, то в настоящее время это правило нуждается в серьезной корректировке. Так как МК представляет собой стандартный массовый (относительно недорогой) логический блок, конкретное назначение которого определяет пользователь с помощью программного обеспечения, то с ростом степени интеграции и, следовательно, функционально-логических возможностей МК резко понижается стоимость изделия в пересчете на выполняемую функцию, что в конечном итоге и обеспечивает достижение высоких технико-экономических показателей изделий на МК. При этом затраты на разработку программного обеспечения изделия в 2–10 раз превышают (за время жизни изделия) затраты на приобретение и изготовление аппаратурных средств.

В настоящее время наибольшее распространение получили методологический прием, при котором весь цикл разработки контроллеров рассматривается как последовательность трех фаз проектирования:

- 1) анализа задачи и выбора (и/или разработки) аппаратурных средств контроллера;
- 2) разработки прикладного программного обеспечения;
- 3) комплексирования аппаратурных средств и программного обеспечения в прототипе контроллера и его отладки.

Фаза разработки программного обеспечения, т.е. фаза получения прикладных программ, в свою очередь, разбивается на два существенно различных этапа:

- 1) "от постановки задачи к исходной программе";
- 2) "от исходной программы к объектному модулю".

Этап разработки "от исходной программы к объектному модулю" имеет целью получение машинных кодов прикладных программ, работающих в МК. Этот этап разработки прикладного программного обеспечения легко поддается формализации и поддержан всей мощью системного программного обеспечения МК, направленного на автоматизацию процесса получения прикладных программ. В состав средств системного программного обеспечения входят трансляторы с различных алгоритмических языков высокого уровня, ассемблеры, редакторы текстов, программы-отладчики, программы-документаторы и т.д. Наличие всех этих системных средств придает инженерной работе на этом этапе проектирования контроллеров характер ремесла, а не инженерного творчества. Так как в конечном изделии (контроллере) имеются только "голый" МК и средства его сопряжения с объектом, то выполнять отладку разрабатываемого прикладного программного обеспечения на нем невозможно (из-за отсутствия средств ввода, вывода, ОЗУ большой емкости и операционной системы), и, следовательно, разработчик вынужден обращаться к средствам вычислительной техники для выполнения всех формализуемых стадий разработки: трансляции, реактивации, отладки, загрузки объектных кодов в программируемую постоянную память МК. Попутно отметим, что системные средства авто-

матизации разработки прикладных программ МК на этапе "от исходной программы к объектному модулю" широко распространены и существуют в среде операционных систем микроЭВМ "Электроника-60", СМ-1800, СМ-1300 и присутствуют в операционных системах персональных компьютеров типа ЕС-1840, ЕС-1841.

Совсем по-другому выглядит инженерный труд на этапе разработки программного обеспечения "от постановки задачи к исходной программе", так как он практически не поддается формализации и, следовательно, не может быть автоматизирован. Проектная работа здесь носит творческий характер, изобилует решениями, имеющими "волевую" или "вкусовую" окраску, и решениями, продиктованными конъюнктурными соображениями. В силу перечисленных обстоятельств именно на этапе проектирования "от постановки задачи к исходной программе" разработчик сталкивается с наибольшим количеством трудностей.

Качество получаемого прикладного программного обеспечения контроллера всецело зависит от уровня проектных решений, принятых на этапе разработки "от постановки задачи к исходной программе". Уровень проектных решений в свою очередь из-за отсутствия теории проектирования программируемых контроллеров определяется только опытом, квалификацией и интуицией разработчика. Однако накопленный опыт убеждает в том, что *систематический подход к процессу разработки прикладных программ для контроллеров обеспечивает достижение хороших результатов даже начинающими разработчиками*.

1.3.

Структура МК-системы управления

Типовая структура МК-системы управления показана на рис. 1.1 и состоит из объекта управления, микроконтроллера и аппаратуры их взаимной связи.

Микроконтроллер путем периодического опроса осведомительных слов (ОС) генерирует в соответствии с алгоритмом управления последовательности управляющих слов (УС). Осведомительные слова это сигналы состояния объекта (ОС), сформированные датчиками объекта управления, и флаги. Выходные сигналы датчиков вследствие их различной физической природы могут потребовать промежуточного преобразования на аналого-цифровых преобразователях (АЦП) или на схемах формирователей сигналов (ФС), которые чаще всего выполняют функций гальванической развязки и формирования уровней двоичных сигналов стандарта ТТЛ.

Микроконтроллер с требуемой периодичностью обновляет управляющие слова на своих выходных портах. Некоторая часть управляющего слова интерпретируется как совокупность прямых двоичных сигналов управления (СУ), которые через схемы формирователей сигналов (усилители мощности, реле, оптраны и т.п.) поступают на исполнительные механизмы (ИМ) и устройства индикации. Другая часть управляющего слова представляет собой упакованные двоичные коды, которые

через цифро-аналоговые преобразователи (ЦАП) воздействуют на исполнительные механизмы аналогового типа. Если объект управления использует цифровые датчики и цифровые исполнительные механизмы, то наличие ЦАП и АЦП в системе не обязательно.

В состав аппаратуры связи, которая как правило, строится на интегральных схемах серии ТТЛ, входит регистр флагов, на котором фиксируется некоторое множество специфицируемых признаков как объекта управления, так и процесса работы контроллера. Этот регистр флагов используется в качестве аппаратурного средства реализации механизма взаимной синхронизации относительно медленных и вероятностных процессов в объекте управления и быстрых процессов в контроллере.

Регистр флагов доступен как контроллеру, так и датчикам. Вследствие этого он является удобным местом фиксации сигналов ГОТОВ/ОЖИДАНИЕ при передачах с квитированием или сигналов ЗАПРОС ПРЕРЫВАНИЯ/ПОДТВЕРЖДЕНИЕ при взаимодействии контроллера и объекта в режиме прерывания. Если МК-система имеет многоуровневую систему прерываний, то регистр флагов содержит схему упорядочивания приоритетов.

Для аппаратурной реализации временных задержек, формирования сигналов требуемой частоты и скважности в состав аппаратуры связи включают программируемые интерваловые таймеры в том случае, если их нет в составе МК или их число недостаточно.

Законы функционирования МК-системы управления со структурой, показанной на рис. 1.1, всецело определяются прикладной программой, размещаемой в резидентной памяти программ МК. Иными словами, специализация контроллера типовой структуры на решение задачи управления конкретным объектом осуществляется путем разработки прикладных программ МК и аппаратуры связи МК с датчиками и исполнительными механизмами объекта.

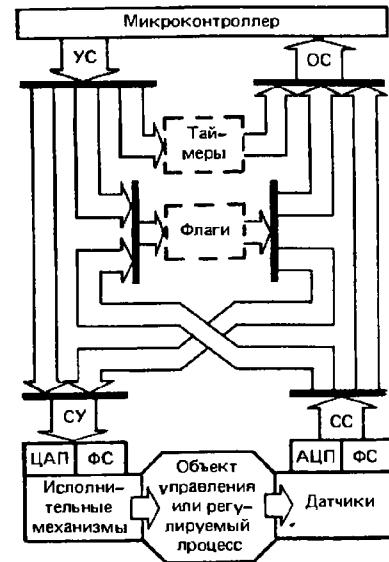


Рис. 1.1. Структура цифровой системы управления на основе МК

1.4.

Особенности разработки аппаратурных средств МК-систем

Применение однокристальных МК в устройствах управления объектами привело к кардинальным изменениям в разработке аппаратурных средств устройств и систем. И дело здесь заключается в следующем. Микроконтроллер представляет собой логический автомат с высокой степенью детерминированности, который допускает очень немного вариантов его системного включения. Именно поэтому типовой состав аппаратурных средств ядра любой МК-системы (МК, ППЗУ, ОЗУ, интерфейсные БИС, схемы синхронизации и системного управления) оформляется конструктивно в виде одноплатных универсальных программируемых контроллеров, которые предназначены для встраивания в контур управления объектом или процессом. На печатной плате такой МК-системы имеются гнезда для установки БИС пользователя. На некоторых моделях таких плат имеется еще и так называемое "монтажное поле пользователя", на котором он имеет возможность смонтировать свои специфические схемы, такие как оптронные развязки, АЦП, ЦАП, реле и т.п. Кроме того, на плате МК-системы может быть размещен источник электропитания. Если разработчик положит в основу проектируемого изделия такой одноплатный контроллер, то он будет избавлен от необходимости разрабатывать и сопровождать конструкторской и эксплуатационной документацией самую сложную, центральную часть изделия. Как известно, вес документации на систему автоматического управления примерно на порядок превышает вес самой системы, а изготовление и согласование этой документации растягивается на годы. При использовании стандартного контроллера в качестве комплектующего системного элемента объем документации на аппаратурные средства изделия многократно сокращается и может содержать только документацию на аппаратуру сопряжения ядра системы с датчиками и исполнительными механизмами объекта управления.

В результате этих структурных сдвигов *объем трудозатрат на разработку аппаратурных средств изделия постоянно уменьшается по отношению к суммарным затратам на разработку и отладку прикладного программного обеспечения*. Объем проектной документации на аппаратурные средства изделия постоянно уменьшается из-за все более широкого применения в аппаратуре сопряжения интегральных ЦАП, АЦП и интерфейсных БИС. Кроме того, во множестве конкретных применений для реализации ядра системы оказывается достаточно средств, содержащихся в единственный БИС микроконтроллера. Объем документации на программное обеспечение изделия имеет устойчивую тенденцию роста из-за стремления разработчиков использовать все более совершенные и, следовательно, более сложные алгоритмы управления. В этих условиях основным проектным документом на изделие становится листинг (распечатка исходного текста и машинных кодов) прикладной программы МК.

Таким образом, освященный традициями стереотип мышления, суть

которого сводится к лозунгу "главное – это разработать аппаратуру, а программу потом "прилепим", оказывается не только неверным, но и гарантирует неудачу разработки. Появление однокристальных МК иллюстрирует тот факт, что все более сложные функционально насыщенные части аппаратурных средств контроллеров в процессе интеграции переходят из разряда подсистем в разряд комплектующих изделий. Так как эти комплектующие изделия являются сложно организованными приборами, функционирующими под управлением программ, то удельный вес прикладного программного обеспечения МК-систем имеет устойчивую тенденцию к увеличению, а удельный вес аппаратурных средств – к снижению.

1.5.

Особенности разработки прикладного программного обеспечения МК-систем

Как уже отмечалось, при проектировании МК-систем прежде всего возникает необходимость решения задачи об оптимальном (по ряду критериев) распределении функций между аппаратурными средствами и программным обеспечением.

При этом в самом общем случае необходимо исходить из того, что использование специализированных интерфейсных БИС упрощает разработку и обеспечивает высокое быстродействие системы в целом, но сопряжено с увеличением стоимости, объема и потребляемой мощности. Большой удельный вес программного обеспечения позволяет сократить число компонентов системы и стоимость ее аппаратурных средств, но это приводит к снижению быстродействия и увеличению затрат и сроков разработки и отладки прикладных программ. При этом еще может несколько увеличиться и число БИС внешней памяти МК-системы. Решение о выборе того или иного варианта распределения функций между аппаратурными и программными средствами системы принимается в зависимости от тиражности изделия, ограничений по стоимости, объему, потребляемой мощности и быстродействию изделия. Попутно отметим, что время жизни изделия, в котором большая часть функций реализована в программном обеспечении, многократно возрастает за счет того, что срок "морального старения" изделия может быть существенно отодвинут. Программная реализация основных элементов алгоритма работы контроллера допускает его модификацию относительно простыми средствами (путем перепрограммирования), в то время как возможность изменения уже существующей фиксации элементов алгоритма в аппаратуре контроллера практически отсутствует.

Опыт проектирования МК-систем свидетельствует, что имеет место неподготовленность значительной части потенциальных пользователей МК к переходу на принципиально новую элементную базу, отсутствие навыков программной реализации тех функций, которые в данной предметной области профессиональных знаний, в конкретной прикладной задаче традиционно выполнялись на основе аппаратурных средств.

Отчасти такое положение объясняется отсутствием общей теории проектирования программируемых контроллеров, и это в свою очередь приводит к необходимости использовать программистов высокой квалификации для программирования и сопровождения простейших программ. Довольно распространенная практика работы "тандемом", когда над разработкой прикладных программ для МК совместно работают *профессиональный программист и непрограммирующий профессионал*, т.е. специалист, владеющий "тайнами ремесла" в конкретной предметной области, имеет серьезным недостатком то, что при попытках изложить программисту смысл прикладной задачи этот смысл зачастую ускользает. В результате такой практики формализуются и программируются наиболее очевидные, грубо говоря – тривиальные, прикладные задачи, а наиболее профессионально интересные остаются вне пределов досягаемости. Видимо, это объясняется тем, что время, необходимое на формализацию профессиональных знаний при работе "тандемом", нередко составляет до 70% всего времени, требующегося для получения за конченного микроконтроллерного изделия.

Работа "тандемом" в огромном большинстве случаев приводит к тому, что конечный пользователь МК-системы отказывается от своих ранее сформулированных требований на программу и утверждает, что "имелось в виду нечто похожее, но не это". Такое положение скорее всего объясняется тем, что начало процесса программирования задач, которые ставят конечный пользователь, немедленно изменяет его собственное представление об этих задачах. Отметим попутно, что до 60% ошибок прикладных программ МП-контроллеров вызваны не ошибками в машинных кодах, не логическими ошибками в программе, а ошибочной формализацией прикладной задачи. Трудоемкость устранения этих ошибок, наработанных "тандемом" (профессиональный программист – непрограммирующий профессионал), столь велика, что зачастую вынуждает приступить к разработке прикладной программы МК-системы заново и с иными средствами.

Ресурсы, затрачиваемые собственно на программирование, т.е. на получение машинных кодов, столь незначительны по сравнению с ресурсами, затрачиваемыми на процесс формализации прикладной задачи и разработку алгоритма, что следует говорить *не о проблеме разработки прикладного программного обеспечения МК-систем, а о проблеме формализации профессиональных знаний конечного пользователя микроконтроллерных изделий*.

Подобно тому как появление микропроцессорных и микроконтроллерных средств привело к продолжающемуся процессу *перемещения основного объема затрат на проектирование контроллеров из сферы разработки аппаратурных средств в сферу разработки программного обеспечения*, так и стремительное расширение возможных областей применения МК приводит к *перемещению центра тяжести усилий по разработке прикладного программного обеспечения с фазы реализации на фазу постановки и формализации задачи*.

Сложившаяся к настоящему времени структура трудозатрат в разработке МК-систем позволяет выделить три основные стадии проектирования прикладного программного обеспечения:

- 1) анализ предметной области с целью определения задач, автоматизация решения которых на основе МК обещает наибольший эффект;
- 2) разработку алгоритма решения поставленной задачи (или комплекса задач);
- 3) собственно программирование, или, точнее, сопровождение разработки прикладных программ системными средствами поддержки проектирования.

Распределение трудозатрат в процентах по этим трем стадиям выглядит примерно так: 40–50–10. Это означает, что если первая стадия работы уже выполнена с участием специалиста по системному анализу, т.е. если задача уже поставлена, то наиболее сложной, слабоформализуемой (из-за тесной связности с областью приложения будущей программы) и трудоемкой стадией работы является стадия анализа задачи, ее инженерной интерпретации и разработки "функциональной спецификации" программы для формирования алгоритма решения поставленной задачи. Вся последующая работа по преобразованию алгоритма в машинные коды, т.е. создание прикладного программного обеспечения, представляет собой просто совокупность процессов трансляции. Эти процессы хорошо формализуемы, и их реализация опирается на уже существующие системные средства поддержки (трансляторы, редакторы, отладчики). Именно вследствие этого собственно программирование требует только около 10% общих трудозатрат. Очевидно, что основную творческую нагрузку при разработке прикладных программ для МК-систем несет не профессиональный программист, а непрограммирующий профессионал – специалист в данной области знаний. Если этот специалист овладеет основами программирования и станет *программирующим профессионалом*, то можно ожидать, что процесс формализации его профессиональных знаний будет протекать много результ ativнее, чем при "игре в испорченный телефон", т.е. при алгоритмизации прикладной задачи "тандемом".

Ориентация на разработку прикладных программ для МК-систем силами программирующих профессионалов получает распространение еще и потому, что в условиях быстро дешевеющей памяти изменились стиль и технология разработки программ. Экономят теперь уже не память МК-системы, а время разработчика программного обеспечения, т.е. сокращают сроки разработки изделия. Вследствие этого прикладные программы, созданные программирующим профессионалом, с точки зрения профессионального программиста зачастую выглядят неуклюжими и неизящными. Но зато они обладают одним общим достоинством – они действительно работают в контроллерах, чего нельзя сказать о девяти из каждых десяти изящных программ, созданных профессиональным программистом, не могущим (по определению) быть профессионалом и в каждой конкретной предметной области знаний.

С учетом масштабов выпуска и перспектив применения средств микроконтроллерной техники выход из создавшегося положения нам видится в том, чтобы побудить специалистов, работающих в своих предметных областях знаний, *взять дело разработки прикладного программного обеспечения МК-систем в свои руки полностью* (при некоторой технической поддержке профессиональных программистов). Для этого нужны не очень значительные усилия и первоначальные затраты: надо прежде всего решиться взять всю полноту ответственности за программное обеспечение на свои плечи, надо овладеть одним из языков программирования и освоить "кухню" программной реализации ограниченного множества наиболее употребимых процедур и функций данной предметной области. Вслед за этим потребуется, конечно, выработать навыки алгоритмизации более сложных процедур и функций объекта автоматизации. Однако эта задача относительно легко решается с использованием метода декомпозиции (разбиения сложной функции на множество простых взаимосвязанных функций). К подобной постановке вопроса организации разработки прикладного программного обеспечения для МК-систем приводит и вполне очевидное соображение о том, что быстрый рост числа выпускаемых МК и областей их проблемных применений не может более сопровождаться соответствующим ростом числа программистов.

Глава 2

Структурная организация и система команд микроконтроллера КМ1816ВЕ48

2.1.

Функциональное назначение выводов корпуса МК48

Микроконтроллер конструктивно выполнен в корпусе БИС с 40 внешними выводами. Все выводы электрически совместимы с элементами ТТЛ: входы представляют собой единичную нагрузку, а выходы могут быть нагружены одной ТТЛ-нагрузкой. Цоколевка корпуса МК48 показана на рис. 2.1. Ниже приводятся символические имена выводов корпуса и даются краткие пояснения их назначения:

- | | |
|---------------|--|
| ОБЩ (VSS) | — потенциал земли; |
| +5В.ОЧ (VCC) | — основное напряжение питания +5 В; подается во время работы и при программировании РПП; |
| +5В.ДОП (VDD) | — дополнительное напряжение питания +5 В; во время работы обеспечивает электропитание только для РПД; на этот вывод при программировании РПП подается уровень +25 В; |
| ПРОГ (PROG') | — вход для подачи программирующего импульса +25 В при загрузке РПП; выход стробирующего сигнала для БИС расширителя ввода/вывода; |
| X1 | — вход для подключения вывода кварцевого резонатора или вход для сигнала от внешнего источника синхронизации; |
| X2 | — вход для подключения второго вывода резонатора; |
| СБР (RST) | — вход сигнала общего сброса при запуске МК; сигнал 0 при программировании и проверке РПП; |
| ШАГ (SS) | — сигнал, который совместно с сигналом САВП позволяет при отладке выполнять программу с остановом после исполнения очередной команды; |

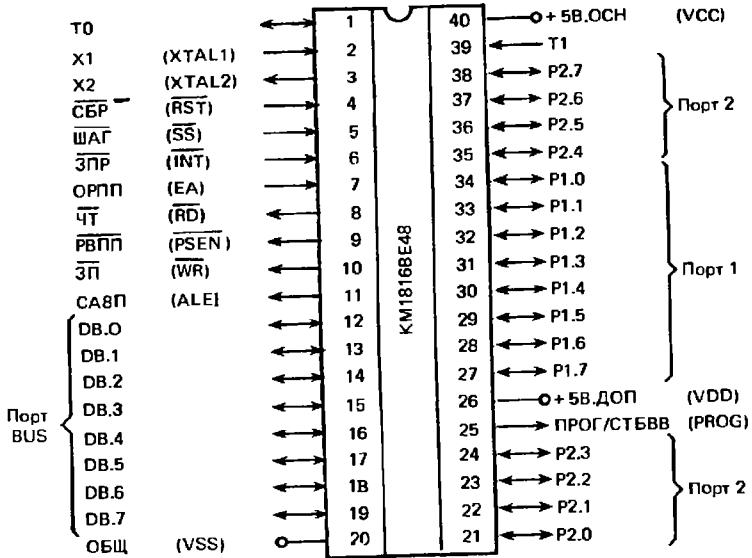


Рис. 2.1. Цоколевка корпуса МК48

PBПП (PSEN)

- разрешение внешней памяти программ; сигнал выдается только при обращении к внешней памяти программ;

САВП (ALE)

- строб адреса внешней памяти; сигнал используется для приема и фиксации адреса внешней памяти на внешнем регистре; сигнал является идентификатором машинного цикла, так как всегда выводится из МК с частотой, в 5 раз меньшей основной частоты синхронизации;
- стробирующий сигнал при чтении из внешней памяти данных или УВВ;
- стробирующий сигнал при записи во внешнюю память данных или УВВ;

ЧТ (RD)

- входной сигнал, опрашиваемый по командам условного перехода JT0 и JNT0; кроме того, используется при программировании РПП; может быть использован для вывода сигнала синхронизации после команды ENTO CLK;
- входной сигнал, опрашиваемый командами условного перехода JT1 и JNT1; кроме того, используется в качестве входа внутреннего счетчика внешних событий после исполнения команды STRT CNT;

ЗП (WR)

T0

T1

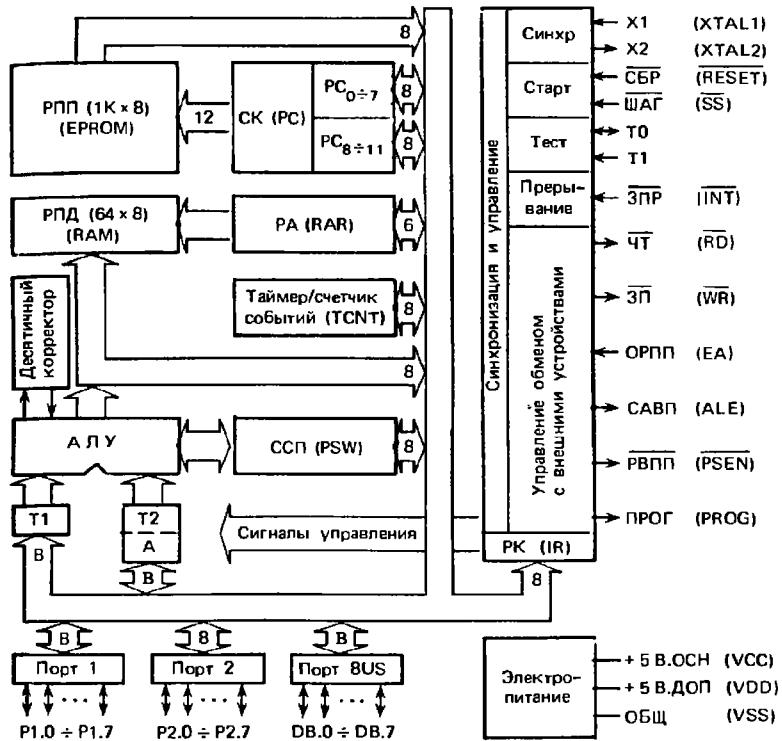


Рис. 2.2. Структурная схема МК48

ЗИР (INT)

- сигнал запроса прерывания от внешнего источника; вызывает подпрограмму обслуживания прерывания, если прерывание разрешено ранее по команде EN I; сигнал СБР запрещает прерывания;

ОРПП (EA)

- отключение РПП; уровень 1 на этом входе заставляет МК выполнять выборку команд только из внешней памяти программ; используется при тестировании прикладной программы и отладке МК; 25 В при программировании РПП;

Порт 1 (P1)

- 8-битный квазидвунаправленный порт ввода/вывода информации; каждый разряд порта может быть запрограммирован на ввод или вывод;

Порт 2 (P2)

- 8-битный квазидвунаправленный порт ввода/вывода информации; каждый разряд порта может быть запрограммирован на ввод или вывод; биты P2₀₋₃ во время чтения из ВПП содержат старшие четыре бита счетчика команд СК₈₋₁₁; используются для подключения БИС расширителя ввода/вывода (порты P4-P7);
- 8-битный двунаправленный порт ввода/вывода информации; может быть отключен от нагрузки; может выполнять прием и выдачу байтов синхронно с сигналами ЧТ и ЗП; при обращении к ВПП содержит 8 младших бит счетчика команд и затем по сигналу РВПП принимает выбранную команду; при обращении к ВПД содержит младшие 8 бит адреса синхронно с сигналом САВП и байт данных синхронно с сигналами ЧТ или ЗП.

2.2.

Структурная схема МК48

На рис. 2.2. показана структурная схема МК48. Основу структуры МК образует внутренняя двунаправленная 8-битная шина, которая связывает между собой все устройства БИС: арифметико-логическое устройство (АЛУ), устройство управления, память и порты ввода/вывода информации. Рассмотрим последовательно основные элементы структуры и особенности организации МК.

2.2.1. Арифметико-логическое устройство

В состав АЛУ входят следующие блоки: комбинационная схема обработки байтов, регистры T, регистр-аккумулятор A, схема десятичного корректора и схема формирования признаков. Аккумулятор используется в качестве регистра операнда и регистра результата. Регистр временного хранения операнда T1 программно недоступен и используется для временного хранения второго операнда при выполнении двухоперандных команд. Комбинационная схема АЛУ может выполнять следующие операции: сложение байтов с переносом или без него; логические операции И, ИЛИ и исключающее ИЛИ; инкремент, декремент, инверсию; циклический сдвиг влево, вправо через (или минуя) флаг переноса, обмен тетрад в байте; десятичную коррекцию содержимого аккумулятора.

При выполнении операций обработки данных в АЛУрабатываются флаги (признаки), которые (за исключением флага переноса С) формируются на комбинационной схеме и не фиксируются на триггерах. К таким флагам относятся флаг нулевого содержимого аккумулятора и флаг наличия единицы в селектируемом бите аккумулятора. Логика условных переходов по указанным флагам позволяет выполнять команды передачи управления (JZ, JNZ, JB0-JB7) без их фиксации на триггерах.

Порт BUS (DB)

Флаги переноса и вспомогательного переноса (перенос из младшей тетрады в старшую) фиксируются на триггерах, входящих в состав регистра слова состояния программы (CCP). Формат CCP показан на рис. 2.3. Кроме перечисленных признаков логика условных переходов МК оперирует флагами F0 и F1, функциональное назначение которых определяется разработчиком; флагом переполнения таймера TF, сигналами на входах Т0 и Т1. Программистом могут быть также использованы флаги рабочего банка регистров BS и выбранного банка внешней памяти программ MB. Кроме того, логикой переходов после окончания каждого машинного цикла опрашивается еще один флаг, а именно флаг разрешения/запрета прерываний.

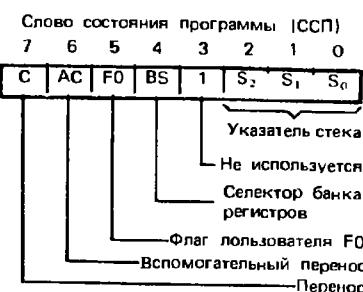


Рис. 2.3. Формат слова состояния программы

2.2.2. Память микроконтроллера

Память программ. Память программ и память данных в МК48 физически и логически разделены. Память программ реализована в резидентном СППЗУ емкостью 1 Кбайт. Максимальное адресное пространство, отводимое для программ, составляет 4 Кбайта. Счетчик команд (СК) содержит 12 бит, но инкрементируются в процессе счета только младшие 11 бит. Поэтому счетчик команд из предельного состояния 7FFF (если только по этому адресу не расположена команда передачи управления) перейдет в состояние 000H. Состояние старшего бита счетчика команд может быть изменено специальными командами (SEL MBO, SEL MB1). Подобный режим работы счетчика команд позволяет создать два банка памяти емкостью по 2 Кбайта каждый. Карта адресов памяти программ показана на рис. 2.4.

В резидентной памяти программ имеется три специализированных адреса:

адрес 0, к которому передается управление сразу после окончания сигнала СБР; по этому адресу должна находиться команда безусловного перехода к началу программы;

адрес 3, по которому расположен вектор прерывания от внешнего источника;

адрес 7, по которому расположены вектор прерывания от таймера или начальная команда подпрограммы обслуживания прерывания по признаку переполнения таймера/счетчика.

Память программ разделяется не только на банки емкостью 2 Кбайта, но и на страницы по 256 байт в каждой. В командах условного перехода задается 8-битный адрес передачи управления в пределах текущей стра-

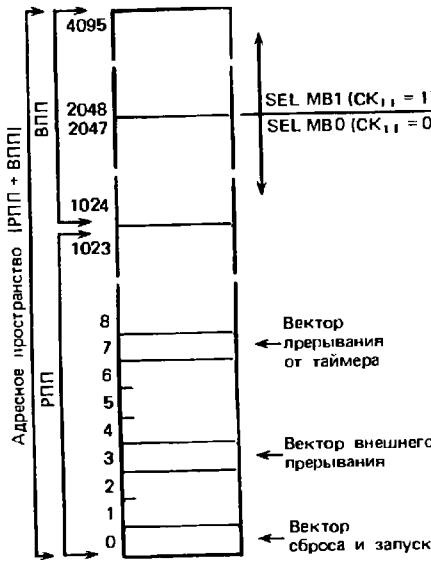


Рис. 2.4. Карта адресов памяти программ

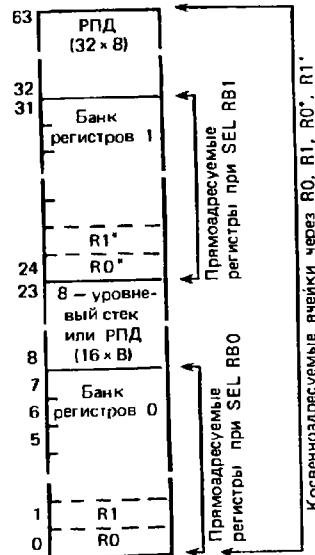


Рис. 2.5. Карта адресов РПД

ници. В случае, когда в программе необходимо иметь много переходов по условию, из-за небольшого размера страницы возникает проблема размещения соответствующих программных модулей в границах страницы. Команда вызова подпрограмм модифицирует 11 бит счетчика команд, обеспечивая тем самым межстраничные переходы в пределах выбранного банка памяти программ.

В МК-системе, работающей с внешней памятью программ, возникает проблема размещения подпрограмм в двух банках памяти. Проблема эта связана с тем, что МК не имеет средств считывания и анализа флага MB, равного содержимому старшего бита счетчика команд CK₁₁. Поэтому в каждый текущий момент исполнения программы, состоящей из потока вызовов подпрограмм, нет возможности определения номера банка памяти, из которого осуществляется выборка. Так как переходы между банками выполняются только по командам SEL MB, необходимо следить за тем, чтобы подпрограммы, взаимно вызывающие друг друга, располагались в пределах одного банка памяти. В противном случае возникает необходимость модификации признака MB в вызываемой подпрограмме и восстановления его при возврате в вызывавшую подпрограмму. Но если вызов такой подпрограммы носит условный характер, то проблема восстановления может оказаться неразрешимой.

При обработке запросов прерываний в МК48 старший бит счетчика команд CK₁₁ принудительно устанавливается в 0. Это приводит к необ-

ходимости подпрограмму обслуживания прерывания и все подпрограммы, вызываемые ею, размещать в пределах банка памяти 0.

Память данных. Резидентная память данных емкостью 64 байта имеет в своем составе два банка рабочих регистров 0–7 и 24–31 по восемь регистров в каждом. Выбор одного из банков регистров выполняется по команде SEL RB. Рабочие регистры доступны по командам с прямой адресацией, а все ячейки РПД доступны по командам с косвенной адресацией. В качестве регистров косвенного адреса используются регистры R0, R1 и R0*, R1* (рис. 2.5.).

Ячейки РПД с адресами 8–23 адресуются указателем стека из ССП и могут быть использованы в качестве 8-уровневого стека. В случае, если уровень вложенности подпрограмм меньше восьми, незадействованные в стеке ячейки могут использоваться как ячейки РПД. При переполнении стека регистр-указатель стека, построенный на основе 3-битного счетчика, переходит из состояния 7 в состояние 0. Малая емкость стека ограничивает число возможных внешних источников прерывания в МК-системе. МК48 не имеет команд загрузки байта в стек или его извлечения из стека, и в нем фиксируются только содержимое счетчика команд и старшая четверть ССП (флаги). В силу этого разработчику необходимо следить за тем, чтобы вложенные подпрограммы не использовали одни и те же рабочие регистры.

Практически все команды с обращением к РПД оперируют с одним байтом. Однако по командам вызова и возврата осуществляется доступ к двухбайтным словам. В памяти данных слова хранятся так, что старший байт слова располагается в ячейке с большим адресом. Отметим, что в памяти программ порядок расположения байтов по старшинству при хранении двухбайтных слов обратный.

В МК-системах, где используется внешнее ОЗУ, через регистры косвенного адреса R0 и R1 возможен доступ к ВПД емкостью 256 байт.

2.2.3. Организация ввода/вывода информации

Для связи МК48 с объектом управления, для ввода и вывода информации используются 27 линий. Эти линии сгруппированы в три порта по восемь линий в каждом и могут быть использованы для вывода, ввода или для ввода/вывода через двунаправленные линии. Кроме портов ввода/вывода имеются три линии, сигналы на которых могут изменять ход программы по командам условного перехода: линия ЗПР используется для ввода в МК сигнала запроса прерывания от внешнего источника; линия T0 используется для ввода тестирующего сигнала от двоичного датчика объекта управления; кроме того, под управлением программы (по команде ENTO CLK) по этой линии из МК может выдаваться сигнал синхронизации; линия T1 используется для ввода тестирующего сигнала или в качестве входа счетчика событий (по команде STRT CNT).

Порты ввода/вывода P1 и P2. Специальная схемотехника портов P1 и P2, которая получила название квазидвунаправленной, позволяет вы-

полнять ввод, вывод или ввод/вывод. Каждая линия портов P1 и P2 может быть программным путем настроена на ввод, вывод или на работу с двунаправленной линией передачи. Для того чтобы настроить некоторую линию на режим ввода в МК, необходимо перед этим в буферный триггер этой линии записать 1. Сигнал СБР автоматически записывает во все линии портов P1 и P2 сигнал 1. Квазидвунаправленная структура портов P1 и P2 для программиста МК 1816 специфична тем, что в процессе ввода информации выполняется операция логического И над вводимыми данными и текущими (последними) выведенными данными. Квазидвунаправленные схемы портов P1 и P2 и команды логических операций ANL и ORL предоставляют разработчику эффективное средство маскирования для обработки однобитных входных и выходных переменных.

В системе команд МК есть команды, которые позволяют выполнять запись нулей и единиц в любой разряд или группу разрядов порта, но так как в этих командах маска задается непосредственным операндом, то необходимо знать распределение сбрасываемых и устанавливаемых линий на этапе разработки прикладной программы. В том случае, если маска вычисляется программой и заранее не известна, в ОЗУ необходимо иметь копию состояния порта вывода. Эта копия по командам логических операций объединяется с вычисляемой маской в аккумуляторе и затем загружается в порт. Необходимость этой процедуры вызвана тем, что в МК отсутствует возможность выполнить операцию чтения значений портов Р1 и Р2 для определения прежнего состояния порта вывода. Порт Р2 отличается от порта Р1 тем, что его младшие четыре бита могут быть использованы для расширения МК-системы по вводу/выводу. Через младшую четверть порта Р2 по специальным командам обращений возможен доступ к четырем внешним четырехбитным портам ввода/вывода Р4–Р7. Работа этих внешних портов синхронизируется сигналом ПРОГ.

Порт ввода/вывода BUS представляет собой двунаправленный буфер с тремя состояниями и предназначен для побайтного ввода, вывода или ввода/вывода информации. Если порт BUS используется для двунаправленных передач, то обмен информацией через него выполняется по командам MOVX. При выводе байта генерируется стробирующий сигнал ЗП, а выводимый байт фиксируется в буферном регистре. При вводе байта генерируется стробирующий сигнал ЧТ, но вводимый байт в буферном регистре не фиксируется. В отсутствие передач порт BUS по своим выходам находится в высокомпелдансном состоянии. Если порт BUS используется как однонаправленный, то вывод через него выполняется по команде OUTL, а ввод – по команде INS.

Вводимые и выводимые через порт BUS байты можно маскировать с помощью команд ORL и ANL, что позволяет выделять и обрабатывать в байте отдельный бит или группу бит.

В МК-системах простой конфигурации, когда порт **BUS** не используется в качестве порта-расширителя системы, обмен выполняется по командам **INS**, **OUTL** и **MOVX**. Возможно попеременное использование

команд OUTL и MOVX. Однако при этом необходимо помнить, что выводимый по команде OUTL байт фиксируется в буферном регистре порта BUS, а команда MOVX уничтожает содержимое буферного регистра порта BUS. (Команда INS не уничтожает содержимое буферного регистра порта.) В МК-системах, имеющих внешнюю память программ, порт BUS используется для выдачи адреса внешней памяти и для приема команды из внешней памяти программ. Следовательно, в таких системах использование команды OUTL лишено смысла.

2.2.4. Устройство управления микроконтроллера

Устройство управления МК совместно с логической схемой переходов в каждом цикле команды формирует последовательность сигналов, управляющих функциями всех блоков МК и системой их взаимосвязи. Рассмотрение МК и особенностей реализации тех или иных процедур удобно выполнить путем анализа работы отдельных блоков МК в различных режимах его работы.

Синхронизация МК. Опорную частоту синхронизации определяет или кварцевый резонатор, подключаемый к выводам X1 и X2, или LC-цепь. X1 является входом, а X2 – выходом генератора, способного работать в диапазоне частот от 1 до 6 МГц. На вход X1 может подаваться сигнал от источника внешней синхронизации. Варианты схем синхронизации МК показаны на рис. 2.6. В состав генератора МК входят два счетчика с модулями пересчета 3 и 5. Первый используется для формирования сигнала системной синхронизации СС (0,5 мкс). Этот же сигнал поступает на счетчик машинных циклов, на выходе которого через каждые пять сигналов синхронизации формируется сигнал САВП (2,5 мкс), идентифицирующий машинный цикл и используемый в расширенных МК-системах для стробирования адреса внешней памяти.

Системный сброс. В обслуживаемых МК-системах для инициализации используется кнопка СБРОС, которая заземляет соответствующий вход МК. В необслуживаемых МК-системах к входу СБР подсоединяется конденсатор емкостью 1 мкФ, что обеспечивает подачу сигнала, близ-

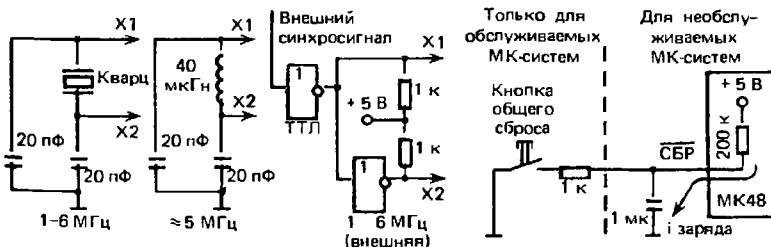


Рис. 2.6. Варианты схем синхронизации МК48 Рис. 2.7. Схема начальной установки МК48

кого к потенциалу земли, длительностью не менее 50 мс после того, как напряжение электропитания установится (рис. 2.7). Сигнал СБР производит следующие действия: сбрасывает счетчик команд и указатель стека; устанавливаетпорт BUS в высокомпреданное состояние, а порты P1 и P2 – на режим ввода; выбирает банк регистров 0 и банк памяти 0; запрещает прерывания; останавливает таймер и выдачу синхросигнала на вывод T0; сбрасывает флаг переполнения таймера TF и флаги пользователя F0 и F1.

Логика условных переходов. Логическая схема условных переходов МК позволяет программе проверять не только признаки, но и условия, внешние по отношению к МК. По командам условного перехода в случае удовлетворения проверяемого условия в счетчик команд (биты 0–7) из второго байта команды загружается адрес перехода. Логика переходов МК оперирует с набором условий, перечисляемых в табл. 2.1.

Режим прерывания. Линия запроса прерывания от внешнего источника ЗПР проверяется каждый машинный цикл во время действия сигнала САВП, но передача управления ячейке 3, где расположена команда JMP, выполняется только по завершению цикла команды. При обработке прерывания, как и при вызове подпрограммы, содержимое счетчика команд и старшей четверти ССП сохраняется в стеке. Ко входу ЗПР микроконтроллера через монтажное ИЛИ от схем с открытым коллектором могут быть подключены несколько источников прерывания. После распознавания прерывания все последующие запросы прерывания игнорируются до тех пор, пока по команде возврата RETR вновь будет разрешена работа логики прерываний. Режим прерываний может быть запрещен или разрешен программой по командам DIS I и EN I. Сигнал ЗПР должен быть снят внешним устройством перед окончанием подпрограммы обслуживания, т.е. до исполнения команды RETR. В том случае, если внешнее устройство не сбрасывает свой флаг запроса прерываний при обращении МК к его буферному регистру, одна из выходных линий МК используется подпрограммой обслуживания прерывания для сброса этого флага во внешнем устройстве. Так как вход ЗПР может быть проверен по команде условного перехода JNL, то при запрещенном режиме прерывания вход ЗПР может быть использован в качестве дополнительного тестирующего входа подобно входам T0 и T1.

Таблица 2.1. Условия переходов по программе

Устройство	Условие перехода	
	инверсное	прямое
Аккумулятор	Не все нули	Все нули
Выбранный бит аккумулятора	–	1
Флаг переноса С	0	1
Флаги пользователя F0 и F1	–	1
Флаг переполнения таймера TF	–	1
Тестовые входы (T0, T1)	0	1
Вход запроса прерывания ЗПР	0	–

При необходимости в МК можно создать двухуровневую систему прерываний. Для этого надо разрешить прерывания от таймера, загрузить в него число FFH и перевести в режим подсчета внешних событий, фиксируемых на входе T1. Переход сигнала на входе T1 из состояния 1 в состояние 0 приведет к прерыванию по вектору в ячейке 7. В случае одновременного запроса прерываний от внешнего источника и запроса от флага переполнения таймера приоритет остается за источником, воздействующим на вход ЗПР.

При входе в подпрограммы обслуживания прерываний старший бит счетчика команд CK11 принудительно устанавливается в нуль. Следовательно, вся процедура обработки прерывания должна быть размещена в банке памяти 0.

Таймер/счетчик. Внутренний 8-битный двоичный суммирующий счетчик может быть использован для формирования временных задержек и для подсчета внешних событий. Содержимое таймера/счетчика (T/C) можно прочитать (MOV A, T) или изменить (MOV T, A). Две команды STRT T и STRT CNT настраивают и запускают таймер/счетчик в режиме таймера или в режиме счетчика событий соответственно. Остановить работу (но не сбросить содержимое) таймера/счетчика можно или командой STOP TCNT, или сигналом системного сброса СБР. Из максимального состояния FFH таймер/счетчик переходит в начальное состояние 00H. При этом устанавливается в 1 флаг переполнения, который может вызвать прерывание, если оно разрешено командой EN TCNTI. Прерывание от таймера/счетчика может быть запрещено командой DIS TCNTI, но при этом флаг переполнения может быть опрошен по команде условного перехода JTF. Выполнение команды JTF, как и переход к подпрограмме обработки прерывания по вектору с адресом 7, сбрасывает флаг переполнения таймера/счетчика.

В режиме таймера на вход таймера/счетчика через делитель частоты на 32 поступают основные синхросигналы машинного цикла САВП (400 КГц), и счетчик увеличивает свое состояние на 1 через каждые 80 мкс (12,5 КГц). Путем программной установки таймера/счетчика в исходное состояние и анализа флага переполнения могут быть реализованы различные временные задержки, лежащие в диапазоне от 80 мкс до 20,48 мс. Временные задержки, превышающие по длительности 20 мс (256 состояний счетчика), могут быть получены накоплением переполнений в рабочем регистре под управлением программы.

В режиме счетчика событий внутренний счетчик увеличивает свое состояние на 1 каждый раз, когда сигнал на входе T1 переходит из состояния 1 в состояние 0. Минимально возможное время между двумя входными сигналами равно 7,5 мкс (3 машинных цикла при использовании резонатора 6 МГц). Минимальная длительность единичного сигнала на входе T1 составляет 0,5 мкс.

2.3.

Система команд MK48

2.3.1. Общие сведения о системе команд

Система команд MK48 включает в себя 96 основных команд и ориентирована на реализацию процедур управления. Все команды имеют формат один или два байта (70% команд однобайтные). Время выполнения команд составляет 2,5 или 5,0 мссы [один или два машинных цикла (МЦ) соответственно] при тактовой частоте 6,0 МГц. Большинство команд выполняется за один машинный цикл. За два машинных цикла выполняются команды с непосредственным операндом, ввода/вывода и передачи управления. Микроконтроллер оперирует с командами четырех типов (рис. 2.8).

В MK48 используются четыре способа адресации: прямая, непосредственная, косвенная и неявная.

Все множество команд можно разбить на пять групп по функциональному признаку: команды пересылки данных, арифметических операций, логических операций, передачи управления и управления режимами работы МК.

К достоинствам системы команд MK48 можно отнести: эффективный ввод/вывод, включая маскирование и возможность управления отдельными битами портов; возможность ветвления по значению отдельных бит; возможность обработки как двоичных, так и десятичных двоично-кодированных чисел.

При выполнении команд могут использоваться значения отдельных флагов, входящих в ССП, и флагов пользователя. Все команды, в результате выполнения которых модифицируются флаги, перечислены в табл. 2.2.

Ниже приводится краткое описание команд MK48, сгруппированных по функциональному признаку. При описании команд используются мнемокоды языка ассемблера MK48, а операции, выполняемые по командам, описываются на языке микрооператоров с использованием символьических имен и сокращений, которые перечисляются в списке сокращений.

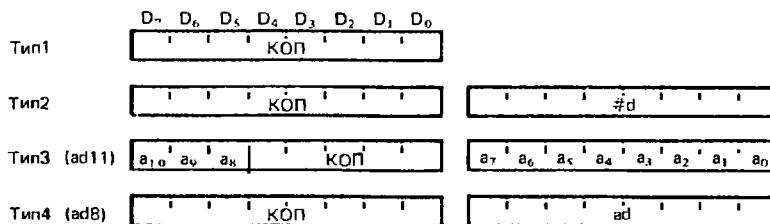


Рис. 2.8. Типы команд MK48

Таблица 2.2. Команды, модифицирующие флаги

Команды	Флаги	Команды	Флаги
ADD, ADDC	C, AC	JTF	TF = 0
CLR C	C = 0	MOV PSW, A	C, AC, F0, BS
CPL C	C	RETR	C, AC, F0, BS
CLR F0	F0 = 0	RLC A	C
CLR F1	F1 = 0	RRC A	C
CPL F0	F0		
CPL F1	F1	SEL MBO, SEL MBI	DBF
DA A	C, AC	SEL RBO, SEL RBI	BS

2.3.2. Группа команд пересылки данных

Данная группа состоит из 24 команд (табл. 2.3). Все команды (кроме MOV PSW, A) не оказывают воздействия на флаги. Команды пересылки данных внутри МК выполняются за один машинный цикл, обмен с внешней памятью и портами требует двух машинных циклов.

Структура информационных связей. На рис. 2.9 представлен график возможных пересылок, который иллюстрирует структуру информационных связей MK48. Можно выделить девять типов операндов, между которыми выполняется информационный обмен. Операнды, участвующие в операциях пересылки, различаются по месту расположения и способу адресации. К операндам относятся: аккумулятор, регистры общего назначения, ССП, таймер, порты, непосредственный операнд, внешняя и резидентная память данных и память программ (ПП). Аккумулятор является как бы "почтовым ящиком", через который остальные устройства (операнды) могут обмениваться данными. К памяти программ существует только односторонний доступ для чтения.

Форматы данных. Большинство команд выполняет пересылку 8-битных (1-байтных) операндов. Существуют также несколько команд, оперирующих с 4-битными операндами (тетрадами). Команды пересылки тетрад используются при обращении к 4-битным portам внешней схемы расширения ввода/вывода (P4–P7).

Режимы передачи данных. В MK48 возможна передача данных в двух режимах: пересылки (загрузки) и обмена. Пересылка предполагает передачу данных в направлении от источника к приемнику. При этом источник не изменяет своего содержимого. Обмен предполагает одновременную передачу данных в двух направлениях: в результате операции обмена изменяются значения обоих операндов, участвующих в операции.

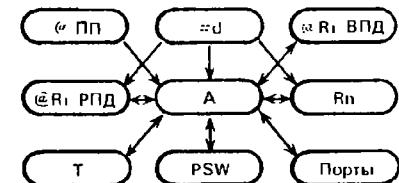


Рис. 2.9. Граф пересылок данных

Таблица 2.3. Группа команд пересылки данных (Т – тип команды, Б – формат в байтах, Ц – число машинных циклов)

Название команды	Методика	КОП	Т	Б	Ц	Операция
Пересылка регистра в аккумулятор	MOV A, Rn	11111rrr	1	1	1	(A) \leftarrow (Rn)
Пересылка байта из РПД в аккумулятор	MOV A, @Ri	00100001i	1	1	1	(A) \leftarrow ((Ri))
Пересылка непосредственного операнда в аккумулятор	MOV A, #d	00100011	2	2	2	(A) \leftarrow #d
Пересылка Rn в А	MOV Rn, A	10101rrr	1	1	1	(Rn) \leftarrow (A)
Пересылка аккумулятора в регистр	MOV Rn, #d	10111rrr	2	2	2	(Rn) \leftarrow #d
Пересылка непосредственного операнда в РПД	MOV @Ri, A	10100001i	1	1	1	((Ri)) \leftarrow (A)
Пересылка непосредственного операнда в РПД	MOV (@Ri), #d	10110001i	2	2	2	((Ri)) \leftarrow #d
Пересылка ССП в аккумулятор	MOV A, PSW	11000111	1	1	1	(A) \leftarrow (PSW)
Пересылка аккумулятора в ССП	MOV PSW, A	11010111	1	1	1	(PSW) \leftarrow (A)
Пересылка содержимого таймера/счетчика в аккумулятор	MOV A, T	01000010	1	1	1	(A) \leftarrow (T)
Пересылка аккумулятора в таймер/счетчик	MOV T, A	01100010	1	1	1	(T) \leftarrow (A)
Пересылка байта из ВПД в аккумулятор	MOVX A, @Ri	10000001	1	1	2	(A) \leftarrow ((Ri))
Пересылка аккумулятора в ВПД	MOVX (@Ri), A	10010001	1	1	2	((Ri)) \leftarrow (A)
Пересылка байта из текущей страницы программной памяти в аккумулятор	MOVPA, @A	10100011	1	1	2	((PC ₀₋₇) \leftarrow (A))
Пересылка байта из третьей страницы программной памяти в аккумулятор	MOVPA3 A, @A	11100011	1	1	2	(PC ₀₋₇) \leftarrow (A) (PC ₈₋₁₁) \leftarrow 0011 (A) \leftarrow ((PC))
Обмен регистра с аккумулятором	XCH A, Rn	00101rrr	1	1	1	(A) \leftarrow (Rn)
Обмен аккумулятора с РПД	XCHD A, @Ri	00110001i	1	1	1	(A ₀₋₃) \leftarrow ((Ri) ₀₋₃)
Обмен машинных тетрад аккумулятора и байта РПД	XCHD A, @Ri	00110001i	1	1	2	(A ₀₋₃) \leftarrow (P ₀₋₃)
Пересылка данных из порта Pp (p = 1..2) в аккумулятор	IN A, Pp	000010pp	1	1	2	(A) \leftarrow (BUS)
Стробируемый ввод данных из порта BUS	IMS A, BUS	000011000	1	1	2	(A) \leftarrow (BUS)
Стробируемый аккумулятор в порт Pp (p = 1..2)	OUTL Pp, A	001110pp	1	1	2	(BUS) \leftarrow (A)
Стробируемый вызов данных из аккумулятора в порт BUS	MOVD A, Pp	00000010	1	1	2	(A ₀₋₃) \leftarrow (Pp)
Ввод тетрады из порта Pp (p = 4..7) схемы расширителя	MOVD Pp, A	000111pp	1	1	2	(A ₄₋₇) \leftarrow 0000 (Pp) \leftarrow (A ₀₋₃)
Выход тетрады в порт Pp (p = 4..7) схемы расширителя						

Способы адресации. Для обращения к данным используются четыре способа адресации:

- 1) прямая, когда адрес операнда содержится в теле самой команды, например

MOV A,RN : (A) \leftarrow (RN)

Номер регистра, пересылаемого в аккумулятор, указывается в трех младших битах кода операции (КОП);

- 2) непосредственная, когда сам 8-битный operand (константа) располагается непосредственно в теле команды (второй байт команды), например

MOV #05 : (A) \leftarrow 05

Содержимое второго байта команды (05) пересыпается в аккумулятор;

- 3) косвенная, при которой адрес опранда располагается в регистре R0 или R1, например

MOV A,@R0 : (A) \leftarrow ((R0))

Содержимое ячейки РПД по адресу, хранимому в регистре R0, пересыпается в аккумулятор;

- 4) неявная, при которой в коде операции содержится неявное (по умолчанию) указание на один из operandов. Чаще всего таким operandом является аккумулятор, как, например, в команде

MOV A,RN : (A) \leftarrow (RN)

Особенности передач через порты. Порты P1 и P2 представляют собой управляемые буферные регистры. При выдаче информации выводимый байт данных фиксируется в буферном регистре порта, а при вводе он не фиксируется и должен быть прочитан контроллером в течение периода присутствия байта на входах порта. Ориентация МК на применение в устройствах управления объектами привела к появлению в его системе команд таких команд, которые позволяют выполнять операции ввода/вывода информации с использованием маскирования, что предоставляет разработчику удобные средства обмена не только байтами, но отдельными битами и их произвольными комбинациями. Схемотехника портов P1 и P2 такова, что ввод данных в некоторую линию возможен только в том случае, если предварительно в данный бит порта программой МК была записана 1.

Порт BUS может выполнять все функции, перечисленные для портов P1 и P2, но в отличие от них он не может специфицировать отдельные линии на ввод или вывод. Все восемь линий порта BUS должны одновременно быть либо входными, либо выходными. По командам MOVX порт BUS используется в качестве двунаправленного синхронного канала для доступа к ВПД.

2.3.3. Группа команд арифметических операций

Данная группа состоит из 12 команд (табл. 2.4) и позволяет выполнять следующие операции над 8-битными целыми двоичными числами без знака: двоичное сложение (ADD), двоичное сложение с учетом переноса (ADDC), десятичная коррекция (DA), инкремент (INC) и декремент (DEC).

При сложении используется неявная адресация источника первого операнда и места назначения результата, в качестве которых выступает аккумулятор. Содержимое аккумулятора А можно сложить с регистром, константой и ячейкой РПД. В результате суммирования возможно появление переноса, который фиксируется в специальном триггере переноса (флаг С). Команда сложения с учетом переноса позволяет выполнять суммирование многобайтных чисел.

При необходимости выполнять двоичное вычитание требуется перевести вычитаемое в дополнительный код и произвести сложение с уменьшаемым.

Все более сложные операции (умножение, деление) выполняются по подпрограммам.

2.3.4. Группа команд логических операций

Данная группа состоит из 28 команд (табл. 2.5) и позволяет выполнять следующие операции над байтами: дизъюнкцию, конъюнкцию, исключающее ИЛИ, инверсию, сброс и сдвиг. Две команды (сброс и инверсия) позволяют выполнять операции над битами.

Широко используется неявная адресация аккумулятора в качестве источника операнда и места фиксации результата. Вторым операндом в командах может быть регистр, константа или ячейка РПД. Существуют команды (ANL, ORL), оперирующие с портами, что позволяет эффективно управлять значениями отдельных бит при вводе/выводе информации.

2.3.5. Группа команд передачи управления

Данную группу образуют 19 команд передачи управления, из них две команды безусловного перехода, 14 команд условного перехода, команда вызова подпрограмм и две команды возврата из подпрограмм. В табл. 2.6 приводится описание команд передачи управления.

Команды ветвления с прямой адресацией. В большинстве команд прямо указывается адрес перехода. В теле команды при этом содержится 8 (ad) или 11 (ad11) бит адреса перехода. Команда JMP позволяет передать управление в любое место 2048-байтного банка памяти программ (ПП). Номер банка ПП определяется флагом DBF, значение которого копируется в старший бит счетчика команд (PC₁₁) при выполнении команды JMP или CALL. Для перехода из нулевого банка ПП в первый недостаточно только установить флаг DBF, необходимо также вы-

Таблица 2.4. Группа команд арифметических операций

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Сложение регистра с аккумулятором	ADD A, Rn	01101rrr	1	1	1	(A) ← (A) + (Rn)
Сложение байта из РПД с аккумулятором	ADD A, @Ri	0110000i	1	1	1	(A) ← (A) + ((Ri))
Сложение константы с аккумулятором	ADD A, #d	00000011	2	2	2	(A) ← (A) + #d
Сложение регистра с аккумулятором и переносом	ADDC A, Rn	01111rrr	1	1	1	(A) ← (A) + (Rn) + (C)
Сложение байта из РПД с аккумулятором и переносом	ADDC A, @Ri	0111000i	1	1	1	(A) ← (A) + ((Ri)) + (C)
Сложение константы с аккумулятором и переносом	ADDC A, #d	00010011	2	2	2	(A) ← (A) + #d + (C)
Десятичная коррекция аккумулятора	DA A	01010111	1	1	1	если ((A ₀₋₃) > 9) V ((AC) = 1), то (A ₀₋₃) ← (A ₀₋₃) + 6, затем, если ((A ₄₋₇) > 9) V ((C) = 1), то (A ₄₋₇) ← (A ₄₋₇) + 6 (A) ← (A) + 1 (Rn) ← (Rn) + 1
Инкремент аккумулятора	INC A	00010111	1	1	1	
Инкремент регистра	INC Rn	00011rrr	1	1	1	
Инкремент байта в РПД	INC @Ri	0001000i	1	1	1	((Ri)) ← ((Ri)) + 1
Декремент аккумулятора	DEC A	00000111	1	1	1	(A) ← (A) - 1
Декремент регистра	DEC Rn	11001rrr	1	1	1	(Rn) ← (Rn) - 1

Таблица 2.5. Группа команд логических операций

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Логическое И регистра и аккумулятора	ANL A, Rn	01011rr	1	1	1	(A) \leftarrow (A) \wedge (Rn)
Логическое И бита из РПД и аккумулятора	ANL A, @Ri	0101000i	1	1	1	(A) \leftarrow (A) \wedge ((Ri))
Логическое ИЛИ константы и аккумулятора	ORL A, #d	01010011	2	2	2	(A) \leftarrow (A) \wedge d
Логическое ИЛИ регистра и аккумулятора	ORL A, Rn	01001rr	1	1	1	(A) \leftarrow (A) \vee (Rn)
Логическое ИЛИ байта из РПД и аккумулятора	ORI A, @Ri	0100000i	1	1	1	(A) \leftarrow (A) \vee ((Ri))
Логическое ИЛИ константы и аккумулятора	ORL A, #d	01000011	2	2	2	(A) \leftarrow (A) \vee *d
Исклучающее ИЛИ регистра и аккумулятора	XRL A, Rn	11011rr	1	1	1	(A) \leftarrow (A) $\vee\neq$ (Rn)
Исклучающее ИЛИ байта из РПД и аккумулятора	XRL A, @Ri	1101100i	1	1	1	(A) \leftarrow (A) $\vee\neq$ ((Ri))
Исклучающее ИЛИ константы и аккумулятора	XRL A, #d	11010011	2	2	2	(A) \leftarrow (A) $\vee\#$ d
Сброс аккумулятора	CLR A	00100111	1	1	1	(A) \leftarrow 0
Инверсия аккумулятора	CPL A	00110111	1	1	1	(A) \leftarrow \bar{A}
Обмен тетрад в аккумуляторе	SWAP A	01000111	1	1	1	(A ₀₋₃) \leftarrow (A ₄₋₇)
Циклический сдвиг влево аккумулятора	RL A	11100111	1	1	1	(A _{n+1}) \leftarrow (A _n); n = 0 ÷ 6
Сдвиг влево аккумулятора через перенос	RLC A	11110111	1	1	1	(A _{n+1}) \leftarrow (A _n); n = 0 ÷ 6 (A _n) \leftarrow (C); (C) \leftarrow (A ₇)
Циклический сдвиг вправо аккумулятора	RR A	01110111	1	1	1	(A _n) \leftarrow (A _{n+1}); n = 0 ÷ 6 (A ₇) \leftarrow (A ₀)
Сдвиг вправо аккумулятора через перенос	RRC A	01100111	1	1	1	(A _n) \leftarrow (A _{n+1}); n = 0 ÷ 6 (A ₇) \leftarrow (C); (C) \leftarrow (A ₀)
Логическое И константы и порта Pp (p = 1, 2)	ANL Pp, #d	100110pp	2	2	2	(Pp) \leftarrow (Pp) \wedge #d
Логическое И константы и порта BUS (p = 4 ÷ 7)	ANL BUS, #d	10011000 100111pp	2	2	2	(BUS) \leftarrow (BUS) \wedge #d (Pp) \leftarrow (Pp) \wedge (A ₀₋₃)
Логическое ИЛИ константы и порта Pp (p = 1, 2)	ORL Pp, #d	100010pp	2	2	2	(Pp) \leftarrow (Pp) V #d
Логическое ИЛИ константы и порта BUS Pp (p = 4 ÷ 7)	ORL BUS, #d	100010000 100011pp	2	2	2	(BUS) \leftarrow (BUS) V #d (Pp) \leftarrow (Pp) V (A ₀₋₃)
Сброс переноса	CLR C	10010111	1	1	1	(C) \leftarrow 0
Сброс флага F0	CLR F0	10000101	1	1	1	(F0) \leftarrow 0
Сброс флага F1	CLR F1	10100101	1	1	1	(F1) \leftarrow 0
Инверсия переноса	CPL C	10100111	1	1	1	(C) \leftarrow \bar{C}
Инверсия флага F0	CPL F0	10010101	1	1	1	(F0) \leftarrow (F0)
Инверсия флага F1	CPL F1	10110101	1	1	1	(F1) \leftarrow (F1)

Таблица 2.6. Группа команд передач управления

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Безусловный переход	JMP ad11	a10393800100	3	2	2	(PC ₀₋₇) ← ad11, (PC ₁₁) ← DBF
Косвенный переход в текущей странице ПП	JMPP &A	10110011	1	1	2	(PC ₀₋₇) ← ((A))
Декrement регистра и переход, если не нуль	DJNZ R,n,ad	11101011T	4	2	2	(Rn) ← (Rn) - 1; если (Rn) ≠ 0, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если перенос	JC ad	11110110	-	4	2	Если (C) = 1, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если нет переноса	JNC ad	11100110	4	2	2	Если (C) = 0, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если аккумулятор содержит нуль	JZ ad	11000110	4	2	2	Если (A) = 0, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если аккумулятор содержит не нуль	JNZ ad	10010110	4	2	2	Если (A) ≠ 0, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если на входе T0 высокий уровень	JTO ad	00110110	4	2	2	Если T0 = 1, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если на входе T0 низкий уровень	JNT0 ad	00100110	4	2	2	Если T0 = 0, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если на входе T1 высокий уровень	JT1 ad	01010110	4	2	2	Если T1 = 1, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если на входе T1 низкий уровень	JNT1 ad	01000110	4	2	2	Если T1 = 0, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если флаг F0 установлен	J1:0 ad	10110110	4	2	2	Если (F0) = 1, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если флаг F1 установлен	JF1 ad	01110110	4	2	2	Если (F1) = 1, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если флаг прерывания таймера установлен	JTF ad	00010110	4	2	2	Если TF = 1, то TF ← 0, (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если на входе ЭПР низкий уровень	JNI ad	10000110	4	2	2	Если ЭПР = 0, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Переход, если бит аккумулятора равен единице (b = 0 ÷ 7)	JBb ad	bbb10010	4	2	2	Если (Bb) = 1, то (PC ₀₋₇) ← ad, иначе (PC) ← (PC) + 2
Вызов подпрограммы	CALL ad11	a10393810100	3	2	2	((SP)) ← (PC), (PSW ₄₋₇), (SP) ← (SP) + 1, (PC ₁₁) ← DBF, (PC ₀₋₀) ← ad 11
Возврат из подпрограммы	RFT	10000011	1	1	2	(SP) ← (SP) - 1, (PC) ← (SP) - 1,
Возврат из подпрограммы и восстановление ССП	RETR	10010011	1	1	2	((SP)) ← (SP) - 1, (PC) ← (SP), (PSW ₄₋₇) ← (SP)

полнить команду перехода JMP, которая и изменит значение старшего бита счетчика команд.

Все остальные команды (кроме команд возврата) содержат только восемь младших битов адреса перехода. При этом оказывается возможным осуществить переход только в пределах одной страницы ПП (256 байт).

Если команда короткого перехода расположена на границе двух страниц (т.е. первый байт команды на одной странице, а второй — на следующей), то переход будет выполнен в пределах той страницы, где располагается второй байт команды. Для условного перехода с одной страницы на другую можно воспользоваться tandemом из команды условного перехода и длинного безусловного перехода (JMP).

Переход по косвенному адресу. Команда JMPP осуществляет переход по адресу, содержащемуся в ячейке ПП, на которую указывает содержимое аккумулятора. Таким образом, аккумулятор содержит адрес адреса перехода. Ячейка с адресом перехода должна находиться на той же странице ПП, что и команда перехода JMPP. Команда косвенного перехода обеспечивает простой доступ к таблице, содержащей векторы переходов по программе в зависимости от содержимого аккумулятора, что позволяет легко реализовать механизм множественных ветвлений.

Условные переходы. По командам условных переходов могут проводиться не только внутренние флаги, но и некоторые сигналы на внешних входах МК. Это позволяет эффективно выполнять ветвления в программе без использования процедуры предварительного ввода и последующего сравнения. Все команды условных переходов используют прямую короткую адресацию, что накладывает определенные ограничения на размещение программ в памяти. Анализируемые признаки за исключением C и F0 не фиксируются в специальных триггерах флагов, а представляются мгновенными значениями сигналов в АЛУ или на соответствующих входах МК.

Программные циклы. Для организации циклов удобно использовать команду DJNZ. Счетчик циклов организуется в одном из регистров текущего банка, для чего в этот регистр загружается число повторений цикла. При выполнении команды DJNZ производится декремент и последующая проверка на нуль содержимого регистра — счетчика циклов. Если его содержимое оказывается не нулевым, то происходит переход к началу цикла, иначе — выход из цикла. Структура программы при этом будет следующей:

```
MOV  RN,BH  ИНИЦИАЛИЗАЦИЯ СЧЕТЧИКА ЦИКЛОВ
LOOP: ...
...
CYCLE: DJNZ  RN,LOOP ?ДЕКРЕМЕНТ РН, И ПЕРЕХОД, ЕСЛИ НЕ НУЛЬ
```

Следует отметить, что команды от метки LOOP до метки CYCLE включительно должны находиться в пределах одной страницы памяти программы.

Работа с подпрограммами. Для вызова подпрограмм существует команда CALL, позволяющая обратиться в любое место текущего

банка ПП. При вызове подпрограммы в стеке запоминается адрес возврата и часть ССП. Глубина вложений подпрограмм ограничена емкостью стека (16 байт) и не должна превышать восьми.

Для возврата из подпрограммы необходимо выполнить команду RET, которая восстановит в счетчике команд адрес возврата. Для выхода из подпрограммы обработки прерывания служит команда RETR, которая кроме адреса возврата восстанавливает ССП и разрешает прерывания от данного источника.

При программировании МК необходимо внимательно отслеживать ситуацию вызова подпрограммы, находящейся в альтернативном (по отношению к текущему) банке ПП. В этом случае перед вызовом подпрограммы необходимо выбрать соответствующий банк памяти, а перед возвратом — восстановить старое значение DBF. Если в подпрограмме нет команды восстановления DBF, то возврат все же будет выполнен правильно (так как в стеке сохранен полный действительный адрес возврата), однако первая же команда длинного перехода передаст управление в альтернативный банк ПП. Если к подпрограмме производятся обращения из разных банков памяти, то оказывается затруднительно восстановить значение DBF перед возвратом. В этом случае можно рекомендовать использовать команду восстановления DBF в основной программе вслед за командой вызова. Обращаем внимание читателя на указанные особенности как на возможный источник ошибок программирования. Пусть, например, требуется вызвать подпрограмму с именем SUBROUT, находящуюся в первом банке ПП, из программы, расположенной в нулевом банке ПП. Корректный вызов подпрограммы реализуется последовательностью команд:

SEL	M81	УСТАНОВКА ФЛАГА DBF
CALL	SUBROUT	ВЫЗОВ ПОДПРОГРАММЫ (ПОСЛЕ ВОЗВРАТА ; ИЗ ПОДПРОГРАММЫ ФЛАГ DBF ОСТАНЕТСЯ ; РАВНЫМ 1)
SEL	M80	ВОССТАНОВЛЕНИЕ ВФ В ИЗБЕЖАНИЕ ОШИБОЧНОГО ; ПЕРЕХОДА В БАНК 1

2.3.6. Группа команд управления режимом работы МК

В эту группу входят команды управления таймером/счетчиком, прерываниями и флагами переключения банков регистров и банков ПП. В табл. 2.7 перечислены команды этой группы.

Операции с таймером. Кроме рассмотренных ранее команд обмена между таймером и аккумулятором (MOV A,T и MOV T,A), по которым содержимое таймера может быть прочитано во время остановки счета и во время счета ("на лету") или изменено (перезагружено), в МК выполняются специальные команды управления режимом работы таймера. Таймер может быть (в зависимости от команды) использован как счетчик тактов или как счетчик событий от внутреннего или внешнего источника сигналов соответственно. Система команд МК располагает средствами разрешения или запрета прерывания от таймера.

Таблица 2.7. Группа команд управления режимами работы МК48

Название команды	Мнемокод	КОП	Ф	Б	Ц	Операция
Запуск таймера	STR T	01010101	1	1	1	
Запуск счетчика	STR CNT	01000101	1	1	1	
Останов таймера/счетника	STOP TCNT	01100101	1	1	1	
Разрешение прерывания от таймера/счетника	EN TCNTI	00100101	1	1	1	{ Описание команды приведено в тексте}
Запрещение прерывания от таймера/счетника	DIS TCNTI	00110101	1	1	1	
Разрешение внешнего прерывания	EN I	00000101	1	1	1	
Запрещение внешнего прерывания	DIS I	00010101	1	1	1	
Выбор нулевого банка регистров	SEL RBO	11000101	1	1	1	(BS) \leftarrow 0
Выбор первого банка регистров	SEL RBI	11010101	1	1	1	(BS) \leftarrow 1
Выбор нулевого банка ПП	SEL MBO	11100101	1	1	1	(DBF) \leftarrow 0
Выбор первого банка ПП	SEL MBI	11110101	1	1	1	(DBF) \leftarrow 1
Разрешение выдачи синхросигнала на выход Т0	ENTO CLC	01110101	1	1	1	T0 – синхросигнал (2 МГц)
Холостая команда	NOP	00000000	1	1	1	(PC) \leftarrow (PC) + 1

Специальной командой ENTO на вывод Т0 разрешается передача импульсов с частотой опорного синхросигнала, поделенной на три. Выдача этого сигнала может быть отключена только сигналом общего сброса. Синхросигнал на выходе Т0 используется для общей синхронизации внешних устройств, согласованных с МК по частоте работы.

Переключение банков регистров и ПП. Переключение банка памяти программ (т.е. изменение старшего бита счетчика команд) происходит в момент выполнения команды длинного перехода или вызова подпрограммы.

Наличие команд переключения банков регистров позволяет при вызове подпрограмм и обработке прерываний эффективно использовать второй банк регистров в качестве рабочего, сохраняя параметры вычислительного процесса не в стеке, а в исходном банке регистров. При программировании процедур обработки прерывания можно переключать или не переключать банки регистров. В том случае, когда банки регистров переключаются, возврат к исходному банку регистров будет выполнен автоматически, если подпрограмма обработки прерывания оканчивается командой возврата с восстановлением ССП (RETR).

2.4.

Особенности работы МК48 на этапе отладки прикладных программ

2.4.1. Микроконтроллер в пошаговом режиме работы и в режиме внешнего доступа

Пошаговый режим. Схема запуска и временная диаграмма работы МК в пошаговом режиме показаны на рис. 2.10. Этот режим используется на этапе отладки МК-системы и предоставляет разработчику возможность выполнить прикладную программу покомандно.

Сигнал ШАГ = 0 останавливает работу МК после окончания цикла текущей команды, после чего МК выдает подтверждающий сигнал САВП = 1. Для вывода МК из состояния останова необходимо подать сигнал ШАГ = 1, на который МК откликается генерацией подтверждающего сигнала САВП = 0. Для того чтобы МК остановился после следующей команды, на вход ШАГ вновь должен быть подан сигнал 0, как только сигнал САВП станет равным нулю. Переход от команды к команде осуществляется по нажатию кнопки ШАГ, устанавливающей буферный D-триггер по входу синхронизации. Сигнал ШАГ на выходе D-триггера не приобретет значение 1 до тех пор, пока сигнал САВП не будет равен 1. Так как вход R асинхронного сброса D-триггера эквивалентен трем ТТЛ-нагрузкам, то между выходом САВП и входом сброса D-триггера необходимо установить усилитель.

Режим внешнего доступа. Подача уровня +5 В на вход ОРПП позволяет отключить резидентную память программ МК и осуществлять выборку команд только из внешней памяти. Этот режим работы удобен на

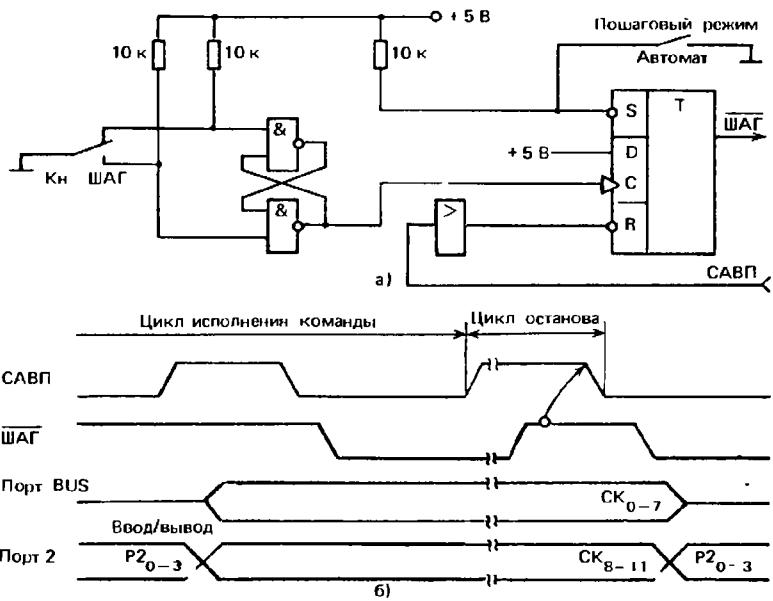


Рис. 2.10. Схема (а) и временная диаграмма (б) пошагового режима работы МК48

этапе отладки МК-системы, так как позволяет пользователю подключать внешнюю память, содержащую или диагностическую программу, или варианты прикладной программы. При помощи этого же сигнала ОРПП с уровнем +25 В возможно выполнить чтение содержимого РПП извне, что необходимо на этапе программирования МК для проверки вводимой информации.

2.4.2. Загрузка прикладных программ в резидентную память микроконтроллера

Процесс загрузки прикладных программ в РПП микроконтроллера, не очень точно называемый процессом программирования МК, требует использования программатора, генерирующего специальные сигналы и предоставляющего средства ввода в МК адресов/данных и средства визуального или иного контроля введенной информации. Процесс загрузки программ в РПП состоит из ряда последовательных действий:

- 1) настройки на режим загрузки РПП ($VDD = +5$ В, $X1$ синхросигнал с частотой 1–6 МГц, СБР = 0; ТО = +5 В, ОРПП = +5 В);
- 2) установки БИС МК48 в панельку (сокет) программатора;
- 3) инициализации режима загрузки ($T0 = 0$, ОРПП = +25 В; ПРОГ = 0);
- 4) подачи кода адреса на шину порта BUS и линии 0 и 1 порта P2 в соответствии со схемой на рис. 2.11, а;
- 5) подачи +5 В на вход СБР (зашелка адреса);

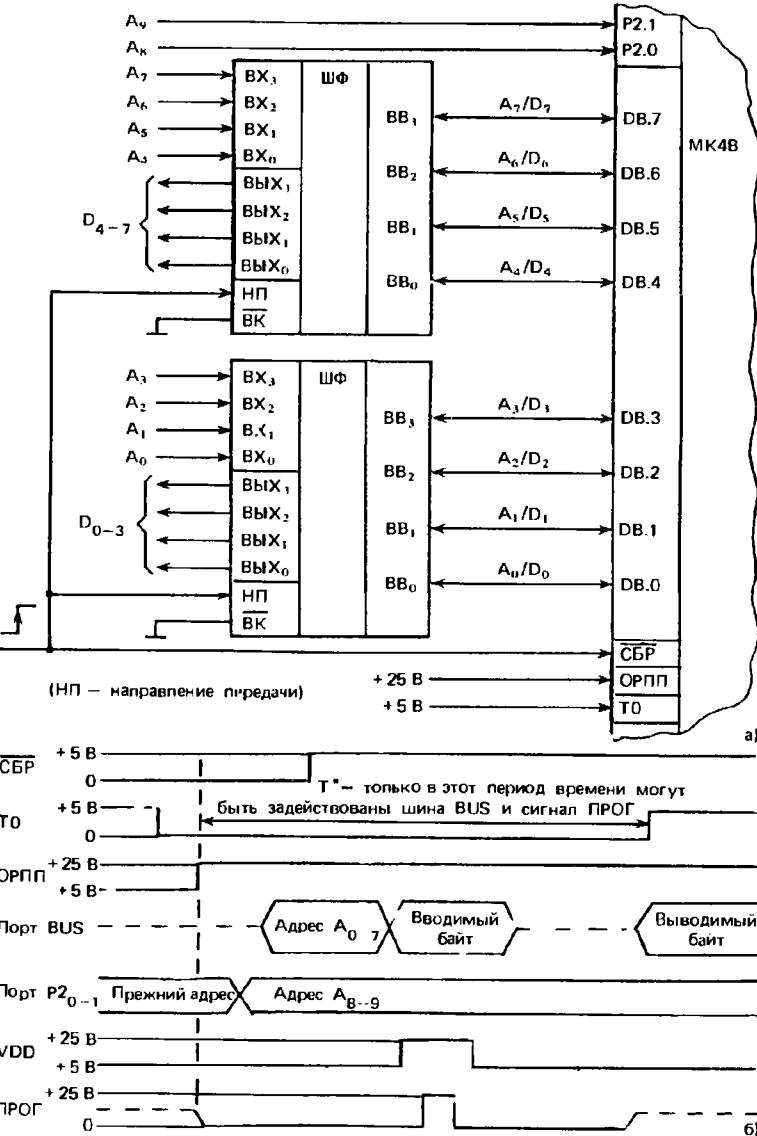


Рис. 2.11. Схема (а) и временная диаграмма (б) загрузки РПП с контролем вводимой информации

- 6) подачи байта на шину порта BUS;
- 7) подачи электропитания загрузки ($VDD = +25$ В);
- 8) подачи импульса загрузки (ПРОГ = +25 В длительностью 50 мс);
- 9) снятия электропитания загрузки ($VDD = +5$ В);
- 10) задания режима проверки ($T_0 = +5$ В);
- 11) чтения и проверки байта на шине BUS;
- 12) перехода к загрузке следующего байта прикладной программы ($T_0 = 0$, СБР = 0; перехода к п. 4 данной последовательности действий);
- 13) выполнения п. 1 по окончании загрузки прикладной программы;
- 14) удаления БИС MK48 из панельки программатора.

При загрузке программ в РПП участвует большая группа контактов корпуса МК, сигналы на которых должны формироваться в строго определенной последовательности. Эта временная последовательность приводится на рис. 2.11, б. Верификация прикладной программы, содержащейся в резидентной памяти МК, может быть выполнена путем прочтения ее содержимого в режиме внешнего доступа.

2.5.

Варианты структурной организации систем на основе MK48

В тех случаях, когда функционально-логических возможностей однокристального МК оказывается недостаточно, можно относительно простыми средствами расширить МК-систему до следующих размеров: память программ – до 4 Кбайт; память данных – до 320 байт; линии ввода/вывода – практически неограниченно.

Кроме того, путем подключения специализированных БИС, входящих в микропроцессорный комплект K580, в МК-системе могут быть реализованы различные вспомогательные функции: связь с дисплеем и клавиатурой, многоуровневая программируемая система прерываний, сложная система таймирования, связь с телеграфно-телефонной линией передачи информации и т.д. С использованием средств буферирования и мультиплексирования адресов/данных можно под управлением программы создавать МК-системы любых требуемых состава и назначения.

2.5.1. МК-система с внешней памятью программ

Шина BUS по своим свойствам подобна двунаправленнойшине данных микропроцессора K580, и все расширения МК выполняются с использованием этой шины. При обращении к резидентной памяти программ МК не генерирует внешних управляющих сигналов (за исключением САВП, который всегда идентифицирует каждый машинный цикл). Начиная с адреса 1024, МК автоматически формирует управляющие сигналы, обеспечивающие выборку команд из внешней памяти. Рисунок 2.12 иллюстрирует процесс выборки команды из внешней памяти:

содержимое счетчика команд выводится через порт BUS и младшую тетраду порта P2;

по срезу сигнала САВП на внешнем регистре фиксируется адрес;
сигналом РВПП разрешается работа внешней памяти:
по спаду сигнала РВПП шина BUS переходит в режим ввода.
Ячейки памяти с адресами, лежащими за пределами банка памяти 0, могут быть доступны после выполнения команды переключения банков памяти SEL MBI, за которой должна следовать команда перехода (JMP или CALL).

На рис. 2.13 показана структура МК-системы с внешней памятью программ. Три дополнительные БИС памяти емкостью по 1 Кбайт подключаются к шине BUS своими информационными выходами. Младший байт адреса по сигналу САВП фиксируется на внешнем буферном регистре. Старшая тетрада адреса, выводимая через порт P2, не нуждается в буферизации, так как она стабильна на протяжении всего цикла выборки. Два самых старших бита адреса заводятся на внешний дешифратор (стробируемый сигналом РВПП) для селекции требуемой БИС памяти программ.

2.5.2. МК-система с внешней памятью данных

На рис. 2.14 показана схема МК-системы, в состав которой включены две дополнительные БИС ОЗУ суммарной емкостью 256 байт. Внешняя оперативная память доступна МК по командам пересылки MOVX A, @ R и MOVX @ R,A, которые по косвенному адресу (регистры R0 и R1) выполняют операции передачи байта между ВПД и аккумулятором. Сигналом САВП косвенный адрес, выводимый по шине BUS, фиксирует-

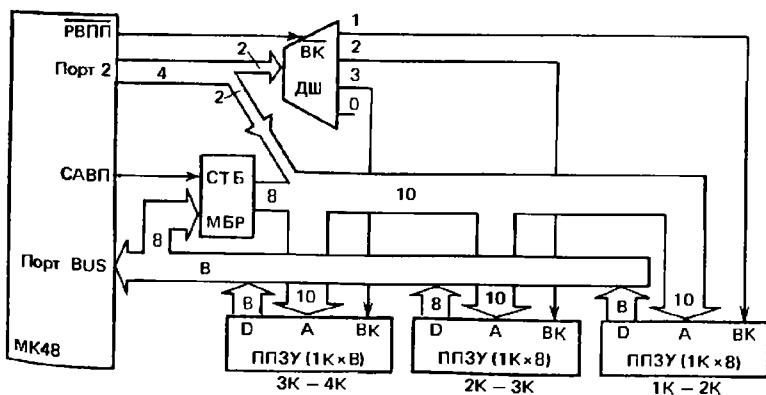


Рис. 2.13. Структура МК-системы с внешней памятью программ

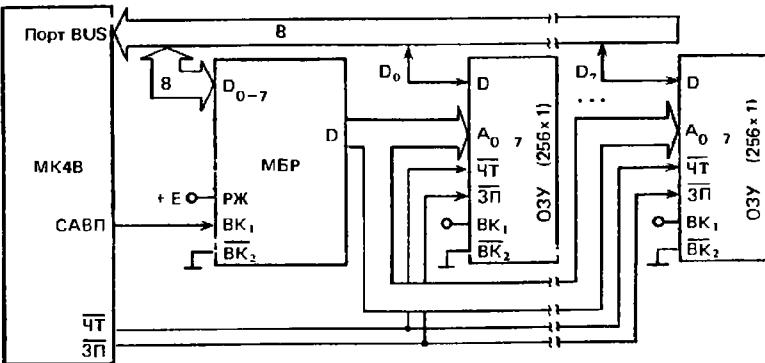


Рис. 2.14. Структура МК-системы с внешней памятью данных

ся в многорежимном буферном регистре МБР. Сигналы $\overline{ЗП}$ и $\overline{ЧТ}$ определяют режим работы БИС ОЗУ. Так как косвенный адрес имеет формат байта, то схема на рис. 2.14 обеспечивает адресацию 256 ячеек ОЗУ в дополнение к 64 ячейкам резидентной памяти данных МК48.

При необходимости дальнейшего наращивания объема внешнего ОЗУ можно программным способом реализовать механизм "перелистывания" страниц памяти через дополнительные линии порта P2 подобно тому, как это сделано на рис. 2.13.

2.5.3. МК-система с расширенным вводом/выводом

Для сопряжения МК с объектом, имеющим большое число входов/выходов, можно расширить резидентную систему ввода/вывода, подключив к МК необходимое количество внешних портов. Такое расширение может быть выполнено двумя способами: с использованием стандартного расширителя ввода/вывода (РВВ) KР580ВР43 или интерфейсных БИС (KР580ВВ55, KР580ВВ51).

Расширитель (см. приложение П2) подключается к МК48 так, как показано на рис. 2.15, а. Каждый из четырех портов РВВ может использоваться для ввода или вывода информации независимо от остальных и обеспечивает высокую нагрузочную способность. Для вывода байта данных в порты P4 и P5 можно воспользоваться следующей последовательностью команд:

```
MOV  P4,A ; ВЫХОД А(0...3) В ПОРТ 4
SWAP A ; ОБМЕН ТЕТРАДА АККУМУЛЯТОРА
MOV  P5,A ; ВЫХОД ВТОРОЙ ТЕТРАДЫ В ПОРТ 5
```

На рис. 2.15, б, в показаны два варианта расширения ввода/вывода с использованием параллельного периферийного адаптера (ППА) KР580ВВ55. В первом варианте (рис. 2.15, б) порты адаптера адресуют-

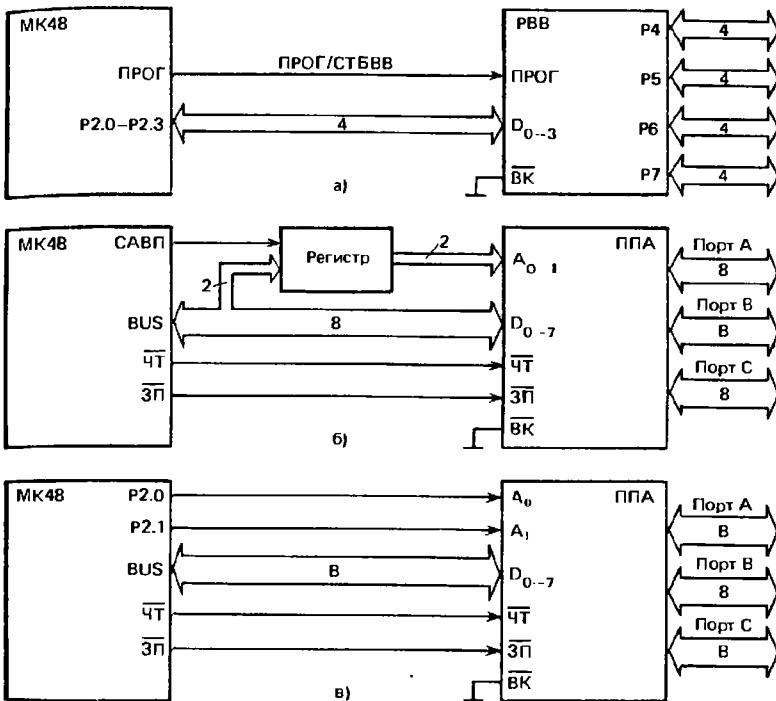


Рис. 2.15. Варианты расширения ввода/вывода информации в МК-системах

как ячейки ППД, доступ к которым осуществляется по командам MOVX. Например, для вывода байта впорт В необходимо выполнить две команды:

```
MOV  R1, #1 ; (R1) --- АДРЕС ПОРТА В
MOVX R1, A ; ВЫХОД В ПОРТ В ИЗ АККУМУЛЯТОРА
```

Для второго варианта подключения ППА (рис. 2.15, в) выбор порта осуществляется через установку/сброс двух разрядов порта P2. Так, для вывода байта данных в порт А можно воспользоваться следующими командами:

```
ANL  P2, #0FCH ; СБРОС А0 И А1 АДРЕСА
OUTL BUS, A ; ВЫХОД БАЙТА В ПОРТ А
```

В рассмотренных выше схемах расширения используется только по одной внешней БИС, и поэтому их вход $\overline{ВК}$ подсоединяется к земле. При необходимости к МК может быть подключено несколько РВВ и ППА.

Структурная организация и система команд микроконтроллера KM1816BE51

Микроконтроллер выполнен на основе высокочастотной n-MOS технологии и выпускается в корпусе БИС, имеющем 40 внешних выводов. Цоколевка корпуса MK51 и наименования выводов показаны на рис. 3.1. Для работы MK51 требуется один источник электропитания +5 В. Через четыре программируемых порта ввода/вывода MK51 взаимодействует со средой в стандарте ТТЛ-схем с тремя состояниями выхода.

Корпус MK51 имеет два вывода для подключения кварцевого резонатора, четыре вывода для сигналов, управляющих режимом работы MK, и восемь линий порта 3, которые могут быть запрограммированы пользователем на выполнение специализированных (альтернативных) функций обмена информацией со средой.

3.1.

Структурная схема МК51

Основу структурной схемы MK51 (рис. 3.2) образует внутренняя двунаправленная 8-битная шина, которая связывает между собой все основные узлы и устройства: резидентную память, АЛУ, блок регистров специальных функций, устройство управления и порты ввода/вывода.

Рассмотрим основные элементы структуры и особенности организации вычислительного процесса в МК51.

3.1.1. Арифметико-логическое устройство

8-битное АЛУ может выполнять арифметические операции сложения, вычитания, умножения и деления; логические операции И, ИЛИ, исключающее ИЛИ, а также операции циклического сдвига, сброса, инвертирования и т.п. В АЛУ имеются программно недоступные регистры T1 и T2, предназначенные для временного хранения операндов, схема десятичной коррекции и схема формирования признаков.

Простейшая операция сложения используется в АЛУ для инкрементирования содержимого регистров, продвижения регистра-указателя данных и автоматического вычисления следующего адреса РПП. Простейшая операция вычитания используется в АЛУ для декрементирования регистров и сравнения переменных.

Простейшие операции автоматически образуют "тандемы" для выполнения в АЛУ таких операций, как, например, инкрементирование

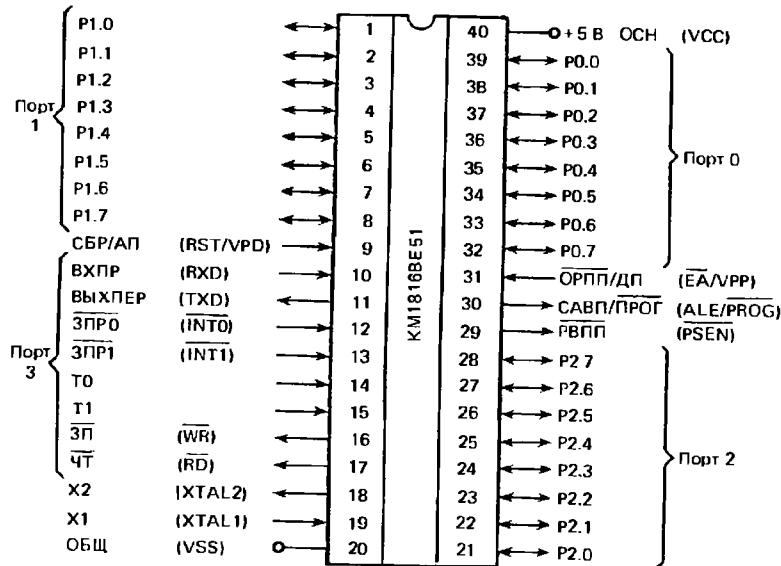


Рис. 3.1. Цоколевка корпуса МК51 и наименование выводов

16-битных регистровых пар. В АЛУ реализуется механизм каскадного выполнения простейших операций для реализации сложных команд. Так, например, при выполнении одной из команд условной передачи управления по результату сравнения в АЛУ трижды инкрементируется СК, дважды производится чтение из РПД, выполняется арифметическое сравнение двух переменных, формируется 16-битный адрес перехода и принимается решение о том, делать или не делать переход по программе. Все перечисленные операции выполняются в АЛУ за время 1 наносекунду.

Важной особенностью АЛУ является его способность оперировать не только байтами, но и битами. Отдельные программино-доступные биты могут быть установлены, сброшены, инвертированы, переданы, проверены и использованы в логических операциях. Эта способность АЛУ оперировать битами столь важна, что во многих описаниях MKS1 говорится о наличии в нем "булевского процессора". Для управления объектами часто применяются алгоритмы, содержащие операции над входными и выходными булевскими переменными (истина/ложь), реализация которых средствами обычных микропроцессоров сопряжена с определенными трудностями.

Таким образом, АЛУ может оперировать четырьмя типами информационных объектов: булевскими (1 бит), цифровыми (4 бита), байтами (8 бит) и адресными (16 бит). В АЛУ выполняется 51 различная операция пересылки или преобразования этих данных. Так как используемый в АЛУ регистор имеет 16 бит, то для выполнения операций с 8-битными объектами в АЛУ используется дополнительный 8-битный регистор.

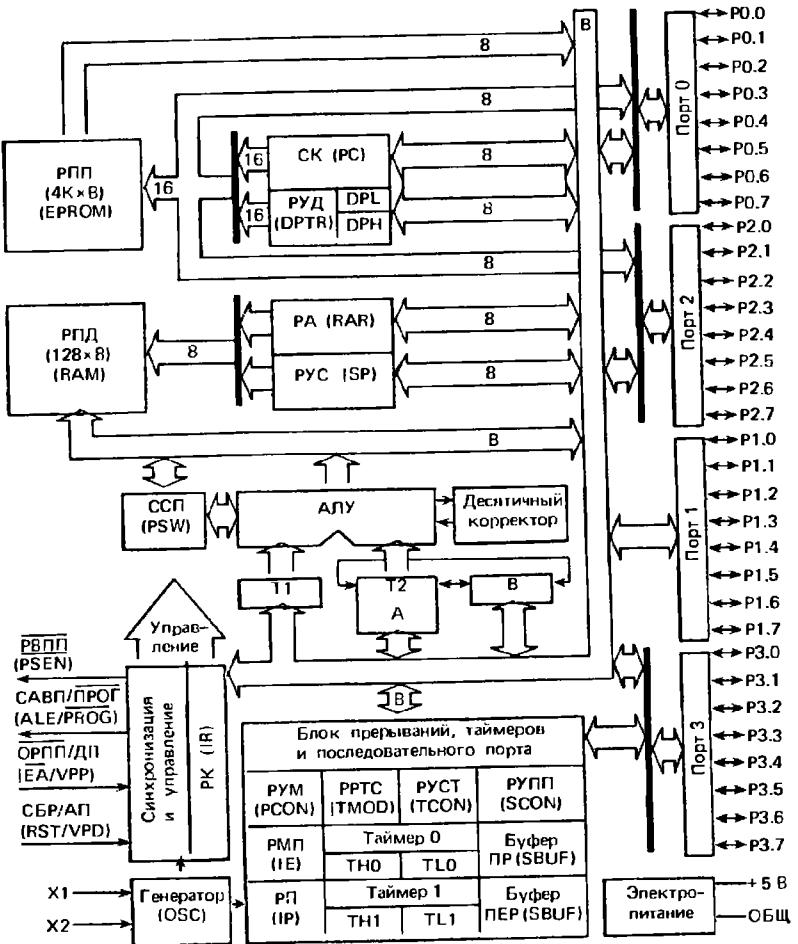


Рис. 3.2. Структурная схема MK51

зуется 11 режимов адресации (7 для данных и 4 для адресов), то путем комбинирования "операция/режим адресации" базовое число команд 111 расширяется до 255 из 256 возможных при однобайтном коде операций.

3.1.2. Резидентная память

Память программ и память данных, размещенные на кристалле MK51, физически и логически разделены, имеют различные механизмы адреса-

ции, работают под управлением различных сигналов и выполняют различные функции.

Память программ (ПЗУ или СППЗУ) имеет емкость 4 Кбайта и предназначена для хранения команд, констант, управляющих слов инициализации, таблиц перекодировки входных и выходных переменных и т.п. РПП имеет 16-битную шину адреса, через которую обеспечивается доступ из счетчика команд или из регистра-указателя данных. Последний выполняет функции базового регистра при косвенных переходах по программе или используется в командах, оперирующих с таблицами.

Память данных (ОЗУ) предназначена для хранения переменных в процессе выполнения прикладной программы, адресуется одним байтом и имеет емкость 128 байт. Кроме того, к адресному пространству РПД примыкают адреса регистров специальных функций (РСФ), которые перечислены в табл. 3.1.

Память программ, так же как и память данных, может быть расширена до 64 Кбайт путем подключения внешних БИС.

Аккумулятор и 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.

Таблица 3.1. Блок регистров специальных функций

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

П р и м е ч а н и е . Регистры, имена которых отмечены знаком (*), допускают адресацию отдельных бит.

Таблица 3.2. Формат слова состояния программы (ССП)

Символ	Позиция	Имя и назначение																				
C	PSW.7	Флаг переноса. Устанавливается и сбрасывается аппаратурными средствами или программой при выполнении арифметических и логических операций																				
AC	PSW.6	Флаг вспомогательного переноса. Устанавливается и сбрасывается только аппаратурными средствами при выполнении команд сложения и вычитания и сигнализирует о переносе или засме в бите 3																				
F0	PSW.5	Флаг 0. Может быть установлен, сброшен или проверен программой как флаг, специфицируемый пользователем																				
RS1	PSW.4	Выбор банка регистров. Устанавливается и сбрасывается программой для выбора рабочего банка регистров (см. примечание)																				
RS0	PSW.3																					
OV	PSW.2	Флаг переполнения. Устанавливается и сбрасывается аппаратурно при выполнении арифметических операций																				
-	PSW.1	Не используется																				
P	PSW.0	Флаг паритета. Устанавливается и сбрасывается аппаратурно в каждом цикле команды и фиксирует нечетное/четное число единичных бит в аккумуляторе, т.е. выполняет контроль по четности																				
Примечание. Выбор рабочего банка регистров																						
<table border="1"> <thead> <tr> <th>RS1</th> <th>RS0</th> <th>Банк</th> <th>Границы адресов</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>00H-07H</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>08H-0FH</td> </tr> <tr> <td>1</td> <td>0</td> <td>2</td> <td>10H-17H</td> </tr> <tr> <td>1</td> <td>1</td> <td>3</td> <td>18H-1FH</td> </tr> </tbody> </table>			RS1	RS0	Банк	Границы адресов	0	0	0	00H-07H	0	1	1	08H-0FH	1	0	2	10H-17H	1	1	3	18H-1FH
RS1	RS0	Банк	Границы адресов																			
0	0	0	00H-07H																			
0	1	1	08H-0FH																			
1	0	2	10H-17H																			
1	1	3	18H-1FH																			

При выполнении многих команд в АЛУ формируется ряд признаков операции (флагов), которые фиксируются в регистре ССП. В табл. 3.2 приводится перечень флагов ССП, даются их символические имена и описываются условия их формирования.

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 четырех регистровых банков.

Широкое распространение получило представление о том, что в микропроцессорах, архитектура которых опирается на аккумулятор, боль-

шинство команд работают с ним, используя адресацию "по умолчанию" (неявную). В MK51 дело обстоит иначе. Хотя процессор в MK51 имеет в своей основе аккумулятор, однако он может выполнять множество команд и без участия аккумулятора. Например, данные могут быть переданы из любой ячейки РПД в любой регистр, любой регистр может быть загружен непосредственным операндом и т.д. Многие логические операции могут быть выполнены без участия аккумулятора. Кроме того, переменные могут быть инкрементированы, декрементированы и проверены (test) без использования аккумулятора. Флаги и управляющие биты могут быть проверены и изменены аналогично.

Регистры-указатели. 8-битный указатель стека (РУС) может адресовать любую область РПД. Его содержимое инкрементируется прежде, чем данные будут запомнены в стеке в ходе выполнения команд PUSH и CALL. Содержимое РУС декрементируется после выполнения команд POP и RET. Подобный способ адресации элементов стека называют прединкрементным/постдекрементным. В процессе инициализации MK51 после сигнала СБР в РУС автоматически загружается код 07H. Это значит, что если прикладная программа не переопределяет стек, то первый элемент данных в стеке будет располагаться в ячейке РПД с адресом 08H.

Двухбайтный регистр-указатель данных (РУД) обычно используется для фиксации 16-битного адреса в операциях с обращением к внешней памяти. Командами MK51 регистр-указатель данных может быть использован или как 16-битный регистр, или как два независимых 8-битных регистра (DPH и DPL).

Таймер/счетчик. В составе средств MK51 имеются регистровые пары с символическими именами TH0, TL0 и TH1, TL1, на основе которых функционируют два независимых программируемых 16-битных таймера/счетчика событий.

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, IE, TMOD, TCON, SCON и PCON используются для фиксации и программного изменения управляющих бит и бит состояния схемы прерывания, таймера/счетчика, приемопередатчика последовательного порта и для управления мощностью электропитания MK51. Их организация будет описана ниже при рассмотрении особенностей работы MK51 в различных режимах.

3.1.3. Устройство управления и синхронизации

Кварцевый резонатор, подключаемый к внешним выводам X1 и X2 корпуса MK51, управляет работой внутреннего генератора, который в свою очередь формирует сигналы синхронизации.

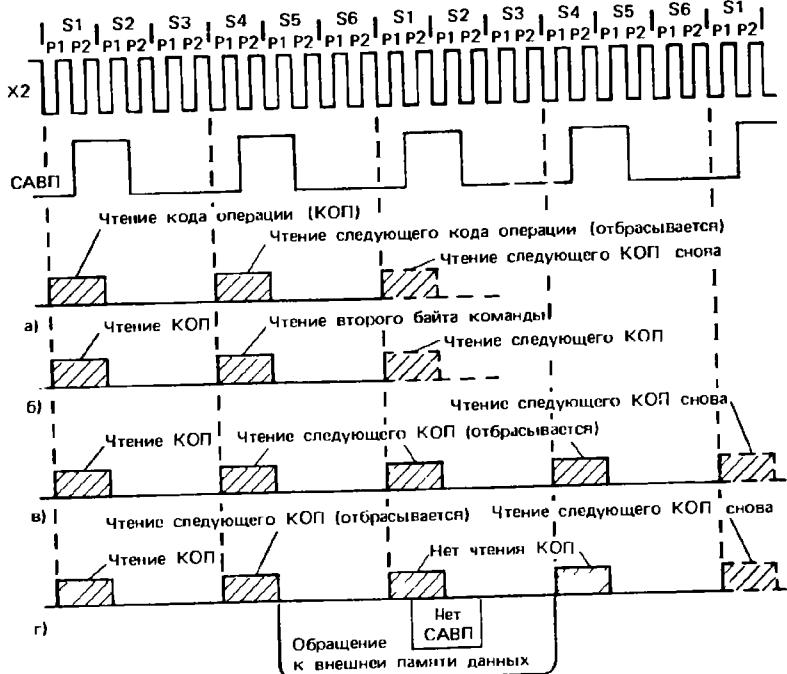


Рис. 3.3. Последовательности выборки и выполнения команд в MK51:
 а – команда 1 байт/1 цикл, например INC A; б – команда – 2 байта/1 цикл, например ADD A,#6; в – команда 1 байт/2 цикла, например INC DPTR;
 г – команда 1 байт/2 цикла, например MOVX

Устройство управления MK51 на основе сигналов синхронизации формирует машинный цикл фиксированной длительности, равной 12 периодам резонатора или шести состояниям первичного управляющего автомата (S1–S6). Каждое состояние управляющего автомата содержит две фазы (P1, P2) сигналов резонатора. В фазе P1, как правило, выполняется операция в АЛУ, а в фазе P2 осуществляется межрегистровая передача. Весь машинный цикл состоит из 12 фаз, начиная с фазы S1P1 и кончая фазой S6P2, как показано на рис. 3.3. Эта временная последовательность иллюстрирует работу устройства управления MK51 при выборке и исполнении команд различной степени сложности. Все защищенные сигналы являются внутренними и недоступны пользователю MK51 для контроля. Внешними, наблюдаемыми сигналами являются только сигналы резонатора и строба адреса внешней памяти. Как видно из временной диаграммы, сигнал CAWP формируется дважды за один машинный цикл (S1P2–S2P1 и S4P2–S5P1) и используется для управления процессом обращения к внешней памяти.

Большинство команд MK51 выполняется за один машинный цикл. Некоторые команды, оперирующие с 2-байтными словами или связанные с обращением к внешней памяти, выполняются за два машинных цикла. Только команды деления и умножения требуют четырех машинных циклов. На основе этих особенностей работы устройства управления MK51 производится расчет времени исполнения прикладных программ.

3.2. Порты ввода/вывода информации

Все четыре порта MK51 предназначены для ввода или вывода информации побайтно. Схемотехника портов ввода/вывода MK51 для одного бита показана на рис. 3.4 (порты 1 и 2 имеют примерно такую же структуру, как и порт 3). Каждый порт содержит управляемые регистры-защелки, 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. 3.3. Альтернативные функции могут быть задействованы путем записи 1 в соответствующие биты регистра-защелки (Р3.0–Р3.7) порта 3.

Порт 0 является двунаправленным, а порты 1, 2 и 3 – квазидвунаправленными. Каждая линия портов может быть использована независимо для ввода или вывода информации. Для того чтобы некоторая линия порта использовалась для ввода, в D-триггер регистра-защелки порта должна быть записана 1, которая закрывает МОП-транзистор выходной цепи.

По сигналу СБР в регистры-защелки всех портов автоматически записываются единицы, настраивающие их тем самым на режим ввода.

Все порты могут быть использованы для организации ввода/вывода информации по двунаправленным линиям передачи. Однако порты 0 и 2 не могут быть использованы для этой цели в случае, если МК-система имеет внешнюю память, связь с которой организуется через общую разделяемую шину адреса/данных, работающую в режиме временного мультиплексирования.

Запись в порт. При выполнении команды, которая изменяет содержимое регистра-защелки порта, новое значение фиксируется в регистре в момент S6P2 последнего цикла команды. Однако опрос содержимого регистра-защелки выходной схемой осуществляется во время фазы P1 и, следовательно, новое содержимое регистра-защелки появляется на выходных контактах порта только в момент S1P1 следующего машинного цикла.

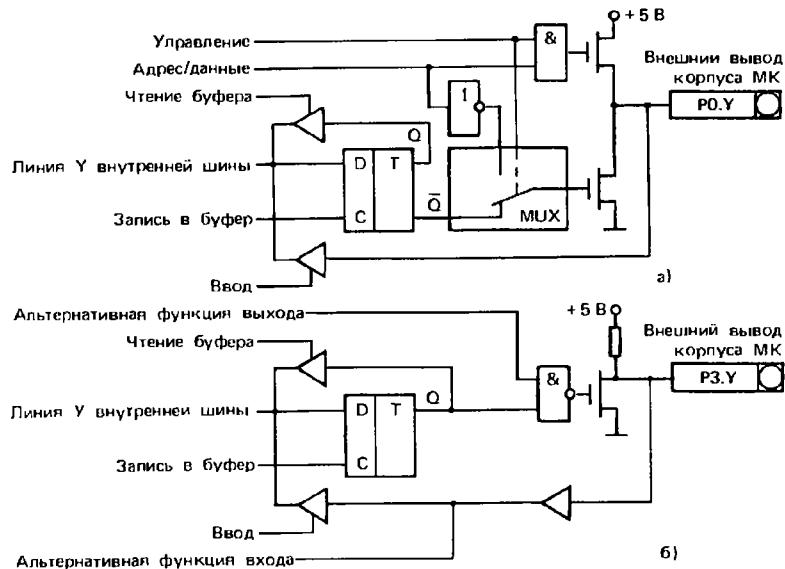


Рис. 3.4. Схемотехника портов ввода/вывода MK51:

а – порт 0; б – порт 3

Таблица 3.3. Альтернативные функции порта 3

Символ	Позиция	Имя и назначение
RD	P3.7	Чтение. Активный сигнал низкого уровня формируется аппаратурно при обращении к ВПД
WR	P3.6	Запись. Активный сигнал низкого уровня формируется аппаратурно при обращении к ВПД
T1	P3.5	Вход таймера/счетчика 1 или тест-вход
T0	P3.4	Вход таймера/счетчика 0 или тест-вход
INT1	P3.3	Вход запроса прерывания 1. Воспринимается сигнал низкого уровня или срез
INT0	P3.2	Вход запроса прерывания 0. Воспринимается сигнал низкого уровня или срез
TXD	P3.1	Выход передатчика последовательного порта в режиме УАПП. Выход синхронизации в режиме сдвигающего регистра
RXD	P3.0	Вход приемника последовательного порта в режиме УАПП. Ввод/вывод данных в режиме сдвигающего регистра

Нагрузочная способность портов. Выходные линии портов 1, 2 и 3 могут работать на одну ТТЛ-схему. Линии порта 0 могут быть нагружены на два входа ТТЛ-схемы каждая. Линии порта 0 могут работать и на μ -МОП-схемы, однако при этом их необходимо подключать на источник электропитания через внешние нагрузочные резисторы за исключением случая, когда шина порта 0 используется в качестве шины адреса/данных внешней памяти.

Входные сигналы для MK51 могут формироваться ТТЛ-схемами или μ -МОП-схемами. Допустимо использование в качестве источников сигналов для MK51 схем с открытым коллектором или открытым стоком. Однако при этом время изменения входного сигнала при переходе из 0 в 1 окажется сильно затянутым.

Особенности работы портов. Обращение к портам ввода/вывода возможно с использованием команд, оперирующих с 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". Этот режим обращения предполагает ввод сигналов не с внешних выводов порта, а из его регистра-защелки, что позволяет исключить неправильное считывание ранее выведенной информации.

Подобный механизм обращения к портам реализован в следующих командах:

ANL – логическое И, например ANL P1,A;

ORL – логическое ИЛИ, например ORL P2,A;

XRL – исключающее ИЛИ, например XRL P3,A;

JBC – переход, если в адресуемом бите единица, и последующий сброс бита, например JBC P1.1, LABEL;

CPL – инверсия бита, например CPL P3.3;

INC – инкремент порта, например INC P2;

DEC – декремент порта, например DEC P2;

DJNZ – декремент порта и переход, если его содержимое не равно нулю, например DJNZ P3, LABEL;

MOV PX.Y, C – передача бита переноса в бит Y порта X;

SET PX.Y – установка бита Y порта X;

CLR PX.Y – сброс бита Y порта X.

Совсем не очевидно, что последние три команды в приведенном списке являются командами "чтение-модификация-запись". Однако это именно так. По этим командам сначала считывается байт из порта, а затем записывается новый байт в регистр-защелку.

Причиной, по которой команды "чтение-модификация-запись" обеспечивают разделенный доступ к регистру-защелке порта и к внешним выводам порта, является необходимость исключить возможность неправильного прочтения уровней сигналов на внешних выводах. Предположим для примера, что линия Y порта X соединяется с базой мощного

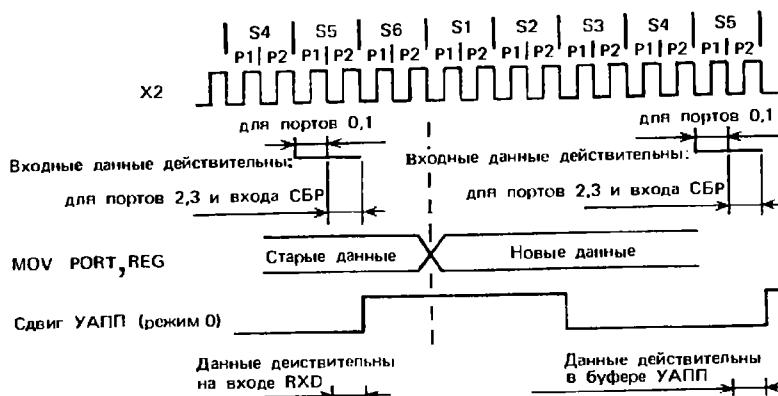


Рис. 3.5. Временные диаграммы операций ввода/вывода

транзистора и выходной сигнал на ней предназначен для его управления. Когда в данный бит записана 1, то транзистор включается. Если для проверки состояния исполнительного механизма (в нашем случае – мощного транзистора) прикладной программе требуется прочитать состояние выходного сигнала в том же бите порта, то считывание сигнала с внешнего вывода порта, а не из D-триггера регистра-защелки порта приведет к неправильному результату: единичный сигнал на базе транзистора имеет относительно низкий уровень и будет интерпретирован в МК как сигнал 0. Команды "чтение-модификация-запись" реализуют считывание из регистра-защелки, а не с внешнего вывода порта, что обеспечивает получение правильного значения 1.

На рис. 3.5 приведены временные диаграммы, иллюстрирующие процесс выполнения операций ввода/вывода информации через порты MK51.

3.3.

Доступ к внешней памяти

В микроконтроллерных системах, построенных на основе MK51, возможно использование двух типов внешней памяти: постоянной памяти программ (ВПП) и оперативной памяти данных (ВПД). Доступ к ВПП осуществляется при помощи управляющего сигнала РВПП, который выполняет функцию строб-сигнала чтения. Доступ к ВПД обеспечивается управляемыми сигналами ЧТ и ЗП, которые формируются в линиях P3.7 и P3.6 при выполнении портом 3 альтернативных функций (см. табл. 3.3).

При обращении к ВПП всегда используется 16-битный адрес. Доступ к ВПД возможен с использованием 16-битного адреса (MOVX A,@ DPTR) или 8-битного адреса (MOVX A,@ Ri).

В любых случаях использования 16-битного адреса старший байт

адреса фиксируется (и сохраняется неизменным в течение одного цикла записи или чтения) в регистре-защелке порта 2.

Если очередной цикл внешней памяти (MOVX A,@ DPTR) следует не сразу же за предыдущим циклом внешней памяти, то неизменяемое содержимое регистра-защелки порта 2 восстанавливается в следующем цикле. Если используется 8-битный адрес (MOVX A,@ Ri), то содержимое регистра-защелки порта 2 остается неизмененным на его внешних выводах в течение всего цикла внешней памяти.

Через порт 0 в режиме временного мультиплексирования осуществляется выдача младшего байта адреса и передача байта данных. Сигнал САВП должен быть использован для записи байта адреса во внешний регистр. Затем в цикле записи выводимый байт данных появляется на внешних выводах порта 0 только перед появлением сигнала ЗП. В цикле чтения выводимый байт данных принимается в порт 0 по фронту стробирующего сигнала ЧТ.

При любом обращении к внешней памяти устройство управления

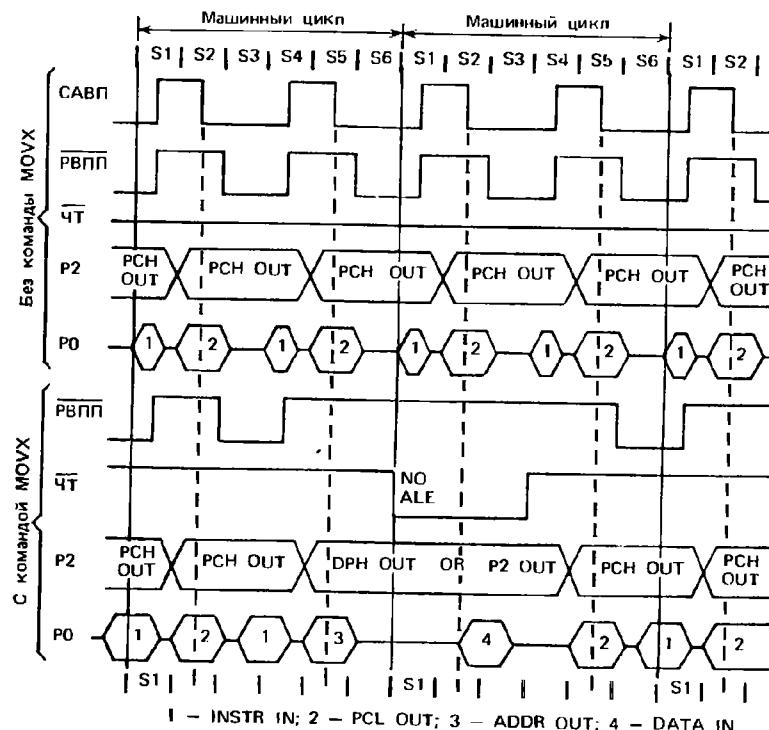


Рис. 3.6. Временные диаграммы операций с обращением к внешней памяти

MK51 загружает в регистр-защелку порта 0 код OFFH, стирая тем самым информацию, которая могла в нем храниться.

Доступ к ВПП возможен при выполнении двух условий: либо на вход отключения резидентной памяти программ (ОРПП) подается активный сигнал, либо содержимое счетчика команд превышает значение OFFFH. Наличие сигнала ОРПП необходимо для обеспечения доступа к младшим 4К адресам адресного пространства ВПП при использовании MK31 (микроконтроллера без резидентной памяти программ).

Временные диаграммы на рис. 3.6 иллюстрируют процесс генерации управляющих сигналов САВП и РВПП при обращении к внешней памяти.

Основная функция сигнала САВП – обеспечить временное согласование передачи из порта 0 на внешний регистр младшего байта адреса в цикле чтения из ВПП. Сигнал САВП приобретает значение 1 дважды в каждом машинном цикле. Это происходит даже тогда, когда в цикле выборки нет обращения к ВПП. Доступ к ВПД возможен только в том случае, если сигнал САВП отсутствует. Первый сигнал САВП во втором машинном цикле команды MOVX блокируется. Следовательно, любой МК-системе, не использующей ВПД, сигнал САВП генерируется с постоянной частотой, равной 1/16 частоты резонатора, и может быть использован для синхронизации внешних устройств или для реализации различных временных функций.

При обращении к РПП сигнал РВПП не генерируется, а при обращении к ВПП он выполняет функцию строб-сигнала чтения. Полный цикл чтения ВПД, включая установку и снятие сигнала ЧТ, занимает 12 периодов резонатора.

Временные диаграммы на рис. 3.7 и 3.8 иллюстрируют процесс выборки команды из ВПП и работу с ВПД в режимах чтения и записи соответственно.

Особый режим работы MK51. Содержимое памяти программ MK51 заполняется единожды на этапе разработки МК-системы и не может быть модифицировано в завершенном (конечном) изделии. По этой причине микроконтроллеры не являются машинами классической "фон-неймановской" архитектуры. Оперативная память данных (резидентная или внешняя) не может быть использована для хранения кодов программы, так как в МК выборка команд производится только из области адресов памяти программ. Эта особенность архитектуры МК объясняется тем, что в большинстве применений МК требуется наличие одной неизменяемой прикладной программы, хранимой в ПЗУ, наличие ОЗУ небольшой емкости для временного хранения переменных и эффективных, а следовательно, разных методов адресации памяти программ и памяти данных.

Однако на этапе разработки и отладки прикладных программ машина "фон-неймановского" типа оказывается очень удобной, так как позволяет разработчику оперативно изменять коды прикладной программы, размещаемой в ОЗУ. С этой целью МК-система может быть модифицирована для совмещения адресного пространства ВПП и ВПД путем под-

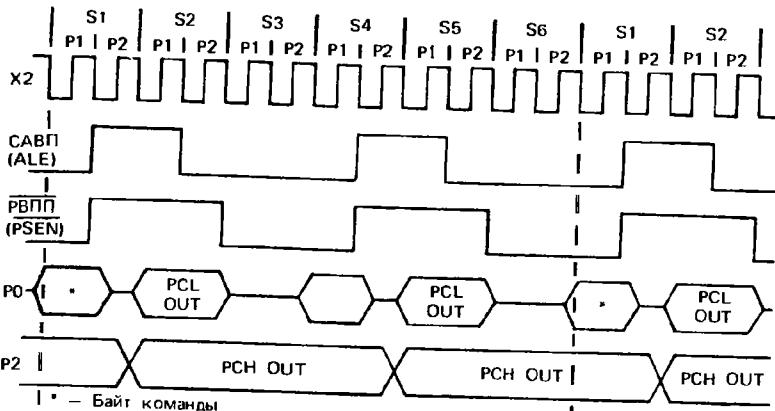
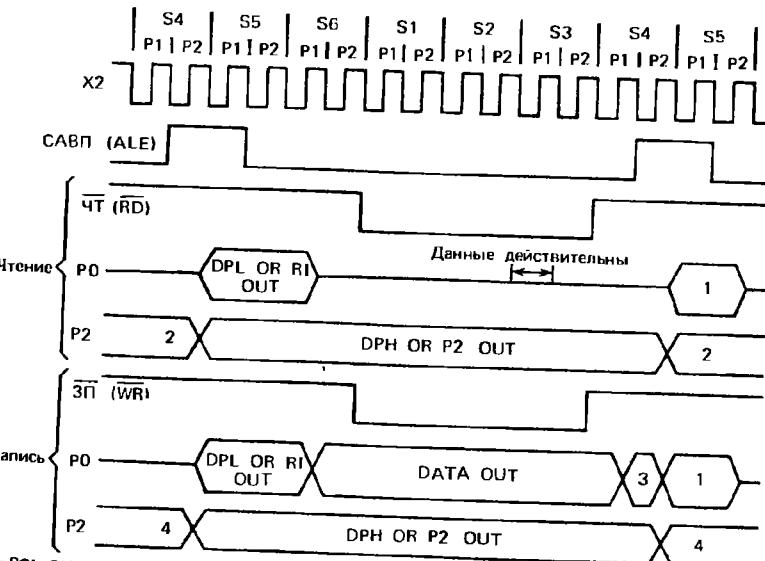


Рис. 3.7. Временная диаграмма выборки команды из ВПП



1—PCL OUT, если используется ВПП; 2—DPH OR P2 OUT; 3—PCL OUT; 4—PCH OR P2

Рис. 3.8. Временная диаграмма работы с ВПД

Ключения внешней логики, как показано на рис. 3.9. Здесь на выходе схемы И формируется строб-сигнал чтения, который может быть использован для объединения памяти программ и памяти данных во внешнем ОЗУ. При этом необходимо учитывать, что в MK51 на склонном уровне реализуются пять различных и независимых механизмов

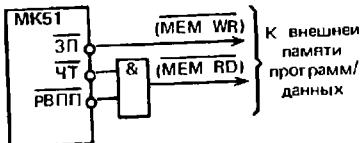


Рис. 3.9. Схема совмещения адресного пространства ВИП и ВПД

Подобный способ организации управления внешней памятью может быть использован в тех применениях MK51, где требуется оперативная перезагрузка или модификация прикладных программ (с помощью УВВ), как в ЭВМ классической архитектуры.

3.4. Таймер/счетчик

Два программируемых 16-битных таймера/счетчика (T/C0 и T/C1) могут быть использованы в качестве таймеров или счетчиков внешних событий. При работе в качестве таймера содержимое T/C инкрементируется в каждом машинном цикле, т.е. через каждые 12 периодов резонатора. При работе в качестве счетчика содержимое T/C инкрементируется под воздействием перехода из 1 в 0 внешнего входного сигнала, подаваемого на соответствующий (T0, T1) вывод MK51. Опрос значения внешнего входного сигнала выполняется в момент времени SSP2 каждого машинного цикла. 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 момент SSP1 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 частоты резонатора. На длительность периода входных сигналов ограничений сверху нет. Для гарантированного прочтения входного сигнала он должен удерживать значение 1 как минимум в течение одного машинного цикла MK51.

Для управления режимами работы T/C и для организации взаимодействия таймеров с системой прерывания используются два регистра специальных функций (РРТС и РУСТ), описание которых приводится в табл. 3.4 и 3.5 соответственно. Как следует из описания управляющих бит РРТС, для обоих T/C режимы работы 0, 1 и 2 одинаковы. Режимы 3 для T/C0 и T/C1 различны. Рассмотрим кратко работу T/C во всех четырех режимах.

Режим 0. Перевод любого T/C в режим 0 делает его похожим на таймер MK48 (8-битный счетчик), на вход которого подключен 5-битный преобразователь частоты на 32. Работу T/C в режиме 0 на примере T/C1 иллюстрирует рис. 3.10, а. В этом режиме таймерный регистр имеет

адресации для доступа к ПРР, РПД, ВПП, ВПД и блоку регистров специальных функций. Вследствие этого перемещаемая версия прикладной программы, которая отапливается в среде внешней памяти программ/данных, будет отличаться от загружаемой в РП (окончательной) версии программы.

Таблица 3.4. Регистр режима работы таймера/счетчика

Символ	Позиция	Имя и назначение
GATE	TMOD.7	Управление блокировкой. Если бит установлен, то таймер/счетчик "x" разрешен до тех пор, пока на входе "INT x" высокий уровень и бит управления "TRx" установлен. Если бит сброшен, то T/C разрешается, как только бит управления "TRx" устанавливается
C/T	TMOD.6	Бит выбора режима таймера или счетчика событий. Если бит сброшен, то работает таймер от внутреннего источника сигналов и TMOD.2 синхронизации. Если бит установлен, то работает счетчик от внешних сигналов на входе "Tx"
M1	TMOD.5 для T/C1 и TMOD.1 для T/C0	Режим работы (см. примечание)
M0	TMOD.4 для T/C1 и TMOD 0 для T/C0	

Примечание.

M1	M0	Режим работы
0	0	Таймер MK48. "TLx" работает как 5-битный преобразователь
0	1	16-битный таймер/счетчик. "THx" и "TLx" включены последовательно
1	0	8-битный автонерезагружаемый таймер/счетчик. "THx" хранит значение, которое должно быть перезагружено в "TLx" каждый раз по заполнению
1	1	Таймер/счетчик 1 останавливается. Таймер/счетчик 0 : TL0 работает как 8-битный таймер/счетчик, и его режим определяется управляющими битами таймера 0. TH0 работает только как 8-битный таймер, и его режим определяется управляющими битами таймера 1

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 T/C), когда управляющий бит TR1 установлен в 1 и либо управляющий бит GATE (блокировка) равен 0, либо на внешний вывод запроса прерывания INT1 поступает уровень 1.

Отметим попутно, что установка бита GATE в 1 позволяет использовать таймер для измерения длительности импульсного сигнала, подаваемого на вход запроса прерывания.

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

Режим 2. В режиме 2 работа организована таким образом, что переносение (переход из состояния "все единицы" в состояние "все нули")

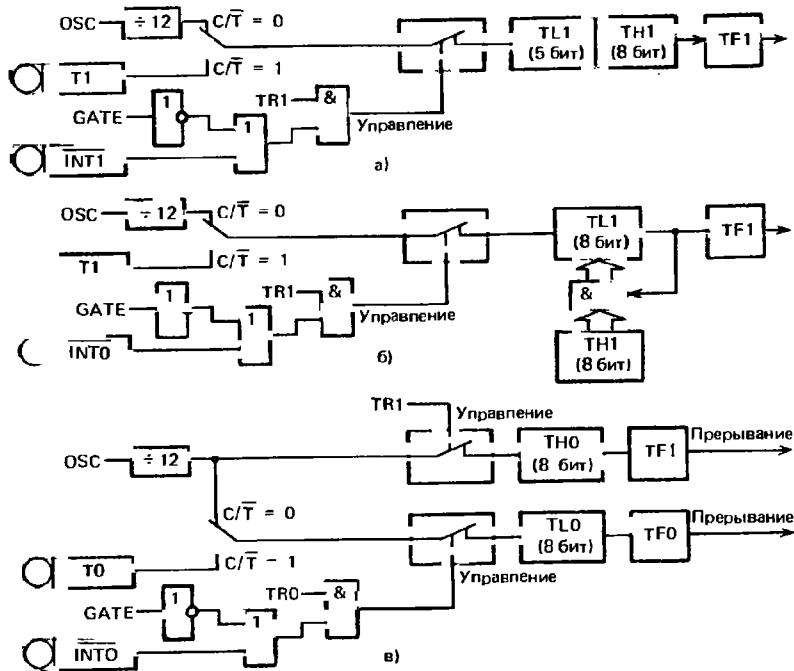


Рис. 3.10. Таймер/счетчик событий:

а – T/C1 в режиме 0: 13-битный счетчик; б – T/C1 в режиме 2: 8-битный автоперезагружаемый счетчик; в – T/C0 в режиме 3: два 8-битных счетчика

8-битного счетчика TL1 приводит не только к установке флага TF1 (рис. 3.10, б), но и автоматически перезагружает в TL1 содержимое старшего байта (TH1) таймерного регистра, которое предварительно было задано программным путем. Перезагрузка оставляет содержимое TH1 незмененным. В режиме 2 T/C0 и T/C1 работают совершенно одинаково.

Режим 3. В режиме 3 T/C0 и T/C1 работают по-разному. T/C1 сохраняет неизменным свое текущее содержимое. Иными словами, эффект такой же, как и при сбросе управляющего бита TR1 в нуль.

Работу T/C0 в режиме 3 иллюстрирует рис. 3.10, в. В режиме 3 TL0 и TH0 функционируют как два независимых 8-битных счетчика. Работу TL0 определяют управляющие биты T/C0 (C/T, GATE, TR0), входной сигнал INTO и флаг переполнения TF0. Работу TH0, который может выполнять только функции таймера (подсчет машинных циклов МК), определяет управляющий бит TR1. При этом TH0 использует флаг переполнения TF1.

Режим 3 используется в тех случаях применения MK51, когда требуется наличие дополнительного 8-битного таймера или счетчика собы-

Таблица 3.5. Регистр управления/статуса таймера

Символ	Позиция	Имя и назначение
TF1	TCON.7	Флаг переполнения таймера 1. Устанавливается аппаратурно при переполнении таймера/счетчика. Сбрасывается при обслуживании прерывания аппаратурно.
TR1	TCON.6	Бит управления таймера 1. Устанавливается/сбрасывается программой для пуска/останова
TF0	TCON.5	Флаг переполнения таймера 0. Устанавливается аппаратурно. Сбрасывается при обслуживании прерывания
TR0	TCON.4	Бит управления таймера 0. Устанавливается/сбрасывается программой для пуска/останова таймера/счетчика
IE1	TCON.3	Флаг фронта прерывания 1. Устанавливается аппаратурно, 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 (срез/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 (срез/низкий уровень)

тий. Можно считать, что в режиме 3 MK51 имеет в своем составе три таймера/счетчика. В том случае, если T/C0 используется в режиме 3, T/C1 может быть или включен, или выключен, или переведен в свой собственный режим 3, или может быть использован последовательным портом в качестве генератора частоты передачи, или, наконец, может быть использован в любом применении, не требующем прерывания.

3.5. Последовательный интерфейс

Через универсальный асинхронный приемопередатчик (УАПП) осуществляется прием и передача информации, представленной последовательным кодом (младшими битами вперед), в полном дуплексном режиме обмена. В состав УАПП, называемого часто последовательным портом, входят принимающий и передающий сдвигающие регистры, а также специальный буферный регистр (SBUF) приемопередатчика. Запись байта в буфер приводит к автоматической переписи байта в сдвигающий регистр передатчика и инициирует начало передачи байта. Наличие буферного регистра приемника позволяет совмещать операцию чтения ранее принятого байта с приемом очередного байта. Если к моменту окончания приема байта предыдущий байт не был считан из SBUF, то он будет потерян.

Последовательный порт MK51 может работать в четырех различных режимах.

Г а б л и ц а 3.6. Регистр управления/статуса УАПП

Режим 0. В этом режиме информация и передается и принимается через внешний вывод входа приемника (RXD). Принимаются или передаются 8 бит данных. Через внешний вывод выхода передатчика (TXD) выдаются импульсы сдвига, которые сопровождают каждый бит. Частота передачи бита информации равна 1/12 частоты резонатора.

Режим 1. В этом режиме передаются через TXD или принимаются из RXD 10 бит информации: старт-бит (0), 8 бит данных и стоп-бит (1). Скорость приема/передачи – величина переменная и задается таймером.

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

Режим 3. Режим 3 совпадает с режимом 2 во всех деталях, за исключением частоты приема/передачи, которая является величиной переменной и задается таймером.

3.5.1. Регистр управления/статуса УАПП

Управление режимом работы УАПП осуществляется через специальный регистр с символическим именем SCON. Этот регистр содержит не только управляющие биты, определяющие режим работы последовательного порта, но и девятый бит принимаемых или передаваемых данных (RB8 и TB8) и биты прерывания приемопередатчика (RI и TI).

Функциональное назначение бит регистра управления/статуса УАПП приводится в табл. 3.6.

Прикладная программа путем загрузки в старшие биты специального SCON 2-битного кода определяет режим работы УАПП. Во всех четырех режимах работы передача из УАПП инициируется любой командой, в которой буферный регистр SBUF указан как получатель байта. Прием в УАПП в режиме 0 осуществляется при условии, что RI = 0 и REN = 1. В режимах 1, 2, 3 прием начинается с приходом старт-бита, если REN = 1.

В бите TB8 программно устанавливается значение девятого бита данных, который будет передан в режиме 2 или 3. В бите RB8 фиксируется в режимах 2 и 3 девятый принимаемый бит данных. В режиме 1, если SM2 = 0, в бит RB8 заносится стоп-бит. В режиме 0 бит RB8 не используется.

Флаг прерывания передатчика TI устанавливается аппаратурно в конце периода передачи восьмого бита данных в режиме 0 и в начале периода передачи стоп-бита в режимах 1, 2 и 3. Соответствующая подпрограмма обслуживания прерывания должна сбрасывать бит TI.

Флаг прерывания приемника RI устанавливается аппаратурно в конце периода приема восьмого бита данных в режиме 0 и в середине периода

Символ	Позиция	Имя и назначение
SM0	SCON.7	Биты управления режимом работы УАПП. Устанавливаются/сбрасываются программно (см. примечание)
SM1	SCON.6	
SM2	SCON.5	Бит управления режимом УАПП. Устанавливается программно для запрета приема сообщения, в котором девятый бит имеет значение 0
REN	SCON.4	Бит разрешения приема. Устанавливается/сбрасывается программно для разрешения/запрета приема последовательных данных
TB8	SCON.3	Передача бита 8. Устанавливается/сбрасывается программно для задания девятого передаваемого бита в режиме УАПП-9
RB8	SCON.2	Прием бита 8. Устанавливается/сбрасывается аппаратурно для фиксации девятого принимаемого бита в режиме УАПП-9 бит
TI	SCON.1	Флаг прерывания передатчика. Устанавливается аппаратурно при окончании передачи байта. Сбрасывается программно после обслуживания прерывания
RI	SCON.0	Флаг прерывания приемника. Устанавливается аппаратурно при приеме байта. Сбрасывается программно после обслуживания прерывания
П р и м е ч а н и е		
SM0	SM1	Режим работы УАПП
0	0	Сдвигающий регистр расширения ввода/вывода
0	1	УАПП-8бит. Изменяется скорость передачи
1	0	УАПП-9 бит. Фиксированная скорость передачи
1	1	УАПП-9 бит. Изменяется скорость передачи

приема стоп-бита в режимах 1, 2 и 3. Подпрограмма обслуживания прерывания должна сбрасывать бит RI.

3.5.2. Работа УАПП в мульти микроконтроллерных системах

В системах децентрализованного управления, которые используются для управления и регулирования в топологически распределенных объектах (например, прокатных станах, электроподвижном составе железных дорог и метрополитена, сборочных конвейерах и линиях гибких автоматизированных производств), возникает задача обмена информацией между множеством микроконтроллеров, объединенных в локальную вычислительно-управляющую сеть. Как правило, локальные сети на основе МК51 имеют магистральную архитектуру с разделяемым моноканалом (коаксиальный кабель, витая пара, оптическое волокно), по которому осуществляется обмен информацией между МК [31].

В регистре специальных функций SCON микроконтроллера имеется управляющий бит SM2, который в режимах 2 и 3 УАПП позволяет относительно простыми средствами реализовать межконтроллерный обмен информацией в локальных управляющих сетях.

Механизм межконтроллерного обмена информацией через последовательный порт MK51 построен на том, что в режимах 2 и 3 программируемый девятый бит данных при приеме фиксируется в бите RB8. УАПП может быть запрограммирован таким образом, что при получении стоп-бита прерывание от приемника будет возможно только при условии RB8 = 1. Это выполняется установкой управляющего бита SM2 в регистре SCON.

Поясним процесс межконтроллерного обмена информацией на примере. Пусть ведущему МК требуется передать блок данных некоторому (или нескольким) ведомому МК. С этой целью ведущий МК в протокольном режиме "широковещательной" передачи (всем ведомым МК) выдает в моноканал байт-идентификатор абонента (код адреса МК-получателя), который отличается от байтов данных только тем, что в его девятом бите содержится 1. Программа реализации протокола сетевого обмена информацией должна быть построена таким образом, чтобы при получении байта-идентификатора (RB8 = 1) во всех ведомых МК произошли прерывание прикладных программ и вызов подпрограммы сравнения байта-идентификатора с кодом собственного сетевого адреса. Адресуемый МК сбрасывает свой управляющий бит SM2 и готовится к приему блока данных. Остальные ведомые МК, адрес которых не совпал с кодом байта-идентификатора, оставляют неизменным состояние SM2 = 1 и передают управление основной программе. При SM2 = 1 информационные байты, передаваемые по моноканалу и поступающие в УАПП ведомых МК, прерывания не вызывают, т.е. игнорируются.

В режиме 1 УАПП автономного МК управляющий бит SM2 используется для контроля истинности стоп-бита (при SM2 = 1 прерывание не произойдет до тех пор, пока не будет получено истинное (единичное) значение стоп-бита). В режиме 0 бит SM2 не используется и должен быть сброшен.

3.5.3. Скорость приема/передачи

Скорость приема/передачи, т.е. частота работы УАПП в различных режимах, определяется различными способами.

В режиме 0 частота передачи зависит только от резонансной частоты квадрового резонатора $f_0 = f_{\text{рез}}/12$. За один машинный цикл последовательный порт передает один бит информации.

В режимах 1, 2 и 3 скорость приема/передачи зависит от значения управляющего бита SMOD в регистре специальных функций РУМ (табл. 3.7).

В режиме 2 частота передачи определяется выражением $f_2 = (2^{\text{SMOD}}/64)f_{\text{рез}}$. Иными словами, при SMOD = 0 частота передачи равна $(1/64)f_{\text{рез}}$, а при SMOD = 1 равна $(1/32)f_{\text{рез}}$.

Таблица 3.7. Регистр управления мощностью РУМ

Символ	Позиция	Наименование и функция
SMOD	PCON.7	Удвоенная скорость передачи. Если бит установлен в 1, то скорость передачи вдвое больше, чем при SMOD = 0
-	PCON.6	Не используются
-	PCON.5	
-	PCON.4	
GF1	PCON.3	Флаги, специфицируемые пользователем (флаги общего назначения)
GFO	PCON.2	
PD	PCON.1	Бит пониженной мощности. При установке бита в 1 МК переходит в режим пониженной потребляемой мощности
IDL	PCON.0	Бит холостого хода. Если бит установлен в 1, то МК переходит в режим холостого хода

Примечание. При одновременной записи 1 в PD и IDL бит PD имеет преимущество. Сброс содержимого РУМ выполняется путем загрузки в него кода 0XXXX0000.

В режимах 1 и 3 в формировании частоты передачи кроме управляющего бита SMOD принимает участие таймер 1. При этом частота передачи зависит от частоты переполнения (OVT1) и определяется следующим образом: $f_{1,3} = (2^{\text{SMOD}}/32)f_{\text{OVT1}}$. Прерывание от таймера 1 в этом случае должно быть заблокировано. Сам T/C1 может работать и как таймер, и как счетчик событий в любом из трех режимов. Однако наиболее удобно использовать режим таймера с автоперезагрузкой (старшая четверть TMOD = 0010B). При этом частота передачи определяется выражением $f_{1,3} = (2^{\text{SMOD}}/32)(f_{\text{рез}}/12) (256 - (\text{TH1}))$. В табл. 3.8 приводится описание способов настройки T/C1 для получения типовых частот передачи данных через УАПП.

3.5.4. Особенности работы УАПП в различных режимах

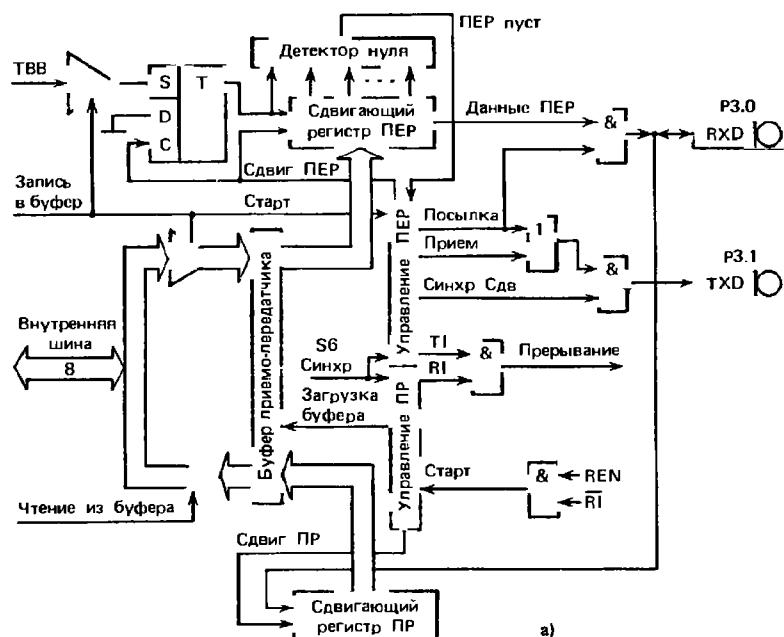
Режим 0. На рис. 3.11 показаны упрощенная структурная схема УАПП и временная диаграмма его работы в режиме 0. Данные передаются и принимаются через вывод RXD. Через вывод TXD выдаются синхросигналы сдвига.

Передача начинается любой командой, но которой в SBUF поступает байт данных. В момент времени S6P2 устройство управления MK51 по сигналу Запись в буфер записывает байт в сдвигающий регистр передатчика, устанавливает триггер девятого бита и запускает блок управления передачей, который через один машинный цикл вырабатывает разрешающий сигнал Пуск. При этом в момент S6P2 каждого машинного цикла содержимое сдвигающего регистра сдвигается вправо (младшими битами вперед) и поступает на вывод RXD. В освобождающиеся старшие биты сдвигающего регистра передатчика записываются нули.

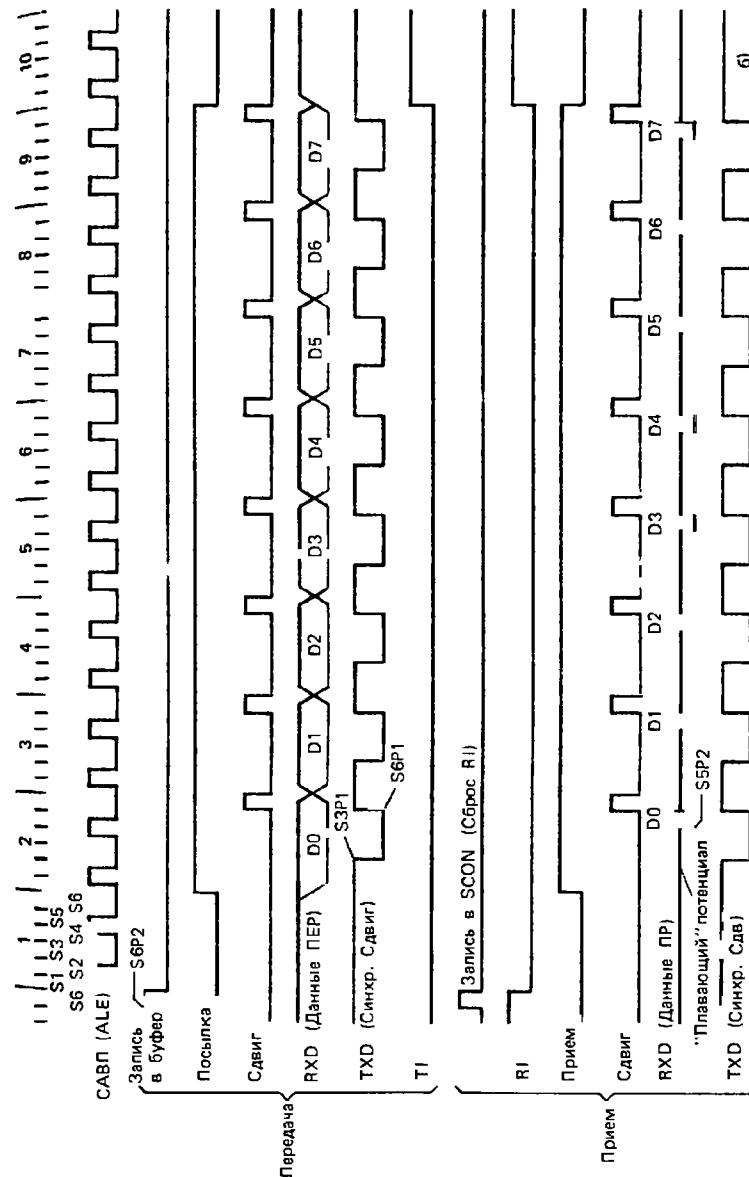
При получении от детектора нуля сигнала *Передатчик* пуст блок управления передатчиком снимает сигнал *Посылка* и устанавливает флаг TI (момент S1P1 десятого машинного цикла после поступления сигнала *Запись в буфер*).

Прием начинается при условии $REN = 1$ и $RI = 0$. В момент $S6P2$ следующего машинного цикла блок управления приемником формирует разрешающий сигнал *Прием*, по которому на выход TXD передаются синхросигналы сдвига и в сдвигающем регистре приемника начинают формироваться значения бит данных, которые считываются с входа RXD в моменты $S5P2$ каждого машинного цикла. В момент $S1P1$ десятого машинного цикла после сигнала *Запись в SC0N* блок управления приемником переписывает содержимое сдвигающего регистра в буфер, снимает разрешающий сигнал *Прием* и устанавливает флаг RI .

Режим 1. На рис. 3.12 показаны структурная схема (рис. 3.12, а) и временные диаграммы работы УАПП при приеме и передаче данных (рис. 3.12, б, в). Через вывод TXD УАПП передает, а с вывода RXD принимает 10 бит: старт-бит (0), 8 бит данных и стоп-бит (1). При приеме стоп-бит поступает в бит RB8 регистра SCON.



а – структурная схема; *б* – временная диаграмма работы



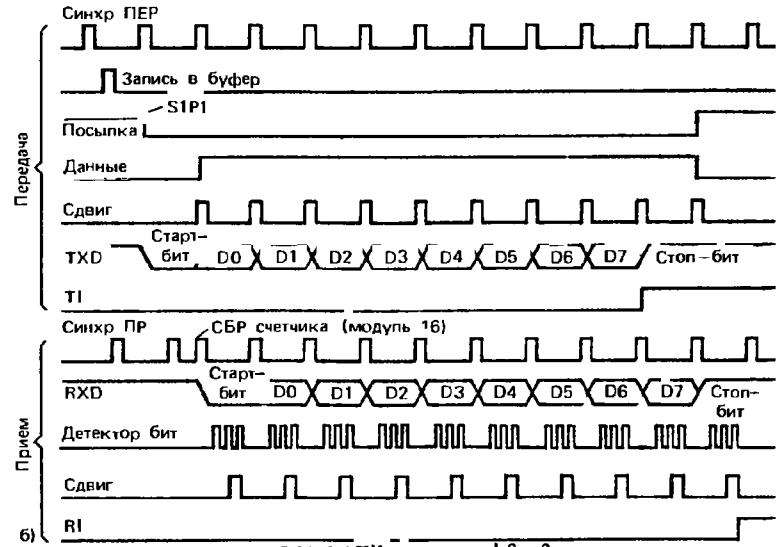
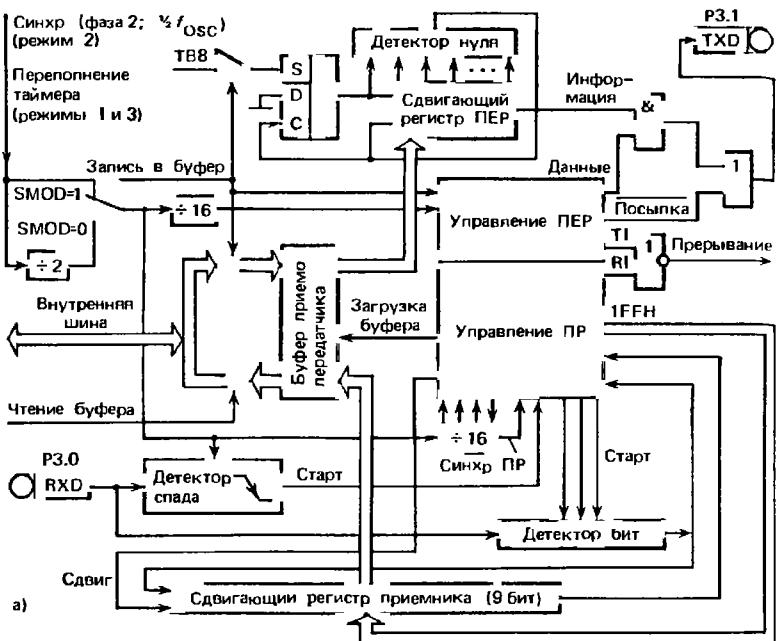
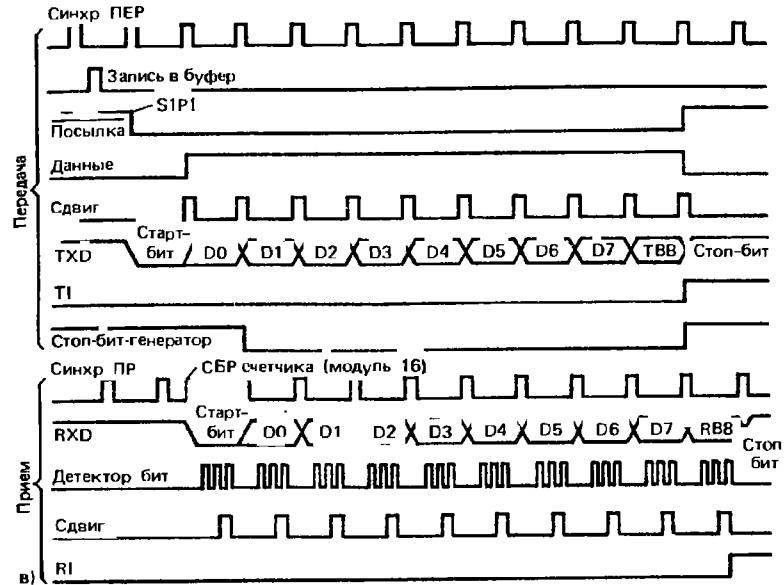


Рис. 3.12. УАПП в режимах 1, 2 и 3.
а – структурная схема; б – временная диаграмма в режиме 1; в – временная диаграмма в режимах 2 и 3



Передача инициируется любой командой, в которой получателем байта является регистр SBUF. Генерируемый при этом управляющий сигнал Запись в буфер загружает 1 в девятый бит сдвигающего регистра передатчика, запускает блок управления передачей и в момент времени S1P1 формирует разрешающий сигнал Посылка. По этому сигналу на вывод TXD сначала поступает старт-бит, а затем (по разрешающему сигналу Данные) биты данных. Каждый период передачи бита равен 16 тактам внутреннего счетчика.

Прием начинается при обнаружении перехода сигнала на входе RxD из состояния 1 в состояние 0. Для этого под управлением внутреннего счетчика вход RxD опрашивается 16 раз за период представления бита. Как только переход из 1 в 0 на входе RxD обнаружен, в сдвигующий регистр приемника загружается код 1FFH, внутренний счетчик по модулю 16 немедленно сбрасывается и перезапускается для выравнивания его переходов с границами периодов представления принимаемых бит. Таким образом, каждый период представления бита делится на 16 периодов внутреннего счетчика. В состояниях 7, 8 и 9 счетчика в каждом периоде представления бита производится опрос сигнала на входе RxD. Считанное значение принимаемого бита – это то, которое было получено по меньшей мере дважды из трех замеров (мажоритарное голосование по принципу "два из трех"). Если значение, принятное в первом такте, не равно 0, то блок управления приемом вновь возвращается к поиску перехода из 1 в 0. Этот механизм обеспечивает подавление ложных

Таблица 3.8. Настройка таймера 1 для управления частотой работы УАПП

Частота приема/передачи (BAUD RATE)	Частота резонатора, МГц	SMOD	Таймер/счетчик 1			
			C/T	Режим (MODE)	Переза- гружаемое число	
Режим 0, макс:	1 МГц	12	X	X	X	X
Режим 2, макс: 375 кГц	12	1	X	X		X
Режимы 1,3:	62,5 кГц	12	1	0	2	0FFH
	19,2 кГц	11,059	1	0	2	0FDH
	9,6 кГц	11,059	0	0	2	0FDH
	4,8 кГц	11,059	0	0	2	0FAH
	2,4 кГц	11,059	0	0	2	0F4H
	1,2 кГц	11,059	0	0	2	0E8H
	137,5 Гц	11,059	0	0	2	1DH
	110 Гц	6	0	0	2	72H
	110 Гц	12	0	0	1	0FEEBH

(сбойных) старт-бит. Истинный старт-бит сдвигается в регистре приемника, и продолжается прием остальных бит посылки. Блок управления приемом сформирует сигнал *Загрузка буфера*, установит RB8 и флаг RI только в том случае, если в последнем такте сдвига выполняются два условия: бит RI = 0, или либо SM2 = 0, либо принятый стоп-бит равен 1. Если одно из этих двух условий не выполняется, то принятая последовательность бит теряется. В это время вне зависимости от того, выполняются указанные условия или нет, блок управления приемом вновь начинает отыскивать переход из 1 в 0 на входе RXD.

Режимы 2, 3. Через вывод TXD УАПП передает или с вывода RXD принимает 11 бит: старт-бит (0), 8 бит данных, программируемый девятый бит и стоп-бит (1). На временной диаграмме (рис. 3.12, в) показана работа УАПП при передаче и приеме данных в режимах 2 и 3. Как видно, режимы 2 и 3 отличаются от режима 1 только наличием девятого программируемого бита. Вследствие этого несколько изменяются условия окончания цикла приема: блок управления приемником сформирует управляющий сигнал *Загрузка буфера*, загрузит RB8 и установит флаг RI только в том случае, если в последнем такте сдвига выполняются два условия: бит RI = 0 и либо SM2 = 0, либо значение принятого девятого бита данных равно 1.

3.6. Система прерываний

Упрощенная схема прерываний MK51 показана на рис. 3.13.

Внешние прерывания INT0 и INT1 могут быть вызваны либо уровнем, либо переходом сигнала из 1 в 0 на входах MK51 в зависимости от значений управляющих бит ITO и IT1 в регистре TCON. От внешних прерываний устанавливаются флаги IE0 и IE1 в регистре TCON, которые инициируют вызов соответствующей подпрограммы обслуживания

прерывания. Сброс этих флагов выполняется аппаратурно только в том случае, если прерывание было вызвано по переходу (срезу) сигнала. Если же прерывание вызвано уровнем входного сигнала, то сбросом флага IE управляет соответствующая подпрограмма обслуживания прерывания путем воздействия на источник прерывания с целью снятия им запроса.

Флаги запросов прерывания от таймеров TFO и TFI сбрасываются автоматически при передаче управления почтограммам обслуживания. Флаги запросов прерывания RI и TI устанавливаются блоком управления УАПП аппаратурно, но сбрасываться должны программой.

Прерывания могут быть вызваны или отменены программой, так как все перечисленные флаги программно-доступны и могут быть установлены/сброшены программой с тем же результатом, как если бы они были установлены/сброшены аппаратурными средствами.

В блоке регистров специальных функций есть два регистра, предназначенных для управления режимом прерываний и уровнями приоритета. Форматы этих регистров, имеющих символические имена IE и IP описаны в табл. 3.9 и 3.10 соответственно. Возможность программной установки/сброса любого управляющего бита в этих двух регистрах делает систему прерываний MK51 исключительно гибкой.

Флаги прерываний опрашиваются в момент SSP2 каждого машинного цикла. Ранжирование прерываний по уровню приоритета выполняется в течение следующего машинного цикла. Система прерываний сформирует аппаратурно вызов (LCALL) соответствующей подпрограммы обслуживания, если она не заблокирована одним из следующих условий:

- 1) в данный момент обслуживается запрос прерывания равного или более высокого уровня приоритета;

Таблица 3.9. Регистр масок прерывания (PMII)

Символ	Позиция	Имя и назначение
EA	IE.7	Снятие блокировки прерываний. Сбрасывается программно для запрета всех прерываний независимо от состояний IE4 – IE0
–	IE.6	Не используются
–	IE.5	
ES	IE.4	Бит разрешения прерывания от УАПП. Установка/сброс программой для разрешения/запрета прерываний от флагов TI или RI
ET1	IE.3	Бит разрешения прерывания от таймера 1. Установка/сброс программой для разрешения/запрета прерываний от таймера 1
EX1	IE.2	Бит разрешения внешнего прерывания 1. Установка/сброс программой для разрешения/запрета прерываний
ETO	IE.1	Бит разрешения прерывания от таймера 0. Работает аналогично IE.3
EX0	IE.0	Бит разрешения внешнего прерывания 0. Работает аналогично IE.2

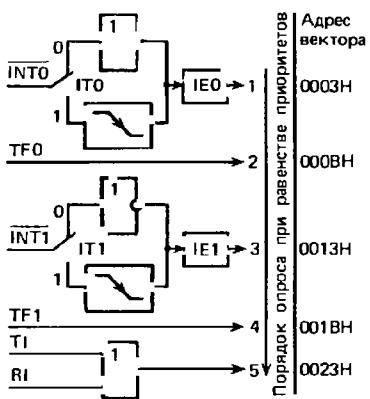


Рис. 3.13. Схема прерываний MK51

2) текущий машинный цикл — не последний в цикле выполняемой команды;

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

Отметим, что если флаг прерывания был установлен, но по одному из перечисленных выше условий не получил обслуживания и к моменту окончания блокировки уже был сброшен, то запрос прерывания теряется и нигде не запоминается.

По аппаратурно-сформированному коду LCALL система прерывания помещает в стек только содержимое счетчика команд (PC) и загружает

Таблица 3.10. Регистр приоритетов прерываний

Символ	Позиции	Имя и назначение
—	IP.7—IP.5	Не используются
PS	IP.4	Бит приоритета УАПП. Установка/сброс программой для присваивания прерыванию от УАПП высшего/нижнего приоритета
PT1	IP.3	Бит приоритета таймера 1. Установка/сброс программой для присваивания прерыванию от таймера 1 высшего/нижнего приоритета
PX1	IP.2	Бит приоритета внешнего прерывания 1. Установка/сброс программой для присваивания высшего/нижнего приоритета внешнему прерыванию INT1
PT0	IP.1	Бит приоритета таймера 0. Работает аналогично IP.3
PX0	IP.0	Бит приоритета внешнего прерывания 0. Работает аналогично IP.2

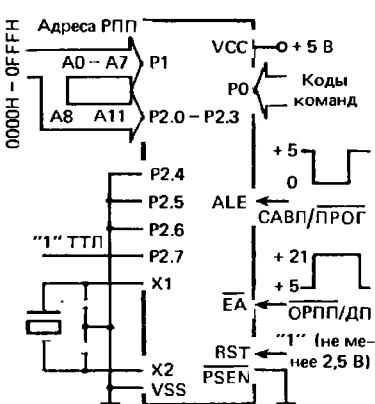


Рис. 3.14. Программирование MK51 (загрузка программ в РПП микроконтроллера)

в счетчик команд адрес вектора соответствующей подпрограммы обслуживания. По адресу вектора должна быть расположена команда безусловной передачи управления (JMP) к начальному адресу подпрограммы обслуживания прерывания. Подпрограмма обслуживания в случае необходимости должна начинаться командами записи в стек (PUSH) слова состояния программы (PSW), аккумулятора, расширителя, указателя данных и т.д. и заканчиваться командами восстановления из стека (POP). Подпрограммы обслуживания прерывания обязательно завершаются командой RETI, по которой в счетчик команд перезагружается из стека сохраненный адрес возврата в основную программу. Команда RET также возвращает управление прерванной основной программе, но при этом не снимает блокировку прерываний, что приводит к необходимости иметь программный механизм анализа окончания процедуры обслуживания данного прерывания.

3.7. Особые режимы работы MK51

3.7.1. Режим загрузки и верификации прикладных программ

Под воздействием внешних электрических сигналов MK51 может быть электрически запрограммирован или, иными словами, в РПП микроконтроллера могут быть загружены объектные коды прикладных программы. Содержимое РПП микроконтроллера может быть уничтожено выдержкой под ультрафиолетовым источником света (стирание) для последующего перепрограммирования. Микроконтроллер имеет средство защиты от программного "разбоя", обеспечивающее невозможность прочтения содержимого РПП в конечном изделии и, следовательно, сохранение профессиональных секретов разработчика прикладного программного обеспечения.

Загрузка программ в РПП. В режиме программирования MK51 должен работать на пониженной частоте (с резонатором 4–6 МГц) из-за необходимости мультиплексирования на внутреннейшине адресной и кодовой информации. На рис. 3.14 приведена схема подключения MK51 к программатору. Адрес ячейки РПП, в которую должен быть загружен байт прикладной программы, подается на выводы порта 1 и выводы P2.0–P2.3 порта 2. При этом загружаемый байт поступает в MK через порт 0. Выходы P2.4–P2.6 и PSEN должны быть заземлены, а на выводы P2.7 и RST необходимо подать уровень логической 1 (для входа RST уровень логической 1 – не менее 2,5 В, для остальных входов – стандартный уровень ТТЛ). На входе EA/VPP поддерживается уровень +5 В, но в момент загрузки байта он должен быть подключен к источнику напряжения с уровнем +21 В. В это время уровень на входе ALE/PROG должен быть не менее чем на 50 мс сброшен в 0. После этого напряжение на входе EA/VPP возвращается к уровню +5 В. Источник напряжения +21 В (VPP) должен быть очень хорошо стабилизирован,

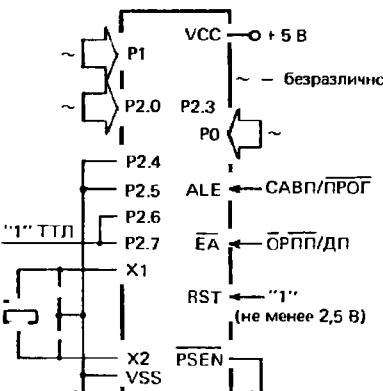


Рис. 3.15. Запись бита защиты РПП

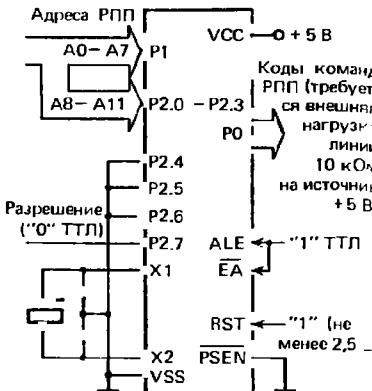


Рис. 3.16. Верификация программ
(проверка содержимого РПП)

так как превышение предельного значения +21,5 В на входе \overline{EA}/VPP приводит к необратимым повреждениям РПП.

Запись бита защиты. Бит защиты РПП, будучи установлен, запрещает доступ к РПП любыми внешними средствами. Схема записи бита защиты показана на рис. 3.15. Процедура записи бита защиты такая же, как и при загрузке программ в РПП, но на вывод P2.6 должен подаваться уровень 1. Сигналы на выводах портов P0, P1 и P2.0–P2.3 могут быть в любом состоянии. Однажды установленный бит защиты можно сбросить только путем полного стирания РПП под источником УФ-излучения.

Верификация программ. Если бит защиты не запрограммирован, то содержимое РПП может быть прочитано с целью проверки правильности загрузки прикладной программы либо по ходу программирования либо после окончания программирования MK51. Доступ к ячейкам РПП (рис. 3.16) осуществляется так же, как и при программировании РПП, за исключением того, что на вывод P2.7 подается сигнал 0, используемый в качестве сброс-сигнала чтения. Считанные коды команд и соответствующие им адреса РПП могут отображаться для визуального контроля на любом внешнем индикаторе и, кроме того, могут быть переданы в ОЗУ инструментальной ЭВМ для дистассемблирования.

Стирание РПП. Для стирания содержимого РПП микроконтроллер следует поместить под источник УФ-излучения с длиной волны менее $4 \cdot 10^{-7}$ м. Если кварцевая лампа имеет световую мощность 12 000 мкВт/см², а расстояние между источником света и МК равно 2–3 см, то выдержка 20–30 мин обеспечивает световую дозу, достаточную для надежного стирания РПП. После стирания в матрице РПП содержатся все единицы.

Так как в спектре солнечного света и люминесцентного освещения содержится излучение с длиной волны менее $4 \cdot 10^{-7}$ м, то пребывание

МК под воздействием этих источников света дольше установленного предела (около одной недели на солнечном свете или около трех лет при люминесцентном освещении) может привести к искажению содержимого РПП. Для защиты содержимого РПП от разрушения под воздействием света на окне корпуса МК укрепляют светонепроницаемый экран.

3.7.2. Работа MK51 в пошаговом режиме

На этапе отладки прикладной программы в прототипе МК-системы очень удобным для разработчика оказывается режим пошагового (покомандного) исполнения программы. Система прерываний MK51 позволяет реализовать этот режим работы путем использования нескольких дополнительных команд.

Как было описано ранее (см. § 3.6), внешний запрос прерывания не будет обслужен до тех пор, пока обрабатывается прерывание с равным приоритетом. Этот запрос будет воспринят лишь после того, как одна команда после команды RETI будет выполнена. Иными словами, однажды вызвав подпрограмму обслуживания прерывания, вызвать ее вновь невозможно до тех пор, пока хотя бы одна команда основной программы не будет исполнена. Использовать это свойство системы прерываний MK51 для реализации пошагового режима можно следующим образом: одно из внешних прерываний, например INT0 (вывод P3.2 микроконтроллера), запрограммировать для представления запроса уровнем сигнала (T0 = 0), а подпрограмма обслуживания этого прерывания должна заканчиваться последовательностью команд

JNB	P3.2,*	запросение уровня 1 на входе INT0
JB	P3.2,*	запросение уровня 0 на входе INT0
		возврат и выполнение самой команды
RETI		

(Здесь символом \square обозначено текущее содержимое счетчика команд). Теперь если на вывод INT0 подавать сигнал от клавиши ШАГ отладочного модуля системы, то MK51 по сигналу INT0 = 0 вызовет подпрограмму обслуживания внешнего прерывания 0 и будет в ней находиться до тех пор, пока не обнаружит на входе INT0 новый импульс (переход из 0 в 1 и снова в 0). По команде RETI произойдет возврат в основную программу, выполнение в ней одной команды и немедленный вызов подпрограммы обслуживания внешнего прерывания 0.

3.7.3. Сброс, режим холостого хода и режим пониженного энергопотребления

Сброс. В отличие от MK48 сброс MK51 осуществляется путем подачи на вход СБР (RST) сигнала 1. Для уверенного сброса MK51 этот сигнал 1 должен бытьдержан на входе СБР по меньшей мере в течение двух машинных циклов (24 периода резонатора). Квазидвунаправленные буферные схемы внешних выводов ALE и PSEN находятся при этом в режиме ввода. Под воздействием сигнала СБР сбрасывается содержи-

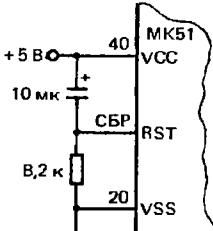


Рис. 3.17. Схема сброса при включении электропитания

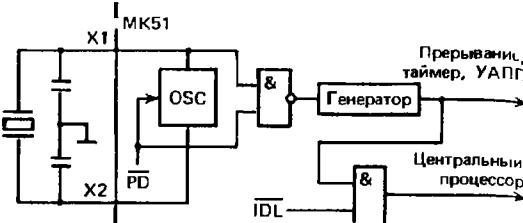


Рис. 3.18. Схема управления MK51 от сигналов IDL и PD

мое регистров: PC, ACC, B, PSW, DPTR, TMOD, TCON, T/C0, T/C1, II IP и SCON, в регистре PCON сбрасывается только старший бит, в регистр-указатель стека загружается код 07H, а в порты P0–P3 – коды OFFH. Состояние регистра SBUF неопределенное. Сигнал СБР не воздействует на содержимое ячеек РПД. Когда включается электропитание (VCC), содержимое РПД неопределенно, за исключением операции возврата из режима пониженного энергопотребления.

На рис. 3.17 показана схема автоматического формирования сигнала СБР при включении электропитания. Время, необходимое для полного заряда емкости, обеспечивает уверенный запуск резонатора и его работу в течение более чем двух машинных циклов.

Режим холостого хода. Любая команда, по которой устанавливается управляющий бит IDL(PCON.0) в регистре управления мощностью, переведет MK51 в режим холостого хода. При этом продолжает работу внутренний генератор синхросигналов (рис. 3.18). Все регистры сохраняют свое значение. На выводах всех портов удерживается то логическое состояние, которое на них было в момент перехода в режим холостого хода. На выводах ALE и PSEN формируется уровень 1.

Выйти из режима холостого хода можно или по сигналу СБР, или по прерыванию. Любой из разрешенных сигналов прерывания приведет к аппаратурному сбросу бита PCON.0 и прекратит тем самым режим холостого хода. После выполнения команды RETI (выход из подпрограммы обслуживания прерывания) будет исполнена команда, которая следует в программе за командой, переводящей MK51 в режим холостого хода. (В некоторых модификациях MK51 этот режим может отсутствовать.)

Режим пониженного энергопотребления. В тех применениях MK-систем, где потребление электроэнергии, а следовательно, габаритные размеры и масса источника электропитания являются одними из основных показателей качества изделия, возможно использование MK51 в режиме пониженного энергопотребления. Переход MK51 в этот режим возможен по команде, которая установит бит PCON.1 в регистре управления мощностью (см. табл. 3.7). В этом режиме останавливается генератор синхросигналов, содержимое РПД и регистров специальных функций

сохраняется, а на выходных контактах портов удерживаются значения, соответствующие содержимому их буферных регистров. Выходы сигналов ALE и PSEN сбрасываются. При этом электропитание осуществляется через вывод RST/VPD.

В режиме пониженного энергопотребления напряжение электропитания (VCC) может быть отключено. Перед выходом из режима оно должно быть восстановлено до номинального значения.

Выход из режима пониженного энергопотребления возможен только по сигналу RST. При этом переопределются все регистры специальных функций, но содержимое РПД не изменяется. (В некоторых модификациях MK51 этот режим может отсутствовать.)

Защита от падения напряжения. При немгновенных отказах блока электропитания MK-системы можно обеспечить сохранность содержимого РПД с помощью маломощного (батарейного) аварийного источника электропитания, подсоединяемого к выводу RST/VPD. Для этого MK при получении сообщения о грозящем падении напряжения VCC (прерывание от системы контроля электропитания) должен перегрузить в РПД основные параметры прерванного процесса и выдать сигнал, разрешающий подключение к выводу RST/VPD аварийного источника питания. Все эти процедуры MK должен успеть выполнить до того, как напряжение основного источника электропитания упадет ниже рабочего предела. После восстановления номинального значения напряжения в основной цепи электропитания выполняется системный сброс и источник аварийного электропитания может быть отключен.

3.8.

Система команд MK51

3.8.1. Общие сведения о системе команд

Система команд MK51 содержит 111 базовых команд, которые удобно разделить по функциональному признаку на пять групп: команды передачи данных, арифметических операций, логических операций, передачи управления и операций с битами.

Система команд MK51 многое общее и шире системы команд MK48, так как кроме всех команд MK48 в ее состав входят команды умножения, деления, вычитания, операций над битами, операций со стеком и расширенный набор команд передачи управления.

Большинство команд (94) имеют формат один или два байта и выполняются за один или два машинных цикла. При тактовой частоте 12 МГц длительность машинного цикла составляет 1 мкс.

На рис. 3.19 показаны 13 типов команд MK51. Первый байт команды любых типа и формата всегда содержит код операции (КОП). Второй и третий байты содержат либо адреса операндов, либо непосредственные операнды.

Типы операндов. Состав операндов MK51 шире, чем MK48, и включает в себя операнды четырех типов: биты, 4-битные цифры, байты и 16-битные слова.

D ₇ . . . D ₀		
1 КОП	D ₇ . . . D ₀	
2 КОП	# d	
3 КОП	ad	
4 КОП	bit	
5 КОП	rel	
6 a ₁₀ a ₉ a ₈ КОП	a ₇ . . . a ₀	D ₇ . . . D ₀
7 КОП	ad	# d
8 КОП	ad	rel
9 КОП	ads	add
10 КОП	# d	rel
11 КОП	bit	rel
12 КОП	ad16h	ad16l
13 КОП	# d16h	# d16l

Рис. 3.19. Типы команд MK51

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

Способы адресации данных. В MK51 используются такие же способы адресации данных, как и в MK48: прямая, непосредственная, косвенная и пеявица. Следует отметить, что при косвенном способе адресации РПД в отличие от MK48 используются все восемь бит адресных регистров R0 и R1.

Система команд MK51 по сравнению с MK48 допускает больше комбинаций способов адресации операндов в командах, что делает ее более гибкой и универсальной.

Флаги результата. Слово состояния программы (PSW) включает в себя четыре флага: C – перенос, AC – вспомогательный перенос, OV – переполнение и P – паритет.

Флаг паритета (отсутствует в MK48) напрямую зависит от текущего значения аккумулятора. Если число единичных бит аккумулятора нечетное, то флаг P устанавливается, а если четное – сбрасывается. Все попытки изменить флаг P, присваивая ему новое значение, будут безуспешными, если содержимое аккумулятора при этом останется неизменным. Флаг AC устанавливается в случае, если при выполнении опера-

ции сложения/вычитания между тетрадами байта возник перенос/засм. Флаг C устанавливается, если в старшем бите результата возникает перенос или засм. При выполнении операций умножения и деления флаг C сбрасывается. Флаг OV (отсутствует в MK48) 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.

Таблица 3.11. Команды, модифицирующие флаги результата

Команды	Флаги	Команды	Флаги
ADD	C,OV, AC	CLR C	C = 0
ADDC	C,OV, AC	CPI C	C = C
SUBB	C,OV, AC	ANL C,b	C
MUL	C = 0, OV	ANL C,/b	C
DIV	C = 0, OV	ORL C,b	C
DA	C	ORL C,/b	C
RRC	C	MOV C,b	C
RIC	C	CJNE	C
SETB C	C = I		

ции сложения/вычитания между тетрадами байта возник перенос/засм. Флаг C устанавливается, если в старшем бите результата возникает перенос или засм. При выполнении операций умножения и деления флаг C сбрасывается. Флаг OV (отсутствует в MK48) 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.

В табл. 3.11 перечисляются команды, при выполнении которых модифицируются флаги результата. В таблице отсутствует флаг паритета, так как его значение изменяется всеми командами, которые изменяют содержимое аккумулятора. Кроме команд, приведенных в таблице, флаги модифицируются командами, в которых местом назначения результата определены PSW или его отдельные биты, а также командами операций над битами.

Символическая адресация. При использовании ассемблера MK51 (ASM51) для получения объектных кодов программ допускается применение в программах символьических имен регистров специальных функций (PCF), портов и их отдельных бит (рис. 3.21).

Для адресации отдельных бит PCF и портов (такая возможность имеется не у всех PCF) можно использовать символическое имя бита следующей структуры:

⟨имя PCF или порта⟩ . ⟨номер бита⟩

Например, символическое имя пятого бита аккумулятора будет следующим: ACC.5. Символические имена PCF, портов и их бит являются зарезервированными словами для ASM51, и их не надо определять с помощью директив ассемблера.

3.8.2. Группа команд передачи данных

Большую часть команд данной группы (табл. 3.12) составляют команды передачи и обмена байтов. Команды пересылки бит представлена в группе команд битовых операций. Все команды данной группы

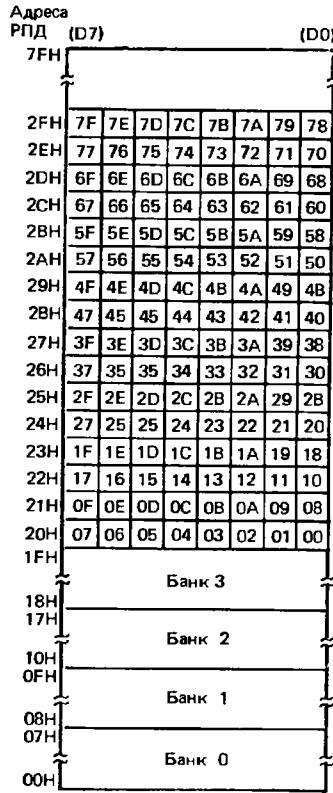


Рис. 3.20. Карта адресуемых бит в резидентной памяти данных

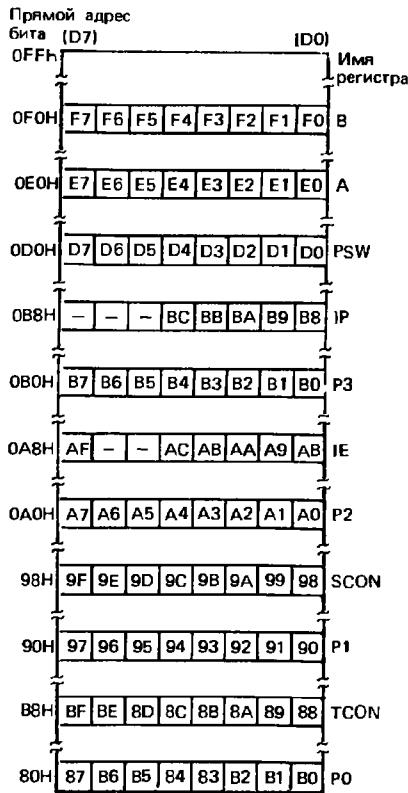


Рис. 3.21. Карта адресуемых бит в блоке регистров специальных функций

не модифицируют флаги результата, за исключением команд загрузки PSW и аккумулятора (флаг паритета).

Структура информационных связей. В зависимости от способа адресации и места расположения операнда можно выделить девять типов operandов, между которыми возможен информационный обмен. Граф возможных операций передачи данных показан на рис. 3.22. Аккумулятор (A) представлен на этом графе отдельной вершиной, так как многие команды используют неявную (подразумеваемую) адресацию.

В отличие от MK48 передачи данных в MK51 могут выполняться без участия аккумулятора.

Аккумулятор. В отличие от MK48 обращение к аккумулятору может быть выполнено в MK51 с использованием неявной и прямой адресации.

Таблица 3.12. Группа команд передачи данных

Название команды	Мнемоник	КОП	Т	Б	И	Операция
Пересылка в аккумулятор из регистра (i = 0 ÷ 7)	MOV A, Rn	11101rrr	1	1	1	(A) ← (Rn)
Пересылка в аккумулятор прямoadресуемого байта	MOV A, ad	11100101	3	2	1	(A) ← (ad)
Пересылка в аккумулятор байта из РПД (i = 0..1)	MOV A, @Ri	11100111	1	1	1	(A) ← ((Ri))
Загрузка в аккумулятор константы	MOV A, #d	01110100	2	2	1	(A) ← # d
Пересылка в регистр из аккумулятора	MOV Rn, A	11111rrr	1	1	1	(Rn) ← (A)
Пересылка в регистр прямoadресуемого байта	MOV Rn, ad	10101rrr	3	2	2	(Rn) ← (ad)
Загрузка в регистр константы	MOV Rn, #d	01111rrr	2	2	1	(Rn) ← # d
Пересылка по прямому адресу аккумулятора	MOV ad, A	11110101	3	2	1	(ad) ← (A)
Пересылка по прямому адресу регистра	MOV ad, Rn	10001rrr	3	2	2	(ad) ← (Rn)
Пересылка прямoadресуемого байта по прямому адресу	MOV add, ads	10000101	9	3	2	(add) ← (ads)
Пересылка байта из РПД по прямому адресу	MOV ad, @Ri	10000111	3	2	2	(ad) ← ((Ri))
Пересылка по прямому адресу константы	MOV ad, #d	01110101	7	3	2	(ad) ← # d
Пересылка в РПД из аккумулятора	MOV @Ri, A	11110111	1	1	1	((Ri)) ← (A)
Пересылка в РПД прямoadресуемого байта	MOV @Ri, ad	01100111	3	2	2	((Ri)) ← (ad)
Пересылка в РПД константы	MOV DPTR, # d16	10010000	13	3	2	(DPTR) ← # d16
Загрузка указателя данных	MOVC A, @A +	10010011	1	1	2	(A) ← ((A) +
Пересылка в аккумулятор байта из ПП	+ D PTR					+ (D PTR))
Пересылка в аккумулятор байта из ВПД	MOVC A, @A + PC	10000011	1	1	2	(PC) ← (PC) + 1
Пересылка в аккумулятор байта из расширенной ВПД						(A) ← ((A) + (PC))
Пересылка в ВПД из аккумулятора	MOVX A, @Ri	11100011	1	1	2	(A) ← ((Ri))
Пересылка в ВПД из аккумулятора	MOVX A, @DPTR	11100000	1	1	2	(A) ← ((DPTR))
Пересылка в расширенную ВПД из аккумулятора	MOVX @Ri, A	11110011	1	1	2	((Ri)) ← (A)
Пересылка в стек	MOVX @DPTR, A	11110000	3	2	2	((DPTR) ← (SP) + 1)
Извлечение из стека	PUSHI ad	11010000	3	2	2	(SP) ← (SP)
	POP ad	11010000	3	2	2	(SP) ← (SP) - 1

Продолжение табл. 3.12.

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Обмен аккумулятора с регистром	XCH A, Rn	110001rr	1	1	1	(A) \leftrightarrow (Rn)
Обмен аккумулятора с прямoadресуемым байтом	XCH A, ad	11000101	3	2	1	(A) \leftrightarrow (ad)
Обмен аккумулятора с байтом из РПД	XCH A, @Ri	00100111	1	1	1	(A) \leftrightarrow (@(Ri))
Обмен младшей четверды аккумулятора с младшней четвердой байта РПД	XCHD A, @Ri	11000111	1	1	1	(A ₀₋₃) \leftrightarrow ((Rd) ₀₋₃)

Таблица 3.13. Группа команд арифметических операций

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Сложение аккумулятора с регистром ($n = 0 \div 7$)	ADD A, Rn	00101rr	1	1	1	(A) \leftarrow (A) + (Rn)
	ADD A, ad	00100101	3	2	1	(A) \leftarrow (A) + (ad)
Сложение аккумулятора с прямoadресуемым байтом	ADD A, @Ri	00100111	1	1	1	(A) \leftarrow (A) + (@(Ri))
Сложение аккумулятора с байтом из РПД ($i = 0..1$)	ADD A,#d	00100100	2	2	1	(A) \leftarrow (A) + # d
Сложение аккумулятора с константой	ADDC A, Rn	00111rr	1	1	1	(A) \leftarrow (A) + (Rn) + (C)
Сложение аккумулятора с регистром и переносом	ADDC A, ad	00110101	3	2	1	(A) \leftarrow (A) + (ad) + (C)
Сложение аккумулятора с прямoadресуемым байтом и переносом	ADDC A, @Ri	00110111	1	1	1	(A) \leftarrow (A) + (@(Ri)) + (C)
Сложение аккумулятора с байтом из РПД и переносом	ADDC A,#d	00110100	2	2	1	(A) \leftarrow (A) + # d + (C)
Сложение аккумулятора с константой и переносом	DA A	11010100	1	1	1	Если (A ₀₋₃) > 9 V (AC) = 1, то (A ₀₋₃) \leftarrow (A ₀₋₃) + 6,
Десятичная коррекция аккумулятора						затем если (A ₄₋₇) > 9 V (C) = 1, то (A ₄₋₇) \leftarrow (A ₄₋₇) + 6

88

Вычитание из аккумулятора регистра и заемса байта из заема	SUBB A, Rn	10011rr	1	1	1	(A) \leftarrow (A) - (C) - (Rn)
Вычитание из аккумулятора байта РПД и заемса	SUBB A, ad	10010101	3	2	1	(A) \leftarrow (A) - (C) - ((ad))
Вычитание из аккумулятора константы и заемса	SUBB A, @Ri	10010111	1	1	1	(A) \leftarrow (A) - (@(Ri))
Инкремент аккумулятора	SUBB A, d	10010100	2	2	1	(A) \leftarrow (A) - # d
INC A	00000100	1	1	1	1	(A) \leftarrow (A) + 1
INC Rn	00001rr*	1	1	1	1	(Rn) \leftarrow (Rn) + 1
INC ad	00000101	3	2	1	1	(ad) \leftarrow (ad) + 1
INC @Ri	00000111	1	1	1	1	((Ri)) \leftarrow ((Ri)) + 1
INC DPTR	10100011	1	1	2	1	(DPTR) \leftarrow (DPTR) + 1
DEC A	00010100	1	1	1	1	(A) \leftarrow (A) - 1
D: C Rn	00011rr	1	1	1	1	(Rn) \leftarrow (Rn) - 1
DEC ad	00010101	3	2	1	1	(ad) \leftarrow (ad) - 1
DEC @Ri	00010111	1	1	1	1	((Ri)) \leftarrow ((Ri)) - 1
Инкремент узкоданных	MUL AB	10100100	1	1	4	(B)(A) \leftarrow (A) \times (B)
Декремент аккумулятора на регистр В	DIV AB	10000100	1	1	4	(A)(B) \leftarrow (A)/(B)
Деление аккумулятора на регистр В						

Таблица 3.14. Группа команд логических операций

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Логическое И аккумулятора и регистра	ANL A, Rn	01011rr	1	1	1	(A) \leftarrow (A) \wedge (Rn)
Логическое И аккумулятора и прямoadресуемого байта	ANL A, ad	01010101	3	2	1	(A) \leftarrow (A) \wedge (ad)
Логическое И аккумулятора и байта из РПД	ANL A, @Ri	01010111	1	1	1	(A) \leftarrow (A) \wedge (@(Ri))
Логическое И аккумулятора и константы	ANL A, # d	01010100	2	2	1	(A) \leftarrow (A) \wedge # d
Логическое И прямoadресуемого байта и аккумулятора	ANL ad, A	01010010	3	2	1	(ad) \leftarrow (ad) \wedge (A)
Логическое И прямoadресуемого байта и константы	ANL ad, # d	01010011	7	3	2	(ad) \leftarrow (ad) \wedge # d
Логическое ИЛИ аккумулятора и регистра	ORL A, Rn	01001rr	1	1	1	(A) \leftarrow (A) \vee (Rn)
Логическое ИЛИ аккумулятора и прямoadресуемого байта	ORL A, ad	01000101	3	2	1	(A) \leftarrow (A) \vee (ad)

89

Название команды	Мнемокод	КОЛ	ТБЦ	Операция
Логическое ИЛИ аккумулятора и байта из РПД	ORL A, @Ri	01000111	1	(A) \leftarrow (A) V ((Ri))
Логическое ИЛИ аккумулятора и константы	ORL A, #d	01000100	2	(A) \leftarrow (A) V #d
Логическое ИЛИ прямодресуемого байта и аккумулятора	ORL ad, A	01000010	3	(ad) \leftarrow (ad) V (A)
Логическое ИЛИ прямодресуемого байта и константы	ORL ad, #d	01000011	7	(ad) \leftarrow (ad) V #d
Исключающее ИЛИ аккумулятора и регистра	XRL A, Rn	01101111	1	(A) \leftarrow (A) V (Rn)
Исключающее ИЛИ аккумулятора и прямодресуемого байта	XRL A, ad	01100101	3	(A) \leftarrow (A) V (ad)
Исключающее ИЛИ аккумулятора и константы	XRL A, #d	01100111	1	(A) \leftarrow (A) V ((Ri))
Исключающее ИЛИ прямодресуемого байта и аккумулятора	XRL ad, A	01100100	2	(A) \leftarrow (A) V #d
Исключающее ИЛИ прямодресуемого байта и константы	XRL ad, #d	01100010	3	(ad) \leftarrow (ad) V (A)
Сброс аккумулятора	CLR A	01100011	7	(ad) \leftarrow (ad) V #d
Инверсия аккумулятора	CPL A	11100100	1	(A) \leftarrow 0
Сдвиг аккумулятора влево циклический	RL A	11110100	1	(A) \leftarrow (A _n)
		00100011	1	(A _{n+1}) \leftarrow (A _n), n = 0 ÷ 6,
Сдвиг аккумулятора влево через перенос	RLC A	00110011	1	(A ₀) \leftarrow (A ₇), (A _{n+1}) \leftarrow (A _n), n = 0 ÷ 6,
Сдвиг аккумулятора вправо циклический	RR A	00000011	1	(A ₀) \leftarrow (C), (C) \leftarrow (A ₇)
Сдвиг аккумулятора вправо через перенос	RRC A	00010011	1	(A _n) \leftarrow (A _{n+1}), n = 0 ÷ 6, (A ₇) \leftarrow (A ₀)
Обмен местами тетрад в аккумуляторе	SWAP A	11000100	1	(A ₀₋₃) \leftarrow (A ₄₋₇)

В зависимости от способа адресации аккумулятора применяется одно из символьических имен: A или ACC (прямой адрес). При прямой адресации обращение к аккумулятору производится как к одному из РСФ, и его адрес указывается во втором байте команды.

Использование неявной адресации аккумулятора предпочтительнее, однако не всегда возможно, например при обращении к отдельным битам аккумулятора.

Обращение к внешней памяти данных. Режим косвенной адресации ВПД, имеющийся в MK48, реализован также и в MK51. При использовании команд MOVX @Ri обеспечивается доступ к 256 байтам внешней памяти данных.

Существует также режим обращения к расширенной ВПД, когда для доступа используется 16-битный адрес, хранящийся в регистре-указателе данных (DPTR). Команды MOVX @DPTR обеспечивают доступ к 65 536 байтам ВПД.

3.8.3. Группа команд арифметических операций

Данную группу образуют 24 команды (табл. 3.13), выполняющие операции сложения, десятичной коррекции, инкремента/декремента байтов. Дополнительно по сравнению с MK48 введены команды вычитания, умножения и деления байтов.

Команды ADD и ADC аналогичны командам сложения MK48, но допускают сложение аккумулятора с большим числом operandов. Аналогично командам ADC существуют четыре команды SUBB, что позволяет более просто, чем в MK48, производить вычитание байтов и многобайтных двоичных чисел. В MK51 реализуется расширенный (по сравнению с MK48) список команд инкремента/декремента байтов, введена команда инкремента 16-битного регистра-указателя данных.

3.8.4. Группа команд логических операций

Данную группу образуют 25 команд (табл. 3.14), реализующих те же логические операции над байтами, что и в MK48. Однако в MK51 значительно расширено число типов operandов, участвующих в операциях.

В отличие от MK48 имеется возможность производить операцию "исключающее ИЛИ" с содержимым портов. Команда XRL может быть эффективно использована для инверсии отдельных битов портов.

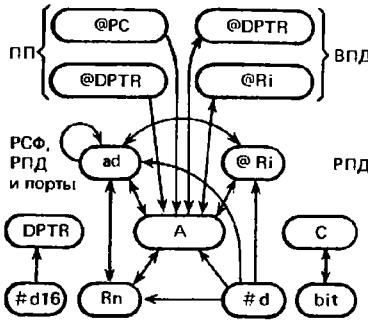


Рис. 3.22. Граф путей передачи данных в MK51

3.8.5. Группа команд операций с битами

Отличительной особенностью данной группы команд (табл. 3.15) является то, что они оперируют с однобитными операндами. В качестве таких операндов могут выступать отдельные биты некоторых регистров специальных функций (РСФ) и портов, а также 128 программных флагов пользователя.

Существуют команды сброса (CLR), установки (SETB) и инверсии (CPL) бит, а также конъюнкции и дизъюнкции бита и флага переноса. Для адресации бит используется прямой восьмиразрядный адрес (bit). Косвенная адресация бит невозможна.

3.8.6. Группа команд передачи управления

К данной группе команд (табл. 3.16) относятся команды, обеспечивающие условное и безусловное ветвление, вызов подпрограмм и возврат из них, а также команда пустой операции NOP. В большинстве команд используется прямая адресация, т.е. адрес перехода целиком (или его часть) содержится в самой команде передачи управления. Можно выделить три разновидности команд ветвления по разрядности указываемого адреса перехода.

Длинный переход. Переход по всему адресному пространству ПП. В команде содержится полный 16-битный адрес перехода (ad 16). Трехбайтные команды длинного перехода содержат в мнемокоде букву L (Long). Всего существует две такие команды: LJMP – длинный переход и LCALL – длинный вызов подпрограммы. На практике редко возникает необходимость перехода в пределах всего адресного пространства и чаще используются укороченные команды перехода, занимающие меньше места в памяти.

Абсолютный переход. Переход в пределах одной страницы памяти программ размером 2048 байт. Такие команды содержат только 11 младших бит адреса перехода (ad11). Команды абсолютного перехода имеют формат 2 байта. Начальная буква мнемокода – A (Absolute). При выполнении команды в вычисленном адресе следующей по порядку команды ((PC) = (PC) + 2) 11 младших бит заменяются на ad11 из тела команды абсолютного перехода.

Относительный переход. Короткий относительный переход позволяет передать управление в пределах – 128 ÷ +127 байт относительно адреса следующей команды (команда, следующей по порядку за командой относительного перехода). Существует одна команда безусловного короткого перехода SJMP (Short). Все команды условного перехода используют данный метод адресации. Относительный адрес перехода (rel) содержится во втором байте команды.

Косвенный переход. Команда JMP @A + DPTR позволяет передавать управление по косвенному адресу. Эта команда удобна тем, что предоставляет возможность организации перехода по адресу, вычисляемому самой программой и неизвестному при написании исходного текста программы.

Таблица 3.15. Группа команд операций с битами

Название команды	Мнемокод	КОП	Т	Б	Ц	Операции
Сброс переноса	CLR C	11000011	1	1	1	(C) ← 0
Сброс бита	CLR bit	11000010	4	2	1	(b) ← 0
Установка переноса	SETB C	11010011	1	1	1	(C) ← 1
Установка бита	SETB bit	11010010	4	2	1	(b) ← 1
Инверсия бита и переноса	CPL C	10110011	1	1	1	(C) ← (C̄)
Инверсия бита	CPL bit	10110010	4	2	1	(b) ← (b̄)
Логическое И бита и переноса	ANL C,bit	10000010	4	2	2	(C) ← (C) ∧ (b)
Логическое ИЛИ бита и переноса	ANL C,bit	10110000	4	2	2	(C) ← (C) ∧ (b̄)
Логическое ИЛИ инверсии бита и переноса	ORL C,bit	01110010	4	2	2	(C) ← (C) ∨ (b)
Переслалка бита в перенос	MOV C,bit	10100000	4	2	1	(C) ← (b)
Переслалка переноса в бит	MOV bit,C	10010010	4	2	2	(b) ← (C)

Таблица 3.16. Группа команд передачи управления

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Линейный переход в полном объеме памяти программ	LJMP ad 16	00000010	12	3	2	(PC) \leftarrow ad 16
Абсолютный переход внутри страницы в 2 Кбайта	AJMP ad 11	a10493800001	6	2	2	(PC) \leftarrow (PC) + 2 (PC ₀₋₁₀) \leftarrow ad 11
Короткий относительный переход внутри страницы в 256 байт	SIMP rel	10000000	5	2	2	(PC) \leftarrow (PC) + 2 (PC) \leftarrow (PC) + rel
Косвенный относительный переход, если аккумулятор равен нулю	JMP @A + DPTR rel	01110011 01100000	1 5	1 2	2	(PC) \leftarrow (A) + (DPTR) (PC) \leftarrow (PC) + 2, если (A) = 0, то (PC) \leftarrow (PC) + rel
Переход, если аккумулятор не равен нулю	JZ rel	01110000	5	2	2	(PC) \leftarrow (PC) + 2, если (A) \neq 0, то (PC) \leftarrow (PC) + rel
Переход, если переход равен единице	JC rel	01000000	5	2	2	(PC) \leftarrow (PC) + 2, если (C) = 1, то (PC) \leftarrow (PC) + rel
Переход, если переход равен нулю	JNC rel	01010000	5	2	2	(PC) \leftarrow (PC) + 2, если (C) = 0, то (PC) \leftarrow (PC) + rel
Переход, если бит равен единице	JB bit, rel	00100000	11	3	2	(PC) \leftarrow (PC) + 3, если (b) = 1, то (PC) \leftarrow (PC) + rel
Переход, если бит равен нулю	JNB bit, rel	00110000	11	3	2	(PC) \leftarrow (PC) + 3, если (b) = 0, то (PC) \leftarrow (PC) + rel
Переход, если бит установлен, с последующим сбросом бита	JBC bit, rel	00010000	11	3	2	(PC) \leftarrow (PC) + 3, если (b) = 1, то (b) \leftarrow 0 и (PC) \leftarrow (PC) + rel
Декремент регистра и переход, если не нуль	DJNZ Rn, rel	11011111	5	2	2	(PC) \leftarrow (PC) + 2, (Rn) \leftarrow (Rn) - 1, если (Rn) \neq 0, то (PC) \leftarrow (PC) + rel
Декремент при модифицируемого – байта и переход, если не пуль	DJNZ ad, rel	11010101	8	3	2	(PC) \leftarrow (PC) + 2, (ad) \leftarrow (ad) - 1, если (ad) \neq 0, то (PC) \leftarrow (PC) + rel
Сравнение аккумулятора с прямодействуемым байтом и переход, если равно	CJNE A, ad, rel	10110101	8	3	2	(PC) \leftarrow (PC) + 3, если (A) \neq (ad), то (PC) \leftarrow (PC) + rel, если (A) \leq (ad), то (C) \leftarrow 1, иначе (C) \leftarrow 0

Продолжение табл. 3.16

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Сравнение аккумулятора с константой и переход, если не равно	CJNE A, # d, rel	10110100	10	3	2	(PC) \leftarrow (PC) + 3, если (A) \neq d, то (PC) \leftarrow (PC) + rel, если ((R)) $<$ # d, то (C) \leftarrow 1, иначе (C) \leftarrow 0
Сравнение регистра с константой и переход, если не равно	CJNE Rn, # d, rel	10111111	10	3	2	(PC) \leftarrow (PC) + 3, если (Rn) \neq d, то (PC) \leftarrow (PC) + rel
Линейный вызов подпрограммы	LCALL ad 16	00010010	12	3	2	иначе (C) \leftarrow 0 (PC) \leftarrow (PC) + 3, если ((R)) $<$ 0, то (PC) \leftarrow (PC) + 1, ((SP)) \leftarrow (PC ₀₋₇), (SP) \leftarrow (SP) + 1, ((SP)) \leftarrow (PC ₈₋₁₅), (SP) \leftarrow (SP) - 1, (PC) \leftarrow (PC ₈₋₁₅), (PC) \leftarrow (SP) - 1, (SP) \leftarrow (SP) + 1, иначе (C) \leftarrow 0
Абсолютный вызов подпрограммы в пределах страницы в 2 Кбайта	ACALL ad 11	a10493810001	6	2	2	((SP)) \leftarrow (PC ₀₋₇), (SP) \leftarrow (SP) + 1, ((SP)) \leftarrow (PC ₈₋₁₅), (PC ₀₋₁₀) \leftarrow ad 11
Возврат из подпрограммы	RETI	00100010	1	1	2	(PC ₈₋₁₅) \leftarrow ((SP)), (SP) \leftarrow (SP) - 1, (PC ₀₋₇) \leftarrow ((SP)), (SP) \leftarrow (SP) - 1,
Возврат из подпрограммы обработки прерывания	RETI	00110010	1	1	2	(PC ₈₋₁₅) \leftarrow ((SP)), (SP) \leftarrow (SP) - 1, (PC ₀₋₇) \leftarrow ((SP)), (SP) \leftarrow (SP) - 1
Холостая команда	NOP	00000000	1	1	1	(PC) \leftarrow (PC) + 1

Примечание. Ассемблер допускает использование обобщенного имени команд LJMP и CALL, которые в процессе трансляции заменяются оптимальными по формату командами перехода (AJMP, SJMP, LJMP) или вызова (ACALL, LCALL).

Условные переходы. Развитая система условных переходов предоставляет возможность осуществлять ветвление по следующим условиям: аккумулятор содержит нуль (JZ); содержимое аккумулятора не равно нулю (JNZ); перенос равен единице (JC); перенос равен нулю (JNC); адресуемый бит равен единице (JB); адресуемый бит равен нулю (JNB).

Для организации программных циклов удобно пользоваться командой DJNZ, которая работает аналогично соответствующей команде MK48. Однако в качестве счетчика циклов в MK51 может использоваться не только регистр, но и прямоадресуемый байт (например, ячейка РПД).

Команда CJNE эффективно используется в процедурах ожидания какого-либо события. Например, команда

```
WAIT: CJNE A, #0, WAIT
```

будет выполняться до тех пор, пока на линиях порта 0 не установится информация, совпадающая с содержимым аккумулятора.

Все команды данной группы, за исключением CJNE и JBC, не оказывают воздействия на флаги. Команда CJNE устанавливает флаг C, если первый операнд оказывается меньше второго. Команда JBC сбрасывает флаг C в случае перехода.

Подпрограммы. Для обращения к подпрограммам необходимо использовать команды вызова подпрограмм (LCALL, ACALL). Эти команды в отличие от команд перехода (LJMP, AJMP) сохраняют в стеке адрес возврата в основную программу. Для возврата из подпрограммы необходимо выполнить команду RET. Команда RETI отличается от команды RET тем, что разрешает прерывания обслуженного уровня.

Методика разработки прикладного программного обеспечения МК-систем

4.1.

Формализованный подход к разработке прикладных программ

Если целевая функция контроллера сформулирована, т.е. задача на разработку поставлена, то для получения текста исходной программы необходимо выполнить ряд последовательных действий:

- 1) подробное описание задачи;
- 2) анализ задачи;
- 3) инженерную интерпретацию задачи, желательно с привлечением того или иного аппарата формализации (граф автомата, сети Петри, матрицы состояний и связности и т.п.);
- 4) разработку общей блок-схемы алгоритма (БСА) работы контроллера;
- 5) разработку детализированных БСА отдельных процедур, выделенных на основе модульного принципа составления программ;
- 6) детальную проработку интерфейса контроллера и внесение исправлений в общую и детализированные БСА;
- 7) распределение рабочих регистров и памяти МК;
- 8) формирование текста исходной программы.

В результате работы по трем первым пунктам данного перечня получают так называемую функциональную спецификацию прикладной программы МК, в которой основное внимание уделяется детализации способов формирования входной и выходной информации.

На языке схем алгоритмов разработчик описывает метод, выбранный для решения поставленной задачи. Довольно часто бывает, что одна и та же задача может быть решена различными методами. Способ решения задачи, выбранный на этапе ее инженерной интерпретации, на основе которого формируется БСА, определяет не только качество разрабатываемой прикладной программы, но и качественные показатели конечного изделия.

Разработка БСА очень похожа на разработку аппаратуры средств систем автоматики и обработки данных. В основу разработки БСА положена та же самая процедура модульного проектирования, которая

традиционно используется разработчиками аппаратурных средств. Отличие состоит в том, что при разработке аппаратурных средств в качестве "строительного" материала используются логические схемы, триггеры, регистры и другие интегральные элементы, а при создании программного обеспечения разработчик оперирует командами, подпрограммами, таблицами и другими программными объектами из арсенала средств обработки данных.

Так как алгоритм есть точно определенная процедура, предписывающая контроллеру однозначно определенные действия по преобразованию "сырых" исходных данных в обработанные выходные данные, то разработка БСА требует предельной точности и однозначности используемой атрибутику: символьических имен переменных, констант (установок), подпрограмм (модулей), символьических адресов таблиц, портов ввода/вывода и т.п. Основное внимание при разработке БСА следует уделить тому разделу функциональной спецификации прикладной программы, в котором приводится описание аппаратуры сопряжения МК с объектом управления. (Это описание для успешной разработки программного обеспечения должно быть детализировано вплоть до электрических и временных характеристик каждого входного и выходного сигнала или устройства.)

Секрет успеха разработки прикладной программы МК заключается в использовании метода декомпозиции, при котором вся задача последовательно разделяется на меньшие функциональные модули, каждый из которых можно анализировать, разрабатывать и отлаживать отдельно от других. При выполнении прикладной программы в МК управление без всяких двусмысленностей передается от одного функционального модуля к другому. Схема связности этих функциональных модулей, каждый из которых реализует некоторую процедуру, образует общую (или системную) БСА прикладной программы. Это разделение задачи на модули и субмодули выполняется последовательно до такого уровня, когда разработка БСА модуля становится простым и понятным делом. Метод последовательной декомпозиции обладает достаточной гибкостью, что позволяет привести степень детализации БСА в соответствие со сложностью процедуры. Не следует стесняться при выполнении декомпозиции дойти до модулей, которые почти тривиальны. Ведь именно эту цель (получение очень простого и "прозрачного" алгоритма модуля) преследует разработчик, когда он стремится заставить МК надежно выполнять требуемую работу по управлению объектом. Язык графических образов БСА можно использовать на любом уровне детализации описания модулей вплоть до того, что каждому оператору БСА будет соответствовать единственная команда МК.

Структурное программирование есть процесс построения прикладной программы из строгого набора программных модулей, каждый из которых реализует определенную процедуру обработки данных. *Программные модули должны иметь только одну точку входа и одну точку выхода.* Только в этом случае отдельные модули можно разрабатывать и отлаживать независимо, а затем объединять в законченную приклад-

ную программу с минимальными проблемами их взаимосвязей. Источником подавляющего большинства ошибок программирования является использование модулей, имеющих один вход и несколько выходов. При необходимости организации множественных ветвлений в программе декомпозицию задачи выполняют таким образом, чтобы каждый функциональный модуль имел только один вход и один выход. Для этого условные операторы (имеющие два выхода) или включают внутрь модуля, объединяя их с операторами обработки, или выносят в систему межмодульных связей, формируя тем самым БСА более высокого ранга.

В международном стандарте на программный продукт NIPO (Hierarchical Input-Process-Output) ("хай-по") декларируется аналогичный подход к разработке прикладных программ.

Разработка БСА функционального модуля программы имеет ярко выраженный итеративный характер, т.е. требует многократных проб, прежде чем возникает уверенность, что алгоритм реализации процедуры правильный и завершенный. Вне зависимости от функционального назначения процедуры при разработке ее БСА необходимо придерживаться следующей очередности работы:

1. *Определить, что должен делать модуль* (это уже было сделано при разработке системной БСА, но теперь разработчик имеет дело с фрагментом прикладной программы, а не с целой программой, и, следовательно, может потребоваться доопределение и уточнение целевого назначения процедуры).

2. *Определить способы получения модулем исходных данных* (от датчиков через порты ввода, или из таблиц в памяти, или через рабочие регистры). Для реализации ввода исходных данных в модуль в его БСА надо включить соответствующие операторы.

3. *Определить необходимость какой-либо предварительной обработки введенных исходных данных* (маскирование, сдвиг, масштабирование, перекодировка). Если до использования "сырых" данных требуется их предобработка, то в состав БСА включаются соответствующие операторы.

4. *Определить способ преобразования входных данных в требуемые выходные.* Используя операторы процедур и условные операторы принятия решения, отобразить на языке БСА выбранный метод содержательной обработки исходных данных.

5. *Определить способы выдачи из модуля обработанных данных* (передать в память, или в вызываемую программу, или в порты вывода информации). Необходимые действия отобразить в БСА.

6. *Определить необходимость какой-либо постобработки выводимых данных* (изменение формата, перекодирование, масштабирование, маскирование). Ввести в БСА операторы подготовки данных для вывода из модуля.

7. Вернуться к п. 1 настоящего перечня работ и проанализировать полученный результат. Выполнить итеративную корректировку БСА с целью сделать ее простой, логичной, стройной и обладающей четким графическим образом.

8. Проверить работоспособность алгоритма на бумаге путем подстановки в него действительных данных. Убедиться в его сходимости и результативности.

9. Рассмотреть предельные случаи и попытаться определить граничные значения информационных объектов, с которыми оперирует алгоритм, за пределами которых он теряет свойства конечности, сходимости или результативности. (Особое внимание при этом следует уделить анализу возможных ситуаций переполнения разрядной сетки МК, изменения знака результата операции, деления на переменную, которая может принять иульевое значение.)

10. Провести мысленный эксперимент по определению работоспособности алгоритма в реальном масштабе времени, когда стохастические события, происходящие в объекте управления, могут оказывать влияние на работу алгоритма. При этом самому тщательному анализу следует подвергнуть реакцию алгоритма на возможные прерывания с целью определения критических операторов, которые необходимо защитить от прерываний. Кроме того, в ходе этого мысленного эксперимента следует проанализировать логику алгоритма с целью определения таких последовательностей операторов, при выполнении которых МК может "не заметить" кратковременных событий в объекте управления. При обнаружении таких ситуаций в логику БСА следует внести корректиды.

Практика разработки программного обеспечения для МК показала, что *последовательное использование описанной поэтапной процедуры проектирования алгоритмов, составляющей основу метода структурного программирования, позволяет уверенно получать работоспособные прикладные программы*. Дисциплинированное следование этой поэтапной процедуре проектирования прикладных программ обеспечивает успех проекта! В противном случае "Вы рискуете заболеть страшным программным заболеванием, которое называется "вползающие особенности". Эта инфекция возникает, когда неадекватная спецификация задачи позволяет вползать в программу организмам, называемым "изящные особенности". Те изменения, которые легко учесть на этапе планирования, могут потребовать огромных усилий на этапе реализации программы. Болезнь эта к моменту обнаружения становится уже серьезной и привела к фатальному концу много программных проектов [5]. Чаще всего носителями этой болезни являются профессиональные программисты, которые способны заразить ею программирующих профессионалов. Если Вы стали жертвой "вползающих особенностей", то должны или начать заново разрабатывать функциональную спецификацию программного обеспечения, или быть готовыми к исключительно высоким трудозатратам на этапе отладки прикладной программы.

Преобразование разработанной БСА в исходный текст программы – дело несложное. Но прежде чем приступить к написанию программы, необходимо специфицировать память и выбрать язык программирования.

Спецификация памяти (и рабочих регистров) заключается в определении адреса первой команды прикладной программы, действительных

начальных адресов стека, таблиц данных, буферных зон передачи параметров между процедурами, подпрограмм обслуживания прерываний и т.д. При этом следует помнить, что в МК память программ и память данных физически и логически разделены.

Диапазон языков написания исходного текста прикладной программы простирается от машинного кода до почти естественного языка. В машинном коде или на языке ассемблера программировать труднее и дольше, чем на алгоритмическом языке высокого уровня, но зато получается более короткий код программы, требуется меньшая емкость памяти программы и выполняется такая программа быстрее. Объектные коды, полученные путем трансляции исходных программ, написанных на алгоритмическом языке высокого уровня, занимают в памяти МК системы много больше места и требуют большего времени на исполнение. Выбор языковых средств составления исходных программ в каждом конкретном применении зависит от характеристик прикладной задачи, опыта программиста и допустимых затрат на разработку.

По нашему мнению, огромное большинство прикладных задач управления объектами вследствие того, что они должны решаться в реальном времени, предъявляет столь высокие требования по быстродействию, что для их решения основным языковым средством написания прикладных программ еще долгие годы будет оставаться язык ассемблера. Это положение о преимущественном использовании языка ассемблера подкрепляется и тем обстоятельством, что однокристальные МК имеют ограниченный объем РПП и, следовательно, критичны к длине прикладных программ.

4.2.

Элементы формализации в разработке алгоритмов

Процесс творческого освоения технических возможностей МК может быть облегчен тем, что на начальном этапе проектирования МК-систем можно с успехом использовать некоторые формализованные приемы синтеза комбинационных схем и цифровых автоматов на интегральных элементах.

К настоящему времени формализованные методы проектирования комбинационных и последовательностных (автоматных) схем на интегральных элементах низкой степени интеграции получили широкое теоретическое развитие и практическое применение [3]. Справедливости ради необходимо отметить, что в связи с ростом степени интеграции элементов, с приобретением ими свойств полифункциональности и программной перенастройки возможная область применения формализованных методов синтеза схем постоянно сужается. Эти формализованные методы не могут быть непосредственно применены для проектирования систем на основе МК, так как сам МК является программно-управляемым автоматом с жесткой структурой и однозначно кодированной системой состояний и команд. Однако некоторые классические приемы проектирования "схемной логики" могут быть применены для разра-

ботки МК-систем. К таким приемам относятся прежде всего методы минимизации булевых функций и автоматных таблиц переходов/выходов, методы описания работы последовательностных устройств на языке графов автоматов Мили и/или Мура.

Булевые функции. Законы функционирования отдельных устройств системы в процессе их разработки часто описывают на языке булевых функций или на языке релейно-контактных схем. Такие устройства реализуются на основе схем комбинационной логики (чаще всего с использованием интегральных схем серии ТТЛ) или на основе релейной логики. Все эти устройства могут быть с успехом реализованы программными средствами в МК. При этом следует иметь в виду, что программная реализация булевых функций в отличие от их реализации на комбинационных схемах, как правило, не требует дополнительной аппаратуры, но сопряжена с потерей быстродействия.

В качестве примера рассмотрим комбинационную схему, показанную на рис. 4.1 и реализующую булеву функцию от шести переменных

$$F = (A \wedge (B \vee C)) \vee (E \wedge \bar{H}) \vee \bar{L}$$

Предположим, что группа датчиков объекта управления генерирует для МК пять независимых двоичных сигналов: A и B от двух концевых переключателей; C, E и H – три бита от 3-позиционного переключателя, установленного на пульте управления оператора. Пусть шестая переменная L – это флаг некоторого программного признака, например признака непрохождения теста. Выходной сигнал F через порт подается на реле исполнительного механизма.

Сигналы от указанных датчиков подаются в строго определенные биты некоторого порта ввода информации. Следовательно, в процессе программной реализации заданной булевой функции необходимо после операции ввода обеспечить серию проверок состояния отдельных бит введенного байта и формирование значения выходного сигнала F.

На рис. 4.2 показана схема алгоритма программной реализации заданной булевой функции, учитывающая последовательность проверок, которая определяется скобочной формой записи функции F. Заметим, что при программной реализации булевых функций практически нет никаких ограничений на число переменных. Однако с ростом числа переменных возрастают времена задержки формирования выходного сигнала F.

Граф автомата. Законы функционирования последовательностных схем задаются или таблицами переходов/выходов или в виде графа (диаграммы состояний) автомата. Переход от графа проектируемого МК-устройства к БСА является ключевым этапом методики, позволяющим отобразить таблицы состояний и входов/выходов автомата на систему команд используемого МК.

Формализованные методы синтеза последовательностных схем описаны в [3] и здесь рассматриваться не будут. Примеры программной реализации автоматов, приводимые в гл. 8, дают детальное представление об использовании некоторых классических приемов синтеза "схемной логики" применительно к МК.

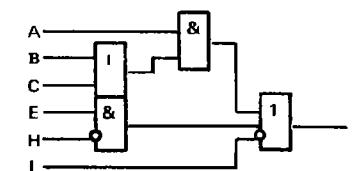
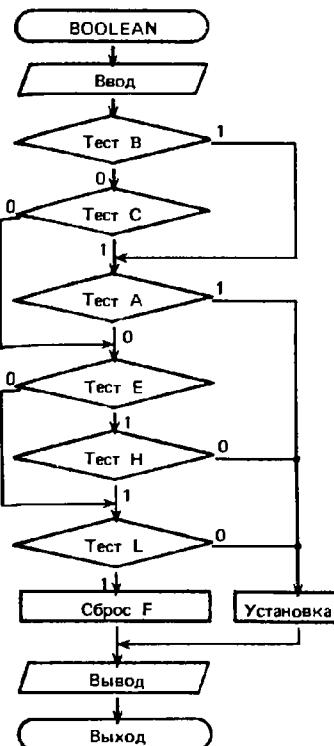


Рис. 4.1. Схемная реализация булевой функции

Рис. 4.2. Программная реализация булевой функции

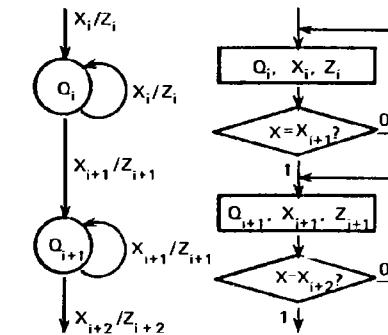


Рис. 4.3. Граф автомата и эквивалентная ему БСА

Сравнив фрагменты графа автомата Мили (автомат Мура здесь не рассматривается, так как эти две модели автомата однозначно связанны) и фрагмент БСА, показанные на рис. 4.3, можно заметить, что они эквивалентны.

Здесь X и Z – двоичные векторы, принадлежащие множествам наборов входных {x} и выходных {z} сигналов, а Q – состояния автомата из множества возможных состояний {q}. Состояние автомата Q_i и выходной вектор Z_i являются функциями предыдущего автомата состояния Q_{i-1} и текущего входного вектора X_i . Операторные прямоугольники в БСА могут быть интерпретированы как вершины (устойчивые состояния) графа автомата, а условные операторы – как дуги графа X_i/Z_i , т.е. переходы к иному автоматному состоянию. При этом операторный прямоугольник представляет собой фрагмент программы. В течение всего периода устойчивого состояния Q_i выходной вектор Z_i остается неизменным до тех пор, пока не изменит своего значения входной вектор X_i . Для этого в течение устойчивого состояния Q_i условные

операторы алгоритма программной реализации автомата селектируют (с заданным периодом дискретизации) изменение входного вектора X_i , если оно произошло под воздействием внешних (по отношению к МК) событий. В результате установления факта изменения вектора X_i контроллер переходит к новому "устойчивому" автоматному состоянию Q_{i+1} и генерирует новый выходной вектор Z_{i+1} , значения которого полностью определены графиком автомата.

Так как любой оператор БСА может быть представлен последовательностью команд МК, то дальнейшие действия по разработке прикладных программ для МК-системы представляют собой трансляцию (отображение алгоритма на систему команд МК), в результате которой формируется исходный текст прикладной программы.

4.3.

Процедуры и подпрограммы

При разработке МК-систем могут быть использованы два способа организации прикладных программ: монолитный и модульный. При первом способе вся прикладная программа МК разрабатывается как единое целое, а при втором строится из отдельных программных блоков, каждый из которых реализует некоторую процедуру обработки данных или управления. Взаимосвязь блоков определяется разработчиком при монтаже из этих блоков БСА и заключенной прикладной программы.

Отдельные фрагменты прикладной программы МК могут быть получены в виде линейной последовательности блоков, другие (многократно используемые) обычно оформляются в виде подпрограмм, к которым прикладная программа, называемая основной, имеет возможность обращаться по мере необходимости. Подпрограмма должна обладать следующими свойствами: выполнять законченную процедуру обработки данных, иметь только один вход и один выход и не обладать эффектом последействия, при котором текущее выполнение подпрограммы оказывало бы влияние на ее последующие выполнения.

Вызов подпрограммы. Обращение к подпрограмме осуществляется по команде вызова CALL MARK, где MARK – символьическое имя процедуры. Имя процедуры используется в качестве метки, отмечающей одну из команд (чаще всего первую) подпрограммы. Для MK51 mnemonic значение CALL является обобщенным и транслируется в одну из команд ACALL или LCALL в зависимости от адресного расстояния вызываемой подпрограммы.

По команде CALL в стеке сохраняется значение счетчика команд, и возврат из подпрограммы осуществляется в то место основной программы, откуда был осуществлен вызов (к команде основной программы, следующей за командой CALL). Для этого любая подпрограмма должна заканчиваться командой возврата RET, осуществляющей восстановление содержимого счетчика команд из стека.

Достаточно часто возникает необходимость такой организации вычислительного процесса, при которой подпрограмма вызывает другую под-

программу, та в свою очередь вызывает следующую и т.д. Этот процесс называется *вложением подпрограмм*. Число подпрограмм, которые могут быть вызваны таким образом (глубина вложности подпрограмм), ограничивается только емкостью стека.

Сохранение параметров основной программы. Иногда при обращении к подпрограмме возникает необходимость сохранить не только адрес возврата в основную программу, но и содержимое отдельных рабочих регистров. Удобным способом для этого является переключение банка регистров. Например, если основная программа использует банк регистров 0, то подпрограмма может использовать банк регистров 1. Однако переключение банка регистров не обеспечивает сохранение содержимого аккумулятора, что приводит к необходимости создавать в одном из рабочих регистров или в РПД "копию" аккумулятора.

Параметризуемые подпрограммы. Для успешной работы любой подпрограммы необходимо однозначно определить способ передачи в нее исходных данных и способ вывода результата ее работы. Подпрограмма, которой требуется дополнительная информация в виде параметров ее настройки или операндов, называется параметризуемой. Примером параметризуемой подпрограммы может служить подпрограмма временной задержки, если основной программе требуется реализация временных задержек различной длительности. Основная программа при этом должна обеспечить передачу в подпрограмму уставок, обеспечивающих требуемое время задержки.

Получили распространение три способа передачи параметров: через память, через регистры общего назначения и через регистр признаков. При передаче входных параметров через память основная программа обязательно содержит команды загрузки некоторых ячеек памяти, а подпрограмма – команды считывания из этих ячеек. При передаче выходных параметров подпрограмма должна загрузить некоторые ячейки памяти, а основная программа – считать. Передача параметров через регистры осуществляется аналогичным образом. Третий способ передачи параметров – через регистр признаков – удобно использовать при передаче выходных параметров (например, в подпрограммах сравнения чисел). В этом случае подпрограмма должна установить (или сбросить) соответствующие признаки, а основная программа – проанализировать их значение. В MK48 для этой цели целесообразно использовать флаги, сохраняемые в PSW (C и F0). Гораздо большими возможностями для передачи параметров через признаки обладает MK51, в котором имеется 128 флагов пользователя, доступных для модификации и анализа. Кроме перечисленных способов передачи параметров (общих для MK48 и MK51) в MK51 имеется еще возможность передачи параметров через стек. Этот способ, в частности, позволяет использовать в качестве параметра содержимое счетчика команд.

Использование процедур, оформленных в виде подпрограмм, при разработке программного обеспечения имеет ряд достоинств. Прежде всего относительно простые модули, выделенные из сложной программы, могут программироваться несколькими разработчиками с целью сокра-

шения времени проектирования. Еще более важным является то, что любая подпрограмма допускает автономную отладку. Это, как правило, многократно сокращает время отладки всего прикладного программного обеспечения. И, наконец, механизм использования подпрограмм, реализующих требуемый набор процедур, уменьшает длину прикладной программы, что имеет своим следствием уменьшение требующейся емкости памяти программ.

Существенным является и то обстоятельство, что отложенные процедуры организуются разработчиками в библиотеки параметризуемых подпрограмм и могут быть многократно использованы в проектной работе. Отметим, что библиотека параметризуемых подпрограмм строится на основе соглашения о едином способе обмена параметрами.

4.4.

Правила записи программ на языке ассемблера

Исходный текст программы на языке ассемблера имеет определенный формат. Каждая команда (и псевдокоманда) представляет собой строку четырехзвенной конструкции:

МЕТКА ОПЕРАЦИЯ ОПЕРАНД(Ы) КОММЕНТАРИЙ
Звенья (поля) могут отделяться друг от друга произвольным числом пробелов.

Метка. В поле метки размещается символическое имя ячейки памяти, в которой хранится отмеченная команда или операнд. Метка представляет собой буквенно-цифровую комбинацию, начинающуюся с буквы. Используются только буквы латинского алфавита. Ассемблер МК48 допускает использование в метках символа подчеркивания (_). Длина метки не должна превышать шесть символов для МК48 и 31 – для МК51. Метка всегда завершается двоеточием (:).

Псевдокоманды ассемблера не преобразуются в двоичные коды, а потому не могут иметь меток. Исключение составляют псевдокоманды резервирования памяти и определения данных (DS, DB, DW). У псевдокоманд, осуществляющих определение символьических имен, в поле метки записывается определяемое символьическое имя, после которого двоеточие не ставится.

В качестве символьических имен и меток не могут быть использованы мнемокоды команд, псевдокоманд и операторов ассемблера, а также мнемонические обозначения регистров и других внутренних блоков МК.

Операция. В поле операции записывается мнемоническое обозначение команды МК или псевдокоманды ассемблера, которое является сокращением (аббревиатурой) полного английского наименования выполняемого действия. Например: MOV – move – переместить, JMP – jump – перейти, DB – define byte – определить байт.

Для МК48 и МК51 используется строго определенный и ограниченный набор мнемонических кодов. Любой другой набор символов, размещенных в поле операции, воспринимается ассемблером как ошибочный.

Операнды. В этом поле определяются операнды (или operand), участвующие в операции. Команды ассемблера могут быть без-, одно- или двухоперандными. Операнды разделяются запятой (,).

Операнд может быть задан непосредственно или в виде его адреса (прямого или косвенного). Непосредственный операнд представляется числом (MOV A, #15) или символическим именем (ADDC A,#OPER2) с обязательным указанием префикса непосредственного операнда (#). Прямой адрес операнда может быть задан мнемоническим обозначением (IN A, P1), числом (INC 40), символическим именем (MOV A, MEMORY). Указанием на косвенную адресацию служит префикс @. В командах передачи управления операндом может являться число (LCALL 0135H), метка (JMP LABEL), косвенный адрес (JMPP @A) или выражение (JMP \square -2, где \square – текущее содержимое счетчика команд).

Используемые в качестве операндов символические имена и метки должны быть определены, а числа представлены с указанием системы счисления, для чего используется суффикс (буква, стоящая после числа): В – для двоичной, Q – для восьмеричной, D – для десятичной и H – для шестнадцатеричной. Число без суффикса по умолчанию считается десятичным.

Обработка выражений в процессе трансляции. Ассемблеры МК48 и МК51 допускают использование выражений в поле операндов, значения которых вычисляются в процессе трансляции.

Выражение представляет собой совокупность символьических имен и чисел, связанных операторами ассемблера. Операторы ассемблера обеспечивают выполнение арифметических ("+" – сложение, "-" – вычитание, * – умножение, / – целое деление, MOD – деление по модулю) и логических (OR – ИЛИ, AND – И, XOR – исключающее ИЛИ, NOT – отрицание) операций в формате 2-байтных слов.

Например, запись ADD A.##((NOT 13) + 1) эквивалентна записи ADD A, #0F3H и обеспечивает сложение содержимого аккумулятора с числом –13, представленным в дополнительном коде.

Широко используются также операторы LOW и HIGH, позволяющие выделить младший и старший байты 2-байтного операнда.

Комментарий. Поле комментария может быть использовано программистом для текстового или символьного пояснения логической организации прикладной программы. Поле комментария полностью игнорируется ассемблером, а потому в нем допустимо использовать любые символы. По правилам языка ассемблера поле комментария начинается после точки с запятой (:).

Псевдокоманды ассемблера. Ассемблирующая программа транслирует исходную программу в объектные коды. Хотя транслирующая программа берет на себя многие из рутинных задач программиста, таких как присвоение действительных адресов, преобразование чисел, присвоение действительных значений символьным переменным и т.п., программист все же должен указать ей некоторые параметры: начальный адрес прикладной программы, конец ассемблируемой программы,

форматы данных и т.п. Всю эту информацию программист вставляет в исходный текст своей прикладной программы в виде псевдокоманд (директив) ассемблера, которые только управляют процессом трансляции и не преобразуются в коды объектной программы.

Псевдокоманда ORG 10H задает ассемблеру адрес ячейки памяти (10H), в которой должна быть расположена следующая за ней команда прикладной программы.

Псевдокомандой EQU можно любому символическому имени, используемому в программе, поставить в соответствие определенный операнд. Например, запись

```
RET    EQU    13
```

приводит к тому, что в процессе ассемблирования всюду, где встретится символическое имя RET, оно будет заменено числом 13.

Символические имена операндов, определяемых в процессе исполнения программы, определяются псевдокомандой SET:

```
ALFA    SET    3
...
...
ALFA    SET    ALFA+1
```

Ассемблер MK51 позволяет определить символическое имя как адрес внутренних (псевдокоманда DATA), внешних (XDATA) данных или адрес бита (псевдокоманда BIT). Например, директива

```
ERROR_FLAG    BIT    25H.3
```

определяет символическое имя ERROR_FLAG как третий бит ячейки ОЗУ с адресом 25H.

- Псевдокоманда DB обеспечивает занесение в ПП константы, представляющей собой байт.

Псевдокомандой END программист дает ассемблеру указание об окончании трансляции.

В результате трансляции должна быть получена карта памяти программ, где каждой ячейке памяти поставлен в соответствие хранящийся в ней код.

В соответствии с форматом команд для представления их объектных кодов отводятся одна, две или три ячейки памяти программ. В первой ячейке всегда располагается код операции, во второй (а для MK51 и в третьей) — непосредственный операнд, адрес прямоадресуемого операнда, адрес перехода внутри страницы памяти программ (для команд передачи управления MK48) или смещение (для команд передачи управления MK51). Для команд LCALL и LJMP во втором и третьем байтах объектного кода указывается адрес передачи управления (во втором — старшая часть, в третьем — младшая).

4.5.

Ввод, редактирование, трансляция и отладка прикладных программ в кросс-системах разработки

Написанием текста программы заканчивается первый этап разработки прикладного программного обеспечения — "от постановки задачи к исходной программе" и начинается следующий — "от исходной программы к объектному модулю".

Для простых программ объектный код может быть получен вручную (ручная трансляция). Однако для более сложных программ требуются специальные средства автоматизации подготовки программ. Обычно такие средства используют большие емкости памяти и широкий набор периферийных устройств, в силу чего они не могут быть резидентными, а используются только в кросс-режиме на универсальных мини- и микроЭВМ (СМ-1800, СМ-1810, СМ-4, ЕС-1841).

В минимальный состав программного обеспечения кросс-средств входят:

системная программа для ввода исходного текста прикладной программы, его редактирования и записи на внешней носитель информации — так называемый редактор текстов, или символьный редактор (наиболее распространенные названия CREDIT, EDITER, ED);

программа-транслятор, обеспечивающая преобразование исходного текста прикладной программы в объектный модуль (ASM48, ASM51).

Более мощные кросс-средства предполагают наличие редактора внешних связей (LINK), позволяющего включать в программу модули, разработанные независимо друг от друга, и программу, обеспечивающую настройку перемещаемых программных модулей на абсолютные адреса (LOCATE).

Для ввода исходного текста прикладной программы необходимо вызвать редактор текстов, указав ему тип носителя, на котором будет создан исходный файл. Чаще всего в качестве носителя используется копипаста на гибком магнитном диске. Ниже приводится фрагмент диалога с микроЭВМ при подготовке программ в среде ДОС1800 на микроЭВМ СМ-1800, обеспечивающий создание исходного файла PROBL1 на гибком магнитном диске, расположенном на дисководе № 1.

Вызов редактора:

```
-CREU11 :F1:PROBL1.ASM<(CR)>
ISIS-II CRT-BASED EDITER VX.Y
NEW FILE
MAKFILEAARL
-----
```

Ввод текста программы;
запись созданного файла на дискету;

```
<ПОСЛЕДНЯЯ СТРОКА ПРОГРАММЫ>(CR)
3 <(HOME)>
* EX<(CR)>
EDITB :F1:PROBL1.ASM
```

Кросс-система автоматически выдает на экран выделенные сообщения. В двойных угловых скобках указаны имена функциональных клавиш, нажимаемых оператором.

Для трансляции исходного текста программы необходимо вызвать транслятор, указав ему файл с исходным текстом, место размещения объектного кода, а также условия формирования и вывода листинга. Например, диалог

```
-ASM48 :F1:PROBL1.ASM<(CR)>
== DOS-MACRO ASSEMBLER ==
```

обеспечивает формирование объектного кода в файле PROBL1.HI:X и листинга в файле PROBL1.LST на гибком магнитном диске. После окончания трансляции при отсутствии синтаксических ошибок выдается сообщение

```
ASSEMBLY COMPLETE NO ERRORS
```

или сообщение

```
ASSEMBLY COMPLETE NNN ERRORS(LL)
```

с указанием числа ошибок (NNN) и номера последней ошибочной строки (LLL) при наличии синтаксических ошибок.

Все обнаруженные ошибки исправляются в исходном тексте прикладной программы (это относится и к ошибкам, обнаруженным на этапе отладки). Для этого необходимо вновь вызвать редактор текста и осуществить редактирование исходного текста программы, а затем выполнить повторную трансляцию.

Если исходный текст прикладной программы не имел внешних ссылок и содержал директиву ORG, то после успешного завершения трансляции этап разработки программного обеспечения "от исходной программы к объектному модулю" можно считать законченным.

4.6.

Отладка прикладного программного обеспечения микроконтроллеров

После получения объектного кода прикладной программы неизбежно наступает этап отладки, т.е. установления факта ее работоспособности, а также выявления (локализации) и устранения ошибок. Без этого этапа разработки никакое программное обеспечение вообще не имеет

права на существование. Отладка программного обеспечения представляет собой отдельную сложную задачу, которая почти не поддается формализации и требует для своего выполнения высокого профессионализма и глубоких знаний разработчика.

Обычно отладка прикладного программного обеспечения осуществляется в несколько этапов. Простые (синтаксические) ошибки выявляются уже на этапе трансляции. Далее необходимо выполнить:

автономную отладку каждой процедуры в статическом режиме, позволяющую проверить правильность проводимых вычислений, правильность последовательности переходов внутри процедуры (отсутствие "зацикливания") и т.п.;

комплексную отладку программного обеспечения в статическом режиме, позволяющую проверить правильность алгоритма управления (по последовательности формирования управляющих воздействий);

комплексную отладку в динамическом режиме без подключения объекта для определения реального времени выполнения программы и ее отдельных фрагментов.

Следует иметь в виду, что автономная отладка отдельных модулей настолько проще и эффективнее отладки всей прикладной программы, что переходить к этапу комплексной отладки целесообразно только после исчерпания всех средств автономной отладки.

Вышеперечисленные этапы отладки осуществляются обычно с использованием кросс-систем.

В состав кросс-систем входят программы-отладчики (обобщенное имя – DEBUG), интерпретирующие (моделирующие) выполнение программ, написанных для МК. Такие программные имитаторы позволяют эффективно отлаживать вычислительные процедуры, а также алгоритм функционирования контроллера.

Разработчику предоставлен доступ к любому ресурсу МК, имеется возможность покомандного и пофрагментного исполнения программ и останова по условию, а также подсчета числа тактов выполнения тех или иных фрагментов программы, инициирования прерывания, дисассемблирования содержимого ПИ и т.д.

Кросс-отладчики позволяют промоделировать практически все возможные варианты работы программы и тем самым убедиться в ее работоспособности. На этом же этапе возможна проверка работоспособности программы при нештатных ситуациях в условиях поступления некорректных входных воздействий (для применений с повышенными требованиями по безопасности).

Наиболее мощные имитаторы должны позволять моделировать и среду обитания МК, т.е. различного рода объекты и датчики, подключаемые к нему. При этом появляется возможность выполнять комплексную отладку программного обеспечения, не опасаясь, что возможные ошибки в программе, алгоритме или некорректные действия оператора приведут к выходу из строя технических средств разрабатываемой системы. Главным недостатком кросс-систем является невозможность прогоня программы в реальном масштабе времени, т.е. со скоростью,

близкой к скорости выполнения программы в самом МК, а также невозможность комплексирования аппаратурных и программных средств разрабатываемой системы. В силу этих причин достоверность прикладных программ, отложенных в кросс-режиме, недостаточно высока.

Отдельные фрагменты программного обеспечения, требующие отладки в реальном времени, могут быть проверены на отладочном модуле. Отладочный модуль представляет собой небольшую, как правило, одноплатную микроЭВМ (правильнее, микроконтроллер), построенную на однотипном МК. Однако при отладке приходится учитывать ограничения, связанные с тем, что часть ресурсов отладочного модуля (пространство адресов памяти программ и памяти данных, некоторые линии портов и уровни прерываний) не может быть использована прикладным ПО, так как вынужденно используется резидентной операционной системой (ОС). Резидентная ОС, или Монитор, — это программа, обеспечивающая взаимодействие оператора с отладочным модулем и предоставляющая ему ряд команд, облегчающих отладку прикладного программного обеспечения. К отладочному модулю может быть также подключено дополнительное оборудование, необходимое разрабатываемой системе (внешняя память, порты, таймеры).

Наиболее полная и комплексная отладка прикладного программного обеспечения совместно с аппаратурными средствами контроллера может быть произведена на инструментальной микроЭВМ с так называемым внутрисхемным эмулятором (ВСЭ). Прототип разрабатываемой системы через панельку (сокет) для установки МК плоским многожильным кабелем соединяется с ВСЭ, который в свою очередь обеспечивает доступ ко всем техническим средствам инструментальной микроЭВМ. При этом почти все ресурсы МК остаются в распоряжении Прикладного программного обеспечения. Под управлением микроЭВМ ВСЭ позволяет прогонять прикладную программу или ее отдельные фрагменты в реальном темпе, останавливать выполнение программы по многим признакам, делать трассировку внешних сигналов МК и системы во время исполнения программы. Достоверность программного обеспечения, отложенного на инструментальной микроЭВМ с помощью ВСЭ, высока, хотя и не равна единице.

В любом случае для доводки прикладного программного обеспечения контроллера необходимы комплексные и всесторонние испытания разработанной системы в реальном окружении и во всевозможных режимах.

Обработка данных в микроконтроллерах МК48 и МК51

5.1.

Примеры программ обработки данных в МК48

5.1.1. Примеры использования команд передачи данных

Пример 5.1. Записать в РПД, в ячейки с адресами 41 и 42 число 1C3FH:

LOAD:	MOV	R0,\$41	;ЗАГРУЗКА В R0 УКАЗАТЕЛЯ РПД
	MOV	ERO,\$1CH	;ЗАЙМСЯ В РПД ЧИСЛА 1СН
	INC	R0	;ПРОВЕДЕНИЕ УКАЗАТЕЛЯ АДРЕСА РПД
	MOV	ERO,\$3FH	;ЗАЙТИ В РПД ЧИСЛА ЗН

Пример 5.2. Переслать текущее содержимое таймера в R5 без потери содержимого аккумулятора:

XCHNG:	XCH	A,R5	;ФОРМЕН R5 И АККУМУЛЯТОРА
	MOV	A,T	;ПЕРЕСЫЛКА СОДЕРЖИМОГО ТАЙМЕРА
	XCH	A,R5	;В АККУМУЛЯТОР
			;ФОРМЕН R5 И АККУМУЛЯТОРА

Пример 5.3. Передать содержимое регистров банка 0 в ВПД, начиная с адреса 50H:

	MOV	A,\$10000B	;ВЫБОР БАНКА РЕГИСТРОВ 1
	MOV	PSW,A	
	MOV	R0,\$50H	;ПОГРЕДЕНИЕ ИЧАЛЬНОГО АДРЕСА ВРА
	MOV	R1,\$0	;ПОГРЕДЕНИЕ ИЧАЛЬНОГО АДРЕСА БАНКА РЕГИСТРОВ
	MOV	R2,\$8	;СЧЕТЧИК РЕГИСТРОВ (ЧИКЛОВ) (--- 8)
LOOP:	MOV	A,R0	;ПЕРЕСЫЛКА БАНКА ИЗ РЕГИСТРА В ВРА
	MOVX	ERO,A	;ЧЕРЕЗ АККУМУЛЯТОР
	INC	R0	;ПРОДВИЖЕНИЕ УКАЗАТЕЛЕЙ
	DJNZ	R2,LOOP	;ПРОДОЛЖИТЬ, ЕСЛИ ПЕРЕДАНЫ
			;НЕ ВСЕ РЕГ ТР

Пример 5.4. Ввести байт из порта 1 и передать его в порт 2:

TRAN:	MOV	A#\$OFFH	;НАСТРОЙКА ПОРТА 1 НА ВВОД
	OUTL	P1,A	
	IN	A#P1	;ВВОД БАЙТА ИЗ ПОРТА 1
	OUTL	P2,A	;ВЫХОД БАЙТА В ПОРТ 2

Пример 5.5. Ввести данные из порта P7:

```
INPUT: MOVD A,P7           ;ПЕРЕСЫЛКА ЧЕТЫРЕХ БИТОВ ИЗ ПОРТА 7
       ; В МЛАДШУЮ ТЕТРАДУ АККУМУЛЯТОРА
```

Пример 5.6. Вычислить произведение двух 4-битных чисел, расположенных в младших тетрадах регистров R0 и R1. Для вычисления используется таблица произведений для всех комбинаций сомножителей (всего 256). Произведение двух тетрад имеет формат 1 байт. Таким образом, необходимая таблица произведений занимает одну страницу памяти. Данную таблицу удобно разместить на третьей странице РПП:

```
;Вычисление Z=XY
;R0=0000.XXXX
;R1=0000.YYYY
;X И Y ПРИНИМАЮТ ЗНАЧЕНИЯ 0 И 1

ORG 0           ;ДИРЕКТИВА АССЕМБЛЕРА, ЗАДАЮЩАЯ
                 ;Начальный адрес программы
MOV A,R0         ;ПЕРЕСЫЛКА МЛАДШЕГО В АККУМУЛЯТОР
SWAP A           ;Обмен тетрад акумулятора
ORL A,R1         ;ФОРМИРОВАНИЕ В АККУМУЛЯТОРЕ АДРЕСА ПРОИЗВЕДЕНИЯ
MOVWF3 A,RA      ;ЗАГРУЗКА В АККУМУЛЯТОР ПРОИЗВЕДЕНИЯ
ORG 0300H        ;ДИРЕКТИВА АССЕМБЛЕРА, ЗАДАЮЩАЯ НАЧАЛЬНЫЙ
                 ;адрес таблицы на третьей странице РПП
;ДИРЕКТИВЫ АССЕМБЛЕРА, ФОРМИРУЮЩИЕ ТАБЛИЦУ ПРОИЗВЕДЕНИЙ
DB 0,0,0,0,0,0,0,0 ;Z=0*Y
DB 0,0,0,0,0,0,0,0
DB 1=0,1=1,1=2,...,1=OFH ;Z=1*Y
...
...
DB 0F00,0FH=1,0FH=2,...,0FH=OFH ;Z=OF*Y
```

Время выполнения программы 12,5 мкс.

5.1.2. Примеры использования команд арифметических операций

Пример 5.7. Сложить содержимое регистра R7 и ячейки РПД с адресом 60H:

```
MOV R0,60H ;ЗАГРУЗКА В R0 АДРЕСА РПД
MOV A,R7   ;ЗАГРУЗКА ОПЕРАНДА В АККУМУЛЯТОР
ADD A,R0   ;СЛОЖЕНИЕ
```

Результат операции сложения фиксируется в аккумуляторе, установка флага переноса будет свидетельствовать о переполнении.

Пример 5.8. Сложить десятичные двоично-кодированные числа (BCD-числа), расположенные в A и R7:

```
ADD A,R7   ;ДВОИЧНОЕ СЛОЖЕНИЕ
BA A       ;КОРРЕКЦИЯ РЕЗУЛЬТАТА
```

Пример 5.9. Инкрементировать содержимое ячеек РПД по адресам 10–18:

```
INCR: MOV R0,$10    ;ЗАГРУЗКА В R0 НАЧАЛЬНОГО АДРЕСА
       MOV R3,$(18-10+1) ;ЗАГРУЗКА В R3 ЧИСЛА ЯЧЕК
LOOP: INC R0        ;ИНКРЕМЕНТ ЯЧЕКИ РПД
      INC R0        ;ПРОДВИЖЕНИЕ УКАЗАТЕЛЯ АДРЕСА
      DJNZ R3,LOOP  ;ИНКРЕМЕНТ R3 И ПОВТОР,
                  ;ПОКА R3 НЕ РАВНО НУЛЮ
```

Пример 5.10. Сложить многобайтные BCD-числа, расположенные в РПД. Регистры R0 и R1 указывают начальные адреса слагаемых. Слагаемые расположены в РПД, начиная с младших байтов. Формат слагаемых одинаков и задается в R2 числом байтов. Результат сложения поместить на место первого слагаемого:

```
;СУММИРОВАНИЕ Z=W+Y
;R(R0) - НАЧАЛЬНЫЙ АДРЕС W
;R(R1) - НАЧАЛЬНЫЙ АДРЕС Y
;R(R2) - ДЛИНА СЛАГАЕМЫХ И Y
CLR C           ;СБРОС ФЛАГА ПЕРЕНОСА
LOOP: MOV A,R0   ;ЗАГРУЗКА ТЕКУЩЕГО БАЙТА W
      ADDC A,R1   ;СЛОЖЕНИЕ
      DA A        ;КОРРЕКЦИЯ
      MOV R0,A    ;ПРАЗМЕНИЕ ТЕКУЩЕГО БАЙТА РЕЗУЛЬТАТА
      INC R0      ;ПРОДВИЖЕНИЕ УКАЗАТЕЛЕЙ БАЙТ СЛАГАЕМЫХ
      INC R1      ;ИНКРЕМЕНТ R2, ПОВТОР, ПОКА R2 НЕ РАВНО 0
      DJNZ R2,LOOP ;ИНКРЕМЕНТ R2, ПОВТОР, ПОКА R2 НЕ РАВНО 0
```

Время суммирования составит $(1 + 8N) \times 2,5$ мкс, где N – длина слагаемых в байтах.

Пример 5.11. Вычитание байтов. Операция вычитания может быть выполнена двумя способами: переводом вычитаемого как отрицательного числа в дополнительный код с последующим сложением; переведом уменьшаемого в обратный код с последующей инверсией суммы.

Пусть требуется вычесть из A содержимое регистра R6. Вычитание выполнить в соответствии с выражением

$$(A) \leftarrow \overline{(A)} + (R6).$$

```
CPL A           ;ИНВЕРСИЯ АККУМУЛЯТОРА
ABD A,R6       ;СЛОЖЕНИЕ
CPL A           ;ПОЛУЧЕНИЕ РАЗНОСТИ
```

Установка флага С после выполнения сложения будет свидетельствовать об отрицательном переполнении.

Пример 5.12. Получить разность 2-байтных чисел без знака. Операнды располагаются в РПД. Адрес уменьшаемого хранится в R1, а вычитаемого – в R0. Результат поместить на место уменьшаемого:

```

;ВЫЧИСЛЕНИЕ Z=X-Y
;X,Y - РВА
;R0 - АРЕСТ Y
;R1 - АРЕСТ X
;РЕЗУЛЬТАТ НА МЕСТО X
SUBST: MOV A,000 ;ЗАГРУЗКА МЛАДШЕГО БАЙТА Y
        CPL A ;ПОЛУЧЕНИЕ ЛОГОВНІТЕЛЬНОГО КОДА Y
        INC A
        A,BR1 ;ВЧИТАННІ ІСІДІВІХ БАЙТІВ
        MOV BR0,A ;ЗАПОМІННІЕ МЛАДШЕГО БАЙТА РАЗНОСТИ
        INC R0 ;СЕРЕХОК К СТАРШИМ БАЙТАМ X И Y
        INC R1
        MOV A,BR0 ;ЗАГРУЗКА СТАРШЕГО БАЙТА Y
        CPL A ;ОБРАТНИЙ КОД Y
        A,BR1 ;ВЧИТАННІ СТАРШИХ БАЙТІВ
        MOV BR0,A ;ЗАПОМІННІЕ РЕЗУЛЬТАТА

```

Пример 5.13. Умножить однобайтные целые числа без знака. В регистре R1 размещён множитель, в регистре R2 – множимое. Двухбайтный результат умножения будет размещен в аккумуляторе (старший байт) и в R1 (младший байт) вместо множителя. В регистр R3, выполняющий функции счетчика программных циклов, загружается число 8 (число бит множителя). Умножение выполняется младшими битами вперед со сдвигом вправо частичного произведения. Последовательность действий при этом методе умножения следующая:

1. Содержимое аккумулятора и регистра-расширителя R1 сдвигаются вправо на один бит так, что младший бит множителя, выдвигаемый из регистра R1, помещается в триггер флага C.
2. Если C = 1, то множимое добавляется к содержимому аккумулятора, в противном случае никаких операций не производится.
3. Декрементируется счетчик циклов R3, и если его содержимое не равно нулю, то все действия повторяются.
4. Перед выходом из подпрограммы формируется окончательный результат сдвигом частичного результата на один бит вправо:

```

MPLY1B: MOV R3,#8 ;ЗАГРУЗКА СЧЕТЧИКА ЦИКЛОВ
        CLR A ;ОЧИСТКА АККУМУЛЯТОРА
        CLR C ;ОЧИСТКА ПРИЗНАКА ПЕРЕНОСА
SHIFT: RRC A ;САВІГ АККУМУЛЯТОРА ВПРАВО
        XCH A,R1 ;ОБМЕН АККУМУЛЯТОРА И R1
        RRC A ;САВІГ МНОЖИТЕЛЯ С ЗАНЕСЕNIЕМ
        ; ВЫВИДАЕМОГО БИТА В С
        XCH A,R1 ;ОБМЕН АККУМУЛЯТОРА И R1
        JNC RESULT ;ЕСЛИ C=1, ТО СУММІРОВАНІЕ
        ADD A,R2 ;ПРИДАВАННІЕ МНОЖИМОГО
        RESULT: DJNZ R3,SHIFT ;ДЕКРЕМЕНТ СЧЕТЧИКА И ПРОВЕРКА
                           ; ОКОНЧАНИЯ ОПЕРАЦІІ (R3=0)
        RRC A ;САВІГ АККУМУЛЯТОРА
        XCH A,R1 ;ОБМЕН
        RRC A ;САВІГ СОДЕРЖИМОГО R1
        XCH A,R1 ;ОБМЕН

```

Максимальное время выполнения программы составляет 200 мкс.

5.1.3. Примеры использования команд логических операций

Пример 5.14. Маскирование при вводе. Ввести в регистр R7 информацию из линий 0, 1, 3, 4 и 7 порта 1:

IN	A,P1	;ВВОД БАЙТА ИЗ ПОРТА 1
ANL	A,\$10011011B	;МАСКІРОВАННІЕ
MOV	R7,A	;ПЕРЕДАЧА

Пример 5.15. Ввести в аккумулятор данные из порта 2 и выделить требуемые биты по маске, находящейся в R0:

IN	A,P2	;ВВОД ИЗ ПОРТА 2
ANL	A,R0	;МАСКІРОВАННІЕ

Пример 5.16. Выполнить логический сдвиг влево двухбайтного слова, расположенного в (R2) (A):

SHFLL: RLC	A	;САВІГ МЛАДШЕГО БАЙТА
XCH	A,R2	;ОБМЕН АККУМУЛЯТОРА И РАСВІРІТЕЛЯ
RLC	A	;САВІГ СТАРШЕГО БАЙТА
XCH	A,R2	;ОБМЕН

Пример 5.17. Выполнить арифметический сдвиг двухбайтного слова (R2) (A) вправо:

SIFAR: CLR	C	;СВРОС ФЛАГА ПЕРЕНОСА
CPL	C	;УСТАНОВКА ФЛАГА ПЕРЕНОСА
XCH	A,R2	;ОБМЕН БАЙТАМИ
JB7	#3	;ЕСЛИ R2>НЕ РАВНО 1, ТО СВРОС
CLR	C	;ФЛАГА ПЕРЕНОСА
RRC	A	;САВІГ ФЛАГА ПЕРЕНОСА В РАСВІРІТЕЛЬ
XCH	A,R2	;ОБМЕН
RRC	A	;САВІГ МЛАДШЕГО БАЙТА

Пример 5.18. Умножить аккумулятор на число 2 в степени X, где X – число (не более 8), хранящееся в R2. Умножение на 2 заменяется арифметическим сдвигом влево аккумулятора и расширителя R1:

LOOP: MOV	R1,#0	;СВРОС R1
CLR	C	;СВРОС ФЛАГА ПЕРЕНОСА
RLC	A	;АРИФМЕТИЧЕСКИЙ САВІГ ВЛЕВО ОВ'ЄДИНЕННОГО
XCH	A,R1	; 16-БІТНОГО РЕЗУЛЬТАТА В
RLC	A	; РЕГІСТРОВОЮ ПАРУ (R1)(A)
XCH	A,R1	
DJNZ	R2,LOOP	;ЦИКЛ

Пример 5.19. Выдать содержимое аккумулятора в последовательном коде через нулевую линию порта 1, оставляя без изменения остальные биты порта. Передачу вести, начиная с младшего бита:

```

LOOP: MOV R1.00 :СЧЕТЧИК БИТ
      JBO ONE :ПЕРЕХОД, ЕСЛИ БИТ A.0 РАВЕН 1
      ANL P1.0#(NOT 1) :СБРОС Р1.0
      JMP NEXT
NEXT: ORI P1.01 :УСТАНОВКА Р1.0
      JMP NEXT :ИЗБЫТОЧНАЯ КОМАНДА ДЛЯ ВЫРАВНИВАНИЯ
      ; ВРЕМЕННИ ПЕРЕЛАМИ 0 И 1
NEXT: RR A :САГИВ АККУМУЛЯТОРА ПРАВО (ПОДГОТОВКА
      DJNZ R1,LOOP ; ЧЕРЕМОГО БИТА)

```

Пример 5.20. Настроить биты 0–3 порта 1 на ввод:

```
ORL P1.4 F ;УСТАНОВКА БИТОВ P1.0...P1.3
```

Пример 5.21. Очистить биты 4–7 порта 2:

```
ANL P2.40FH :СБРОС БИТОВ P2.4...P2.7
```

Пример 5.22. Выдать в линию 0 порта 4 значение четвертого бита аккумулятора:

```

SWAP A :ИЗМЕНЕНИЕ БИТОВ 0 И 4 АККУМУЛЯТОРА
ANL A#1 :ВЫЛЕЖЕНИЕ БИТА A.0
ORL D P4,A :УСТАНОВКА Р4.0, ЕСЛИ A.0=1
ANL A:#0EH :УСТАНОВКА БИТОВ 1...3 АККУМУЛЯТОРА
ANLD P4,A :СБРОС Р4.0, ЕСЛИ A.0=0

```

Пример 5.23. Определить четность числа единиц в аккумуляторе:

```

CLR F0 :СБРОС F0
MOV R7,#B :ЧИСЛО ПОВТОРОВ
LOOP: RNC A :ПЕРЕСЫЛКА БИТА A.0 В ПЕРЕНОС
      JNC NEXT :ПРОПУСТИТЬ, ЕСЛИ БИТ РАВЕН 0
      CPL F0 :ПОЧАСТ ПАРИТЕТА
      DJNZ R7,LOOP :ПОВТОРИТЬ В РАЗ

```

После выполнения программы аккумулятор сохранит свое значение, флаг F0 будет установлен, если число единиц в аккумуляторе было нечетно. Флаг F0 входит в состав PSW и в данном примере специфицирован пользователем для выполнения функций флага паритета.

5.1.4. Примеры использования команд передачи управления и команд управления режимом MK48

Пример 5.24. Передать управление по метке LL, если переключатель банка регистров (бит PSW.4) установлен:

```
JBSSET: MOV A#PSW LL :ПЕРЕДАЧА PSW В АККУМУЛЯТОР
      JB4 LL :ПЕРЕХОД, ЕСЛИ A.4=1
```

Пример 5.25. Передать управление по метке LABFL, если счетчик событий достиг состояния 64:

```
TESTC: MOV A#1 :ПЕРЕСЫЛКА СОДЕРЖИМОГО СЧЕТЧИКА В АККУМУЛЯТОР
      JB6 LABEL :ПЕРЕХОД НА МЕТКУ, ЕСЛИ A.6=1
```

Пример 5.26. Осуществить переход из нулевого банка ПП к программе с именем ROUT, расположенной в первом банке ПП:

```
SEL MB1 :УСТАНОВКА ФЛАГА BRF
JMP ROUT :ПЕРЕХОД К ПРОГРАММЕ ROUT
```

Пример 5.27. Множественное ветвление программы.

Допустим, что результатом работы некоторой программы является число X (в пределах от 0 до 15). Необходимо организовать передачу управления 16 различным программам с именами ROUT0 – ROUTF в зависимости от вычисленного значения X:

```

ORG 0 :ЗАДАНИЕ НАЧАЛЬНОГО АДРЕСА ПРОГРАММЫ
ANL A:#0FH :СБРОС СТАРШЕЙ ТЕРДЫ А
              ; ВО ИЗБЕЖАНИЕ ОШИБКИ ПЕРЕХОДА
JMPP OA :ОБРАЩЕНИЕ К ТАБЛИЦЕ ВЕКТОРОВ ПЕРЕХОДОВ
;ТАБЛИЦА ВЕКТОРОВ ПЕРЕХОДОВ
DB ROUT0 :НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ ROUT0
DB ROUT1 :НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ ROUT1
DB ROUT2 :НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ ROUT2
...
...
DB ROUTF :НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ ROUTF

```

Заметим, что команда JMPP, таблица векторов и программы ROUT0 – ROUTF должны находиться на одной странице ПП.

Пример 5.28. Организовать ожидание появления нулевого уровня на входе T0:

```
WAIT: JNE A#T :ПЕРЕХОД А ИЗ А, ЕСЛИ НА ВХОДЕ ЗПР НУЛЬ
```

Пример 5.29. Организовать ожидание появления единичного уровня на входе ЗПР в предположении, что внешние прерывания запрещены:

```
WAIT: JTO WAIT :ПЕРЕХОД НА WAIT, ЕСЛИ НА ВХОДЕ ТО ЕДИНИЦА
```

Пример 5.30. Передать управление одной из восьми программ ROUT0 – ROUT7 при появлении нулевого уровня на соответствующем входе регистра 1. Наивысшим приоритетом обладает вход P1.0:

```

LOOP: ORL P1,$0FFH      ;НАСТРОЙКА ПОРТА 1 НА ВВОД
      IN A,P1          ;ВВОД ДАННЫХ ИЗ ПОРТА 1
      CPL A             ;ИНВЕРСИЯ АККУМУЛЯТОРА
      JZ LOOP           ;ОЖИДАНИЕ ПОЯВЛЕНИЯ ПЕРВОГО НУЛЯ
      JB0 ROUT0         ;ПЕРЕХОД К ROUT0; ЕСЛИ P1.0=0 -
      JB1 ROUT1         ;ПЕРЕХОД К ROUT1; ЕСЛИ P1.1=0 -
      JB2 ROUT2         ;ПЕРЕХОД К ROUT2; ЕСЛИ P1.2=0 -
      JB3 ROUT3         ; ...
      JB4 ROUT4         ; ...
      JB5 ROUT5         ; ...
      JB6 ROUT6         ; ...
      JB7 ROUT7         ;ПЕРЕХОД К ROUT7; ЕСЛИ P1.7=0

```

Ветвление осуществляется группой из восьми команд JB_b. Приоритеты входов порта 1 определяются очередностью проверки.

Пример 5.31. При поступлении на вход T0 последовательности из восьми нулевых импульсов установить выход P2.7:

```

ONE:  MOV R7,$8      ;ЗАГРУЗКА В R7 ЧИСЛА ИМПУЛЬСОВ
      JTO ONE          ;ОЖИДАНИЕ СИГНАЛА 0 НА ВХОДЕ TO
      JMP ZERO         ;ОЖИДАНИЕ СИГНАЛА 1 НА ВХОДЕ TO
ZERO: SKIP           ; ...
      ZERO            ; ...
SKIP: DJNZ R7,ONE    ;ПОВТОРЯТЬ, ПОКА НЕ ПОСТУПИТ ВОСЬМОЙ ИМПУЛЬС
      ORL P2,$0001     ;УСТАНОВКА ЕДИНИЦЫ НА ВЫХОДЕ 7 ПОРТА 2

```

Длительность нуля и единицы на входе устройства должна быть не менее четырех машинных циклов, т.е. 10 мкс.

Пример 5.32. Дождаться поступления на вход T1 100 импульсов и перейти по метке PULSE:

```

MOV A,$156    ;(A) (--- (256-100)
MOV T:A        ;ПРЕУСТАНОВКА СЧЕТЧИКА
STRT CNT      ;ЗАПУСК СЧЕТЧИКА
WAIT: JTF PULSE;ПЕРЕХОД, ЕСЛИ ПРОВЛО 100 ИМПУЛЬСОВ
      JNP WAIT
PULSE: ...

```

Пример 5.33. Запретить прерывания от таймера, но разрешить прерывание после восьми сигналов переполнения таймера. При переходе к процедуре обработки прерывания остановить таймер. Сигналы переполнения подсчитывать в регистре 5:

```

START: DIS TCNT1 ;ЗАПРЕТ ПРЕРЫВАНИЯ ОТ ТАЙМЕРА
      CLR A      ;СБРОС АККУМУЛЯТОРА
      MOV T:A      ;СБРОС ТАЙМЕРА
      MOV RS:A     ;СБРОС РЕГИСТРА 5
      STRT T      ;ЗАПУСК ТАЙМЕРА
      JMP M1       ;ЧИКЛ
COUNT: INC RS      ;ИНКРЕМЕНТ РЕГИСТРА 5
      MOV A,RS      ;ПЕРЕСЫЛКА СОВРЕМЕННОГО RS В АККУМУЛЯТОР
      JB3 INT      ;ПЕРЕХОД К ПОДПРОГРАММЕ ОВСЛУЖИВАНИЯ ПРЕРЫВАНИЯ INT,
                  ;ЕСЛИ БИТ A.3 РАВЕН 1
      JNP M1       ;ПЕРЕХОД, ЕСЛИ БИТ A.3 НЕ РАВЕН 1
      ...
      ...
INT:   STOP TCNT      ;ОСТАНОВКА ТАЙМЕРА
      JNP 07H      ;ПЕРЕХОД К ЯЧЕЙКЕ 7 (ВЕКТОР ПРЕРЫВАНИЯ
                  ;01 СЧЕТЧИКА СОБЫТИЙ)

```

5.2.

Примеры программ обработки данных в MK51

5.2.1. Примеры использования команд передачи данных

Пример 5.34. Передать содержимое буфера УАПП в РПД по косвенному адресу из R0:

```

MOV GRO,SBUF      ;ПЕРЕДАЧА ПРИНЯТОГО ПО ВОСЛЕДОВАТЕЛЬНОМУ
                   ;КАНАЛУ БАЙТА В РПД

```

Пример 5.35. Загрузить в указатель данных начальный адрес 7FO0H массива данных, расположенного в ВПД:

```

MOV DPTX,$7FO0H    ;ЗАГРУЗКА НАЧАЛЬНОГО ЗНАЧЕНИЯ УКАЗАТЕЛЯ ДАННЫХ

```

Пример 5.36. Загрузить управляющее слово в регистр управления таймером:

```

MOV TCON,$00000101B

```

Пример 5.37. Сбросить все флаги пользователя (область РПД с адресами 20H – 2FH):

```

MOV R0,$20H    ;ЗАДАНИЕ НАЧАЛЬНОГО АДРЕСА ОБЛАСТИ ФЛГОВ
MOV R1,$0FH    ;СЧЕТЧИК (ЛИНКА ОБЛАСТИ ФЛГОВ)
LOOP: MOQ GRO+$0  ;СБРОС ОДНОГО БАЙТА (0 ФЛГОВ)
      INC R0      ;ПЕРЕХОД К СЛЕДУЮЩЕМУ БАЙТУ
      DJNZ R0,LOOP ;ЦИКЛ

```

Пример 5.38. Запомнить в ВПД содержимое регистров банка 0. Начальный адрес ВПД – 5000H:

```

MOV PSW,$01000B  ;ВЫБОР БАНКА РЕГИСТРОВ 1
MOV R0+$D          ;СЧЕТЧИК (--- 8
MOV DPTR,$5000H    ;ОПРЕДЕЛЕНИЕ НАЧАЛЬНОГО АДРЕСА ВПД
MOV R1,$0          ;ОПРЕДЕЛЕНИЕ НАЧАЛЬНОГО АДРЕСА РПД
LOOP: MOV A,BR1    ;(A) (--- (РЕГИСТР)
      MOVX BDTR,A  ;ПЕРЕДАЧА ИЗ АККУМУЛЯТОРА В ВПД
      INC R1        ;ПЕРЕХОД К СЛЕДУЮЩЕМУ РЕГИСТРУ
      INC DPTR      ;ПРИРАЩЕНИЕ УКАЗАТЕЛЯ АДРЕСА
      DJNZ R0,LOOP   ;R0=R0-1, ЕСЛИ R0=0, ТО ПОВТОРИТЬ

```

Пример 5.39. Обращение к памяти программ. Часто необходимо иметь в памяти программы таблицы готовых решений. Для возможности работы с такими таблицами, хранящимися в РПП и ВПП, имеются специальные команды обращения к памяти программ – MOVC. Поясним использование этих команд на следующем примере. Требуется составить подпрограмму вычисления синуса угла X (X меняется в пределах от 0 до 89° с дискретностью 1°). Наиболее быстрое вычисление функции можно получить путем выборки готового значения синуса из таблицы. Такая таблица для диапазона 0–89° займет 90 байтов при погрешности 0,4%. Каждый байт таблицы будет содержать дробную часть двоичного пред-

ставления синуса. Исходным параметром для подпрограммы служит значение угла X, находящееся в аккумуляторе:

```
#ЧИСЛЕНИЕ SIN(X) ПО ТАБЛИЦЕ ЗНАЧЕНИЯ
#ВХОД: (A)(-- X В ДЛЯЛАХ ОТ 0 ДО 89 ГРАУСОВ
#ВЫХОД: (A)(-- ДРОБНАЯ ЧАСТЬ ЗНАЧЕНИЯ СИНУСА
```

```
SINX: INC A           ;ИНКРЕМЕНТ АККУМУЛЯТОРА
MOV C A,0A+PC          ;ЗАГРУЗКА ЗНАЧЕНИЯ СИНУСА ИЗ ТАБЛИЦЫ
RET                   ;ВОЗВРАТ

;ТАБЛИЦА ЗНАЧЕНИЯ СИНУСА
SINUS: DB 0           ;:SIN(0)=0
DB 00000100B          ;:SIN(1)=0.017
DB 00001001B          ;:SIN(2)=0.035
...
DB 11111111B          ;:SIN(89)=0.999
```

Примечательно, что данная подпрограмма обходится без использования указателя данных DPTR. Инкремент А перед обращением к таблице необходим из-за наличия однобайтной команды возврата, расположенной между командой MOV C и началом таблицы значений синуса.

Пример 5.40. Операции со стеком. Механизм доступа к стеку MK51 аналогичен MK48: перед загрузкой в стек содержимое регистра-указателя стека (SP) инкрементируется, а после извлечения из стека декрементируется.

По сигналу системного сброса в SP заносится начальное значение 07H. Для переопределения SP можно воспользоваться командой MOV SP,#d.

Таким образом, стек может располагаться в любом месте РПД. Стек используется для организации обращений к подпрограммам и при обработке прерываний, может быть использован для передачи параметров подпрограммам и для временного хранения содержимого регистров специальных функций.

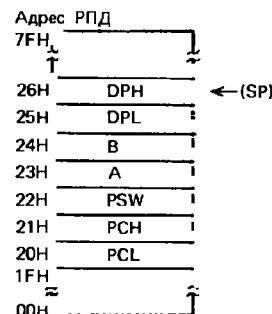


Рис. 5.1. Содержимое стека после выполнения команды CALL и серии команд PUSH

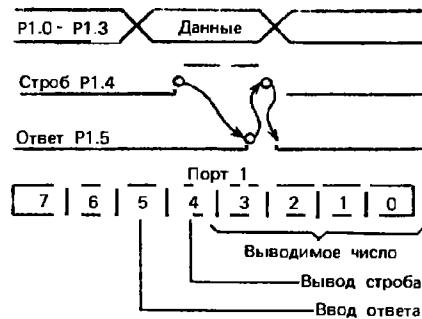


Рис. 5.2. Протокол передачи и распределения бит порта I

Подпрограмма обработки прерывания должна сохранить в стеке содержимое тех регистров, которые она сама будет использовать, а перед возвратом в прерванную программу должна восстановить их значения.

Подпрограмма обработки внешнего прерывания уровня 0 может, например, иметь следующую структуру:

```
ORG 3           ;ЗАДАНИЕ АДРЕСА ВЕКТОРА ПРЕРЫВАНИЯ
SJMP SUBINO    ;ПЕРЕХОД НА ПОДПРОГРАММУ ОБРАБОТКИ
ORG 30H
SUBINO: PUSH PSW      ;СОХРАНЕНИЕ В СТЕКЕ PSW
PUSH ACC      ;СОХРАНЕНИЕ АККУМУЛЯТОРА
PUSH B       ;СОХРАНЕНИЕ В
PUSH DPL      ;СОХРАНЕНИЕ DPTR
PUSH DPH      ;СОХРАНЕНИЕ DPTK
MOV PSW,$1000B ;ВЫБОР БАНКА РЕГИСТРОВ 1
...           ;СОБСТВЕННО ОБРАБОТКА ПРЕРЫВАНИЯ
...
POP DPH      ;ВОССТАНОВЛЕНИЕ DPTK
POP DPL      ;ВОССТАНОВЛЕНИЕ В
POP B       ;ВОССТАНОВЛЕНИЕ АККУМУЛЯТОРА
POP ACC      ;ВОССТАНОВЛЕНИЕ PSW И НОМЕРА
POP PSW      ;РЕГИСТРОВОГО БАНКА
RETI         ;ВОЗВРАТ
```

Если предположить, что SP перед возникновением прерывания содержал значение 1FH, то размещение регистров в стеке после входа в подпрограмму обработки будет таким, как на рис. 5.1.

5.2.2. Примеры использования команд арифметических операций

Пример 5.41. Сложить два двоичных многобайтных числа. Оба слагаемых располагаются в РПД, начиная с младшего байта. Начальные адреса слагаемых заданы в R0 и R1. Формат слагаемых в байтах задан в R2:

CLR	C	#СВРОС ПЕРЕНОСА
MOV	A,R0	ЗАГРУЗКА В АККУМУЛЯТОР ТЕКУЩЕГО
		БАЙТА ПЕРВОГО СЛАГАЕМОГО
ADD	A,R1	СЛОЖЕНИЕ БАЙТА С УЧЕТОМ ПЕРЕНОСА
MOV	R0,A	РАЗМЕЩЕНИЕ БАЙТА РЕЗУЛЬТАТА
INC	R0	ПРОИЗВЕДЕНИЕ УКАЗАТЕЛЕЙ
INC	R1	
DJNZ	K2+LOOP	ЦИКЛ, ЕСЛИ НЕ ВСЕ БАЙТА ПРОСУММИРОВАНЫ

При сложении чисел без знака на переполнение укажет флаг C, а в случае сложения чисел со знаком – флаг OV. Время суммирования составит $(1 + 7N)$ мкс, где N – формат операндов в байтах.

Пример 5.42. Умножение. Команда MUL вычисляет произведение двух целых беззнаковых чисел, хранящихся в регистрах A и B. Младшая часть произведения размещается в A, а старшая – в регистре-расширителе B. Если содержимое B оказывается равным нулю, то флаг OV сбрасывается, иначе – устанавливается. Флаг переноса всегда сбрасывается. Например, если аккумулятор содержал число 200

(0C8H)), а расширитель 160 (0AOH), то в результате выполнения команды MUL AB получится произведение 32 000 (7D00H). Аккумулятор будет содержать нуль, а расширитель – 7DH, флаг OV будет установлен, а флаг C – сброшен.

Пусть требуется умножить целое двоичное число произвольного формата на константу 173. Исходное число размещается в РПД, адрес младшего байта находится в регистре R0. Формат числа в байтах хранится в регистре R1:

```

LOOP: MOV A, #0      ;СБРОС АККУМУЛЯТОРА
      A, #00      ;ЗАГРУЗКА МНОЖИТЕЛЯ
      MOV B, #173   ;ЗАГРУЗКА МНОЖИТЕЛЯ
      MUL AB       ;УМНОЖЕНИЕ
      MOV R0,A     ;ЗАВОДЫ МЛАДШЕГО БАЙТА ЧАСТИЧНОГО ПРОИЗВЕДЕНИЯ
      INC R0       ;ПРИРАЩЕНИЕ АДРЕСА
      MOV A,B     ;ПЕРЕСМАКА СТАРШЕГО БАЙТА ЧАСТИЧНОГО ПРОИЗВЕДЕНИЯ
      ; В АККУМУЛЯТОР
      XCH A, #00    ;ПРЕДВАРИТЕЛЬНОЕ ОФОРМИВАНИЕ ОЧЕРЕДНОГО
      ; БАЙТА ПРОИЗВЕДЕНИЯ
      DJNZ R1,LOOP  ;ЦИКЛ, ЕСЛИ НЕ ВСЕ БАЙТЫ ИСХОДНОГО ЧИСЛА
      ; УМНОЖЕНИЯ НА КОНСТАНТУ

```

Полученное произведение размещается на месте исходного числа и занимает в РПД на один байт больше. Время вычисления составляет $(1 + 13N)$ мкс, где N – формат исходного числа в байтах.

Пример 5.43. Деление. Команда DIV производит деление содержащегося аккумулятора на содержимое регистра-расширителя. После деления аккумулятор содержит целую часть частного, а расширитель – остаток. Флаги C и OV сбрасываются. При делении на нуль устанавливается флаг переполнения, а частное остается неопределенным. Команда деления может быть использована для быстрого преобразования двоичных чисел в десятичные двоично-кодированные (BCD-числа).

В качестве примера рассмотрим программу, которая переводит двоичное число, содержащееся в аккумуляторе, в BCD-код. При таком преобразовании может получиться трехразрядное BCD-число. Старшая цифра (число сотен) будет размещена в регистре R0, а две младшие – в аккумуляторе:

```

MOV B, #100  ;(B) (--) 100 ДЛЯ ВЫЧИСЛЕНИЯ
; КОЛИЧЕСТВА СОТЕК В ЧИСЛЕ
DIV AB      ;АККУМУЛЯТОР СОДЕРЖИТ ЧИСЛО СОТЕК,
; Т.Е. СТАРШУЮ ЦИФРУ
MOV R0,A    ;ПЕРЕСМАКА В R0 СТАРШЕЮ ЦИФРУ
XCH A,B    ;ПЕРЕСМАКА ОСТАТКА ИСХОДНОГО ЧИСЛА
; В АККУМУЛЯТОР
MOV B, #10   ;(B) (--) 10 ДЛЯ ВЫЧИСЛЕНИЯ
; КОЛИЧЕСТВА ДЕСЯТКОВ В ЧИСЛЕ
DIV AB      ;А СОДЕРЖИТ ЧИСЛО ДЕСЯТКОВ, В – ЧИСЛО ЕДИНИЦ
SWAP A      ;РАЗМЕЩЕНИЕ ЧИСЛА ДЕСЯТКОВ В СТАРШЕЙ ТЕТРАДЕ
; АККУМУЛЯТОРА
ADD A,B    ;ПОДСУММИРОВАНИЕ ОСТАТКА (ЧИСЛА ЕДИНИЦ).
; ТЕПЕРЬ АККУМУЛЯТОР СОДЕРЖИТ
; ДВЕ МЛАДШИЕ ЦИФРЫ

```

Время преобразования составляет 16 мкс.

5.2.3. Примеры использования команд логических операций

Пример 5.44. Выбрать нулевой регистровый банк:

```
ANL P2,#10111010B ;СБРОС БИТОВ 0,2,6 ПОРТА 2
```

Пример 5.45. Установить биты 0–3 порта 1:

```
ORL P1,#00001111B ;(P1.0...P1.3) (--- 1111
```

Пример 5.46. Сбросить биты 0, 2, 6 порта 2:

```
ANL PSW,#11100111B ;СБРОС БИТОВ RS1 Н RS0
```

Пример 5.47. Проинвертировать биты порта P1, соответствующие единичным битам аккумулятора:

```
XRL P1,A      ;ИСКЛЮЧАЮЩЕЕ ИЛИ ПОРТА 1
; И АККУМУЛЯТОРА
```

Пример 5.48. Проинвертировать биты 7, 6, 5 порта 0:

```
XRL A,#0FH    ;ИСКЛЮЧАЮЩЕЕ ИЛИ АККУМУЛЯТОРА
; И КОНСТАНТЫ
```

Пример 5.49. Проинвертировать биты 0–3 аккумулятора:

```
XRL P0,#1110000B ;ИСКЛЮЧАЮЩЕЕ ИЛИ ПОРТА 0
; И КОНСТАНТЫ
```

Пример 5.50. Управление группой бит порта. В РПД находится массив распакованных десятичных цифр. Требуется передать массив внешнему устройству в соответствии с протоколом, поясняемым рис. 5.2. Для передачи четырех бит данных используются младшие линии порта 1. Линии P1.4 и P1.5 используются как сигналы квитирования. Передачу данных на выход МК сопровождает сигнал на линии P1.4. Внешнее устройство, приняв данные, сообщает об этом в МК сигналом на входе P1.5. Биты P1.6–P1.7 в процессе передачи данных не должны изменять своего значения. Исходными параметрами для программы являются начальный адрес массива (R0) и длина массива (R1). Неиспользуемые в программе биты порта 1 необходимо сохранить в неизменном виде:

```

LOOP: ORL P1,#0010000B ;НАСТРОЙКА P1.5 НА ВВОД
      MOV A,#00      ;ЗАГРУЗКА БАЙТА В АККУМУЛЯТОР
      ANL P1,#1110000B ;СБРОС ДАННЫХ И СТРОБА
      ORL P1,A      ;ВЫЛАЧА ДАННЫХ
      ORL P1,#0001000B ;ВЫЛАЧА СТРОБА
      WAIT: JNB P1.5,WAIT ;ОЖИДАНИЕ ОТВЕТА
      INC R0        ;ПРОМЕНЬШЕНИЕ УКАЗАТЕЛЯ АДРЕСА
      DJNZ R1,LOOP  ;ЦИКЛ, ЕСЛИ НЕ ВСЕ ДАННЫЕ ПЕРЕДАНЫ

```

5.2.4. Примеры операций с битами

Пример 5.51. В примере 5.19 рассмотрен способ программной реализации процедуры передачи байта последовательным кодом. Если исходные данные те же, что и в примере для MK48, то аналогичная процедура в MK51 реализуется много проще:

```

LOOP: MOV R7+8B ;ИНИЦИАЛИЗАЦИЯ СЧЕТЧИКА ЦИКЛОВ
      RRC A ;ПРИСВОЕНИЕ ФЛАГУ ПЕРЕНОСА ЗНАЧЕНИЯ БИТА A.0
      MOV P1.0,C ;ПЕРЕДАЧА БИТА
      DJNZ R7,LOOP ;ЧИКА, ЕСЛИ НЕ ВСЕ БИТЫ ПЕРЕДАНЫ
  
```

Время исполнения программы – 41 мкс, время передачи бита – 5 мкс (скорость 200 кбит/с).

Пример 5.52. Вычислить булеву функцию трех переменных $Y = X \wedge \bar{V} \vee W \wedge (X \vee V)$. Переменные X, \bar{V} и W поступают на линии 2, 1 и 0 порта 1 соответственно. Результат Y необходимо вывести на линию 3 порта 1:

```

Y BIT P1.3 ;СПЕЦИФИКАЦИЯ БИТ ПОРТА 1
X BIT P1.2
V BIT P1.1
W BIT P1.0

MOV C,X ;БВОЛ X
ANL C,V ;FX AND NOT(V)
MOV F0,C ;ЗАПОМИНАНИЕ РЕЗУЛЬТАТА В F0
MOV C,X ;БВОЛ X
ORL C,V ;FX OR V
ANL C,W ;FW AND (X OR V)
ORL C,F0 ;FXW AND (X OR V) OR (X AND NOT(V))
MOV Y,C ;БВМВОЛ РЕЗУЛЬТАТА
  
```

Флаг F0 используется для промежуточного хранения первой конъюнкции $X \wedge \bar{V}$. Время выполнения программы составляет 14 мкс.

Пример 5.53. Организовать последовательную передачу данных из аккумулятора на нулевой вывод порта 2. Передачу необходимо вести в манчестерском коде (каждый бит передается двумя интервалами: первый интервал содержит инверсию бита, второй – прямое значение):

```

LOOP: MOV R0,8B ;СЧЕТЧИК БИТ
      RRC A ;(--) БИТ
      CPL C ;ИНВЕРСИЯ БИТА
      MOV P2.0,C ;ПЕРЕДАЧА ИНВЕРСИИ БИТА
      CPL C ;ВОССТАНОВЛЕНИЕ ПРВНОГО ЗНАЧЕНИЯ БИТА
      NOP ;ТРИ КОМАНДЫ NOP ДЛЯ ВЫРАВНИВАНИЯ
      NOP ;ДЛИТЕЛЬНОСТИ ИНТЕРВАЛОВ
      NOP
      MOV P2.0,C ;ПЕРЕДАЧА ПРВНОГО ЗНАЧЕНИЯ БИТА
      DJNZ R0,LOOP ;ЧИКА, ЕСЛИ СЧЕТЧИК БИТ НЕ НУЛЕВОЙ
  
```

Передача выполняется младшими битами вперед. Длительность одного интервала равна шести машинным циклам (6 мкс), время передачи бита равно 12 мкс, время передачи байта – 96 мкс (скорость 83 кбит/с или 10,4 кбайт/с).

5.3.

Сравнительный анализ эффективности команд MK48 и MK51

Основным преимуществом системы команд MK51 по сравнению с системой команд MK48 является наличие команд операций с битами, команд вычитания, умножения и деления, команд операций со стеком. Кроме того, MK51 имеет 144 флага, специфицируемых пользователем, и более гибкую логику ветвлений программ.

Мы уже отмечали, что система команд MK48 является подмножеством системы команд MK51. Программы, написанные для MK48, могут быть перекодированы (существует специальная программа-конвертор) для исполнения в MK51. Тем самым обеспечивается преемственность (но не программная совместимость) MK48 и MK51.

Оценку эффективности систем команд MK48 и MK51 выполним на основе сравнения времени выполнения и длины программ, реализующих некоторые типовые процедуры.

Пример 5.54. Установить флаг переноса:

! ВЕРСИЯ ДЛЯ MK48	! ВЕРСИЯ ДЛЯ MK51
CLR C ;СБРОС ФЛАГА С	SETB C ;УСТАНОВКА ФЛАГА С
CPL C ;ИНВЕРСИЯ ФЛАГА С	
! 2 БАЙТА, 5 МИКРОСЕКУНД	
! 1 БАЙТ, 1 МИКРОСЕКУНДА	

Пример 5.55. Установить бит 5 порта 1:

! ВЕРСИЯ ДЛЯ MK48	! ВЕРСИЯ ДЛЯ MK51
IN A,P2 ;БВОЛ ИЗ ПОРТА 2	CPL P2.4 ;ИНВЕРСИЯ P2.4
XRL A,#10H ;ИНВЕРСИЯ БИТА	
OUTL P2,A ;БВМОЛ В ПОРТ 2	
! 4 БАЙТА, 15 МИКРОСЕКУНД	
! 12 БАЙТА, 1 МИКРОСЕКУНДА	

Пример 5.56. Проинвертировать бит 4 порта 2:

! ВЕРСИЯ ДЛЯ MK48	! ВЕРСИЯ ДЛЯ MK51
ORL P1,#20H ;УСТАНОВКА БИТА	SETB P1.5 ;УСТАНОВКА БИТА
! 2 БАЙТА, 5 МИКРОСЕКУНД	
! 2 БАЙТА, 1 МИКРОСЕКУНДА	

Пример 5.57. Сбросить флаг в РПД:

! ВЕРСИЯ ДЛЯ MK48	! ВЕРСИЯ ДЛЯ MK51
MOV R0, #FLGABR ;R0=АДРЕС ФЛАГА	FLAG BIT 20H.0 ;ОПРЕДЕЛЕНИЕ ФЛАГА
MOV A,R0 ;ЧТЕНИЕ ФЛАГА	CLR FLAG ;СБРОС ФЛАГА В РПД
ANL A,#FLGMASK ;СБРОС ФЛАГА	
MOV R0,A ;ЗАПИСЬ ФЛАГА	
! 16 БАЙТА, 15 МИКРОСЕКУНД	
! 12 БАЙТА, 1 МИКРОСЕКУНДА	

Пример 5.58. Перейти, если программный флаг F0 сброшен:

! ВЕРСИЯ ДЛЯ MK48	! ВЕРСИЯ ДЛЯ MK51
JFO x+4 ;ПОХОД, ЕСЛИ F0=1,	JNB PSW.5,ZERO
JMP ZERO ;ИНАЧЕ К ZERO	
! 14 БАЙТА, 10 МИКРОСЕКУНД	
! 13 БАЙТА, 2 МИКРОСЕКУНДА	

Пример 5.59. Проверить на нуль сигнал на линии 5 порта 1:

```
! ВЕРСИЯ ДЛЯ MK48          ! ВЕРСИЯ ДЛЯ MK51
IN A,P1    ; ВВОД ИЗ ПОРТА 1   JNB P1.5,ZERO; ПЕРЕХОД К ZERO,
CPL A      ; СИНВЕРСИЯ           ; ЕСЛИ P1.5=0
JBS ZERO   ; ПЕРЕХОД К ZERG,
; ЕСЛИ P1.5=0
:4 БАЙТ, 12.5 НИКРОСЕКУНА  :13 БАЙТ, 2     :СЕКУНДА
```

Пример 5.60. Ожидание заданного кода на входах порта, например кода ОАН на входах порта 1:

```
! ВЕРСИЯ ДЛЯ MK48          ! ВЕРСИЯ ДЛЯ MK51
WAIT: IN A,P1    ; ВВОД ИЗ ПОРТА 1   MOV A,0BAH  ; ЗАГРУЗКА ОАН
XRL A,0BAH  ; СРАВНЕНИЕ С ОАН      WAIT: CJNE A,P1,WAIT ; ОПЫСКАНИЕ
JNZ WAIT    ; ПЕРЕХОД В НАЧАЛО      ; ПРИХОДА
; ЕСЛИ НЕ РАВНО
:5 БАЙТ, 15 НИКРОСЕКУНА  ;5 БАЙТ, 3 НИКРОСЕКУНА
```

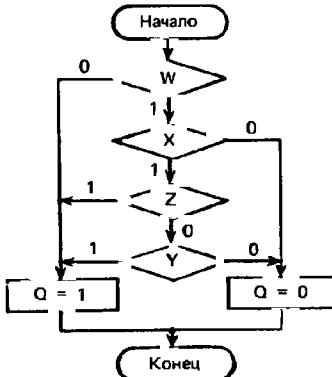


Рис. 5.3. Блок-схема программы реализации булевой функции

Пример 5.61. Вычислить булеву функцию $O = XA(Z \vee Y) \vee \overline{W}$. Переменные X, Y, Z и W поступают на входные линии порта 1 (биты 0–3 соответственно), функция выдается на линию 4 того же порта.

В MK48, который не имеет команд логической обработки бит (все битовые команды MK48 однооперандные), можно реализовать заданную логическую функцию, используя команды ветвления в соответствии со схемой, показанной на рис. 5.3:

```
! ВЕРСИЯ ДЛЯ MK48          ! ВЕРСИЯ ДЛЯ MK51
* ORL P1,00FH  ; НАСТРОЙКА ПОРТА 1  0 BIT P1.4  ; ОПРЕДЕЛЕНИЕ
* IN A,P1    ; ВВОД РЕПЕРЕМЕННЫХ  1 X BIT P1.0  ; СИНВОЛИЧЕСКИЙ
* JBS M1    ; НАТИ, ЕСЛИ W=1  1 Y BIT P1.1  ; ИМЕН
JMP ONE  ; НАТИ, ЕСЛИ W=0  1 Z BIT P1.2  ; БИТОВ
M1: JBO M2  ; НАТИ, ЕСЛИ X=1  1 W BIT P1.3
* JMP ZERO  ; НАТИ, ЕСЛИ X=0  1 MOV C,Z  ; ВВОД Z
M2: JB2 ONE  ; НАТИ, ЕСЛИ Z=1  1 ORL C,Y  ; Z OR Y
JB1 ONE  ; НАТИ, ЕСЛИ Y=1  1 ANL C,X  ; X AND (Z OR Y)
* ZERO:ANL P1,00FH  ; W=0  1 ORL C,W  ; (X AND (Z OR Y)) OR NOT(W)
* JMP EXIT  ; ВЫХОД  1 MOV W,C  ; ВВОД Q
ONE: ORL P1,010H  ; Q=1
EXIT: ...
:21 БАЙТ, МАКСИМУМ 35 НИКРОСЕКУНА
; КОМАНДЫ ВХОДЯЩИЕ В САМОУДАЛЯЮЩУЮ
; ОТВЕТВУ ПРОГРАММЫ, ПОМЕЧЕНЫ СИМВОЛОМ *
```

Из анализа примеров 5.53–5.60 видно, что программы реализаций типовых процедур обработки данных выполняются в MK51 в 4–15 раз быстрее и занимают в 1,5–3 раза меньше места в памяти, чем в MK48.

Организация взаимодействия микроконтроллера с объектом управления

6.1.

Ввод информации с датчиков

6.1.1. Опрос двоичного датчика. Ожидание события

В устройствах и системах логического управления объектами события в объекте управления фиксируются с использованием разнообразных датчиков цифрового и аналогового типов. Наиболее распространение имеют двоичные датчики типа да/нет, например концевые выключатели, которые подключаются к МК так, как показано на рис. 6.1.

Ожидание статического сигнала. Типовая процедура ожидания события (WAIT) состоит из следующих действий: ввода сигнала от датчика, анализа значения сигнала и передачи управления в зависимости от состояния датчика. На рис. 6.2 представлена блок-схема алгоритма процедуры ожидания события, фиксируемого замыканием контакта двоичного датчика. Конкретная программная реализация процедуры зависит не только от типа МК, но и от того, каким образом датчик подключен к МК. Он может быть подключен к одной из линий портов МК или к специальным тестируемым входам (T0, T1 для MK48).

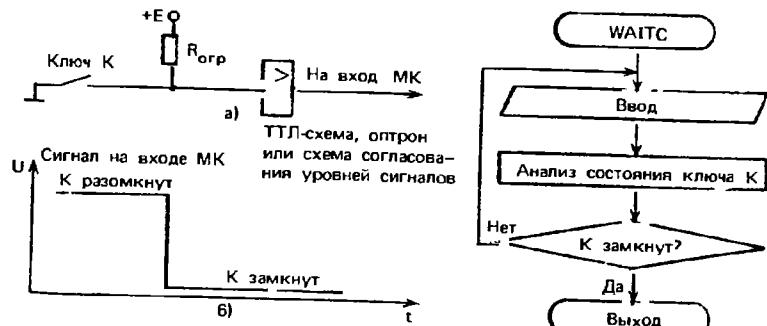


Рис. 6.1. Схема двоичного датчика (а) и сигнал на его выходе (б)

Рис. 6.2. Блок-схема процедуры ожидания события

Например, при подключении датчика к линии бита 3 порта 1 программа реализации процедуры ожидания замыкания контакта будет иметь вид

```
#ВЕРСИЯ ДЛЯ МК48
WAITC: IN A,P1      ;ВВОД СИГНАЛА ОТ ДАТЧИКА
JB3    WAITC        ;ЕСЛИ КОНТАКТ ДАТЧИКА РАЗОМКНУТ,
;   ТО ПОВТОРЯТЬ ВВОД;
;   ИНАЧЕ ВЫХОД ИЗ ПРОЦЕДУРЫ
...
```

```
#ВЕРСИЯ ДЛЯ МК51
WAITC: JNB P1,3,WAITC ;ОЖИДАНИЕ РАЗЫМКАНИЯ КОНТАКТА ДАТЧИКА
```

При подключении датчика к тестовому входу T0 микроконтроллера MK48 программа будет иметь вид

```
#ВЕРСИЯ ДЛЯ МК48
WAITC: JTO WAITC    ;ЕСЛИ КОНТАКТ РАЗОМКНУТ, ТО ЦИКЛ
```

Другим частным случаем типовой процедуры ожидания события является процедура ожидания размыкания контакта, которая может быть реализована следующим образом:

```
#ВЕРСИЯ ДЛЯ МК48
WAITC: IN A,P1      ;ВВОД БАЙТА
CPL   A             ;ИНВЕРТИРОВАНИЕ
JB3    WAITC        ;ЕСЛИ КОНТАКТ ЗАМКНУТ, ТО ЦИКЛ
```

```
#ВЕРСИЯ ДЛЯ МК51
WAITC: JB P1,3,WAITC ;ОЖИДАНИЕ ЗАМЫКАНИЯ КОНТАКТА ДАТЧИКА
```

Наравне с входами T0 и T1 для опроса датчика может использоваться и вход ЗПР. В этом случае надо предварительно запретить прерывания и использовать вход ЗПР как тестовый.

Режим прерывания целесообразно использовать только для опроса особо важных датчиков с целью уменьшения времени реакции на исключительную (аварийную) ситуацию в объекте управления.

Ожидание импульсного сигнала. Схема подключения датчика импульсного сигнала аналогична схеме на рис. 6.1. Особенность процедуры ожидания импульсного сигнала состоит в том, что МК должен обнаружить не только факт появления, но и факт окончания сигнала.

Для программирования этой процедуры удобно воспользоваться рассмотренными выше примерами ожидания события, смонтировав их последовательно в линейную программу. Оформлять процедуры WAITC и WAITO в виде подпрограмм нецелесообразно, так как это удлиняет программу, а длина и, следовательно, время исполнения программы определяют минимальную длительность импульса, который может быть обнаружен программой.

Последовательность схемирования процедур WAITC и WAITO зависит от формы импульса. Для "отрицательного" импульса ($1 \rightarrow 0 \rightarrow 1$) процедура WAITC предшествует процедуре WAITO, для "положительного" ($0 \rightarrow 1 \rightarrow 0$) следует за ней

Ниже приведены примеры программной реализации процедуры ожидания "отрицательного" импульсного сигнала при подключении датчика к биту 3 порта 1 при условии, что начальное состояние входа — единичное:

```
#ВЕРСИЯ ДЛЯ МК48
WAITC: IN A,P1      ;ВВОД БАЙТА
JB3    WAITC        ;ЕСЛИ P1,3=1, ТО ВВОД
WAITO: IN A,P1      ;ВВОД БАЙТА
CPL   A             ;ИНВЕРСИЯ
JB3    WAITO        ;ЕСЛИ P1,3=0, ТО ВВОД
```

```
#ВЕРСИЯ ДЛЯ МК51
WAITC: JB P1,3,WAITC ;ОЖИДАНИЕ P1,3=0
WAITO: JNB P1,3,WAITO ;ОЖИДАНИЕ P1,3=1
```

Аналогичным образом строится программа при подключении датчика импульсного "отрицательного" сигнала к тестовому входу MK48:

```
#ВЕРСИЯ ДЛЯ МК48
WAITC: JTO WAITC    ;ОЖИДАНИЕ ПОЯВЛЕНИЯ ИМПУЛЬСА
WAITO: JNTO WAITO   ;ОЖИДАНИЕ ОКОНЧАНИЯ ИМПУЛЬСА
```

Программная реализация цикла ожидания накладывает ограничения на длительность импульса: импульсы длительностью меньше времени выполнения цикла ожидания могут быть "не замечены" МК. Минимально допустимые длительности импульсов для различных способов подключения импульсного датчика к МК приведены в табл. 6.1

Для обнаружения кратковременных импульсов можно использовать способ фиксации импульса на внешнем триггере флага (рис. 6.3). На

Таблица 6.1. Минимально допустимые длительности импульсов для различных способов подключения датчика к МК

Способ подключения датчика к MK48/MK51	Минимально допустимая длительность импульса, мкс	
	отрицательного	положительного
P1, P2, BUS/P0	10/2	12,5/2
T0, T1	5/2	5/2
ЗПР	10/2	5/2

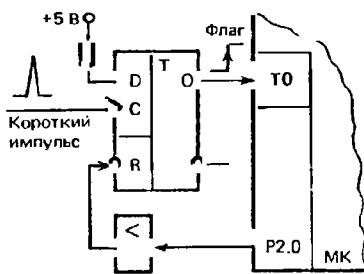


Рис. 6.3. Схема фиксации короткого импульса на триггер флага

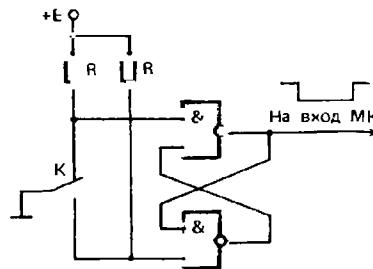


Рис. 6.4. Схема подавления дребезга контактов

вход МК в этом случае поступает не кратковременный сигнал с датчика, а флаг, формируемый триггером. Триггер устанавливается по фронту импульса, а сбрасывается программным путем — выдачей специального управляющего воздействия. Длительность импульса при этом будет ограничена снизу только быстродействием триггера.

Текст программы для МК48 приведен ниже:

```

FLAGIN:      ;ПРОЦЕДУРА ВВОДА ФЛАГА В МК48
    ANL     P2,#0FEN ;СЕРВОС ФЛАГА
    ORL     P2,#01H
    WAITC: JTO      MAITC ;ОЖИДАНИЕ ПРИХОДА ИМПУЛЬСА

```

6.1.2. Устранение дребезга контактов

При работе МК с датчиками, имеющими механические или электромеханические контакты (кнопки, клавиши, реле и клавиатуры), возникает явление, называемое дребезгом. Это явление заключается в том, что при замыкании контактов возможно появление отскока (BOUNCI) контактов, которое приводит к переходному процессу. При этом сигнал с контакта может быть прочитан МК как случайная последовательность нулей и единиц. Подавить это нежелательное явление можно схемотехническими средствами с использованием буферного триггера (рис. 6.4). Но чаще это делается программным путем.

Наибольшее распространение получили два программных способа ожидания установленвшегося значения: 1) подсчет заданного числа совпадающих значений сигнала; 2) временная задержка. Схемы процедур подавления помех от дребезга контактов (DEBOUNCE) при вводе сигнала 0 показаны на рис. 6.5. Суть первого способа состоит в многократном считывании сигнала с контакта. Подсчет удачных опросов (т.е. опросов, обнаруживших, что контакт устойчиво замкнут) ведется программным счетчиком. Если после серии удачных опросов встречается неудачный, то подсчет начинается сначала. Контакт считается устойчиво замкнутым (дребезг устранен), если последовало N удачных опросов. Число N под-

бирается экспериментально для каждого типа используемых датчиков и лежит в пределах от 5 до 50.

Пример программного подавления дребезга контакта приводится для случая, когда датчик импульсного сигнала подключен к входу T0, счет удачных опросов ведется в регистре R3, N = 20:

```

;ВЕРСИЯ ДЛЯ МК48
DBNC: MOV   R3,#20
DBNC1: JTO   DBNC
DJNZ  R3,DBNC1

```

ИНИЦИАЛИЗАЦИЯ СЧЕТЧИКА
;ЕСЛИ КОНТАКТ РАЗОМКНУТ, ТО НАЧАТЬ
; ; ОТСЧЕТ ОПРОСОВ СНАЧАЛА
;ДЕКРЕМЕНТИ СЧЕТЧИКА, И ЕСЛИ СОДЕЙСТВИЕ
; СЧЕТЧИКА НЕ РАВНО 0, ТО
; ПОВТОРИТЬ АНАЛИЗ СОСТОЯНИЯ КОНТАКТ

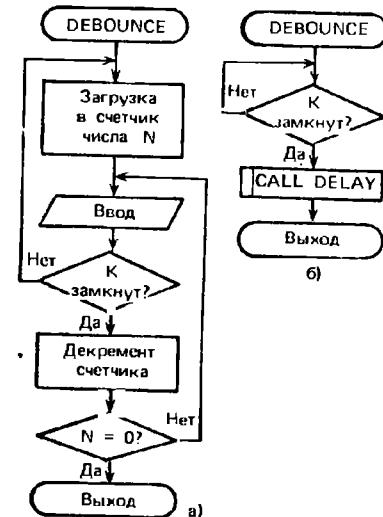
```

;ВЕРСИЯ ДЛЯ МК51
DBNC: MOV   R3,#20
DBNC1: JB    P3.4,DBNC
BJNZ  R3,DBNC1

```

ИНИЦИАЛИЗАЦИЯ СЧЕТЧИКА
;ЕСЛИ КОНТАКТ РАЗОМКНУТ, ТО
; НАЧАТЬ ОТСЧЕТ ОПРОСОВ СНАЧАЛА
;ПОВТОРИТЬ, ПОКА R3 НЕ СТАНЕТ РАВНЫМ 0

Рис. 6.5. Блок-схемы процедур подавления дребезга контактов путем многократного считывания (а) и с использованием временной задержки (б)



Устранение дребезга контакта путем введения временной задержки заключается в следующем. Программа, обнаружив замыкание контакта К, запрещает опрос состояния этого контакта на время, заведомо большее длительности переходного процесса. Программа, соответствующая БСА на рис. 6.5, б, написана для случая подключения датчика к входу T0 и программной реализацией временной задержки

```

;ВЕРСИЯ ДЛЯ МК48
DBNC0L: JTO   DBNC0L ;ОЖИДАНИЕ НУЖНА НА ВХОДЕ TO
    CALL  DELAY ;ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
EXIT: ...          ;ВЫХОД ИЗ ПРОЦЕДУРЫ

```

Временная задержка (в пределах 1–10 мс) подбирается экспериментально для каждого типа датчиков и реализуется подпрограммой **DELAY**.

6.1.3. Подсчет числа импульсов

Часто в управляющих программах возникает необходимость ожидания цепочки событий, представляемой последовательностью импульсных сигналов от датчиков. Рассмотрим две типовые процедуры: подсчет числа импульсов между двумя событиями и подсчет числа импульсов за заданный интервал времени.

Подсчет числа импульсов между двумя событиями. Эту типовую процедуру удобно проиллюстрировать на конкретном примере. Предположим, что необходимо подсчитать число деталей, сошедших с конвейера от момента его включения до момента выключения. Факт схода детали с конвейера фиксируется фотодиодом, на выходе которого формируется импульсный сигнал (рис. 6.6).

Для простоты реализации программы считаем, что общее количество деталей не превышает 99:

```
:ВЕРСИЯ ДЛЯ МК48
COUNT: CLR A      ;СБРОС СЧЕТЧИКА ДЕТАЛЕЙ
WAITC1: JTO WAITC1;ОЖИДАНИЕ ВКЛЮЧЕНИЯ КОНВЕЙЕРА
WAITC2: JTI WAITC2;ОЖИДАНИЕ НАЧАЛА ИМПУЛЬСА
WAITC02: JNTI WAITC02;ОЖИДАНИЕ КОНЦА ИМПУЛЬСА
INC A          ;ИНКРЕМЕНТ СЧЕТЧИКА ДЕТАЛЕЙ
DA A           ;ДЕСЯТИЧНАЯ КОРРЕКЦИЯ
JTO WAITC2    ;ЕСЛИ КОНВЕЙЕР НЕ ВЫКЛЮЧЕН, ТО ВРОДОЖДАТЬ
               ; ПОСЧЕТ, ИНАЧЕ ВЫХОД ИЗ ПРОЦЕДУРЫ
EXIT: ...       ;
```

По окончании выполнения процедуры в аккумуляторе фиксируется число деталей, представленное в двоично-десятичном коде.

Процедура подсчета импульсов может быть реализована иначе, если использовать вход **T1** не как тестовый, а как вход счетчика событий:

```
:ВЕРСИЯ ДЛЯ МК48
COUNT2: CLR A      ;СБРОС СЧЕТЧИКА
        MOV T,A
WAITC1: JTO WAITC1;ОЖИДАНИЕ ВКЛЮЧЕНИЯ КОНВЕЙЕРА
        STRT CNT    ;ЗАВУС СЧЕТЧИКА СОБЫТИЙ
WAITC01: JNTO WAITC01;ОЖИДАНИЕ ОТКЛЮЧЕНИЯ КОНВЕЙЕРА
        STOP TCNT   ;ОСТАНОВ СЧЕТЧИКА
        MOV A,T      ;ПЕРЕДАЧА СОДЕРЖИМОГО СЧЕТЧИКА
               ; В АККУМУЛЯТОР
```

В аккумуляторе фиксируется число деталей, представленное в двоичном коде (максимальное количество 255).

```
:ВЕРСИЯ ДЛЯ МК51
MOV TH0D,$01000000;ИАСТРОИКА СЧЕТЧИКА 1
MOV TH1,$0           ;СБРОС СЧЕТЧИКА ДЕТАЛЕЙ
WAIT0: JDI P3.4,WAIT0;ОЖИДАНИЕ ВКЛЮЧЕНИЯ КОНВЕЙЕРА
SETB TCON.6          ;ВЛУСК СЧЕТЧИКА 1
WAIT1: JNB P3.4,WAIT1;ОЖИДАНИЕ ВЫКЛЮЧЕНИЯ КОНВЕЙЕРА
CLR TCON.6          ;ОСТАНОВ СЧЕТЧИКА 1
MOV A,TH1            ;(АККУМУЛЯТОР) (-- ЧИСЛО ДЕТАЛЕЙ
                     ;ВЫХИ ИЗ ПРОЦЕДУРЫ
```

Подсчет числа импульсов за заданный промежуток времени. При решении задачи преобразования число-импульсного кода в двоичный код, а также в ряде других задач может возникнуть необходимость подсчета числа импульсов за заданный интервал времени. Эта процедура может быть реализована тремя различными способами:

программной реализацией временного интервала и программным подсчетом числа импульсов на входе МК;

программной реализацией временного интервала и аппаратурным подсчетом числа импульсов (на внутреннем таймере/счетчике);

аппаратурной реализацией временного интервала и программным подсчетом числа импульсов.

Для МК51, имеющего в своем составе два таймера/счетчика, возможен четвертый способ: аппаратурная реализация временного интервала с аппаратурным подсчетом числа импульсов.

Первый способ неэффективен и значительно сложнее других, а потому не рассматривается, второй и третий являются альтернативными, поскольку у МК48 имеется только один таймер/счетчик.

При аппаратурной реализации подсчета числа событий импульсный датчик должен быть подключен к входу **T1** микроконтроллера:

```
:ВЕРСИЯ ДЛЯ МК48
STUDY: CLR A      ;СБРОС СЧЕТЧИКА ИМПУЛЬСОВ
        MOV T,A
        STRT CNT    ;ЗАВУС СЧЕТЧИКА
        CALL DELAY  ;ВРЕМЕННАЯ ЗАДЕРЖКА
        STOP TCNT   ;ОСТАНОВ СЧЕТЧИКА
        MOV A,T      ;ФИКСАЦИЯ РЕЗУЛЬТАТА
```

Подсчет импульсов производится счетчиком событий, а отсчет заданного временного интервала – подпрограммой **DELAY**.

При аппаратурной реализации отсчета временного интервала импульсные сигналы удобнее всего принимать на вход **T0**:

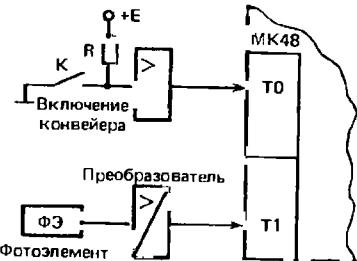


Рис. 6.6. Схема подключения датчиков микроконтроллеру MK48

```

:ВЕРСИЯ ДЛЯ МК48
СТДОУТ: MOV A,$0 :НАСТРОЙКА ТАЙМЕРА
MOV T,A
STRT
MOV R1,A :ЗАПУСК ТАЙМЕРА
MOV STP :СБРОС СЧЕТЧИКА ИМПУЛЬСОВ
WAITC: JTF STP :ЕСЛИ ВРЕМЕННОЙ ИНТЕРВАЛ ИСТЕК, ТО ИДТИ К STP
JTO WAITC :ОБНОВЛЕНИЕ ИМПУЛЬСА
WAITD: JTO WAITC
INC R1 :ИНКРЕМЕНТ СЧЕТЧИКА
JMP WAITC :ПЕРЕХОД ДЛЯ ПРОДОЛЖЕНИЯ СЧЕТА
БТР: STOP TCNT :ОСТАНОВ ТАЙМЕРА
EXIT: ... :ВЫХОД ИЗ ПРОЦЕДУРЫ

```

В приведенном примере таймер настроен на максимально возможный временной интервал – 20 мс, счетчик импульсов реализован в регистре R1. Проверка истечения заданного временного интервала осуществляется по флагу переполнения таймера (TF). Прерывание должно быть замаскировано.

Поскольку аппаратурный таймер не может реализовать временные задержки длительностью более 20 мс, "длинные" временные задержки должны реализовываться программно, например они могут "набираться" из интервалов в 20 мс с подсчетом числа прерываний от таймера.

Четвертый (полностью аппаратурный) способ подсчета числа импульсов требует использования двух аппаратурных счетчиков и поэтому возможен лишь для MK51. На T/C1 можно выполнять подсчет числа импульсов, а на T/C0 – отсчет заданного интервала. Датчик импульсов должен быть подключен к входу T1:

```

:ВЕРСИЯ ДЛЯ МК51
TIME EQU NOT(10000)+1 :ПОДСЧЕТНЫЕ КОНСТАНТЫ TIME ДЛЯ
                           :ОТСЧЕТА ИНТЕРВАЛА 10 МС
                           :НАСТРОЙКА Т/С1: 16 БИТ
                           : 1 – СЧЕТЧИК
                           : 0 – ТАЙМЕР
                           :СБРОС АККУМУЛЯТОРА
                           :СБРОС Т/С1
CLR A
MOV TM0,$01010001B :ЗАГРУЗКА В Т/С0
MOV TL1,A :КОНСТАНТЫ TIME
MOV TM0,$HIGH(TIME)
MOV TL0,$LOW(TIME)
DRL TCON,$50H :ЗАПУСК Т/С1 И Т/С0
                :ПУСК Т/С1 И Т/С0
WAIT: JBC TCON,5,EXIT :ПРОВЕРКА ПЕРЕПОЛНЕНИЯ Т/С0
SJMP WAIT :ЦИКЛ, ЕСЛИ TF=0
CYC: MOV B,T1 :СЧЕТЧИК (--- ЧИСЛО ИМПУЛЬСОВ ЗА 10 МС
MOV A,TL1
MOV ...
:ВЫХОД ИЗ ПРОЦЕДУРЫ

```

6.1.4. Опрос группы двоичных датчиков

Микроконтроллеры чаще всего имеют дело не с одним датчиком, как в рассмотренных выше примерах, а с группой автономных (логически независимых) или взаимосвязанных (формирующих двоичный код) датчиков (группу взаимосвязанных датчиков называют композицией). При этом МК может выполнять процедуру опроса датчиков и передачи управления отдельным фрагментам прикладной программы в зависимости от принятого кода.

Программную реализацию процедуры ожидания заданного кода (WTCODE) рассмотрим для случая подключения группы из восьми взаимосвязанных статических датчиков к входам порта 1 МК:

```

:ВЕРСИЯ ДЛЯ МК48
CODE EQU 10 :ОПРЕДЕЛЕНИЕ ЭТАЛОЧНОГО ЗНАЧЕНИЯ КОДА
WTCODE: IN A,P1 :ОПРОС ГРУППЫ ДАТЧИКОВ
XRL A,$CODE :СРАВНЕНИЕ ПРИНЯТОГО КОДА
               ; С ЗАДАННЫМ ЗНАЧЕНИЕМ CODE
               ; ЕСЛИ КОДЫ НЕ СОВПАДАЮТ, ТО ПОВТОРИТЬ ВРОД,
               ; ИНАЧЕ ВЫХОД ИЗ ПРОЦЕДУРЫ
               EXIT: ...

```

Сравнение принятого кода с заданным осуществляется операцией "исключающее ИЛИ". В приведенном примере число CODE равно 10.

```

:ВЕРСИЯ ДЛЯ МК51
WTCODE: MOV A,$10 :ЗАГРУЗКА В АККУМУЛЯТОР ЭТАЛОЧНОГО КОДА
WAIT: CJNE A,P1,WT1 :ЕСЛИ КОДОВАЯ КОМБИНАЦИЯ НА ВХОДАХ ПОРТА 1
      ; НЕ СОВПАДАЕТ С ЭТАЛОЧНЫМ ЗНАЧЕНИЕМ, ТО ИДТИ
      ; ВЫХОД ИЗ ПРОЦЕДУРЫ
      EXIT: ...

```

При опросе композиции двоичных датчиков передачу управления удобно осуществлять по таблице переходов. Ниже приводится текст программы, осуществляющей передачу управления одной из восьми прикладных программ PROGO–PROG7 (которые расположены в пределах одной страницы памяти программ) в зависимости от кодовой комбинации, набранной на переключателях, подключенных к входам P1.0 P1.2 MK48:

```

CODE: MOV RO,$LOW BASE :ЗАГРУЗКА В RO НАЧАЛЬНОГО АДРЕСА
      ; ТАБЛИЦЫ ПЕРЕХОДОВ
IN A,P1 :ВРОД БАНТА
ANL A,$00000011B :ВЫДЕЛЕНИЕ МАКСИМУМА БИТ
ABD A,RO :УФОРМИРОВАНИЕ АДРЕСА СТРОКИ
              ; В ТАБЛИЦЕ ПЕРЕХОДОВ
BASE: JMPP DB :ПЕРЕДАЧА УПРАВЛЕНИЯ
      DB LOW PROGO :ТАБЛИЦА ПЕРЕХОДОВ
      ...
      DB LOW PROG7

```

Программа обеспечивает опрос и выделение сигналов от трех датчиков путем маскирования старших бит аккумулятора.

Адрес строки таблицы, в которой хранится адреса переходов, вычисляется как сумма относительного (внутри текущей страницы РПП) начального адреса таблицы BASE и кода, принятого от датчиков. Команда JMPP @A, таблица BASE и программы PROGO–PROG7 должны располагаться на одной странице ПП.

При работе с композицией датчиков часто возникает необходимость осуществлять передачу управления не только в зависимости от двоичного эквивалента принятого кода, как в рассмотренном примере, но и в зависимости от соотношения принятого кода и некоторой заранее определенной уставки. Пусть, например, в порте I от группы двоичных датчиков формируется восьмибитный двоичный код. Если код равен

десятичному эквиваленту 135, то необходимо передать управление программе с меткой LABELA, в противном случае – программе с меткой LABELB:

```
;ВЕРСИЯ ДЛЯ МК48
IN    A,P1      ;ВВОД КОДА
ADD  A,$LOW(256 - 135) ;СРАВНИНИЕ С УСТАВКОЙ
JZ   LABELA    ;ПЕРЕЛАЧА УПРАВЛЕНИЯ
LABELB: ...
```

Так как в системе команд МК48 отсутствует команда сравнения, то сравнение с уставкой (уставками) осуществляется сложением кода с числом, дополняющим уставку до 256. Если число в точности равно уставке, то результат сложения равен нулю и управление передается по команде JZ или JNZ.

```
;ВЕРСИЯ ДЛЯ МК51
MOV  A,$135      ;ЗАГРУЗКА УСТАВКИ
CJNE A,P1,LABELD ;СРАВНИНИЕ И ПЕРЕЛАЧА УПРАВЛЕНИЯ
LABELD: ...
```

Опрос группы импульсных датчиков. Эта процедура состоит из последовательности действий: ожидания замыкания одного из контактов, устранения дребезга, ожидания размыкания замкнутого контакта.

Программная реализация процедуры для случая подключения четырех импульсных датчиков к входам 0–3 порта 1 будет иметь вид

```
;ВЕРСИЯ ДЛЯ МК48
KBRD: IN    A,P1      ;ВВОД КОДА
       CPL  A      ;ИНВЕРСИЯ КОДА
       ANL  A,$00001111B ;ЕСТЬ ЗАМКНУТЫЙ КОНТАКТ?
       JZ   KBRD    ;ЕСЛИ НИ ОДИН КОНТАКТ НЕ ЗАМКНУТ, ТО ЗДЕТЬ
       MOV  R2,A    ;ПЕРЕЛАЧА ПРИНЯТОГО КОДА В R2
       DBMC: CALL  DELAY ;УСТРАНЕНИЕ ДРЕБЕЗГА
       WAIT: IN    A,P1      ;ВВОД КОДА
       CPL  A      ;ИНВЕРСИЯ КОДА
       ANL  A,$00001111B ;ЕСТЬ ЗАМКНУТЫЙ КОНТАКТ?
       JNZ  WAIT    ;ЕСЛИ КОНТАКТ ЗАМКНУТ, ТО ЗДЕТЬ, ИНАЧЕ
       EXIT: ...
```

Анализ состояния контактов осуществляется наложением маски на принятый от датчиков код. Для датчиков, формирующих "отрицательный" импульс, принятый код удобно предварительно проинвертировать.

Для группы импульсных датчиков, представляющих собой клавишный регистр, процедура KBRD должна быть дополнена процедурами идентификации нажатой клавиши и защиты от одновременного нажатия двух и более клавиш.

Идентификация нажатой клавиши может осуществляться двумя способами: по таблице или программно. При табличном способе перекодирования в памяти программ должна находиться таблица двоичных эквивалентов кодов клавиши.

Программное преобразование унитарного кода, принятого от клавиатуры, в двоичный может быть выполнено методом сдвигов исходного

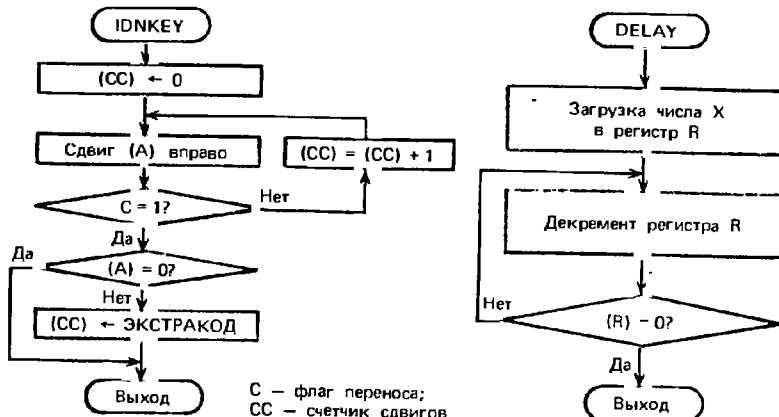


Рис. 6.7. Блок-схема процедуры преобразования кода

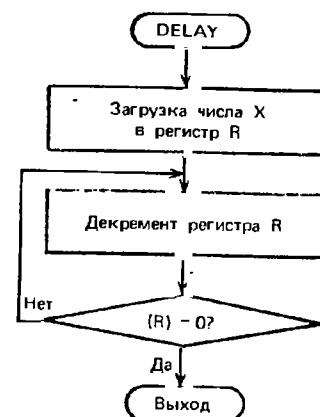


Рис. 6.8. Блок-схема процедуры формирования временной задержки малой длительности

унитарного кода и подсчетом числа сдвигов на счетчике до появления первого переноса. Схема алгоритма такой процедуры приведена на рис. 6.7, программа для МК48 будет иметь вид

IDNKEY: MOV	A,R2	:ПЕРЕЛАЧА ПРИНЯТОГО КОДА В АККУМУЛЯТОР
MOV	R1,\$0	:СБРОС СЧЕТЧИКА СДВИГОВ
CLR	C	:СБРОС ФЛАГА ПЕРЕНОСА
ROTATE: RRC	A	:СВИДГ УНИТАРНОГО КОДА
JC	CHECK	:ЕСЛИ ВОЗНИК ПЕРЕНОС, ТО ПРОВЕРКА
		И НА МНОПЛЕСТВЕННОЕ НАЖАТИЕ
INC	R1	:ИНКРЕМЕНТ СЧЕТЧИКА СДВИГОВ
JMP	ROTATE	:ПРОДОЛЖЕНИЕ СДВИГОВ
CHECK: JZ	EXIT	:ЕСЛИ (A)=0, ТО ВЫХОД ИЗ ПРОЦЕДУРЫ
MOV	R1,\$OFF	:ЗАНЕСЕНИЕ В R1 КОДА ОДНОВРЕМЕННОГО
		И НАЖАТИЯ НЕСКОЛЬКИХ КЛАВИШ
EXIT: ...		:ВЫХОД ИЗ ПРОЦЕДУРЫ

Результат формируется в регистре R1. В программе предполагается, что в R2 находится инверсия унитарного кода, принятого от группы датчиков процедурой KBRD. В результате работы программы IDNKEY в R1 будет сформирован двоичный код нажатой клавиши или экстракод (FFH) "нажато несколько клавиш".

6.2.

Вывод управляющих сигналов из МК

6.2.1. Формирование статических сигналов

Для управления исполнительным механизмом, работающим по принципу включено/выключено, на соответствующей выходной линии порта МК необходимо сформировать статический сигнал 0 или 1, что реализуется командами вывода непосредственного операнда, содержащего в требуемом бите значение 0 или 1.

В случае параллельного управления группой автономных исполнительных механизмов, подключенных к выходному порту, формируется не двоичное управляющее воздействие, а управляющее слово (УС), имеющее формат байта, каждому разряду которого ставится в соответствие 1 или 0 в зависимости от того, какие исполнительные механизмы должны быть включены, а какие выключены.

Управляющие слова удобно формировать командами логических операций над содержимым порта. Команда ANL используется для сброса тех бит УС, которые в операнде (маске) заданы нулем. Команда ORL используется для установки бит УС. Командой XRL осуществляется инверсия бит в соответствии с выражением $x \oplus 1 = \bar{x}$.

Для формирования сложных последовательностей УС удобно пользоваться табличным способом, при котором все возможные УС упакованы в таблицу, а прикладная программа МК вычисляет адрес требуемого УС, выбирает его из таблицы и передает в Порт вывода.

6.2.2. Формирование импульсных сигналов

Управляющее воздействие типа "импульс" можно получить последовательной выдачей сигналов *включить* и *отключить* с промежуточным вызовом подпрограммы временной задержки:

```
PULS:    R1.01110111B ; ВЫДАЧА ИМПУЛЬСА В ЛИНИИ З ПОРТА 1 МК48
ON:      ANL    R1.01110111B ; ВКЛЮЧЕНИЕ ИСПОЛНИТЕЛЬНОГО МЕХАНИЗМА
CALL     RELAY
OFF:     ORL    R1.00001000B ; ОТКЛЮЧЕНИЕ ИСПОЛНИТЕЛЬНОГО МЕХАНИЗМА
...

```

Длительность импульса определяется временной задержкой, реализуемой подпрограммой *DELAY*.

Генерация периодического управляющего воздействия (меандра). Для генерации меандрита удобно воспользоваться процедурой выдачи импульсного управляющего воздействия (PULS) и подпрограммой реализации временной задержки, равной половине периода сигнала (DLYX).

:ВЕРСИЯ МК МК48	:ВЕРСИЯ МК МК51
MEANDR:	MEANOR:
ON: ANL R1.00F7H	XCOR: CPL P1.3
CALL BLYX	ACALL DLYX
OFF: ORL R1.004H	SJMP XCOR
CALL DLX	
JMP ON	

Бесконечный периодический сигнал формируется в линии 3 порта 1; на остальных линиях порта 1 сигналы остаются неизменными.

Формирование апериодического управляющего сигнала. Последовательность импульсных сигналов с произвольной длительностью и скважностью может быть получена аналогичным образом, т.е. путем чередования процедур выдачи изменяющегося значения сигнала (0 или 1) и вызова подпрограмм временных задержек заданных длительностей.

6.3.

Масштабирование

При вводе и выводе информации часто возникает задача согласования диапазонов представления чисел в МК и во внешней аппаратуре, обеспечивающей связь МК с объектом управления. Эта задача называется масштабированием и сводится к операции умножения/деления числа на некоторую константу k .

Так как в системе команд МК48 отсутствуют команды умножения и деления, то масштабирование двоичных кодов, вводимых с цифровых датчиков для последующей обработки в МК или выводимых из МК на исполнительный механизм, нужно выполнять с использованием стандартных подпрограмм умножения и деления. Однако во многих применениях более эффективным (по быстродействию и длине программы) может оказаться способ умножения числа на константу путем сдвигов. Данный способ основан на том, что сдвиг двоичного кода числа на один бит влево (в сторону старших бит) эквивалентен его умножению на 2, а сдвиг на один бит вправо – делению на 2.

Поясним на двух простых примерах этот прием выполнения процедуры масштабирования. Пусть необходимо двоичный код X , полученный от цифрового датчика, умножить на константу 2,5. Результат масштабирования числа может быть получен в соответствии с выражением $2,5 X = 2X + X/2$, т.е. путем суммирования сдвинутых влево и вправо двоичных кодов числа X :

CLR C	; СБРОС ФЛАГА ПЕРЕНОСА
IN A, P1	; ВВОД ЧИСЛА X
MOV R2, A	; СОЗДАНИЕ КОПИИ X В R2
RL A	; УМНОЖЕНИЕ X НА 2
MOV R3, A	; СОЗДАНИЕ КОПИИ 2X В R3
MOV A, R2	; ВОССТАНОВЛЕНИЕ X В АККУМУЛЯТОРЕ
RRC A	; ДЕЛЕНИЕ X НА 2
ADD C, R3	; РЕЗУЛЬТАТ В АККУМУЛЯТОРЕ

Пусть требуется сформированное в аккумуляторе МК управляющее слово Y перед выдачей его в ЦАП умножить (с целью согласования масштабов УС и его аналогового эквивалента) на константу 17. Операция умножения на 17 может быть (в соответствии с выражением $17Y = 16Y + Y$) заменена четырьмя операциями сдвига и сложением:

MOV R3, A	; СОЗДАНИЕ КОПИИ Y В R3
RL A	; (A) <- 2Y
RL A	; (A) <- 4Y
RL A	; (A) <- 8Y
RL A	; (A) <- 16Y
ADD A, R3	; (A) <- 16Y+Y

Для упрощения примеров программ они составлены в предположении, что при умножении не происходит переноса одного байта (результат меньше 255).

При использовании МК51 проблема масштабирования снимается, так как имеются быстрые команды умножения и деления байтов.

6.4.

Реализация функций времени

6.4.1. Программное формирование временной задержки

Временная задержка малой длительности. Процедура реализации временной задержки использует метод программных циклов. При этом в некоторый рабочий регистр загружается число, которое затем в каждом проходе цикла уменьшается на 1. Так продолжается до тех пор, пока содержимое рабочего регистра не станет равным нулю, что интерпретируется программой как момент выхода из цикла. Время задержки при этом определяется числом, загруженным в рабочий регистр, и временем выполнения команд, образующих программный цикл. Схема алгоритма такой программы показана на рис. 6.8. Программа имеет символическое имя *DELAY*.

Предположим, что в управляющей программе необходимо реализовать временную задержку 100 мкс. Фрагмент программы, реализующей временную задержку, требуется оформить в виде подпрограммы, так как предполагается, что основная управляющая программа будет производить к ней многократные обращения для формирования выходных импульсных сигналов, длительность которых кратна 100 мкс:

```
;ВЕРСИЯ ДЛЯ МК48
DELAY: MOV R2+X ;(R2) <-- X
        COUNT: DJNZ R2,COUNT;ДЕКРЕМЕНТ R2 И ЦИКЛ, ЕСЛИ НЕ НУЛЬ
        RET      ;ВОЗВРАТ
```

Для получения требуемой временной задержки необходимо определить число *X*, загружаемое в рабочий регистр. Определение числа *X* выполняется на основе расчета времени выполнения команд, образующих данную подпрограмму. При этом необходимо учитывать, что команды *MOV* и *RET* выполняются однократно, а число повторений команды *DJNZ* равно числу *X*. Кроме того, обращение к подпрограмме временной задержки осуществляется по команде *CALL DELAY*, время исполнения которой также необходимо учитывать при подсчете временной задержки. В описании команд МК указывается, за сколько машинных циклов (МЦ) исполняется каждая команда. На основании этих данных определяется суммарное число машинных циклов в подпрограмме: *CALL* – 2 МЦ, *MOV* – 2 МЦ, *DJNZ* – 2 МЦ, *RET* – 2 МЦ.

При тактовой частоте 6 МГц каждый машинный цикл выполняется за 2,5 мкс. Таким образом, подпрограмма выполняется за время $5 + 5 + 5X + 5 = 15 + 5X$ мкс. Для реализации временной задержки 100 мкс число *X* = $(100 - 15)/5 = 17$.

В данном случае при загрузке в регистр R2 числа 17 требуемая временная задержка (100 мкс) реализуется точно. Если число *X* получается дробным, то временную задержку можно реализовать лишь приблизительно. Для более точной подстройки в подпрограмму могут быть

включены команды NOP, время выполнения каждой из которых равно 2,5 мкс.

Минимальная временная задержка, реализуемая подпрограммой *DELAY*, составляет 20 мкс (*X* = 1). Временную задержку меньшей длительности программным путем можно реализовать, включая в программу цепочки команд NOP.

Максимальная длительность задержки, реализуемая подпрограммой *DELAY*, составляет 1,29 мс (*X* = 255).

Для реализации задержки большей длительности можно рекомендовать увеличить тело цикла включением дополнительных команд или использовать метод вложенных циклов. Так, например, если в подпрограмму *DELAY* перед командой *DJNZ* вставить дополнительно две команды NOP, то максимальная задержка составит $15 + X(5 + 5) = 15 + 10 \times 255 = 2565$ мкс (т.е. почти в 2 раза больше).

Временная задержка большой длительности. Схема алгоритма программной реализации временной задержки большой длительности методом вложенных циклов показана на рис. 6.9. Там же обозначено, сколько раз выполняется каждый фрагмент программы. Числа *X* и *Y* выбираются из соотношения $T = 5 + 5 + X(5 + 5Y + 5) + 5$, где *T* – реализуемый временной интервал в микросекундах. Максимальный временной интервал, реализуемый таким способом, при *X* = *Y* = 255 составляет 327,69 мс, т.е. приблизительно 0,3 с.

В качестве примера рассмотрим подпрограмму, реализующую временную задержку 100 мс:

```
;ВЕРСИЯ ДЛЯ МК48
DELAY: MOV R1,$84    ;ЗАГРУЗКА X
        LOOPX: MOV R2,$236   ;ЗАГРУЗКА Y
        LOOPIN: R2,LOOPIN  ;ДЕКРЕМЕНТ R2 И ВНУТРЕННИЙ ЦИКЛ
                ;ЕСЛИ (R2) НЕ РАВНО НУЛЮ
                ;ДЕКРЕМЕНТ R1 И ВНЕШНИЙ ЦИКЛ
        DJNZ R1,LOOPEX   ;ДЕКРЕМЕНТ R1 И ВНЕШНИЙ ЦИКЛ
                ;ЕСЛИ (R1) НЕ РАВНО НУЛЮ
        MOV R3,$4        ;ТОЧНАЯ ПОДСТРОЙКА
        LOOPAD: R3,LOOPAD ;ВРЕМЕННАЯ ЗАДЕРЖКА
        RET
```

Здесь два вложенных цикла реализуют временную задержку длительностью $15 + 84(10 + 5 \times 236) = 99\ 975$ мкс, а дополнительный цикл *LOOPAD* реализует задержку 25 мкс и тем самым обеспечивает точную подстройку временного интервала.

Временная задержка длительностью 1 с. Из рассмотренного примера видно, что секунда является очень большим интервалом времени по сравнению с частотой тактирования МК. Такие задержки сложно реализовать методом вложенных циклов, поэтому их обычно набирают из точно подстроенных задержек меньшей длительности. Например, задержку в 1 с можно реализовать десятикратным вызовом подпрограммы, реализующей задержку 100 мс:

```
ONESEC: MOV R3,$10 ;ЗАГРУЗКА В R3 ЧИСЛА ВЫЗОВОВ ПОДПРОГРАММЫ DELAY
LOOP: CALL DELAY ;ЗАДЕРЖКА 100 МС
      F3,LOOP ;ДЕКРЕМЕНТ R3 И ЦИКЛ, ЕСЛИ R3 НЕ РАВНО 0
```

Погрешность подпрограммы составляет 55 мкс. Для очень многих применений это достаточно высокая точность, хотя реализованные на основе этой программы часы астрономического времени за сутки "убегут" примерно на 5 с.

6.4.2. Формирование временной задержки на основе таймеров

Задержка малой длительности. Недостатком программного способа реализации временной задержки является нерациональное использование ресурсов МК: во время формирования задержки МК практически простаивает, так как не может решать никаких задач управления объектом. В то же время аппаратурные средства МК позволяют реализовать временные задержки на фоне основной программы работы.

При использовании таймера в МК48 можно получить временные задержки длительностью от 80 мкс до 20 мс.

Например, для реализации временной задержки 240 мкс необходимо выполнить следующие действия:

```

MOV A, #00T (240/80-1) ;ЗАГРУЗКА ТАЙМЕРА
MOV T,A                 ;ЗАПУСК ТАЙМЕРА
STRT T                  ;РАЗРЕШЕНИЕ ПРЕРЫВАНИЯ
EN TCNTI
    
```

Появление сигнала прерывания от таймера соответствует истечению временного интервала 240 мкс. Погрешность будет составлять 7,5 мкс (время выполнения команды передачи управления по вектору прерывания и команды STOP TCNT).

В МК51 на вход таймера/счетчика (T/C) могут поступать сигналы синхронизации с частотой 1 МГц (T/C в режиме таймера) или сигналы от внешнего источника (T/C в режиме счетчика). Оба эти режима могут быть использованы для формирования задержек. Если использовать T/C в режиме таймера полного формата (16 бит), то можно получить задержки в диапазоне 1–65 536 мкс.

В качестве примера рассмотрим организацию временной задержки длительностью 50 мс в МК51. Предполагается, что бит IE.7 установлен.

```

Организация перехода к метке NEXT при переполнении T/C
ORG ORB
CLR TCON.4
RETI

ORG 100H
MOV TMOD, #01H
MOV TL0, #LOW(NOT(50000-1))
MOV TH0, #HIGH(NOT(50000-1))
SETB TCON.4
SETB IE.1
SETB PCON.0

NEXT: ...
    
```

ПОДРОБНОСТИ ПРИ ОПИСАНИИ ПРОГРАММЫ

- ORG ORB: АДРЕС ВЕКТОРА ПРЕРЫВАНИЯ ОТ T/C
- CLR TCON.4: ОСТАНОВ Т/С
- RETI: ВЫХОД ИЗ ПРОГРАММЫ
- ORG 100H: ОБРАБОТКА ПРЕРЫВАНИЯ
- MOV TMOD, #01H: НАСТРОЙКА Т/С
- MOV TL0, #LOW(NOT(50000-1)): ЗАГРУЗКА ТАЙМЕРА
- MOV TH0, #HIGH(NOT(50000-1)): СТАРТ Т/С
- SETB TCON.4: РАЗРЕШЕНИЕ ПРЕРЫВАНИЯ ОТ Т/С
- SETB IE.1: ПЕРЕВОД МК51 В РЕЖИМ ХОЛОДОГО ХОДА
- SETB PCON.0: ПЕРЕВОД МК51 В РЕЖИМ ХОЛОДОГО ХОДА

6.4.3. Измерение временных интервалов

В задачах управления часто возникает необходимость измерения промежутка времени между двуми событиями. Схема алгоритма типовой процедуры измерения (MEASURE) приведена на рис. 6.10. Обычно события в объекте управления представляются сигналами от двоичных датчиков. Считая событиями фронт и спад импульса, можно определять временные характеристики импульсных сигналов: длительность, период и скважность. Кроме того, с помощью процедуры MEASURF можно определять скорость перемещения подвижного органа объекта по эталонному (заданной длины) участку. Начало и конец участка должны быть снабжены датчиками (концевыми выключателями).

Простейшим способом измерения длительности импульса является программный. Для обнаружения событий (фронт и спад импульсного сигнала) в этом случае используются типовые процедуры WAIT, а отсчет времени ведется программным способом. Для "положительного" импульсного сигнала, поступающего на вход T0, программа измерения его длительности будет иметь вид

ФУНКЦИЯ ДЛЯ МК48	
MSCONT: MOV R7, #00	СБРОС СЧЕТЧИКА
WAIT0: JNTO WAIT0	ПОЛЯРИЗАЦИЯ ФРОНТА СИГНАЛА
COUNT: INC R7	ИНКРЕМЕНТ СЧЕТЧИКА
JTO COUNT	ПОЛЯРИЗАЦИЯ СПАДА СИГНАЛА
EXIT: ...	ВЫХОД ИЗ ПРОЦЕДУРЫ

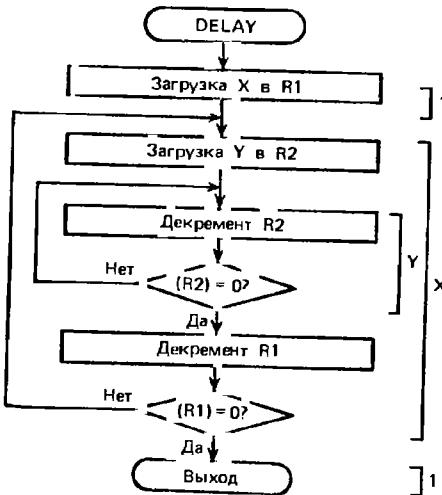


Рис. 6.9. Блок-схема процедуры временной задержки большой длительности

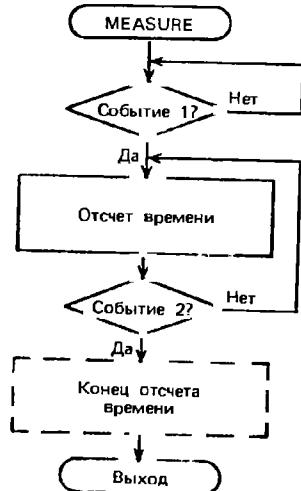


Рис. 6.10. Блок-схема процедуры измерения времени

После выхода из процедуры содержимое счетчика (R7) пропорционально длительности импульса.

Для нормальной работы этой программы необходимо, чтобы обращение к ней производилось в моменты, когда на входе T0 присутствует сигнал нулевого уровня. Верхний предел измеряемой длительности "положительного" импульса составит $255(1+2)2,5 \text{ мкс} = 8,925 \text{ мс}$. Этот предел может быть увеличен включением в цикл COUNT дополнительных команд NOP. Максимальная погрешность измерений 7,5 мкс.

Для измерения длительности сигнала может быть использован таймер. Особенно эффективно использование для этой цели таймера в MK51, имеющего вход разрешения счета (альтернативная функция входа ЗПР). Измеряемый сигнал можно, например, подавать на вход ЗПР, измерение длительности при этом будет выполняться в T/C0. Программа измерения длительности "положительного" импульса будет выглядеть так:

```
VERGСИЯ АЛЯ MK51
MOV TH0D, #00001001B : НАСТРОЙКА Т/С0
MOV TH0, $0 : СВРОС ТАЙМЕРА
MOV TL0, $0
SETB TCON.4 : СТАРТ Т/С0
WAIT0: JNB P3.2, WAIT0 : ОЖИДАНИЕ "1"
WAITC: JB P3.2, WAITC : ОЖИДАНИЕ "0"
CLR TCON.4 : СТОП Т/С0
EXIT: ... : ВЫХОД ИЗ ПРОЦЕДУРЫ
```

Управление программе должно быть передано при условии, что на входе ЗПР присутствует низкий уровень. Прерывания от T/C0 и внешнее прерывание по входу ЗПР должны быть запрещены. По завершению программы в T/C0 будет находиться число, пропорциональное длительности "положительного" импульса на входе ЗПР. Верхний предел измерения равен 65 536 мкс, а максимальная погрешность 1 мкс.

При необходимости измерения временных интервалов большей длительности можно программным способом подсчитывать число переполнений от таймера, т.е. расширять его разрядность за счет рабочего регистра или ячейки РПД.

6.5.

Преобразование кодов

В задачах управления может возникнуть необходимость преобразования информации из одной формы представления в другую. Это связано с тем, что обработка данных в МК осуществляется в параллельном двоичном коде, а поступать в МК и выводиться из него информация может в иной форме представления.

Наиболее распространены в задачах логического управления следующие преобразования: из унитарного кода в двоичный (при вводе информации с клавиатуры или от оцифрованных переключателей) и из двоичного в унитарный; из одной системы счисления в другую (при работе со специальными клавиатурами); специальные преобразования (для о

ганизации индикации и при выводе информации на периферийные устройства); из последовательного кода в параллельный (при вводе) и из параллельного в последовательный (при выводе); из аналоговой формы представления в цифровую и наоборот (шия связи с аналоговыми датчиками и исполнительными механизмами).

6.5.1. Простейшие преобразования

Преобразование унитарного кода в двоичный позиционный. Двоичный эквивалент унитарного кода равен номеру бита, в котором находится единственная единица. Поэтому преобразование унитарного кода в двоичный позиционный удобно осуществлять путем сдвига исходного унитарного кода в сторону младших бит с одновременным подсчетом числа сдвигов. При "выдвигании" из младшего бита значения 1 сдвиги прекращаются, а в счетчике сдвигов будет содержаться двоичный эквивалент унитарного кода (см. 6.1.4).

Преобразование двоичного позиционного кода в унитарный. Необходимость такого преобразования возникает, например, при выборке одного из исполнительных устройств, мультиплексирующих шину BUS (P0) по его номеру.

Пусть, например, требуется преобразовать 3-битный двоичный код из регистра R4 в 8-битный унитарный:

VERGСИЯ	АЛЯ	MK48
DECODE:	CLR	A : СБРОС АККУМУЛЯТОРА
	CLR	C : СБРОС ФЛАГА ПЕРЕНОСА
	CPL	C : УСТАНОВКА ФЛАГА ПЕРЕНОСА
ROTATE:	RLC	A : СВАЙП УНИТАРНОГО КОДА
	DJNZ	R4, ROTATE : ПЛЕКР И ЦИКЛ, ПОКА НЕ НУЛЬ

Преобразование осуществляется путем "вдвигания" единицы из флага переноса. Результат формируется в аккумуляторе за N сдвигов (где N – эквивалент исходного двоичного кода). Недостатком приведенной программы является то, что для преобразования N = 0 понадобится 256 сдвигов. Избежать этого можно, модифицировав программу DECODE следующим образом:

DECODE:	INC	R4	: ИНКРЕМЕНТ ИСХОДНОГО КОДА
	MOV	A, #00H	: УСТАНОВКА СТАРШЕГО БИТА АККУМУЛЯТОРА
ROTATE:	RL	A	: СВАЙП УНИТАРНОГО КОДА
	DJNZ	R4, ROTATE	: ИНКРЕМЕНТ ПОЗИЦИОННОГО КОДА, ; И ЦИКЛ, ПОКА НЕ 0

Преобразование в этом случае осуществляется за N + 1 сдвигов. Преобразование кодов из одной системы счисления в другую. Преобразование кода из одной позиционной системы счисления в другую осуществляется делением исходного числа на основание новой системы счисления. При этом деление должно выполняться по правилам исходной системы счисления. Например, для преобразования двоичного числа в двоично-десятичное исходное двоичное число должно быть поделено

на 10. Деление должно осуществляться по правилам двоичной арифметики.

Пусть требуется выполнить преобразования 8-битного двоичного числа в двоично-десятичное. Исходный двоичный код хранится в аккумуляторе. Результат преобразования состоит из 12 бит; младшие 4 бита — единицы, представляют собой остаток от деления исходного числа на 10; следующие 4 бита — десятки, представляют собой остаток от деления на 10 полученного частного; старшие 4 бита — сотни, являются частным от второго деления:

```
:ВЕРСИЯ ДЛЯ МК48
BBD: CALL DIV10 ;ДЕЛЕНИЕ ИСХОДНОГО КОДА НА 10
MOV R7,A ;СОХРАНЕНИЕ ОСТАТКА В R7
MOV A,R1 ;ЗАГРУЗКА В АККУМУЛЯТОР ЧАСТНОГО
CALL DIV10 ;ДЕЛЕНИЕ ЧАСТНОГО НА 10
SWAP A ;ПЕРЕДАЧА ОСТАТКА СТАРШИЙ
ORL A,R7 ;ПЕРЕДАЧА R7 В МЛАДШИЙ ТЕТРАДУ
;АККУМУЛЯТОРА
JMP EXIT ;ВЫХОД ИЗ ПРОЦЕДУРЫ

;ПОДПРОГРАММА ДЕЛЕНИЯ НА 10
;ИСХОДНЫЙ ДВОИЧНЫЙ КОД В АККУМУЛЯТОРЕ
;РЕЗУЛЬТАТ: В R1- ЧАСТНОЕ - В АККУМУЛЯТОРЕ - ОСТАТOK
DIV10: MOV R1,$0 ;СБРОС R1
SUB10: ADD A,$(NOT(10)+1) ;ВЫЧИТАНИЕ 10 ИЗ ИХЛМОГО
INC R1 ;ИНКРЕМЕНТ ЧАСТНОГО
JC SUB10 ;ЦИКЛ, ЕСЛИ ОСТАТОК !=0
BEC RI ;ВОССТАНОВЛЕНИЕ ЧАСТНОГО
ADD A,$10 ;ВОССТАНОВЛЕНИЕ ОСТАТКА
RET ;ВОЗВРАТ
EXIT: ...
```

В результате выполнения процедуры в младшей тетраде R1 хранятся единицы, в аккумуляторе — десятки и единицы двоично-десятичного эквивалента исходного двоичного числа.

Обратное преобразование (из двоично-десятичного кода в двоичный) осуществляется делением исходного числа на 16 по правилам десятичной арифметики. Программа преобразования приведена ниже. Исходный двоично-десятичный код хранится в аккумуляторе. После окончания работы программы в А находится двоичный эквивалент исходного двоично-десятичного числа.

Программа универсальна и может выполняться в МК48 и в МК51, так как команда DIV AB (МК51) не может быть использована для деления чисел по правилам десятичной арифметики:

```
BDB: CALL DIV16 ;ДЕЛЕНИЕ ИСХОДНОГО КОДА НА 16
SWAP A ;ПЕРЕДАЧА ОСТАТКА В СТАРШУЮ ТЕТРАДУ АККУМУЛЯТОРА
ORL A,R1 ;ЗАГРУЗКА ЧАСТНОГО В МЛАДШУЮ ТЕТРАДУ АККУМУЛЯТОРА
SWAP A ;ФОРМИРОВАНИЕ РЕЗУЛЬТАТА В АККУМУЛЯТОРЕ
JMP EXIT ;ВЫХОД ИЗ ПРОЦЕДУРЫ ПРЕОБРАЗОВАНИЯ

;ПОДПРОГРАММА ДЕСЯТИЧНОГО ДЕЛЕНИЯ НА 16
;ДЕЛИМОЕ - В АККУМУЛЯТОРЕ
;РЕЗУЛЬТАТ: ЧАСТНОЕ - В R1, ОСТАТOK - В АККУМУЛЯТОРЕ
DIV16: MOV R1,$0 ;СБРОС ЧАСТНОГО
SUD16: ABD A,$0AH ;ВЫЧИТАНИЕ ИЗ ДЕЛИМОГО ЧИСЛА 16,
;ПРЕСТАВЛЕННОГО В ДЕСЯТИЧНОЙ СИСТЕМЕ
DA A ;КОРРЕКЦИЯ
INC R1 ;ИНКРЕМЕНТ ЧАСТНОГО
JC SUB16 ;ЦИКЛ, ЕСЛИ ОСТАТОК >=0
DEC R1 ;ВОССТАНОВЛЕНИЕ ЧАСТНОГО
ADD A,$16H ;ВОССТАНОВЛЕНИЕ ОСТАТКА
DA A ;КОРРЕКЦИЯ
RET ;ВОЗВРАТ
EXIT: ...
```

Кроме рассмотренного способа преобразования чисел из одной системы счисления в другую можно воспользоваться более медленным, но зато и более простым способом "двух счетчиков". При этом способе из исходного кода вычитается, а к новому коду прибавляется по единице до обнуления исходного кода, причем вычитание осуществляется "в старой", а прибавление — в "новой" системе счисления. Пример программы преобразования двоичного числа в двоично-десятичное методом двух счетчиков приводится ниже. Для упрощения программы принято, что исходное двоичное число, заданное в А, не превышает его десятичного эквивалента 99:

```
:ВЕРСИЯ ДЛЯ МК48
BBD: MOV R2,A ;ПЕРЕДАЧА ИСХОДНОГО КОДА В R2
CLR A ;СБРОС АККУМУЛЯТОРА
CONV: ADD A,$1 ;ИНКРЕМЕНТ "НОВОГО" КОДА
DA A ;КОРРЕКЦИЯ
DJNZ R2,CONV ;ИНКРЕМЕНТ ИСХОДНОГО КОДА И ЦИКЛ, ЕСЛИ НЕ НУЛЬ
EXIT: .. ;ВЫХОД ИЗ ПРОЦЕДУРЫ
```

После выхода из процедуры в А находится двоично-десятичный эквивалент исходного двоичного кода.

6.5.2. Преобразования параллельных и последовательных кодов

Преобразование данных из параллельного кода в последовательный. Наиболее важным применением процедур преобразования формы представления линий "параллельный/последовательный" является связь с Удаленными датчиками, исполнительными механизмами и другими МК по однопроводным линиям передачи информации. Обычно при передаче байта данных в прямом последовательном коде для обеспечения согласования работы приемника и передатчика используют старт-стопный (асинхронный) режим обмена. Передача последовательного кода байта

предваряется посылкой старт-бита (0) и завершается выдачей стоп-бита (1).

Процедура выдачи одного бита последовательного кода реализуется в линию передачи статического сигнала 0 или 1 и вызовом подпрограммы временной задержки заданной длительности:

```

;ВЕРСИЯ ДЛЯ МК48
ANL P1, #0FH    ;ВЫДАЧА СТАРТ-БИТА
CALL DELAYT   ;ВРЕМЕННАЯ ЗАДЕРЖКА T
PSCONV: MOV R7, #8 ;ИНИЦИАЛИЗАЦИЯ СЧЕТЧИКА БИТ
ROTATE: RRC A   ;СВИД В ПРАВО, Т.Е. (C) -- А0
JC OFF        ;ЕСЛИ ПЕРЕНОС, ТО УПРАВЛЕНИЕ ПЕРЕХОДАЕТСЯ
               ;  ПРОЦЕДУРЕ ВЫДАЧИ 1
ON: ANL P1, #11110111B ;ВЫДАЧА НА Р1.3 СИГНАЛА 0
JMP DLY      ;ВЫДАЧА НА Р1.3 СИГНАЛА 1
OFF: ORL P1, #00001000B ;ВРЕМЕННАЯ ЗАДЕРЖКА T
JMP DLY      ;ВРЕМЕННАЯ ЗАДЕРЖКА T
DLY: CALL DELAYT   ;ВРЕМЕННАЯ ЗАДЕРЖКА T
DJNZ R7, ROTATE ;ДЛЯ ПОДДЕРЖКИ ЦИКЛА,
               ;  ЕСЛИ НЕ НУЛЬ
ORL P1, #0BH    ;ВЫДАЧА СТОП-БИТА
CALL DELAYT   ;ВРЕМЕННАЯ ЗАДЕРЖКА T
EXIT: ...      ;ВЫХОД ИЗ ПРОЦЕДУРЫ

```

В MK51 имеются средства аппаратурного преобразования параллельного кода в последовательный с использованием УАПП. Вся программа преобразования параллельного кода в последовательный сводится при этом к одной команде передачи байта в буфер УАПП: MOV SBUF,A.

Преобразование данных из последовательного кода в параллельный. Эта процедура является составной частью процедуры приема информации в последовательном коде. При обмене информацией в последовательном коде необходима предварительная настройка приемника на начальную слова. При асинхронном обмене процедура настройки сводится к ожиданию старт-бита.

После обнаружения старт-бита на входе приемника начинается преобразование кода, совмещенное с процедурой ввода последовательного кода. Схема алгоритма процедуры SPCONV, реализуемой в MK48, представлена на рис. 6.11. MK48 обеспечивает прием бита последовательного кода и его распознавание (конкретный вид этого блока зависит от способа представления бита информации в последовательном коде). При длине слова не более 8 бит для формирования параллельного кода удобно воспользоваться аккумулятором; тогда дешифрированный бит "вдвигается" в аккумулятор через флаг переноса. Поскольку последовательный код принимается младшими битами вперед, сдвиг параллельного кода осуществляется вправо. Аккумулятор предварительно сбрасывается, а после N сдвигов (где N – разрядность слова, передаваемого в последовательном коде) в нем фиксируется параллельный код принятого слова данных.

Для обеспечения временного согласования работы приемника и передатчика, а также для снижения вероятности неправильного прочтения сигнала при вводе стартовый бит после его обнаружения необходимо пристройовать в середине интервала его представления. С этой целью про-

грамма повторяет анализ старт-бита через время, равное половине периода бита, и если старт-бит не подтвердился, то процедура ожидания повторяется.

В MK48 для приема последовательного кода удобно использовать тестовый вход TO. Опрос входа TO повторяется через время T, равное периоду представления бита и смещено на полпериода относительно его начала. Для синхронизации приема/передачи подпрограмма DELAYT приемника должна реализовать ту же самую задержку, что и подпрограмма DELAYT передатчика:

```

;ВЕРСИЯ ДЛЯ МК48
WAIT: JTO WAIT ;ОЖИДАНИЕ СТАРТ-БИТА
      CALL DELAYT ;ЗАДЕРЖКА T/2
      JTO WAIT ;ЕСЛИ TO=1, ТО КОТОРИТЬ
      SPCONV: MOV R7, #8 ;ЗАГРУЗКА СЧЕТЧИКА БИТ, И=8
      CLR A ;СБРОС АККУМУЛЯТОРА
      LOOP: CLR C ;СБРОС ФЛГА ПЕРЕНОСА
              CALL DELAYT ;ЗАДЕРЖКА НА ОДИН Г
              JNTO ROTATE ;ЕСЛИ TO=0, ТВ СОХРАНЕНИЕ НУЛЕВОГО
                           ;  СОСТОЯНИЯ ФЛГА ПЕРЕНОСА,
                           ;  УСТАНОВКА ФЛГА ПЕРЕНОСА
              CPL C ;УСТАНОВКА ФЛГА ПЕРЕНОСА
              ROTATE: RRC A ;СВИД КАРАЛЛЕЛЬНОГО КОДА
              DJNZ R7, LOOP ;ДЛЯ ПОДДЕРЖКИ СЧЕТЧИКА БИТ,
                           ;  И ЕСЛИ НЕ НУЛЬ, ТО ВИДА
                           ;  ВЫХОД ИЗ ПРОЦЕДУРЫ
EXIT: ...

```

Поскольку в системе команд MK48 отсутствует команда установки флага переноса C, установка C выполняется в два этапа: сначала сброс, потом инверсия. Удобно осуществить сброс флага до команды анализа вводимого бита, тогда, если принятый бит является нулем, значение C сохраняется, если единицей – инвертируется.

В MK51 прием последовательного кода в УАПП и его преобразование в параллельный код инициируются старт-битом и выполняются аппаратурными средствами без участия программы. Основная программа MK51 должна только считать содержимое буфера УАПП после завершения приема очередного байта.

6.5.3. Цифро-аналоговые и аналого-цифровые преобразования

Цифро-аналоговое преобразование. Преобразование информации из цифровой формы в аналоговую осуществляется путем подключения БИС ЦАП к одному из портов МК. Выдача аналогового управляющего воздействия в этом случае сводится к одной команде, например OUTL P1, A. При этом на выходе ЦАП появится напряжение (ток), пропорциональное двоичному коду, загруженному впорт 1.

Некоторые объекты управления могут требовать непрерывного управляющего воздействия сложной формы. Для реализации такого воздействия в МК используются цифровые методы интегрирования: на каждом интервале времени Δt непрерывная функция заменяется ее средним дискретным значением. Таким образом, управляющее воздействие становится ступенчатым (рис. 6.12) и программа его формирования может

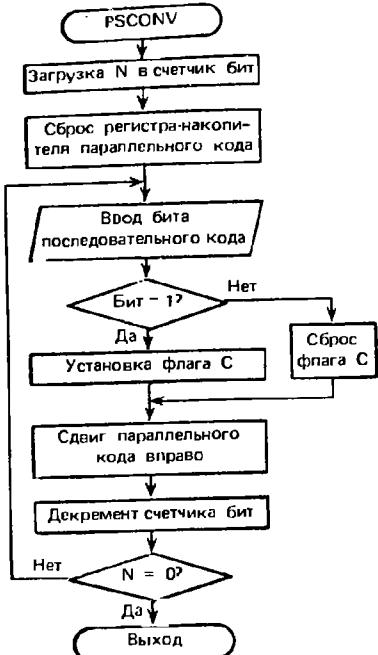


Рис. 6.11. Блок-схема процедуры преобразования последовательного кода в параллельный

из периодов дискретизации. Если допустить, что длина TABL в байтах хранится в ее первом байте, а выводимые из таблицы двоичные коды не требуют масштабирования, то программа формирования управляющего воздействия в МК48 будет иметь вид

```

FACNCNV: MOV R3, $01 TABL ;ФОРМИРОВАНИЕ ВНУТРИСТРАНИЧНОГО АДРЕСА
          MOV A,R3      ; ПЕРВОГО БАЙТА ТАБЛИЦЫ TABL
          MOVR3 A,0A      ;ЧТЕНИЕ АДРЕСА ТАБЛИЦЫ
          ; ИЗ ПАМЯТИ ПРОГРАММЫ
          ; R(R2) (-- АДРЕСА ТАБЛИЦЫ TABL
          ; УПРОЛІШЕННЯ УКАЗАТЕЛЯ TABL
LOOP:   MOV R2,A
          INC R3
          MOV A,R3
          MOVR3 A,0A      ;ЧТЕНИЕ КОДА ИЗ TABL
          OUTL PL,A       ;ВЫВОД КОДА НА ЦАП
          CALL DELAY      ;ЗАДЕРЖКА НА АДРЕСУ ПЕРИОДА
          ADD R2, #1       ;АДЕМЕНТ R2, И ЦИКЛ, ЯКЩО НЕ МИЛЬ
          BZJZ R2,LOOP     ;ВХОДА ИЗ ПРОЦЕДУРЫ
EXIT:   ...
    
```

Аналогово-цифровое преобразование. Преобразование аналогового сигнала от датчика в цифровой код, принимаемый и обрабатываемый в МК можно осуществить несколькими способами:

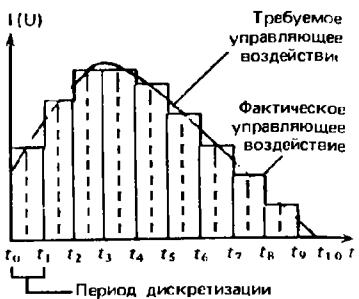


Рис. 6.12. Дискретизация непрерывной функции

быть реализована с использованием процедур выдачи кода и временной задержки заданной длительности.

Если предположить, что для управления некоторым объектом требуется сформировать управляющее воздействие, показанное на рис. 6.12, с периодом дискретизации 100 мкс, то прежде всего в ПП следует сформировать таблицу (TABL) двоичных эквивалентов дискретных значений функции для каждого

1) аппаратурным на основе БИС АЦП, подключаемой к порту МК. В этом случае МК только инициирует АЦП и через заданные периоды дискретизации считывает из него цифровой код. Данный способ характеризуется самым высоким быстродействием, но требует использования БИС АЦП, что далеко не во всех применениях МК является оправданным:

2) аппаратурно-программным на основе БИС ЦАП и программы взвешивания бит (последовательных приближений, побитного уравновешивания). Данный способ характеризуется хорошим быстродействием и требует использования относительно простых и дешевых микросхем ЦАП и операционного усилителя;

3) программно-аппаратурным на основе метода двойного интегрирования. Это самый дешевый, но и наиболее медленный способ: может обеспечить достижение очень высокой точности преобразования. Из дополнительного оборудования требуются два операционных усилителя и аналоговый мультиплексор на два входа;

4) аппаратурно-программным на основе использования преобразователя «напряжение→частота» и программы измерения периода сигнала.

Наиболее практический интерес представляют способы 2 и 3, так как их использование обеспечивает получение высоких технико-экономических характеристик МК-систем относительно простыми средствами.

Метод последовательных приближений. Из дополнительной аппаратуры в МК-устройстве используются интегральные ЦАП на восемь входов и сравнивающий операционный усилитель (компаратор), включенные так, как показано на рис. 6.13.

На выходе компаратора формируется сигнал 0, если $U_{ЦАП} < U_{вх}$, и сигнал 1, если $U_{ЦАП} > U_{вх}$. Микроконтроллер через порт Р1, работающий в режиме вывода, передает двоичные коды в ЦАП, выход которого подсоединен к входу «плюс» компаратора. Выход компаратора (двоичный сигнал) заводится на вход тестирования Т0.

Программа аналого-цифрового преобразования работает следующим образом: МК выдает через порт 1 байт данных, преобразуемый ЦАП в аналоговый сигнал $U_{ЦАП}$ и сравниваемый с входным аналоговым сигналом $U_{вх}$, а затем анализирует результат сравнения. В зависимости от значения сигнала на входе Т0 МК или оставляет старший бит выводимого байта в 1, если $U_{ЦАП} > U_{вх}$, или сбрасывает его в 0, если $U_{ЦАП} < U_{вх}$. Затем аналогичным образом в порядке убывания весовых значений проверяется каждый бит выводимого байта:

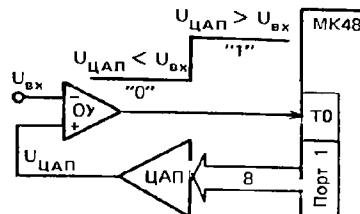


Рис. 6.13. Схема аналого-цифрового преобразования

```

;ВЕРСИЯ ДЛЯ МК48
;R4 - РЕГИСТР АНАЛОГОВОГО ЭКВИВАЛЕНТА
;R3 - РЕГИСТР БЕГУЩЕГО ЕДИНИЦЫ ДЛЯ УКАЗАНИЯ
;    ТЕКУЩЕГО ВЗВЕШИВАЕМОГО БИТА
;CONVERT: MOV R3,$08H ;ЗАГРУЗКА СЧЕТЧИКА ЦИКЛОВ
        MOV R3,$1
        MOV R4,$0
LOOP:  MOV A,R3 ;СДЕЛАЙ (R3) ВПРАВО НА 1 РАЗНЯ
        RR
        MOV R3,A
        ORL P1,A ;ВЗДЕЛИВАНИЕ ТЕКУЩЕГО РАЗРЯДА
        OUTL P1,A ;ВЫДАЧА ПРОИЗВОЛЮЧКОГО ЦИФРОВОГО
                    ;ЭКВИВАЛЕНТА НА ЧАСТЬ
JTO:   ENOUGH: JECAM TO=1, TO ;ЕСЛИ TO=1, ТО ВЫДАННЫЙ БЫЛ БОЛЬШЕ
                    ;ДВОЙЧКОГО ЭКВИВАЛЕНТА ОДНХ, И
                    ;СОХРАНЯЕТСЯ СТАРОЕ ЗНАЧЕНИЕ R4
                    ;ЕСЛИ TO=0, ТО УСТАНОВЛЕННЫЙ БЫЛ
MOV R4,A ;ЗАКОНЧИМАСЯ В РЕГИСТРЕ R4
ENOUGH: DJNZ RS,LOOP ;ЗАКРЫЕМЕР RS И ЕСЛИ НЕ НУЛЬ, ТО
                    ;ПЕРЕХОД К АНАЛИЗУ СЛЕДУЮЩЕГО (J-1)-Го БИТА

```

Метод двойного интегрирования. Схема подключения к МК дополнительной аппаратуры показана на рис. 6.14. Первоначально на вход интегратора подается положительное напряжение $E_{\text{оп}}$. При этом на выходе интегратора через некоторое время установится отрицательный уровень, а на выходе компаратора будет сформирован сигнал 0. Процесс преобразования состоит из двух этапов. Сначала производится интегрирование входного аналогового сигнала в течение строго определенного времени T_1 . Отсчет интервала T_1 производится от момента t_0 перехода напряжения на выходе интегратора через нуль. Входной преобразуемый сигнал (для данной схемы) должен быть отрицательного напряжения. Затем в момент времени t_1 на вход интегратора подается опорное напряжение $E_{\text{оп}}$ (противоположной полярности) и измеряется время интегрирования T_2 , которое и будет пропорционально входному напряжению ($U_{\text{вх}}$).

Время T_1 (период первого интегрирования) выбирается так, чтобы при максимальном входном напряжении ($U_{\text{вх}, \text{ макс}} = E_{\text{оп}}$) интегратор не вошел в насыщение:

```

;ВЕРСИЯ ДЛЯ МК51
MOV TMOD,$01H ;НАСТРОЙКА T/CO НА РЕЖИМ 16 БИТ
MOV TH0,$HIGH(NOT(T1)+1) ;ЗАГРУЗКА T/CO
MOV TL0,$LOW(NOT(T1)+1)
SETB P1.1 ;НАСТРОЙКА P1.1 НА ВВОД
SETB P1.0 ;ПОДАЧА ЕОП НА ВХОД ИНТЕГРАТОРА
WAIT: JB P1.1,WAIT ;ПОДАЧА ПОЯСНЕНИЯ НА ВЫХОДЕ
                    ;ИНТЕГРАТОРА ОТРИЦАТЕЛЬНОГО УРОВНЯ
CLR P1.0 ;ПОДАЧА УВХ НА ВХОД ИНТЕГРАТОРА
WAIT10: JNB P1.1,WAIT10 ;ПОДАЧА ЕОП НА ВХОД ИНТЕГРАТОРА
SETB TCON.4 ;СТАРТ T/CO
WAIT11: JNB TCON.5,WAIT11 ;ПОДАЧА ЕОП НА ВХОД ИНТЕГРАТОРА
SETB P1.0 ;НАЧАЛО ОБРАТНОГО ИНТЕГРИРОВАНИЯ
WAITT2: JB P1.1,WAITT2 ;ПОДАЧА ЕОП НА ВХОД ИНТЕГРАТОРА
CLR TCON.4 ;СТОП T/CO
CLR TCON.5 ;СБРОС ФЛАГА TFO
MOV B,TH0 ;СФОРМИРОВАНИЕ РЕЗУЛЬТАТА В
MOV A,TLO ;РЕГИСТРОВОЙ ПАРЕ (B/A)

```

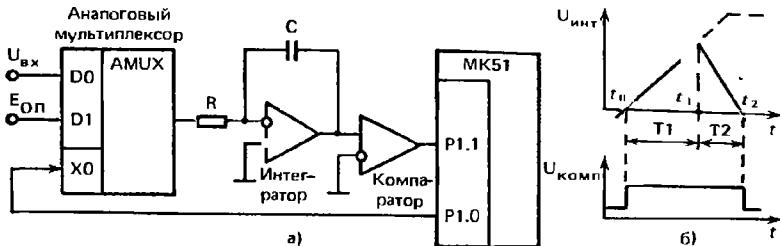


Рис. 6.14. Схема и временная диаграмма работы АЦП

Программа позволяет сформировать 16-битный код, эквивалентный входному сигналу в диапазоне, например, от 0 до -10 В, т.е. обеспечивает высокую точность преобразования. Максимальное время ($T_1 + T_2$) преобразования (при $U_{\text{вх}} = U_{\text{вх}, \text{ макс}}$) составляет $2 \times 65,535$ мс. Исходя из этого времени подбираются значения R и C . Если столь высокая точность преобразования не требуется в МК-системе, то можно использовать T/CO в режиме 8-битного таймера. При этом максимальное время преобразования сокращается до 2×256 мкс.

Организация связи с оператором в обслуживаемых МК-системах

7.1.

Ввод информации с клавиатуры

Во многих применениях МК работают автономно по заранее заданной программе без вмешательства человека. Паряду с этим существуют интерактивные МК-системы, включающие в контур управления человека-оператора. Простейший пример интерактивной управляющей системы – обслуживаемый МК, требующий ввода оперативной информации и ее отображения.

В качестве устройств ввода/вывода информации наиболее широкое распространение в МК-системах получили цифровые, алфавитно-цифровые и специальные клавиатуры, а также линейные дисплеи на семисегментных и матричных светодиодных индикаторах, алфавитно-цифровые и графические дисплеи на газоразрядных, жидкокристаллических и плазменных панелях.

Разновидности клавиатур. В различных по сложности и назначению управляющих системах используются разнообразные клавиатуры для ввода информации:

простейшие, состоящие из клавиш управления типа СБРОС, ПУСК, ОСТАНОВ и т.п.;

цифровые, предназначенные для ввода данных и управления режимом работы МП-системы и состоящие из шестнадцатеричной клавиатуры и управляющих клавиш ЗАГРУЗКА, АДРЕС/ДАННЫЕ, ПОШАГОВЫЙ РЕЖИМ, ИНДИКАЦИЯ и т.д.;

алфавитно-цифровые;

специализированные клавиатуры, в которых каждой клавише соответствует некоторая процедура процесса управления, например ПОВЫСИТЬ ДАВЛЕНИЕ В МАСЛЯНОЙ МАГИСТРАЛИ, ПОНИЗИТЬ ТЕМПЕРАТУРУ ОХЛАЖДАЮЩЕЙ ЖИДКОСТИ и т.п.;

многофункциональные клавиатуры на основе сенсорных переключателей, дополняемых сменяемыми шильдиками (лицевыми панелями) с соответствующими надписями. Эти клавиатуры при наличии соответствующих программных средств позволяют на одних и тех же аппаратуарных средствах реализовать набор разнообразных технологических языков и обеспечить их оперативную замену.

По способам аппаратурной реализации различают два типа клавиатур: кодирующую и некодирующую. В клавиатурах первого типа схемным путем на выходе формируется код, соответствующий нажатой клавише. Из-за значительного объема неунифицированной аппаратуры схем преобразования кодов и высокой стоимости, которые резко возрастают с ростом числа знаков, такие клавиатуры в МК-системах применяются редко. Значительно более широкое распространение получили дешевые некодирующие (матричные) клавиатуры, которые представляют собой простую матрицу двоичных переключателей (требуемой размерности), включенных на пересечении строк и колонок матрицы. Идентификация (кодирование) нажатой клавиши в таких клавиатурах выполняется программой.

Ввод кода нажатой клавиши. Для обслуживания клавиатур в МК-системах используются две процедуры: опрос состояния клавиатуры и ввод кода нажатой клавиши.

Первая процедура производит однократное обращение к матрице клавиш для определения, нажата ли хотя бы одна из клавиш. Вторая осуществляет циклический опрос клавиатуры до тех пор, пока не будет нажата (а часто и освобождена) клавиша. Быдучи встроена в основную программу, вторая процедура блокирует процесс управления объектом на время ожидания нажатия клавиши, а потому обращение к ней осуществляется только при обнаружении нажатой клавиши процедурой опроса состояния клавиатуры.

Вместо процедуры опроса состояния клавиатуры можно использовать аппаратурные средства, формирующие сигнал внешнего прерывания для МК в случае нажатия любой клавиши.

Процедуру ввода информации с некодирующей матричной клавиатурой удобно рассмотреть на примере клавиатуры 4×5 , включающей 16 цифровых клавиш (0 – F) и 4 управляющих. Структура матрицы клавиатуры аналогична структуре матрицы двоичных датчиков, способ подключения клавиатуры к МК представлен на рис. 7.1.

Линии порта 1 используются для сканирования, а линии порта 2 – для опроса матрицы клавиш. Каждая клавиша в такой матрице имеет свой номер, соответствующий ее местоположению. На цифровые клавиши можно нанести обозначения, соответствующие их кодам (от 0 до F). Коды управляющих клавиш больше числа 15. Диоды обеспечивают защиту от замыкания между собой сканирующих линий в случае одновременного нажатия более чем одной клавиши.

Процедура ввода кода нажатой клавиши состоит из последовательности частных процедур (некоторые из них уже были рассмотрены ранее): сканирования матрицы клавиш, устранения дребезга контактов, ожидания освобождения клавиши и идентификации кода нажатой клавиши. Для некоторого типа клавиатур может отсутствовать процедура устранения дребезга контактов (клавиатуры на основе герконов). Процедуру сканирования иногда бывает удобно совместить с процедурой идентификации.

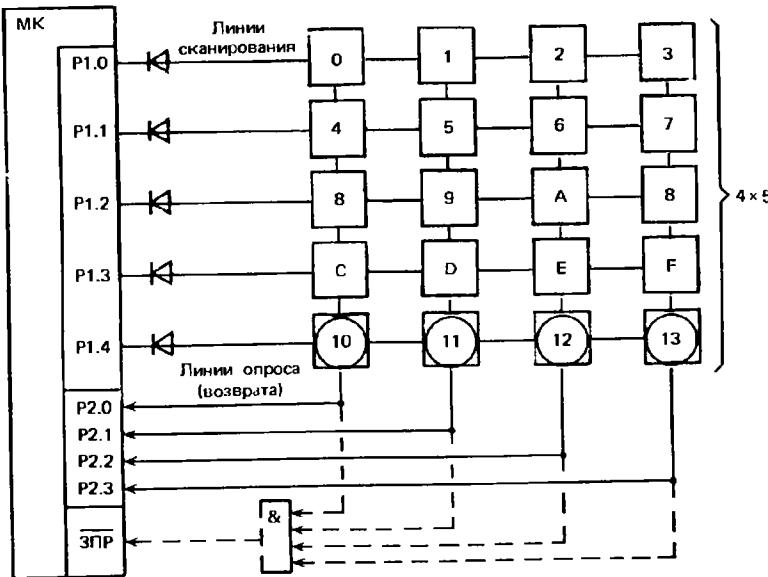


Рис. 7.1. Схема подключения клавиатуры 4 × 5 к МК

Для программного ввода информации с некодирующих клавиатур характерен один недостаток, а именно — срабатывание по отпусканью клавиши, а не по нажатию. Однако при кратковременных нажатиях клавиш этот эффект не имеет особого значения.

Рассмотрим отдельно каждую из перечисленных частных процедур. **Сканирование.** Частная процедура сканирования служит для обнаружения нажатой клавиши и последующей ее идентификации. Процедура сводится к поочередному обнулению каждой из линий сканирования и опросу линий возврата. В порт 1 выдается байт сканирования (БС), содержащий 0 только в одном бите. Если на пересечении линии сканирования и линии возврата находится нажатая клавиша, то в соответствующем бите байта возврата (БВ), принимаемого в порт 2, будет находиться 0.

Последовательность байтов сканирования представляет собой код "бегущий нуль"; формирование очередного байта сканирования осуществляется путем сдвига его предыдущего значения. Направление сдвига определяет последовательность опроса клавиши. Схема алгоритма процедуры сканирования представлена на рис. 7.2. Если при полном цикле сканирования не было обнаружено нажатой клавиши, то процедура сканирования повторяется сначала.

Оператор "Есть нажатая клавиша?" может быть реализован двумя способами: наложением маски или сдвигом. Первый способ подробно рассмотрен в типовой процедуре опроса группы импульсных датчиков.

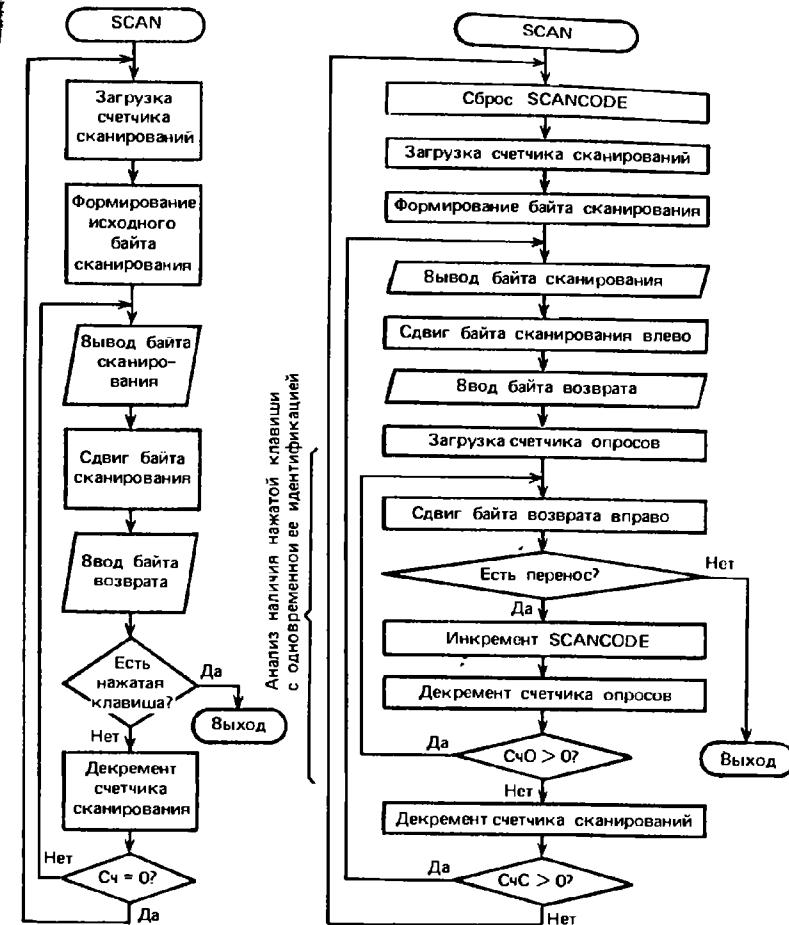


Рис. 7.2. Блок-схема процедуры сканирования клавиатуры

На рис. 7.3 представлена блок-схема анализа нажатой клавиши с одновременной ее идентификацией. При втором способе клавиши, подключенные к каждой линии сканирования, анализируются последовательно. Если после анализа каждой клавиши осуществлять прибавление единицы к счетчику SCANCODE, то процедуре сканирования можно совместить с процедурой идентификации нажатой клавиши (рис. 7.3). После выхода из процедуры SCAN в SCANCODE будет находиться код нажатой клавиши. Кроме того, процедура SCAN осуществляет защиту от одновременного нажатия нескольких клавиш. Порядок анализа клавиши таков, что при одновременном нажатии клавиши с большим кодом инициируется:

```

;ВЕРСИЯ ДЛЯ МК48 И МК51
;R4 - СЧЕТЧИК КОДА СКАНИРОВАНИЯ
SCAN: MOV R4,$0      ;СБРОС SCancode
      MOV R7,$5      ;ЗАГРУЗКА СЧЕТЧИКА СКАНИРОВАНИЯ
      MOV R6,$11111110B ;ЗАГРУЗКА ИХСОЛНОГО БАЙТА СКАНИРОВАНИЯ
LDOF: MOV A,R6      ;ВЫХОД ТЕКУЩЕГО БАЙТА СКАНИРОВАНИЯ
      OUTL P1,A      ;СДВИГ БАЙТА СКАНИРОВАНИЯ
      RL A           ;СДВИГ БАЙТА ВОЗВРАТА
      MOV R6,A      ;СОХРАНЕНИЕ ТЕКУЩЕГО БАЙТА
      ; СКАНИРОВАНИЯ
      IN  A,P2      ;ВВОД БАЙТА ВОЗВРАТА
      MOV R5,$4      ;ЗАГРУЗКА СЧЕТЧИКА ОПРОСОВ
      ;СДВИГ БАЙТА ВОЗВРАТА
      RRC A          ;ВЫХОД ИЗ ПРОЦЕДУРЫ ПРИ ОБНАРУЖЕНИИ
      ; ПЕРВОЙ НАЖАТОЙ КЛАВИШИ
      INC R4          ;ИНКРЕМЕНТ SCancode
      DJNZ R5,ROTATE ;АИНКРЕМЕНТ СЧЕТЧИКА ОПРОСОВ И ЧИКИ,
      ; ЕСЛИ НЕ НУЛЬ
      DJNZ R7,LOOP   ;АИНКРЕМЕНТ СЧЕТЧИКА СКАНИРОВАНИЯ И ЧИКИ,
      ; ЕСЛИ НЕ НУЛЬ
      JMP SCAN      ;НАЧАТЬ СКАНИРОВАНИЯ

```

После выхода из процедуры SCAN управление передается частной процедуре устранения дребезга контактов DBNC.

Устранение дребезга контактов. Устранение дребезга контактов при вводе символа с клавиатуры осуществляется, как правило, программной реализацией временной задержки 5–20 мс (в зависимости от механических характеристик клавиатуры):

```
DBNC: CALL  DELAY . ;ЗАДЕРЖКА
```

Если возможно возникновение дребезга контактов и при освобождении клавиши, то процедуру DBNC необходимо вставить и после процедуры ожидания освобождения клавиши.

Ожидание освобождения клавиши. Для того чтобы при повторном обращении МК к клавиатуре не был введен код той же самой клавиши, вводится процедура ожидания освобождения. После выполнения сканирования активной осталась та линия сканирования, в которой обнаружена нажатая клавиша. Поэтому процедура ожидания освобождения нажатой клавиши сводится к считыванию и анализу байта возврата:

```

;ВЕРСИЯ ДЛЯ МК48
WAITOP: IN  A,P2      ;ВВОД БАЙТА ВОЗВРАТА
      CPL A          ;ФИНВЕРСИЯ БАЙТА ВОЗВРАТА
      JNZ WAITOP    ;ЕСЛИ КЛАВИША НАЖАТА, ТО ЗАТЬ

```

```

;ВЕРСИЯ ДЛЯ МК51
WAITOP: MOV A,$OFFH
WAIT: CJNE A,P2,WAIT ;ЕСЛИ КЛАВИША НАЖАТА, ТО ЗАТЬ

```

Процедура WAITOP в том виде, в котором она приведена выше, может быть использована только в системах, защищенных от "залипания" контактов.

Обнаружить "залипание" контакта можно путем подсчета "неудачных" опросов. Если число "неудачных" опросов превысило N и клавиша не была освобождена, выдается сообщение оператору о "залипании" контакта. Число N (и соответственно разрядность счетчика) определяется исходя из механических свойств клавиатуры.

Идентификация нажатой клавиши. Каждой клавише клавиатуры должен быть поставлен в соответствие код (ее вес), являющийся функцией номеров линий сканирования (С) и линии возврата (В), на пересечении которых нажата клавиша. Процедура идентификации нажатой клавиши KEYW может быть совмещена с процедурой сканирования (как в рассмотренном выше примере). Тогда после выхода из процедуры SCAN в регистре SCANCODE будет размещен код нажатой клавиши.

Для сложных клавиатур SCancode не всегда удается совместить с истинным весом клавиши. В этом случае необходима дополнительная перекодировка, которая выполняется табличным способом с использованием SCancode в качестве указателя.

Процедура KEYW может быть оформлена и как самостоятельная. В этом случае наиболее распространенный способ вычисления веса нажатой клавиши – аналитический в соответствии с выражением $W = n \times C + B$, где n – количество линий возврата. Номера активных линий сканирования и возврата представлены в унитарном коде в виде байта сканирования и байта возврата. Поэтому процедура KEYW реализуется на основе процедур преобразования унитарного кода в двоичный и вычисления W.

Кроме аналитического существует табличный способ определения кода нажатой клавиши.

Оформление процедуры ввода. Процедура ввода кода клавиши KEYBRD оформляется в виде линейной последовательности рассмотренных выше частных процедур:

KEYRD:	SCAN: ...	;ВЕРСИЯ ДЛЯ МК48 И МК51 ;СКАНИРОВАНИЕ КЛАВИАТУРЫ
BBNC:	CALL DELAY	;УСТРАНЕНИЕ ДРЕБЕЗГА ПРИ НАЖАТИИ ;ВОЗВРАЩЕНИЕ ОСВОБОЖДЕНИЯ КЛАВИШИ
DBNC:	CALL DELAY	;УСТРАНЕНИЕ ДРЕБЕЗГА КОНТАКТОВ ПРИ ;ОСВОБОЖДЕНИИ КЛАВИШИ (МОЖЕТ ;БЫ ОТСУСТВОВАТЬ)
KEYW:	... ;ИДЕНТИФИКАЦИЯ НАЖАТОЙ КЛАВИШИ ... ;МОЖЕТ ОТСУСТВОВАТЬ	

Процедура опроса состояния клавиатуры. Выше уже отмечалось, что в МК-системах, реализующих непрерывное управление, процедуре KEYBRD должна предшествовать процедура опроса состояния клавиатуры ASK. Пример программной реализации процедуры ASK, оформленной в виде подпрограммы, приведен ниже.

Выходной параметр передается в основную программу через флаг переноса С, который устанавливается, если хотя бы одна клавиша нажата:

ИВЕРСИЯ	ДЛЯ	МК48	
ASK:	CLR	A	: СБРОС АККУМУЛЯТОРА
	CLR	C	: СБРОС ФЛАГА ПЕРЕНОСА.
	OUTL	P1>A	: ВВОД БАЙТА "ВСЕ НУЛИ" ДЛЯ : ОДНОВРЕМЕННОГО ОПРОСА ВСЕХ КЛАВИШ
IN	A>P2		: ВВОД БАЙТА ВОЗВРАТА
CPL	A		: ИНВЕРСИЯ БАЙТА ВОЗВРАТА
JZ	EXIT		: ВЫХОД, ЕСЛИ НЕТ НАЖАТИЯ КЛАВИШИ
CPL	C		: УСТАНОВКА ФЛАГА ПЕРЕНОСА
EXIT:	KET		: ВОЗВРАТ

Подпрограмма производит одновременный опрос всех клавиш. В случае, если хотя бы одна клавиша нажата (байт возврата – не все единицы), устанавливается флаг переноса, иначе – сбрасывается.

7.2.

Вывод и отображение информации

Индикаторы. Многие МК-устройства требуют наличия только простейшей индикации типа Да/Нет, ВКЛ/Выкл. Такая индикация реализуется на основе отдельных светодиодов.

Семисегментные индикаторы (ССИ) широко используются для отображения цифровой и буквенной информации. Семь отображающих элементов позволяют высвечивать десятичные и шестнадцатеричные цифры, некоторые буквы русского и латинского алфавитов, а также некоторые специальные знаки. Структура ССИ и способы его подключения к МК показаны на рис. 7.4. Для засветки одного сегмента большинства типов ССИ необходимо обеспечить протекание через сегмент тока 10–15 мА при напряжении 2,0–2,5 В. Низкая нагрузочная способность МК не допускает прямого соединения с ССИ. В качестве промежуточных усилителей тока могут использоваться логические элементы серии K155 или интегральные схемы преобразователей кодов для управления ССИ.

Преобразование двоичных кодов в коды для ССИ может осуществляться либо программно, либо аппаратурно с использованием преобразователей K514ИД1, K514ИД2, 133ПП4, 564ИД5.

Матричные светодиодные индикаторы (МСИ) используются для отображения алфавитно-цифровой информации. Каждый из таких МСИ, выполненный в виде интегральной микросхемы, представляет собой матрицу светодиодов размерностью $m \times n$, где n – число колонок, m – число строк матрицы. Наибольшее распространение получили МСИ с размерностью матрицы 7×5 и 9×7 (рис. 7.5).

Для включения одного светодиода матрицы необходимо обеспечить протекание через него тока 10–15 мА при напряжении 2,0–2,5 В. Подключение матричного индикатора к МК осуществляется через управляемые схемы формирования тока колонок и строк (рис. 7.6).

Для отображения многосимвольной информации используются **линейные (односторочные) дисплеи**. Такие дисплеи представляют собой "ленту", смонтированную из отдельных ССИ или МСИ. Число зна-

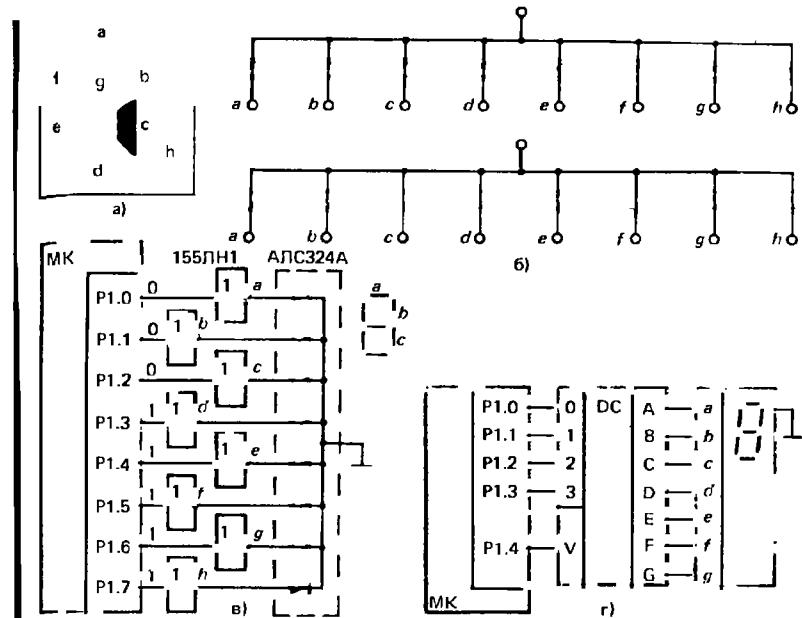


Рис. 7.4. Семисегментный индикатор:
а – внешний вид; б – схемы; в, г – способы подключения к МК

комест дисплея определяется в соответствии с требованиями к МК-системе.

Существует два способа организации интерфейса МК с линейным дисплеем: статический и динамический.

Первый требует наличия на входах каждого индикатора специальных буферных регистров для хранения кодов выводимых символов. Естественно, что с увеличением разрядности дисплея возрастает число дополнительных микросхем, а следовательно, и стоимость МК-системы.

Второй способ (динамический) основан на том, что любой световой индикатор является инерционным прибором, а человеческому глазу отображаемая на дисплее информация, если ее обновлять с частотой примерно 20 раз в секунду, представляется неизменяемой. Динамический способ вывода информации на дисплей требует значительно меньших аппаратурных затрат, но более сложного программного обеспечения. Именно этот способ организации вывода информации получил преимущественное распространение в МК-системах.

Вывод символа на ССИ. При использовании внешних (по отношению к МК) схем преобразователей кодов процедура индикации одного символа сводится к выдаче двоичного кода символа в соответствующий порт вывода МК.

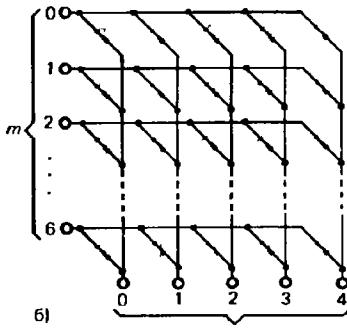
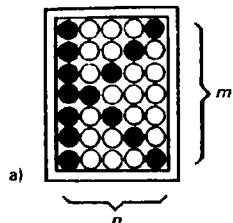


Рис. 7.5. Матричный индикатор:
а – общий вид; б – схема

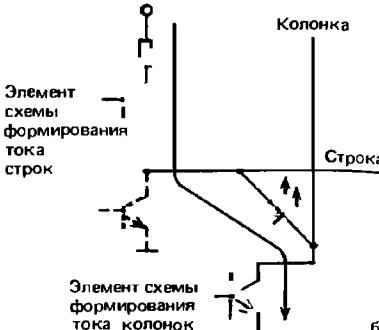
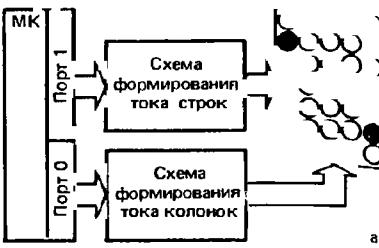


Рис. 7.6. Схема подключения матричного индикатора к МК (а) и цепь протекания тока через светодиод (б)

При программной перекодировке МК должен поставить в соответствие двоичному коду индицируемого символа определенный байт индикации (БИ), который и выдается в выходной порт. Перекодировку удобнее всего проводить табличным способом. Для этого байты индикации размещаются в смежных ячейках РПП в порядке возрастания исходных двоичных кодов символов. Такое расположение байтов индикации упрощает процесс перекодировки, так как в этом случае требуемый байт находится в строке таблицы с номером, равным двоичному коду индицируемого символа:

```
#ФОРСИЯ ДЛЯ МК48
#ПРОГРАММА СТАТИЧЕСКОЙ ИНДИКАЦИИ СИМВОЛА
; НА СЕМІСЕГМЕНТНОМ ИНДИКАТОРЕ
SYMBOL: MOV A,0E0 ;ЗАГРУЗКА В АККУМУЛЯТОР КОД СИМВОЛА
ADD A, #CODETBL ;ФОРМОВАННИЕ АДРЕСА БАЙТА ИНДИКАЦІЇ
MOV#3 A,0A ;СЧИТИВАННІ БАЙТА ИНДИКАЦІЇ ІЗ ТАБЛИЦІ
OUTL P1,A ;ВЫВОД БАЙТА ИНДИКАЦІЇ НА ИНДИКАТОР ЧЕРЕЗ ПОРТ 1
```

Приведенный фрагмент программы рассчитан на то, что гашение ССИ осуществляется при инициализации системы. Одновременно с этим в регистре R0 формируется адрес ячейки CODE, в которой хранится двоичный код индицируемого символа:

```
INIT: ... R2, #0FFH ;ГАШЕНІЕ ИНДИКАТОРА
      MOV R0, #CODE ;ЗАГРУЗКА В R0 АДРЕСА CODE
      ...
```

Вывод информации на линейный дисплей ССИ. При динамической индикации байт индикации поступает одновременно на входы всех ССИ, образующих линейный дисплей, а выбор знакоместа осуществляется байтом выборки, представляющим собой код "бегущий нуль" (рис. 7.7). При бездесифраторном способе формирования байта выборки максимальное число знакомест линейного дисплея ограничено разрядностью порта. Использование для формирования кода "бегущий нуль" внешнего дешифратора позволяет значительно увеличить число знакомест линейного дисплея.

Для динамической индикации группы символов удобно воспользоваться процедурой индикации символа, оформив ее в виде параметризируемой подпрограммы в соответствии с БСА, показанной на рис. 7.8. Входными параметрами для подпрограммы DSPLY являются исходный код отображаемого символа и номер знакоместа, на которое осуществляется вывод. Исходный код символа задается текущим адресом в сдвиге CODE (регистр R0), а номер знакоместа – текущим значением байта выборки (регистр R2) и значением счетчика знакомест регистра R7:

```
#ФОРСИЯ ДЛЯ МК48
DSPLY: MOV A, #0FFH ;БЛАНКИРОВАНИЕ (ГАШЕНИЕ
      OUTL BUS_A ; ВСЕХ ИНДИКАТОРОВ)
      MOV A, #R0 ;ВЫБОРКА КОДА СИМВОЛА
      ADD A, #CODETRL ;ПЕРЕКОДИРОВКА СИМВОЛА
      MOV#3 A, #0A ;БАЙТ ИНДИКАЦІЇ
      OUTL P1,A ;ВЫВОД БАЙТА ИНДИКАЦІЇ
      RDV A, #R2 ;СЛІДУЮЧИЙ БАЙТ ВИБОРКИ
      OUTL BUS_A ;ВЫВОД БАЙТА ВИБОРКИ
      RL A ;СЛІДУЮЧИЙ БАЙТ ВИБОРКИ
      MOV R2, A ; В СТОРІНУ СТАРШИХ БІТ
      INC R0 ;ПРОДЛІЖЕННЯ ПО МАСИВУ СОДЕ
      DJNZ R7, EXIT ;ДЕРЖЕМЕНТ СЧЕТЧИКА ЗНАКОМЕСТ,
; И ВЫХОД, ЕСЛИ НЕ НУЛЬ
      EXIT: RDV R2, #0FFH ;ЗАГРУЗКА В R2 ИСХОДНОГО БАЙТА ВИБОРКИ
      RDV R7, #8 ;ЗАГРУЗКА СЧЕТЧИКА ЗНАКОМЕСТ
      MOV R0, #CODE ;ЗАГРУЗКА В R0 ИНІЦІАЛЬНОГО АДРЕСА
; МАСИВА СОДЕ
      RET ;ВОЗВРАТ
```

Подпрограмма DSPLY реализует выборку кода очередного символа из РПД (исходные коды символов должны быть размещены в последовательно расположенных ячейках памяти), его перекодировку и отображение на текущем знакоместе. Для получения яркой и ровной (немигающей) индикации необходимо обеспечить: во-первых, запрет выборки знакомест на время изменения байта индикации в порте 1 (бланкирование), во-вторых, регенерацию изображения на каждом знакоместе

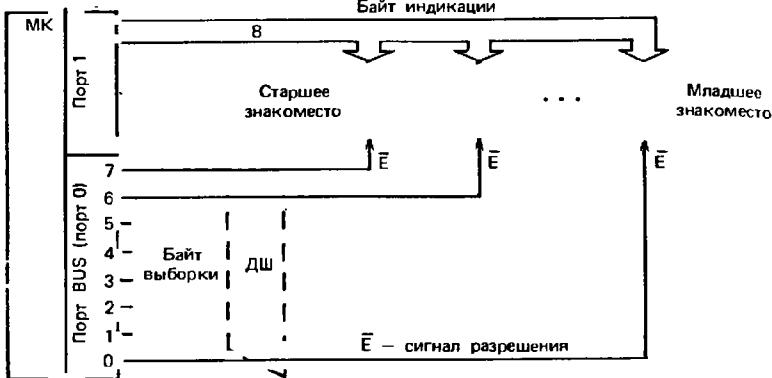


Рис. 7.7. Линейный дисплей на семисегментных светодиодных индикаторах

с частотой 20 раз в секунду, т.е. обращение к подпрограмме DSPLY через каждые $50/N$ мс, где N – число знакомест на дисплее. Бланкирование осуществляется выдачей байта выборки "все единицы". Требуемая частота регенерации изображения обеспечивается основной программой МК-системы, если она организована в соответствии со структурой:

```

INIT: ...           ;ИНИЦИАЛИЗАЦИЯ
      CALL INITU   ;ФОРМИРОВАНИЕ ИСХОДНЫХ ПАРАМЕТРОВ
                  ;ДЛЯ ЗАПРОГРАММИИ DSPLY

      ...           ;ВЫВОД ПЕРВОГО СИМВОЛА
;ФРАГМЕНТ ОСНОВНОЙ ПРОГРАММЫ ДЛЯ ТЕКУЩЕГО СИМВОЛА НЕ БОЛЕЕ 50/Н МС
      CALL DSPLY    ;ВЫВОД ВТОРОГО СИМВОЛА
;ФРАГМЕНТ ОСНОВНОЙ ПРОГРАММЫ ДЛЯ ТЕКУЩЕГО СИМВОЛА НЕ БОЛЕЕ 50/Н МС
      ...           ;ВЫВОД ТРЕТЬЕГО СИМВОЛА
      ...
      ...
      CALL BSPLY    ;ВЫВОД N-ГО СИМВОЛА
;ФРАГМЕНТ ОСНОВНОЙ ПРОГРАММЫ ДЛЯ ТЕКУЩЕГО СИМВОЛА НЕ БОЛЕЕ 50/Н МС
      CALL DSPLY    ;ВЫВОД ПЕРВОГО СИМВОЛА
      ...

```

Требуемая частота обращений к подпрограмме DSPLY может быть обеспечена также с помощью прерывания от таймера.

Вывод символа на МСИ. Для отображения символа на матричный индикатор обычно используется принцип динамической индикации по колонкам. Байт индикации в этом случае представляет собой код заставки светодиодов колонки, и графический образ символа "набирается" из последовательности байтов индикации путем перебора колонок (рис. 7.9). Процедуру вывода символа на МСИ целесообразно

представить в виде двух частных процедур: перекодировки и отображения.

Перекодировка. Представление исходного двоичного кода символа в виде последовательности кодов колонок осуществляется таблично. Каждому символу в таблице CODTBL соответствует пять ячеек. Адрес первого кода колонки (KK) для символа формируется как CODTBL + $+ 5 \times$ (двоичный код символа):

ВЕРСИЯ	АДР	МК1		
CODER1	MDV	DPTR, CODTBL	;ЗАГРУЗКА В DPTR НАЧАЛЬНОГО АДРЕСА	
	MOV	A, #R0	;ЧТЕНИЕ ТАБЛИЦЫ CODTBL	
	MOV	D, #5	;ЧТЕНИЕ ДВОИЧНОГО КОДА СИМВОЛА	
	MUL	AB	;ФОРМИРОВАНИЕ АДРЕСА ПЕРВОГО КОДА	
	MOV	R3, A	;КОЛОНКИ	
	MOV	R7, #5		
LOOP:	MOV	A, #R3	;ЗАГРУЗКА СЧЕТЧИКА КОДОВ	
	MOVC	A, #A+DPTR	;ЧТЕНИЕ ОЧЕРЕДНОГО КОДА КОЛОНКИ	
	MOV	BR1, A	;ИЗ ТАБЛИЦЫ CODTBL	
	INC	DPTR	;ЗАПИСЬ КОДА КОЛОНКИ В МАССИВ CWORD	
	INC	R1	;ПРОДВИЖЕНИЕ ВО ТАБЛИЦЕ CODTBL	
	DJNZ	R7, LOOP	;ПРОДВИЖЕНИЕ ВО МАССИВУ CWORD	
	JMP	EXIT	;ИНКРЕМЕНТ СЧЕТЧИКА КОДОВ,	
			;И МИКС, ЕСЛИ НЕ О	
			;ВЫХОД ИЗ ПРОЦЕДУРЫ	
ТАБЛИЦА	CODTBL:	DB	3EH, 41H, 41H, 41H, 3EH	;КОДЫ КОЛОНК СИМВОЛА "0"
		DB	0, 0, 10H, 20H, 0EFH	;КОДЫ КОЛОНК СИМВОЛА "1"
		DB	21H, 43H, 45H, 49H, 31H	;КОДЫ КОЛОНК СИМВОЛА "2"
		...		
		...		
	EXIT:	...		

Длина таблицы CODTBL для шестнадцатеричных символов (0 – F) равна 80 ячейкам памяти программ (5×16).

Процедуре CODFR должна предшествовать процедура инициализации, обеспечивающая загрузку регистров-указателей и гашение МСИ:

INIT:	MDV	RO, #CODE	;ЗАГРУЗКА В RO НАЧАЛЬНОГО АДРЕСА
	MOV	R1, #CWORD	;МАССИВА ИСХОДНЫХ КОДОВ СИМВОЛОВ
		SETB R1, #0	;ЗАГРУЗКА В R1 НАЧАЛЬНОГО АДРЕСА
	SETB	PO, 7	;МАССИВА КОДА КОЛОНК
	MOV	R2, #0	;ГАШЕНИЕ МАТРИЧНОГО ИНДИКАТОРА
			;СЕРОС БАЙТ ВЫБОРКИ

Отображение. Частная процедура отображения символа реализуется на основе параметризованной подпрограммы индикации текущего столбца. Подпрограмма строится по тому же принципу, что и рассмотренная ранее подпрограмма отображения символа на текущем знакоместе линейного дисплея на основе ССИ. Отличием подпрограммы COLUMN является то, что блок перекодировки вынесен в самостоятельную процедуру, а формирование нового байта выборки осуществляется не свивтом, а инкрементом регистра. Входными параметрами для подпрограммы являются: номер колонки (регистр R2), текущий адрес колонки (KK) в массиве CWORD (регистр R1) и текущее значение счетчика колонок (регистр R7):

```

:ВЕРСИЯ ДЛЯ МК51
COLUMN: SETB P0.7      :БЛАНКИРОВАНИЕ
MOV  P1.0R1                :ВЫВОД БАЙТА ИНДИКАЦИИ
MOV  P0,R2                :ВЫВОД БАЙТА ВЫБОРКИ И
                           :СНЯТИЕ БЛАНКИРОВАНИЯ
INC  R1                   :ПРОГРУЗКА ПО СЫХОД
INC  R2                   :ИДЕНТИФИКАЦИЯ БАЙТА ВЫБОРКИ
DJNZ R7,EXIT              :ДЕКРЕМЕНТ СЧЕТЧИКА КОЛОНОК;
                           :И ВЫХОД, ЕСЛИ НЕ 0
MOV  R1,SCWORD            :ПЕРЕИНИЦИАЛИЗАЦИЯ
MOV  R2,00
MDV  R7,^_OL

```

Перед первым вызовом подпрограммы в нее должны быть переданы исходные параметры:

```

MOV  R1,SCWORD            :ЗАГРУЗКА В R1 НАЧАЛЬНОГО АДРЕСА
                           :МАССИВА КОДА КОЛОНОК
MDV  R2,00                 :ЗАДАНИЕ ИНДЕКСНОГО АДРЕСА
MOV  R7,CTCOL              :ЗАГРУЗКА СЧЕТЧИКА КОЛОНОК
CALL COLUMN                :ПЕРВЫЙ ЭМЭДВ 10 ГРАММЫ

```

Задание исходного значения содержимого счетчика колонок символьическим именем CTCOL позволяет использовать подпрограмму COLUMN для реализации процедуры динамической индикации группы символов на линейном дисплее. Символическое имя CTCOL должно быть определено директивой ассемблера. Например, для MCS 5 x 7

```

CTCOL EQU 5               :ПРИСВОЕНИЕ СИМВОЛИЧЕСКОМУ ИМЕНИ
                           : CTCOL ЗНАЧЕ " "

```

Для обеспечения немерцающей и яркой индикации обращение к процедуре COLUMN должно осуществляться через каждые $50/5 = 10$ мс.

Вывод строки символов. Линейный дисплей на основе МСИ представляет собой устройство, показанное на рис. 7.10. С точки зрения разработчика программы это совокупность из $5N$ колонок, где N – число знакомест. Для вывода группы символов на дисплей с помощью процедуры COLUMN необходимо предварительно определить символическое имя CTCOL как $5N$. Для 8-позиционного линейного дисплея это определение выглядит так:

```

CTCOL EQU 40              :ПРИСВОЕНИЕ СИМВОЛИЧЕСКОМУ ИМЕНИ
                           : CTCOL ЗНАЧЕНИЯ 40

```

Коды всех колонок должны быть сформированы до обращения к подпрограмме COLUMN процедурой перекодировки, которая для рассматриваемого случая будет иметь вид

```

CONER_FOR_BISPLAY:
MOV  R0,SCODE              :ЗАГРУЗКА НАЧАЛЬНОГО АДРЕСА МАССИВА
                           :ИСХОДНЫХ КОДОВ СИМВОЛОВ
MOV  R1,SCWORD              :ЗАГРУЗКА НАЧАЛЬНОГО АДРЕСА ОБЛАСТИ
                           :РАЗМЕЩЕНИЯ КОДОВ СТОЛБОВ
MOV  R6,9N                  :ЗАГРУЗКА СЧЕТЧИКА ЗНАКОМЕСТ
                           :ПЕРЕКОДИРОВКА СИМВОЛА (ПОДПРОГРАММА
                           :CONER РАССМОТРЕНА РАНЕЕ)
                           :ДЕКРЕМЕНТ СЧЕТЧИКА ЗНАКОМЕСТ И ЧИЛ,
                           :ЕСЛИ НЕ НУЛЬ
LOOP1: CALL CONER
DJNZ R6,LDDP1

```

Рис. 7.8. Схема процессуры отображения символа на текущем знакоместе линейного дисплея

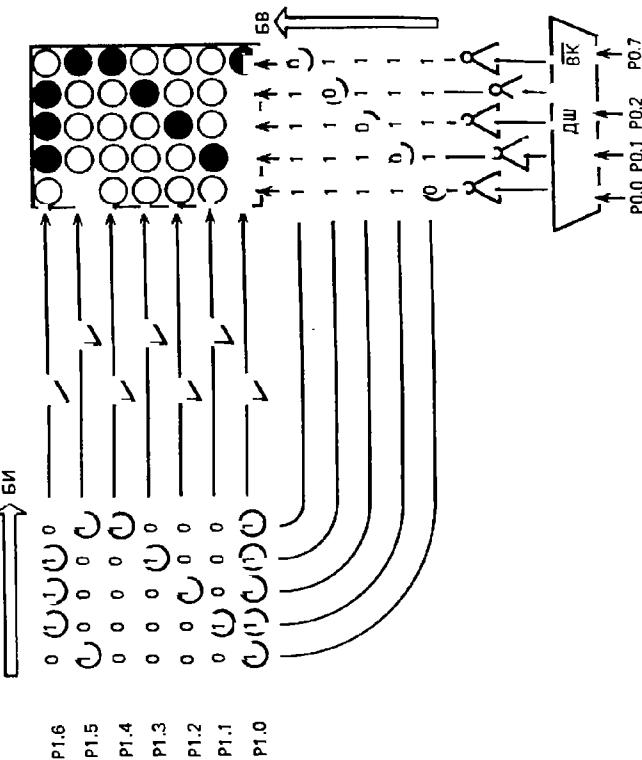
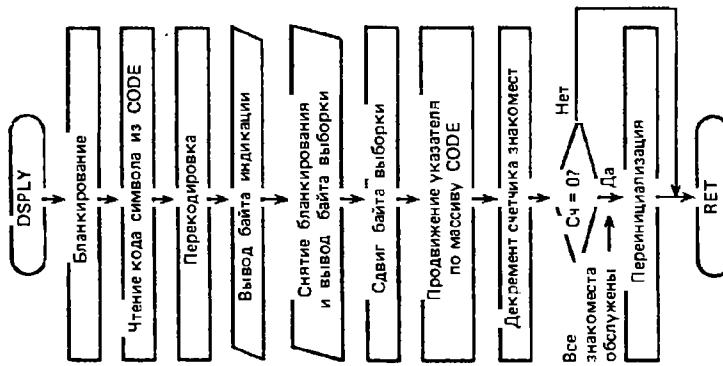


Рис. 7.9. Формирование символа на матричном индикаторе



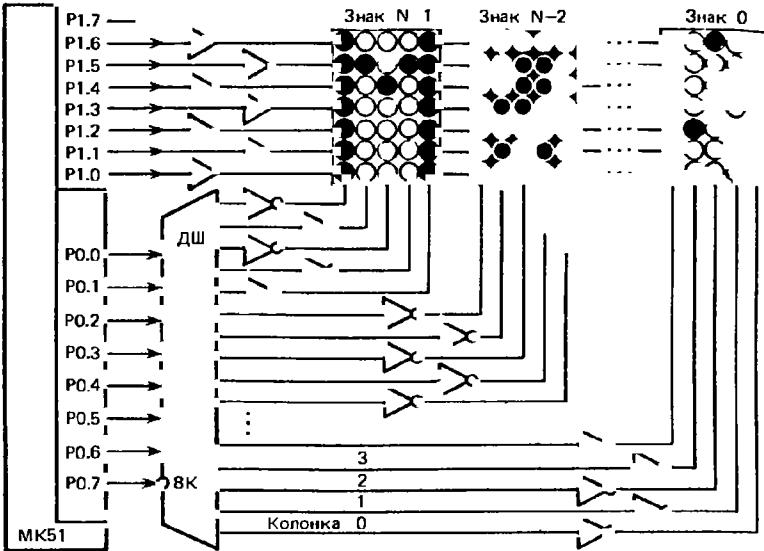


Рис. 7.10. Линейный дисплей на матричных индикаторах

Для обеспечения немерцающей и яркой индикации обращение к подпрограмме COLUMN должно осуществляться через каждые $50/40 = 1,25$ мс. Удобнее всего это делать по прерыванию от таймера.

7.3. Сопряжение МК с клавиатурой и линейным дисплеем на основе БИС KP580ВД79

Описание БИС контроллера клавиатуры/дисплея (ККД) КР580ВД79 приведено в приложении П1. Использование ККД позволяет разгрузить МК от рутинных операций опроса клавиатуры и поддержания (рефреша) изображения на однострочном дисплее.

Одна из возможных схем подключения контроллера клавиатуры к микропроцессору MK51 показана на рис. 7.11. При таком подключении контроллер входит в адресное пространство ВПД. Линия P1.0 соединяется с линией A₀ контроллера и должна быть установлена/сброшена через обращением МК к контроллеру в зависимости от типа обращения (управление/данные). Вход выборки контроллера соединен с общей точкой, и, таким образом, контроллер всегда готов к обмену с МК. Выход сигнала запроса прерывания (IRQ) контроллера соединен с линией P1.1 МК и может быть программно определен для определения факта нажатия клавиши.

Шина данных и линии чтения/записи контроллера соединяются напрямую с соответствующими линиями MK51. На вход С1 К подается сигнал

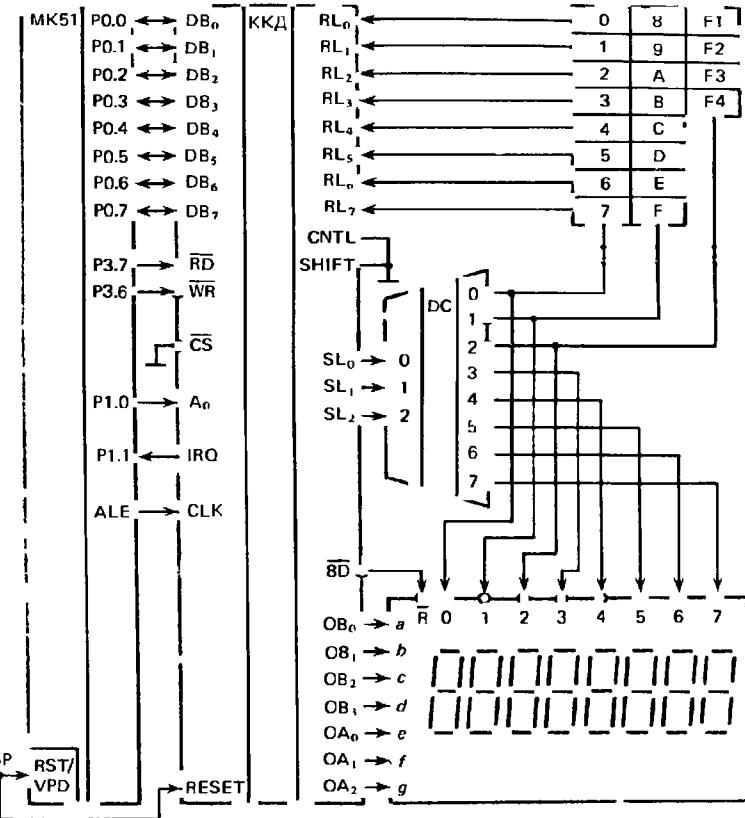


Рис. 7.11. Схема подключения ККД к МК51

частотой примерно 2 МГц с выхода ALE. Контроллер клавиатуры/дисплея в нашем примере обеспечивает ввод в МК кода нажатой клавиши (одной из 20) и поддерживает изображение на восьмипозиционном односторочном дисплее. Для сканирования клавиатуры и дисплея дополнительно используется инвертирующий дешифратор на восемь выходов. Диоды на выходе дешифратора необходимы для защиты от короткого замыкания между собой его выходов при одновременном нажатии нескольких клавиш.

Дисплей показан на схеме условно; предполагается, что он снабжен необходимыми буферными схемами для обеспечения требуемых токов нагрузки. Сигнал 0 на входе \bar{K} обеспечивает гашение всех индикаторов одновременно.

Матрица клавиш расположена таким образом, чтобы SCAN-код кла-

Примеры проектирования МК-устройств и систем

виши совпадал с двоичным кодом шестнадцатеричной цифры, нанесенной на клавишу. Входы SHIFT и CNTL заземлены. Клавиатура состоит из 16 цифровых клавиш (0–F) и 4 клавиш управления (функциональных), коды которых больше 0FH.

Для того чтобы настроить контроллер клавиатуры/дисплея на определенный режим работы, основная программа МК-системы должна загрузить в него управляющие слова инициализации (УСИ) и в требуемый момент выдать управляющее слово операции (УСО). Ниже приводится программа инициализации, которая настраивает контроллер на режим работы с восьмипозиционным дисплеем, распознавания одиночного нажатия клавиш и сканирования знакомест дисплея счетчиком:

```
;ВЕРСИЯ ДЛЯ МК51
;ИНИЦИАЛИЗАЦИЯ КОНТРОЛЛЕРА КЛАВИАТУРЫ/ДИСПЛЕЯ
SETB P1.0      ;УСТАНОВКА А0
MOV A:#00H     ;ЗАГРУЗКА УПРАВЛЯЮЩЕГО СЛОВА
MOVX R0,A      ;ИНИЦИАЛИЗАЦИИ (УСИ) В КОНТРОЛЛЕР
MOV A:#(20H+20) ;ЗАГРУЗКА КОЭФФИЦИЕНТА ДЕЛЕНИЯ
MOVX R0,A      ;СИ. РЕСИГНАЛА
```

Коэффициент деления входного синхросигнала (CLK) необходимо установить таким образом, чтобы внутренняя опорная частота контроллера получилась около 100 кГц. Так как частота сигнала ALE равна примерно 2 МГц, то коэффициент деления выбирается равным 20. Содержимое регистра R0 не имеет значения, так как в нашем примере контроллер "закрывает" собой все адресное пространство ВИД. После инициализации контроллер будет работать параллельно с МК и избавит его от необходимости выполнения программ опроса клавиатуры и поддержания изображения на дисплее.

Для ввода кода нажатой клавиши в МК необходимо выполнить следующие действия:

```
;ВЕРСИЯ ДЛЯ МК51
;ВВОД КОДА КЛАВИШИ
WAIT: JNB P1.1,WAIT    ;ОЖИДАНИЕ НАЖАТИЯ КЛАВИШИ
SETB P1.0      ;УСТАНОВКА А0
MOV A:#40H     ;ПОДГОТОВКА ЧТЕНИЯ НУЛЕВОЙ
                ;ЯЧЕИКИ БУФЕРА КЛАВИАТУРЫ
MOVX R0,R0      ;ЗАГРУЗКА УПРАВЛЯЮЩЕГО СЛОВА
                ;ОНЕГРАНИЧИСТВО В КОНТРОЛЛЕР
CLR P1.0      ;СБРОС А0
MOVX R0,R0      ;ЧТЕНИЕ КОДА КЛАВИШИ ИЗ БУФЕРА
```

Как только контроллер клавиатуры/дисплея зафиксирует нажатие клавиши и определит ее SCAN-код (совпадающий с двоичным кодом шестнадцатеричной цифры клавиши), он сразу оповестит об этом МК, установив сигнал на выходе запроса прерывания (IRQ). После обнаружения сигнала 1 на входе P1.1 МК может прочитать код клавиши из контроллера. Но для этого надо предварительно загрузить в контроллер УСО "Чтение", специфицирующее источник информации (буфер кодов клавищ) и адрес ячейки. После этого при A0 = 0 можно вводить код клавиши в МК.

8.1.

Устройство формирования звуковых сигналов

При помощи МК можно достаточно просто воспроизводить (генерировать) различные звуковые сигналы, причем не только "бип-сигналы", но и разнообразные музыкальные фразы. Воспроизведение мелодии сводится к генерации последовательностей импульсных сигналов определенной звуковой частоты (нот) в течение определенных интервалов времени. Звуковой сигнал характеризуется двумя параметрами: частотой и длительностью. В табл. 8.1 приводятся значения частот для 12 нот, начиная с ноты ЛЯ первой октавы. Отношение соседних частот для равномерно темпированного музыкального строя выражается иррациональным числом $\sqrt[12]{2}$. От октавы к октаве частота звукового сигнала меняется ровно в 2 раза.

Таким образом, генерация одной ноты в музыкальной фразе может быть осуществлена двумя рассмотренными ранее процедурами: выдачей импульсного сигнала и задержкой. Допустим, что в наборе воспроизводимых микроконтроллером музыкальных фраз длительность звучания ноты может принимать только значения 1, 1/2, 1/4 и 1/8 с.

Кодированная запись музыкальной фразы может быть, например, такой, как показано на рис. 8.1. Схема процедуры генерации музыкальной фразы (MUSIC) представлена на рис. 8.2. Вначале производится загрузка регистра-указателя (DPTR) начальным адресом закодированной нотной записи. Затем выполняется настройка обоих таймеров на 16-битный формат, режим 1. Далее производится обращение к нотной записи, которая располагается в ВИД (для нашего примера), за очередным кодом. После этого проверяется на нуль прочитанный код, т.е. анализируется, достигнут или нет конец записи. При достижении конца нотной записи осуществляется выход из процедуры. Если же код не равен нулю, то подготавливаются данные для процедуры генерации ноты (SOUND). Длительность интервала звучания ноты при этом удваивается, так как процедура SOUND выдерживает интервал, равный (1/16) D с. Далее включается в работу процедура генерации ноты, после чего осуществляется переход к следующему коду нотной записи.

Таблица 8.1. Значения частот основных нот

Нота	Частота, Гц	Длительность полупериода, мкс	Код ноты
ЛЯ ₁	440	1136	0
ЛЯ ₁ #	466,2	1073	1
СИ ₁	494	1012	2
ДО ₂	523	956	3
ДО ₂ #	554,36	902	4
РЕ ₂	588	850	5
РЕ ₂ #	622,25	804	6
МИ ₂	660	758	7
ФА ₂	698	716	8
ФА ₂ #	739,99	677	9
СОЛЬ ₂	784	638	A
СОЛЬ ₂ #	830,65	602	B

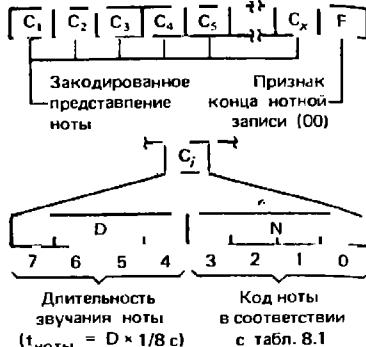


Рис. 8.1. Принятый вариант кодирования нотной записи

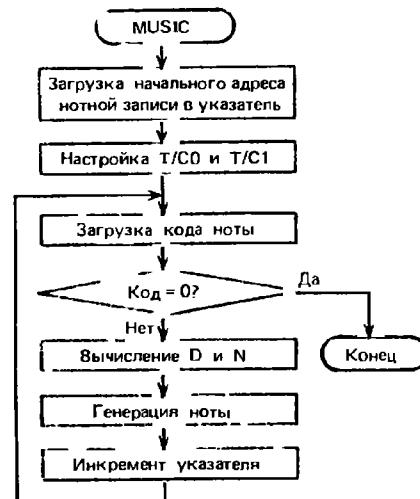


Рис. 8.2. Блок-схема алгоритма генерации музыкальной фразы

Схема процедуры генерации ноты приведена на рис. 8.3. Она ориентирована на MK51, так как использует два таймера: T/C1 – для формирования частоты сигнала и T/C0 – для формирования интервала звучания ноты. Звуковой сигнал формируется на выводе P1.0 путём его инверти-

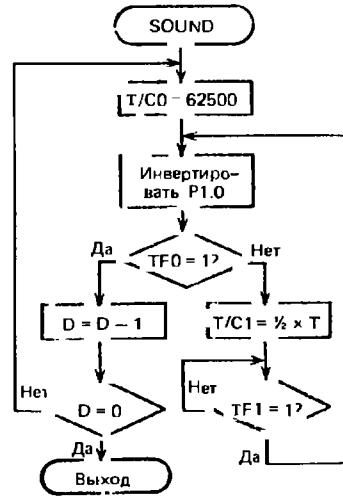


Рис. 8.3. Схема процедуры генерации ноты



Рис. 8.4. Нотная запись песенки

рования каждые 0,5T, где T – период звучания ноты. Задержка длительностью 0,5T реализуется на T/C1 (к выходу P1.0 через усилитель подключен громкоговоритель).

Вначале осуществляется загрузка в T/C0 числа 62 500, что обеспечивает получение задержки в 1/16 с. При переполнении T/C0 производится уменьшение счетчика интервала D. При достижении условия D = 0 генерация прекращается.

Процедура SOUND имеет два входных параметра: длительность интервала генерации ноты D, измеряемая в единицах, равных 1/16 с (регистр R7), и удвоенный код ноты N (регистр R6).

Текст программы генерации музыкальной фразы, ориентированной на MK51, приводится ниже. Программа снабжена подробными комментариями. Для определения длительности полупериода сигнала соответствующей ноты используется метод табличного преобразования. В таблице с именем TAB хранятся значения полупериодов соответствующих нот в микросекундах, уменьшенные на 16. Такое уменьшение необходимо, так как с момента инвертирования сигнала на выходе P1.0 и до загрузки T/C1 проходит импульс 16 мкс, необходимые для выполнения команд, расположенных между метками S1 и S0.

Если в конкретном применении МК закодированную нотную запись оказывается удобнее хранить в постоянной памяти программ, то программу придется немножко модифицировать. Такую модификацию предлагается выполнить читателю самостоятельно.

Закодированная запись новогодней английской песенки, нотная запись которой представлена на рис. 8.4, занимает 50 байтов (49 нот и признак конца записи) и выглядит следующим образом:

```

SONG: DB 27H,27H,47H,27H,27H,47H,27H,2AH
      DB 23H,25H,67H,28H,28H,28H,28H,28H
      DB 27H,27H,27H,25H,25H,27H
      DB 45H,4AH
      DB 27H,27H,47H,27H,27H,47H,27H,2AH
      DB 23H,25H,67H,28H,28H,28H,28H,28H
      DB 27H,27H,27H
      DB 2AH,28H,27H,25H,83H
      BB 00H :ЗНАК-ТЕРМИНАТОР "КОНЕЦ ТЕКСТА"

;ПРОГРАММА ВОСПРОИЗВЕДЕНИЯ МУЗЫКАЛЬНОЙ ФРАЗЫ,
; ЗАКОДИРОВАННАЯ ЗАПИСЬ КОТОРОЙ НАХОДИТСЯ

; ВО ВНЕШНЕЙ ПАМЯТИ ПРОГРАММ ПО АДРЕСУ SDNG

```

```

MUSIC: MOV DPTR, $SONG :ЗАГРУЗКА НАЧАЛЬНОГО АДРЕСА SONG
       MOV TMOD, $11H :ИНИЦИАЛИЗАЦИЯ T/CO И T/C1
M1:   MOVX A, @DPTR :ЗАГРУЗКА КОДА НОТЫ
       JZ EXIT :ВЫХОД, ЕСЛИ КОД РАВЕН НУЛЮ
       ;Вычисление значения в и н
       ANL A, OFH :Выделение тетраграмм из байта
       RL A :УДОБНОЕСТЬ КОДА Н
       MOV R6, A
       MOVX A, @DPTR :ПОВТОРНАЯ ЗАГРУЗКА КОДА НОТЫ
       ANL A, $0F0H :Выделение тетраграмм в из байта
       SWAP A
       RL A :УДОБНОЕСТЬ КОДА Н
       MOV R7, A
       ;Процедура ГЕНЕРАЦИИ ОДНОЙ НОТЫ (R6=N, R7=D)
SOUND: CLR TCON.4 :СТОП T/CO, ИНИЦИАЛИЗАЦИЯ
       MOV TL0, $LOW(NDT(62500)+1)
       MOV TH0, $HIGH(NDT(62500)+1)
       SETB TCON.4 :СТАРТ T/CO
       SJL CPL F1, 0 :ИНВЕРСИЯ БИХДА F1.0
       JBC TCON.5, S2 :ПРОВЕРКА ФЛАГА ПЕРЕПОЛНЕНИЯ T/CO
       ;Задержка на 1/2 периода
       CLR TCON.6 :СТОП T/C1
       MOV A, R6 :ФОРМИРОВАНИЕ В АККУМУЛЯТОРЕ
       ADD A, $TAB(-x+2) :РАЗНОСТИ АДРЕСОВ СТАРШЕГО
                           :БАЙТА ПОЛУПЕРИОДА НОТЫ
                           :И КОМПАНИ ОБРАЩЕНИЯ К ТАБЛИЦЕ ТАВ
                           :ПОБРАЗНЕНИЕ К ТАБЛИЦЕ ТАВ ЗА СТАРШИМ
                           :БАЙТОМ ПОЛУПЕРИОДА НОТЫ
       MOVC A, BA+PC :ЗАГРУЗКА СТАРШЕГО БАЙТА T/C1
       MOV TH1, A
       MOV R6
       ADD A, $TAB(-x+2+1)
       MOVC A, BA+PC :ПОБРАЗНЕНИЕ К ТАБЛИЦЕ ТАВ ЗА МЛАДШИМ
                           :БАЙТОМ ПОЛУПЕРИОДА НОТЫ
       MOV TL1, A :ЗАГРУЗКА МЛАДШЕГО БАЙТА T/C1
       SETB TCON.6 :СТАРТ T/C1
       SJMP TCON.7, S1 :ПРОВЕРКА ФЛАГА ПЕРЕПОЛНЕНИЯ T/CO
S01:   SJMP S0

       ;АКРЕМЕНТ СЧЕТЧИКА ДЛИННОСТИ ЗВУКА
S2:   DJNZ R7, SOUND :ПОВТОРЕНИЕ ПОКА D НЕ РАВНО НУЛЮ
       ;ПЕРЕХОД К СЛЕДУЮЩЕЙ НОТЕ
       INC R7 :ИНКРЕМЕНТ УКАЗАТЕЛЯ КОДОВ
       SJMP M1 :ПЕРЕХОД К НАЧАЛУ ПРОЦЕДУРЫ

```

ТАБЛИЦА ПОЛУЖЕРИОДОВ НОТ		
ТАБ:	DN	1136-16
	DN	1073-16
	DN	1012-16
	BN	956-16
	BN	902-16
	DN	850-16
	DN	804-16
	DN	758-16
	DN	716-16
	BN	677-16
	DN	638-16
	DN	602-16
EXIT:	...	:ВЫХОД ИЗ ПРОЦЕДУРЫ

Хотя принятый способ кодирования нот (рис. 8.1) обеспечивает получение компактного объектного модуля, сам процесс кодирования нотной записи должен производиться полностью вручную, что сопряжено с ошибками. В тех применениях, где основным критерием качества МК-устройства является не минимальная емкость памяти, занимаемая прикладной программой, а легкость и простота кодирования музыкальных фраз, целесообразно использовать иной способ кодирования нот, при котором каждая нота определяется двумя байтами: байтом длительности ноты и байтом звуковой частоты. Такой способ кодирования позволит использовать символические имена нот (РЕ, МИ и т.п.), предварительно присвоив им соответствующие значения генерируемых частот. При этом каждый элемент нотной записи будет состоять из двух байтов. Первый байт D будет выражать длительность ноты. Так как длительность задается правильными дробями (1/2, 1/4, 1/8 и 1/16), то при кодировании удобно указывать, например, только знаменатель дроби. Преобразование знаменателя в количество шестнадцатых долей секунды, как этого требует процедура SOUND, может быть легко выполнено делением 16 на D.

Второй байт кодового элемента представляет код (номер) ноты согласно табл. 8.1. Например, кодовый элемент (08) (05) будет соответствовать ноте РЕ₂ с длительностью 1/8 с. Признаком конца кодовой записи может служить, как и раньше, элемент (D), равный нулю.

Для определения символьических имен нот в программу необходимо вставить следующие псевдокоманды:

ЛЯ	EQU	0	:СИМВОЛИЧЕСКОМУ ИМЕНИ "ЛЯ"
СИ	EQU	1	:ПРИСВОИТЬ КОД НОТЫ ЛЯ1 (0)
ХО	EQU	2	:СИ1
ДО	EQU	3	:АО2
ДО-	EQU	4	:АО2#
РЕ	EQU	5	:РЕ2
РЕ-	EQU	6	:РЕ2#
НИ	EQU	7	:МИ2
ФА	EQU	8	:ФА2
ФА-	EQU	9	:ФА2#
СОЛЬ	EQU	DAH	:СОЛЬ2
СОЛЬ-	EQU	DBH	:СОЛЬ2#

Данные псевдокоманды могут располагаться в любом месте программы, например в самом начале.

Возможно, что в конкретном применении удобнее окажется вариант процедуры воспроизведения мелодии, оформленный в виде подпрограммы. Текст программы претерпит небольшие изменения и примет следующий вид:

```
;ПОДПРОГРАММА ВОСПРОИЗВЕДЕНИЯ МЕЛОДИИ
; В ВХОДНОМ ПАРАМЕТР - НАЧАЛЬНЫЙ АДРЕС КОДОВОЙ ЗАПИСИ
; В ВЫХОДНОМ ПАРАМЕТР - АДРЕС ЧЕРЕЗ DPTR
MUSIC: MOV TMOD,$11H      ;НАСТРОЙКА T/CO, T/C1
        CLR A          ;(A) ---- 0
        MOVC A,@A+DPTR   ;ЗАГРУЗКА D
M1:    JZ EXIT            ;ВЫХОД, ЕСЛИ D=0
        MOV B,A          ;(B) ---- 0
        MOV A,$16         ;(A) ---- 16
        DIV AB           ;(A) ---- 16/U
        MOV R7,A          ;(R7) ---- АМПИЛЬНОСТЬ
        INC DPTR          ;ПЕРЕХОД К СЛЕДУЮЩЕЙ НОТИ
        CLR A          ;(A) ---- 0
        MOVC A,@A+DPTR   ;?
        RL A          ;УДВОЕНИЕ КОДА НОТИ, ТАК КАК АМПИЛЬНОСТЬ
        MOV R6,A          ;ПОЛУПЕРИОДА НОТИ ЗАДАЕТСЯ ДВУМЯ БАЙТАМИ
;ПРОЦЕДУРА SOUND БЕЗ ИЗМЕНЕНИЙ
SOUND: ...
...
S2:    DJNZ R7,SOUND       ;ПОВТОРЯТЬ, ПОКА НЕ ОБНУЛЯЕТСЯ
        INC DPTR          ;СЧЕТЧИК АМПИЛЬНОСТИ
        SJMP M1             ;ПЕРЕХОД К СЛЕДУЮЩЕМУ ЭЛЕМЕНТУ КОДА
EXIT:  RET
;ТАБЛИЦА ПОЛУПЕРИОДОВ НОТ БЕЗ ИЗМЕНЕНИЙ
TAB:  ...
...

```

Используя символические имена, определенные выше, можно легко составить новую кодированную нотную запись новогодней песенки:

```
SONG: BB 4#МИ+4#МИ+2#МИ+4#МИ
       4#МИ+2#МИ+4#МИ+4#СОЛЬ
DB 4#ДО+4#РЕ+1#МИ+4#ФА
DB 4#ФА+4#ДО+4#ДО+4#ФА
DB 4#МИ+4#МИ+4#МИ+4#МИ
DB 4#ФЕ+4#РЕ+4#МИ+2#РЕ
BB 2#СОЛЬ
DB 4#МИ+4#МИ+2#МИ+4#МИ
DB 4#МИ+2#МИ+4#МИ+4#СОЛЬ
DB 4#ДО+4#РЕ+1#МИ+4#ФА
DB 4#ФА+4#ДА+4#ФА+4#ФА
DB 4#МИ+4#МИ+4#МИ+4#СОЛЬ
DB 4#ФА+4#МИ+4#РЕ+1#АО
DB 0                                     ;КОНЕЦ
```

8.2.

Кодовый замок зуммерного типа

Постановка задачи. Пусть требуется разработать кодовый замок зуммерного типа, обеспечивающий доступ в помещение лаборатории только лиц, знающих код замка. Устройство должно быть реализовано на MK51.

В нормальном состоянии замок закрыт, и вход в лабораторию невозможен. Чтобы открыть замок, необходимо сначала один раз нажать кнопку. После этого МК инициирует три световые вспышки длительностью по 6 с с паузой в 1 с. Во время каждой вспышки необходимо определенное число раз нажать ту же кнопку. Таким образом, с помощью единственной кнопки организуется ввод трех секретных чисел, являющихся ключом для отпирания замка. Если набранный код совпадает с эталоном, хранящимся в МК, то замок открывается и загорается сигнальная лампа. В этом случае можно повернуть ручку и открыть дверь. После входа в лабораторию и закрытия двери лампа гаснет и МК переходит в начальное состояние.

При попытке открывания двери с неправильно набранным кодом МК должен включить сигнал тревоги: перемежающиеся световой и звуковой сигналы длительностью по 1 с каждый. Выключение сигнала тревоги происходит после прекращения попыток открыть дверь при неправильном наборе кода, но не ранее чем через 10 с после включения. После отключения сигнализации можно повторить попытку открыть дверь.

Анализ задачи. Для решения поставленной задачи необходимо наличие специальных датчиков и исполнительных механизмов. Требуются следующие датчики: кнопка для ввода чисел, датчик поворота ручки двери (дверь имеет дополнительную защелку, открываемую ручкой), датчик закрытия двери. Кроме того, необходимо наличие исполнительных механизмов: громкоговорителя или сирены для подачи звукового сигнала тревоги; сигнальной лампы и соленоида для втягивания ригеля (задвижки) кодового замка.

Разработка схемы устройства. Схема контроллера замка может быть, например, такой, как показано на рис. 8.5. Кроме MK51 потребуется RC-цепь для формирования сигнала сброса при включении питания и кварцевый резонатор 12 МГц. Если предположить, что для хранения прикладной программы используется РПП, то на вход отключения РПП (EA) подается уровень 1.

Связь MK51 с датчиками и исполнительными механизмами можно обеспечить через один из имеющихся портов, а незадействованные порты могут быть впоследствии использованы при расширении функциональных возможностей контроллера.

Из-за низкой нагрузочной способности выходов МК для всех исполнительных механизмов потребуются усилители мощности, показанные на рис. 8.5. Интерфейс МК построен таким образом, что для включения исполнительного механизма необходимо на соответствующем выходе МК сформировать сигнал низкого уровня. При этом в начальном состоянии МК после включения электропитания все исполнительные механиз-

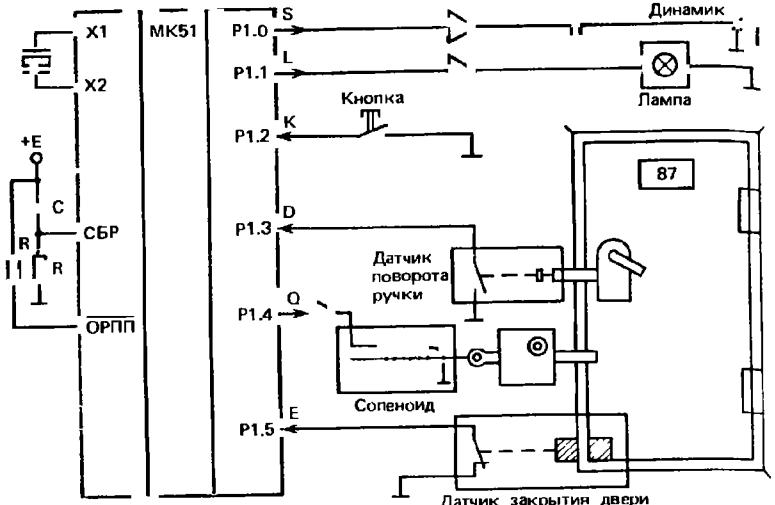


Рис. 8.5. Интерфейс контроллера кодового замка

мы оказываются в пассивном состоянии. Аналогично и все датчики сообщают о контролируемом событии подачей сигнала низкого уровня на соответствующий вход МК.

Инженерная интерпретация задачи. На основе изучения поставленной задачи можно выделить пять состояний, в которых может находиться МК-устройство в процессе работы. Анализ работы МК-устройства иногда удобно проводить с использованием аппарата автоматных графов.

Граф контроллера кодового замка показан на рис. 8.6.

В начальном состоянии (НАЧ) контроллер оказывается после включения питания. Все исполнительные механизмы при этом выключены, замок закрыт. В этом состоянии контроллер может находиться неопределенное время, пока не будет нажата кнопка. После нажатия кнопки ($K = 0$) осуществляется переход в состояние ввода и сравнения кода (ВСК). В состоянии ВСК контроллер инициирует засветку лампы и производит ввод трех контрольных цифр. Если коды не совпадут, то осуществляется возврат в состояние НАЧ, иначе выполняется переход в состояние ОТКР (открытие двери). В состоянии ОТКР зажигается сигнальная лампа, срабатывает соленоид открывания замка и контроллер ожидает открытия двери.

После закрытия двери (состояние ОЖ ЗАКР) контроллер переходит в начальное состояние. Если в состояниях НАЧ и ВСК контроллер обнаружит попытку открытия двери (поворот ручки защелки, $D = 0$), то осуществляется переход в состояние тревожной сигнализации (СИГНАЛ).

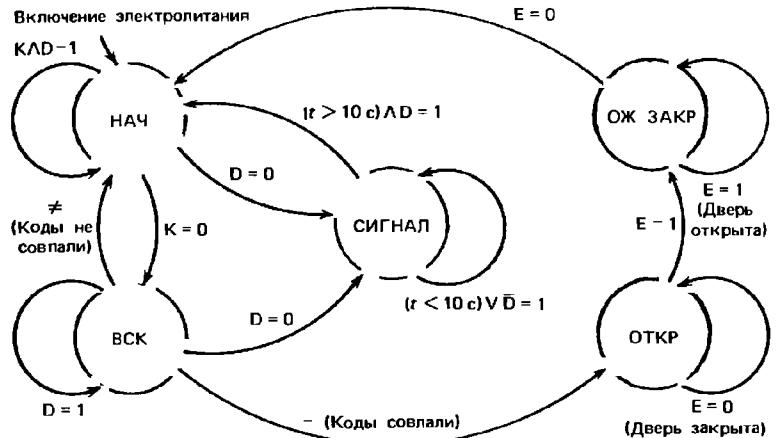


Рис. 8.6. Граф состояний контроллера кодового замка

Как видно, граф состояний позволяет (к сожалению, далеко не всегда) выполнить формализованное описание работы МК-устройства, которое не только помогает уточнить задачу, но и облегчает разработку схемы алгоритма ее решения.

Разработка блок-схемы алгоритма. На основе графа состояний разрабатывается недетализированная блок-схема алгоритма (рис. 8.7). Состояние НАЧ реализуется операторами 1–2, состояние ВСК – операторами 3–10, состояние ОТКР – операторами 11–13, состояние ОЖ ЗАКР – операторами 14–16, а состояние СИГНАЛ – операторами 17–24. Оператор 5 является наиболее сложным для программной реализации. Как видно, из него даже имеется второй выход для отслеживания попытки открыть дверь при неправильно набранном коде ($D = 0$). В блок-схеме отсутствуют операторы ввода. С целью упрощения графического образа алгоритма они заменены операторами проверки. Это упрощение не искажает смысла процедур, так как в MK51 имеется возможность совмещать в одной команде операцию ввода и операцию проверки бита.

Разработка прикладной программы. Текст исходной программы работы контроллера кодового замка является основным документом для последующих процедур трансляции, отладки и загрузки объектных кодов в РПП, в которых возможно использование средств автоматизации разработки прикладного программного обеспечения.

Исходный текст программы SEZAM, составленный в полном соответствии со схемой алгоритма, приводится ниже.

Для реализации временных задержек используются две подпрограммы. Подпрограмма DELAY реализует задержку 50 мс с использованием T/C0. Подпрограмма ONESEC реализует задержку длительностью в 1 с. Для большей наглядности введены символические имена линий порта 1,

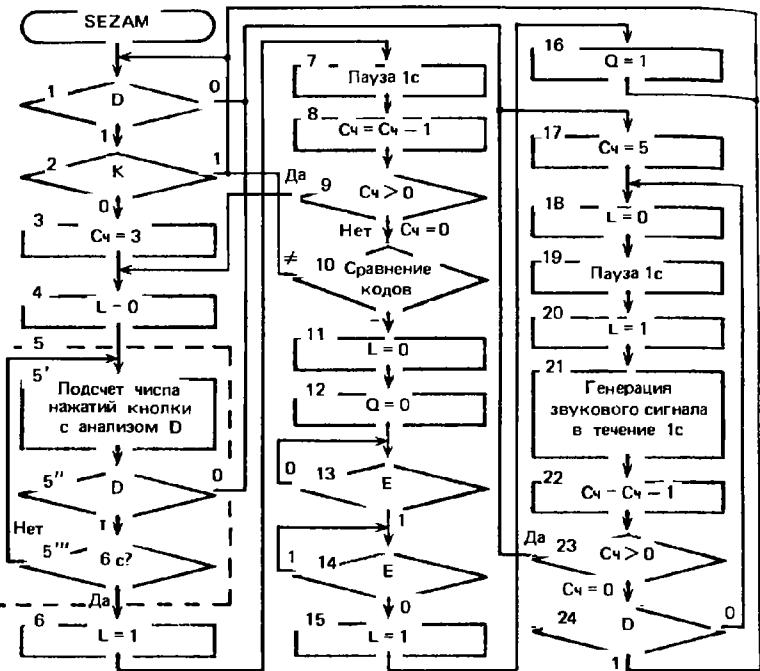


Рис. 8.7. Блок-схема алгоритма работы контроллера кодового замка

используемых в работе контроллера замка. После включения питания по сигналу сброса выключаются все исполнительные механизмы, в РУС заносится код 071H, выбирается банк регистров 0 и запрещаются все прерывания. Допустим, что секретная (отпирающая) последовательность цифр образована цифрами 7, 3 и 5, коды которых задаются в тексте программы.

Опрос кнопки в состоянии ВСК ведется дискретно (один раз в 50 мс), тем самым устраняется дребезг ее контактов. Введенные цифры кода замка сохраняются в РПД, начиная с адреса 20H (CODE_EX). Генерация звукового сигнала реализуется чисто программным способом, для чего используется трехкратно вложенный цикл. Внутренний цикл задает длительность импульса и паузы примерно по 500 мкс. Тем самым получается периодический сигнал с частотой около 1 кГц и скважностью 2. Следующий цикл (счетчик R4) обеспечивает временной интервал генерации 0,2 с, а внешний цикл (счетчик RS) ловит время звучания до 1 с.

Объектные коды программы должны быть записаны в РПП, начиная с нулевой ячейки.

Длина программы составляет 121 байт; это значит, что МК-устройство

допускает значительное расширение своих функциональных возможностей. (Например, можно организовать через УАПИ МК51 связь множества котовых замков с центральным диспетчером для сообщения о количестве человек, находящихся в лабораториях, для экстренных сообщений о многократных попытках несанкционированного входа в лабораторию, для получения от диспетчера нового значения кода и т.п.).

```

***** ОПРЕДЕЛЕНИЕ СИМВОЛИЧЕСКИХ ИМЕН *****
; ОПРЕДЕЛЕНИЕ СИМВОЛИЧЕСКИХ ИМЕН БИТ ХОРТА 1
K DIT P1.2 #КОНОК
D BIT P1.3 #ВАТИК ПОВОРОТА РУЧКИ
E BIT P1.5 #ВАТИК ЗАКРЫТИЯ ДВЕРИ
Q DIT P1.4 #СОЛНЕЧНАЯ ЗАНКА
S BIT P1.0 #ВЫХОД НА ГРОМОГОВОРИТЕЛЬ
L DIT P1.1 #ВЫХОД НА ЛАЙНУ

CODE_EX DATA 20H #НАЧАЛО ОБЛАСТИ КОДОВ

***** ПРОГРАММА SEZAM *****

URG 0 #НАЧАЛЬНЫЙ АДРЕС ПРОГРАММЫ
REPEAT: JNB D,ALARM #ПЕРЕХОД, ЕСЛИ D=0
        JB K,REPEAT #ПОВТОРИТЬ, ЕСЛИ K=0
        JNB K,x #ОПИСАНИЕ ОТПАТИЯ КНОПКИ

#ВВОД КОДОВ ОТ КНОПКИ
MOV R6,$3
MOV R0,$CODE_EX #СЧЕТЧИК 1 (- 3
#НАЧАЛЬНЫЙ АДРЕС ОБЛАСТИ
#ВВЕДЕНИИ КОДОВ
CYCLE: CLR A #СБРОС РЕГИСТРА КОДА
        CLR L #ВКЛЮЧИТЬ ЛАМПУ
        MOV R5,$120 #СЧЕТЧИК 2 (- 120
        LOOP: ACALL DELAY #ЗАМЕРКА 0,05 СЕКУНДЫ
                JNB D,ALARM #ПЕРЕХОД, ЕСЛИ K=0
                JNB K,KZERO #ФИНАЛ ОЖИДАНИЯ НАЖАТИЯ
                SJMP PAUSE #ПЕРЕХОД, ЕСЛИ ИСТЕКЛИ 6 СЕКУНДЫ
KZERO: ACALL DELAY #СЧЕТЧИК 1 (- 3
                JB K,ACCEPT #ПЕРЕХОД, ЕСЛИ КНОПКА D НАЖАТА
                DJNZ R5,LOOP #ОПИСАНИЕ ОЖИДАНИЯ
                SJMP PAUSE #ПЕРЕХОД, ЕСЛИ ИСТЕКЛИ 6 СЕКУНДЫ
ACCEPT: INC A #ИНКРЕМЕНТ СЧЕТЧИКА НАЖАТИЯ
        SJMP LOOP #СОХРАНЕНИЕ ЧИСЛА НАЖАТИЙ
PAUSE: MOV R0,A #ВЫКЛЮЧЕНИЕ ЛАМПЫ
        INC R0 #ЗАМЕРКА 1 СЕКУНДЫ
        SETB L #ВЫКЛЮЧЕНИЕ ЛАМПЫ
        ACALL ONESEC #ЗАМЕРКА 1 СЕКУНДЫ
        DJNZ R6,CYCLE #ПОВТОРИТЬ 3 РАЗА

#СРАВНЕНИЕ ВВЕДЕНИИ ЧИСЕЛ С ЗАПОННЫМИ БАЙТАМИ
MOV R0,$CODE_EX
CJNE R0,$7,REPEAT #СРАВНЕНИЕ СТАРШИХ ЦИФР
INC R0
CJNE R0,$3,REPEAT #СРАВНЕНИЕ СРЕДНИХ ЦИФР
INC R0
CJNE R0,$5,REPEAT #СРАВНЕНИЕ МЛАДШИХ ЦИФР

#ОТКРЫТИЕ ЗАНКИ
CLR L #ВКЛЮЧЕНИЕ ЛАМПЫ
CLR D #ОТКРЫТИЕ ЗАНКИ
JNB E,> #ОЖИДАНИЕ ОТКРЫТИЯ ДВЕРИ

```

ЗАКРЫТИЕ ЗАМКА	
JR E,X	ЗАКРЫТИЕ ЗАМКА
SETB L	ВЫКЛЮЧЕНИЕ ЛАМПЫ
SETB O	ЗАКРЫТИЕ ЗАМКА
SJMP REPEAT	ВОЗВРАТ К НАЧАЛУ

ГРЕБОЖНАЯ СИГНАЛИЗАЦИЯ

ФОРМИРОВАНИЕ СВЕТОВОГО СИГНАЛА	
ALARM: NOV R6+5	!СЧЕТЧИК) <-- 5
L3: CLR L	!ВЫКЛЮЧЕНИЕ ЛАМПЫ
CALL ONESEC	!ПАУЗА 1 СЕКУНДА
SETB L	!ВЫКЛЮЧЕНИЕ ЛАМПЫ

ФОРМИРОВАНИЕ ЗВУКОВОГО СИГНАЛА	
MUV R5+5	:СЧЕТЧИК ПОВТОРЕНИЯ
L2: NOV R4+200	:ИНТЕРВАЛ 0,2 СЕКУНДЫ
L1: NOV R3+248	:ПЕРИОД ПРИБЛИЖЕНИЕ РАВЕН
; 1 МИЛЛИСЕКУНДА	
CPL S	:ИНВЕРТИРОВАНИЕ S
DJNZ R3, Y	:ПОЛНОТА ИНВЕРСИИ
DJNZ R4,L1	:ПРИМЕРНО 0,5 МИЛЛИСЕКУНДЫ
DJNZ RS,L2	:СРЕДНИЙ ЦИКЛ
DJNZ R6,L3	:ВНЕШНИЙ ЦИКЛ
JB REPEAT	!ВОЗВРАХА К НАЧАЛУ, ЕСЛИ D=1
NOV R6+1	!ПРОДОЛЖЕНИЕ ТРЕВОГИ
SJMP L3	

***** ПОДПРОГРАММЫ *****	
; ПОДПРОГРАММА ЗДЕРЖКА 50 МС. ИСПОЛЬЗУЕТСЯ Т/СО;	
; ВОГРЕЧИСТЬ НЕ БОЛЕЕ 2 МС	
DELAY: NOV	TMOB, #0001B ;НАСТРОЙКА Т/СО
NOV	TH0, #HIGH(NOT(50000-16))
NOV	TLO, #LOW(NOT(50000-16))
SETB	TC0N, 4 ;СТАРТ Т/СО
DEL_W: JNB	TC0N, 5, DEL_W ;ФОРМИРОВАНИЕ
AM1	TC0N, #NOT(30H) ;СТОП Т/СО, СЕРОС ТФ0
KET	;ВОЗВРАТ
; ПОДПРОГРАММА ЗДЕРЖКА 1 СЕКУНДЫ,	
; ИСПОЛЬЗУЕТСЯ: ПОДПРОГРАММА DELAY И РЕГИСТР R7	
; ВОГРЕЧИСТЬ НЕ БОЛЕЕ 123 МС	
ONESEC: NOV	R7, #20 ;СЧЕТЧИК ЦИКЛОВ
SEC_W: ACALL	DELAY ;ЗДЕРЖКА 50 МС
DJNZ	R7, SEC_W ;ОРГАНИЗАЦИЯ ЦИКЛА
RET	;ВОЗ...

8.3.

Отладочный модуль на основе МК48

Назначение устройства. Отладочный модуль предназначается главным образом для приобретения навыков программирования и отладки программ для МК48. Он может быть также использован как макет (прототип) реальной системы управления для отладки ее прикладного программного обеспечения совместно с объектом управления в реальном масштабе времени.

Состав технических средств. Простейший отладочный модуль должен содержать следующие средства:

постоянную память с программой Монитор, обеспечивающей управление всей системой и прежде всего взаимодействие с оператором; оперативную память для занесения программ, данных и создания буферов Монитора;

простейшую клавиатуру для загрузки кодов и взаимодействия с оператором; 16 цифровых клавиш (от 0 до F), несколько управляющих и клавишу системного сброса;

простой дисплей, обычно односторонний (на основе семисегментных индикаторов), который используется для визуального контроля вводимой информации и отображения данных.

Кроме того, отладочный модуль может иметь собственный источник электропитания и (в некоторых случаях) место на печатной плате типа "терка" для монтажа пользовательских средств связи с объектом управления.

Логическая организация. Функциональная схема простейшего отладочного модуля приведена на рис. 8.8. Предполагается, что Монитор расположен в РПП МК48, а БИС ОЗУ (например, КР537РУ8) совмещает в себе функции внешней памяти программ и данных. Такое совмещение обеспечивается объединением управляющих сигналов чтения ВПП (PSEN) и чтения ВПД (RD). Внешняя память имеет страничную организацию (8 страниц по 256 байт). Номер страницы при обращении к внешней памяти задается на младших линиях порта 2 (P2.0-P2.2).

При исполнении программ пользователю обращение к ОЗУ осуществляется с помощью сигнала выборки PSEN. Старший бит адреса (A11) служит для селектирования ОЗУ и контроллера клавиатуры/дисплея (КР580ВД79), который входит в адресное пространство ВПД.

Адресное пространство (4096 адресов) имеет следующее распределение: РПП (Монитор) 000H ÷ 3FFH ; ВПД (программы и данные пользователя) 800H ÷ OFFH ; ККД (данные) 400H ÷ 4FFH . 600H ÷ 6FFH ; ККД (управление) 500H ÷ 5FFH , 700H ÷ 7FFH .

Контроллер клавиатуры/дисплея (ККД) закрывает адресное пространство ВПД размером в 1024 ячейки, хотя и имеет в своем составе всего два регистра (регистр данных и регистр управления/состояния), обращаться к которым можно по любому из представленных адресов. Обмен МК с ККД осуществляется по командам обмена (MOVX) или по командам ввода/вывода через порт BUS.

Для синхронизации работы ККД используется сигнал с выхода ALE с частотой 400 кГц. Выход сигнала прерывания из ККД (IRQ) соединен через инвертор с входом запроса прерывания МК. Это позволяет достаточно просто обнаружить факт нажатия клавиши, а также выполнять переход из программы пользователя в Монитор по прерыванию от клавиатуры, что полезно при зацикливании отлаживаемой программы. Клавиатура состоит из 16 цифровых клавиш и 4 управляющих. Цифровые клавиши расположены таким образом, что их SCAN-код совпадает с двоичным кодом соответствующей цифры. Такое решение позволяет избежать процедуры перекодировки. Для сканирования клавиатуры и дисплея используется дополнительный дешифратор на 8 выходов, что

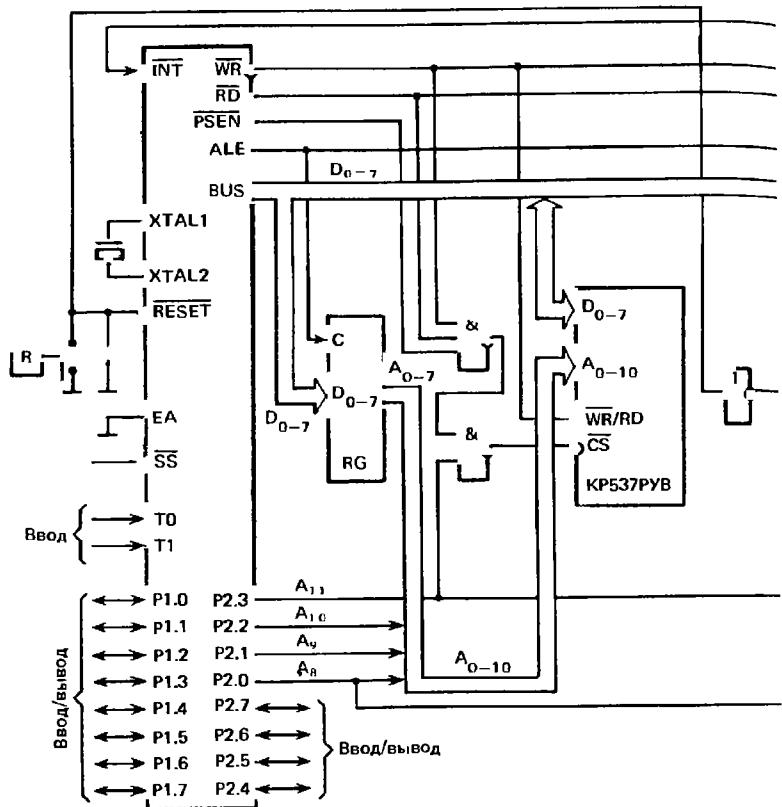
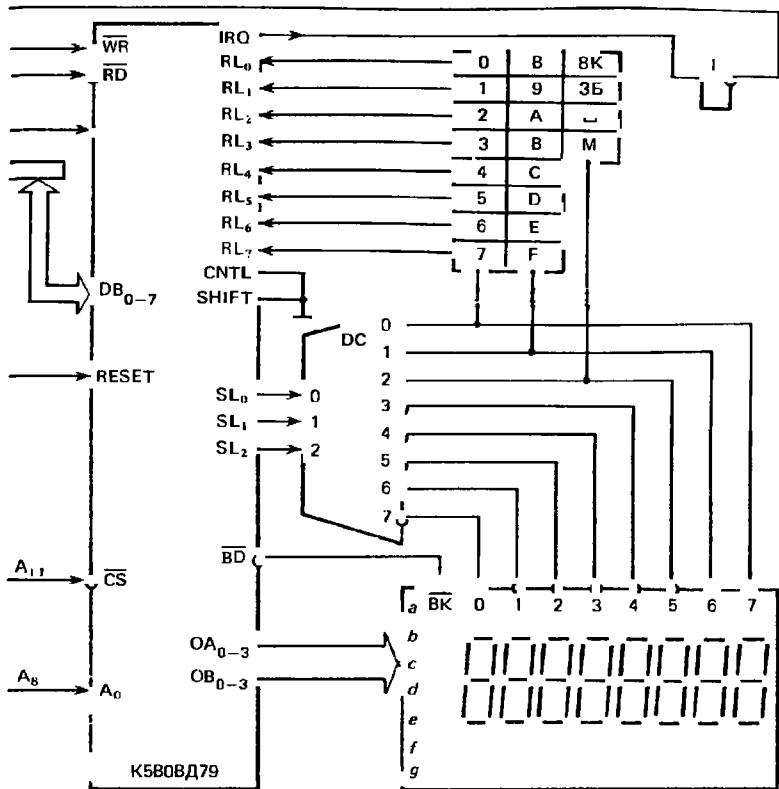


Рис. 8.8. Функциональная схема отладочного модуля для МК48

позволяет иметь в отладочном модуле 8-знаковый односторонний дисплей. Дисплей может быть построен на базе семисегментных индикаторов типа АЛС324 или аналогичных. Сигнал блокировки (BD) используется для гашения дисплея на время смены кодов на выходах ОА и ОВ и переключения позиций дисплея. Это позволяет избежать наложения символов в соседних позициях дисплея.

Порт BUS используется для построения шины данных (D₀₋₇). С помощью специального регистра-зашелки для младшей части адреса (RG) образуется шина адреса (A₀₋₁₀). Остальные линии ввода/вывода (порт 1, старшая часть порта 2, а также тестовые входы) могут быть задействованы по усмотрению пользователя.

Подключение конденсатора к входу сброса МК обеспечивает выработку сигнала сброса при включении питания. При необходимости произвести системный сброс без выключения питания (т.е. без потери данных



в РПД) можно воспользоваться клавишей сброса R. При сбросе происходит установка программного счетчика в состояние "все нули" и управление передается Монитору.

Монитор. Весь процесс взаимодействия пользователя с отладочным модулем обеспечивается Монитором ("игрушечной" операционной системой), расположенным в РПД. Взаимодействие пользователя с отладочным модулем называется мониторингом и сводится к выполнению специальных команд Монитора. Перечислим минимальный набор команд Монитора, позволяющих осуществлять ввод, редактирование и отладку прикладных программ.

Команда Е служит для просмотра и изменения содержимого ячеек ВПД. С ее помощью можно ввести программу или данные, исправить содержимое отдельных ячеек и просмотреть результаты выполнения программ. Команда может иметь следующий формат: Е XXX ВК, где

XXX — начальный адрес ВПП, т.е. число от 800Н до 0F1'FH. После ввода команды (ввод любой команды Монитора завершается нажатием клавиши ВК) на дисплее отобразится текущий адрес ячейки ВПП (начальное значение XXX) и ее содержимое. Далее можно изменить содержимое ячейки на новое и/или перейти к следующей по порядку ячейке, нажав клавишу **—**.

Команда I служит для просмотра и изменения содержимого ячеек РПД. Формат команды: **I XX ВК**, где XX — начальный адрес РПД, т.е. число от 00Н до 3FH. Выполняется аналогично команде E.

Команда Г (большая буква R не может быть отображена на семисегментном индикаторе) служит для просмотра и изменения содержимого внутренних регистров MK. Формат команды: **Г N ВК**, где N — номер банка регистров, т.е. число 0 или 1. Если число N опустить, то будет осуществляться обращение к регистрам A, PSW и PC. При выполнении команды на дисплее отображается номер регистраного банка, имя регистра и его содержимое. В остальном эта команда аналогична вышеуказанной.

Команда Р служит для просмотра и изменения содержимого портов P1 и P2. Формат команды: **P N ВК**, где N — номер порта (1 для P1 и 2 для P2). Для обращения к обоим портам необходимо выполнить две команды Р — по одной на каждыйпорт. При N = 0 на дисплей выводится состояние тестовых входов T0 и T1.

Команда G служит для запуска программы на исполнение. Формат команды: **G XXX — YYY ВК**, где XXX — начальный адрес программы, а YYY — адрес первой неисполнимой команды. Параметры XXX и

YYY могут быть опущены: в этом случае за начальный адрес принимается содержимое PC, а при отсутствии YYY программа будет запущена без задания точки останова.

К этим основным командам Монитора могут быть при желании до-

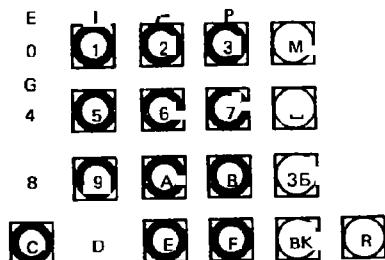


Рис. 8.9. Клавиатура отладочного модуля

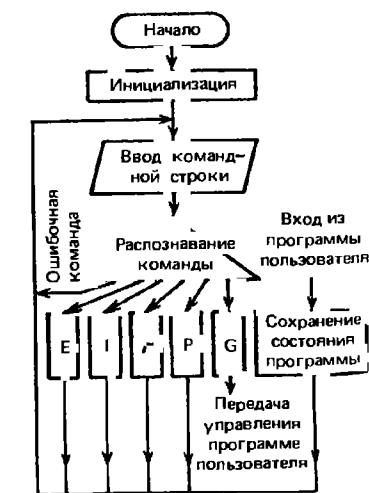


Рис. 8.10. Блок-схема Монитора

бавлены команды заполнения области памяти константами, сравнения областей памяти, поиска кода, тестовые команды и многие другие. Однако увеличение числа команд может привести к тому, что Монитор не сможет быть размещен в РПП.

Для ввода команд Монитора используется клавиатура (рис. 8.9). Перед началом ввода каждой команды Монитора необходимо нажать клавишу M; при этом дисплей очистится и в крайней левой позиции появится черточка, означающая, что Монитор готов к приему команды. Для ввода команды Монитора необходимо нажать клавишу под символическим именем команды. Таким образом, цифровые клавиши используются для двух целей: для ввода кодов команд и данных прикладной программы и для ввода команд Монитора. Цифровая клавиша имеет смысл имени команды только при нажатии ее сразу же после клавиши M. Для прекращения выполнения команд E, I и Г также необходимо нажать клавишу M.

При нажатии на клавишу R происходит системный сброс.

Укрупненная блок-схема Монитора приведена на рис. 8.10. Блок инициализации производит настройку ККД, отображение начального сообщения (например, HELLO) и присвоение начальных значений регистрам программы пользователя и внутренним переменным Монитора. Далее включается блок ввода командной строки. При вводе возможно стирание (забой) последнего введенного символа (клавиша 3Б). Конец командной строки распознается по нажатию клавиши ВК. Далее управление передается блоку распознавания команды, а затем процедуре выполнения соответствующей команды. Ошибочно набранная команда игнорируется. После выполнения команд E, I, Г и Р управление передается блоку ввода следующей команды. По команде G управление передается прикладной программе. При достижении точки останова или по прерыванию управление вновь возвращается Монитору. В этом случае включается блок сохранения состояния программы (т.е. всех регистров MK).

Команды E, I, Г и Р реализуются достаточно просто, команда G требует специальных пояснений.

Для обеспечения возврата в Монитор при достижении точки останова обычно используется простой способ, заключающийся в следующем. Перед запуском программы по адресу останова два байта программы заменяются командой CALL KEEP, где KEEP — начальный адрес блока сохранения состояния программы. При достижении точки останова команда CALL выполняется и управление передается по адресу KEEP, т.е. Монитору, причем адрес следующей команды прикладной программы оказывается в стеке (его необходимо уменьшить на 2, чтобы получить адрес точки останова). После этого осуществляется восстановление испорченной команды. Таким образом, содержимое PC сохраняется в стеке, а значения остальных регистров, сформированные программой пользователя, могут быть сохранены в ВПД (обычно в зоне старших адресов 0F00H — OFFFH). При загрузке содержимого регистров в

ВПД для кратковременного сохранения значений Р2.0–Р2.3, А и Т/С можно использовать свободные элементы стека.

Перед передачей управления программе пользователя по команде С следует установить точку останова (если нужно), восстановить значения всех регистров. Передача управления может быть выполнена командой RETR, которая восстановит PSW и разрешит прерывания. Управление будет передано в точку предыдущего останова или в любое другое место, если ранее был введен первый параметр (XXX) команды G.

Для передачи управления в точку KEEP по прерыванию необходимо, начиная с ячейки 003Н, расположить команду JMP KEEP.

Отдельные модули Монитора целесообразно оформить в виде подпрограмм (ввод кода клавиши, упаковка, распаковка, загрузка в память дисплея, перекодировка для инициации и др.). В этом случае подпрограммы будут доступны и для прикладной программы пользователя, что создаст дополнительные удобства.

При написании прикладных программ в распоряжении пользователя (разработчика) остаются: оперативная память по адресам 800Н–0FFH; РПД, включая оба банка регистров; таймер; пять ячеек стека; 16 линий ввода/вывода и все стандартные подпрограммы ввода/вывода и перекодировки Монитора.

Используя данное описание аппаратуры и программного обеспечения отладочного модуля, можно самостоятельно разработать и быстро создать весьма эффективное и простое средство отладки прикладных программ для МК48.

8.4.

Локальная управляющая микросеть на основе MK51

При управлении сложными протяженными объектами используются системы с распределенным управлением, состоящие из множества контроллеров, управляющих отдельными агрегатами объекта (например, силовой установкой корабля или локомотива, прокатным станом и т.п.). Между отдельными подсистемами должна обеспечиваться информационная связь. Поэтому обязательным элементом распределенной системы управления является локальная сеть, объединяющая отдельные контроллеры в систему.

На такую сеть возлагаются обычно простые функции передачи сообщений за гарантированное время. Протяженность линий связи обычно не превышает десятков метров, размер сообщения – нескольких десятков байтов, а время доставки сообщения – в пределах от 0,01 до 1,0 с. Типичными являются два режима информационного обмена в сети: *широковещательный*, когда передаваемое сообщение предизначается для всех остальных подсистем (микроконтроллеров сети), и *абонентский*, когда сообщение предизначается только одному МК. Обычно первым способом передаются различные информационные параметры, используемые многими подсистемами. Это позволяет уменьшить загрузку сети за счет исключения множественных передач одного и того же

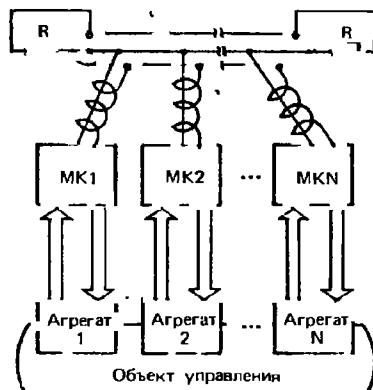


Рис. 8.11. Структура локальной управляющей микросети на основе моноканала

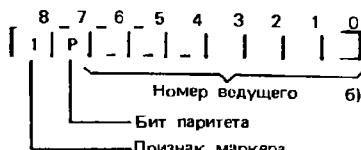
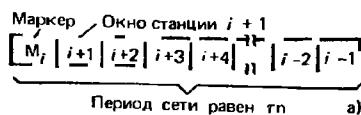


Рис. 8.12. Интервально-маркерный доступ к моноканалу:

a – диаграмма одного периода сети; *b* – маркер; *c* – формат сообщения

сообщения различным адресатам. Вторым способом обычно передаются команды управления от центрального устройства к исполнительным или сообщения экстренного характера.

Наиболее широко распространены локальные сети двух структур: кольцевые и моноканальные (типа BITBUS). Последние, на наш взгляд, являются более удобными для рассматриваемого класса управляющих микросетей, так как допускают простую наращиваемость и модифицируемость системы. Кроме того, в моноканальной микросети время доставки сообщения не зависит от общего числа МК и они обладают большей живучестью и надежностью.

В сетях с единым моноканалом все МК связаны между собой одной общей (разделяемой) линией связи (рис. 8.11). В качестве линии передачи данных обычно используется коаксиальный кабель или витая пара с согласующими резисторами на концах.

Существует несколько известных методов доступа к разделяемому линии (протоколов), позволяющих осуществлять обмен данными между многими МК сети, т.е. обеспечивающих разделение канала связи между многими подсистемами [6, 23, 28, 31].

Наиболее перспективным методом доступа к разделяемому моноканалу, на наш взгляд, является *интервально-маркерный метод*, позволяющий устранить конфликты в канале и наиболее полно использовать пропускную способность канала. Вкратце этот метод сводится к следующему.

1. При нулевой загрузке в канале периодически появляется *маркер*, генерируемый одним из МК сети. Маркер содержит номер МК, являющегося ведущим. Главная обязанность ведущего – поддерживать синхронизм в сети путем периодической выдачи маркера в канал.

2. Период генерации маркера состоит из определенного числа "окон", равного числу МК в сети. Каждое окно имеет свой номер и принадлежит одной из подсистем.

3. В процессе захвата канала МК, желающий выдать свой пакет (сообщение), должен дождаться появления маркера и отсчитать от него свое окно. Если при этом его не опередят другие МК, то, дождавшись своего окна, подсистема может, не опасаясь конфликтов, начинать передачу данных (рис. 8.12, а).

4. После выдачи сообщения МК генерирует свой маркер и становится новым ведущим. Старый ведущий микросети, распознав, что моноканал захвачен, освобождается от этой роли.

5. Отсчет момента времени от маркера до своего окна производится по следующему правилу. Длительность окна принимается равной времени передачи одного байта данных. Если ведущий имел номер l , то первое окно будет принадлежать МК с номером $l + 1$, затем МК с номером $l + 2$ и т.д. Время ожидания своего окна (T) можно определить как $T = \tau \times X$, где τ — время передачи одного байта, т.е. длительность окна. Число X определяется следующим образом:

$$X = \begin{cases} k - l - 1, & \text{если } k > l; \\ k - l + 1, & \text{если } k < l, \end{cases}$$

где k — номер МК, пытающегося захватить канал, $k = 0 \div (n - 1)$; l — номер ведущего, $l = 0 \div (n - 1)$; n — число МК в сети.

6. Если самому ведущему необходимо выдать сообщение, то он может захватить канал во время своего окна, т.е. вместо генерации очередного маркера начать передавать свое сообщение.

7. Выдав маркер, ведущий МК запускает таймер на время $\tau \times (n - 1)$, а если за это время никто не захватит канал, то весь цикл повторяется еще раз и т.д.

8. Каждый МК принимает все байты, передаваемые по каналу. Для контроля пропажи маркера каждый МК после приема каждого байта запускает таймер на задержку $\tau \times (n + 1)$. Таким образом, пропажа маркера (а следовательно, и синхронизма микросети) фиксируется, если за время $\tau \times (n + 1)$ не было передано ни одного байта.

9. При обнаружении пропажи маркера для восстановления синхронизма в микросети каждый МК выполняет следующие простые действия: выдерживает паузу длительностью $\tau \times (i + 1)$, где i — собственный номер МК; если во время паузы вновь не было принято никакой информации, то данный МК становится ведущим и генерирует новый маркер.

Этой процедурой обеспечивается автоматическое восстановление работы микросети при отказе МК, являющегося в данный момент ведущим.

10. При интервалально-маркерном методе удается избежать любых конфликтов в канале в силу следующих причин:

контроль пропажи маркера ведется постоянно всеми МК и полностью синхронно, так как счетчики паузы корректируются примерно одно-

временно при приеме каждого байта и, следовательно, все МК обнаружат пропажу маркера одновременно;

одновременно начинается отсчет паузы $\tau(i + 1)$ всеми МК;

микроконтроллер с меньшим номером первым генерирует маркер и восстановит синхронизм в микросети.

Последовательный порт MK51 допускает передачу 9-битных кодов. Используя это, можно легко ввести признак маркера таким образом, что байт маркера будет отличаться от любого информационного байта. На рис. 8.12, б представлена структура маркера; старший бит является признаком маркера (для маркера — 1). Бит 7 используется для простейшего контроля по паритету. Семибитное поле адреса позволяет иметь в системе до 127 подсистем с номерами от 0 до 126. Адрес 127 зарезервирован для широковещательной передачи.

Рекомендуемый формат сообщения представлен на рис. 8.12, в и предусматривает следующие поля: A_n — адрес получателя; A_o — адрес отправителя; L — длина поля данных (0–255); CRC — байт контрольной суммы.

Можно определить четыре состояния, в которых будет находиться каждый МК микросети.

Приемник (R). В этом состоянии МК прослушивает канал, принимает сообщения и выбирает из них необходимую ему информацию.

Передатчик (W). В этом состоянии МК, захватив канал, передает свое сообщение.

Ведущий (H). МК является ведущим и поддерживает синхронизм в сети.

Специальное состояние (RM). МК реализует процедуру восстановления синхронизма в сети.

Граф состояний МК представлен на рис. 8.13.

Запрос на передачу сообщения формируется в МК прикладной программой управления объектом и обозначен RQ. Задержки, реализуемые таймером, имеют следующий смысл:

TM1 — контроль пропажи маркера, задержка равна $\tau \times (n + 1)$;

TM2 — ожидание своего окна ($\tau \times X$);

TM3 — ожидание окончания периода сети ($\tau \times n$);

TM4 — пауза перед выдачей маркера при восстановлении синхронизма, задержка равна $\tau \times (i + 1)$.

Для реализации подсистемы требуются следующие ресурсы: УАПП, таймер, два уровня прерываний. Этими ресурсами располагает MK51, позволяющий вести передачу и прием данных со скоростью до 375 кбит/с. Время передачи одного байта, обрамленного стартовым и стоповым битами (плюс 9-й разряд), составляет 58,7 мкс. Пропускная способность микросети при этом равна примерно 17 кбит/с.

Микроконтроллер, работающий в составе распределенной системы управления на основе локальной микросети, должен кроме прикладной программы управления иметь еще и программные средства доступа к

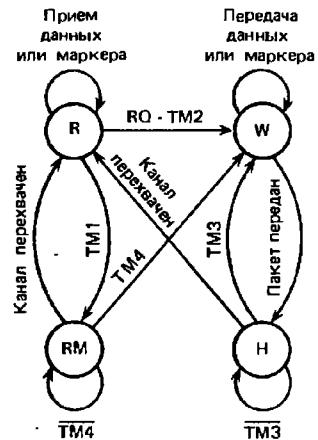


Рис. 8.13. Граф состояний МК микросистемы

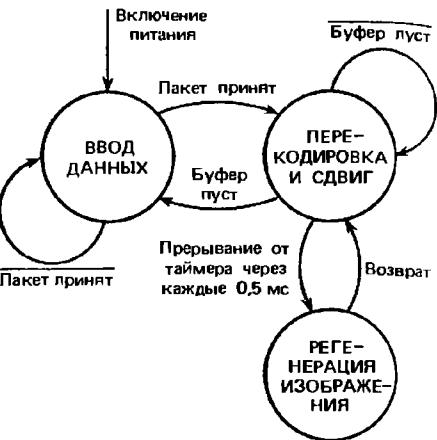


Рис. 8.14. Граф состояния контроллера дисплея

моноканалу. Таким образом, МК должен работать в двухпрограммном режиме с разделением всех ресурсов между этими двумя сопрограммами. Разумеется, должен быть реализован механизм взаимодействия между сетевой и прикладной программами. Чаще всего этот механизм реализуется путем присвоения сетевой программе более высокого приоритета.

8.5.

Контроллер односторочного дисплея на матричных индикаторах

Постановка задачи. Во многих применениях может оказаться полезным автономный линейный дисплей на матричных индикаторах, удаленный от основного контроллера или микроЭВМ. Имея ограниченное число знакомест (от 8 до 16), выстроенных в одну линию, он тем не менее пригоден для вывода больших текстовых сообщений в режиме "бегущей строки". Для привлечения внимания в момент появления сообщения часто подается короткий звуковой сигнал (бип-сигнал). Связь такого удаленного дисплея с ведущим контроллером может осуществляться, например, по последовательному каналу в асинхронном режиме. Портативность и низкая стоимость выгодно отличают дисплеи такого типа от стандартных на основе электронно-лучевой трубки.

Рассмотрим пример разработки дисплея подобного типа со следующими техническими характеристиками: число знакомест – 8; размерность знакоместа – 7×5 световых точек; число строк – одна; скорость вывода (передвижение) сообщения – примерно 2 знака/с; интерфейс – последовательный канал, асинхронный режим (уровни сигналов – ТТЛ); скорость приема данных – 9600 бит/с; протокол DTR, в котором ведущим устройством используется сигнал готовности дисплея; коди-

рование символов – КОИ-7; код подачи бип-сигнала – 07; длительность бип-сигнала – 0,2 с.

Анализ задачи. На автономный дисплей возлагаются следующие функции: прием данных по последовательному каналу, перекодировка символов для отображения на матрице точек 7×5 , сдвиг влево изображения на одну колонку каждые 0,1 с и, иаконец, постоянная регенерация изображения. После выдачи звукового сигнала целесообразно предусмотреть паузу в 1–2 с для того, чтобы оператор успел подготовиться к приему сообщения.

Для снижения затрат времени ведущего контроллера на передачу сообщения целесообразно организовать в контроллере матричного дисплея (КМД) входной буфер емкостью 20–30 байт, т.е. примерно на одно сообщение. При этом контроллер будет идентифицировать конец сообщения либо по заполнению своего входного буфера, либо по приему специального кода "конец передачи" – (04). В обоих случаях контроллер должен сразу жебросить сигнал готовности.

"Бегущая строка" будет останавливаться после появления на экране последнего принятого символа, если новые символы не поступили. Это позволит ведущей микроЭВМ задерживать на дисплее важные цифровые данные. При необходимости очистить дисплей это можно сделать путем передачи восьми кодов пробела (20)₁₆.

Инженерная интерпретация задачи. Можно выделить три автоматических состояния в работе контроллера: ВВОД ДАННЫХ, ПЕРЕКОДИРОВКА И СДВИГ, РЕГЕНЕРАЦИЯ ИЗОБРАЖЕНИЯ (рис. 8.14). После включения электропитания и выполнения процедуры инициализации контроллер попадает в состояние ВВОД ДАННЫХ. Схема алгоритма работы контроллера в этом состоянии приводится на рис. 8.15, а. В этом состоянии устанавливается сигнал готовности (ГТ) к работе, после чего ведущая микроЭВМ может начинать передачу данных в контроллер дисплея. Процедура ввода данных в последовательном коде описывалась выше, поэтому на схеме алгоритма она не раскрывается. После окончания ввода данных сигнал готовности сбрасывается, значение счетчика байтов (СЧБ) фиксируется как входной параметр для состояния ПЕРЕКОДИРОВКА И СДВИГ, схема которого приводится на рис. 8.15, б. В этом состоянии контроллер осуществляет перекодировку полученных символов, заполнение и сдвиг буфера отображения. Процедуры перекодировки, выдачи бип-сигнала и временной задержки рассматривались ранее и поэтому здесь не раскрываются. Процесс отображения символов и организации режима индикации "бегущая строка" поясняется на рис. 8.16. Буфер, содержащий коды индикации (буфер отображения), имеет 9 позиций, в то время как знакомест всего 8. Каждая позиция буфера отображения состоит из 5 байт – по числу колонок знакоместа. Через дополнительную (неотображаемую) девятую позицию буфера отображения осуществляется вывод символа на индикацию.

После инициализации все позиции дисплея погашены. Из входного буфера извлекается символ, осуществляется его перекодировка и 5 байт индикации загружаются в девятую позицию буфера отображения.

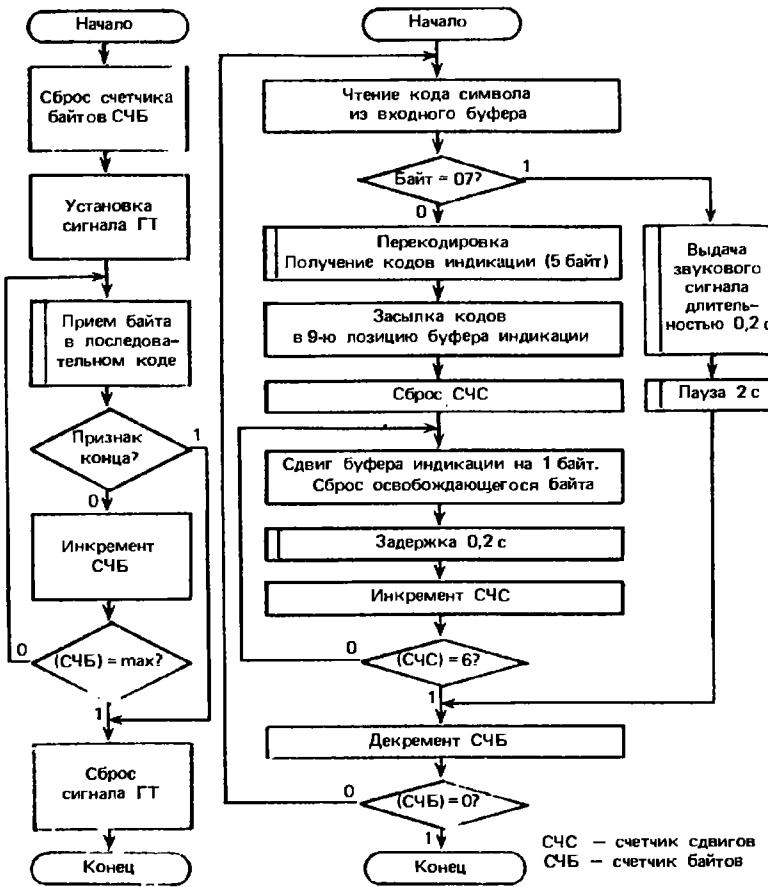
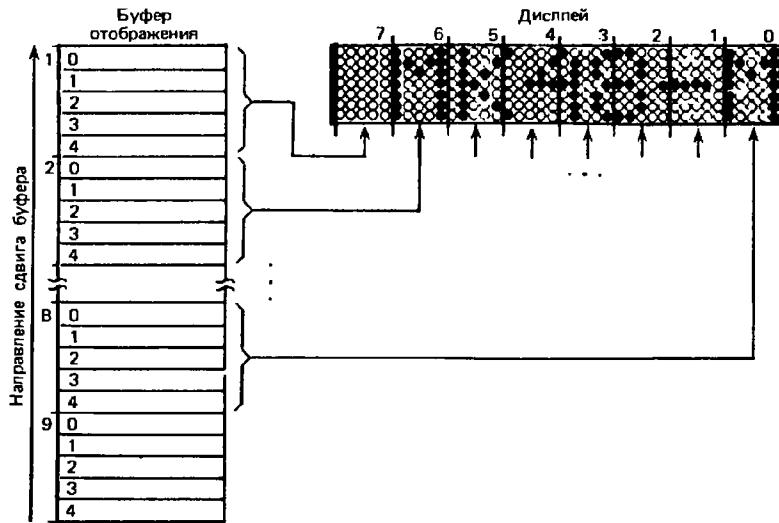


Рис. 8.15. Блок-схемы алгоритма работы контроллера в состояниях ВВОД ДАННЫХ (а) и ПЕРЕКОДИРОВКА И СДВИГ (б)

Далее осуществляется сдвиг всего буфера отображения на 1 байт в сторону первой позиции, при этом код первой (левой) колонки символа перемещается в восьмую позицию и появляется на экране. Задержка в 0,1 с задает темп "бега" строки. При каждом сдвиге на место четвертого байта девятой позиции записывается нулевой код, т.е. колонка гасится. Таким образом, после пяти сдвигов символ целиком перемещается в восьмую позицию буфера отображения и весь появляется на экране. Далее следует еще один сдвиг буфера для разделения знакомест



(межсимвольный разделитель). После выполнения шести сдвигов можно обращаться к входному буферу за кодом очередного символа. Если в потоке данных встречается код бип-сигнала (07), то производится передача управления процедуре генерации бип-сигнала с последующей задержкой в 2 с. В это время изображение остается неизменным.

При исчерпании входного буфера и вывода последнего символа изображение останавливается, контроллер дисплея переходит в состояние ВВОД ДАННЫХ.

Если необходимо, чтобы сообщение целиком прошло по экрану и бесследно исчезло, то ведущему контроллеру необходимо дополнить его восемью пробелами.

Процедура регенерации изображения подробно рассматривалась выше, поэтому здесь остановимся только на некоторых особенностях ее конкретной реализации. Процедура регенерации (или поддержания) изображения обычно включается в работу по прерыванию от таймера. При каждом обращении к ней отображается одна из 40 колонок дисплея. Каждая колонка должна засветиться не менее 20 раз в секунду. Исходя из этого вычисляется период вызова процедуры: $1/20 \cdot 40 = 1,25$ мс.

На время выполнения процедуры ввода байта из последовательного канала нужно запретить прерывания от таймера (во время ожидания стартового бита прерывания все же должны быть разрешены), так как вклинивание подпрограммы регенерации в процедуру ввода байта может привести к рассогласованию частот приема и передачи. При частоте передачи данных 9600 бит/с символ (7 бит + старт-бит + стоп-бит) будет

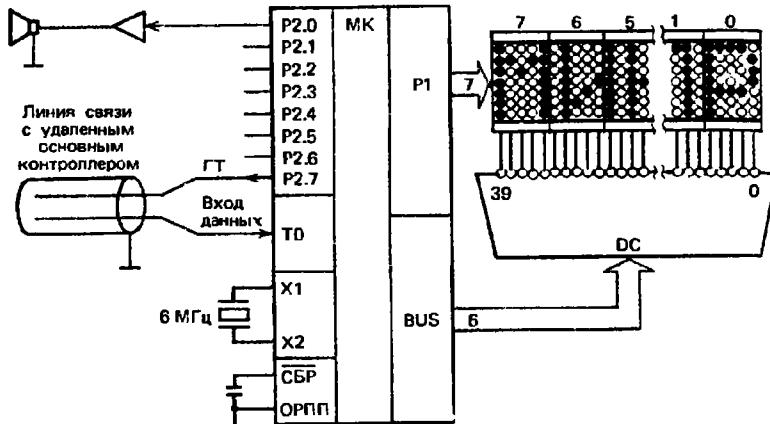


Рис. 8.17. Структурная схема конгломера дисплея

передан за $1/9600 \times 9 = 937,5$ мкс, входной буфер емкостью 32 байта будет загружен примерно за 30 мс. Сбой изображения такой длительности не будет заметен глазу человека. Периодичность появления такого сбоя изображения при длине сообщений 16 байт составит один раз за $16 \times 6 \times 0,2 = 19,2$ с.

Разработка схемы устройства. Рассмотрим вариант схемной реализации контроллера дисплея на МК48.

Для связи с ведущей микроЭВМ требуется одна линия ввода и одна линия вывода; для подачи бип-сигнала — одна линия вывода; для сканирования матричного дисплея — семь линий для вывода кода индикации и еще шесть линий вывода для выборки отображаемой колонки. Для удобства сканирования дисплея можно выделить отдельный порт вывода кода индикации и отдельный порт выборки колонок, например порты P1 и BUS соответственно. Для ввода последовательного кода удобнее всего использовать тестовый вход, например T0. Для выдачи бип-сигнала можно воспользоваться свободной линией порта P2, например P2.0. Одна из возможных схем контроллера дисплея приведена на рис. 8.17.

Необходимые временные задержки придется реализовывать программно, так как таймер будет задействован для инициализации процедуры регенерации изображения.

Длина входного буфера должна определиться после разработки всего программного обеспечения контроллера. Под входной буфер следует отвести все незадействованные ячейки РПД, включая свободные ячейки стека и банка регистров 1.

Авторы надеются, что написание исходного текста прикладной программы контроллера дисплея не займет много времени у заинтересовавшегося читателя. Желаем успеха!

Приложение

Приложение П.1

Контроллер клавиатуры/дисплея KP580ВД79

П1.1. Общие сведения

Микросхема KP580ВД79, именуемая для краткости ККД (контроллер клавиатуры/дисплея), представляет собой универсальное программируемое устройство сопряжения с клавиатурой и дисплеем на основе семисегментных светодиодных индикаторов (ССИ).

Клавиатуриальная часть обеспечивает работу с клавиатурой размером 8 × 8 + 2 клавиши или с матрицей 8 × 8 датчиков. Обеспечиваются различные линии сканирования нажатых клавиш, антидребезговый контроль. Имеется память кодов нажатых клавиш на 8 байт, организо-

Линии возврата	ЛВ ₂	(RL ₂)	1	40	+ 5 В ОСН (VCC)
	ЛВ ₃	(RL ₃)	2	39	ЛВ ₁ (RL ₁)
	СИНХР	(CLK)	3	38	ЛВ ₀ (RL ₀)
	ЗПР	(IRC)	4	37	УС/СТБ (CNTL/STB)
	ЛВ ₄	(RL ₄)	5	36	СДВИГ (SHIFT)
	ЛВ ₅	(RL ₅)	6	35	СКАНЛИН ₃ (SL ₃)
	ЛВ ₆	(RL ₆)	7	34	СКАНЛИН ₂ (SL ₂)
	ЛВ ₇	(RL ₇)	8	33	СКАНЛИН ₁ (SL ₁)
	СБР	(RESET)	9	32	СКАНЛИН ₀ (SL ₀)
	ЧТ	(RD)	10	31	ВЫХВ ₀ (OB ₀)
	ЗП	(WR)	11	30	ВЫХВ ₁ (OB ₁)
ШД ₀	ШД ₀	(DB ₀)	12	29	ВЫХВ ₂ (OB ₂)
	ШД ₁	(DB ₁)	13	28	ВЫХВ ₃ (OB ₃)
ШД ₂	ШД ₂	(DB ₂)	14	27	ВЫХА ₀ (OA ₀)
	ШД ₃	(DB ₃)	15	26	ВЫХА ₁ (OA ₁)
ШД ₄	ШД ₄	(DB ₄)	16	25	ВЫХА ₂ (OA ₂)
	ШД ₅	(DB ₅)	17	24	ВЫХА ₃ (OA ₃)
ШД ₆	ШД ₆	(DB ₆)	18	23	ПРОБЕЛ (БЛАНК) (BD)
	ШД ₇	(DB ₇)	19	22	ВК (CS) Выбор корпуса
Общ	ОБЩ	(GND)	20	21	A ₀ Линия адреса

Рис. П.1. Цоколевка БИС контроллера клавиатуры/дисплея

ванная в виде очереди FIFO. При занесении в эту память более 8 кодов устанавливается признак переполнения. Нажатие клавиши возбуждает линию прерывания, которая может опознаваться в МК.

Дисплейная часть обеспечивает работу с дисплеем на семисегментных индикаторах (их может быть до 32 шт.). Имеется ОЗУ дисплея на 16 байт, организованное в виде двух массивов 16×4 бита. Память дисплея может быть загружена из МК и прочитана им. И в том, и в другом случае возможно автоинкрементирование адреса ОЗУ дисплея. Таким образом, ККД освобождает МК от задач постоянного сканирования клавиатуры и поддержания изображения на дисплее.

П 1.2. Назначение выводов ККД

На рис. П1.1 представлена цоколевка БИС ККД, в табл. П1.1 приводится описание каждого вывода.

Таблица П1.1. Назначение выводов ККД

Номер вывода	Коли-чество	Обозна-чение	Описание функции
12 19	8	DB0–DB7	Линии направления шина данных. По ней персылаются все данные между ККД и МК
3	1	CLK	По этой линии подаются синхросигналы. Их частота не должна превышать 3200 кГц
9	1	RFSET	Линия сброса ККД в начальное состояние. Активный уровень высокий
22	1	CS	Выбор корпуса. Активный уровень низкий
21	1	A0	Единица на этом входе означает, что из МК в ККД передается управляющее слово, а из ККД в МК байт состояния. Нуль указывает на то, что персылаются байты данных
10, 11	1,1	RD, WR	Чтение, запись. Нулевые уровни на этих линиях разрешают соответственно выдачу/прием данных из/в ККД
4	1	IRQ	Линия прерывания. В режиме клавиатуры устанавливается в единицу при наличии данных в памяти клавиатуры. При каждом считывании данных из памяти клавиатуры линия сбрасывается в нуль, но устанавливается вновь, если память не пуста. В режиме матрицы датчиков прерывание устанавливается при любом изменении состояния датчиков
32–35	4	SL0–SL3	Линии сканирования. Они используются одновременно для сканирования клавиатуры (матрицы датчиков) и для сканирования позиций дисплея. Могут работать в режиме счетчика или в режиме инверсного дешифратора
38–39 1, 2 5–8	8	RL0–RL7	Линии возврата. Они соединяются со сканирующими линиями через клавиатуру (матрицу датчиков), нажатие клавиши должно приводить к появлению нуля на одной из линий
36	1	SHIFT	Сдвиг. Состояние этой линии заноминается в одном из бит кода нажатой клавиши и может быть использовано как признак переключения регистра клавиатуры. Незадействованный вход воспринимается как единица

Номер вывода	Коли-чество	Обозна-чение	Описание функции
37	1	CNTL/ STB	В режиме клавиатуры используется так же, как SHIFT. В режиме "стробируемый ввод" используется как линия стробирования ($- \uparrow -$)
27 24	4	OA0– OA3	Выходы двух регистров чаных дисплея, на которые выдаются коды из ОЗУ дисплея. Стробируемые линии SL, они могут рассматриваться как один 8-битный порт или как два 4-битных. Каждый из них может быть очищен независимо: (OA – старшая тетрада, OB – младшая тетрада)
31–28	4	OB0– OB3	
23	1	BD	Очистка. Используется для гашения дисплея в момент переключения цифр или при выдаче УС очистки
20	1	GND	Земля. Общая точка
40	1	VCC	Питание. Напряжение питания +5 В

П1.3. Управление ККД с помощью сигналов на внешних выводах

Управляющие слова (УС) загружаются в регистры управления ККД при $A_0 = 1$. Загружая определенные УС, можно настроить ККД на работу в требуемом режиме и предписать выполнение некоторой операции.

Операции, выполняемые в ККД, определяются не только управляющим словом, но и комбинацией управляющих сигналов на его входах:

CS	A0	RD	WR	Операция ввода/вывода
1	X	X	X	ККД не выбран
0	X	1	1	ККД не выбран
0	0	0	1	Чтение данных из памяти клавиатуры или дисплея
0	0	1	0	Запись данных в память дисплея
0	1	0	1	Чтение байта состояния ККД
0	1	1	0	Загрузка УС в ККД

Операция записи в память клавиатуры невыполнима.

П1.4. Управляющие слова (УС)

УС инициализации клавиатуры и дисплея (УС0): 0.0.0.D.D.K.K.S. Здесь DD кодирует режим работы дисплея, KK – режим работы клавиатуры, S – режим сканирования в соответствии со следующими правилами:

- | | |
|--------|--|
| DD: 00 | Дисплей на 8 символов с вводом слева |
| 01 | Дисплей на 16 символов с вводом слева |
| 10 | Дисплей на 8 символов с вводом справа |
| 11 | Дисплей на 16 символов с вводом справа |

Описание режимов правого и левого вводов приводится ниже. Если сканирование идет в режиме дешифратора, то дисплей не может быть больше, чем на четыре символа (так как линии SL общие).

- KK: 00 Клавиатура в режиме одиночного нажатия клавиш
01 Клавиатура в режиме N-клавишного нажатия
10 Сканирование матрицы датчиков
11 Режим стробируемого ввода
- S: 0 Сканирование в режиме 4-битного двоичного счетчика
1 Сканирование в режиме инверсного дешифратора на четыре выхода

После сброса ККД оказывается в режиме, соответствующем УС0-00001000. При сканировании в режиме счетчика цикл опроса клавиатуры укладывается в восемь состояний счетчика от "0000" до "0111" и от "1000" до "1111". Таким образом, в этом режиме опрос клавиатуры происходит дважды в каждом цикле полного пересчета счетчика SL0-SL3, т.е. для сканирования клавиатуры используются только три младшие линии SL0-SL2.

УС инициализации опорной частоты (УС1): 0.0.1.P.P.P.P.P. Здесь PPPPP устанавливает коэффициент деления частоты внешнего синхронизирующего сигнала CLK для получения внутреннего опорного сигнала с частотой не более 100 кГц. После сброса устанавливается максимальный коэффициент PPPPP, равный 11111. При частоте внутреннего опорного сигнала 100 кГц один полный цикл сканирования занимает приблизительно 10,2 мс.

УС чтения памяти клавиатуры/датчиков (УС2): 0.1.0.I.X.A.A.A. УС2 должно предшествовать чтению данных из памяти клавиатуры. Здесь I есть признак автономной адресации; AAA устанавливает адрес байта памяти, который должен быть считан. Если бит I установлен, то последующие команды чтения данных будут вызывать автоматическое увеличение адреса. Таким образом, если необходимо прочитать всю память клавиатуры, то это можно сделать, загрузив УС2 с битом I = 1 и затем 8 раз считать данные из ККД.

УС чтения памяти дисплея (УС3): 0.1.1.I.A.A.A.A. УС3 должно предшествовать чтению данных из памяти дисплея. Здесь I есть признак автономной адресации: AAAA – номер позиции дисплея, которая должна быть считана. Если I = 1, то адрес будет инкрементироваться после каждого чтения.

УС записи в память дисплея (УС4): 1.0.0.I.A.A.A.A. Кодирование аналогично УС3. Поле AAAA определяет адрес байта в ОЗУ дисплея.

УС запрета записи в память дисплея и бланкирования дисплея (УС5): 1.0.1.X.IWA.IWB.BLA.BLB. Здесь IW – указатель запрета записи (тетрады A и B), BL – указатель бланкирования (очистки) (тетрады A или B). Если дисплей используется как двойной 4-позиционный, то удобно маскировать одну из тетрад, чтобы работа процессора с одной тетрадой не отражалась на другой. Эту возможность дает указатель IW. Указатель BL используется для бланкирования дисплея; если он установлен, то на выходах A и/или B устанавливается специальный бланкирующий

код (см. УС6). Содержимое памяти дисплея при этом не изменяется. Если дисплей используется как единый 8-позиционный, то необходимо устанавливать оба указателя.

УС сброса памяти дисплея (УС6): 1.1.0.CD.BC.BC.CF.CA. УС6 служит для: выбора кода бланкирования (BC), сброса байта состояния (CF) и сброса памяти дисплея (CD). Биты BC позволяют выбрать один из трех возможных бланкирующих кодов:

BC	BC	Коды бланкирования
0	X	Все нули (00H)
1	0	20H (в KOM-7 код пробела)
1	1	Все единицы (FFH)

После общего сброса контроллера бланкирующий код устанавливается равным 00H.

Процедура сброса памяти дисплея осуществляется путем заполнения кодами бланкирования. Процедура инициируется при установке бита CD и продолжается примерно 160 мкс. В это время память дисплея недоступна, на что указывает старший бит байта состояния контроллера.

Бит CF, будучи установлен, сбрасывает байт состояния, сигнал прерывания и устанавливает указатель памяти матрицы датчиков на строку 0. Управляющий бит CA работает как биты CD и CF в совокупности, а также сбрасывает схему внутренней синхронизации.

УС сброса прерывания/установки режима ошибки (УС7): 1.1.1.E.X.X.X.X. В состоянии матрицы датчиков УС7 сбрасывает линию прерывания и разрешает дальнейшую запись в память клавиатуры (матрицы датчиков). В режиме N-клавишного нажатия, если установлен бит E, контроллер переходит в режим ошибки.

П1.5. Режим работы клавиатуры

Режим одиночного нажатия клавиш. Дисциплина опознания нажатых клавиш в этом режиме такова: если обнаружено нажатие одной клавиши, то в течение следующих двух циклов сканирования клавиатуры будет производиться проверка нажатия других клавиш. Если таких клавиш не будет, то нажатая клавиша признается единственной и код ее записывается в память клавиатуры. Если в течение этих двух циклов будет обнаружено нажатие еще одной клавиши, то в память клавиатуры не заносится код ни одной клавиши до тех пор, пока не будут освобождены все клавиши, кроме одной. После того как все, кроме одной, клавиши будут освобождены и не будут нажаты новые в течение двух циклов, код этой клавиши будет занесен в память клавиатуры. Код клавиши заносится в память клавиатуры только один раз на каждое нажатие.

Режим N-клавишного нажатия. В этом режиме нажатие каждой клавиши фиксируется независимо от состояния остальных клавиш. Когда клавиша нажата, пропускаются два цикла опроса клавиатуры, а затем проверяется, осталась ли данная клавиша нажатой. Если да, то ее код

заносится в память. При одновременном нажатии распознавание клавиш производится в порядке их опроса в цикле сканирования клавиатуры. Опознание всех клавиш ведется независимо. На каждое нажатие код клавиши вводится только один раз.

Специальный режим ошибки при сканировании клавиатуры. Для режима сканирования клавиатуры с N-клавищным нажатием с помощью УС7 можно запрограммировать специальный режим ошибки. Если в течение одного цикла сканирования будут нажаты две клавиши, это трактуется как совместное нажатие и устанавливается флаг ошибки в байтке состояния. Этот флаг запрещает дальнейшую запись в память клавиатуры и устанавливает линию прерывания (если она не была установлена). Флаг ошибки можно сбросить с помощью УС6, установив в нем CF = 1.

Режим матрицы датчиков. В этом режиме "антидребезговая" логика не работает. Состояние датчиков непосредственно записывается в память матрицы датчиков (память клавиатуры). Хотя антидребезговый контроль и не обеспечивается, МК может иметь информацию о том, как долго датчик находится в единичном или нулевом состоянии (у датчика подразумеваются только два состояния). Линия прерывания устанавливается в единицу, если в конце цикла сканирования матрицы хотя бы один из датчиков изменил свое состояние, и сбрасывается при первой же операции чтения, если не был установлен указатель автоинкрементирования.

Форматы данных. В режиме клавиатуры байт, записываемый в память клавиатуры, отражает положение клавиши на клавиатуре, а также состояние входов CNTL и SHIFT. Формат данных в этом режиме CNTL.SHIFT.SL2.SL1.SL0.R2.R1.R0.

Здесь SL2–SL0 – двоичный код строки, в которой находится клавиша, R2–R0 – двоичный код колонки, в которой находится клавиша (номер линии RL, на которую поступил нулевой сигнал).

В состоянии матрицы датчиков данные, приходящие на линии RL7–RL0, непосредственно вводятся в память клавиатуры. Таким образом, каждое изменение в датчиках в течение одного цикла будет отражено в памяти. Состояния SHIFT и CNTL игнорируются.

Формат данных в режиме стробируемого ввода: RL7.RL6.RL5.RL4.RL3.RL2.RL1.RL0. В этом режиме данные также вводятся в память по линиям RL7–RL0, но при этом ввод стробируется линией CNTL/STB. Данные заносятся в память по фронту сигнала CNTL/STB.

Следует помнить, что цикл сканирования клавиатуры постоянен, не зависит от размера дисплея и соответствует максимальной конфигурации клавиатуры 8 × 8 при сканировании в режиме счетчика и 4 × 8 при сканировании в режиме дешифратора.

П1.6. Режимы работы дисплея

Ввод слева. Это самый простой формат ввода. При нем каждой позиции на дисплее однозначно соответствует байт в памяти дисплея. Нул-

вой адрес памяти дисплея соответствует самой левой нулевой позиции дисплея, адрес 15 (7 для 8-позиционного дисплея) соответствует самой правой позиции дисплея. При переходе за 16 (8) позиций следующие символы будут опять вводиться слева с нулевой позиции, т.е. 17-й (9) символ займет крайнюю левую позицию.

Ввод справа. Этот способ ввода используется в калькуляторах. Первый вводимый символ заносится в крайнюю Правую позицию. Следующие символы также заносятся в крайнюю правую позицию, но после того, как все символы на дисплее сдвинутся на одну позицию влево. Самый левый символ при этом теряется. Заметим, что здесь номера позиций на дисплее уже не соответствуют адресам памяти дисплея. Адрес памяти дисплея в этом случае соответствует порядковому номеру нажатой клавиши. Указание позиции, в которую должен быть введен символ при автоинкрементированном вводе (см. далее), может иметь непредсказуемый результат.

Автоинкрементирование. В состоянии ввода слева это означает, что следующий символ, поступающий из МК, будет размещен по адресу, на единицу большему, чем предыдущий. При отсутствии автоинкремента ввод будет произведен в ту же самую позицию. Использование автоинкремента при вводе слева не порождает непредсказуемых эффектов, даже если после ввода нескольких символов нужно ввести очередной символ не в следующую, а в произвольную позицию. Это обусловлено тем, что позиция на дисплее при вводе слева однозначно соответствует ресурсу памяти дисплея. При вводе справа с автоинкрементированием попытка ввести символ в фиксированную позицию будет иметь непредсказуемый результат.

Формат дисплея. Если используется 8-позиционный дисплей, то время цикла сканирования дисплея в 2 раза меньше, чем для 16-позиционного дисплея. При внутренней частоте 100 кГц это время составляет 5,1 и 10,2 мс соответственно.

П1.7. Байт состояния памяти клавиатуры/датчиков

Байт состояния используется в режиме клавиатуры и стробируемого ввода для отображения числа введенных символов и индикации ошибок. Возможны два вида ошибок: переполнение и переопустошение. Флаг переполнения устанавливается, когда делается попытка записи в заполненную память клавиатуры. Флаг переопустошения устанавливается, если была попытка считать из пустой памяти клавиатуры. Байт состояния имеет также бит DU, показывающий, что дисплей недоступен из-за того, что не завершена очистка памяти дисплея. В специальном режиме ошибки бит S/E является флагом ошибки и указывает на множественное нажатие клавиши. В режиме матрицы датчиков этот бит показывает, что имеется хотя бы один "замкнутый" (нулевой) датчик.

Формат байта состояния памяти клавиатуры/датчиков:

U.S/E.O.U.F.N.N, где

DU – дисплей недоступен;

S/E — датчик "замкнут"/ошибка многократного нажатия;

O — ошибка переполнения;

U — ошибка переопустошения;

F — память клавиатуры заполнена;

NNN — количество символов в памяти клавиатуры.

П р и м е ч а н и е. В литературе встречаются разные наименования данной БИС: Intell-8279, KP580BB79, KP580BI79, KP580BD79.

Приложение II. 2

Расширитель ввода/вывода KP580BP43

П2.1. Общие сведения

Расширитель ввода/вывода (PBB) KP580BP43 предназначен для увеличения числа линий связи МК48 с объектом управления. Расширитель содержит четыре 4-битных двунаправленных порта, которые являются прямым расширением резидентной системы ввода/вывода МК48. Эти порты, именуемые P4, P5, P6 и P7, уже упоминались при рассмотрении системы команд МК48. Обращение к портам PBB осуществляется по специальным командам (MOVD, ANLD и ORLD), выполняемым совместно МК48 и расширителем. Выходы расширителя обладают высокой нагрузочной способностью: выходной ток составляет 5 и 0,24 мА для нуля и единицы соответственно.

П2.2. Назначение выводов PBB

Цоколевка выводов PBB представлена на рис. П2.1, а, а назначение выводов приводится в табл. П2.1

Таблица П2.1. Назначение выводов PBB

Номер вывода	Количество	Обозначение	Назначение вывода
7	1	ПРОГ	Синхронизация работы PBB и МК48; соединяется с выходом ПРОГ МК48
6	1	ВК	Выбор корпуса; высокий уровень на входе ВК приводит к невозможности изменения состояния выходов PBB
11–8	4	P2.0–P2.3	Двунаправленная шина для связи PBB и МК48; соединяется с младшей половиной порта P2 МК48
2–5	4	P4.0–P4.3	Четыре двунаправленных 4-битных порта с высокой нагрузочной способностью, используемые для ввода или вывода информации
1,23–21	4	P5.0–P5.3	
20–17	4	P6.0–P6.3	
13–16	4	P7.0–P7.3	
24	1	+5 В	Напряжение питания +5 В (потребляемый ток не более 20 мА)
12	1	—	Общая точка (земля)

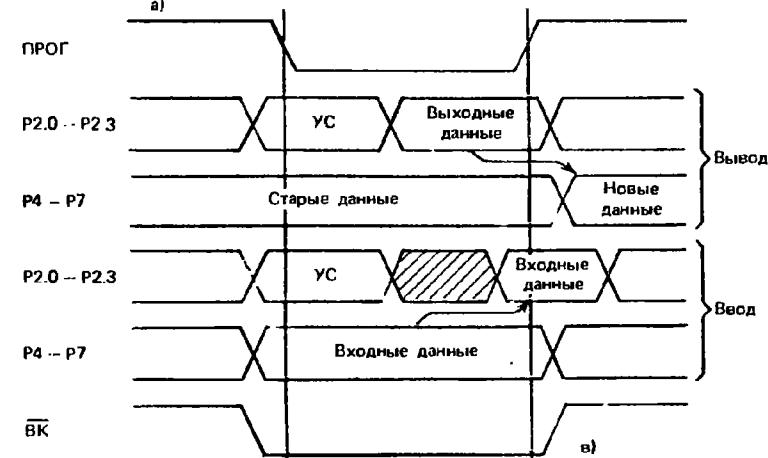
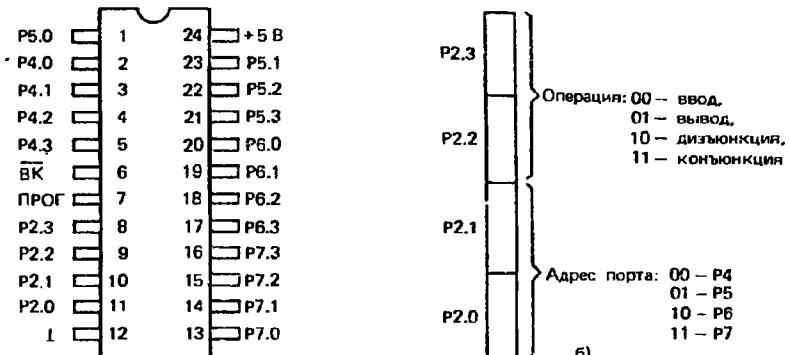


Рис. П2.1. Цоколевка корпуса (а), формат УС (б) и временная диаграмма работы (в) БИС расширителя ввода/вывода

П2.3. Основные режимы работы

Подключение PBB к МК48 производится путем соединения одноименных выводов ПРОГ и P2.0–P2.3. Все пересылки между МК48 и PBB осуществляются через линии P2.0–P2.3 по синхросигналу ПРОГ, вырабатываемому МК48 при выполнении команд обмена с портами PBB (MOVD, ANLD и ORLD). В процессе выполнения каждой команды обмена между PBB и МК передаются две тетрады: первая содержит код выполняемой операции и адрес порта, вторая – данные. Сначала, при переходе сигнала ПРОГ из 1 в 0, передается управляющее слово, а затем, при переходе сигнала ПРОГ из 0 в 1, передаются данные. Формат УС представлен на рис. П2.1, б.

Изменить состояние сигналов на выходах РВВ можно тремя способами: непосредственно с помощью команды MOVD Pp,A загрузить в требуемый порт новое значение; с помощью команды ORLD Pp,A произвести логическое сложение порта и аккумулятора и результат загрузить в порт и, наконец, с помощью команды ANLD Pp,A произвести логическое умножение старого содержимого порта и аккумулятора и результат загрузить в порт. Следует особо отметить, что логические операции над данными выполняет сам РВВ, а не МК48.

Прочитать данные из входного порта можно с помощью команды MOVD A, Rp. При выполнении операции ввода указанный порт настраивается на ввод и его выходы переключаются в третье состояние. Таким образом, для настройки какого-либо порта РВВ на ввод данных необходимо выполнить для него команду ввода. Если до выполнения команды ввода порт находился в режиме вывода, то результат первого ввода окажется неопределенным, а все последующие команды ввода будут давать правильный результат.

Выполнение команд ввода/вывода поясняется временными диаграммами на рис. П2.1, в.

При включении питания все порты РВВ переходят в третье состояние, а линии P2.0–P2.3 настраиваются на режим ввода.

Приложение П. 3

Использование последовательного порта микроконтроллера MK51 для связи с интерфейсом RS-232

Если устройство, построенное на базе MK51, должно иметь выход на последовательный интерфейс RS-232, то перед разработчиком встают следующие проблемы:

- согласование уровней сигналов RS-232 и MK51 (TTL);
- поддержание стандартной скорости приемо-передачи;
- поддержание стандартных форматов посылки;
- поддержание стандартных протоколов обмена.

Согласование уровней легко осуществляется с помощью буферных схем приемо-передатчиков K170AII2 (два передатчика) и K170UII2 (четыре приемника), разработанных специально для этой цели. При этом потребуются дополнительные источники электропитания напряжением +12 и -12 В.

Поддержание стандартного ряда скоростей (19200, 9600, 4800, 2400, 1200, 600, 300 бит/с) является более сложной задачей, так как скорость приемо-передачи УАПП зависит от тактовой частоты MK51.

В силу того, что считывание значения бита УАПП производит в центре битового интервала, максимальный временной сдвиг приема бита равен половине битового интервала (0,5 Т бита). Для того, чтобы все биты информационной посылки были восприняты правильно, расхождение скорости приема и передачи не должно превышать величины $0,5/L$, где L – общая длина посылки, включая биты данных, бит паритета, стартер и стоповые биты.

В табл. П3.1 приведены допустимые погрешности скорости передачи в зависимости от общей длины информационной посылки.

Таблица П3.1. Допустимые погрешности скорости приемо-передачи

Общая длина посылки, бит	Допустимая погрешность скорости, %	Общая длина посылки, бит	Допустимая погрешность скорости, %
7	7,1	10	5,0
8	6,25	11	4,55
9	5,5	12	4,2

Для сопряжения с интерфейсом RS-232 УАПП может работать в режимах 2 (8 бит) или 3 (9 бит), когда скорость приемо-передачи определяется таймером 1.

Ниже приводится программа на языке BASIC, которая позволяет определить, какие стандартные значения скорости передачи достижимы при заданной тактовой частоте (частоте кристаллового резонатора) микроконтроллера.

```

10 REM Расчет скорости передачи для УАПП MK51
20 SMOD=0
30 INPUT "Fosc[MHz]=", FOSC: FOSC=FOSC * 1000: CLS
40 PRINT TAB(20); "Fosc="; FOSC; "Khz": PRINT
50 PRINT "Vct [бит/с]", "TH1", "Vsbuf",
60 PRINT "погрешность [%]"; TAB(65); "SMOD": PRINT
70 V=19.2
80 FOR I=1 TO 7
90 Y=CINT(FOSC * 2^SMOD/(32 * 12 * V)): X=256-Y
100 IF Y=0 THEN PRINT V, "-----": GOTO 130
110 V0=FOSC * 2^SMOD/(32 * 12 * Y): N=(V-V0)/V * 100
120 PRINT V * 1000, X, V0 * 1000, N; TAB(65); SMOD
130 V=V/2
140 NEXT I
150 IF SMOD=0 THEN SMOD=1: PRINT: GOTO 70

```

В табл. П3.2 представлены результаты работы этой программы для тактовой частоты 11 МГц.

Как видно из результатов работы программы, при тактовой частоте 11 МГц скорость передачи 19200 бит/с может быть достигнута только при $SMOD=1$ и максимальном значении уставки таймера (255 или OFFH). Предполагается, что таймер работает в режиме 2. Используя данную программу, можно убедиться, что при тактовой частоте 6 МГц максимально достижимая скорость передачи составит 2400 бит/с.

Однако можно подобрать такую тактовую частоту работы MK51,

Таблица П3.2. Ряд действительных скоростей приема-передачи

V _{ст} , бит/с	TH1	V _{sbuf} , бит/с	Погрешность, %	SMOD
19200	255	28645.84	-49.19705	0
9600	253	9548.611	.5353093	0
4800	250	4774.305	.5353093	0
2400	244	2387.153	.5353093	0
1200	232	1193.576	.5353093	0
60	208	596.7881	.5353093	0
300	161	301.5351	- .5353093	1
9600	250	9548.611	.5353093	1
4800	244	4774.305	.5353093	1
2400	232	2387.153	.5353093	1
1200	208	1193.576	.5353093	1
600	161	603.0702	- .5116959	1
300	65	299.9564	1.454353E-02	1

при которой будет возможно использовать весь стандартный ряд скоростей передачи данных с минимальной погрешностью. Сделать это может другая программа, которая рассчитывает допустимые значения тактовой частоты MK51 в зависимости от требуемой верхней границы стандартного ряда скоростей передачи и значения бита SMOD. Программа и результаты ее работы (табл. П3.3) приведены ниже.

```

10 REM расчет Fosc для стандартного ряда скоростей RS-232
20 INPUT "V [кбит/с], smod=:", V, SMOD: CLS
30 PRINT TAB(10); "Значения Fosc для V [бит/с] =";
40 PRINT V * 1000, "SMOD=", SMOD: PRINT
50 PRINT "TH1", "Fosc [KHz]", "допустимый разброс 5%": PRINT
60 FOR I=1 TO 255
70 F=I * 32/(2*ASMOD) * 12 * V
80 IF F > 12000 THEN 110
90 PRINT 256-I,F,F * 95; TAB(40); "-"; F * 1.05
100 NEXT
110 END

```

Таблица П3.3. Значения тактовой частоты (Fosc) для скорости 9600 бит/с при SMOD=1

TH1	Fosc [KHz]	Допустимый разброс 5%
255	1843.2	1751.04–1935.36
254	3686.4	3502.08–3870.72
253	5529.6	5253.12–5806.08
252	7372.801	7004.16–7741.44
251	9216	8755.2–9676.8
250	11059.2	10506.24–11612.16

Хотя аппаратура УАПП и не поддерживает формирование контрольного разряда и выбора различного числа битов данных и стоповых бит, почти все форматы посылки могут быть сформированы программно. Например, если требуется передавать 8 бит данных с контролем по четности, то можно использовать УАПП в режиме 3 (9 бит) и формировать контрольный разряд программно. При этом для передачи контрольный разряд необходимо загружать в разряд TB8 регистра SCON, а при приеме он будет помещен в бите RB8 того же регистра. Напомним, что MK51 имеет бит паритета P, который следует за четностью числа единиц в аккумуляторе. Использование бита паритета сводит к минимуму действия по формированию контрольного разряда.

И, наконец, любой из известных протоколов последовательного обмена с сигнальным квитированием – DTR, ACK, DC1/DC3 легко реализуется программным путем. При этом для формирования сигналов квитирования (CTS, RTS, DTR, DSR и т.п.) могут быть использованы любые свободные разряды любого порта. Напомним, что для подключения к любой линии интерфейса RS-232 (кроме, SG и PG–земля) необходимо использовать схемы преобразования уровней K170УП2, K170АП2.

(Х – ШЕСТНАДЦАТИЧНАЯ ЦИФРА)

ШЕСТНАДЦАТИЧНЫЕ КОДЫ КОМАНД КМ1816ВЕ48

Старшая цифра	Младшая цифра														Старшая цифра		
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0 NOP		OUTL BUS, A	ADD A, #d	JMP 0XXH	EN I		DEC A		INS A, BUS	IN A, P1	IN A, P2		MOVD A, P4	MOVD A, P5	MOVD A, P6	MOVD A, P7	0
1 INC @R0	INC @R1	JB0 ad	ADDC A, #d	CALL 0XXH	DIS I	JTF ad	INC A		INC R0	INC R1	INC R2	INC R3	INC R4	INC R5	INC R6	INC R7	1
2 XCH A, @R0	XCH A, @R1		MOV A, #d	JMP 1XXH	EN TCNTI	JNT0 ad	CLR A		XCH A, R0	XCH A, R1	XCH A, R2	XCH A, R3	XCH A, R4	XCH A, R5	XCH A, R6	XCH A, R7	2
3 XCHD A, @R0	XCHD A, @R1	B1 ad		CALL 1XXH	DIS TCNTI	JT0 ad	CPL A		OUTL P1, A	OUTL P2, A		MOVD P4, A	MOVD P5, A	MOVD P6, A	MOVD P7, A		3
4 ORL A, @R0	ORL A, @R1	MOV A, T	ORL A, #d	JMP 2XXH	STRT CNT	JNT1 ad	SW A		ORL A, R0	ORL A, R1	ORL A, R2	ORL A, R3	ORL A, R4	ORL A, R5	ORL A, R6	ORL A, R7	4
5 ANL A, @R0	ANL A, @R1	JB2 ad	ANL A, #d	CALL 2XXH	STRT T	JT1 ad	DA A		ANL A, R0	ANL A, R1	ANL A, R2	ANL A, R3	ANL A, R4	ANL A, R5	ANL A, R6	ANL A, R7	5
6 ADD A, @R0	ADD A, @R1	MOV T, A		MP 3XXH	STOP TCNT		RRC A		ADD A, R0	ADD A, R1	ADD A, R2	ADD A, R3	ADD A, R4	ADD A, R5	ADD A, R6	ADD A, R7	6
7 ADDC A, @R0	ADDC A, @R1	JB3 ad		CALL 3XXH	ENT0 CLK	JF1 ad	RR A		ADDC A, R0	ADDC A, R1	ADDC A, R2	ADDC A, R3	ADDC A, R4	ADDC A, R5	ADDC A, R6	ADDC A, R7	7
8 MOVX A, @R0	MOVX A, @R1		RET	JMP 4XXH	CLR F0	JNI ad			ORL BUS, #d	ORL P1, #d	ORL P2, #d		ORLD P4, A	ORLD P5, A	ORLD P6, A	ORLD P7, A	8
9 MOVX @R0, A	MOVX @R1, A	JB4 ad	RETR	CALL 4XXH	CPL F0	JNZ ad	CLR C		ANL US, #d	ANL P1, #d	ANL P2, #d		ANLD P4, A	ANLD P5, A	ANLD P6, A	ANLD P7, A	9
A MOV @R0, A	MOV @R1, A		MOV A, @A	JMP 5XXH	CLR F1		CPL C		MOV R0, A	MOV R1, A	MOV R2, A	MOV R3, A	MOV R4, A	MOV R5, A	MOV R6, A	MOV R7, A	A
B MOV @R0, #d	MOV @R1, #d	JB5 ad	JMPP @A	CALL 5XXH	CPL F1	JF0 ad			MOV R0, #d	MOV R1, #d	MOV R2, #d	MOV R3, #d	MOV R4, #d	MOV R5, #d	MDV R6, #d	MOV R7, #d	B
C				JMP 6XXH	SEL RB0	JZ ad	MOV A, PSW		DEC R0	DEC R1	DEC R2	DEC R3	DEC R4	DEC R5	DEC R6	DEC R7	C
D XRL A, @R0	XRL A, @R1	JB6 ad	XRL A, #d	CALL 6XXH	SEL RB1		MOV PSW, A		XRL A, R0	XRL A, R1	XRL A, R2	XRL A, R3	XRL A, R4	XRL A, R5	XRL A, R6	XRL A, R7	D
E			MOV P3 A, @A		JMP 7XXH	SEL MB0	JNC ad	RL A	DJNZ R0, ad	DJNZ R1, ad	DJNZ 2, ad	DJNZ R3, ad	DJNZ R4, ad	DJNZ R5, ad	DJNZ R6, ad	DJNZ R7, ad	E
F MOV A, @R0	MOV A, @R1	B7 ad		CALL 7XXH	SEL MB1	JC ad	RLC A		MOV A, R0	MOV A, R1	MOV A, R2	MOV A, R3	MOV A, R4	MOV A, R5	MOV A, R6	MOV A, R7	F

ШЕСТНАДЦАТИЧНЫЕ КОДЫ КОМАНД КМ1816ВЕ51

(Х – ШЕСТНАДЦАТИЧНАЯ ЦИФРА)

Старшая цифра	Младшая цифра								Старшая цифра
	0	1	2	3	4	5	6	7	
0 NOP	AJMP 0XXH	LJMP ad16	RR A	INC A	INC ad	INC @R0	INC @R1		
1 JBC bit, rel	ACALL 0XXH	LCALL ad16	RRC A	DEC A	DEC ad	DEC @R0	DEC @R1		
2 JB bit, rel	AJMP 1XXH	RET	RL A	ADD A, #d	ADD A, ad	ADD A, @R0	ADD A, @R1		
3 JNB bit, rel	ACALL 1XXH	RETI	RLC A	ADDC A, #d	ADDC A, ad	ADDC A, @R0	ADDC A, @R1		
4 JC rel	AJMP 2XXH	ORL ad, A	ORL ad, #d	ORL A, ad	ORL A, @R0	DRL A, @R1			
5 JNC rel	ACALL 2XXH	ANL ad, A	ANL ad, #d	ANL A, #d	ANL A, ad	ANL A, @R0	ANL A, @R1		
6 JZ rel	AJMP 3XXH	XRL ad, A	XRL ad, #d	XRL A, #d	XRL A, ad	XRL A, @R0	XRL A, @R1		
7 JNZ rel	ACALL 3XXH	ORL C, bit	JMP @A+DPTR	MOV A, #d	MOV ad, #d	MOV @R0, #d	MOV @R1, #d		
8 SJMP rel	AJMP 4XXH	ANL C, bit	MOVC A,@A+PC	DIV AB	MDV add, ads	MOV ad, @R0	MOV ad, @R1		
9 MOV DPTR,#d1	ACALL 4XXH	MOV bit, C	MOVC @A+DPTR	SUBB A, #d	SUBB A, ad	SUBB A, @R0	SUBB A, @R1		
A ORL C,/bit	AJMP 5XXH	MOV C, bit	INC DPTR	MUL AB	MOV @R0, ad	MOV @R1, ad			
B ANL C,/bit	ACALL 5XXH	CPL bit	CPL C	CJNE A, #d, rel	CJNE A, ad, rel	CJNE 0, #d, rel	CJNE R1, #d, rel		
C PUSH ad	AJMP 6XXH	CLR bit	CLR C	SWAP A	XCH A, ad	XCH A, @R0	XCH A, @R1		
D POP ad	ACALL 6XXH	SETB bit	SETB C	DA A	DJNZ ad, rel	XCHD A, @R0	XCHD A, @R1		
E MOVX A,@DPTR	AJMP 7XXH	MOVX A, @R0	MOVX A, @R1	CLR A	MOV A, ad	MOV A, @R0	MOV A, @R1		
F MOVX @DPTR, A	ACALL 7XXH	MOVX @R0, A	MOVX @R1, A	CPL A	MOV ad, A	MOV @R0, A	MOV @R1, A		

Младшая цифра								Старшая цифра
8	9	A	B	C	D	E	F	
INC R0	INC R1	INC R2	INC R3	INC R4	INC R5	INC R6	INC R7	0
DEC R0	DEC R1	DEC R2	DEC R3	DEC R4	DEC R5	DEC R6	DEC R7	1
ADD A, R0	ADD A, R1	ADD A, R2	ADD A, R3	ADD A, R4	ADD A, R5	ADD A, R6	ADD A, R7	2
ADDC A, R0	ADDC A, R1	ADDC A, R2	ADDC A, R3	ADDC A, R4	ADDC A, R5	ADDC A, R6	ADDC A, R7	3
ORL A, R0	ORL A, R1	ORL A, R2	ORL A, R3	ORL A, R4	ORL A, R5	ORL A, R6	ORL A, R7	4
ANL A, R0	ANL A, R1	ANL A, R2	ANL A, R3	ANL A, R4	ANL A, R5	ANL A, R6	ANL A, R7	5
XRL A, R0	XRL A, R1	XRL A, R2	XRL A, R3	XRL A, R4	XRL A, R5	XRL A, R6	XRL A, R7	6
MOV R0, #d	MOV R1, #d	MOV R2, #d	MOV R3, #d	MOV R4, #d	MOV R5, #d	MOV R6, #d	MOV R7, #d	7
MOV ad, R0	MOV ad, R1	MDV ad, R2	MOV ad, R3	MOV ad, R4	MOV ad, R5	MOV ad, R6	MOV ad, R7	8
SUBB A, R0	SUBB A, R1	SUBB A, R2	SUBB A, R3	SUBB A, R4	SUBB A, R5	SUBB A, R6	SUBB A, R7	9
MOV R0, ad	MOV R1, ad	MOV R2, ad	MOV R3, ad	MOV R4, ad	MOV R5, ad	MOV R6, ad	MOV R7, ad	A
CJNE 0, #d, rel	CJNE R1, #d, rel	CJNE R2, #d, rel	CJNE R3, #d, rel	CJNE R4, #d, rel	CJNE R5, #d, rel	CJNE R6, #d, rel	CJNE R7, #d, rel	B
XCH A, R0	XCH A, R1	XCH A, R2	XCH A, R3	XCH A, R4	XCH A, R5	XCH A, R6	XCH A, R7	C
DJNZ R0, rel	DJNZ R1, rel	DJNZ R2, rel	DJNZ R3, rel	DJNZ R4, rel	DJZN R5, rel	DJNZ R6, rel	DJNZ R7, rel	D
MOV A, R0	MOV A, R1	MOV A, R2	MOV A, R3	MOV A, R4	MOV A, R5	MOV A, R6	MOV A, R7	E
MOV R0, A	MOV R1, A	MOV R2, A	MOV R3, A	MOV R4, A	MOV R5, A	MOV R6, A	MOV R7, A	F

Приложение П.6

Таблица П6.1. Система команд однокристального микроконтроллера КМ1816ВЕ48 в алфавитном порядке

Мнемокод	КОП	Мнемокод	КОП	Мнемокод	КОП
ADD A, R0	68	CALL 7xxII	F4	JB2 ad	52
ADD A, R1	69	CLR A	27	JB3 ad	72
ADD A, R2	6A	CLR C	97	JB4 ad	92
ADD A, R3	6B	CLR F0	85	JB5 ad	B2
ADD A, R4	6C	CLR F1	A5	JB6 ad	D2
ADD A, R5	6D	CPL A	37	JB7 ad	F2
ADD A, R6	6E	CPL C	A7	JC ad	F6
ADD A, R7	6F	CPL F0	95	JF0 ad	B6
ADD A, @R0	60	CPL F1	B5	JF1 ad	76
ADD A, @R1	61	DA A	57	JMP 0xxH	04
ADD A, #d	03	DEC A	07	JMP 1xxII	24
ADDC A, R0	78	DEC R0	C8	JMP 2xxH	44
ADDC A, R1	79	DEC R1	C9	JMP 3xxH	64
ADDC A, R2	7A	DEC R2	CA	JMP 4xxII	84
ADDC A, R3	7B	DEC R3	CB	JMP 5xxH	A4
ADDC A, R4	7C	DEC R4	CC	JMP 6xxH	C4
ADDC A, R5	7D	DEC R5	CD	JMP @A	B3
ADDC A, R6	7E	DEC R6	CE	JNC ad	E6
ADDC A, R7	7F	DEC R7	CF	JNI ad	86
ADDC A, @R0	70	DIS I	15	JNT0 ad	26
ADDC A, @R1	71	DJNZ R0, ad	E8	JNT1 ad	46
ADDC A, #d	13	DJNZ R1, ad	F9	JNZ ad	96
ANL A, R0	58	DJNZ R2, ad	EA	JNF ad	16
ANL A, R1	59	DJNZ R3, ad	EB	JNO ad	36
ANL A, R2	5A	DJNZ R4, ad	EC	JTI ad	56
ANL A, R3	5B	DJNZ R5, ad	ED	JZ ad	C6
ANL A, R4	5C	DJNZ R6, ad	EE	MOV A, PSW	C7
ANL A, R5	5D	DJNZ R7, ad	EF	MOV A, R0	F8
ANL A, R6	5E	EN TCNT1	05	MOV A, RI	F9
ANL A, R7	5F	FNT0 CLK	25	MOV A, R2	FA
ANL A, @R0	50	IN A, P1	75	MOV A, R3	FB
ANL A, @R1	51	IN A, P2	09	MOV A, R4	FC
ANL A, #d	53	IN A, P5	0A	MOV A, R5	FD
ANI RUS, #d	98	INC A	17	MOV A, R6	FF
ANL P1, #d	99	INC R0	18	MOV A, R7	FF
ANL P2, #d	9A	INC R1	19	MOV A, T	42
ANLD P4, A	9C	INC R2	1A	MOV A, @R0	F0
ANLD P5, A	9D	INC R3	1B	MOV A, @R1	F1
ANLD P6, A	9E	INC R4	1C	MOV A, #d	23
ANLD P7, A	9F	INC R5	1D	MOV PSW, A	D7
CALL 0xxII	14	INC R6	1E	MOV R0, A	A8
CALL 1xxII	34	INC R7	1F	MOV R0, #d	B8
CALL 2xxII	54	INC @R0	10	MOV R1, A	A9
CALL 3xxII	74	INC @R1	11	MOV R1, #d	B9
CALL 4xxII	94	INS A, BUS	08	MOV R2, A	AA
CALL 5xxII	B4	JB0 ad	12	MOV R2, #d	BA
CALL 6xxII	D4	JB1 ad	32	MOV R2, #d	BA

Мнемокод	КОП	Мнемокод	КОП	Мнемокод	КОП
MOV R3, A	AB	ORL A, R0	48	SEL RBI	D5
MOV R3, #d	BB	ORL A, RI	49	STOP TCNT	65
MOV R4, A	AC	ORL A, R2	4A	STRT CNT	45
MOV R4, #d	BC	ORL A, R3	4B	STRT T	55
MOV R5, A	AD	ORL A, R4	4C	SWAP A	47
MOV R5, #d	BD	ORL A, R5	4D	XCH A, R0	28
MOV R6, A	AE	ORL A, R6	4E	XCH A, R1	29
MOV R6, #d	BE	ORL A, R7	4F	XCH A, R2	2A
MOV R7, A	AF	ORL A, @R0	40	XCH A, R3	2B
MOV R7, #d	BF	ORL A, @RI	41	XCH A, R4	2C
MOV T, A	62	ORL A, #d	43	XCH A, R5	2D
MOV @R0, A	A0	ORL BUS, #d	88	XCH A, R6	2E
MOV @R0, #d	B0	ORL P1, #d	89	XCH A, R7	2F
MOV @R1, A	A1	ORL P2, #d	8A	XCH A, @R0	20
MOV @R1, #d	B1	ORLD P4, A	8C	XCH A, @R1	21
MOVD A, P4	0C	ORLD P5, A	8D	XCHD A, @R0	30
MOVD A, P5	0D	ORLD P6, A	8E	XCHD A, @R1	31
MOVD A, P6	0E	ORLD P7, A	8F	XRL A, R0	D8
MOVD A, P7	0F	OUTL BUS, A	02	XRL A, R1	D9
MOVD P4, A	3C	OUTL P1, A	39	XRL A, R2	DA
MOVD P5, A	3D	OUTL P2, A	3A	XRL A, R3	DB
MOVD P6, A	3E	RET	83	XRL A, R4	DC
MOVD P7, A	3F	RENR	93	XRL A, R5	DD
MOVPA @A	A3	RL A	E7	XRL A, R6	DE
MOVPA @A	81	RLC A	F7	XRL A, R7	DF
MOVX A, @R0	80	RR A	77	XRL A, @R0	D0
MOVX A, @R1	E3	RRC A	67		
MOVX @R0, A	90	SEL MB0	F5	XRL A, @R1	D1
MOVX @R1, A	91	SEL MB1	F5		
NOP	00	SEL RB0	C5	XRL A, #d	D3

Таблица П7.1. Система команд однокристального микроконтроллера КМ1816ВЕ51 в алфавитном порядке

Мнемокод	КОП	Мнемокод	КОП	Мнемокод	КОП
ACALL 0xxII	11	ADD A, R2	2A	ADDC A, R1	39
ACALL 1xxH	31	ADD A, R3	2B	ADDC A, R2	3A
ACALL 2xxH	51	ADD A, R4	2C	ADDC A, R3	3B
ACALL 3xxH	71	ADD A, R5	2D	ADDC A, R4	3C
ACALL 4xxII	91	ADD A, R6	2E	ADDC A, R5	3D
ACALL 5xxH	B1	ADD A, R7	2F	ADDC A, R6	3E
ACALL 6xxH	DI	ADD A, @R0	26	ADDC A, R7	3F
ACALL 7xxH	F1	ADD A, @R1	27	ADDC A, @R0	36
ADD A, ad	25	ADD A, #d	24	ADDC A, @R1	37
ADD A, R0	28	ADDC A, ad	35	ADDC A, #d	34
ADD A, R1	29	ADDC A, R0	38	AJMP 0xxII	01

Мнемокод	КОП	Мнемокод	КОП	Мнемокод	КОП
AJMP	IxxH	21	DEC	R4	IC
AJMP	2xxH	41	DEC	R5	1D
AJMP	3xxH	61	DEC	R6	1E
AJMP	4xxH	81	DEC	R7	1F
AJMP	5xxH	A1	DEC	@R0	16
AJMP	6xxH	C1	DEC	@RI	17
ANL	A, ad	55	DJNZ	ad, rel	D5
ANL	A, R0	58	DJNZ	R0, rel	D8
ANL	A, R1	59	DJNZ	R1, rel	D9
ANL	A, R2	5A	DJNZ	R2, rel	DA
ANL	A, R3	5B	DJNZ	R3, rel	DB
ANL	A, R4	5C	DJNZ	R4, rel	DC
ANL	A, R5	5D	DJNZ	R5, rel	DD
ANL	A, R6	5E	DJNZ	R6, rel	DE
ANL	A, R7	5F	DJNZ	R7, rel	DF
ANL	A, @R0	56	INC	A	04
ANL	A, @R1	57	INC	ad	05
ANL	A, #d	54	INC	DPTR	A3
ANL	ad, A	52	INC	R0	08
ANL	ad, #d	53	INC	R1	09
ANL	C, bit	82	INC	R2	0A
ANL	C/bit	B0	INC	R3	0B
CJNE	A, ad, rel	B5	INC	R4	0C
CJNE	A, #d, rel	B4	INC	R5	0D
CJNE	R6, # d, rel	B8	INC	R6	0E
CJNE	R1, #d, rel	B9	INC	R7	0F
CJNE	R2, # d, rel	BA	INC	@R0	06
CJNE	R3, # d, rel	BB	INC	@RI	07
CJNE	R4, # d, rel	BC	JB	bit, rel	20
CJNE	R5, #d, rel	BD	JBC	bit, rel	10
CJNE	R6, # d, rel	BE	JC	rel	40
CJNE	R7, #d, rel	BF	JMP	@A + DPTR	73
CJNE	@R0, #d, rel	B6	JNB	bit, rel	30
CJNE	@R1, #d, rel	B7	JNC	rel	50
CLR	A	E4	JNZ	rel	70
CLR	bit	C2	JZ	rel	60
CLR	C	C3	LCALL	ad16	12
CPL	A	F4	LJMP	ad16	02
CPL	bit	B2	MOV	A, ad	E5
CPL	C	B3	MOV	A, R0	E8
DA	A	D4	MOV	A, RI	E9
DEC	A	14	MOV	A, R2	EA
DEC	ad	15	MOV	A, R3	EB
DEC	R0	18	MOV	A, R4	EC
DEC	R1	19	MOV	A, RS	ED
DEC	R2	1A	MOV	A, R6	EE
DEC	R3	1B	MOV	A, R7	EF

MOV	@R1,#d	77	ORL	C,/bit	A0	XCH	A, R0	C8
MOVC	A, @+DPTR	93	POP	ad	D0	XCH	A, R1	C9
MOVC	A, @+PC	83	PUSH	ad	C0	XCH	A, R2	CA
MOVX	A, @DPTR	E0	RET		22	XCH	A, R3	CB
MOVX	A, @R0	E2	RET 1		32	XCH	A, R4	CC
MOVX	A, @R1	E3	RL	A	23	XCH	A, R5	CD
MOVX	@DPTR, A	F0	RLC	A	33	XCH	A, R6	CE
MOVX	@R0, A	F2	RR	A	03	XCH	A, R7	CF
MOVX	@RI, A	F3	RRC	A	13	XCH	A, @R0	C6
MUL	AB	A4	SETB	bit	D2	XCH	A, @R1	C7
NOP		00	SETB	C	D3	XCHD	A, @R0	D6
ORL	A, ad	45	SJMP	rel	80	XCHD	A, @RI	D7
ORL	A, R0	48	SUBB	A, ad	95	XRL	A, ad	65
ORL	A, R1	49	SUBB	A, R0	98	XRL	A, R0	68
ORL	A, R2	4A	SUBB	A, R1	99	XRL	A, R1	69
ORL	A, R3	4B	SUBB	A, R2	9A	XRL	A, R2	6A
ORL	A, R4	4C	SUBB	A, R3	9B	XRL	A, R3	6B
ORL	A, R5	4D	SUBB	A, R4	9C	XRL	A, R4	6C
ORL	A, R6	4E	SUBB	A, R5	9D	XRL	A, R5	6D
ORL	A, R7	4F	SUBB	A, R6	9E	XRL	A, R6	6E
ORL	A, @R0	46	SUBB	A, R7	9F	XRL	A, R7	6F
ORL	A, @RI	47	SUBB	A, @R0	96	XRL	A, @R0	66
ORL	A, #d	44	SUBB	A, @R1	97	XRL	A, @R1	67
ORL	ad, A	42	SUBB	A, #d	94	XRL	A, #d	64
ORL	ad, #d	43	SWAP	A	C4	XRL	ad, A	62
ORL	C, bit	72	XCH	A, ad	C5	XRL	ad, #d	63

Список литературы

1. Алексенко А.Г., Галицин А.А., Иванников А.Д. Проектирование радиоэлектронной аппаратуры на микропроцессорах. М.: Радио и связь, 1984.
2. Алексенко А.Г., Шагурин И.И. Микросхемотехника: Учебное пособие/ Под ред. И.П. Степаненко. М.: Радио и связь, 1982.
3. Баранов С.И. Синтез микропрограммных автоматов. Л.: Энергия, Ленингр. отд-ние, 1979.
4. Видениекс П.О., Ветиниш Я.Я., Кривченков А.А. Проблемно-ориентированные микропроцессорные системы в производстве РЭА. М.: Радио и связь, 1987.
5. Вирт Н. Систематическое программирование. Введение. М.: Мир, 1977.
6. Ги К. Введение в локальные вычислительные сети. М.: Радио и связь, 1986.
7. Григорьев В.Л. Программное обеспечение микропроцессорных систем. М.: Энергоатомиздат, 1983.
8. Дмитриенко А.П., Старostenко О.В. Контроллер алфавитно-цифрового индикатора на базе однокристальной микроЭВМ// Микропроцессорные средства и системы. 1986, № 6. С. 90–91.
9. Евлампиев Р.А., Галузо Е.В., Головаинов В.П. Отладочная система для однокристальной микроЭВМ КМ1816ВЕ48// Микропроцессорные средства и системы. 1986, № 3. С. 32–33.
10. Игнатющенко В.В. Организация структур управляющих микропроцессорных вычислительных систем. М.: Энергоатомиздат, 1984.
11. Каган Б.М., Стасин В.В. Микропроцессоры в цифровых системах. М.: Энергия, 1979.
12. Каган Б.М., Стасин В.В. Основы проектирования микропроцессорных устройств автоматики. М.: Энергоатомиздат, 1987.
13. Кобылинский А.В., Липовецкий Г.П. Однокристальные микроЭВМ серии КМ1816// Микропроцессорные средства и системы. 1986, № 1. С. 10.
14. Кобылинский А.В., Сабадаш Н.Г., Тесленко А.К. Система автоматизации программирования однокристальной микроЭВМ// Микропроцессорные средства и системы. 1986, № 3. С. 23–25.
15. Клингман Э. Проектирование микропроцессорных систем. М.: Мир, 1980.
16. Кросс-система для однокристальной микроЭВМ КМ1816ВЕ48/ Г.Я. Зобин, Е.С. Кривопальцев, А.Б. Минкович и др./ Микропроцессорные средства и системы. 1986, № 3. С. 27–30.
17. Кушнир В.Е., Панфилов Д.А., Шаронин С.Г. Учебная микроЭВМ на основе однокристальной ЭВМ КМ1816ВЕ48// Микропроцессорные средства и системы. 1986, № 6. С. 75–82.
18. Микропроцессоры / Под ред. Л.Н. Преснухина. М.: Высшая школа, 1986.
19. Микропроцессоры: Системы программирования и отладки/ В.А. Мясников, М.Б. Игнатьев, А.А. Кочкин, Ю.Е. Шейин; Под ред. В.А. Мясникова, М.Б. Игнатьева. М.: Энергоатомиздат, 1985.
20. Модуль процессора персональной ЭВМ "Ириша"/ В.Н. Барышников, В.П. Быстров, М.А. Воронов и др./ Микропроцессорные средства и системы. 1986, № 2. С. 52–62.

21. Новик Г.Х., Стасин В.В. Проектирование цифровых устройств управления объектами на основе однокристального микроконтроллера КМ1816ВЕ48. М.: Машиностроение, 1986.
22. Перспективные однокристальные ЭВМ/ М.Г. Венноватов, Г.В. Карапуба, В.В. Павлов, В.С. Старшова// Микропроцессорные средства и системы. 1987, № 2. С. 7–8.
23. Прангишвили И.В. Микропроцессоры и локальные сети микроЭВМ в распределенных системах управления. М.: Энерготомиздат, 1985.
24. Полупроводниковые БИС запоминающих устройств: Справочник/ Под ред. А.Ю. Гордонова и Ю.Н. Дьякова. М.: Радио и связь, 1986.
25. Стасин В.В. Микропроцессоры и принципы управления ними. М.: Машиностроение, 1985.
26. Стасин В.В., Мологонцева О.Ф. Микропроцессорная реализация типовых функций управления. М.: Заочный институт ЦП НТО Приборпром, 1987.
27. Универсальный микроконтроллер "Электроника МК-48" из основе однокристальной микроЭВМ КМ1816ВЕ48/ Ю.А. Акуней, Б.В. Антонов, А.Г. Маликов и др./ Микропроцессорные средства и системы. 1986, № 3. С. 39–40.
28. Флинт Д. Локальные сети ЭВМ. Архитектура, принципы построения, реализация. М.: Финанссы и статистика, 1986.
29. Шевкопляс Б.В. Микропроцессорные структуры. Инженерные решения. М.: Радио и связь, 1986.
30. Шереметьевский Н.Н., Долкарт В.М. Магистрально-модульные микросредства управляющей вычислительной техники// Микропроцессорные средства и системы. 1984, № 2. С. 24–27.
31. Intel Distributed control Modules data book. Intel Corp. OEM Modules Operation. 5200 NE. Elam Yang Parkway, Hillsboro, Oregon 97123, 1984. – 96 р.
32. Intel 8051. User's manual. Intel Corp., 1980. – 110 р.
33. ISIS-II MCS51 Macro-assembler. Intel Corp., 1980. – 80 р.
34. ISIS-II MCS48 Macro-assembler. Intel Corp., 1979. – 46 р.
35. ISIS-II. Systems User's Guide. Intel Corp. USA. 1977. – 125 р.
36. MCS 48. Family of single-chip microcomputers. User's manual. Intel Corp. July 1979. – 320 р.
37. Автономные обучающие устройства для ОЭВМ серии 1816/ В.И. Жданов, В.Н. Бобылев, Н.Ф. Гринь, Т.Г. Уткина// Микропроцессорные средства и системы. 1988, № 5. С. 72–74.
38. Бровко В.И. Комплекс для отладки изделий на основе однокристальной ЭВМ КМ1816ВЕ51// Микропроцессорные средства и системы. 1987, № 6. С. 40–43.
39. Бродин Б.В., Шагурин И.И. Микропрограммируемый скомпилируемый эмулятор для отладки микропроцессорных систем// Микропроцессорные средства и системы. 1988, № 5. С. 49–53.
40. Добуш Ю.Д., Старостенко О.В., Натопта Е.Е. Клавиатура на базе однокристальной микроЭВМ КМ1816BF48// Микропроцессорные средства и системы. 1988, № 1. С. 91–92.
41. Каменецкий С.В. Способ подключения 16-контактной клавиатуры к КМ1816ВЕ48// Микропроцессорные средства и системы. 1988, № 4. С. 7.
42. Кушнер В.Е., Панфилов Д.И., Шаронин С.Г. Многофункциональный комплекс программно-аппаратных средств для семейства однокристальных ЭВМ серии 1816// Микропроцессорные средства и системы. 1988, № 6. С. 3–4.
43. Стасин В.В., Урусов А.В., Мологонцева О.Ф. Примеры проектирования устройств и систем на основе однокристального микроконтроллера КМ1816ВЕ51. М.: Заочный ин-т повышения квалификации ИТР ЦП ВНТО им. С.И. Вавилова, 1989.

Оглавление

Предисловие	3
Список сокращений, символьических имен и аббревиатур	5
Г л а в а 1. Особенности проектирования микроконтроллерных устройств управления объектами	9
1.1. Введение в предметную область	9
1.2. Основные положения	10
1.3. Структура МК-системы управления	13
1.4. Особенности разработки аппаратурных средств МК-систем	14
1.5. Особенности разработки прикладного программного обеспечения МК-систем	15
Г л а в а 2. Структурная организация и система команд микроконтроллера КМ1816ВЕ48	19
2.1. Функциональное назначение выводов корпуса МК48	19
2.2. Структурная схема МК48	22
2.2.1. Арифметико-логическое устройство	22
2.2.2. Память микроконтроллера	23
2.2.3. Организация ввода/вывода информации	25
2.2.4. Устройство управления микроконтроллера	27
2.3. Система команд МК48	30
2.3.1. Общие сведения о системе команд	30
2.3.2. Группа команд пересылки данных	31
2.3.3. Группа команд арифметических операций	34
2.3.4. Группа команд логических операций	34
2.3.5. Группа команд передачи управления	34
2.3.6. Группа команд управления режимом работы МК	41
2.4. Особенности работы МК48 на этапе отладки прикладных программ	43
2.4.1. Микроконтроллер в пошаговом режиме работы и в режиме внешнего доступа	43
2.4.2. Загрузка прикладных программ в резидентную память микроконтроллера	44
2.5. Варианты структурной организации систем на основе МК48	46
2.5.1. МК-система с внешней памятью программ	46
2.5.2. МК-система с внешней памятью данных	47
2.5.3. МК-система с расширенным вводом/выводом	48
Г л а в а 3. Структурная организация и система команд микроконтроллера КМ1816ВЕ51	50
3.1. Структурная схема МК51	50
3.1.1. Арифметико-логическое устройство	50
3.1.2. Резидентная память	52
3.1.3. Устройство управления и синхронизации	55
3.2. Порты ввода/вывода информации	57
3.3. Доступ к внешней памяти	60
3.4. Таймер/счетчик	65
3.5. Последовательный интерфейс	67

3.5.1. Регистр управления/статуса УАПП	68
3.5.2. Работа УАПП в мультимикроконтроллерных системах	69
3.5.3. Скорость приема/передачи	70
3.5.4. Особенности работы УАПП в различных режимах	71
3.6. Система прерываний	76
3.7. Особые режимы работы МК51	79
3.7.1. Режим загрузки и верификации прикладных программ	81
3.7.2. Работа МК51 в пошаговом режиме	81
3.7.3. Сброс, режим холостого хода и режим пониженного энергопотребления	81
3.8. Система команд МК51	83
3.8.1. Общие сведения о системе команд	83
3.8.2. Группа команд передачи данных	85
3.8.3. Группа команд арифметических операций	91
3.8.4. Группа команд логических операций	91
3.8.5. Группа команд операций с битами	92
Г л а в а 4. Методика разработки прикладного программного обеспечения МК-систем	97
4.1. Формализованный подход к разработке прикладных программ	97
4.2. Элементы формализации в разработке алгоритмов	101
4.3. Процедуры и подпрограммы	104
4.4. Правила записи программ на языке ассемблера	106
4.5. Ввод, редактирование, трансляция и отладка прикладных программ в кросс-системах разработки	109
4.6. Отладка прикладного программного обеспечения микроконтроллеров	110
Г л а в а 5. Обработка данных в микроконтроллерах МК48 и МК51	113
5.1. Примеры программ обработки данных в МК48	113
5.1.1. Примеры использования команд передачи данных	113
5.1.2. Примеры использования команд арифметических операций	114
5.1.3. Примеры использования команд логических операций	117
5.1.4. Примеры использования команд передачи управления и команд управления режимом МК48	118
5.2. Примеры программ обработки данных в МК51	121
5.2.1. Примеры использования команд передачи данных	121
5.2.2. Примеры использования команд арифметических операций	123
5.2.3. Примеры использования команд логических операций	125
5.2.4. Примеры операций с битами	126
Г л а в а 6. Организация взаимодействия микроконтроллера с объектом управления	127
6.1. Ввод информации с датчиков	129
6.1.1. Опрос двоичного датчика. Ожидание события	129
6.1.2. Устранение дребезга контактов	132
6.1.3. Подсчет числа импульсов	134
6.1.4. Опрос группы двоичных датчиков	136
6.2. Выход управляющих сигналов из МК	139
6.2.1. Формирование статических сигналов	139
6.2.2. Формирование импульсных сигналов	140
6.3. Масштабирование	141
6.4. Реализация функций времени	142
6.4.1. Программное формирование временной задержки	142
6.4.2. Формирование временной задержки на основе таймеров	144
6.4.3. Измерение временных интервалов	145
6.5. Преобразование кодов	146

6.5.1. Простейшие преобразования	147
6.5.2. Преобразования параллельных и последовательных кодов	149
6.5.3. Цифро-аналоговые и аналого-цифровые преобразования	151
Г л а в а 7. Организация связи с оператором в обслуживаемых МК-системах	
7.1. Ввод информации с клавиатуры	156
7.2. Вывод и отображение информации	156
7.3. Сопряжение МК с клавиатурой и линейным дисплеем на основе БИС KP580ВД79	162
Г л а в а 8. Примеры проектирования МК-устройств и систем	
8.1. Устройство формирования звуковых сигналов	170
8.2. Кодовый замок зуммерного типа	174
8.3. Отладочный модуль на основе МК48	180
8.4. Локальная управляющая микросеть на основе МК51	184
8.5. Контроллер одностороннего дисплея на матричных индикаторах	190
Приложение П1. Контроллер клавиатуры/дисплея KP580ВД79	194
Приложение П2. Расширитель ввода/вывода KP580ВР43	199
Приложение П3. Использование последовательного порта микроконтроллера МК51 для связи с интерфейсом RS-232	207
Приложение П4. Система команд микроконтроллера KM1816BE48 (матрица 16 x 16)	208
Приложение П5. Система команд микроконтроллера KM1816BE51 (матрица 16 x 16)	212
Приложение П6. Система команд микроконтроллера KM1816BE48 в алфавитном порядке	214
Приложение П7. Система команд микроконтроллера KM1816BE51 в алфавитном порядке	216
	217

Производственное издание

СТАШИН ВЛАДИСЛАВ ВИКТОРОВИЧ

УРУСОВ АНДРЕЙ ВЛАДИМИРОВИЧ

МОЛОГОНЦЕВА ОЛЬГА ФЕДОРОВНА

**ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ УСТРОЙСТВ
НА ОДНОКРИСТАЛЬНЫХ МИКРОКОНТРОЛЛЕРАХ**

Редактор А.Д. Иванников

Редактор издательства З.И. Михеева

Художественные редакторы Т.А. Дворецкова, Г.И. Панфилова

Технический редактор Г.В. Преображенская

Корректор С.В. Малышева

ИБ № 2801

Набор выполнен в издательстве. Подписано в печать с оригинала-макета 26.01.90
Т-06658. Формат 60 x 90 1/16. Бумага офсетная № 1. Гарнитура Пресс Роман. Печать
офсетная. Усл. печ. л. 14,0. Усл. кр.-отт. 30,5. Уч.-изд. л. 16,42. Тираж 70 000 экз.
Заказ 2720 Цена 1 р. 10 к.

Энергоатомиздат, 113114, Москва, М-114, Шлюзовая наб., 10.

Предприятие малообъемной книги дважды ордена Трудового Красного Знамени
Ленинградского производственного объединения "Типография им. И.Ф. Федорова
при Государственном Комитете СССР по печати.
192007, Ленинград, ул. Боровая, 51.