

СПРАВОЧНИК

Однокристалльные

МИКРОЭЛЕМЕНТЫ

Однокристалльные микроЭВМ



БИНОМ



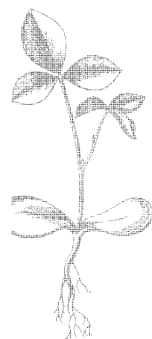
БИНОМ

СПРАВОЧНИК

Однокристалльные микро ЭВМ



БИНОМ
1994г.



Scan AAW

ББК 32.852

*Боборыкин А.В., Липовецкий Г.П., Литвинский Г.В., Оксиль О.Н.,
Прохорчик С.В., Проценко Л.В., Петренко Н.В., Сергеев А.А.,
Сивобород П.В.*

Однокристалльные микроЭВМ. М.: МИКАП, 1994, — 400 с.: ил. —
ISBN 5-85959-030-X

Приведено подробное техническое описание однокристалльных микроЭВМ семейств МК48, МК51 и UPI-42. Рассмотрены зарубежные аналоги описанных микросхем.

Художественное оформление Р. Бушуева

ISBN 5-85959-030-X

© Состав, оформление. Фирма "БИНОМ", 1994

ПРЕДИСЛОВИЕ

Одной из характерных особенностей нынешнего этапа научно-технического прогресса является все более широкое применение микроэлектроники в различных отраслях народного хозяйства. Роль микроэлектроники в развитии общественного производства определяется ее практически неограниченными возможностями в решении различных задач во всех областях народного хозяйства, глубоким влиянием на культуру и быт современного человека.

Особое внимание в настоящее время уделяется внедрению микропроцессоров, обеспечивающих решение задач автоматизации управления механизмами, приборами и аппаратурой. Адаптация микропроцессора к особенностям конкретной задачи осуществляется в основном путем разработки соответствующего программного обеспечения, заносимого затем в память программ. Аппаратная адаптация в большинстве случаев осуществляется путем подключения необходимых интегральных схем обрaмления и организации ввода-вывода, соответствующих решаемой задаче.

В микропроцессорной технике выделился самостоятельный класс больших интегральных схем (БИС) — однокристалльные микроЭВМ (ОМЭВМ), которые предназначены для "интеллектуализации" оборудования различного назначения. Архитектура однокристалльных микроЭВМ — результат эволюции архитектуры микропроцессоров и микропроцессорных систем, обусловленной стремлением существенно снизить их аппаратные затраты и стоимость. Как правило, эти цели достигаются как путем повышения уровня интеграции БИС, так и за счет поиска компромисса между стоимостью, аппаратными затратами и техническими характеристиками ОМЭВМ.

ОМЭВМ представляют собой приборы, конструктивно выполненные в виде одной БИС и включающие в себя все устройства, необходимые для реализации цифровой системы управления минимальной конфигурации: процессор, запоминающее устройство данных, запоминающее устройство команд, внутренний генератор тактовых сигналов, а также программируемые интегральные схемы для связи с внешней средой. Использование ОМЭВМ в системах управления обеспечивает достижение исключительно высоких показателей эффективности при столь низкой стоимости (во многих применениях система может состоять только из одной БИС ОМЭВМ), что им, видимо, нет в ближайшем времени альтернативной элементной базы для построения управляющих и/или регулирующих систем. В настоящее время более двух третей мирового рынка микропроцессорных средств составляют именно БИС ОМЭВМ. В некоторых публикациях однокристалльную микроЭВМ (ОМЭВМ) называют "микроконтроллер". Обосновывается это тем обстоятельством, что такие микросхемы имеют незначительные емкости памяти, физическое и логическое разделение памяти программ (ПЗУ) и памяти данных (ОЗУ), упрощенную и ориентированную на задачи управления систему команд, примитивные методы адресации команд и данных. Специфическая организация ввода-вывода информации предопределяет область их применения в качестве специализированных вычислителей, включенных в контур управления объектом или процессом. Структурная организация, набор команд и аппаратно-программные средства ввода-вывода информации этих микросхем лучше всего приспособлены для решения задач управления и регулирования в приборах, устройствах и системах автоматики, а не для решения задач обработки данных.

Указанные выше соображения отражают технический уровень ОМЭВМ в настоящий момент. Тем не менее, в предлагаемой вниманию читателей книге используется понятие однокристалльная микроЭВМ, так как последние достижения технологии изготовления СБИС значительно увеличивают их

степень интеграции, что позволяет устранить указанные выше ограничения в части возможностей решения задач обработки данных.

Наименование "микроконтроллер" в описании используется в аббревиатуре МК для обозначения семейства ОМЭВМ, объединенных рядом общих признаков, например, разрядностью, системой команд, набором функциональных блоков и т. д.

Настоящая книга посвящена описанию 8-разрядных ОМЭВМ семейств МК48, МК51 и микросхемы ЭКР1847ВГ6. Каждая из моделей ОМЭВМ этих семейств имеет соответствующий аналог ОМЭВМ, входящих в состав широко известных в мире семейств MCS-48, MCS-51 и UPI-42 фирмы Intel, США.

Основное внимание в книге уделено подробному описанию структурной организации, режимов работы и системы команд, приведены электрические параметры, режимы и условия эксплуатации, а также примеры использования микросхем указанных семейств ОМЭВМ.

Авторами глав 1 и 2 являются Липовецкий Г. П., Литвинский Г. В., Оксинь О. Н. Проценко Л. В. Петренко Н. В. и Сивобород П. В.; главы 8 — Боборыкин А. В., Прохорчик С. В. и Сергеев А. А.

Авторы будут благодарны всем читателям, которые сочтут возможным прислать свои замечания и предложения по адресу:

103473, г. Москва—473, а/я 133, БИНОМ.

СОДЕРЖАНИЕ

ГЛАВА 1. ОДНОКРИСТАЛЛЬНЫЕ МИКРОЭВМ СЕМЕЙСТВА МК48	8
1.1. Общие сведения об однокристальных микроЭВМ семейства МК48	8
1.2. Структурная организация ОМЭВМ	11
1.2.1. Процессор.....	11
1.2.2. Память программ.....	16
1.2.3. Память данных	18
1.2.4. Каналы ввода-вывода	19
1.2.5. Таймер-счетчик	22
1.2.6. Система прерываний	24
1.2.7. Устройство управления и синхронизации.....	25
1.3. Режимы работы	27
1.3.1. Режим проверки внутренней памяти программ	27
1.3.2. Режим программирования внутреннего ППЗУ	30
1.3.3. Режим работы с внутренней памятью программ.....	32
1.3.4. Режим работы с внешней памятью	33
1.3.5. Режим пошагового выполнения команд.....	37
1.3.6. Расширение канала ввода-вывода	38
1.3.7. Режим уменьшенного энергопотребления	42
1.4. Система команд	44
1.5. Описание машинных команд	55
1.6. Электрические параметры	87
1.7. Применение ОМЭВМ	95
1.7.1. Совместная работа с устройствами аналогового ввода-вывода	95
1.7.2. Применение таблиц для вычисления функций	98
1.7.3. Реализация приема-передачи последовательного кода.	100
1.7.4. Прием последовательного кода по прерываниям с использованием таймера ОМЭВМ	103
1.7.5. Реализация передачи последовательного кода.	105
1.7.6. Программа реализации контроля по четности.	106
ГЛАВА 2. ОДНОКРИСТАЛЛЬНЫЕ МИКРОЭВМ СЕМЕЙСТВА МК51	107
2.1. Общие сведения об однокристальных микроЭВМ семейства МК51	107
2.2. Структурная организация ОМЭВМ	112
2.2.1. Блок управления. Синхронизация микроЭВМ. Регистр PCON. Режимы уменьшенного энергопотребления.....	116
2.2.2. Арифметико-логическое устройство (АЛУ). Регистр PSW.....	121
2.2.3. Блок таймеров/счетчиков. Регистры TMOD и TCON	121
2.2.4. Блок последовательного интерфейса и прерываний. Регистры SCON, IP, IE	128
2.2.5. Счетчик команд. Регистр DPTR	131
2.2.6. Порты.....	131
2.2.7. Память данных	137
2.2.8. Память программ.....	138
2.3. Описание функционирования ОМЭВМ.....	140
2.3.1. Режимы работы и начальная установка	141
2.3.1.1. Работа с внешней памятью программ	144
2.3.1.2. Работа с внутренней памятью программ	144
2.3.1.3. Работа с памятью данных.....	144
2.3.1.4. Программирование ОМЭВМ KM1816BE751.....	146
2.3.1.5. Проверка внутренней памяти программ	149
2.3.2. Работа с портами	149
2.3.3. Работа с последовательным портом	151
2.3.4. Структура прерываний.....	160
2.3.5. Организация памяти.....	166
2.4. Система команд.....	171
2.4.1. Арифметические команды	182
2.4.2. Логические команды с байтовыми переменными.	185
2.4.3. Команды передачи данных	185
2.4.4. Команды битового процессора.....	186
2.4.5. Команды ветвления и передачи управления	186
2.4.6. Способы адресации операндов.....	186

2.4.6.1	Регистровая адресация.....	187
2.4.6.2	Прямая адресация.....	187
2.4.6.2	Косвенно-регистровая адресация.....	187
2.4.6.4	Непосредственная адресация.....	187
2.4.6.5	Косвенно-регистровая адресация по сумме базового и индексного регистров.....	187
2.5.	Описание машинных команд	187
2.6.	Электрические параметры	219
2.7.	Примеры применения.....	223
2.7.1.	Примеры программирования микроЭВМ	224
2.7.1.1.	Программа преобразования системы счисления	224
2.7.1.2.	Арифметические операции двойной точности	225
2.7.1.3.	Последовательности просмотра таблиц.....	226
2.7.1.4.	Сохранение состояния центрального процессора во время прерываний.....	227
2.7.1.5.	Условный переход с N-ветвлениями.....	228
2.7.1.6.	Последовательная передача параметров	229
2.7.2.	Программные методы сопряжения периферийных устройств	230
2.7.2.1.	Переконфигурация портов ввода-вывода	230
2.7.2.2.	Сопряжение с расширителем ввода-вывода КР580ВР43	231
2.7.2.3.	Программная организация задержки	231
2.7.2.4.	Конфигурация последовательного порта и таймера	231
2.7.2.5.	Простые драйверы для последовательного ввода-вывода	232
2.7.2.6.	Передача символьной строки через последовательный порт.....	233
2.7.2.7.	Чтение содержимого таймера/счетчика.....	233
2.7.3.	Схемы включения ОМЭВМ.....	234
ГЛАВА 3.	УСЛОВИЯ ЭКСПЛУАТАЦИИ.....	237
ГЛАВА 4.	КМОП ОМЭВМ КМ1830ВЕ751/КМ1830ВЕ753	246
4.1.	Специальный режим эмуляции	246
4.2.	Защита внутренней памяти программ.....	246
4.2.1.	Шифровальная таблица.....	246
4.2.2.	Биты защиты памяти программ	247
4.3.	Программирование	247
4.4.	Чтение внутренней памяти программ	250
4.5.	Стирание УФППЗУ.....	250
4.6.	Электрические параметры	251
ГЛАВА 5.	ХАРАКТЕРИСТИКИ АНАЛОГОВ ФИРМЫ INTEL	253
ГЛАВА 6.	БИТОВЫЙ ПРОЦЕССОР ОМЭВМ СЕМЕЙСТВА МК51	258
6.1.	Общие сведения.....	258
6.2.	Прямая адресация битов	259
6.3.	Система команд.....	264
6.4.	Применение битового процессора.....	266
ГЛАВА 7.	АНАЛОГИ, ПРОИЗВОДИМЫЕ В РОССИИ И БЕЛАРУСИ.....	289
7.1.	КМОП БИС 8-разрядной микроЭВМ ЭКР/КР/КА1835ВЕ49(39)	289
7.2.	КМОП БИС 8-разрядной микроЭВМ КР1835ВЕ51(31)	292
7.3.	Однокристалльный эмулятор для микроЭВМ КР(ЭКР)1830ВЕ51.....	294
7.4.	Однокристалльные микроЭВМ ЭКР1830ВЕ31М/51М.....	295
7.5.	Однокристалльная микроЭВМ К(Р)1850ВЕ48/50	295
7.6.	Однокристалльная КМОП микроЭВМ К(Р)1850ВЕС48/50	296
7.7.	Однокристалльная микроЭВМ К(Р)1850ВЕ651	297
ГЛАВА 8.	КМОП МИКРОКОНТРОЛЛЕРЫ ЭКР1847ВГ6. СЕМЕЙСТВО UPI-42	298
8.1.	Общие сведения.....	298
8.2.	Функциональное описание.....	300
8.2.1.	Процессор.....	301
8.2.2.	Память программ.....	305
8.2.3.	Память данных	306
8.2.4.	Устройство генератора и синхронизации.....	308
8.2.5.	Таймер-счетчик (Т/С)	311
8.2.6.	Система прерываний	313
8.2.7.	Прерывания головной системы и DMA.....	315
8.2.8.	Сброс.....	315
8.2.9.	Буфер шины данных	317
8.2.10.	Системный интерфейс 8080.....	318

8.2.11. Примеры работы УПИМК в интерфейсах систем 8085АН, 8086, 8088, 8048	320
8.2.12. Каналы ввода-вывода	323
8.3. Режимы работ.....	327
8.3.1. Режим пошагового выполнения команд.....	327
8.3.2. Режим внешнего доступа	328
8.3.3. Проверка внутренней памяти программ.....	328
8.3.3. Режим микропотребления	330
8.4. Система команд.....	330
8.5. Описание машинных команд	341
8.6. Электрические параметры	374
8.7. Применение УПИМК	376
8.8. Аналоги фирмы Intel	379
Приложение 1. Использование микросхем КР1810ВК56, КМ1821РУ55, КМ1821РЕ55, К573РФ10 совместно с микроЭВМ семейств МК48 и МК51.....	386
Приложение 2. Производители и поставщики микросхем однокристальных микроЭВМ семейств МК48 и МК51.....	389
Приложение 3. ОМЭВМ семейства MCS®-48 и MCS®-51 выпускаемые фирмой Intel	390
Приложение 4. Система команд однокристальных микроЭВМ семейства UPI-42	392
Приложение 5. Система команд однокристальных микроЭВМ семейства МК48	394
Приложение 6. Система команд однокристальных микроЭВМ семейства МК51	396

ГЛАВА 1

ОДНОКРИСТАЛЬНЫЕ МИКРОЭВМ СЕМЕЙСТВА МК48

1.1. Общие сведения об однокристальных микроЭВМ семейства МК48

Семейство МК48 включает ряд моделей ОМЭВМ, функциональный состав и технические характеристики которых отражают как различие в идеологическом подходе к применению ОМЭВМ, так и прогресс технологии СБИС. Все модели, входящие в семейство МК48, являются полностью совместимыми по системе команд, назначению и разводке выводов, совокупности основных функциональных устройств из базового набора семейства.

Первое поколение отечественных ОМЭВМ семейства МК48 — БИС КМ1816ВЕ48 и КР1816ВЕ35 являются функционально-конструктивными аналогами БИС соответственно 8748 и 8035 фирмы Intel, США, выполнены по n-канальной МОП-технологии, что обусловило следующие ограничения: уровень интеграции до 18 тыс. транзисторов на кристалле, частота следования тактовых сигналов — 6,0 МГц, объем внутренней памяти ОЗУ — 64 байта, ППЗУ — 1 Кбайт и минимальное время цикла — 2,5 мкс.

Второе поколение — БИС КР1816ВЕ49, КР1816ВЕ39 (аналоги БИС 8049 и 8039 фирмы Intel) выполнено по n-канальной МОП-технологии с пропорциональным масштабированием, что позволило повысить уровень интеграции до 36 тыс. транзисторов на кристалле, частоту следования тактовых сигналов до 11,0 МГц, увеличить объем ОЗУ до 128 байт, ПЗУ — до 2 Кбайт и снизить минимальное время цикла до 1,36 мкс.

Третье поколение семейства МК48 — БИС ОМЭВМ серии К1830: КР1830ВЕ48, КР1830ВЕ35 (аналоги БИС 80С48, 80С35 фирмы Intel) выполнено по КМОП-технологии, что позволило на порядок уменьшить ток потребления по сравнению с БИС КМ1816ВЕ48, КР1816ВЕ35 при сохранении остальных параметров.

ОМЭВМ КМ1816ВЕ48, КР1816ВЕ35, КР1830ВЕ48 и КР1830ВЕ35 полностью идентичны в части структурной реализации. При этом в БИС КМ1816ВЕ48 программная память размещается во внутреннем ППЗУ с ультрафиолетовым стиранием, а в БИС КР1830ВЕ48 — во внутреннем ПЗУ масочного типа. Таким образом, оперативность программирования ППЗУ позволяет использовать ОМЭВМ КМ1816ВЕ48 при создании контроллеров единичных экземпляров или мелкосерийных изделий. Потребители БИС КР1830ВЕ48 лишены такой возможности, так как программирование ПЗУ осуществляется в процессе изготовления БИС по данным "прошивки" заказа потребителя.

В микросхемах КР1816ВЕ35 и КР1830ВЕ35 в отличие от БИС КМ1816ВЕ48, КР1830ВЕ48 память программ реализуется только за счет подключения внешней памяти любого типа (ОЗУ, ППЗУ, ПЗУ) общим объемом до 4 Кбайт. Эта особенность позволяет использовать их в качестве отладочного варианта, когда память программ реализуется в ОЗУ, что позволяет легко модифицировать отлаживаемые программы.

ОМЭВМ КР1816ВЕ49 и КР1816ВЕ39 имеют одну и ту же структуру, одинаковые схемотехнические решения и технические характеристики, за исключением памяти программ: ОМЭВМ КР1816ВЕ49 имеет внутреннюю память программы объемом 2 Кбайт, выполненную в виде масочного ПЗУ, а ОМЭВМ КР1816ВЕ39 может использоваться только с внешней ЗУ программ. Реализация программной памяти КР1816ВЕ49 в виде ПЗУ обуславливает целесообразность применения этих ОМЭВМ только для изделий средне- и крупносерийного производства, что обеспечивает в этом случае низкую стоимость ОМЭВМ. В качестве отладочной модели, а также при разработке единичных экземпляров изделий целесообразно использовать ОМЭВМ КР1816ВЕ39 с внешней программной памятью.

В общем виде основные отличительные особенности ОМЭВМ семейства МК48 представлены в табл. 1.1.

Таблица 1.1

Микросхемы	Аналог	Объем внутренней памяти программ, байт	Тип памяти программ, байт	Объем памяти данных, байт	Максимальная частота сле- дования тактовых сигналов, МГц	Ток потреб- ления, мА
KP1816BE35	8035	Нет	внешн.	64	6,0	135,0
KP1816BE48	8748	1K	УФПЗУ	64	6,0	135,0
KP1816BE39	8039	Нет	внешн.	128	11,0	110,0
KP1816BE49	8049	2K	ПЗУ	128	11,0	110,0
KP1830BE35	80C35	Нет	внешн.	64	6,0	8,0
KP1830BE48	80C48	1K	ПЗУ	64	6,0	8,0

В ОМЭВМ предусмотрена возможность расширения памяти программ до 4 Кбайт, памяти данных до 384 байт и увеличения числа линий ввода-вывода за счет подключения внешних кристаллов памяти программ, ОЗУ и БИС интерфейсов.

С учетом идентичности структурных решений, системы команд, назначения и расположения выводов всех микросхем семейства МК48, последующие разделы главы 1 посвящены ОМЭВМ КМ1816BE48. Отличительные особенности каждой микросхемы при необходимости оговорены по ходу изложения или отдельными разделами.

Ориентация ОМЭВМ на преимущественное применение в системах управления отразилась на структуре и функциональных характеристиках отдельных устройств. Так, например, процессорное устройство по эффективности вычислительных операций и способов адресации уступает 8-разрядному микропроцессору KP580BM80A, однако, оно реализует ряд логических операций над отдельными разрядами аккумулятора и портов ввода-вывода, что повышает его эффективность при выполнении алгоритмов управления.

Наличие трех 8-разрядных портов ввода-вывода P0, P1, P2 в ОМЭВМ решает проблему расширения как памяти программ, так и памяти данных, а также обеспечивает возможность обмена информацией с периферией.

С точки зрения архитектурных особенностей регистры общего назначения (РОН), косвенная адресация, программные прерывания и асинхронный ввод-вывод, страничная адресация памяти не представляют принципиально новых архитектурных решений. Конструктивной особенностью ОМЭВМ является принцип сведения к минимуму количества разбросанных по кристаллу регистров, реализация которого повлекла за собой образование 8-уровневого стека и регистров общего назначения на общем адресном пространстве ОЗУ как сверхоперативной памяти.

Программист, работающий с ОМЭВМ семейства МК48, располагает двумя программно переключаемыми банками регистров общего назначения, каждый из которых содержит по 8 регистров и расположен на общем адресном пространстве ОЗУ. Нулевой банк РОН0 занимает адреса 0...7, первый банк РОН1 занимает адреса 24...31. Регистры активного в данный момент банка РОН прямо адресуются большим количеством инструкций.

Кроме того, все ячейки ОЗУ, включая стек, адресуются косвенно с использованием нулевого или первого регистров банка РОН.

Ячейки ОЗУ с адресами 8...23 могут использоваться в качестве стека с глубиной 8, каждому уровню которого соответствуют два байта. При обращении к подпрограмме один байт представляет младшие разряды адреса возврата, второй — содержит четыре старших разряда адреса возврата, а остальные полбайта сохраняют четыре старших разряда слова состояния программы.

При объеме ОЗУ 64×8 разрядов, пользователь, к примеру КР1816ВЕ48, имеет два 8-разрядных регистровых банка с прямой адресацией регистров и 32×8 -разрядную резидентную память данных с косвенной адресацией, если используется вся глубина стека, или 56×8 -разрядную память данных с косвенной адресацией, если используется только один регистровый банк и не используется стек.

Доступ к внешней памяти данных осуществляется с помощью команд MOVX A, @R и MOVX @R, A , которые передают данные между аккумулятором и внешней памятью данных с обращением по адресу, содержащемуся в регистрах-указателях R0 и R1 ОЗУ. Таким образом может быть адресовано 256 ячеек в дополнение к резидентной памяти ОЗУ. Для работы с внешней памятью данных используется порт P0 ОМЭВМ.

Мощность ввода-вывода ОМЭВМ семейства МК48 может быть оценена следующим образом:

число входов — $3 \times 8 + 3 = 27$;

число двунаправленных выводов — $3 \times 8 = 24$;

число выводов с возможностью фиксации данных на них — $3 \times 8 = 24$;

число выводов, управляющих вводом-выводом — 4.

Порты P1, P2 обладают сходными возможностями в части фиксации. Данные статически присутствуют на выводах порта и могут быть изменены только с новой выдачей.

При работе с внешней памятью программ функциональная нагрузка на порт P2 увеличивается, так как четыре младших разряда порта используются для расширения адреса обращения к памяти программ. Аналогично загруженность порта возрастает при работе с дополнительными портами ввода-вывода, которые могут быть подключены к ОМЭВМ посредством четырех младших разрядов порта P2.

Порт P1 и старшие разряды порта P2 работают как статический порт независимо от режима использования ОМЭВМ.

При работе с внешней памятью программ и устройствами ввода-вывода 8-разрядная шина данных (порт P0) представляет собой двунаправленный порт с синхронным стробированием. Порт P0 может действовать в трех различных режимах: как статический порт ввода-вывода в автономном режиме; как двунаправленный порт-расширитель адреса/данных в любом режиме и как выход младших разрядов адреса/инструкции/констант при использовании внешней памяти программ.

При помощи команд ввода-вывода адресуются все порты ОМЭВМ, а также четыре дополнительных порта ввода-вывода, которые можно реализовать на дополнительных микросхемах. Предусмотрена возможность выполнения логических операций И и ИЛИ над содержимым дополнительных портов и младшими четырьмя разрядами аккумулятора. Кроме трех 8-разрядных портов ввода-вывода на кристалле имеется три вывода специального назначения, которые могут программно проверяться и один из которых можно использовать для организации прерываний.

Входы тестирования и прерывания позволяют программам разветвляться без необходимости загрузки данных через порт в аккумулятор.

При возникновении прерывания ОМЭВМ переходит на выполнение программы обработки прерывания, начальный адрес которой фиксирован в памяти программ как в автономном режиме, так и в режиме с внешней памятью. Адрес возврата и слово состояния программы (частично) заносятся во внутренний стек и, следовательно, восстанавливаются после окончания выполнения подпрограммы. Команда RETR разрешает дальнейшие прерывания. Если прерывание запрещено, линия прерывания опрашивается как входная линия для обнаружения запроса. Процессор не вырабатывает специального сигнала "Подтверждение прерывания", однако, для этого можно использовать команды установки разрядов на линиях ввода-вывода, то есть выдачу такого сигнала можно организовать программно.

Характерной особенностью архитектуры ОМЭВМ является наличие на кристалле таймера, предназначенного для прерывания по его переполнению. Таймер можно использовать также в качестве счетчика внешних событий по одной из линий

ввода с переходом к подпрограмме обслуживания прерывания по заданному числу событий.

В дополнение к такому значительному количеству линий ввода-вывода ОМЭВМ серии К1816 обладает мощным набором команд, ориентированных на внешний обмен.

Внутренняя синхронизация, с делением частоты на три по отношению к внешней, вырабатывает тактировку внутренних состояний. Сигнал внутренней тактировки с делением частоты на пять поступает на счетчик машинных циклов.

1.2. Структурная организация ОМЭВМ

Микросхема КМ1816ВЕ48 конструктивно выполнена в 40-выводном металлокерамическом корпусе 2123.40-6 с прозрачной для УФ-излучения крышкой. Остальные БИС семейства МК48 конструктивно выполнены в 40-выводных пластмассовых корпусах 2123.40-2.

Все выводы электрически совместимы с элементами TTL.

Условное графическое обозначение микросхем показано на рис. 1.1, чертеж корпусов — на рис. 1.1а, 1.1б, назначение выводов — в табл. 1.2.

Структурная схема ОМЭВМ КМ1816ВЕ48 изображена на рис. 1.2. Отличия в структурах других микросхем семейства МК48 обуславливаются только либо отсутствием внутренней памяти программ (микросхемы КР1816ВЕ35, КР1816ВЕ39, КР1830ВЕ35), либо типом этой памяти (ПЗУ для КР1830ВЕ48), либо ее типом и емкостью (ПЗУ 2 Кбайт для КР1816ВЕ49) и достаточно очевидны.

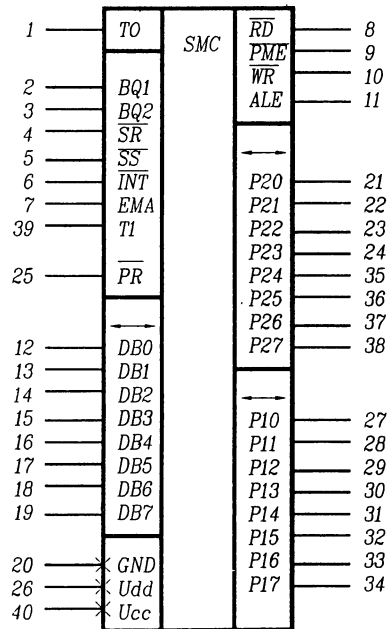


Рис. 1.1. Условное графическое обозначение

1.2.1. Процессор

Основу процессора ОМЭВМ составляет 8-разрядное арифметико-логическое устройство (АЛУ), позволяющее выполнять арифметические, логические операции и операции сдвига над данными, представленными в двоичном коде, а также обрабатывать данные, представленные в двоично-десятичном коде.

В состав АЛУ входят собственно арифметико-логическое устройство, аккумулятор, регистр аккумулятора, регистр временного хранения, схема десятичной коррекции аккумулятора.

Аккумулятор (А) представляет собой 8-разрядный регистр, предназначенный для записи и хранения данных, подаваемых с внутренней шины. Результат выполнения операции АЛУ всегда заносится через шину в аккумулятор. Выход аккумулятора связан со входом регистра аккумулятора (РА) — 8-разрядного регистра, предназначенного для записи и хранения одного из операндов, над которыми производятся операции в АЛУ. Сигналы с выхода РА непосредственно подаются на вход первого операнда АЛУ.

Регистр временного хранения (РВ) представляет собой 8-разрядный регистр и предназначен для записи и хранения второго операнда при выполнении операций в АЛУ. Вход РВ связан шиной данных с выходом ПЗУ констант. Сигналы с выхода РВ непосредственно подаются на вход регистра второго операнда АЛУ.

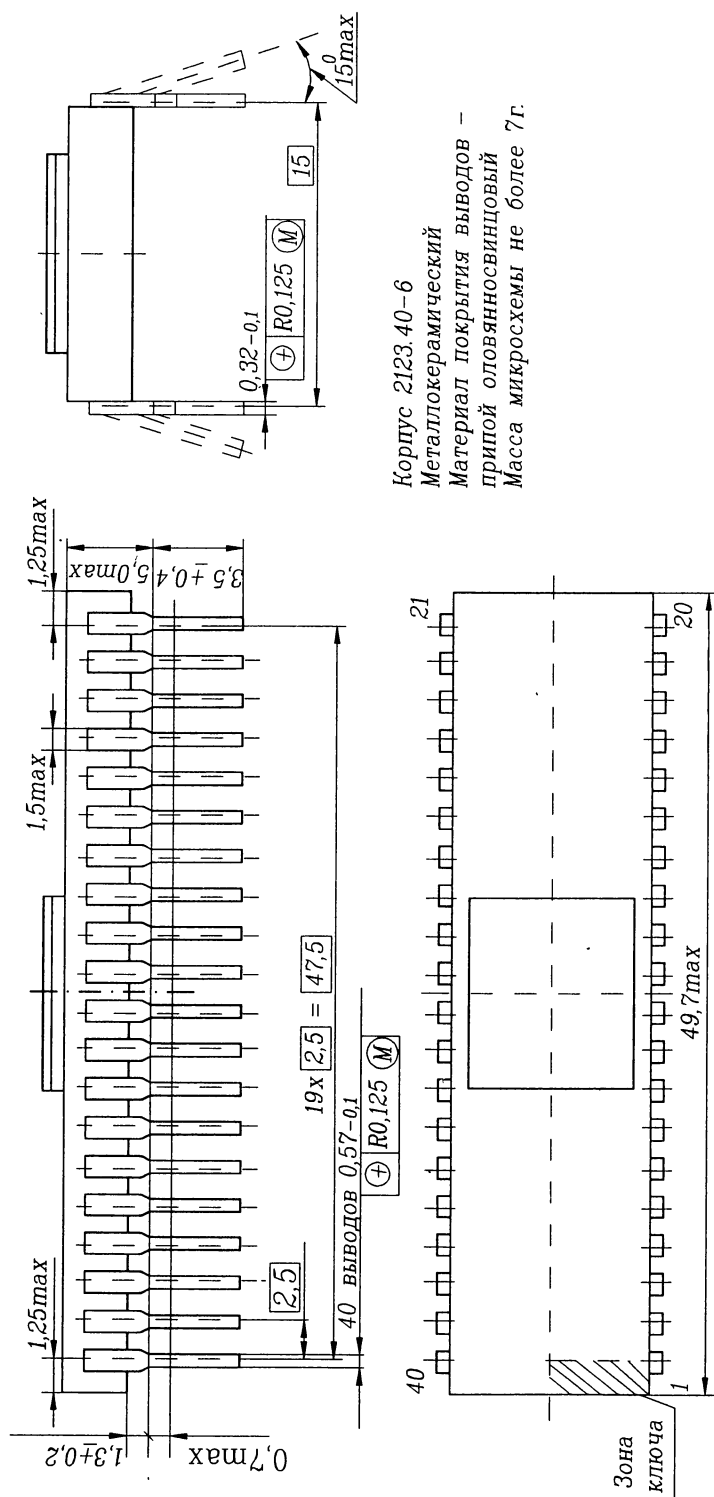


Рис. 1.1а. Конструктивное исполнение микросхемы КМ1816ВЕ48

Таблица 1.2

Номер вывода	Обозначение (Intel)	Обозначение (МК48)	Назначение
1	T0	T0	Тестируемый вход T0 используется при выполнении команд условного перехода JT0 и JNT0; после выполнения команды ENT0 CLK используется как выход тактовых сигналов; в микросхеме КМ1816ВЕ48 – как вывод при программировании и проверке УППЗУ
2	XTAL1	BQ1	Выводы для подключения кварцевого резонатора или LC – цепи
3	XTAL2	BQ2	
4	RESET	SR	Сброс
5	SS	SS	Пошаговое выполнение команд
6	INT	INT	Прерывание
7	EA	EMA	Переключение на режим работы с внешней памятью
8	RD	RD	Разрешение: чтения внешней памяти данных
9	PSEN	PME	чтения внешней памяти программ
10	WR	WR	записи во внешнюю память данных
11	ALE	ALE	фиксации адреса
12...19	D0...D7 (BUS)	DB0...DB7	Шина данных (порт P0)
20	VSS	GND	Общий вывод
21...24	P20...P27	P20...P27	Порт P2
35...38			
25	PROG	PR	Выход строба для расширителя ввода-вывода. В микросхеме КМ1816ВЕ48 используется при программировании УППЗУ
26	VDD	Udd	Напряжение питания РПЗУ для КМ1816ВЕ48 Напряжение резервного питания ОЗУ для КР1816ВЕ49/ ВЕ39 Вход установки режима микропотребления для КР1830ВЕ48/ ВЕ35
27...34	P10...P17	P10...P17	Порт P1
39	T1	T1	Тестируемый вход 1 используется при выполнении команд условного перехода JT1 и JNT1 и как вход счетчика внешних событий после выполнения команд STRT CNT
40	VCC	Ucc	Напряжение питания

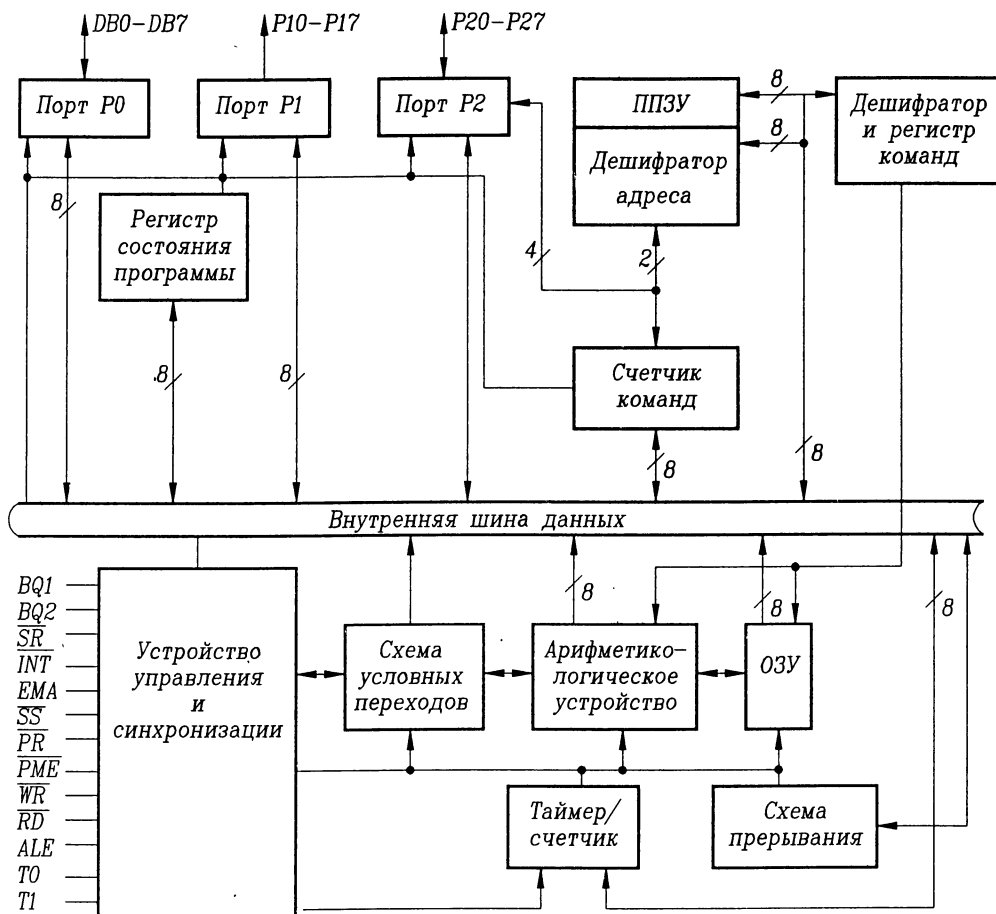


Рис. 1.2. Структурная схема ОМЭВМ

Схема десятичной коррекции (СДК) предназначена для обработки данных, представленных в двоично-десятичном коде. В состав схемы СДК входят узлы анализа содержимого полубайтов (4 разряда старших или младших) аккумулятора, триггер основного и дополнительного переносов и схема формирования корректирующей поправки. Схема десятичной коррекции содержит также ПЗУ констант 00, 06, 60, 66 (шестнадцатеричный код), которые в зависимости от значения младшего и старшего полубайтов и основного и дополнительного переносов подаются через регистр РВ на вход АЛУ.

В состав процессора входят также счетчик команд, дешифратор и регистр команд, регистр состояния программы и схема условных переходов.

Счетчик команд (СК) предназначен для формирования текущего адреса местонахождения команды в памяти программ. Счетчик команд содержит 12 разрядов. Содержимое СК увеличивается после выбора каждого байта команды на единицу. Содержимое СК может изменяться скачкообразно при выполнении команд передачи управления, CALL, RET, RETR и при реализации прерываний. Старший разряд СК изменяется только программно (команды SEL MB0, SEL MB1). Счетчик разбит на две части: счетчик младших разрядов (биты 0...7) и счетчик старших разрядов (биты 8...11). Биты 0...7 передаются при адресации через внутреннюю шину на вход дешифлятора адреса ППЗУ, а биты 8...9 — непосредственно со счетчика на дешифратор адреса. При использовании внешней программной памяти

биты 0...7 СК поступают через порт P0 (выводы DB0...DB7), а биты 8...11 — через порт P2 (выводы P20...P23).

Дешифратор команд представляет собой программируемую логическую матрицу, на вход которой поступает код команды с регистра команд, в котором осуществляется запись и хранение кода команды. С выхода дешифратора команд снимаются управляющие сигналы, осуществляющие выполнение этой команды.

Регистр состояния программы (PSW) предназначен для хранения данных о состоянии микроЭВМ. Назначение разрядов PSW (формат PSW показан в табл. 1.3) следующее: разряды 0...2 — разряды указателя стека (S0...S2); разряд 3 не используется (при чтении всегда "1"); разряд 4 — разряд, указывающий используемый банк рабочих регистров общего назначения (BS); разряд 5 — флаг пользователя (F0), используется по команде условного перехода; разряд 6 — разряд дополнительного переноса (AC), используется по команде десятичной коррекции; разряд 7 — перенос (C), указывающий на переполнение аккумулятора после предыдущей операции.

Таблица 1.3 Формат PSW

7	6	5	4	3	2	1	0
C	AC	F0	BS	1	S2	S1	S0

Сохранение
в стеке
Указатель
стека (SP)

Регистр PSW может программно проверяться, модифицироваться весь и поразрядно. При прерываниях по входу \overline{INT} и по флагу таймера-счетчика содержимое четырех разрядов (4...7) автоматически заносится в стек, а при возврате из программы обработки прерывания содержимое этих разрядов восстанавливается. После загрузки в стек содержимое указателя стека инкрементируется, а перед извлечением из стека декрементируется. При переполнении стека указатель стека переходит из состояния 7 в состояние 0.

Схема условных переходов предназначена для формирования сигналов управления ветвлением программы при выполнении команд условных переходов. Условия перехода определяются:

- содержимым аккумулятора (возможна проверка на "0" или не "0");
- содержимым бита аккумулятора (проверка на "1");
- состоянием флага переноса C (проверка на "0" и на "1");
- состояниями флагов пользователя F0 и F1 (проверка на "1"). Операции установки/сброса флагов F0 и F1 выполняются программно по командам CLR F0, CLR F1, CPL F0, CPL F1. Сигнал системного сброса \overline{SR} сбрасывает флаги F0 и F1;
- триггером таймера-счетчика TF (проверка на "1");
- сигналами на входах ОМЭВМ T0 и T1 (проверка на "0" и на "1");
- сигналом на входе ОМЭВМ \overline{INT} (проверка на "0").

1.2.2. Память программ

Память программ предназначена для хранения и считывания команд, которые поступают в процессор и управляют процессом обработки информации. Общий объем адресуемой памяти программ ОМЭВМ семейства МК48 составляет 4 Кбайт, при этом, в отличие от микросхем КР1816ВЕ35, КР1816ВЕ39 и КР1830ВЕ35, где весь объем памяти сосредоточен во внешних ЗУ, в микросхемах КМ1816ВЕ48, КР1816ВЕ49 и КР1830ВЕ48 память разделена на две части: резидентная программная память объемом 1024 байт (КМ1816ВЕ48, КР1830ВЕ48) и 2048 байт (КР1816ВЕ49) и внешняя программная память, составляющая в сумме с резидентной памятью 4 Кбайт. Отличия ОМЭВМ внутри семейства МК48, связанные с объемом памяти, приведены в табл. 1.1.

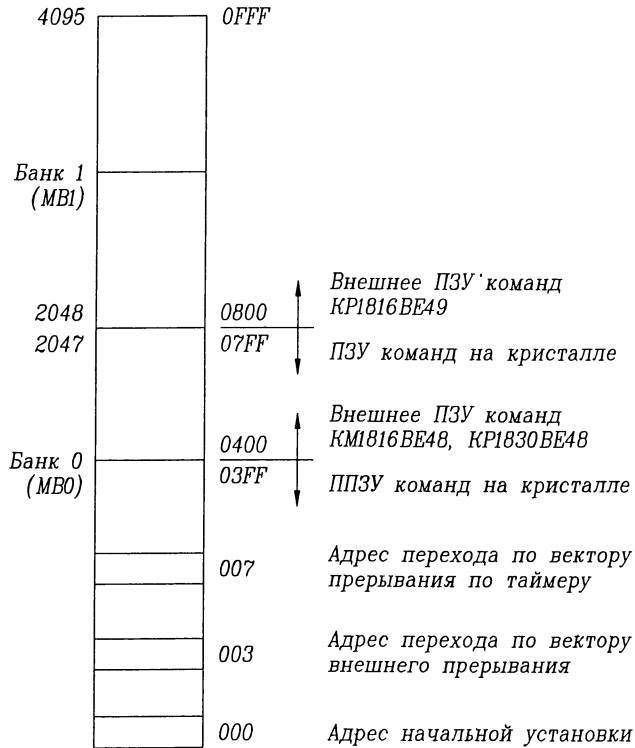


Рис. 1.3 Карта распределения памяти программ

Если адрес выборки команды выходит за пределы резидентной памяти, то автоматически инициализируется внешняя память.

Все выборки из внутренней памяти не сопровождаются никакими внешними сигналами, генерируемыми ОМЭВМ, кроме сигнала ALE, который вырабатывается независимо от режима использования ОМЭВМ и является идентификатором машинного цикла. При обращении к внешней памяти программ содержимое 12-разрядного счетчика команд выводится на 8-разрядную шину данных (порт P0) и четыре младших разряда порта P2. Сигнал ALE задним фронтом фиксирует выставленный адрес. Сигнал PME стробирует выборку байта из внешней памяти программ. Байт из внешней памяти программ принимается в ОМЭВМ через шину данных (порт P0).

Память, расположенная на кристалле микросхемы, занимает адресное пространство от 0000H до 03FFH (КМ1816ВЕ48, КР1830ВЕ48) или до 07FFH (КР1816ВЕ49). Карта распределения памяти программ представлена на рис. 1.3. Все поле адресов от 0000H до 0FFFFH разбито на два банка — банк 0 с адресами от 0000H до 07FFH и банк 1 с адресами от 0800H до 0FFFFH. Счетчик команд ОМЭВМ содержит 12 бит, но инкрементируются в процессе счета только младшие 11 бит. При последовательном счете счетчик команд из состояния 7FFH перейдет в состояние 000H. Переключение программной памяти с одного банка на другой осуществляется по командам SEL MB0 и SEL MB1 и непосредственно связано со старшим разрядом программного счетчика. Устанавливается этот разряд по первой команде JMP или CALL, следующей за командой SEL MB0 или SEL MB1.

Память программ делится не только на два банка емкостью 2 Кбайт, но и на страницы емкостью по 256 байт в каждой. В командах условного перехода задается 8-битный адрес передачи управления в пределах текущей страницы.

Для доступа к памяти программ как к таблицам данных служат команды обращения к текущей странице памяти программ MOVР и к третьей странице MOVРЗ, которые позволяют считывать байт из программной памяти в аккумулятор.

В памяти программ имеется три ячейки специального назначения: адрес 0 — адрес, по которому выполняется первая выборка инструкции по сигналу сброса \overline{SR} ; адрес 03 — начальный адрес подпрограммы, вызываемой по сигналу прерывания ОМЭВМ при условии, что прерывание разрешено; адрес 07 — начальный адрес подпрограммы, вызываемой по переполнению таймера-счетчика при условии, что прерывание разрешено.

1.2.3. Память данных

Память данных предназначена для записи, хранения и считывания данных, получаемых в процессе обработки информации. Память данных, состоящая из 64 ячеек ОЗУ в КР1816ВЕ35, КМ1816ВЕ48, КР1830ВЕ35, КР1830ВЕ48 и 128 ячеек в КР1816ВЕ39, КР1816ВЕ49, разбита на два банка регистров общего назначения (РОН) с адресами от 00Н до 07Н — банк РОН 0 и с адресами от 18Н до 1FH — банк РОН 1. Карта распределения памяти данных изображена на рис. 1.4 (в скобках указаны данные для ИС КР1816ВЕ39 и КР1816ВЕ49). Переключение банков осуществляется программным путем с помощью команд SEL RB0, SEL RB1.

Ячейки ОЗУ с адресами от 20Н до 3FH используются только как ОЗУ данных. Восьмиуровневый 16-разрядный стек с адресами от 08Н до 17Н адресуется указателем стека из PSW. Организация стека показана на рис. 1.4а (адрес ОЗУ приведен в десятичном коде). Кроме того, с использованием косвенной адресации ячейки стека могут адресоваться как ОЗУ данных. ОМЭВМ семейства МК48 не имеют специальных команд загрузки байта в стек или его извлечения из стека.



Рис. 1.4. Карта распределения памяти данных

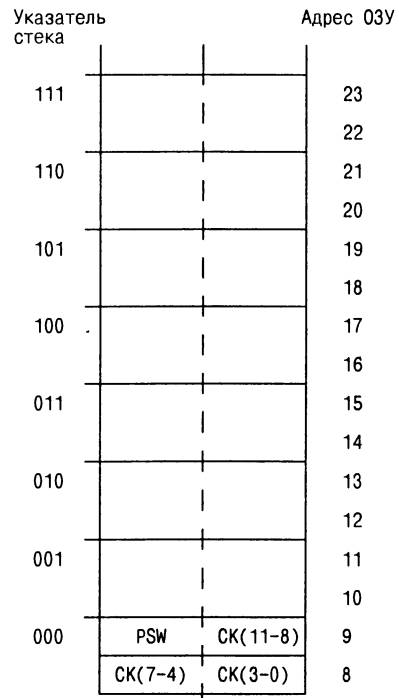


Рис. 1.4а. Организация стека

Для записи и выборки данных из ОЗУ применяются два вида адресации: прямая и косвенная (регистровая). Независимо от типа адресации три младших разряда кода команды указывают один из восьми регистров РОН R0...R7 с учетом принадлежности к ранее выбранному банку регистров. При использовании команд с

прямой адресацией указанный регистр является источником или приемником данных, а при использовании команд с косвенной адресацией указанный регистр содержит адрес данных (в качестве регистров косвенного адреса используются только R0 и R1). С помощью косвенной адресации можно адресоваться к любой ячейке памяти данных. Программист по своему усмотрению может заносить данные для хранения в любые ячейки-регистры банков P0H, стек, а также имеет доступ к любой из ячеек ОЗУ посредством косвенной адресации.

В ОМЭВМ предусмотрена возможность расширения внутренней памяти данных до 384 байт для микросхем КР1816ВЕ49, КР1816ВЕ39 и до 320 байт для остальных микроЭВМ семейства МК48 путем подключения микросхем ОЗУ. Обращение к внешней памяти данных осуществляется с помощью команд MOVX @R, A; MOVX A, @R. Обмен информацией с внешним ОЗУ строится сигналами WR и RD и производится через шину данных (порт P0) ОМЭВМ.

1.2.4. Каналы ввода-вывода

Каналы ввода-вывода служат для организации обмена информацией ОМЭВМ с внешними устройствами. В ОМЭВМ имеется 27 линий ввода-вывода, 24 из которых объединены в три 8-разрядных порта P0, P1, P2.

Порты P1, P2 в режиме вывода обладают возможностью фиксации данных в так называемых триггерах-защелках. Эти данные статически присутствуют на выводах порта и могут быть изменены только новой выдачей по команде OUTL. Каждая выдача сопровождается занесением данных в защелку порта.

В состоянии ввода входная информация не изменяет состояния защелок. При использовании портов P1 и P2 в качестве входов необходимо до подачи входной информации линии портов установить в состояние высокого уровня, выдав на порт байт единиц. В это состояние выходы портов устанавливаются также после подачи сигнала \overline{SR} . Возможна произвольная смешанная настройка линий портов P1 и P2, когда одни линии порта работают на ввод, а другие — на вывод. Для настройки линии на режим ввода необходимо в триггер-защелку этой линии записать "1". Вводимые данные должны присутствовать на линиях порта до тех пор, пока не будут программно прочитаны. Электрическая схема одной из линий портов P1 и P2 показана на рис. 1.5а.

Устройство B1 предназначено для передачи содержимого защелки (D-триггер) на внутреннюю шину данных для дальнейшей модификации по командам ORL PR, #data или ANL PR, #data.

Устройство B2 обеспечивает передачу входной информации порта на внутреннюю шину данных при выполнении команд, осуществляющих ввод информации с выводов порта.

Устройство И обеспечивает включение транзистора VT1 на время $t_{cy}/6$ при изменении содержимого защелки (D-триггер) с "0" на "1" для формирования фронта нарастания сигнала на выводах порта. После выключения транзистора VT1 уровень логической единицы поддерживается на выходе порта с помощью резистора R1. Сопротивление открытого транзистора VT1 составляет приблизительно 5 кОм, сопротивление резистора R1 — около 50 кОм. Время t_{cy} определяется по следующей формуле:

$$t_{cy} = 15/f_{BQ1},$$

где f_{BQ1} — частота тактовых сигналов ОМЭВМ, МГц.

Для расширения адреса обращения к внешней памяти программ, а также для увеличения числа линий ввода-вывода дополнительно используются четыре младших разряда порта P2. Стробирование данных при работе с дополнительным портом осуществляется сигналом \overline{PR} . Входные данные для резидентного порта должны быть поданы в состояние машинного цикла, когда считывание возможно, то есть между соседними сигналами ALE при выполнении команды ввода IN.

Кроме операций ввода-вывода информации, предусмотрена возможность выполнения логических операций И, ИЛИ непосредственно на портах P1 и P2 с помощью команд ANL PR, #data; ORL PR, #data.

Порт $P\emptyset$ — это 8-разрядный двунаправленный порт с тремя состояниями, который может использоваться в качестве статического порта ввода/вывода или двунаправленного порта адреса/данных с тремя состояниями при работе с внешней памятью.

Если порт $P\emptyset$ используется как статический порт, то вывод через него выполняется по команде $OUTL\ BUS,A$, а ввод — по команде INS . Вывод сопровождается сигналом \overline{WR} , а ввод — сигналом \overline{RD} . При этом выводимые данные фиксируются в триггерах-защелках и статически выставляются на выводах порта.

В отличие от портов $P1$ и $P2$, порт $P\emptyset$ допускает только байтовый обмен, когда по всем линиям порта производится либо ввод, либо вывод.

При работе с внешней памятью программ через порт $P\emptyset$ в режиме мультиплексирования сначала выдается младший байт адреса команды, а затем синхронно с сигналом PME вводится из памяти байт команды.

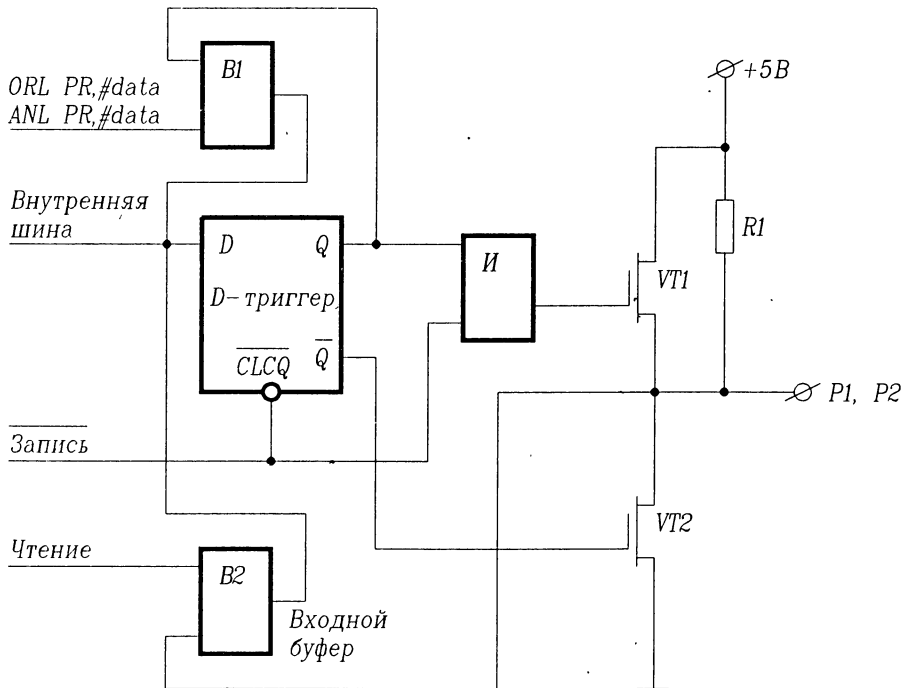


Рис. 1.5а. Электрическая схема одной из линий портов $P1$ и $P2$

При работе с внешней памятью данных через порт $P\emptyset$ в режиме мультиплексирования выдается адрес данных, а затем выполняется обмен байтом данных: ввод синхронно с сигналом \overline{RD} либо вывод синхронно с сигналом \overline{WR} . Для работы с внешней памятью данных служат команды $MOVX$.

В режиме работы с внешней памятью, если не используется команда $OUTL\ BUS,A$, порт $P\emptyset$ при отсутствии передач находится в высокоимпедансном состоянии.

Команды $MOVX$ и $OUTL\ BUS,A$ могут применяться поочередно, но при этом статическая информация, выставленная на порте $P\emptyset$ по команде $OUTL$, будет разрушена последующим выполнением команды $MOVX$, а порт $P\emptyset$ перейдет в высокоимпедансное состояние.

Если порт $P\emptyset$ активизируется командами $MOVX$, то в отсутствии передач порт по своим выходам находится в высокоимпедансном состоянии.

Выполнение команд $MOVX$, $OUTL\ BUS,A$, INS характеризуется идентичными временными параметрами на внешних выводах ОМЭВМ: длительности сигналов \overline{RD} ,

\overline{WR} , временные задержки и т. п. В дальнейшем описании рассматривается только работа команд $MOVX$ при обменах с внешней памятью данных.

На рис. 1.56 показана электрическая схема одной из линий порта $P0$.

Устройство B1 предназначено для передачи содержимого защелки (D-триггер) на внутреннюю шину данных для дальнейшей модификации по командам $ORL\ BUS, \#data$ или $ANL\ BUS, \#data$. По этим командам выполняются логические операции ИЛИ, И непосредственно на порте $P0$. Сигналы \overline{WR} или \overline{RD} при выполнении этих команд не формируются.

Устройство B2 обеспечивает передачу входной информации порта на внутреннюю шину данных при выполнении команд, осуществляющих ввод информации с выводов порта.

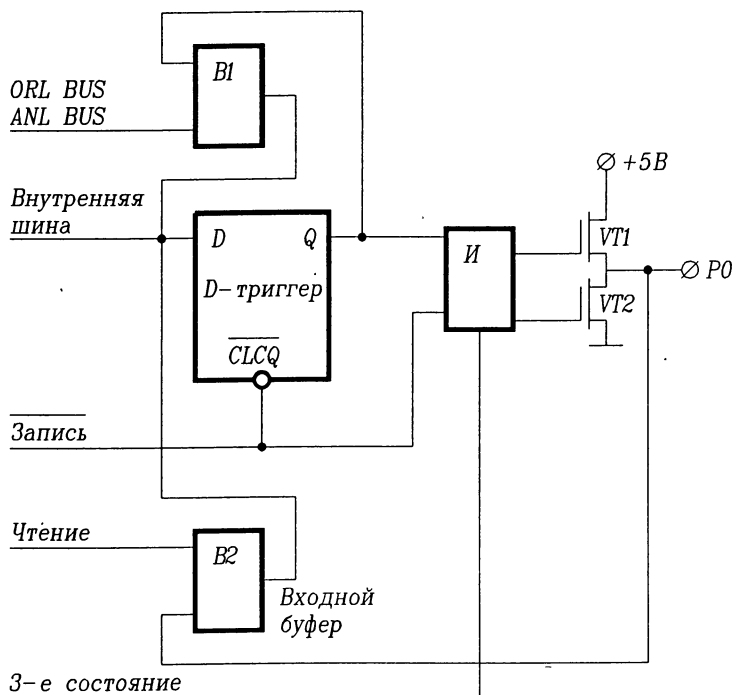


Рис. 1.56. Электрическая схема одной из линий порта $P0$

Устройство И предназначено для управления транзисторами $VT1$ и $VT2$, обеспечивающими выдачу/прием информации. Сигнал "3-е состояние" инициирует схему И на выключение транзисторов $VT1$, $VT2$, обеспечивая тем самым высокоимпедансное состояние на выводах порта $P0$, необходимое при приеме информации. Адрес и данные на $P0$ выдаются через устройство И при отсутствии сигнала "3-е состояние".

На рис. 1.56 не показан тракт выдачи адреса на порт $P0$. Адрес поступает на устройство И, минуя D-триггер. Таким образом, выдача адреса не разрушает защелкнутую по команде $OUTL\ BUS, A$ информацию.

Команда $OUTL\ BUS, A$ открывает порт $P0$, переводя его в режим выходного статического порта. Если после выполнения команды $OUTL$ необходимо подать на выводы порта $P0$ информацию с внешнего устройства с целью ввода ее в ОМЭВМ, то предварительно необходимо выполнить команду $MOVX$ для перевода порта в высокоимпедансное состояние.

Команды $ANL\ BUS, \#data$ и $ORL\ BUS, \#data$, работая с информацией, полученной с выходов защелок (рис. 1.56), не открывают порт $P0$ на выдачу, если

он находится в высокоимпедансном состоянии. Поэтому использование этих команд имеет смысл только в том случае, если порт P0 находится в режиме выходного статического порта.

Три линии ввода-вывода (T0, T1, INT) опрашиваются при выполнении команд условного перехода для реализации ветвления программы. Линия ввода-вывода T0 переключается на выход для выдачи тактовых сигналов с частотой в три раза меньшей частоты задающего генератора по команде ENT0 CLK. Линия T1 является входом счетчика внешних событий для таймера-счетчика, если счет разрешен командой STRT CNT. Линия INT используется для внешнего аппаратного прерывания, если прерывание разрешено командой ENI.

1.2.5. Таймер-счетчик

Таймер-счетчик предназначен для подсчета внешних событий и измерения временных интервалов без участия процессора ОМЭВМ.

В состав таймера-счетчика входят: делитель частоты 1:32; суммирующий счетчик; триггер флага переполнения.

8-разрядный двоичный таймер-счетчик устанавливается в начальное состояние с помощью команды MOV T, A, по которой содержимое A передается в таймер-счетчик. Счетчик останавливается (но не сбрасывается) по сигналу SR или по команде STOP TCNT и остается в этом состоянии, пока не поступит команда STRT T — запуск таймера, или команда STRT CNT — запуск счетчика событий.

Содержимое таймера-счетчика может быть прочитано командой MOV A, T, как в то время, когда таймер-счетчик остановлен, так и во время счета. При этом момент чтения не может попасть на междуразрядный перенос, вызвав тем самым неопределенность считанного отсчета, т. к. инкрементирование таймера-счетчика и чтение его содержимого в аккумулятор выполняются в разных состояниях машинного цикла ОМЭВМ (рис. 1.25).

Момент перехода счетчика из состояния максимального значения FFH в состояние 00H свидетельствует о переполнении таймера-счетчика (Т-С) и фиксируется в триггере флага Т-С и триггере переполнения Т-С, в результате вырабатывается внутренний запрос на аппаратное прерывание. Этот запрос так же, как и внешний запрос прерывания, запоминается в триггере текущего прерывания, как показано на рис. 1.6. Прерывания по переполнению таймера-счетчика считаются внутренними и могут быть разрешены или запрещены соответственно командами EN TCNTI и DIS TCNTI независимо от внешних прерываний. Если внутренние прерывания разрешены, переполнение счетчика вызовет переход к подпрограмме обслуживания прерывания от таймера-счетчика, адрес-вектор которой соответствует ячейке 07 памяти программ. Если запросы на прерывание от таймера-счетчика и внешнего источника приходят одновременно, произойдет переход к подпрограмме обслуживания внешнего прерывания, адрес-вектор которой соответствует ячейке 03 памяти программ.

Поскольку запрос на прерывание от таймера-счетчика зафиксирован, он будет обрабатываться сразу после возвращения из подпрограммы обслуживания внешних прерываний. Запрос на прерывание от таймера (триггер переполнения Т-С на рис. 1.6) сбрасывается командой DIS TCNTI. Однако, эта команда не действует на триггер флага TF таймера-счетчика, который может быть опрошен по команде условного перехода JTF

После перехода счетчика из состояния FFH в состояние 00H счетчик будет продолжать счет, если не выполнена команда STOP TCNT.

По команде STRT CNT начинается счет событий (событие — переход сигнала на внешнем выводе T1 от высокого к низкому уровню) путем инкрементирования значения текущего состояния таймера-счетчика. Подсчет событий прекращается или по команде STOP TCNT, или при подаче сигнала SR.

При выполнении команды STRT CNT разрешается передача состояния со входа T1 на вход счетчика. С этого момента каждый переход сигнала на выводе T1 от высокого к низкому уровню вызывает приращение счетчика. Минимально

допустимый период следования сигналов на входе T1 составляет $3t_{cy}$; минимально допустимая длительность сигналов низкого и высокого уровней составляет соответственно t_{cy} и $t_{cy}/5$; t_{cy} — длительность машинного цикла, рассчитываемая по формуле: $t_{cy} = 15/f_{BQ1}$, где f_{BQ1} — частота генератора тактовых сигналов ОМЭВМ, МГц. Вход T1 не имеет гистерезиса.

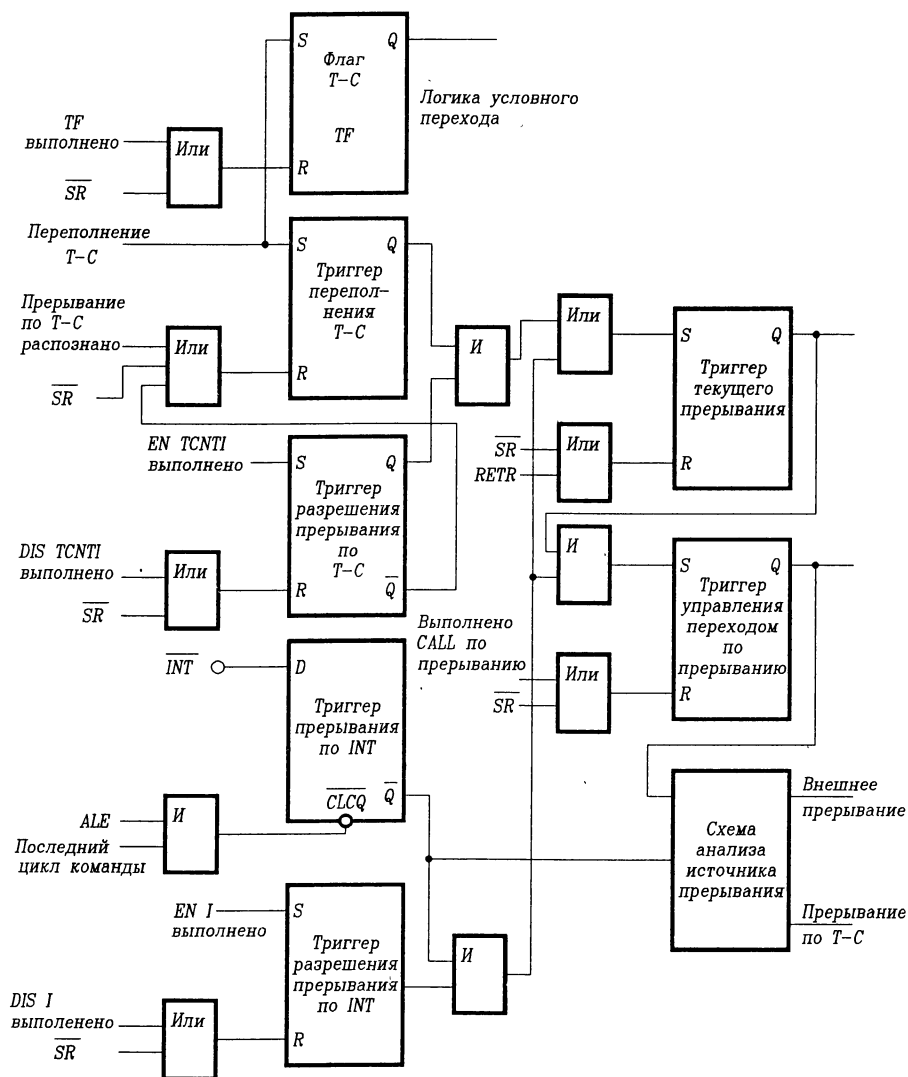


Рис. 1.6. Структурная схема организации прерываний в ОМЭВМ

В результате выполнения команды STRT T внутренние синхросигналы поступают на вход счетчика и запускают его как таймер. Синхросигналы образуются путем деления на 32 частоты основных синхронизирующих импульсов машинного цикла ALE. Делитель сбрасывается во время выполнения команды STRT T. Инкрементирование счетчика осуществляется с частотой внутренней синхронизации. Путем предварительной установки счетчика и фиксации переполнения можно получить задержки до $256 \cdot 32$ периода сигнала ALE. Задержки большей длительности могут быть получены путем накопления переполнений счетчика под управлением программы.

Для временной задержки длительностью менее 32 периодов следования сигнала ALE на вход T1 можно подавать синхроимпульсы нужной частоты и работать со счетчиком в режиме подсчета событий. Внешней синхрочастотой может служить сигнал ALE, деленный на три и более. Структурная схема включения таймера/счетчика изображена на рис. 1.7.

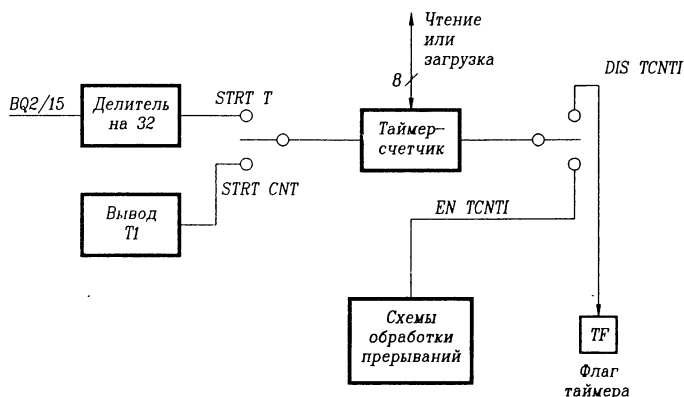


Рис. 1.7. Структурная схема включения таймера/счетчика

1.2.6. Система прерываний

Процессор ОМЭВМ имеет одноуровневую систему прерываний по адресу-вектору, которая воспринимает запросы на прерывания от внешнего источника или от внутреннего таймера-счетчика. Запрос внешнего прерывания поступает на вывод INT. Несколько источников прерываний можно объединить по схеме ИЛИ. Прерывания могут быть избирательно разрешены или запрещены с помощью команд EN и DIS. По команде EN I внешние прерывания разрешаются и воспринимаются при подаче сигнала низкого уровня на вход INT. При поступлении команды запрещения внешних прерываний DIS I или после подачи сигнала SR ОМЭВМ не реагирует на поступление низкого уровня на входе INT. Разрешение прерываний по переполнению таймера-счетчика обеспечивается выполнением команды EN TCNTI. Переполнение таймера-счетчика инициирует последовательность обработки прерывания. Запрещение прерывания по переполнению таймера-счетчика событий происходит по команде DIS TCNTI. Внешнее прерывание обладает более высоким приоритетом, то есть, если запросы на прерывание от внешнего источника и от таймера-счетчика возникают одновременно, внешнее прерывание обслуживается в первую очередь. При поступлении запроса на прерывание процессор (после завершения всех циклов текущей команды) передает управление по адресу 03 при внешнем прерывании, либо по адресу 07 при прерывании по переполнению таймера-счетчика. При этом, как и при выполнении команды CALL, обеспечивается загрузка в стек текущего значения 12-разрядного счетчика команд и четырех старших разрядов слова состояния процессора. Механизм прерывания от таймера-счетчика можно использовать как источник внешнего прерывания. Для этого необходимо загрузить в счетчик с помощью программы значение FFH и установить режим счета внешних событий. В этом случае переход сигнала на входе T1 ОМЭВМ из состояния "1" в состояние "0" вызовет прерывание с обращением по адресу ячейки 07 памяти программ.

Ячейка 03 памяти программ соответствует начальному адресу области памяти программ, где хранится программа обработки внешнего прерывания. Обычно по адресу 03 находится команда безусловного перехода на подпрограмму обслуживания. Завершает процедуру обслуживания прерывания выполнение команды RETR.

Поскольку в данной ОМЭВМ реализована одноуровневая система прерываний, обслуживание вновь поступающих прерываний откладывается до конца обработки текущего прерывания. Обработка очередного прерывания может начаться только после завершения второго машинного цикла команды RETR. Сигнал INT запроса на прерывание, которое обслуживается в данный момент, должен быть снят до исполнения команды RETR, в противном случае процессор начнет повторное обслуживание данного запроса прерывания.

Подпрограмму обработки любого прерывания нельзя прервать до команды RETR.

Если для хранения программы требуется объем памяти, превышающий 2 Кбайт, то при разработке программ обслуживания прерываний пользователю необходимо учитывать особенности управления памятью программ, присущие данной ОМЭВМ. Процессор ОМЭВМ может прямо адресовать только 2048 байт памяти программ. Для расширения объема адресуемой памяти до 4096 байт предусмотрен механизм переключения банков памяти программ с использованием команд SEL MB0, SEL MB1: (0...2047) байт — банк 0; (2048...4096) байт — банк 1. Однако, его нельзя использовать при обработке прерываний, так как при выполнении перехода или вызова подпрограммы в одиннадцать младших разрядов счетчика команд (СК) (A0...A10) производится загрузка адреса из команды, 12-й разряд (A11) загружается из триггера переключения банка памяти DBF, указывающего номер банка памяти программ.

В ОМЭВМ при обслуживании прерываний старший разряд (A11) СК, независимо от состояния триггера DBF, аппаратно устанавливается в "0" всякий раз, когда происходит переход к подпрограмме обработки прерываний. Поэтому все программы обработки прерываний и все подпрограммы, вызываемые ими, должны располагаться в пределах нулевого банка памяти программ (ячейки 0...2047). До выхода из подпрограммы обслуживания прерываний (то есть до исполнения команды RETR) никакая команда не может установить в "1" старший разряд СК. Выполнение команд SEL MB0 и SEL MB1 во время обработки прерываний не рекомендуется, так как эти команды, изменяя состояние триггера DBF, не изменяют значения старшего разряда СК до тех пор, пока происходит обработка текущего прерывания.

Структурная схема организации прерываний в ОМЭВМ показана на рис. 1.6.

1.2.7. Устройство управления и синхронизации

Устройство предназначено для выработки сигналов, обеспечивающих управление выполнением команд, и реализовано на кристалле ОМЭВМ, за исключением источника опорной частоты, в качестве которого можно использовать кварцевый резонатор, LC-цепь или внешний источник синхроимпульсов. Устройство управления и синхронизации состоит из генератора, формирователей тактовых сигналов и формирователей сигналов состояний и режимов работы.

Встроенный генератор представляет собой схему с последовательным резонансом и высоким коэффициентом усиления, обеспечивающую работу в диапазоне частот 1...11 МГц для ИС КР1816ВЕ39, КР1816ВЕ49 и 1...6 МГц для остальных ИС. Внешний вывод BQ1 является входом каскада усиления, а BQ2 — выходом. Кварцевый резонатор или LC-цепочка присоединяются между выводами BQ1 и BQ2 и обеспечивают обратную связь и фазовый сдвиг, необходимые для генерации. Для получения временных соотношений с высокой степенью точности необходимо использовать кварцевые резонаторы. Если не требуется высокое быстродействие и высокая точность задания частоты, можно использовать LC-цепочку. Внешние синхросигналы подаются на входы BQ1 и BQ2. Варианты включения показаны на рис. 1.8а, б, в (в скобках приведены данные для ИС КР1816ВЕ39, КР1816ВЕ49).

Частота генератора f_{BQ1} делится на три в счетчике состояния для получения тактовой частоты (CLK), которая определяет временные соотношения в ОМЭВМ. Сигналы CLK могут быть выведены на внешний вывод T0 по команде ENT0 CLK. Вывод сигналов на внешний вывод T0 блокируется сигналом SR.

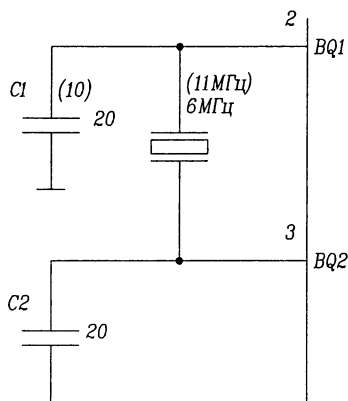


Рис. 1.8а. Схема включения кварцевого резонатора

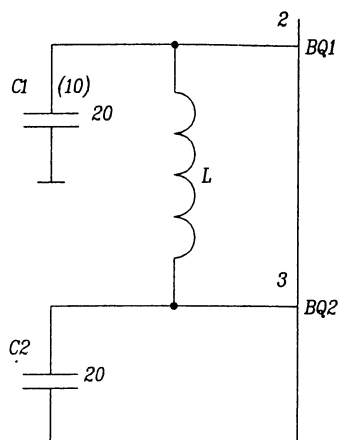


Рис. 1.8б. Схема подключения LC-цепочки

Частота CLK делится на пять в счетчике циклов для получения частоты, определяющей машинный цикл, состоящий из пяти состояний машины (S1...S5). Полученный в результате такого деления опорной частоты синхросигнал назван ALE и используется при работе ОМЭВМ с внешней памятью. Этот сигнал выдается в каждом машинном цикле на выводе ALE независимо от того, выполняется или нет в данном машинном цикле обращение к внешней памяти.

Общий сброс осуществляется благодаря наличию встроенных триггера Шмитта и соединенного с источником питания +5 В резистора, который в сочетании с внешним конденсатором емкостью 1 мкФ, подключенным к выводу \overline{SR} , как показано на рис. 1.9 (в скобках приведены данные для ИС КР1816ВЕ39, КР1816ВЕ49), обеспечивает при включении ОМЭВМ импульс системного сброса низкого уровня достаточной длительности для гарантированной установки в исходное состояние всех цепей. Если импульс сброса подается извне, то он должен быть активным не менее 50 мс после того, как источник питания установится в номинальное значение.

Подача активного импульса системного сброса производит следующие действия:

- сбрасывает счетчик команд и указатель стека;
- устанавливает порт P0 в высокоимпедансное состояние (при EMA=0), а порты P1 и P2 — в режим ввода;
- выбирает банк регистров 0 и банк памяти программ 0;
- запрещает прерывания;
- останавливает таймер-счетчик и выдачу синхросигнала на вывод T0;

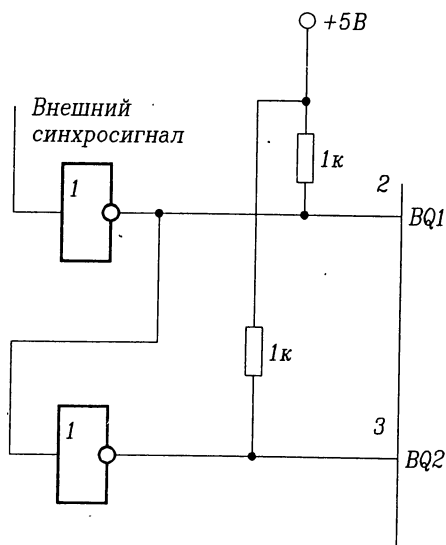


Рис. 1.8в. Схема включения при подаче внешних синхросигналов

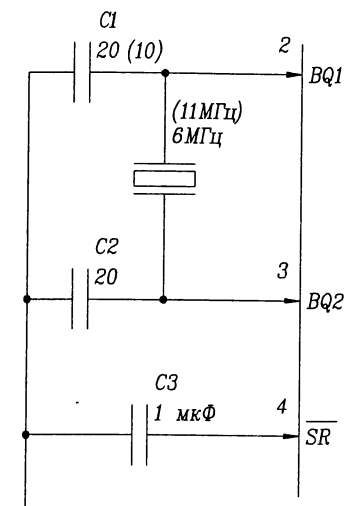


Рис. 1.9 Схема формирования сигнала системного сброса

— сбрасывает триггер флага таймера-счетчика ТФ и флаги пользователя F0 и F1.

При ЕМА=1 при действии активного сигнала системного сброса порт P0 (DB) находится в неопределенном состоянии. В этом случае нельзя считать, что порт P0 (DB) находится в высокоимпедансном состоянии.

1.3. Режимы работы

Микросхемы семейства МК48 могут работать в следующих режимах, с помощью которых осуществляется управление работой ОМЭВМ, контроль и отладка программ: проверка внутренней памяти программ (только для КМ1816ВЕ48, КР1816ВЕ49, КР1830ВЕ48), программирование внутреннего ППЗУ (только для КМ1816ВЕ48), работа с внутренней памятью программ (только для КМ1816ВЕ48, КР1816ВЕ49, КР1830ВЕ48) и внутренней памятью данных, работа с внешней памятью программ и внешней памятью данных, пошаговое выполнение команд, режим уменьшенного энергопотребления (только для КР1816ВЕ39, КР1816ВЕ49) и режим микропотребления (только для КР1830ВЕ35, КР1830ВЕ48).

Режим работы устанавливаются комбинацией входных и выходных сигналов.

Инициализация (сброс) микросхемы осуществляется сигналом \overline{SR} , по которому устанавливается счетчик команд и указатель стека (в PSW) в "0"; выбирается банк 0 регистров (RB0) и памяти (MB0); устанавливается порт P0 (DB) в высокоимпедансное состояние (при ЕМА="0"); подготавливается порт P1 и порт P2 для приема информации; блокируется прерывание по входу \overline{INT} и по переполнению таймера-счетчика; останавливается таймер-счетчик; устанавливаются флаги F0 и F1 в "0"; запрещается выдача импульсов по выводу T0; устанавливается триггер TF в "0".

1.3.1. Режим проверки внутренней памяти программ

Режим проверки внутренней памяти программ используется при контроле правильности информации, занесенной в память в процессе ее программирования или изготовления, а также при контроле "чистоты" памяти после ее стирания в микросхеме КМ1816ВЕ48. Под "чистотой" памяти понимается нахождение всех ячеек памяти после стирания в состоянии низкого порогового напряжения, которое обеспечивает на выводах DB0...DB7 состояние низкого уровня.

Временные диаграммы работы в режиме проверки внутренней памяти программ показаны на рис. 1.10а для микросхем КМ1816ВЕ48, КР1816ВЕ49 и на рис. 1.10б, 1.10в для микросхем КР1830ВЕ48. Значения статических и динамических параметров, необходимых для обеспечения режима, приведены в таблицах 1.8 и 1.9.

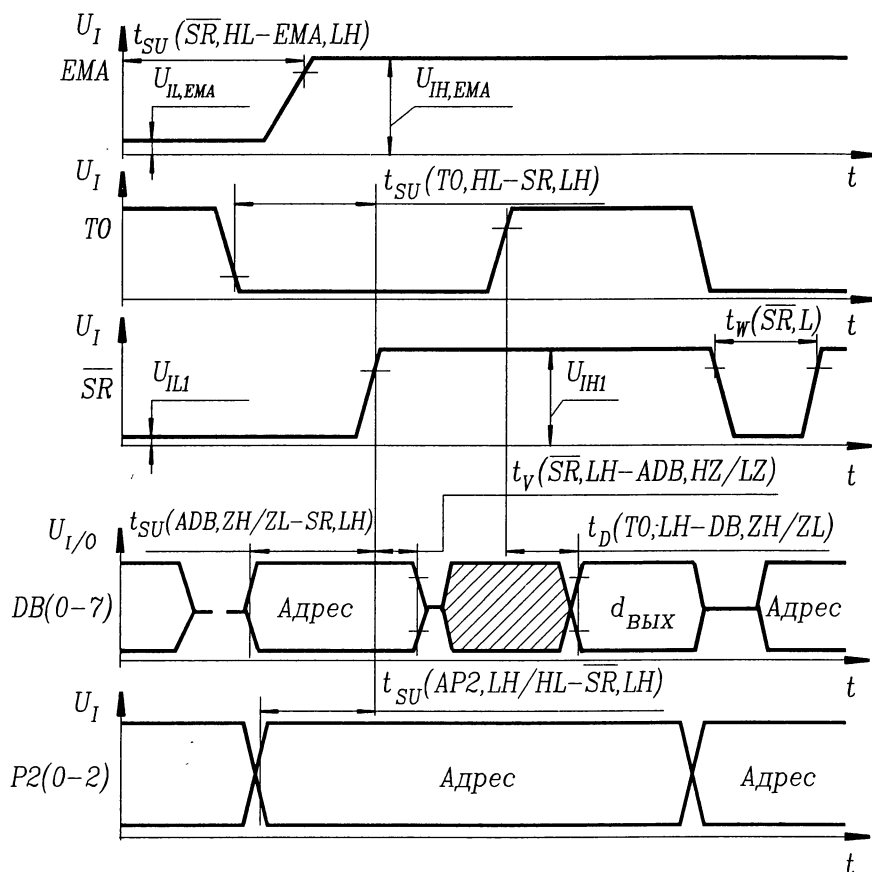


Рис. 1.10а. Временные диаграммы работы микросхем КМ1816ВЕ48 и КР1816ВЕ49 в режиме проверки памяти программ

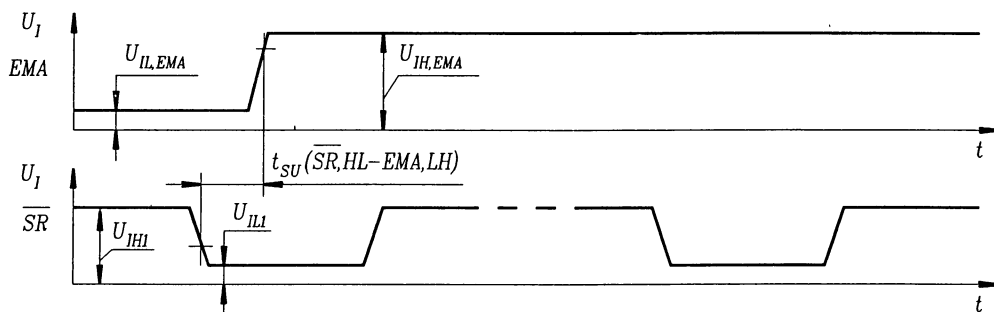


Рис. 1.10б. Первоначальный сигнал \overline{SR} , обеспечивающий сброс КР1830ВЕ48 в режиме чтения внутренней памяти программ

Для микросхем KM1816BE48 и KP1816BE49 процесс проверки внутренней памяти программ включает в себя следующие операции: инициализация (сброс) микросхемы; задание режима проверки внутренней памяти программ; задание адреса; фиксация адреса; чтение данных.

Выводы микросхем EMA, T0, SR, P20...P22 в данном режиме работы ОМЭВМ являются входами. Шина DB0...DB7 сначала работает как входной порт, на который подаются младшие адреса выбираемой ячейки памяти, а затем переходит в режим выходного порта, на котором выставляется записанная в выбранной ячейке памяти информация ($d_{\text{вых}}$ на рис. 1.10а). Работой шины DB0...DB7 управляет сигнал T0. При T0="0" шина DB0...DB7 работает как входной порт, а при T0="1" переводится в режим выходного порта. При EMA=U_{ИН,ЕМА} после перехода сигнала T0 из высокого в низкий логический уровень шина DB0...DB7 не позднее чем через 200 нс переходит в высокоимпедансное состояние.

Подаваемые на одноименные выводы микросхемы сигналы выполняют следующие функции: низкий уровень сигнала SR при EMA=U_{ИЛ,ЕМА} обеспечивает инициализацию микросхемы; EMA — при подаче напряжения высокого уровня U_{ИН,ЕМА} активизирует режим обращения к внутренней памяти программ; T0 — обеспечивает режим контроля при высоком уровне напряжения; SR — при переходе из низкого логического уровня в высокий фиксирует выбранный адрес при низком уровне напряжения на выводе T0; DB0...DB7 — обеспечивают подачу адреса A0...A7 и выдачу данных; P20 P22 — обеспечивают подачу адреса A8, A9 (A10 для KP1816BE49).

Адрес на выводах ОМЭВМ P20—P22 можно выдерживать, как показано на рис. 1.10а, или же снимать по истечении времени $t_V(\overline{SR}, LH-ADB, HZ/LZ)$.

Для работы ОМЭВМ в режиме проверки внутренней памяти программ требуется внутренняя синхронизация, т. е. к выводам ОМЭВМ BQ1, BQ2 должны быть подключены внешние элементы внутреннего генератора или поданы внешние синхросигналы.

Режим проверки внутренней памяти программ для ОМЭВМ KP1830BE48 (рис. 1.10б и рис. 1.10в) существенно отличается от рассмотренного выше для ОМЭВМ серии 1816. Сигнал T0 для KP1830BE48 в рассматриваемом режиме не используется.

На рис. 1.10б показан первоначальный сброс ОМЭВМ, который обязательно должен быть выполнен перед началом чтения ячеек ПЗУ. Временные диаграммы на рис. 1.10в приведены в предположении, что первоначальный сброс уже выполнен, а на входе EMA находится уровень U_{ИН,ЕМА}.

Выводы микросхемы KP1830BE48 EMA, SR, P20, P21 являются входами; вывод ALE — выход. Шина DB0...DB7 может работать в режиме входного порта, на который подают восемь младших разрядов адреса A0...A7 выбираемой ячейки ПЗУ, или в режиме выходного порта, с которого считывают байт информации, записанной в выбранной ячейке ПЗУ ($d_{\text{вых}}$ на рис. 1.10в). Управление при этом производится сигналом SR: если EMA=U_{ИН,ЕМА}, то при SR="0" шина DB0...DB7 переходит в высокоимпедансное состояние и работает как входной порт, а при SR="1" шина DB0...DB7 с некоторой задержкой, определяющей время $t_D(ADB, HZ/LZ-DB, ZH/ZL)$, переводится в режим выходного порта.

На входы ОМЭВМ P20, P21 подают два старших разряда адреса A8, A9 выбираемой ячейки ПЗУ.

Если нарастающим фронтом сигнала ALE (рис. 1.10в) стробируется низкий уровень сигнала SR, то по спаду ALE шина DB0...DB7 уже гарантировано находится в высокоимпедансном состоянии и допустима подача адреса.

Адрес с шины DB0...DB7 лучше всего убирать одновременно с переходом сигнала SR из низкого логического уровня в высокий. В этом случае не позднее, чем через время $t_D(ADB, HZ/LZ-DB, ZH/ZL)$ на шине DB0...DB7 будет выставлена выходная информация $d_{\text{вых}}$. В табл. 1.9 для этого времени приведено два значения:

через время $3t-50$ (максимум) будет выставлен байт $d_{\text{ВЫХ}}$, но это может быть содержимое ячейки ПЗУ со старым адресом, что определяется особенностями внутренней тактировки КР1830ВЕ48; через время $18t-50$ (максимум) на шине $DB0...DB7$ гарантировано будет выставлен байт $d_{\text{ВЫХ}}$ из ячейки ПЗУ с адресом, который был задан по последнему сигналу \overline{SR} .

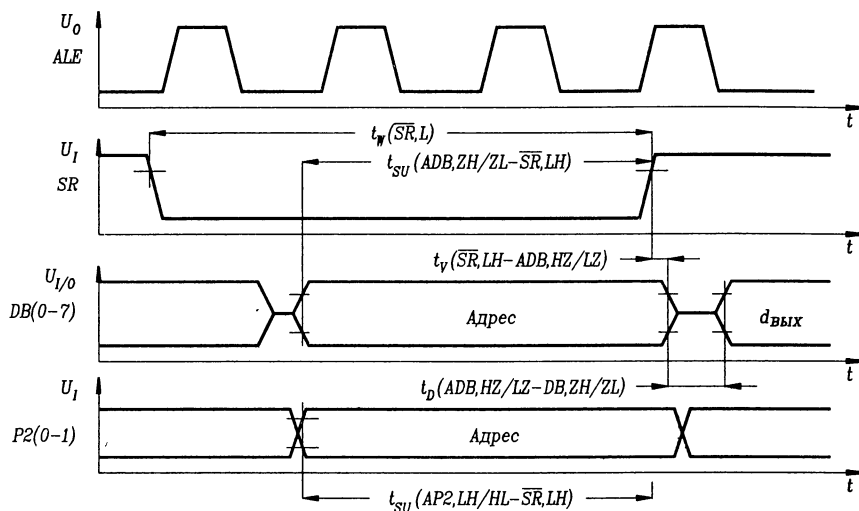


Рис. 1.10в. Чтение произвольного адреса ПЗУ ОМЭВМ КР1830ВЕ48

При организации режима контроля памяти следует иметь в виду, что: по выводам $DB0...DB7$ осуществляется подача сигналов адреса и выдача данных для контроля, поэтому при переходе к режиму контроля необходимо обеспечить высокоимпедансное состояние на выводах $DB0...DB7$, которое исключает попадание на открытые выводы схемы напряжения адресных сигналов, поступающих от источника адресных сигналов; при проверке "чистоты" памяти выходное напряжение на выводах $DB0...DB7$ должно соответствовать напряжению низкого уровня.

1.3.2. Режим программирования внутреннего ППЗУ

Запись информации в память программ микросхемы КМ1816ВЕ48 производится в процессе программирования на специальных устройствах-программаторах, обеспечивающих выбор запоминающих ячеек памяти по заданному адресу.

Временная диаграмма работы микросхемы при программировании и проверке после программирования байта показана на рис. 1.11.

Запись ячейки по заданному адресу производится только для той части информации, которая представлена напряжением высокого уровня, запись "логического 0" по всем адресам ППЗУ осуществляется при стирании с помощью ультрафиолетового облучения.

Выводы микросхемы EMA , $T0$, \overline{SR} , $P20...P21$, U_{dd} , \overline{PR} в данном режиме работы являются входами. Шина $DB0...DB7$ при программировании работает как входной порт, а при проверке после программирования байта переводится в режим выходного порта. Работой шины $DB0...DB7$ управляет сигнал $T0$: при $T0="0"$ шина $DB0...DB7$ работает как входной порт, а при $T0=1$ переводится в режим выходного порта. При $EMA=U_{IN,EMA}$ после перехода сигнала $T0$ из высокого в низкий логический уровень шина $DB0...DB7$ не позднее чем через 200 нс переходит в высокоимпедансное состояние.

Вывод \overline{PR} до программирования должен находиться в высокоимпедансном состоянии.

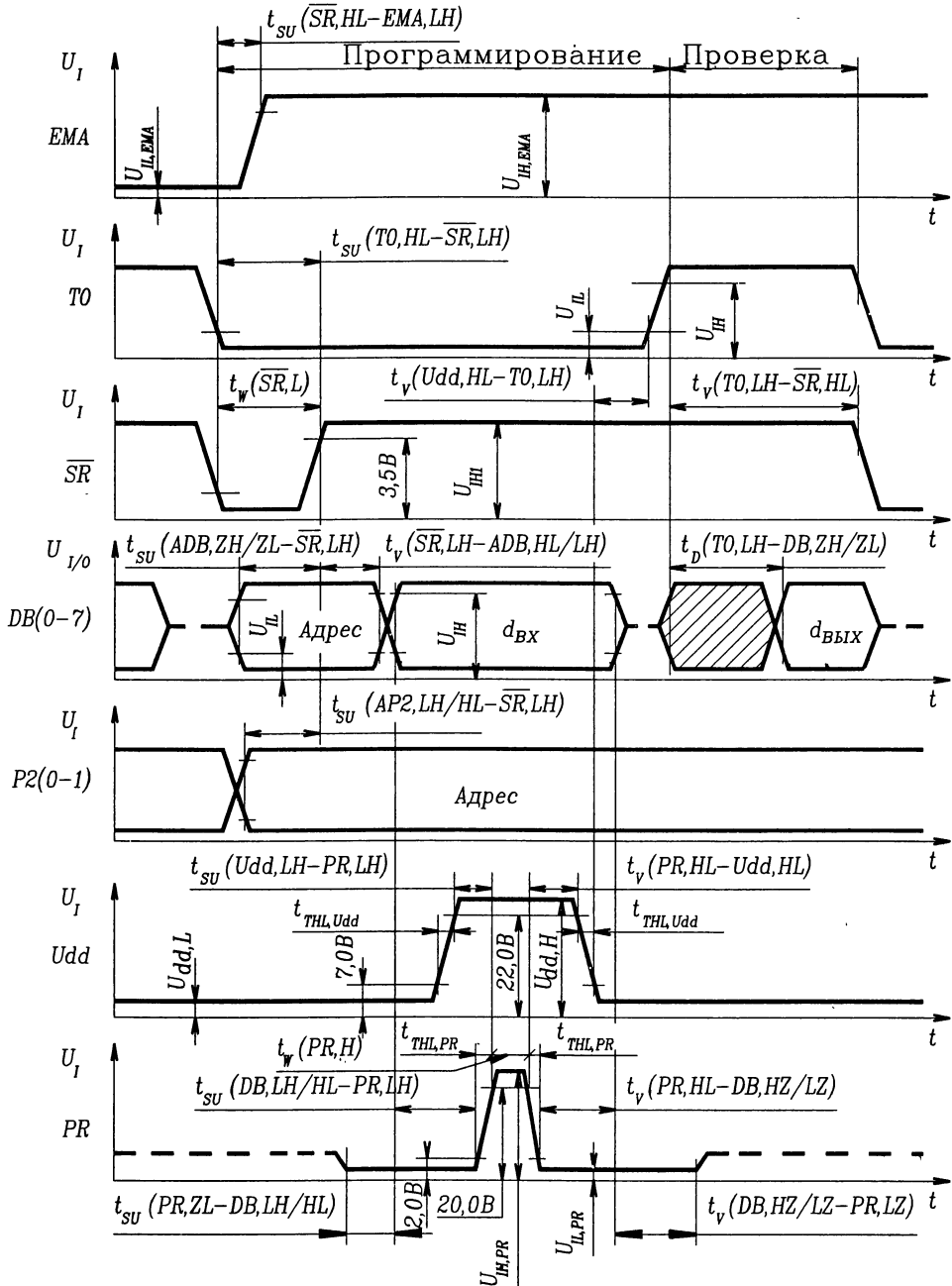


Рис. 1.11. Временная диаграмма работы микросхемы при программировании и проверке после программирования байта

Процесс программирования включает в себя следующие операции: задание режима программирования; задание адреса; фиксация адреса; подача данных; подача импульсов программирования.

После окончания процесса программирования байта данных (до перехода по следующему адресу) при необходимости выполняются операции проверки содержимого запрограммированной ячейки памяти.

Ниже приводится рекомендуемая последовательность действий при программировании. Необходимо особо подчеркнуть, что приводимые при этом в тексте значения напряжений даны условно для наглядности описания. Реальные значения напряжений необходимо брать по рис. 1.11 в табл. 1.8.

В процессе программирования необходимо соблюдать следующую последовательность действий:

1. На выводах контактного устройства установить $U_{dd} = 5В$, $U_{cc} = 5В$, $\overline{SR} = 0В$, $T\emptyset = 5В$, $EMA = 5В$, DB и \overline{PR} в высокоимпедансное состояние. К выводам $BQ1$ и $BQ2$ подключить внешние элементы внутреннего генератора или подать внешние синхроимпульсы.

2. Установить ИС $KM1816BE48$ в контактное устройство программатора, при этом необходимо иметь в виду, что неправильно установленная ИС может выйти из строя.

3. $T\emptyset = \emptyset$ (выбор режима программирования).

4. $EMA = 25 В$ (активизация режима программирования).

5. На выводы DB , $P2\emptyset$, $P21$ подать адрес программируемой ячейки ППЗУ.

6. $\overline{SR} = 5 В$ (фиксация адреса).

7. На выводы DB подать данные ($d_{вх}$ на рис. 1.11).

8. $U_{dd} = 25 В$ (подача напряжения программирования).

9. $\overline{PR} = \emptyset В$, подача импульса длительностью 50 мс и амплитудой 25 В.

10. $U_{dd} = 5 В$.

11. $T\emptyset = 5 В$ (режим проверки ППЗУ).

12. Чтение и проверка данных на выводах DB ($d_{вых}$ на рис. 1.11).

13. $T\emptyset = \emptyset В$.

14. $\overline{SR} = \emptyset В$ и повторить действия, начиная с п. 5.

15. При извлечении микросхемы из контактного устройства выводы его должны быть в состоянии, оговоренном в п. 1.

Для ОМЭВМ $KM1816BE48$ допустим режим программирования серий укороченных импульсов. При этом используется до 25 импульсов \overline{PR} длительностью 1 мс каждый. После подачи каждого импульса \overline{PR} выполняется проверка. Если в результате проверки обнаружено, что байт запрограммировался, дается еще 3 импульса \overline{PR} для закрепления и выполняется переход по следующему адресу. Программирование серий укороченных импульсов рекомендуется вести при повышенном напряжении питания $U_{cc} = 5,25 В$.

В случае, если правильно запрограммировать байт не удалось (как при программировании одним длинным импульсом, так и при программировании серий коротких), микросхема должна быть либо отбракована, либо может быть предпринята еще одна попытка программирования.

1.3.3. Режим работы с внутренней памятью программ

Режим работы с внутренней памятью программ устанавливается при подключении вывода EMA к выводу GND . Выполнение программы начинается с команды по адресу $\emptyset\emptyset$ после сброса. При работе с внутренней памятью программ никакие внешние управляющие сигналы, за исключением ALE , ОМЭВМ не формируются. Если микросхема имеет внутреннюю память программ ($KM1816BE48$, $KP1816BE49$, $KP1830BE48$) и вывод EMA подключен к выводу GND , то обращения к этой памяти выполняются автоматически до тех пор, пока адрес обращения не превышает максимальный адрес внутренней памяти программ микросхемы. Если адрес обращения превышает максимальный адрес внутренней памяти программ, то ОМЭВМ автоматически переходит в режим работы с внешней памятью программ. С целью увеличения производительности ОМЭВМ предусмотрено совмещение выполнения внутренних операций в цикле. Например, выполнение выбранной из памяти команды и подготовка следующего адреса команды производятся одновременно. Для синхронизации внешних устройств ввода-вывода можно использовать сигнал ALE , выдаваемый ОМЭВМ в каждом машинном цикле.

1.3.4. Режим работы с внешней памятью

Наряду с наличием возможности работы ОМЭВМ с внешней памятью программ в качестве дополнения к внутренней памяти, имеется режим работы исключительно с внешней памятью программ, который обеспечивается путем подачи напряжения от источника +5 В на вывод ЕМА. Для микросхем КР1816ВЕ35, КР1816ВЕ39, КР1830ВЕ35, не имеющих внутренней памяти программ, этот режим является единственно возможным, и для этих микросхем вывод ЕМА всегда должен быть подключен к +5 В. Для микросхем КМ1816ВЕ48, КР1816ВЕ49, КР1830ВЕ48 в этом режиме внутренняя память программ отключается и, начиная с нулевого адреса, все обращения выполняются к внешней памяти программ.

Временная диаграмма работы ОМЭВМ с внешней памятью программ показана на рис. 1.12. Заштрихованные участки временной диаграммы указывают интервалы, в которых шина DB находится в высокоимпедансном состоянии.

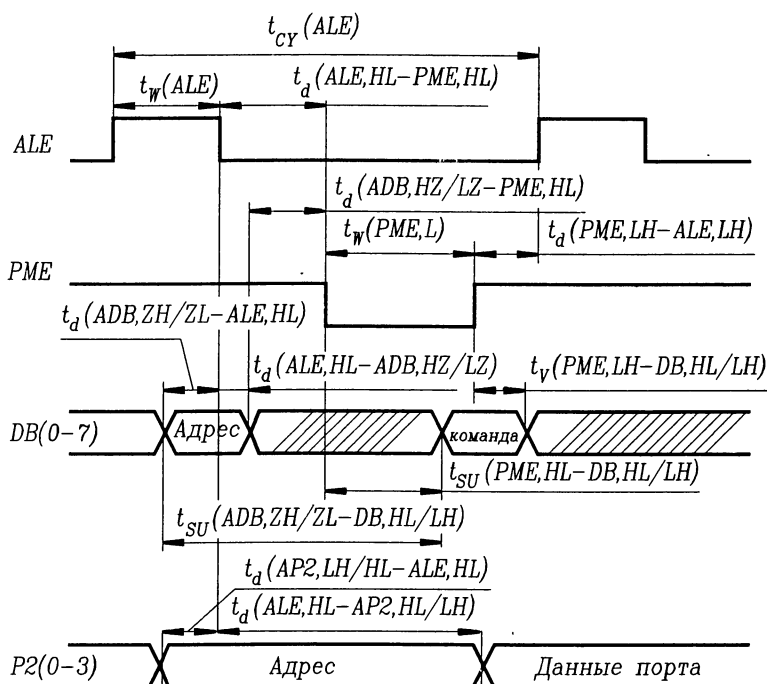


Рис. 1.12. Временные диаграммы работы микросхем с внешней памятью программ

Содержимое двенадцати разрядов счетчика команд выводится в качестве адреса обращения к внешнему ЗУ команд на шину данных DB0...DB7 (разряды 0...7 счетчика команд) и на выходы P20...P23 порта P2 (разряды 8...11 счетчика команд), адрес внешнего ЗУ фиксируется по окончании сигнала ALE; шина данных переходит в режим ввода и процессор принимает 8-разрядное слово команды, выборка команды из внешнего ЗУ фиксируется по окончании сигнала PME. После окончания времени действия адреса на выводах P2(0—3) выставляется информация, находящаяся в соответствующих защелках порта P2, которая будет находиться на выходах порта P2 до следующего обращения к памяти программ (до следующей выдачи адреса).

В ОМЭВМ используется механизм переключения банков памяти программ по 2 Кбайт каждый. Выбор банка определяется содержимым старшего разряда счетчика команд, который загружается содержимым триггера переключения банка памяти DBF каждый раз при выполнении команды перехода JMP или CALL при вызове

подпрограммы. При инкрементировании содержимого счетчика команд в старший разряд переноса нет.

Триггер DBF может устанавливаться в "1" (выбор банка 1 — ячейки с адресами 2048...4097) с помощью команды SEL MB1 и сбрасывается в "0" (выбора банка 0 — ячейки с адресом 0...2047) по команде SEL MB0 или по сигналу общего сброса \overline{SR} .

Выбор банка 0 по команде SEL MB0 или по сигналу \overline{SR} позволяет обращаться к ячейкам 0...2047 исключительно внешней памяти программ, если вход ЕМА подключен к источнику +5 В.

Команда SEL MB должна выполняться в программе перед тем, как необходимо переключить банк памяти программ. Собственно переключение осуществляется при выполнении очередной команды перехода или вызова подпрограмм. Поскольку при выполнении команды CALL все двенадцать разрядов СК, включая старший, запоминаются в стеке, пользователь может переходить к подпрограмме, расположенной за пределами текущего банка памяти программ. После окончания подпрограммы номер банка памяти, из которого она была вызвана, будет восстановлен командой возврата из подпрограммы. Следует помнить, что состояние триггера DBF не изменяется командой возврата.

Выборка из внешней памяти программ команд, расположенных по адресам, меньшим адреса 1024, осуществляется при подключении вывода ЕМА к источнику +5 В. ОМЭВМ КР1816ВЕ35, КР1830ВЕ35 и КР1816ВЕ39 не содержат на кристалле память программ и всегда работают в режиме обращения к внешней памяти.

На рис. 1.13 изображена схема подключения памяти программ к ОМЭВМ с использованием трех стандартных ППЗУ объемом 1К * 8 байт.

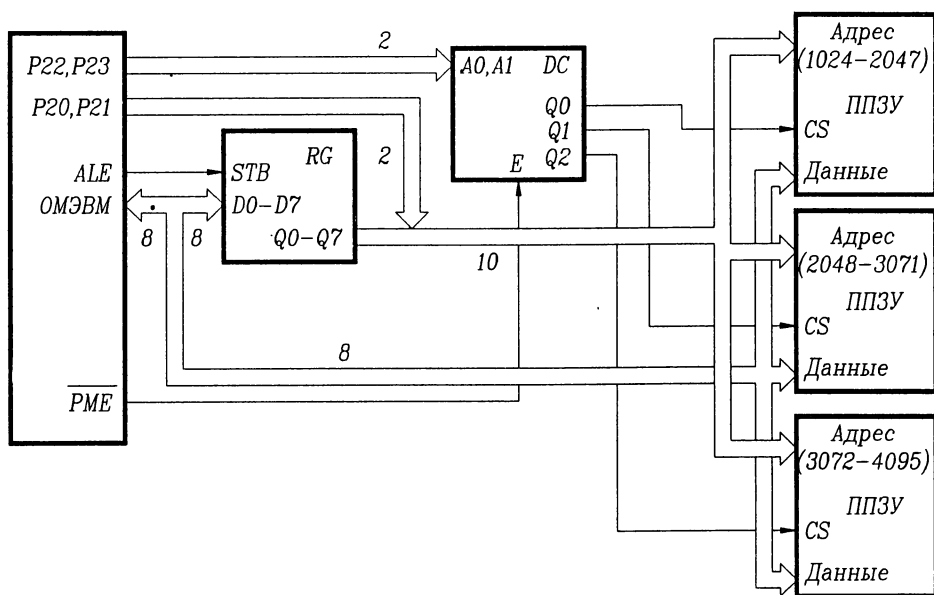


Рис. 1.13. Схема подключения памяти программ к ОМЭВМ

В соответствии с представленной схемой восемь младших бит адреса фиксируются по заднему фронту сигнала ALE в регистре-защелке RG. Выходные сигналы регистра RG и адрес, выставленный на выходах ОМЭВМ P20, P21, образуют 10-разрядный адрес, подаваемый на адресные входы трех микросхем ППЗУ. Два старших разряда адреса с выходов ОМЭВМ P22, P23 подаются на адресные входы дешифратора DC. Низкий уровень сигнала \overline{PME} разрешает работу дешифратора, сигналами с выходов Q0—Q2 которого производится разрешение работы одной из микросхем ППЗУ. С выбранной БИС ППЗУ считывается байт в ОМЭВМ. По окончании сигнала \overline{PME} низкого уровня запрещается работа

дешифратора и шины данных всех БИС ППЗУ переходят в высокоимпедансное состояние. Схема готова к следующему циклу чтения.

Использование регистра RG необходимо для развязки шин адреса и данных, т. к. порт P \bar{O} (DB) по одним и тем же физическим линиям в режиме временного мультиплексирования сначала выдает байт адреса, а затем принимает байт команды (данных) из памяти.

Память данных можно расширять за пределы встроенной памяти с помощью двунаправленной шины данных DB, по которой передаются все адреса и данные. При этом через регистры косвенного адреса R \bar{O} и R1 возможен доступ к внешней памяти данных емкостью до 256 байт.

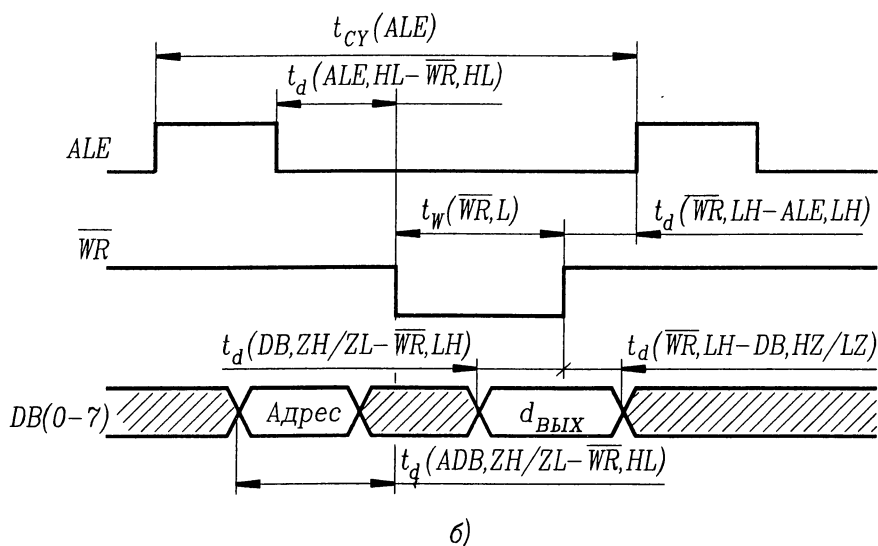
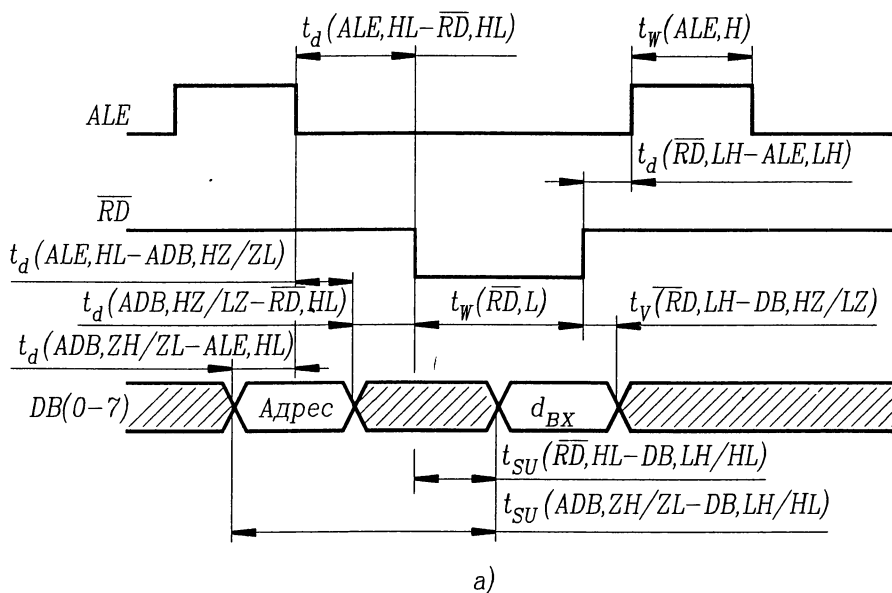


Рис. 1.14. Временные диаграммы работы микросхем с внешней памятью данных:
а — чтение; б — запись

Временная диаграмма работы ОМЭВМ с внешней памятью данных показана на рис. 1.14. Цикл чтение-запись происходит следующим образом:

1. Содержимое регистров R0 или R1 выводится на шину данных.
2. Стробующий сигнал фиксации адреса ALE указывает, что адрес выводится на шину данных. Задний фронт сигнала ALE используется для фиксации адреса внешней памяти данных.
3. Выходной строб управления считыванием \overline{RD} или записью \overline{WR} на соответствующих выводах ОМЭВМ показывает, какой тип доступа к памяти имеет место. Выводимые данные должны быть зафиксированы в момент прохождения заднего фронта \overline{WR} , а вводимые в момент прохождения заднего фронта \overline{RD} .

Входные и выходные данные обозначены на диаграммах на рис. 1.14 соответственно $d_{вх}$ и $d_{вых}$. Заштрихованные участки временной диаграммы указывают интервалы, в которых шина DB находится в высокоимпедансном состоянии.

Доступ к внешней памяти данных осуществляется с помощью специальных двухцикловых команд пересылки MOVX A, @R и MOVX @R, A, по которым происходит передача 8-разрядных данных между аккумулятором и ячейкой внешней памяти, адресуемой содержимым одного из двух регистров указателей R0 или R1. На рис. 1.15 приведена схема подключения внешней памяти данных с использованием двух ИС статического запоминающего устройства произвольной выборки (ЗУПВ) с организацией 256*4. Регистр RG используется для фиксации адреса. Каждая 4-разрядная половина шины данных DB непосредственно соединяется с двунаправленными 4-разрядными шинами данных I/O внешнего ЗУПВ. Выходной стробирующий сигнал ОМЭВМ \overline{WR} управляет входом R/ \overline{W} ЗУПВ, а выходные драйверы шины данных ЗУПВ управляются по входу \overline{OD} сигналом \overline{RD} . Если не требуется использования дополнительного ЗУПВ, вход \overline{CS} (выбор кристалла) ЗУПВ присоединяется к общему выводу GND. На рис. 1.16 показан возможный вариант расширения памяти данных с использованием ЗУПВ с организацией 256*8. Предполагается, что используемая микросхема ЗУПВ имеет внутреннюю защелку адреса (регистр RG на рис. 1.15.)

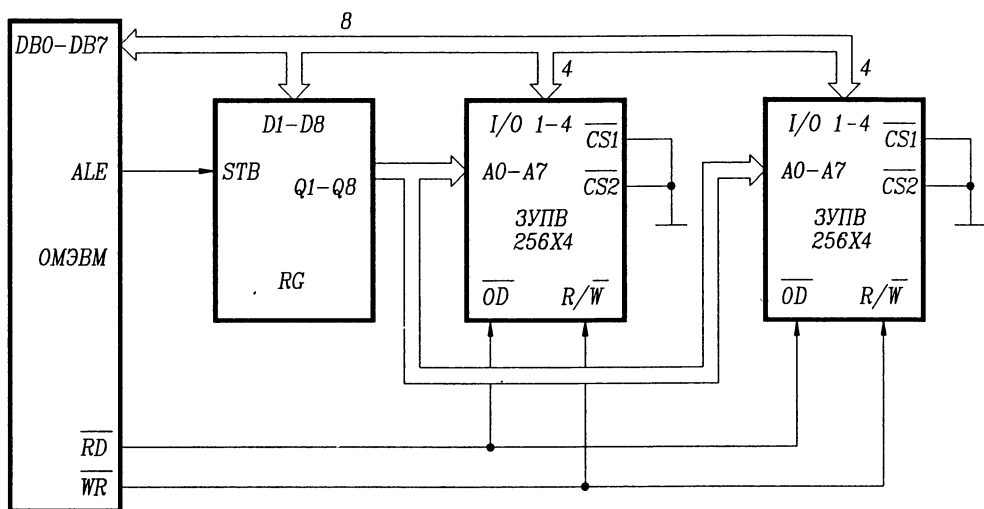


Рис. 1.15. Схема подключения внешней памяти данных с использованием двух ИС статического запоминающего устройства произвольной выборки (ЗУПВ) с организацией 256*4

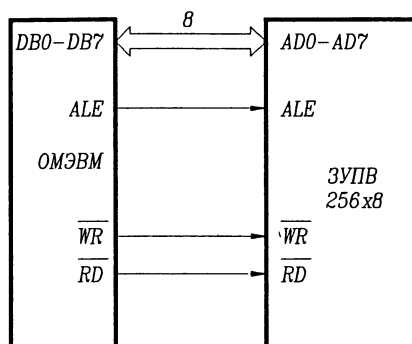


Рис. 1.16. Вариант расширения памяти данных с использованием ЗУПВ с организацией 256×8

1.3.5. Режим пошагового выполнения команд

В ОМЭВМ семейства МК48 предусмотрена возможность организации работы в пошаговом режиме, который используется в процессе отладки и проверки памяти программ. В этом режиме при выполнении программы процессор можно остановить на каждой команде. В режиме останова адрес следующей команды, которая должна быть выбрана, выведен на шину данных и младшие разряды порта P2.

При организации пошагового режима выполняются следующие операции:

1. В ОМЭВМ поступает запрос на останов путем подачи напряжения низкого уровня на вывод \overline{SS} .

2. Процессор ОМЭВМ останавливается на этапе выборки следующей команды; при этом завершается выполнение текущей команды, независимо от того, какая команда по длительности — одноцикловая или двухцикловая.

3. ОМЭВМ подтверждает, что она находится в режиме останова путем установки сигнала ALE в состояние высокого уровня. В этом состоянии (которое можно сохранить сколь угодно долго) адрес следующей команды, которая будет выбираться после текущей, должен присутствовать на шине DB и младших разрядах порта P2.

4. Для того чтобы выйти из режима останова, на выводе \overline{SS} необходимо установить высокий уровень, что обеспечит выборку следующей команды. Процессор подтверждает выход из состояния останова путем установки низкого уровня сигнала ALE.

5. Для того чтобы обеспечить останов на следующей команде, устанавливается низкий уровень сигнала \overline{SS} , как только уровень сигнала ALE станет низким. Если уровень сигнала \overline{SS} остается высоким, ОМЭВМ отрабатывает программу в обычном динамическом режиме.

На рис. 1.17 показан пример реализации пошагового режима, а на рис. 1.18 — диаграмма работы ОМЭВМ в режиме пошагового выполнения программы. Для включения пошагового режима необходимо соответствующий переключатель установить в положение "Пошаговый режим". Кнопкой "шаг" обеспечивается переход на следующую команду.

Необходимо отметить, что схема на рис. 1.17 является иллюстрационной, а не практической. К примеру, режим D-триггера, при котором на оба входа установки R и S одновременно поданы низкие уровни, является некорректным.

Работа ОМЭВМ в режиме пошагового выполнения команд идентична для случаев, когда выполняемая программа находится во внешней или во внутренней памяти программ.

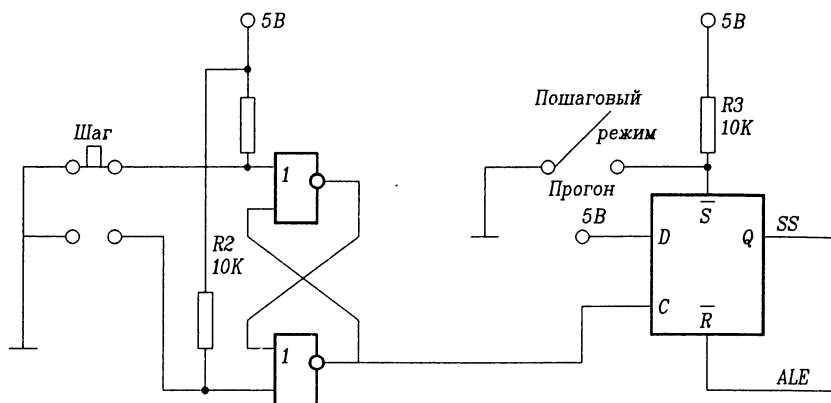


Рис. 1.17. Пример реализации пошагового режима

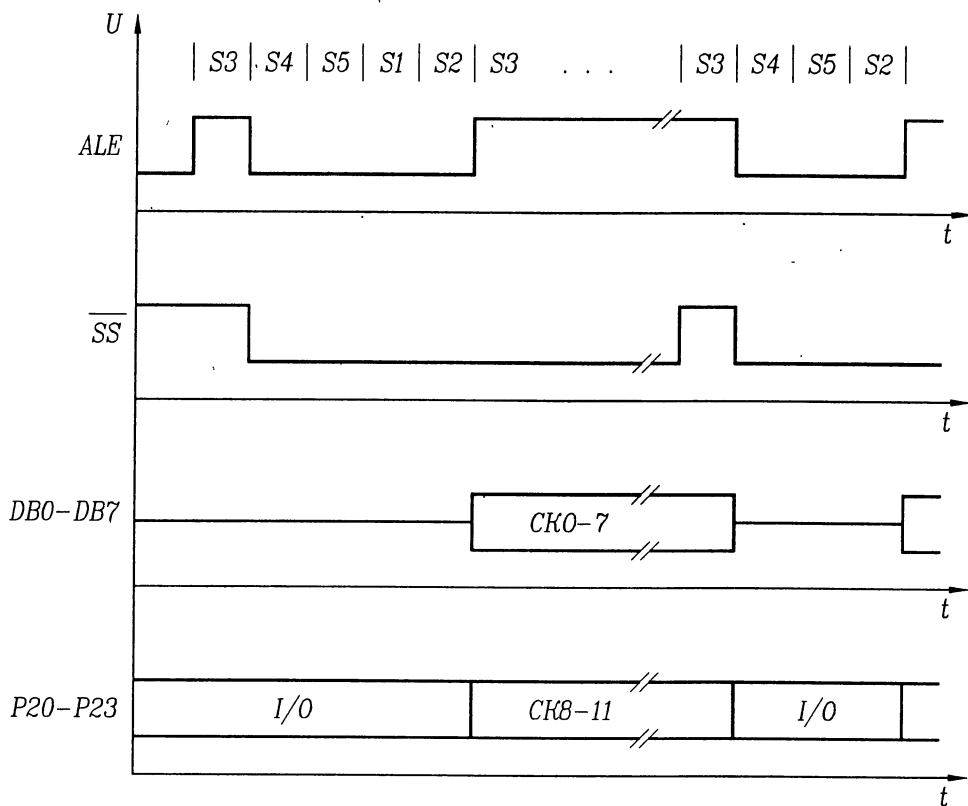


Рис. 1.18 Диаграмма работы ОМЭВМ в режиме пошагового выполнения программы

1.3.6. Расширение канала ввода-вывода.

Шина данных ОМЭВМ совместима с 8-разрядной двунаправленной шиной микропроцессора КР580ВМ80А, что обеспечивает возможность подключения к ОМЭВМ семейства МК48 периферийных устройств в виде микросхем серии КР580, которые можно использовать для реализации ряда дополнительных специальных функций, а также для увеличения числа каналов ввода-вывода и их типов.

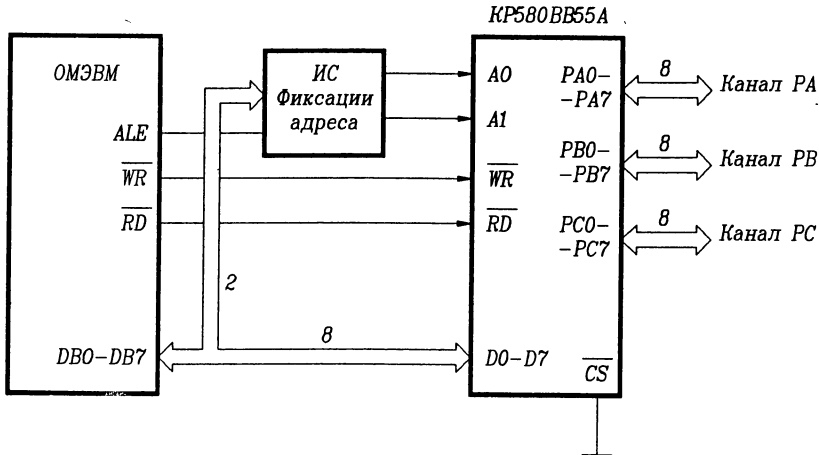


Рис. 1.19. Подключение ОМЭВМ к БИС KP580BB55A. Вариант 1

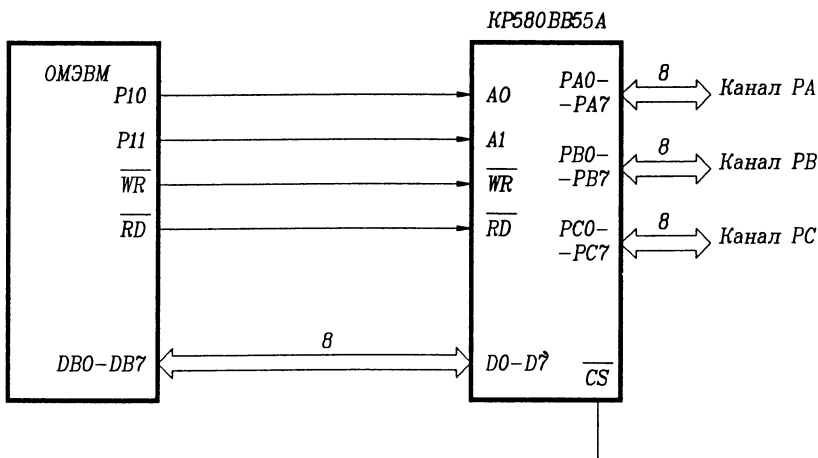


Рис. 1.20. Подключение ОМЭВМ к БИС KP580BB55A. Вариант 2

На рис. 1.19, 1.20 показано подсоединение ОМЭВМ к стандартному периферийному устройству серии KP580 — ИС KP580BB55A. Интегральная схема KP580BB55A обеспечивает три 8-разрядных программируемых канала ввода-вывода (PA, PB, PC).

Для связи ОМЭВМ с ИС KP580BB55A в последней используются 8-разрядная двунаправленная шина данных $D0-D7$, входы \overline{RD} и \overline{WR} для управления чтением-записью, вход "выбор кристалла" \overline{CS} для активизации логики управления чтением-записью и адресные входы $A0$, $A1$ для выбора различных внутренних регистров. Выводы ОМЭВМ \overline{DB} , \overline{RD} , \overline{WR} соединяются с соответствующими выводами ИС KP580BB55A. При реализации этой схемы выбирается способ, с помощью которого должны адресоваться внутренние регистры ИС KP580BB55A. Если регистры адресуются как внешняя память данных с использованием команды MOVX, соответствующее число разрядов адреса (в данном случае 2) должно фиксироваться на шине с помощью сигнала ALE так же, как и при использовании внешней памяти данных (рис. 1.19). Если к шине данных ОМЭВМ подсоединяется только одно

устройство ИС КР580ВВ55А, вход ИС \overline{CS} заземляется. Если используется несколько ИС КР580ВВ55А, должны фиксироваться дополнительные разряды адреса, которые используются для выбора конкретной микросхемы.

Возможно применение другого способа адресации, при котором исключается необходимость во внешних схемах фиксации адреса и дешифраторах выбора кристалла благодаря использованию выходных линий портов ОМЭВМ непосредственно в качестве линий адресации и выбора микросхемы. При этом способе требуется вывод на выходной порт командой OUTL адресной информации перед выполнением команды MOVX (рис. 1.20).

Кроме этого, ОМЭВМ позволяет увеличить число каналов ввода-вывода за счет использования команд MOV D, A; PD; MOV D, PD, A; ANLD PD, A; ORLD PD, A. При этом обмен информацией осуществляется через порт P2 (P20...P23). Для реализации данного режима существует специальная микросхема — расширитель портов.

Схема соединения ОМЭВМ с расширителем портов — ИС КР580ВР43 показана на рис. 1.21. Расширитель портов КР580ВР43 содержит четыре 4-битных двунаправленных порта P4, P5, P6, P7, которые являются расширением резидентной системы ввода/вывода ОМЭВМ. Обращения к портам P4—P7 производятся по командам MOV D, A; ANLD и ORLD, выполняемым совместно ОМЭВМ и расширителем. Каждая из указанных команд реализует обмен между ОМЭВМ и КР580ВР43. Обмен производится по линиям P20...P23 и синхронизируется выходным сигналом ОМЭВМ \overline{PR} . Каждый обмен состоит из двух пересылок 4-разрядных полубайтов, первый из которых содержит код управляющего слова и адрес порта P4—P7, а второй — 4 бита данных (рис. 1.22). Переход сигнала \overline{PR} из состояния высокого уровня в состояние низкого уровня указывает, что на выводах P20...P23 находятся код управляющего слова и адрес порта, а переход сигнала \overline{PR} из состояния низкого уровня в состояние высокого уровня указывает на то, что на выводах P20...P23 находятся данные.

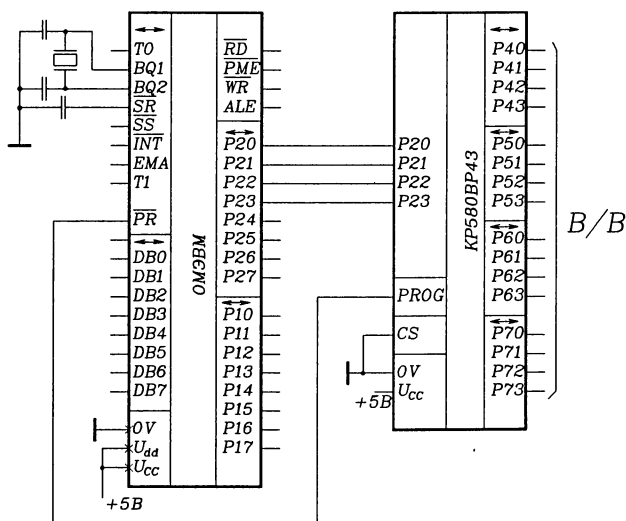


Рис. 1.21. Схема соединения ОМЭВМ с расширителем портов — ИС КР580ВР43

Изменить состояние сигналов на портах P4—P7, работающих в режиме вывода, можно следующими способами: командой MOV D, A загрузить в требуемый порт новое значение; командой ORLD PD, A произвести логическое сложение порта с аккумулятором с загрузкой результата в порт; командой ANLD PD, A произвести логическое умножение порта и аккумулятора с загрузкой результата в порт. Логические операции выполняет КР580ВР43, а не ОМЭВМ.

Для настройки порта расширителя на ввод данных необходимо выполнить для него команду ввода $MOV D, Rr$. Если до выполнения этой команды порт находился в режиме вывода, то результат первого ввода является неопределенным, а все последующие команды ввода будут давать правильный результат. При выполнении операции ввода порт настраивается на ввод и его выходы переключаются в высокоимпедансное состояние.

При включении питания все порты ИС КР580ВР43 переходят в высокоимпедансное состояние, а линии $P20...P23$ настраиваются на ввод.

Временные диаграммы работы ОМЭВМ с дополнительными портами приведены на рис. 1.21а и рис. 1.21б.

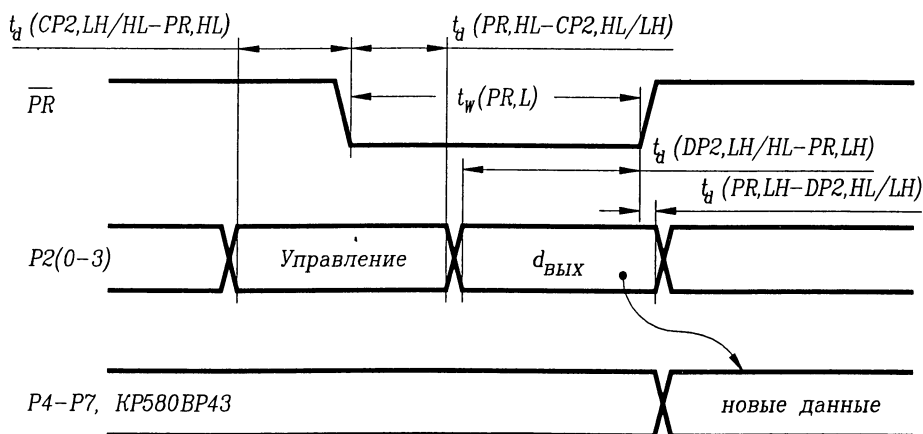


Рис. 1.21а. Временные диаграммы вывода данных при работе с КР580ВР43

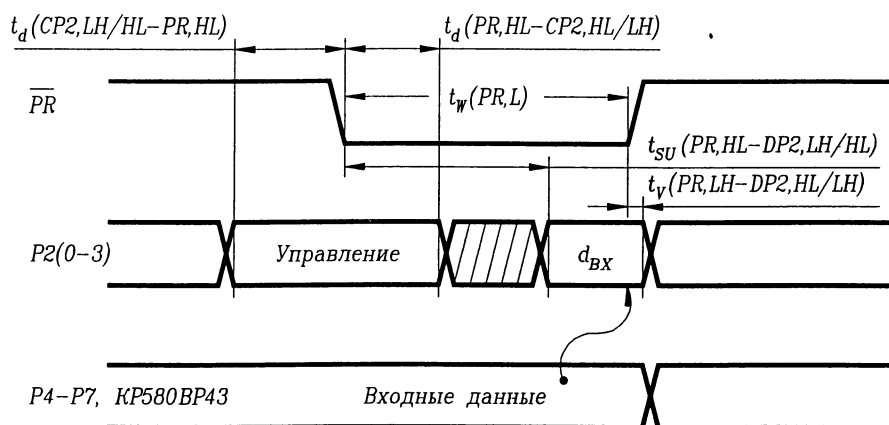


Рис. 1.21б. Временные диаграммы ввода данных при работе с КР580ВР43

Рис. 1.21а иллюстрирует вывод данных $d_{\text{ВЫХ}}$ из ОМЭВМ на порты $P4-P7$ расширителя портов КР580ВР43.

На рис. 1.21б показан ввод данных с портов $P4-P7$ расширителя портов КР580ВР43 в ОМЭВМ ($d_{\text{ВХ}}$).

На рис. 1.23 приведены временные диаграммы работы портов $P1$ и $P2$ в различных режимах с учетом выборки соответствующей команды из внешней памяти программ синхронно с сигналом PME (различные фрагменты диаграмм на рис. 1.23 соответствуют выполнению команд $OUTL P1, A$; $OUTL P2, A$; $MOV D$; $ANLD$;

ORLD). Как видно из рисунка, адрес на выводах P2(0—3) выставлен на время t_d (ALE, HL-AP2, HL/LH) после окончания ALE, после чего адрес снимается и на выводах P2(0—3) выставляется информация, находящаяся в соответствующих выводах P20...P23 защелках порта P2 ("Данные порта" на рис. 1.23). Смена информации на портах P1 и P2 "Данные порта" — "Новые данные" соответствует выполнению команд OUTL P1, A и OUTL P2, A. Три нижних диаграммы на рис. 1.23 относятся к выполнению ОМЭВМ команд MOVD, ANLD и ORLD при расширении портов ввода/вывода.

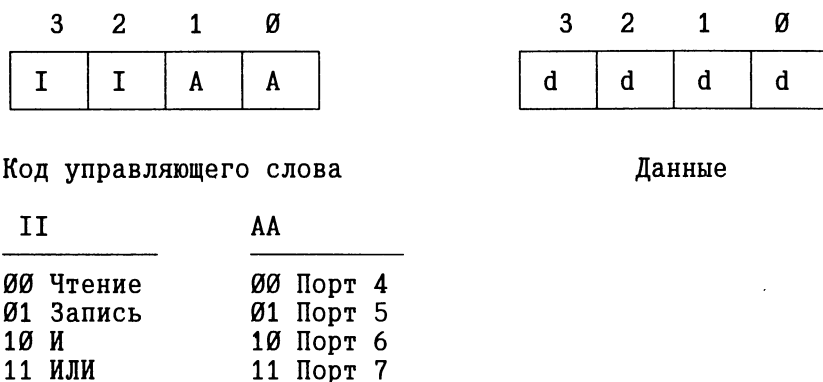


Рис. 1.22. Формат пересылок при обменах ОМЭВМ с KP580BP43

1.3.7. Режим уменьшенного энергопотребления

В отличие от БИС KM1816BE48, KP1816BE35, микросхемы KP1816BE39, KP1816BE49 имеют режим уменьшенного энергопотребления, а микросхемы KP1830BE35, KP1830BE48 — режим микропотребления.

Уменьшенное энергопотребление устанавливается при подаче на выводы U_{cc} и \overline{SR} напряжения низкого уровня, а на вывод U_{dd} — напряжения питания. Рекомендуется следующая последовательность действий:

1. Необходимые данные переслать во внутреннее ОЗУ.
2. Подать напряжение низкого уровня на вывод \overline{SR} , чем обеспечивается сохранение данных во внутреннем ОЗУ.
3. На вывод U_{cc} подать напряжение низкого уровня, при этом на выводе U_{dd} остается напряжение питания +5 В, чем обеспечивается работоспособность ОЗУ.

Выход из режима пониженного энергопотребления обеспечивается подачей на вывод U_{cc} напряжения питания. При этом, после установления на выводе U_{cc} напряжения не менее минимального, на выводе \overline{SR} напряжение низкого уровня должно сохраняться не менее $4 t_{cy}$ ($t_{cy}=15/f_{BQ1}$, где f_{BQ1} — частота внутреннего тактового генератора ОМЭВМ). После этого БИС KP1816BE39, KP1816BE49 начинает выполнение программы с адреса 0000H.

Режим микропотребления в микросхемах KP1830BE35, KP1830BE48 может быть установлен аппаратным или программным способом. В первом случае на вывод U_{dd} необходимо подать напряжение низкого уровня. При этом работа внутреннего генератора тактовых сигналов блокируется и микроЭВМ устанавливается в состояние как и после подачи сигнала \overline{SR} . В этом режиме содержимое внутреннего ОЗУ, аккумулятора и старших разрядов PSW сохраняется. При подаче на вывод U_{dd} напряжения высокого уровня микроЭВМ выходит из режима микропотребления и начинает выполнение программы с нулевого адреса.

Для перевода микроЭВМ в режим микропотребления программным способом необходимо выполнить команду IDL (01H). В этом случае выполнение текущей программы прекращается, внутренние фазовые и тактовые сигналы не формируются.

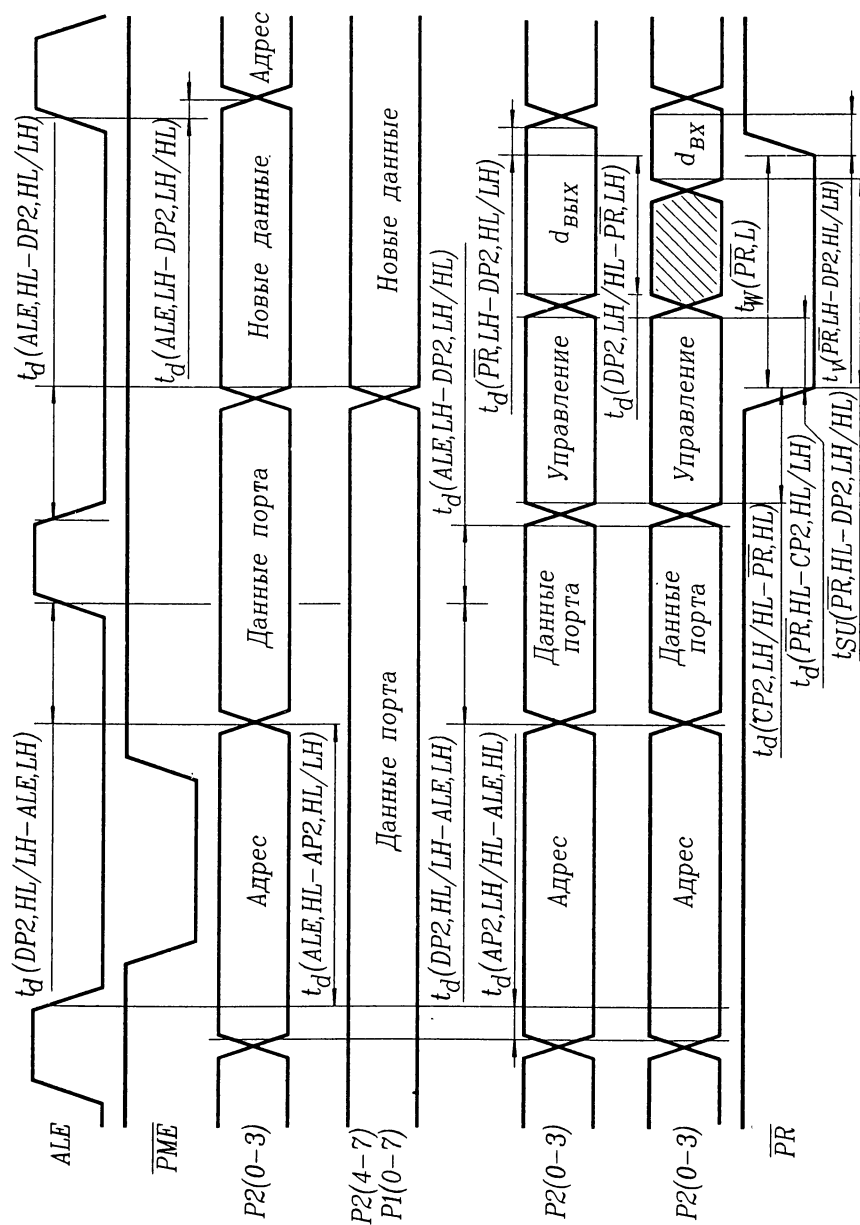


Рис. 1.23. Временные диаграммы работы портов P1 и P2

Продолжение таблицы 1.4

ПЕРЕЧЕНЬ КОМАНД МИКРОЭВМ СЕМЕЙСТВА МК48			
МНЕМОНИКА	КОД	ВРЕМЯ (в циклах)	Страница в описании
ADD A, REG	01101REG	1	56
ADD A, @R	0110000R	1	56
ADD A, #DATA	000000011	2	56
	DDDDDDDD		
ADDC A, REG	01111REG	1	57
ADDC A, @R	0111000R	1	57
ADDC A, #DATA	00010011	2	58
	DDDDDDDD		
ANL A, REG	01011REG	1	58
ANL A, @R	0101000R	1	59
ANL A, #DATA	01010011	2	59
	DDDDDDDD		
ANL BUS, #DATA	10011000	2	59
	DDDDDDDD		
ANL PR, #DATA	100110PR	2	60
	DDDDDDDD		
ANLD PD, A	100111PD	2	60
CALL ADDR	AAA10100	2	60
	AAAAAAAA		
CLR A	00100111	1	61
CLR C	10010111	1	61
CLR F0	10000101	1	61
CLR F1	10100101	1	62
CPL A	00110111	1	62
CPL C	10100111	1	62
CPL F0	10010101	1	62
CPL F1	10110101	1	62
DA A	01010111	1	63
DEC A	00000111	1	63
DEC REG	11001REG	1	63
DIS I	00010101	1	64
DIS TCNTI	00110101	1	64
DJNZ REG, ADDR	11101REG	2	64
	AAAAAAAA		
EN I	00000101	1	65
EN TCNTI	00100101	1	65
ENT0 CLK	01110101	1	65
IDL	00000001	1	66
IN A, PR	000010PR	2	66
INC A	00010111	1	66
INC REG	00011REG	1	66
INC @R	0001000R	1	67
INS A, BUS	00001000	2	67
JBBIT ADDR	BIT10010	2	67
	AAAAAAAA		
JC ADDR	11110110	2	68
	AAAAAAAA		
JF0 ADDR	10110110	2	68
	AAAAAAAA		
JF1 ADDR	01110110	2	69
	AAAAAAAA		
JMP ADDR	AAA00100	2	69
	AAAAAAAA		
JMPP @A	10110011	2	70

Продолжение таблицы 1.4

МНЕМОНИКА	КОД	ВРЕМЯ (в циклах)	Страница в описании
JNC ADDR	11100110 AAAAAAAA	2	70
JNI ADDR	10000110 AAAAAAAA	2	70
JNT0 ADDR	00100110 AAAAAAAA	2	71
JNT1 ADDR	01000110 AAAAAAAA	2	71
JNZ ADDR	10010110 AAAAAAAA	2	72
JTF ADDR	00010110 AAAAAAAA	2	72
JT0 ADDR	00110110 AAAAAAAA	2	73
JT1 ADDR	01010110 AAAAAAAA	2	73
JZ ADDR	11000110 AAAAAAAA	2	73
MOV A, #DATA	00100011 DDDDDDDD	2	74
MOV A, PSW	11000111	1	76
MOV A, REG	11111REG	1	74
MOV A, @R	1111000R	1	75
MOV A, T	01000010	1	77
MOV PSW, A	11010111	1	76
MOV REG, A	10101REG	1	75
MOV REG, #DATA	10111REG DDDDDDDD	2	75
MOV @R, A	1010000R	1	74
MOV @R, #DATA	1011000R DDDDDDDD	2	76
MOV T, A	01100010	1	77
MOVD A, PD	000011PD	2	77
MOVD PD, A	001111PD	2	77
MOVP A, @A	10100011	2	78
MOVP3 A, @A	11100011	2	78
MOVX A, @R	1000000R	2	78
MOVX @R, A	1001000R	2	78
NOP	00000000	1	79
ORL A, REG	01001REG	1	79
ORL A, @R	0100000R	1	79
ORL A, #DATA	01000011 DDDDDDDD	2	79
ORL BUS, #DATA	10001000 DDDDDDDD	2	80
ORL PR, #DATA	100010PR DDDDDDDD	2	80
ORLD PD, A	100011PD	2	80
OUTL BUS, A	00000010	2	81
OUTL PR, A	001110PR	2	81
RET	10000011	2	81
RETR	10010011	2	82
RL A	11100111	1	82
RLC A	11110111	1	82
RR A	01110111	1	83
RRC A	01100111	1	83
SEL MB0	11100101	1	83

Продолжение таблицы 1.4

МНЕМОНИКА	КОД	ВРЕМЯ (в циклах)	Страница в описании
SEL MB1	11110101	1	83
SEL RB0	11000101	1	84
SEL RB1	11010101	1	84
STOP TCNT	01100101	1	84
STRT CNT	01000101	1	85
STRT T	01010101	1	85
SWAP A	01000111	1	85
XCH A, REG	00101REG	1	85
XCH A, @R	0010000R	1	86
XCHD A, @R	0011000R	1	86
XRL A, REG	11011REG	1	86
XRL A, @R	1101000R	1	86
XRL A, #DATA	11010011	2	87
	DDDDDDDD		

* — Только для микросхем КР1830ВЕ35, КР1830ВЕ48.

Команды можно условно разделить на следующие группы:

- команды работы с аккумулятором;
- команды ввода-вывода;
- команды работы с регистрами;
- команды перехода;
- команды обращения к подпрограмме;
- команды работы с флагами;
- команды пересылки данных;
- команды работы с таймером/счетчиком;
- команды управления.

Прохождение данных между узлами микроЭВМ при выполнении команд показано на рис. 1.25.

Группа команд работы с аккумулятором включает команды сложения и логические команды (И, ИЛИ, исключающее ИЛИ), с помощью которых производятся операции сложения и логические операции над содержимым аккумулятора и вторым операндом. В качестве второго операнда могут использоваться данные регистров общего назначения, содержимое внутренней памяти данных и непосредственные данные. Кроме этого, в команды работы с аккумулятором включены: сдвиг вправо и влево, увеличение и уменьшение на 1, дополнение, очистка, обмен ниблами (полубайтами). Арифметические операции и операции сдвига могут также производиться с использованием переноса C. Для задач, связанных с обработкой двоично-десятичных чисел, имеется команда десятичной коррекции содержимого аккумулятора.

Группа команд ввода-вывода осуществляет обмен информацией между портами микроЭВМ и аккумулятором, а также производит логические операции над содержимым портов и непосредственными данными.

Команды работы с регистрами производят две операции: инкремент и декремент содержимого регистров и инкремент содержимого внутренней памяти данных.

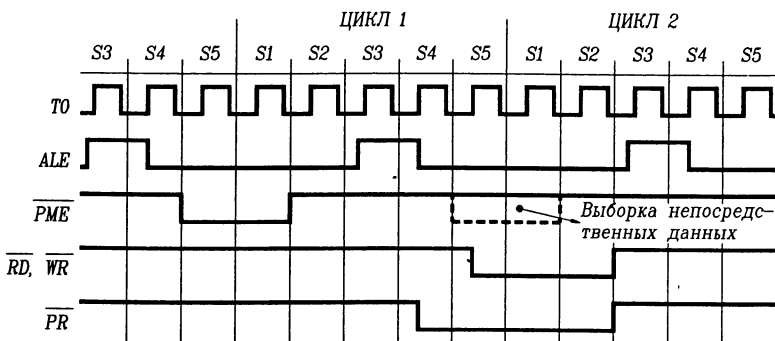
Группа команд перехода предназначена для организации ветвления программ и включает группу команд безусловных переходов и группу команд условных переходов. В качестве условий анализируются входные сигналы T0, T1, INT; состояние флагов F0, F1, флага таймера/счетчика, триггера переноса, содержимого аккумулятора и отдельных его разрядов.

ЦИКЛ 1					
КОМАНДЫ	S1	S2	S3	S4	S5
IN A, P	Выборка команды	Инкрем. СК	—	Инкремент Т/С	—
OUTL P, A	Выборка команды	Инкрем. СК	—	Инкремент Т/С	Вывод на порт
ANL P, #DATA	Выборка команды	Инкрем. СК	—	Инкремент Т/С	Чтение порта
ORL P, #DATA	Выборка команды	Инкрем. СК	—	Инкремент Т/С	Чтение порта
INS A, BUS	Выборка команды	Инкрем. СК	—	Инкремент Т/С	—
OUTL BUS, A	Выборка команды	Инкрем. СК	—	Инкремент Т/С	Вывод на порт
ANL BUS, #DATA	Выборка команды	Инкрем. СК	—	Инкремент Т/С	Чтение порта
ORL BUS, #DATA	Выборка команды	Инкрем. СК	—	Инкремент Т/С	Чтение порта
MOVX @R, A	Выборка команды	Инкрем. СК	Вывод адреса вн. ОЗУ данных	Инкремент Т/С	Вывод данных на вн. ОЗУ
MOVX A, @R	Выборка команды	Инкрем. СК	Вывод адреса вн. ОЗУ данных	Инкремент Т/С	—
MOVD A, Pi	Выборка команды	Инкрем. СК	Выдача КОП и адр. для доп. порта	Инкремент Т/С	—
MOVD Pi, A	Выборка команды	Инкрем. СК	Выдача КОП и адр. для доп. порта	Инкремент Т/С	Вывод данных на P2(0-3)
ANLD P, A	Выборка команды	Инкрем. СК	Выдача КОП и адр. для доп. порта	Инкремент Т/С	Вывод данных
ORLD P, A	Выборка команды	Инкрем. СК	Выдача КОП и адр. для доп. порта	Инкремент Т/С	Вывод данных
J(условия перехода)	Выборка команды	Инкрем. СК	Проверка условия перехода	Инкремент по условию	—
STRT T STRT CNT	Выборка команды	Инкрем. СК	—	—	Старт счетчика
STOP TCNT	Выборка команды	Инкрем. СК	—	—	Стоп счетчика
EN I	Выборка команды	Инкрем. СК	—	Разрешение прерывания	—
DIS I	Выборка команды	Инкрем. СК	—	Запрет прерывания	—
ENT0 CLK	Выборка команды	Инкрем. СК	—	Разреш. отсчета вн. тактов	—

Примечания. 1. Принятые сокращения: Т/С — таймер-счетчик, СК — счетчик команд.
2. При выполнении всех команд в состоянии S4 цикла 1 выполняется инкремент Т/С, если работа Т/С разрешена.

Рис. 1.24. Диаграмма выполнения

ЦИКЛ 2					
КОМАНДЫ	S1	S2	S3	S4	S5
IN A, P	—	Чтение порта	—	—	—
OUTL P, A	—	—	—	—	—
ANL P, #DATA	Выборка непоср. данных	—	Инкрем. СК	Вывод на порт	—
ORL P, #DATA	Выборка непоср. данных	—	Инкрем. СК	Вывод на порт	—
INS A, BUS	—	Чтение порта	—	—	—
OUTL BUS, A	—	—	—	—	—
ANL BUS, #DATA	Выборка непоср. данных	—	Инкрем. СК	Вывод на порт	—
ORL BUS, #DATA	Выборка непоср. данных	—	Инкрем. СК	Вывод на порт	—
MOVX @R, A	—	—	—	—	—
MOVX A, @R	—	Чтение данных	—	—	—
MOVD A, Pi	—	Чтение P2(0-3)	—	—	—
MOVD Pi, A	—	—	—	—	—
ANLD P, A	—	—	—	—	—
ORLD P, A	—	—	—	—	—
J(условия перехода)	Выборка непоср. данных	—	Обновление СК	—	—



команд ввода-вывода и команд перехода

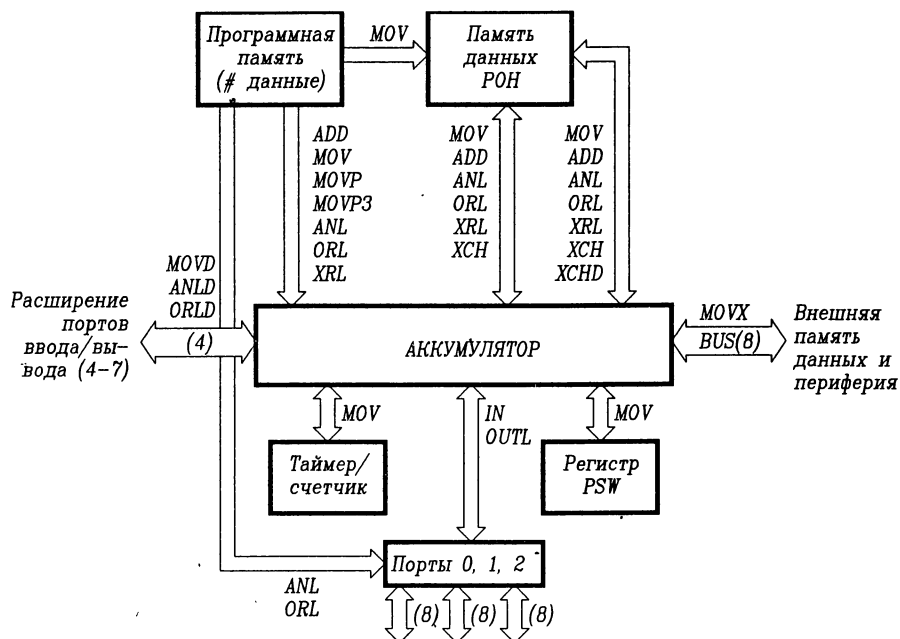


Рис. 1.25. Прохождение данных между узлами микроЭВМ при выполнении команд

Команды обращения к подпрограмме включают в себя: переход к подпрограмме и две команды выхода из подпрограммы, причем одна из них осуществляет выход с восстановлением содержимого регистра состояния программы (PSW). Эту команду рекомендуется использовать для организации выхода из подпрограммы обработки прерываний. При использовании команды RET не происходит снятия блокировки со входа схемы обработки режима прерывания.

Группа команд работы с флагами включает два типа команд: установка в нуль флагов F0, FI, C и установка их инверсии.

Группа команд пересылки данных осуществляет передачу данных между аккумулятором, регистрами общего назначения и внутренней памятью данных, передачу непосредственных данных в вышеперечисленные элементы, передачу данных между аккумулятором и регистром состояния программы (PSW), пересылку данных между аккумулятором и внешней памятью данных, обмен данными между аккумулятором и регистрами общего назначения или внутренней памятью данных, а также пересылку данных из программной памяти в аккумулятор.

В группе команд работы с таймером/счетчиком имеется две команды пересылки данных между таймером/счетчиком и аккумулятором, три команды запуска и остановки таймера/счетчика, четыре команды разрешения и запрета прерывания по таймеру/счетчику.

Команды управления включают команды разрешения и запрета внешнего прерывания, команды по выбору банков РОН и программной памяти, команду вывода тактовых импульсов на выход T0 и команду "Нет операции".

Диаграмма выполнения команд ввода-вывода и команд перехода показана на рис. 1.24.

Перечень команд микроЭВМ, систематизированных по мнемосодам, приведен в табл. 1.5.

Перечень команд, систематизированных по кодам операций — в табл. 1.6.

Коды команд в таблицах 1.5 и 1.6 приведены в шестнадцатеричном формате.

Таблица 1.5.

МНЕМОНИКА	КОД	МНЕМОНИКА	КОД	МНЕМОНИКА	КОД
ADD A, R0	68	CALL ADDRESS		IN A, P1	09
A, R1	69	(PAGE 0)	14	A, P2	0A
A, R2	6A	(PAGE 1)	34	INC A	17
A, R3	6B	(PAGE 2)	54	R0	18
A, R4	6C	(PAGE 3)	74	R1	19
A, R5	6D	(PAGE 4)	94	R2	1A
A, R6	6E	(PAGE 5)	B4	R3	1B
A, R7	6F	(PAGE 6)	D4	R4	1C
A, @R0	60	(PAGE 7)	F4	R5	1D
A, @R1	61	CLR A	27	R6	1E
A, #DATA	03	C	97	R7	1F
ADDC A, R0	78	F0	85	@R0	10
A, R1	79	F1	A5	@R1	11
A, R2	7A	CPL A	37	INS A, BUS	08
A, R3	7B	C	A7	JB0 ADDR	12
A, R4	7C	F0	95	JB1 ADDR	32
A, R5	7D	F1	B5	JB2 ADDR	52
A, R6	7E	DA A	57	JB3 ADDR	72
A, R7	7F	DEC A	07	JB4 ADDR	92
A, @R0	70	R0	C8	JB5 ADDR	B2
A, @R1	71	R1	C9	JB6 ADDR	D2
A, #DATA	13	R2	CA	JB7 ADDR	F2
ANL A, R0	58	R3	CB	JC ADDR	F6
A, R1	59	R4	CC	JF0 ADDR	B6
A, R2	5A	R5	CD	JF1 ADDR	76
A, R3	5B	R6	CE	JMP ADDR	
A, R4	5C	R7	CF	(PAGE 0)	04
A, R5	5D	DIS I	15	(PAGE 1)	24
A, R6	5E	TCNTI	35	(PAGE 2)	44
A, R7	5F	DJNZ R0, ADDR	E8	(PAGE 3)	64
A, @R0	50	R1, ADDR	E9	(PAGE 4)	84
A, @R1	51	R2, ADDR	EA	(PAGE 5)	A4
A, #DATA	53	R3, ADDR	EB	(PAGE 6)	C4
BUS, #DATA	98	R4, ADDR	EC	(PAGE 7)	E4
P1, #DATA	99	R5, ADDR	ED	JMPP @A	B3
P2, #DATA	9A	R6, ADDR	EE	JNC ADDR	E6
ANLD P4, A	9C	R7, ADDR	EF	JNI ADDR	86
P5, A	9D	EN I	05	JNT0 ADDR	26
P6, A	9E	TCNTI	25	JNT1 ADDR	46
P7, A	9F	ENT0 CLK	75	JNZ ADDR	96
		*IDL	01		

Продолжение таблицы 1.5.

МНЕМОНИКА	КОД	МНЕМОНИКА	КОД	МНЕМОНИКА	КОД
JTF ADDR	16	A, P6	0E	MB1	F5
JT0 ADDR	36	A, P7	0F	RB0	C5
JT1 ADDR	56	P4, A	3C	RB1	D5
JZ ADDR	C6	P5, A	3D	STOP TCNT	65
MOV A, #DATA	23	P6, A	3E	STRT CNT	45
A, PSW	C7	P7, A	3F	T	55
A, R0	F8	MOVP A, @A	A3	SWAP A	47
A, R1	F9	MOVP3 A, @A	E3	XCH A, R0	28
A, R2	FA	MOVX A, @R0	80	A, R1	29
A, R3	FB	A, @R1	81	A, R2	2A
A, R4	FC	@R0, A	90	A, R3	2B
A, R5	FD	@R1, A	91	A, R4	2C
A, R6	FE	NOP	00	A, R5	2D
A, R7	FF	ORL A, R0	48	A, R6	2E
A, @R0	F0	A, R1	49	A, R7	2F
A, @R1	F1	A, R2	4A	A, @R0	20
A, T	42	A, R3	4B	A, @R1	21
PSW, A	D7	A, R4	4C	XCHD A, @R0	30
R0, A	A8	A, R5	4D	A, @R1	31
R1, A	A9	A, R6	4E	XRL A, R0	D8
R2, A	AA	A, R7	4F	A, R1	D9
R3, A	AB	A, @R0	40	A, R2	DA
R4, A	AC	A, @R1	41	A, R3	DB
R5, A	AD	A, #DATA	43	A, R4	DC
R6, A	AE	BUS, #DATA	88	A, R5	DD
R7, A	AF	P1, #DATA	89	A, R6	DE
R0, #DATA	B8	P2, #DATA	8A	A, R7	DF
R1, #DATA	B9	ORLD P4, A	8C	A, @R0	D0
R2, #DATA	BA	P5, A	8D	A, @R1	D1
R3, #DATA	BB	P6, A	8E	A, #DATA	D3
R4, #DATA	BC	P7, A	8F		
R5, #DATA	BD	OUTL BUS, A	02		
R6, #DATA	BE	P1, A	39		
R7, #DATA	BF	P2, A	3A		
@R0, A	A0	RET	83		
@R1, A	A1	RETR	93		
@R0, #DATA	B0	RL A	E7		
@R1, #DATA	B1	RLC A	F7		
T, A	62	RR A	77		
MOVD A, P4	0C	RRC A	67		
A, P5	0D	SEL MB0	E5		

Таблица 1.6

КОД	МНЕМОНИКА	КОД	МНЕМОНИКА	КОД	МНЕМОНИКА
00	NOP	2B	XCH A, R3	56	JT1 ADDR
01	*IDL	2C	XCH A, R4	57	DA A
02	OUTL BUS, A	2D	XCH A, R5	58	ANL A, R0
03	ADD A, #DATA	2E	XCH A, R6	59	ANL A, R1
04	JMP (PAGE 0)	2F	XCH A, R7	5A	ANL A, R2
05	EN I	30	XCHD A, @R0	5B	ANL A, R3
06	—	31	XCHD A, @R1	5C	ANL A, R4
07	DEC A	32	JB1 ADDR	5D	ANL A, R5
08	INS A, BUS	33	—	5E	ANL A, R6
09	IN A, P1	34	CALL (PAGE 1)	5F	ANL A, R7
0A	IN A, P2	35	DIS TCNTI	60	ADD A, @R0
0B	—	36	JT0 ADDR	61	ADD A, @R1
0C	MOVD A, P4	37	CPL A	62	MOV T, A
0D	MOVD A, P5	38	—	63	—
0E	MOVD A, P6	39	OUTL P1, A	64	JMP (PAGE 3)
0F	MOVD A, P7	3A	OUTL P2, A	65	STOP TCNT
10	INC @R0	3B	—	66	—
11	INC @R1	3C	MOVD P4, A	67	RRC A
12	JB0 ADDR	3D	MOVD P5, A	68	ADD A, R0
13	ADDC A, #DATA	3E	MOVD P6, A	69	ADD A, R1
14	CALL (PAGE 0)	3F	MOVD P7, A	6A	ADD A, R2
15	DIS I	40	ORL A, @R0	6B	ADD A, R3
16	JTF ADDR	41	ORL A, @R1	6C	ADD A, R4
17	INC A	42	MOV A, T	6D	ADD A, R5
18	INC R0	43	ORL A, #DATA	6E	ADD A, R6
19	INC R1	44	JMP (PAGE 2)	6F	ADD A, R7
1A	INC R2	45	STRT CNT	70	ADDC A, @R0
1B	INC R3	46	JNT1 ADDR	71	ADDC A, @R1
1C	INC R4	47	SWAP A	72	JB3 ADDR
1D	INC R5	48	ORL A, R0	73	—
1E	INC R6	49	ORL A, R1	74	CALL (PAGE 3)
1F	INC R7	4A	ORL A, R2	75	ENT0 CLK
20	XCH A, @R0	4B	ORL A, R3	76	JF1 ADDR
21	XCH A, @R1	4C	ORL A, R4	77	RR A
22	—	4D	ORL A, R5	78	ADDC A, R0
23	MOV A, #DATA	4E	ORL A, R6	79	ADDC A, R1
24	JMP (PAGE 1)	4F	ORL A, R7	7A	ADDC A, R2
25	EN TCNTI	50	ANL A, @R0	7B	ADDC A, R3
26	JNT0 ADDR	51	ANL A, @R1	7C	ADDC A, R4
27	CLR A	52	JB2 ADDR	7D	ADDC A, R5
28	XCH A, R0	53	ANL A, #DATA	7E	ADDC A, R6
29	XCH A, R1	54	CALL (PAGE 2)	7F	ADDC A, R7
2A	XCH A, R2	55	STRT T	80	MOVX A, @R0

Продолжение таблицы 1.6

КОД	МНЕМОНИКА	КОД	МНЕМОНИКА	КОД	МНЕМОНИКА
81	MOVX A, @R1	AA	MOV R2, A	D3	XRL A, #DATA
82	—	AB	MOV R3, A	D4	CALL (PAGE 6)
83	RET	AC	MOV R4, A	D5	SEL RB1
84	JMP (PAGE 4)	AD	MOV R5, A	D6	—
85	CLR F0	AE	MOV R6, A	D7	MOV PSW, A
86	JNI ADDR	AF	MOV R7, A	D8	XRL A, R0
87	—	B0	MOV @R0, #DATA	D9	XRL A, R1
88	ORL BUS, #DATA	B1	MOV @R1, #DATA	DA	XRL A, R2
89	ORL P1, #DATA	B2	JB5 ADDR	DB	XRL A, R3
8A	ORL P2, #DATA	B3	JMPP @A	DC	XRL A, R4
8B	—	B4	CALL (PAGE 5)	DD	XRL A, R5
8C	ORLD P4, A	B5	CPL F1	DE	XRL A, R6
8D	ORLD P5, A	B6	JF0 ADDR	DF	XRL A, R7
8E	ORLD P6, A	B7	—	E0	—
8F	ORLD P7, A	B8	MOV R0, #DATA	E1	—
90	MOVX @R0, A	B9	MOV R1, #DATA	E2	—
91	MOVX @R1, A	BA	MOV R2, #DATA	E3	MOV P3 A, @A
92	JB4 ADDR	BB	MOV R3, #DATA	E4	JMP (PAGE 7)
93	RETR	BC	MOV R4, #DATA	E5	SEL MB0
94	CALL (PAGE 4)	BD	MOV R5, #DATA	E6	JNC ADDR
95	CPL F0	BE	MOV R6, #DATA	E7	RL A
96	JNZ ADDR	BF	MOV R7, #DATA	E8	DJNZ R0, ADDR
97	CLR C	C0	—	E9	DJNZ R1, ADDR
98	ANL BUS, #DATA	C1	—	EA	DJNZ R2, ADDR
99	ANL P1, #DATA	C2	—	EB	DJNZ R3, ADDR
9A	ANL P2, #DATA	C3	—	EC	DJNZ R4, ADDR
9B	—	C4	JMP (PAGE 6)	ED	DJNZ R5, ADDR
9C	ANLD P4, A	C5	SEL RB0	EE	DJNZ R6, ADDR
9D	ANLD P5, A	C6	JZ ADDR	EF	DJNZ R7, ADDR
9E	ANLD P6, A	C7	MOV A, PSW	F0	MOV A, @R0
9F	ANLD P7, A	C8	DEC R0	F1	MOV A, @R1
A0	MOV @R0, A	C9	DEC R1	F2	JB7 ADDR
A1	MOV @R1, A	CA	DEC R2	F3	—
A2	—	CB	DEC R3	F4	CALL (PAGE 7)
A3	MOV P A, @A	CC	DEC R4	F5	SEL MB1
A4	JMP (PAGE 5)	CD	DEC R5	F6	JC ADDR
A5	CLR F1	CE	DEC R6	F7	RLC A
A6	—	CF	DEC R7	F8	MOV A, R0
A7	CPL C	D0	XRL A, @R0	F9	MOV A, R1
A8	MOV R0, A	D1	XRL A, @R1		
A9	MOV R1, A	D2	JB6 ADDR		

* Только для микросхем КР1830ВЕ35, КР1830ВЕ48.

1.5. Описание машинных команд

Описание каждой машинной команды состоит из формата, кода, алгоритма и времени ее выполнения.

В части "Формат" приведено содержимое полей "Мнемокод" и "Аргументы" машинной команды при ее записи в формате предложения языка ассемблера.

В части "Код" приведен двоичный код машинной команды.

В части "Алгоритм" приведено символическое описание операции, выполняемое машинной командой. При описании операции используются обозначения, приведенные в таблице 1.7.

Таблица 1.7

Обозначение	Операция
(X)	Содержимое элемента X
((X))	Содержимое по адресу, хранящемуся в элементе X
X[M]	Разряд M элемента X
X[M1-M2]	Группа разрядов M1-M2 элемента X
+	Операции: сложения
-	вычитания
AND	логического умножения
OR	логического сложения
XOR	сложения по модулю 2
NOT	инверсии
>	больше
=	равно
<>	неравно
X <= Y	пересылки содержимого элемента Y в элемент X
X <=> Y	обмена содержимых элементов X и Y
X : Y	присоединения старшей части, хранящейся в элементе X, к младшей части, хранящейся в элементе Y
если... ,то...	Операторы: Условные
если... ,то...	
иначе...	
для N от... до...	цикла
TMP	Промежуточный регистр
BUS	Порт PØ (DB)
SP	Указатель стека
PC	Счетчик команд
В	Код: двоичный
Н	шестнадцатеричный

В части "Время" приведено количество циклов ОМЭВМ, требуемых для выполнения машинной команды. Время цикла определяется по следующей формуле: $t_{cy} = 15/f_{BQ1}$, t_{cy} — время цикла, мкс; f_{BQ1} — частота генератора тактовых сигналов ОМЭВМ, МГц.

Команда ADD A, <Reg>

По команде ADD A, <Reg> содержимое рабочего регистра <Reg> складывается с содержимым аккумулятора A. Результат вычисления записывается в аккумулятор.

При переносе из 3-го разряда АЛУ триггер признака вспомогательного переноса АС устанавливается в состояние "1", при отсутствии переноса — в состояние "0". При переносе из 7-го разряда АЛУ триггер признака переноса С устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

Формат: ADD A, <Reg>

имя рабочего регистра R0...R7

Код:

0 1 1 0 1	R e g
-----------	-------

код рабочего регистра 000B...111B

Алгоритм: (AC): (TMP[0-3]) <= (A[0-3])+(Reg[0-3])
(C): (A) <= (A)+(Reg)

Время: 1 цикл

Пример: МЕТКА: ADD A, R0 ; (A) <= (A)+(R0)

Команда ADD A, @<R>

По команде ADD A, @<R> содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, складывается с содержимым аккумулятора A. Результат вычисления записывается в аккумулятор.

При переносе из 3-го разряда АЛУ триггер признака вспомогательного переноса АС устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

При переносе из 7-го разряда АЛУ триггер признака переноса С устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

Формат: ADD A, @<R>

имя рабочего регистра R0...R1

Код:

0 1 1 0 0 0 0	R
---------------	---

код рабочего регистра 0B...1B

Алгоритм: (AC): (TMP[0-3]) <= (A[0-3]) <= A[0-3]+((R)[0-3])
(C): (A) <= (A)+((R))

Время: 1 цикл

Команда ADD A, #<DATA>

По команде ADD A, #<DATA> непосредственные данные, определяемые элементом <DATA>, складываются с содержимым аккумулятора. Результат вычисления записывается в аккумулятор.

При переносе из 3-го разряда АЛУ триггер признака вспомогательного переноса АС устанавливается в состояние "1", при отсутствии переноса — состояние "0".

При переносе из 7-го разряда АЛУ триггер признака переноса С устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

Формат: ADD A, #<DATA>

выражение для непосредственных данных

Код:

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

DATA

|
непосредственные данные 00H...FFH

Алгоритм: (AC): (TMP[0-3]) <= (A[0-3])+DATA[0-3]
(C): (A) <= (A)+DATA

Время: 2 цикла

Пример: CONST EQU 17 ; CONST=11H

NEXT: MOV A, #15	; (A) <= 0FH
ADD A, #CONST	; (A) <= 0FH+11H=20H
	; (AC) <= 1, (C) <= 0

Команда ADDC A, <Reg>

По команде ADDC A, <Reg> содержимое триггера признака переноса C складывается с содержимым аккумулятора A. Затем к содержимому аккумулятора прибавляется содержимое рабочего регистра <Reg>. Результат вычисления записывается в аккумулятор.

При переносе из 3-го разряд АЛУ в случае выполнения первой или второй операции сложения триггер признака вспомогательного переноса AC устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

При переносе из 7-го разряд АЛУ в случае выполнения первой или второй операции сложения триггер признака переноса C устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

Формат: ADDC A, <Reg>

|
имя рабочего регистра R0...R7

Код:

0	1	1	1	1	Reg
---	---	---	---	---	-----

|
код рабочего регистра 000B...111B

Алгоритм: (AC): (TMP[0-3]) <= (A[0-3])+(REG[0-3])+(C)
(C): (A) <= (A)+(Reg)+(C)

Время: 1 цикл

Команда ADDC A, @<R>

По команде ADDC A, @<R> содержимое триггера признака переноса C складывается с содержимым аккумулятора A. Затем к содержимому аккумулятора прибавляется содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>. Результат вычислений записывается в аккумулятор.

При переносе из 3-го разряда АЛУ в случае выполнения первой или второй операции сложения триггер признака вспомогательного переноса AC устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

При переносе из 7-го разряда АЛУ в случае выполнения первой или второй операции сложения триггер признака переноса C устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

Формат: ADDC A, @<R>

|
имя рабочего регистра R0...R1

Код:

0 1 1 1 0 0 0	R
---------------	---

код рабочего регистра 0B...1B

Алгоритм: (AC): (TMP[0-3]) <= (A[0-3]+((R)[0-3]))+(C)
 (C): (A) <= (A)+((R))+C

Время: 1 цикл

Команда ADDC A, #<DATA>

По команде ADDC A, #<DATA> содержимое признака переноса C складывается с содержимым аккумулятора A. Затем к содержимому аккумулятора прибавляются непосредственные данные, определяемые элементом <DATA>. Результат вычисления записывается в аккумулятор.

При переносе из 3-го разряда АЛУ в случае выполнения первой или второй операции сложения триггер признака вспомогательного переноса AC устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

При переносе из 7-го разряда АЛУ в случае выполнения первой или второй операции сложения триггер признака переноса C устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

Формат: ADDC A, #<DATA>

выражение для непосредственных данных

Код:

0 0 0 1 0 0 1 1

DATA

непосредственные данные 00H...FFH

Алгоритм: (AC): (TMP[0-3]) <= (A[0-3]+DATA[0-3]+(C))
 (C): (A) <= (A)+DATA+C

Время: 2 цикла

Пример:

NEXT: ADDC A, #3 ;(A)=0FCH, (C)=1
 ;(A) <= 0FCH+03H+1=00H
 ;(C) <= 1

Команда ANL A, <Reg>

По команде ANL A, <Reg> содержимое рабочего регистра <Reg> поразрядно логически умножается на содержимое аккумулятора A. Результат вычисления записывается в аккумулятор.

Формат: ANL A, <Reg>

имя рабочего регистра R0...R7

Код:

0 1 0 1 1	Reg
-----------	-----

код рабочего регистра 000B...111B

Алгоритм: (A) <= (A) AND (REG)

Время: 1 цикл

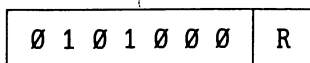
Команда ANL A,@<R>

По команде ANL A,@<R> содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, поразрядно логически умножаются на содержимое аккумулятора A. Результат вычисления записывается в аккумулятор.

Формат: ANL A,@<R>

имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

Алгоритм: (A) <= (A) AND ((R))

Время: 1 цикл

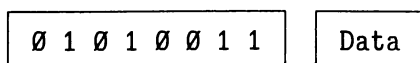
Команда ANL A,#<DATA>

По команде ANL A,#<DATA> непосредственные данные, определяемые элементом <DATA>, поразрядно логически умножаются на содержимое аккумулятора A. Результат вычисления записывается в аккумулятор.

Формат: ANL A,#<DATA>

выражение для непосредственных данных

Код:



непосредственные данные 00H...FFH

Алгоритм: (A) <= (A) AND DATA

Время: 2 цикла

Пример: ;(A)=00001111B

МЕТКА: ANL A,#01011100B

;(A)=00001100B

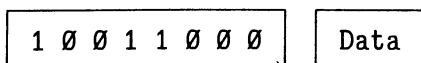
Команда ANL BUS,#<DATA>

По команде ANL BUS, #<DATA> непосредственные данные, определяемые элементом <DATA>, поразрядно логически умножаются на содержимое порта P0, обозначаемого в языке — порт BUS. Результат вычисления записывается в порт.

Формат: ANL BUS,#<DATA>

выражение для непосредственных данных

Код:



непосредственные данные 00H...FFH

Алгоритм: (BUS) <= (BUS) AND DATA

Время: 2 цикла

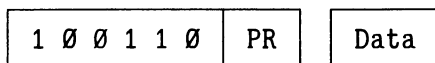
Команда ANL <PR>, #<DATA>

По команде ANL <PR>, #<DATA> непосредственные данные, определяемые элементом <DATA>, поразрядно логически умножаются на содержимое порта <PR>. Результат вычисления записывается в резидентный порт.

Формат: ANL <PR>, #<DATA>

имя порта P1...P2 выражение для непосредственных данных

Код:



код порта 01B...10B непосредственные данные 00H...FFH

Алгоритм: (PR) <= (PR) AND DATA

Время: 2 цикла

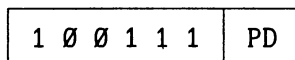
Команда ANLD <PD>, A

По команде ANLD <PD>, A содержимое дополнительного (четырёхразрядного) порта поразрядно логически умножается на содержимое 0...3 разряд аккумулятора A. Результат вычисления записывается в дополнительный порт.

Формат: ANLD <PD>, A

имя дополнительного порта P4...P7

Код:



код дополнительного порта 00B...11B

Алгоритм: (PD) <= (PD) AND (A[0-3])

Время: 2 цикла

Команда CALL <ADR 11>

По команде CALL <ADR 11> осуществляется вызов подпрограммы следующим образом:

— содержимое счетчика команд PC (то есть полный 12-разрядный адрес следующей команды) и разрядов 4...7 слова состояния PSW записывается в вершину стека;

— содержимое указателя стека SP увеличивается на единицу;

— в разряды 0...10 счетчика команд записывается длинный адрес подпрограммы, определяемый элементом <ADR 11>;

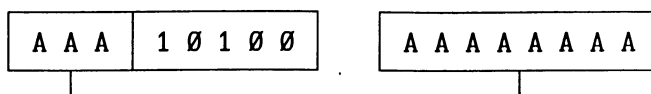
— в разряд 11 счетчика команд записывается содержимое триггера переключения банка памяти программ DBF и, таким образом, устанавливается текущий банк памяти программ.

Команда CALL не должна размещаться в ячейках 2046...2047 и 4094...4095.

Формат: CALL <ADR 11>

выражение для длинного (11-битного) адреса
ячейки банка памяти программ

Код:



|
ADR 11

|
длинный (11- битный) адрес ячейки
банка памяти программ 000H...7FFH

Алгоритм: ((SP)) <= (PC:PSW[4-7])
 (SP) <= (SP)+1
 (PC) <= ADR11
 (PC[11]) <= (DBF)

Время: 2 цикла

Пример: МЕТКА: CALL процед
 CALL процед

ПРОЦЕД: MOV R0, #5

RET

Команда CLR A

По команде CLR A все разряды аккумулятора устанавливаются в состояние "0".

Формат: CLR A

Код:

0 0 1 0 0 1 1 1

Алгоритм: (A) <= 0

Время: 1 цикл

Команда CLR C

По команде CLR C триггер признака переноса C устанавливается в состояние "0".

Формат: CLR C

Код:

1 0 0 1 0 1 1 1

Алгоритм: (C) <= 0

Время: 1 цикл

Команда CLR F0

По команде CLR F0 триггер бита условия (флага) F0 устанавливается в состояние "0".

Формат: CLR F0

Код:

1 0 0 0 0 1 0 1

Алгоритм: (F0) <= 0

Время: 1 цикл

Команда CLR F1

По команде CLR F1 триггер бита условия (флага) F1 устанавливается в состояние "1".

Формат: CLR F1

Код:

1	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: (F1) <= 0

Время: 1 цикл

Команда CPL A

По команде CPL A состояние всех разрядов аккумулятора A изменяется на противоположное.

Формат: CPL A

Код:

0	0	1	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: (A) <= NOT (A)

Время: 1 цикл

Пример: ;(A)=01010011B
 МЕТКА: CPL A ;(A)=10101100B

Команда CPL C

По команде CPL C состояние триггера признака переноса C изменяется на противоположное.

Формат: CPL C

Код:

1	0	1	0	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: (C) <= NOT (C)

Время: 1 цикл

Команда CPL F0

По команде CPL F0 состояние триггера бита условия (флага) F0 изменяется на противоположное.

Формат: CPL F0

Код:

1	0	0	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: (F0) <= NOT (F0)

Время: 1 цикл

Команда CPL F1

По команде CPL F1 состояние триггера бита условия (флага) F1 изменяется на противоположное.

Формат: CPL F1

Код:

1	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: (F1) \leftarrow NOT (F1)

Время: 1 цикл

Команда DA A

По команде DA A результат двоичного сложения упакованных двоично-десятичных чисел в аккумуляторе A преобразуется в упакованное двоично-десятичное число следующим образом. Если число в младших четырех разрядах аккумулятора больше девяти или триггер признака вспомогательного переноса AC установлен в состояние "1", то к содержимому аккумулятора прибавляется число 6; если после этого число в старших четырех разрядах аккумулятора больше девяти или триггер признака переноса C установлен в состояние "1", то к содержимому аккумулятора прибавляют число 60H.

При переносе из 7-го разряда АЛУ триггер признака переноса C устанавливается в состояние "1", в противном случае — в "0".

Формат: DA A

Код:

0	1	0	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: если (A[0-3]) > 9 OR (AC)=1, то (C): (A) \leftarrow (A)+6
если (A[4-7]) > 9 OR (C)=1, то (C): (A) \leftarrow (A)+60H

Время: 1 цикл

Пример: NUMB1 EQU 51H ; NUMB1=51H
NUMB2 EQU 19H ; NUMB2=19H
 . . . ; 51+19=70
BEGIN: MOV A, #NUMB1 ; (A) \leftarrow 51H
 ADD A, #NUMB2 ; (A) \leftarrow 51H+19H=6AH
 DA A ; (A) \leftarrow 6AH+6=70H

Команда DEC A

По команде DEC A содержимое аккумулятора A уменьшается на единицу.

Формат: DEC A

Код:

0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: (A) \leftarrow (A)-1

Время: 1 цикл

Пример: ; (A) \leftarrow 12H
 МЕТКА: DEC A ; (A) \leftarrow 12H-1=11H

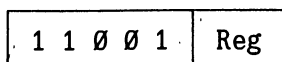
Команда DEC <Reg>

По команде DEC <Reg> содержимое рабочего регистра <Reg> уменьшается на единицу.

Формат: DEC <Reg>

имя рабочего регистра R0...R7

Код:



код рабочего регистра 000B...111B

Алгоритм: ($\langle \text{Reg} \rangle$) \leftarrow ($\langle \text{Reg} \rangle$) - 1

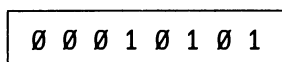
Время: 1 цикл

Команда DIS I

По команде DIS I триггер разрешения внешних прерываний I устанавливается в состояние "0", тем самым запрещаются внешние прерывания по сигналу на выводе INT.

Формат: DIS I

Код:

Алгоритм: (I) \leftarrow 0

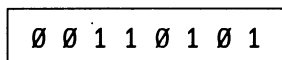
Время: 1 цикл

Команда DIS TCNTI

По команде DIS TCNTI триггер разрешения внутренних прерываний TCNTI устанавливается в состояние "0", тем самым запрещаются внутренние прерывания от таймера/счетчика.

Формат: DIS TCNTI

Код:

Алгоритм: (TCNTI) \leftarrow 0

Время: 1 цикл

Команда DJNZ $\langle \text{Reg} \rangle$, $\langle \text{ADR8} \rangle$

По команде DJNZ $\langle \text{Reg} \rangle$, $\langle \text{ADR8} \rangle$ осуществляется ветвление в программе следующим образом:

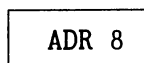
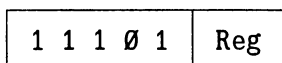
- содержимое рабочего регистра $\langle \text{Reg} \rangle$ уменьшается на единицу;
- если содержимое рабочего регистра $\langle \text{Reg} \rangle$ не равно нулю, то управление передается в точку назначения, адрес которой определен элементом $\langle \text{ADR8} \rangle$, в противном случае осуществляется переход к следующей команде.

Если код первого байта команды DJNZ $\langle \text{Reg} \rangle$, $\langle \text{ADR8} \rangle$ размещается в предпоследней или в последней ячейках страницы памяти программ, то точка назначения такой команды должна находиться на следующей странице. Данное замечание справедливо для всех команд условного перехода.

Формат: DJNZ $\langle \text{Reg} \rangle$, $\langle \text{ADR8} \rangle$

выражение для короткого (8-битного)
адреса памяти программ
имя рабочего регистра R0...R7

Код:

код рабочего регистра
000B...111Bкороткий (8-битный) адрес
ячейки внутри страницы
памяти программ 00H-0FFH

Алгоритм: $(REG) \leq (REG) - 1$
 если $(REG) \neq 0$,
 то $(PC[0-7]) \leq ADR8$,
 иначе $(PC) \leq (PC) + 2$

Время: 2 цикла

Пример: MOV R0, #50 ; (R0) <= 50
 MOV R3, #6 ; (R3) <= 6
 ЦИКЛ: INC @R0 ; ((R0)) <= ((R0)) + 1
 INC R0 ; (R0) <= R0 + 1
 DJNZ R3, ЦИКЛ ; (R3) <= (R3) - 1
 ; если (R3) < 0, то перейти на
 ; метку "цикл"

Команда EN I

По команде EN I триггер разрешения внешних прерываний "I" устанавливается в состояние "1", тем самым разрешаются прерывания по сигналу на выводе INT.

Формат: EN I

Код:

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: $(I) \leq 1$

Время: 1 цикл

Команда EN TCNTI

По команде EN TCNTI триггер разрешения внутренних прерываний TCNTI устанавливается в состояние "1", тем самым разрешаются прерывания от таймера/счетчика (то есть по переносу из 7-го разряда таймера/счетчика управление будет передано подпрограмме обслуживания прерывания с начальным адресом 007H).

Формат: EN TCNTI

Код:

0	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: $(TCNTI) \leq 1$

Время: 1 цикл

Команда ENT0 CLK

По команде ENT0 CLK генератор тактовых сигналов через делитель 1:3 подключается к выводу T0 (вывод T0 становится выходным).

Отключение генератора тактовых сигналов от вывода T0 осуществляется по сигналу системного сброса.

Формат: ENT0 CLK

Код:

0	1	1	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: подключить генератор тактовых сигналов к выводу T0

Время: 1 цикл

Команда IDL

Команда IDL имеется только в системе команд микросхем КР1830ВЕ35 и КР1830ВЕ48.

По команде IDL устанавливается режим микропотребления. При этом выполнение программы прекращается, внутренние фазовые и тактовые сигналы не формируются, сохраняется состояние всех функциональных узлов микросхемы (ОЗУ, портов, аккумулятора, PSW и др.), т. е. получается "спящий режим".

Выход из состояния микропотребления осуществляется с помощью сигнала \overline{SR} или сигнала прерывания \overline{INT} (подача уровня "0").

Формат: IDL

Код:

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

Время: 1 цикл

Команда IN A, <PR>

По команде IN A, <PR> содержимое порта <PR> пересылается в аккумулятор A.

Формат: IN A, <PR>

имя порта P1...P2

Код:

0	0	0	0	1	0	Pr
---	---	---	---	---	---	----

код порта 01B...10B

Алгоритм: (A) <= (PR)

Время: 2 цикла

Команда INC A

По команде INC A содержимое аккумулятора A увеличивается на единицу.

Формат: INC A

Код:

0	0	0	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: (A) <= (A)+1

Время: 1 цикл

Команда INC <Reg>

По команде INC <Reg> содержимое рабочего регистра <Reg> увеличивается на единицу.

Формат: INC <Reg>

имя рабочего регистра R0...R7

Код:

0	0	0	1	1	Reg
---	---	---	---	---	-----

код рабочего регистра 000B...111B

Алгоритм: (<Reg>) <= (<Reg>)+1

Время: 1 цикл

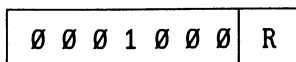
Команда INC @<R>

По команде INC @<R> содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, увеличивается на единицу.

Формат: INC @<R>

имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

Алгоритм: ((R)) <= ((R))+1

Время: 1 цикл

Пример: ;(R1)=40H, (40H)=05H
 МЕТКА: INC @R1 ;(40H) <= 05H+1=06H

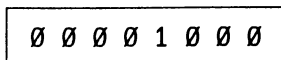
Команда INS A, BUS

По команде INS A, BUS содержимое порта BUS пересылается в аккумулятор A.

Пересылка данных сопровождается выдачей строга на вывод \overline{RD} .

Формат: INS A, BUS

Код:



Алгоритм: (A) <= (BUS)

Время: 2 цикла

Команда JB <ADR8>

По команде JB <ADR8> осуществляется ветвление в программе следующим образом:

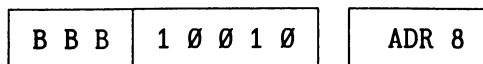
— если разряд аккумулятора установлен в состояние "1", то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JB <ADR8>

выражение для короткого (8-битного) адреса
 внутри страницы памяти программ
 номер разряда аккумулятора: 0 - A[0], 1 - A[1],
 ..., 7 - A[7]

Код:



код номера разряда
 аккумулятора: 000B-A[0],
 001B-A[1], ..., 111B-A[7]

короткий (8-битный) адрес
 ячейки внутри страницы
 памяти программ 00H-0FFH

Алгоритм: если $(A[BBB])=1$,
 то $(PC[0-7]) \leq ADR8$,
 иначе $(PC) \leq (PC)+2$

Время: 2 цикла

Пример: IN A,P1 ; (A) \leq (P1)
 JB4 CONT ; если $(A[4])=1$, то перейти
 ; на метку "CONT"

Команда JC <ADR8>

По команде JC <ADR8> осуществляется ветвление в программе следующим образом:

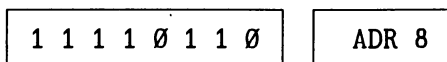
— если триггер признака переноса C установлен в состояние "1", то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JC <ADR8>

выражение для короткого (8-битного) адреса
 внутри страницы памяти программ

Код:



короткий (8-битный) адрес ячейки внутри
 страницы памяти программ 00H-0FFH

Алгоритм: если $(C)=1$,
 то $(PC[0-7]) \leq ADR8$,
 иначе $(PC) \leq (PC)+2$

Время: 2 цикла

Команда JF0 <ADR8>

По команде JF0 <ADR8> осуществляется ветвление в программе следующим образом:

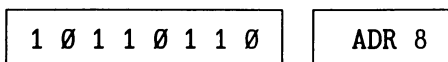
— если триггер бита условия (флага) F0 установлен в состояние "1", то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JF0 <ADR8>

выражение для короткого (8-битного) адреса
 внутри страницы памяти программ

Код:



короткий (8-битный) адрес ячейки внутри
 страницы памяти программ 00H-0FFH

Алгоритм: если $(F0)=1$,
 то $(PC[0-7]) \leq ADR8$,
 иначе $(PC) \leq (PC)+2$

Время: 2 цикла

Команда JF1 <ADR8>

По команде JF1 <ADR8> осуществляется ветвление в программе следующим образом:

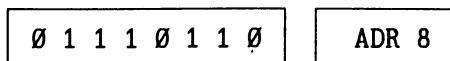
— если триггер бита условия (флага) F1 установлен в состояние "1", то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JF1 <ADR8>

выражение для короткого (8-битного) адреса
 внутри страницы памяти программ

Код:



короткий (8-битный) адрес ячейки внутри
 страницы памяти программ 00H-0FFH

Алгоритм: если $(F1)=1$,
 то $(PC[0-7]) \leq ADR8$,
 иначе $(PC) \leq (PC)+2$

Время: 2 цикла

Команда JMP <ADR11>

По команде JMP <ADR11> осуществляется безусловный прямой переход в точку назначения, адрес которой определен элементом <ADR11> (в разряды 0...10 счетчика команд PC записывается длинный адрес точки назначения, указанный в коде команды, в разряд 11 — содержимое триггера переключения банка памяти программ DBF).

Формат: JMP <ADR11>

выражение для длинного (11-битного) адреса
 ячейки банка памяти программ

Код:



ADR 11

длинный (11-битный) адрес ячейки
 банка памяти программ 000H...7FFH

Алгоритм: $(PC[0-10]) \leq ADR11$
 $(PC[11]) \leq (DBF)$

Время: 2 цикла

Пример: МЕТКА: JMP +5 ;перейти на +5
 JMP МЕТ ;перейти на "МЕТ"

Команда JMPP @A

По команде JMPP @A осуществляется безусловный косвенный переход в точку назначения, адрес которой содержится в ячейке текущей страницы памяти программ, адресуемой аккумулятором А (в разряды 0...7 счетчика команд РС записывается содержимое ячейки памяти программ, адрес которой указан в аккумуляторе).

Формат: JMPP @A

Код:

1	0	1	1	0	0	1	1
---	---	---	---	---	---	---	---

Алгоритм: $(PC[0-7]) \leftarrow ((A))$

Время: 2 цикла

Команда JNC <ADR8>

По команде JNC <ADR8> осуществляется ветвление в программе следующим образом:

— если триггер признака переноса С установлен в состояние "0", то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд РС записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JNC <ADR8>

|
выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

1	1	1	0	0	1	1	0
---	---	---	---	---	---	---	---

ADR 8

|
короткий (8-битный) адрес ячейки внутри
страницы памяти программ 00H-0FFH

Алгоритм: если $(C)=0$,
 то $(PC[0-7]) \leftarrow \text{ADR8}$,
 иначе $(PC) \leftarrow (PC)+2$

Время: 2 цикла

Команда JN1 <ADR8>

По команде JN1 <ADR8> осуществляется ветвление в программе следующим образом:

— если сигнал на выводе \overline{INT} равен нулю, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд РС записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JN1 <ADR8>

|
выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

1 0 0 0 0 1 1 0

ADR 8

короткий (8-битный) адрес ячейки внутри
страницы памяти программ 00H-0FFH

Алгоритм: если $(\overline{INT})=0$,
то $(PC[0-7]) \leq ADR8$,
иначе $(PC) \leq (PC)+2$

Время: 2 цикла

Команда JNT0 <ADR8>

По команде JNT0 <ADR8> осуществляется ветвление в программе следующим образом:

— если сигнал на выводе T0 равен нулю, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JNT0 <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

0 0 1 0 0 1 1 0

ADR 8

короткий (8-битный) адрес ячейки внутри
страницы памяти программ 00H-0FFH

Алгоритм: если $(T0)=0$,
то $(PC[0-7]) \leq ADR8$,
иначе $(PC) \leq (PC)+2$

Время: 2 цикла

Команда JNT1 <ADR8>

По команде JNT1 <ADR8> осуществляется ветвление в программе следующим образом:

— если сигнал на выводе T1 равен нулю, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JNT1 <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

0 1 0 0 0 1 1 0

ADR 8

короткий (8-битный) адрес ячейки внутри
страницы памяти программ 00H-0FFH

Алгоритм: если $(T1)=0$,
 то $(PC[0-7]) \leq ADR8$,
 иначе $(PC) \leq (PC)+2$

Время: 2 цикла

Команда JNZ <ADR8>

По команде JNZ <ADR8> осуществляется ветвление в программе следующим образом:

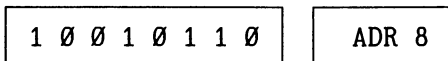
— если содержимое аккумулятора не равно нулю, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JNZ <ADR8>

выражение для короткого (8-битного) адреса
 внутри страницы памяти программ

Код:



короткий (8-битный) адрес ячейки внутри
 страницы памяти программ 00H-0FFH

Алгоритм: если $(A) \neq 0$,
 то $(PC[0-7]) \leq ADR8$,
 иначе $(PC) \leq (PC)+2$

Время: 2 цикла

Команда JTF <ADR8>

По команде JTF <ADR8> осуществляется ветвление в программе следующим образом:

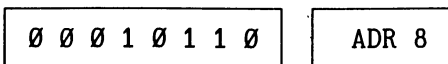
— если триггер переполнения таймера/счетчика TF установлен в состояние "1" (был перенос из 7-го разряда таймера/счетчика), то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JTF <ADR8>

выражение для короткого (8-битного) адреса
 внутри страницы памяти программ

Код:



короткий (8-битный) адрес ячейки внутри
 страницы памяти программ 00H-0FFH

Алгоритм: если $(TF)=1$,
 то $(PC[0-7]) \leq ADR8$,
 иначе $(PC) \leq (PC)+2$
 $(TF) \leq 0$

Время: 2 цикла

Команда JT0 <ADR8>

По команде JT0 <ADR8> осуществляется ветвление в программе следующим образом:

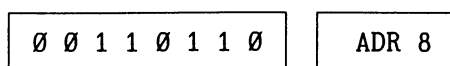
— если сигнал на выводе T0 равен единице, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JT0 <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:



короткий (8-битный) адрес ячейки внутри
страницы памяти программ 00H-0FFH

Алгоритм: если (T0)=1,
 то (PC[0-7]) <= ADR8,
 иначе (PC) <= (PC)+2

Время: 2 цикла

Команда JT1 <ADR8>

По команде JT1 <ADR8> осуществляется ветвление в программе следующим образом:

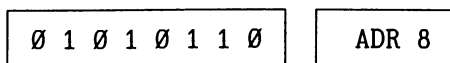
— если сигнал на выводе T1 равен единице, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JT1 <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:



короткий (8-битный) адрес ячейки внутри
страницы памяти программ 00H-0FFH

Алгоритм: если (T1)=1,
 то (PC[0-7]) <= ADR8,
 иначе (PC) <= (PC)+2

Время: 2 цикла

Команда JZ <ADR8>

По команде JZ <ADR8> осуществляется ветвление в программе следующим образом:

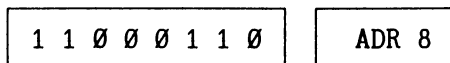
— если содержимое аккумулятора равно нулю, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JZ <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:



короткий (8-битный) адрес ячейки внутри
страницы памяти программ 00H-0FFH

Алгоритм: если (A)=0,
то (PC[0-7]) <= ADR8,
иначе (PC) <= (PC)+2

Время: 2 цикла

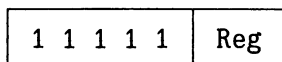
Команда MOV A,<Reg>

По команде MOV A,<Reg> содержимое рабочего регистра <Reg> пересылается в аккумулятор A.

Формат: MOV A,<Reg>

имя рабочего регистра R0...R7

Код:



код рабочего регистра 000B...111B

Алгоритм: <A> <= <Reg>

Время: 1 цикл

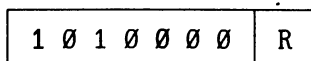
Команда MOV @R,A

По команде MOV @R,A содержимое аккумулятора A пересылается в ячейку внутренней памяти данных, адресуемую рабочим регистром <R>.

Формат: MOV @R,A

имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

Алгоритм: ((R)) <= <A>

Время: 1 цикл

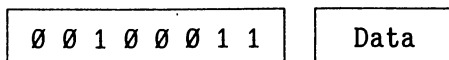
Команда MOV A,#<Data>

По команде MOV A, #<Data> непосредственные данные, определяемые элементом <Data>, пересылаются в аккумулятор A.

Формат: MOV A, #<Data>

выражение для непосредственных данных

Код:



непосредственные данные 00H...FFH

Алгоритм: <A> <= Data

Время: 2 цикла

Команда MOV <Reg>, #<Data>

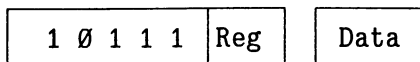
По команде MOV <Reg>, #<Data> непосредственные данные, определяемые элементом <Data> пересылаются в рабочий регистр <Reg>.

Формат: MOV <Reg>, #<Data>

выражение для непосредственных данных

имя рабочего регистра R0...R7

Код:



код рабочего регистра
000B...111B

непосредственные данные
00H...FFH

Алгоритм: (Reg) <= Data

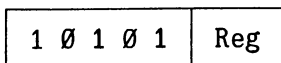
Команда MOV <Reg>, A

По команде MOV <Reg>, A содержимое аккумулятора пересылается в рабочий регистр <Reg>.

Формат: MOV <Reg>, A

имя рабочего регистра R0...R7

Код:



код рабочего регистра 000B...111B

Алгоритм: (Reg) <= <A>

Время: 1 цикл

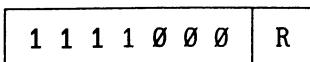
Команда MOV A, @<R>

По команде MOV A, @<R> содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, пересылается в аккумулятор A.

Формат: MOV A, @<R>

имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

Команда MOV A, T

По команде MOV A, T содержимое таймера/счетчика T пересылается в аккумулятор A.

Формат: MOV A, T

Код:

0	1	0	0	0	1	0
---	---	---	---	---	---	---

Алгоритм: $\langle A \rangle \leftarrow (T)$

Время: 1 цикл

Команда MOV T, A

По команде MOV T, A содержимое аккумулятора A пересылается в таймер/счетчик T.

Формат: MOV T, A

Код:

0	1	1	0	0	0	1	0
---	---	---	---	---	---	---	---

Алгоритм: $(T) \leftarrow A$

Время: 1 цикл

Команда MOVD A, <PD>

По команде MOVD A, <PD> содержимое дополнительного (4-разрядного) порта <PD> пересылается в 0...3 разряды аккумулятора A. Разряды 4...7 аккумулятора устанавливаются в состояние "0".

Формат: MOVD A, <PD>

имя дополнительного порта P4...P7

Код:

0	0	0	0	1	1
---	---	---	---	---	---

PD

код дополнительного порта 00B...11B

Алгоритм: $(A[0-3]) \leftarrow (PD)$
 $(A[4-7]) \leftarrow 0$

Время: 2 цикла

Команда MOVD <PD>, A

По команде MOVD <PD>, A содержимое аккумулятора A пересылается в дополнительный (4-разрядный) порт <PD>.

Формат: MOVD <PD>, A

имя дополнительного порта P4...P7

Код:

0	0	1	1	1	1
---	---	---	---	---	---

PD

код дополнительного порта 00B...11B

Алгоритм: $(PD) \leftarrow (A[0-3])$

Время: 2 цикла

Команда MOV_P A, @A

По команде MOV_P A, @A содержимое ячейки текущей страницы памяти программ, адресуемое аккумулятором A, пересылается в аккумулятор.

Формат: MOV_P A, @A

Код:

1	0	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Алгоритм: (A) <= ((PC[8-11]:A))

Время: 2 цикла

Пример: CONST: DB 'ABCDEF'

NEXT: MOV_P A, #LOW(CONST)
MOV_P A, @A ; (A) <= 'A'

Команда MOV_{P3} A, @A

По команде MOV_{P3} A, @A содержимое ячейки третьей страницы памяти программ, адресуемое аккумулятором A, пересылается в аккумулятор.

Формат: MOV_{P3} A, @A

Код:

1	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Алгоритм: (A) <= ((3:A))

Время: 2 цикла

Команда MOV_X A, @<R>

По команде MOV_X A, @<R> содержимое ячейки внешней памяти данных, адресуемой рабочим регистром <R>, пересылается в аккумулятор A.

Формат: MOV_X A, @<R>

имя рабочего регистра R0...R1

Код:

1	0	0	0	0	0	0	R
---	---	---	---	---	---	---	---

код рабочего регистра 0B...1B

Алгоритм: <A> <= ((R))

Время: 2 цикла

Команда MOV_X @R, A

По команде MOV_X @R, A содержимое аккумулятора A пересылается в ячейку внешней памяти данных, адресуемую рабочим регистром <R>.

Формат: MOV_X @R, A

имя рабочего регистра R0...R1

Код:

1	0	0	1	0	0	0	R
---	---	---	---	---	---	---	---

код рабочего регистра 0B...1B

Алгоритм: $\langle A \rangle \leftarrow ((R))$

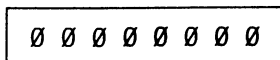
Время: 1 цикл

Команда NOP

По команде NOP выполняется холостая операция (содержимое счетчика команд увеличивается на единицу).

Формат: NOP

Код:



Алгоритм: $(PC) \leftarrow (PC) + 1$

Время: 1 цикл

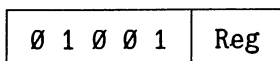
Команда ORL A, <Reg>

По команде ORL A, <Reg> содержимое рабочего регистра <Reg> поразрядно логически складывается с содержимым аккумулятора A. Результат вычисления записывается в аккумулятор.

Формат: ORL A, <Reg>

|
имя рабочего регистра R0...R7

Код:



|
код рабочего регистра 000B...111B

Алгоритм: $(A) \leftarrow (A) \text{ OR } (REG)$

Время: 1 цикл

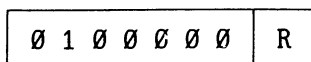
Команда ORL A, @<R>

По команде ORL A, @<R> содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, поразрядно логически складывается с содержимым аккумулятора A. Результат вычисления записывается в аккумулятор.

Формат: ORL A, @<R>

|
имя рабочего регистра R0...R1

Код:



|
код рабочего регистра 0B...1B

Алгоритм: $(A) \leftarrow (A) \text{ OR } ((R))$

Время: 1 цикл

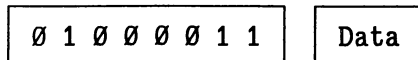
Команда ORL A, #<DATA>

По команде ORL A, #<DATA> непосредственные данные, определяемые элементом <DATA>, поразрядно логически складываются с содержимым аккумулятора A. Результат вычисления записывается в аккумулятор.

Формат: ORL A, #<DATA>

|
выражение для непосредственных данных

Код:



|
непосредственные данные 00H...FFH

Алгоритм: (A) <= (A) OR DATA

Время: 2 цикла

Пример: ; (A)=10101100B
 МЕТКА: ORL A, #0FH ; DATA=00001111B
 ; (A)=10101111B

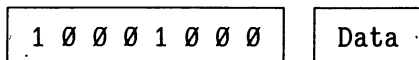
Команда ORL BUS, #<DATA>

По команде ORL BUS, #<DATA> непосредственные данные, определяемые элементом <DATA>, поразрядно логически складываются с содержимым порта BUS. Результат вычисления записывается в порт.

Формат: ORL BUS, #<DATA>

|
выражение для непосредственных данных

Код:



|
непосредственные данные 00H...FFH

Алгоритм: (BUS) <= (BUS) OR DATA

Время: 2 цикла

Команда ORL <PR>, #<DATA>

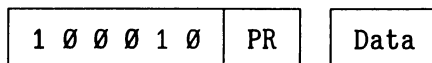
По команде ORL <PR>, #<DATA> непосредственные данные, определяемые элементом <DATA>, поразрядно логически складываются с содержимым порта <PR>. Результат вычисления записывается в резидентный порт.

Формат: ORL <PR>, #<DATA>

|
имя порта
P1...P2

|
выражение для непосредственных данных

Код:



|
код порта
01B...10B

|
непосредственные данные
00H...FFH

Алгоритм: (PR) <= (PR) OR DATA

Время: 2 цикла

Команда ORLD <PD>, A

По команде ORLD <PD>, A содержимое A поразрядно логически складывается с содержимым дополнительного (4-разрядного) порта <PD>. Результат вычисления записывается в дополнительный порт.

Формат: ORLD <PD>, A

|
имя дополнительного порта P4...P7

Код:

1 0 0 0 1 1	PD
-------------	----

|
код дополнительного порта 00B...11B

Алгоритм: $(PD) \leftarrow (PD) \text{ OR } (A[0-3])$

Время: 2 цикла

Команда OUTL BUS, A

По команде OUTL BUS, A содержимое аккумулятора A пересылается в порт BUS. Содержимое порта остается без изменений до последующей операции записи в порт. Пересылка данных сопровождается выдачей stroba на вывод WR.

Формат: OUTL BUS, A

Код:

0 0 0 0 0 0 1 0

Алгоритм: $(BUS) \leftarrow (A)$

Время: 2 цикла

Команда OUTL <PR>, A

По команде OUTL <PR>, A содержимое аккумулятора A пересылается в порт <PR>. Содержимое порта остается без изменений до последующей операции записи в порт.

Формат: OUTL <PR>, A

|
имя порта P1...P2

Код:

0 0 1 1 1 0	PR
-------------	----

|
код порта 01B...10B

Алгоритм: $(PR) \leftarrow (A)$

Время: 2 цикла

Команда RET

По команде RET осуществляется возврат из подпрограммы без восстановления содержимого регистра слова состояния программы PSW следующим образом:

- содержимое указателя стека SP уменьшается на единицу;
- полный адрес возврата из подпрограммы, находящийся в вершине стека, пересылается в счетчик PC;
- содержимое разрядов 4...7 регистра слова состояния программы не изменяется.

Формат: RET

Код:

1 0 0 0 0 0 1 1

Алгоритм: $(SP) \leftarrow (SP) - 1$
 $(PC) \leftarrow ((SP)[0-11])$

Время: 2 цикла

Команда RETR

По команде RETR осуществляется возврат из подпрограммы с восстановлением содержимого регистра слова состояния программы PSW следующим образом:

- содержимое указателя стека SP уменьшается на единицу;
- полный адрес возврата из подпрограммы, находящийся в вершине стека, пересылается в счетчик PC;
- система прерываний разблокируется;
- содержимое разрядов 4...7 регистра слова состояния программы восстанавливается по содержимому стека.

Формат: RETR

Код:

1 0 0 1 0 0 1 1

Алгоритм: $(SP) \leftarrow (SP) - 1$
 $(PC) \leftarrow ((SP)[0-11])$
 $(PSW[4-7]) \leftarrow ((SP)[12-15])$

Время: 2 цикла

Команда RL A

По команде RL A содержимое аккумулятора A циклически сдвигается влево на один двоичный разряд.

Формат: RL A

Код:

1 1 1 0 0 1 1 1

Алгоритм: для N от 6 до 0,
 $(A[N+1]) \leftarrow (A[N])$
 $(A[0]) \leftarrow (A[7])$

Время: 1 цикл

Пример: ; (A)=11110000B
 МЕТКА: RL A ; (A) <= 11100001B

Команда RLC A

По команде RLC A содержимое аккумулятора A и триггера признака переноса C циклически сдвигается влево на один двоичный разряд.

Формат: RLC A

Код:

1 1 1 1 0 1 1 1

Алгоритм: $(TMP) \leftarrow C$
 $(C) \leftarrow (A[7])$
 для N от 6 до 0,
 $(A[N+1]) \leftarrow (A[N])$
 $(A[0]) \leftarrow (TMP)$

Время: 1 цикл

Пример: ; (A)=11110000B, (C)=0
 МЕТКА: RLC A ; (A)<=11100000B, (C)=1

Команда RR A

По команде RR A содержимое аккумулятора A циклически сдвигается вправо на один двоичный разряд.

Формат: RR A

Код:

0	1	1	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: для N от 6 до 0,
 $(A[N]) \leftarrow (A[N+1])$
 $(A[7]) \leftarrow (A[0])$

Время: 1 цикл

Команда RRC A

По команде RRC A содержимое аккумулятора A и триггера признака переноса C циклически сдвигается вправо на один двоичный разряд.

Формат: RRC A

Код:

1	1	1	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: $(TMP) \leftarrow C$
 $(C) \leftarrow (A[0])$
 для N от 0 до 6,
 $(A[N]) \leftarrow (A[N+1])$
 $(A[7]) \leftarrow (TMP)$

Время: 1 цикл

Команда SEL MB0

По команде SEL MB0 триггер выбора банка памяти программ DBF устанавливается в состояние "0" (то есть выбирается нулевой банк памяти программ MB0).

Переключение на текущий банк памяти программ осуществляется командой CALL или JMP.

Формат: SEL MB0

Код:

1	1	1	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: $(DBF) \leftarrow 0$

Время: 1 цикл

Пример: МЕТКА : SEL MB0 ; $(DBF) \leftarrow 0$
 JMP+20H

Команда SEL MB1

По команде SEL MB1 триггер выбора банка памяти программ DBF устанавливается в состояние "1" (то есть выбирается первый банк памяти программ MB1).

Переключение на первый банк памяти программ осуществляется последующей командой CALL или JMP.

Формат: SEL MB1

Код:

1	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: (DBF) <= 1

Время: 1 цикл

Команда SEL RB0

По команде SEL RB0 триггер переключения банка рабочих регистров BS устанавливается в состояние "0" (то есть производится переключение на нулевой банк рабочих регистров RB0). После выполнения данной команды рабочим регистрам R0...R7 будет соответствовать внутренняя память данных с адресами 0...7.

Формат: SEL RB0

Код:

1	1	0	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: (BS) <= 0

Время: 1 цикл

Команда SEL RB1

По команде SEL RB1 триггер переключения банка рабочих регистров BS устанавливается в состояние "1" (то есть производится переключение на первый банк рабочих регистров RB1). После выполнения данной команды рабочим регистрам R0...R7 будет соответствовать внутренняя память данных с адресами 24...31.

Формат: SEL RB1

Код:

1	1	0	1	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: (BS) <= 1

Время: 1 цикл

Пример:

```

ORG 3
JNI INIT
ORG 10H
INIT: MOV R7,A
      SEL RB1
      MOV R7,#0FAH

      SEL RB0
      MOV A,R7
      RETR

```

Команда STOP TCNT

По команде STOP TCNT таймер/счетчик TCNT останавливается (прекращается инкрементирование). Запуск таймера/счетчика осуществляется следующей командой STRT CNT или STRT T.

Формат: STOP TCNT

Код:

0	1	1	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: останов таймера/счетчика

Время: 1 цикл

Команда STRT CNT

По команде STRT CNT таймер/счетчик подключается к выводу T1 и запускается в качестве счетчика внешних событий (то есть по сигналу на выводе T1 содержимое таймера/счетчика увеличивается на единицу).

Формат: STRT CNT

Код:

0	1	0	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: запуск таймера/счетчика в качестве счетчика

Время: 1 цикл

Команда STRT T

По команде STRT T таймер/счетчик подключается к генератору тактовых сигналов и запускается в качестве внутреннего таймера.

Содержимое таймера/счетчика увеличивается на единицу через каждые 32 машинных цикла. Отсчет циклов начинается непосредственно после выполнения команды STRT T.

Формат: STRT T

Код:

0	1	0	1	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: запуск таймера/счетчика в качестве таймера

Время: 1 цикл

Команда SWAP A

По команде SWAP A разряды 0...3 и 4...7 аккумулятора A обмениваются содержимым.

Формат: SWAP A

Код:

0	1	0	0	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: $(A[4-7]) \leftrightarrow (A[0-3])$

Время: 1 цикл

Команда XCH A, <Reg>

По команде XCH A, <Reg> рабочий регистр <Reg> и аккумулятор A обмениваются содержимым.

Формат: XCH A, <Reg>

имя рабочего регистра R0...R7

Код:

0	0	1	0	1	Reg
---	---	---	---	---	-----

код рабочего регистра 000B...111B

Алгоритм: $(A) \leftrightarrow (REG)$

Время: 1 цикл

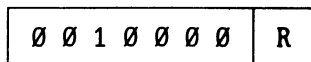
Команда XCH A, @<R>

По команде XCH A, @<R> ячейка внутренней памяти данных, адресуемой рабочим регистром <R>, и аккумулятор A обмениваются содержимым.

Формат: XCH A, @<R>

имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

Алгоритм: <A> <=> ((R))

Время: 1 цикл

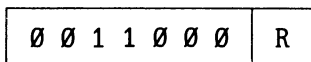
Команда XCHD A, @<R>

По команде XCHD A, @<R> 0...3 разряды ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, и 0...3 разряды аккумулятора A обменяются содержимым (содержимое 4...7 разрядов ячейки памяти и аккумулятора не изменяются).

Формат: XCHD A, @<R>

имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

Алгоритм: (A[0-3]) <=> ((R)[0-3])

Время: 1 цикл

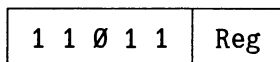
Команда XRL A, <Reg>

По команде XRL A, <Reg> содержимое рабочего регистра <Reg> и содержимое аккумулятора A поразрядно складываются по модулю 2. Результат вычисления записывается в аккумулятор.

Формат: XRL A, <Reg>

имя рабочего регистра R0...R7

Код:



код рабочего регистра 000B...111B

Алгоритм: (A) <=> (A) XOR (REG)

Время: 1 цикл

Команда XRL A, @<R>

По команде XRL A, @<R> содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, и содержимое аккумулятора A поразрядно складываются по модулю 2. Результат вычисления записывается в аккумулятор.

Формат: XRL A,@<R>

имя рабочего регистра R0...R1

Код:

1	1	0	1	0	0	0	R
---	---	---	---	---	---	---	---

код рабочего регистра 0B...1B

Алгоритм: (A) <= (A) XOR ((R))

Время: 1 цикл

Команда XRL A,#<Data>

По команде XRL A,#<Data> непосредственные данные, определяемые элементом <DATA>, и содержимое аккумулятора A по разрядам складываются по модулю 2. Результат вычисления записывается в аккумулятор.

Формат: XRL A,#<Data>

выражение для непосредственных данных

Код:

1	1	0	1	1	0	1	1	Data
---	---	---	---	---	---	---	---	------

непосредственные данные 00H...FFH

Алгоритм: (A) <= (A) XOR DATA

Время: 2 цикла

Пример:

МЕТКА : XRL A,#0FH

; (A) = 0 1 0 1 1 1 0 0 B
; DATA = 0 0 0 0 1 1 1 1 B
; (A) = 0 1 0 1 0 0 1 1 B

1.6. Электрические параметры.

Электрические параметры микросхем семейства МК48 в диапазоне температур от минус 10°C до 70°C приведены в табл. 1.8 (статические параметры) и табл. 1.9 (динамические параметры).

Предельные значения электрических режимов эксплуатации приведены в табл. 1.10.

Таблица 1.8. Электрические параметры микросхем семейства МК48 в диапазоне температур от минус 10°C до 70°C

N п/ п	ПАРАМЕТРЫ	БУКВЕННОЕ ОБОЗНАЧЕНИЕ	ЗНАЧЕНИЕ ПАРАМЕТРОВ					
			KM1816BE48 KP1816BE35		KP1816BE49 KP1816BE39		KP1830BE48 KP1830BE35	
			мин.	макс.	мин.	макс.	мин.	макс.
1	2	3	4	5	6	7	8	9
1	Напряжение питания, В – основное	U _{CC}	4,75	5,25	4,75	5,25	4,75	5,25
	– дополнительное	U _{DD}	4,75	5,25	4,75	5,25	–	–
2	Выходное напряжение, В – высокого уровня	U _{OH}	2,4	–	2,4	–	2,4	–
	– низкого уровня	U _{OL}	–	0,45	–	0,45	–	0,45

Продолжение таблицы 1.8

1	2	3	4	5	6	7	8	9
3	Выходной ток высокого уровня, мА - сигналов DB - сигналов RD, WR, PWE, ALE - остальных сигналов	I_{OH} I_{OH1} I_{OH2}	- 0,1 - 0,1 - 0,04	- - -	-0,4 -0,1 -0,04	- - -	-0,4 -0,1 -0,04	- - -
4	Выходной ток низкого уровня, мА - сигналов DB - сигналов RD, WR, PWE, ALE - сигнала PR - остальных сигналов	I_{OL} I_{OL1} I_{OL2} I_O	- - - -	2,0 1,8 1,0 1,6	- - - -	2,0 1,8 1,0 1,6	- - - -	2,0 1,8 1,0 1,6
5	Входное напряжение высокого уровня, В - сигналов BQ1, BQ2, SR - сигнала EMA(чтение ПЗУ) (программирование) - сигнала PR (программирование) - остальных сигналов	U_{IH1} $U_{IH,EMA}$ $U_{IH,EMA}$ U_{IH}	3,8 21,5 21,5 2,0	U_{CC} 24,5 24,5 U_{CC}	3,8 11,5 - 2,0	U_{CC} 12,5 - U_{CC}	3,8 11,5 - 2,2	U_{CC} 12,5 - U_{CC}
6	Входное напряжение низкого уровня, В - сигналов BQ1, BQ2, SR - сигнала EMA(чтение ПЗУ) (программирование) - сигнала PR - остальных сигналов	U_{IL1} $U_{IL,EMA}$ $U_{IL,PR}$ U_{IL}	-0,5 4,5 - -0,5	0,6 5,25 0,2 0,8	-0,5 4,75 - -0,5	0,6 5,25 - 0,8	-0,5 4,75 - -0,5	0,8 5,25 - 0,8
7	Напряжение питания при программировании, В - высокого уровня - низкого уровня	$U_{dd,H}$ $U_{dd,L}$	24,0 4,75	26,0 5,25	- -	- -	- -	- -
8	Ток утечки на входах, мкА	I_{LI}	-10	10	-10	10	-10	10
9	Выходной ток в состоянии 'выключено', мкА	I_{OZ}	-10	10	-10	10	-10	10
10	Входной ток, мА - на выводах SR, SS, P1, P2 - на выводе PR (программирование) - на выводе EMA (программирование)	I_I $I_{I,PR}$ $I_{I,EMA}$	-0,5 - -	- 16,0 1,0	-0,5 - -	- - -	-0,5 - -	- - -
11	Ток потребления, мА - по выводу U_{dd} - суммарный - по выводу U_{CC} - по выводу U_{dd} при программировании - в режиме микропотребления	I_{dd} $I_{dd}+I_{CC}$ I_{CC} I_{dd1} I_{CC1}	- - - - -	20,0 135,0 - 30 -	- - - - -	10,0 110,0 - - -	- - - - -	- - 8 - 0,1
12	Емкость входа/выхода, пФ	$C_{I/O}$	-	20	-	20	-	20
13	Входная емкость, пФ	C_I	-	10	-	10	-	10

Таблица 1.9. Электрические параметры в диапазоне температур от минус 10°C до 70°C при $U_{CC} = 5 \text{ В} \pm 5\%$

N п/п	НАИМЕНОВАНИЕ ПАРАМЕТРА, ЕДИНИЦА ИЗМЕРЕНИЯ	БУКВЕННОЕ ОБОЗНАЧЕНИЕ	ЗНАЧЕНИЕ ПАРАМЕТРОВ					
			КМ1816ВЕ48 КР1816ВЕ35		КР1816ВЕ49 КР1816ВЕ39		КР1830ВЕ48 КР1830ВЕ35	
			не менее	не более	не менее	не более	не менее	не более
1	2	3	4	5	6	7	8	9
ДИНАМИЧЕСКИЕ ПАРАМЕТРЫ								
1.	Clock Period Период следования импульсов тактового сигнала $\overline{BQ1}$, нс	$T_{BQ1}=t$	167	1000	90,9	1000	167	1000
			t		t		t	
2.	Cycle Time Время цикла сигнала ALE, мкс	$t_{CY}(ALE)$	2,5	15	1,36	15	2,5	15
			15t		15t		15t	
3.	ALE Pulse Width Длительность сигнала ALE, нс	$t_W(ALE, H)$	400	-	150	-	500	-
			3,5t-170		3,5t-170		4t-170	
4.	Data Hold after \overline{WR} Время задержки сигналов данных DB(0-7) относительно сигнала \overline{WR} , нс	$t_D(\overline{WR}, LH-DB, HZ/LZ)$	120	-	40	-	40	-
			t-50		t-50		0,5t-40	
5.	Control Pulse Width (\overline{PSEN}) Длительность сигнала \overline{PME} , нс	$t_W(\overline{PME}, L)$	700	-	350	-	800	-
			6t-200		6t-200		6t-200	
6.	Control Pulse Width (\overline{RD} , \overline{WR}) Длительность сигнала \overline{RD} , нс Длительность сигнала \overline{WR} , нс	$t_W(\overline{RD}, L)$ $t_W(\overline{WR}, L)$	700	-	480	-	1050	-
			7,5t-200		7,5t-200		7,5t-200	
7.	PROG Pulse Width Длительность сигнала \overline{PR} , нс	$t_W(\overline{PR}, L)$	1510	-	700	-	1500	-
			10,5t-250		10,5t-250		10,5t-230	
8.	Addr Setup to ALE Время задержки сиг- нала ALE относительно адреса DB(0-7), нс Время задержки сиг- нала ALE относительно сигналов адреса P2(0-3), нс	$t_D(ADB, ZH/ZL-ALE, HL)$	150	-	160	-	300	-
			2t-110		2t-110		2,5t-110	
		$t_D(AP2, LH/HL-ALE, HL)$	150	-	160	-	60	-
			2t-110		2t-110		t-110	
9.	Addr Hold from ALE Время задержки сиг- налов адреса DB(0-7) и адреса P2(0-3) относительно сигнала ALE, нс	$t_D(ALE, HL-ADB, HZ/LZ)$ $t_D(ALE, HL-AP2, HL/LH)$	80	-	50	-	40	-
			t-40		t-40		0,5t-40	
10.	Data Setup before \overline{WR} Время задержки сиг- нала \overline{WR} относительно сигналов данных DB(0-7), нс	$t_D(DB, ZH/ZL-\overline{WR}, LH)$	500	-	390	-	800	-
			6,5t-200		6,5t-200		6t-200	
11.	Data Hold (\overline{PSEN} , \overline{RD}) Время сохранения сигналов данных DB(0-7) относительно сигнала \overline{PME} , нс Время сохранения сигналов данных DB(0-7) относительно сигнала \overline{RD} , нс	$t_V(\overline{PME}, LH-DB, HZ/LZ)$ $t_V(\overline{RD}, LH-DB, HZ/LZ)$	0	200	0	110	0	50
			1,5t-30		1,5t-30		t+50	

Продолжение таблицы 1.9

1	2	3	4	5	6	7	8	9
12.	RD to Data in Время установления сигналов данных DB(0-7) относительно сигнала RD, нс	$t_{SU}(\overline{RD}, HL-DB, LH/HL)$	-	500	-	350	-	800
			6t-170		6t-170		6t-170	
13.	PSEN to Data in Время установления сигналов данных DB(0-7) относительно сигнала PМЕ, нс	$t_{SU}(\overline{PME}, HL-DB, HL/LH)$	-	500	-	190	-	550
			4,5t-170		4,5t-170		4,5t-170	
14.	Addr Setup to \overline{WR} Время задержки сигнала \overline{WR} относительно адреса DB(0-7), нс	$t_D(ADB, ZH/ZL-\overline{WR}, HL)$	230	-	300	-	750	-
			5t-150		5t-150		5,5t-150	
15.	Addr Setup to Data (RD) Время установления сигналов данных DB(0-7) относительно сигналов адреса DB(0-7) при активном сигнале RD, нс	$t_{SU}(ADB, ZH/ZL-DB, LH/HL)$	-	950	-	800	-	1650
			10,5t-220		10,5t-220		11,5t-220	
16.	Addr Setup to Data (PSEN) Время установления сигналов данных DB(0-7) относительно сигналов адреса DB(0-7) при активном сигнале PМЕ, нс	$t_{SU}(ADB, ZH/ZL-DB, LH/HL)$	-	950	-	570	-	1150
			7,5t-200		7,5t-200		8,5t-220	
17.	Addr Float to \overline{RD} Время задержки сигнала \overline{RD} относительно окончания сигналов адреса DB(0-7), нс	$t_D(ADB, HZ/LZ-\overline{RD}, HL)$	0	-	140	-	370	-
			2t-40		2t-40		2,5t-40	
18.	Addr Float to \overline{PSEN} Время задержки сигнала \overline{PME} относительно окончания сигналов адреса DB(0-7), нс	$t_D(ADB, HZ/LZ-\overline{PME}, HL)$	0	-	10	-	120	-
			0,5t-40		0,5t-40		t-40	
19.	ALE to Control (RD, WR) Время задержки сигнала $\overline{RD}, \overline{WR}$ относительно окончания сигнала ALE, нс	$t_D(ALE, HL-\overline{RD}, HL)$ $t_D(ALE, HL-\overline{WR}, HL)$	400	-	200	-	400	-
			3t-75		3t-75		3t-75	
20.	ALE to Control (\overline{PSEN}) Время задержки сигнала \overline{PME} относительно сигнала ALE, нс	$t_D(ALE, HL-\overline{PME}, HL)$	170	-	60	-	170	-
			1,5t-75		1,5t-75		1,5t-75	
21.	Control to ALE (RD, WR, PR) Время задержки сигнала ALE относительно сигналов RD, WR, PR, нс	$t_D(\overline{RD}, LH-ALE, LH)$ $t_D(\overline{WR}, LH-ALE, LH)$ $t_D(\overline{PR}, LH-ALE, LH)$	120	-	50	-	40	-
			t-40		t-40		0,5t-40	
22.	Control to ALE (\overline{PSEN}) Время задержки сигнала ALE относительно сигнала PМЕ, нс	$t_D(\overline{PME}, LH-ALE, LH)$	620	-	320	-	540	-
			4t-40		4t-40		3,5t-40	
23.	Port Control Setup to PR Время задержки сигнала PR относительно сигналов управления дополнительным портом P2(0-3), нс	$t_D(CP2, LH/HL-\overline{PR}, HL)$	110	-	100	-	170	-
			1,5t-80		1,5t-80		1,5t-80	

Продолжение таблицы 1.9

1	2	3	4	5	6	7	8	9
24.	Port Control Hold to \overline{PR} Время задержки сигналов управления дополнительным портом P2(0-3) относительно сигнала \overline{PR} , нс	$t_D(\overline{PR}, HL-CP2, HL/LH)$	140	-	160	-	160	-
			4t-260		4t-260		1,5t-90	
25.	\overline{PR} to P2 Input Valid Время установления сигналов данных дополнительного порта P2(0-3) относительно сигнала \overline{PR} , нс	$t_{SU}(\overline{PR}, HL-DP2, LH/HL)$	-	810	-	700	-	1350
			8,5t-120		8,5t-120		9t-120	
26.	Input Data Hold from \overline{PR} Время сохранения сигналов данных дополнительного порта P2(0-3) относительно сигнала \overline{PR} , нс	$t_V(\overline{PR}, LH-DP2, HL/LH)$	0	150	0	140	0	80
			1,5t		1,5t		0t+80	
27.	Output Data Setup Время задержки сигнала \overline{PR} относительно сигнала данных для дополнительного порта P2(0-3), нс	$t_D(DP2, LH/HL-\overline{PR}, LH)$	220	-	400	-	700	-
			6t-290		6t-290		6t-290	
28.	Output Data Hold Время задержки сигналов данных для дополнительного порта P2(0-3) относительно сигнала \overline{PR} , нс	$t_D(\overline{PR}, LH-DP2, HL/LH)$	65	-	90	-	160	-
			1,5t-90		1,5t-90		1,5t-90	
29.	Port 2 I/O Setup to ALE Время задержки сигнала ALE относительно данных порта P2(0-3), нс	$t_D(DP2, HL/LH-ALE, LH)$	400	-	160	-	380	-
			4t-200		4t-200		3,5t-200	
30.	Port 2 I/O Hold to ALE Время задержки сигналов данных порта P2(0-3) относительно сигнала ALE, нс	$t_D(ALE, LH-DP2, LH/HL)$	150	-	40	-	60	-
			1,5t-120		1,5t-120		t-110	
31.	Port Output from ALE Время задержки сигналов данных порта P2(0-7) относительно сигнала ALE, нс	$t_D(ALE, HL-DP2, HL/LH)$	-	500	-	510	-	850
			4,5t+100		4,5t+100		4,5t+100	
32.	T0 Rep Rate Время цикла сигнала T0, нс	$t_{CY}(T0)$	500	-	270	-	500	-
			3t		3t		3t	
ПРОГРАММИРОВАНИЕ И ПРОВЕРКА ПЗУ, ПРОВЕРКА ПЗУ								
1.	Address Setup Time to RESET Время установления сигналов адреса DB(0-7) относительно сигнала \overline{SR}	$t_{SU}(ADB, ZH/ZL-\overline{SR}, LH)$	$4t_{CY}$	-	$4t_{CY}$	-	18t	-
***	Время установления адреса порта P2(0-1) относительно сигнала \overline{SR}	$t_{SU}(AP2, LH/HL-\overline{SR}, LH)$	$4t_{CY}$	-	$4t_{CY}$	-	32,5t	-

Продолжение таблицы 1.9

1	2	3	4	5	6	7	8	9
2.	Address Hold Time After RESET Время сохранения сигналов адреса DB(0-7) и P2(0-1) относительно сигнала \overline{SR} , нс	$t_V(\overline{SR}, LH-ADB, HZ/LZ)$ $t_V(\overline{SR}, LH-ADB, HL/LH)$	$4t_{cy}$	-	$4t_{cy}$	-	50	
3.	Data in Setup Time to PR Время установления сигналов данных DB(0-7) относительно сигнала PR	$t_{SU}(DB, LH/HL-\overline{PR}, LH)$	$4t_{cy}$	-	-	-	-	-
4.	Data in Hold Time After PR Время сохранения сигналов данных DB(0-7) относительно сигнала PR	$t_V(\overline{PR}, HL-DB, HZ/LZ)$	$4t_{cy}$	-	-	-	-	-
5.	U_{dd} Hold Time Before PR Время установления сигнала U_{dd} относительно сигнала PR	$t_{SU}(U_{dd}, LH-\overline{PR}, LH)$	$4t_{cy}$	-	-	-	-	-
6.	U_{dd} Hold Time After PR Время сохранения сигнала U_{dd} относительно сигнала PR, мкс	$t_V(\overline{PR}, HL-U_{dd}, HL)$	0	-	-	-	-	-
7.	Program Pulse Width Длительность сигнала PR, мс	$t_W(\overline{PR}, H)$	50	60	-	-	-	-
8.	Test 0 Setup Time for Program Mode Время установления сигнала T0 относительно сигнала \overline{SR}	$t_{SU}(T0, HL-\overline{SR}, LH)$	$4t_{cy}$	-	$4t_{cy}$	-	-	-
9.	Test 0 Hold Time After Program Mode Время сохранения сигнала T0 относительно U_{dd}	$t_V(U_{dd}, HL-T0, LH)$	$4t_{cy}$	-	-	-	-	-
10.	Test 0 to Data Out Delay Время задержки сигналов данных DB(0-7) относительно сигнала T0	$t_D(T0, LH-DB, ZH/ZL)$	$4t_{cy}$	-	$4t_{cy}$	-	-	-
11.	RESET Pulse Width to Latch Address * Длительность сигнала \overline{SR} , нс	$t_W(\overline{SR}, L)$	$4t_{cy}$	-	$4t_{cy}$	-	45t	-
12.	U_{dd} and PR Rise and Fall Times Время перехода сигналов U_{dd} , PR из состояния логической единицы в состояние логического нуля, мкс Время перехода сигналов U_{dd} , PR из состояния логического нуля в состояние логической единицы, мкс	t_{THL}, U_{dd} t_{THL}, \overline{PR} t_{TLH}, U_{dd} t_{TLH}, \overline{PR}	0,5 0,5	2 2	- -	- -	- -	- -

Продолжение таблицы 1.9

1	2	3	4	5	6	7	8	9
13. **	CPU Operation Cycle Time Время цикла, мкс	t_{cy}	5	-	-	-	-	--
14.	RESET Setup Time before EA Время установления сигнала EMA относительно сигнала SR	$t_{SU}(SR, HL-EMA, LH)$	$4t_{cy}$	-	$4t_{cy}$	-	$15t$	-
15.	Время сохранения сигнала SR относительно сигнала T0, нс	$t_V(T0, LH-SR, HL)$	$4t_{cy}$	-	-	-	-	-
16.	Время установления сигнала PR относительно сигналов данных DB(0-7)	$t_{SU}(PR, ZL-DB, LH/HL)$	$4t_{cy}$	-	-	-	-	-
17.	Время сохранения сигнала PR относительно сигналов данных DB(0-7)	$t_V(DB, HZ/LZ-PR, LZ)$	$4t_{cy}$	-	-	-	-	-
18.	Время задержки сигнала ALE относительно сигналов адреса порта P2(0-1), нс	$t_D(AP2, LH/HL-ALE, LH)$	-	-	-	-	$2,5t-50$	-
19.	Время задержки сигналов данных относительно сигналов адреса DB(0-7), нс	$t_D(ADB, HZ/LZ-DB, ZH/ZL)$	-	-	-	-	-	$3t-50$ $18t-$
20.	Время установления сигналов адреса DB(0-7) относительно сигнала ALE, нс	$t_{SU}(ADB, ZH/ZL-ALE, LH)$	-	-	-	-	$2,5t-50$	-

Примечания: 1. * Для первого импульса длительность сигнала SR — $8t_{cy}$

2. ** При $f_{BQ1}=3$ МГц; $t_{cy}=15/f_{BQ1}$.

3.*** Для КР1816ВЕ49 P2(0-2).

4. Символ "LH" ("HL") обозначает переход сигнала из состояния низкого (высокого) уровня в состояние высокого (низкого) уровня. Символ "ZH" ("ZL") обозначает переход сигнала из высокоимпедансного состояния в состояние высокого (низкого) уровня. Символ "HZ" ("LZ") обозначает переход сигнала высокого (низкого) уровня в высокоимпедансное состояние.

5. Черта между символами "HL/LH", "LH/HL", "ZH/ZL", "HZ/LZ" означает, что параметр имеет одинаковое значение для обоих переходов.

6. Значения временных параметров приведены для емкостной нагрузки по входам/выходам $C_{LI/O} < 150$ пФ и емкостной нагрузки по выходам $C_L < 80$ пФ.

7. Значения временных параметров указаны при частоте генератора 6 МГц для КР1816ВЕ35, КМ1816ВЕ48, КР1830ВЕ35, КР1830ВЕ48 и 11 МГц для КР1816ВЕ39 и КР1816ВЕ49.

8. В ряде случаев приведенные в таблице 1.9 числовые значения временных параметров значительно отличаются от подсчитанных по приведенным формулам через параметр t . Для максимальных частот синхронизации (6 МГц и 11 МГц) рекомендуется использовать приводимые числовые значения временных параметров. Для меньших частот синхронизации допустимо пользоваться расчетами по приведенным формулам.

На рисунке примечания 8 иллюстрируется работа ОМЭВМ серии 1816 по тактам частоты f_{BQ1} . Из приведенной диаграммы понятно происхождение формул для расчета временных параметров.

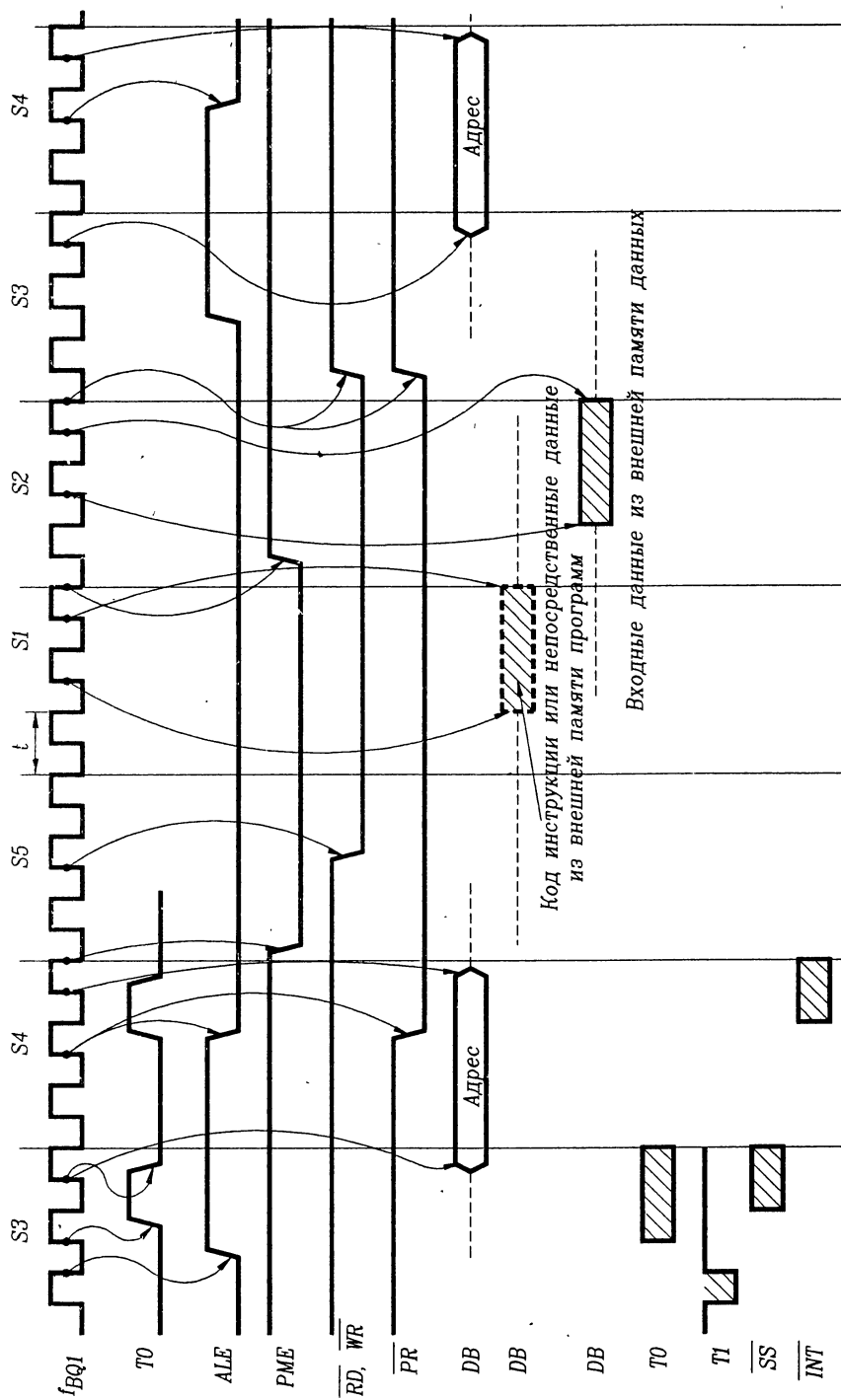


Рис. примечания 8 к табл. 1.9

Таблица 1.10. Предельные значения электрических режимов эксплуатации

N п/ п	НАИМЕНОВАНИЕ	БУКВЕННОЕ ОБОЗНАЧЕНИЕ	Н О Р М А					
			КМ1816ВЕ48 КР1816ВЕ35		КР1816ВЕ49 КР1816ВЕ39		КР1830ВЕ48 КР1830ВЕ35	
			не менее	не более	не менее	не более	не менее	не более
1.	Напряжение питания, В - основное	U_{CC}	-	7,0	-	7,0	-	7,0
	- дополнительное	U_{dd}	-	27,0	-	7,0	-	7,0
2.	Напряжение на выводе \overline{PR} при программировании, В	U_{PR}	-	26,0	-	-	-	-
3.	Напряжение на выводе ЕМА при программировании (чтении ПЗУ,) В	U_{EMA}	-	26,0	-	13,0	-	13,0
4.	Входное напряжение на остальных выводах, В	U_I	-0,5	7,0	-0,5	7,0	-0,5	7,0
5.	Выходной ток высокого уровня, мА	I_{OH}	-0,8	-	-0,8	-	-0,6	-
6.	Выходной ток низкого уровня, мА	I_{OL}	-	3,0	-	2,2	-	3,0
7.	Емкость нагрузки, пФ	C_L	-	300	-	220	-	300

1.7. Применение ОМЭВМ

1.7.1. Совместная работа с устройствами аналогового ввода-вывода

На рис. 1.26 приведена функциональная схема подсистемы аналогового ввода-вывода на основе ОМЭВМ КМ1816ВЕ48, КР1816ВЕ49 или КР1830ВЕ48 (все указанные микросхемы имеют внутреннюю память программ). Рассматриваемая подсистема аналогового ввода-вывода обслуживает восемь аналоговых входов и один аналоговый выход.

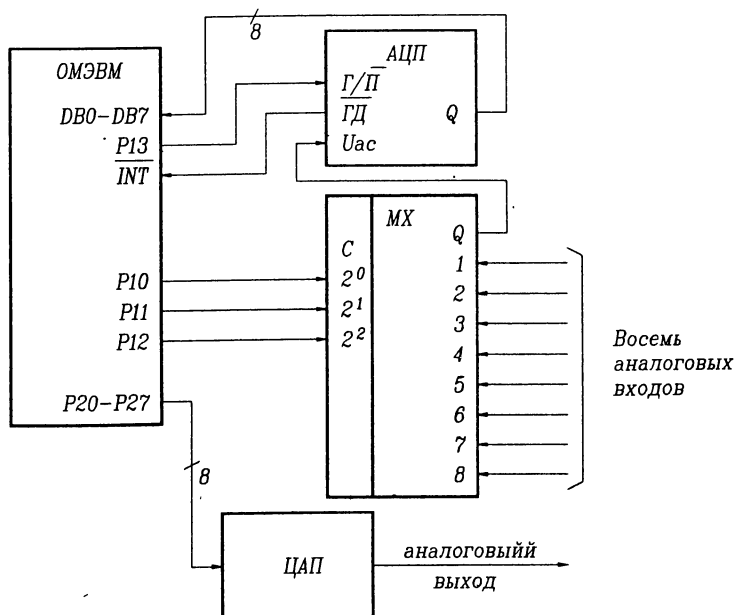


Рис. 1.26. Функциональная схема подсистемы аналогового ввода-вывода

Восьмиканальный аналоговый коммутатор (МХ) подключает один из аналоговых входов подсистемы к аналоговому входу U_{ac} аналого-цифрового преобразователя (АЦП). Подключаемый аналоговый вход определяется кодом на выходах 2^0-2^2 коммутатора и выбирается программно с помощью разрядов 0, 1 и 2 порта P1 ОМЭВМ.

Сигналом запуска по входу Г/П запускается цикл преобразования АЦП, по окончании которого на выходах Q АЦП выставляется код, соответствующий напряжению на входе U_{ac} АЦП, а на выходе ГД АЦП устанавливается низкий логический уровень, означающий готовность данных на выходах АЦП и вызывающий прерывание ОМЭВМ по входу \overline{INT} . В подпрограмме обработки этого прерывания ОМЭВМ вводит код с выходов АЦП через порт P0 (DB). Сигнал запуска АЦП формируется программно.

Через порт P2 ОМЭВМ осуществляется вывод байта данных на входы цифро-аналогового преобразователя (ЦАП) для управления аналоговым выходом подсистемы аналогового ввода-вывода.

Функциональная схема на рис. 1.26 приведена для аналогового коммутатора КР590КН6 и АЦП К1113ПВ1. На рис. 1.26а показана временная диаграмма работы АЦП К1113ПВ1, на рис. 1.26б и 1.26в приведены соответственно функциональная схема и таблица истинности микросхемы аналогового коммутатора КР590КН6.

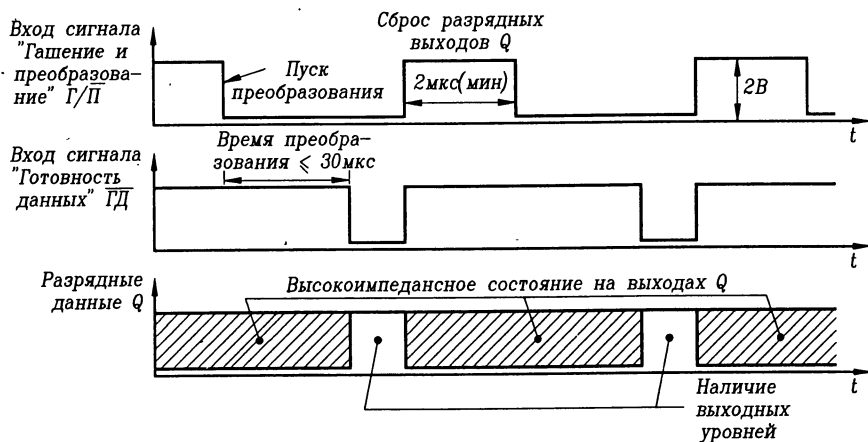


Рис. 1.26а. Временная диаграмма работы АЦП К1113ПВ1

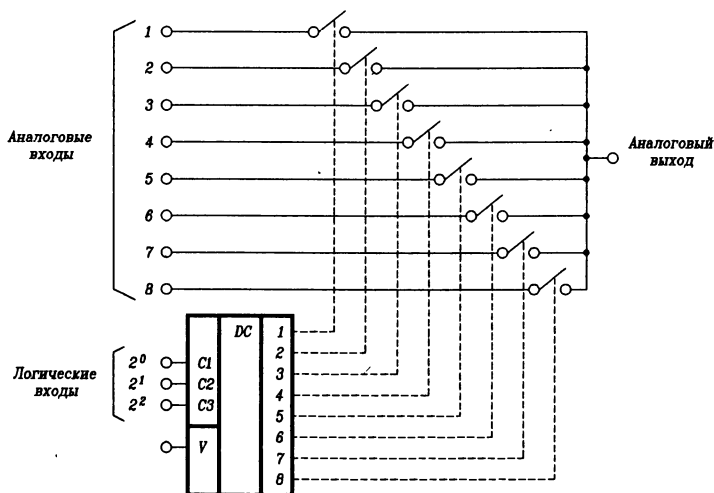


Рис. 1.26б. Функциональная схема аналогового коммутатора КР590КН6

Логические входы				Открыт канал
2 ²	2 ¹	2 ⁰	Разрешение V	
0	0	0	1	1
0	0	1	1	2
0	1	0	1	3
0	1	1	1	4
1	0	0	1	5
1	0	1	1	6
1	1	0	1	7
1	1	1	1	8
X	X	X	0	Все закрыты

X – при любом уровне

Рис. 1.26в. Таблица истинности микросхемы аналогового коммутатора КР590КН6

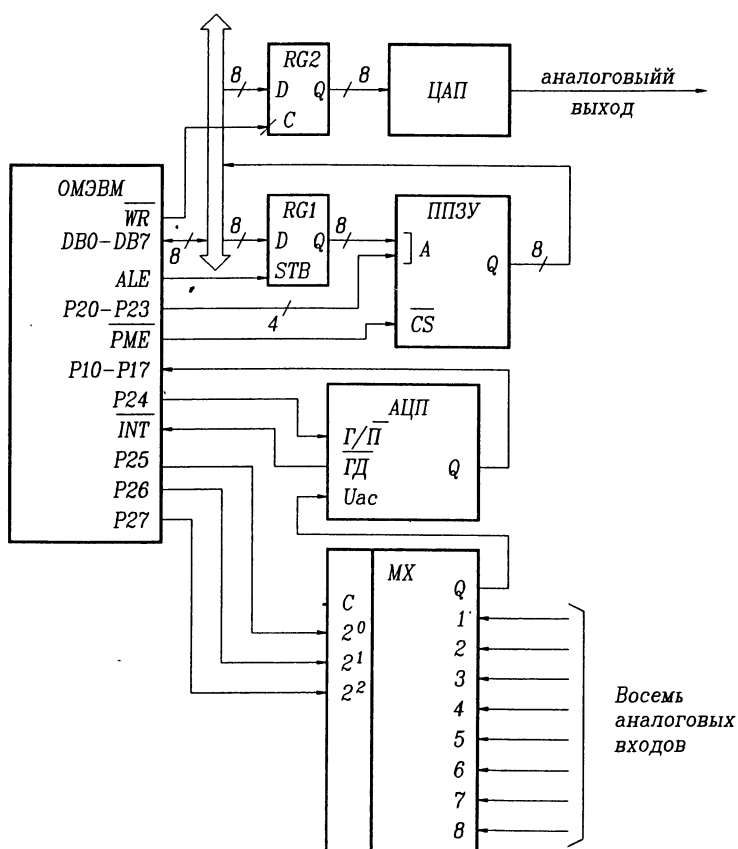


Рис. 1.27. Функциональная схема подсистемы аналогового ввода-вывода

На рис. 1.27 представлена функциональная схема подсистемы аналогового ввода-вывода, в которой ОМЭВМ имеет внешнюю память программ (ППЗУ). Соответственно, в данной подсистеме могут использоваться все микросхемы ОМЭВМ семейства МК48.

В регистре-защелке RG1 по срезу сигнала ALE фиксируется байт младших адресов памяти программ. Сигналы с выходов RG1 и с линий P20—P23 ОМЭВМ образуют 12-разрядный адрес, поступающий на адресные входы ППЗУ. В отсутствии активного уровня сигнала PWE выходы ППЗУ Q находятся в высокоимпедансном состоянии.

В регистре RG2 по фронту сигнала \overline{WR} фиксируется байт данных для управления через ЦАП аналоговым выходом рассматриваемой подсистемы аналогового ввода-вывода.

В остальном схема на рис. 1.27 работает аналогично схеме, представленной на рис. 1.26.

В качестве регистров RG1 и RG2 могут использоваться соответственно микросхемы KP1533IP22 (KP580IP82) и KP1533IP23.

1.7.2. Применение таблиц для вычисления функций.

Большинство процессов, для управления которыми применяются микропроцессорные системы с аналоговым вводом-выводом, по природе своей нелинейны. Математические методы их линеаризации требуют затрат вычислительной мощности, поэтому в ряде случаев целесообразно использовать методы табличного поиска.

При этом вместо вычисления значений функции в реальном масштабе времени производится автономное вычисление этих значений и занесение их в память в виде таблицы.

В качестве примера использования таблиц как части схемы аналогового вывода, рассмотрим систему на базе ОМЭВМ, в которой выходным сигналом ОМЭВМ является синусоидальный сигнал переменной частоты. Одним из методов реализации такой функции может быть использование таймера событий для организации прерываний с фиксированной частотой в 256 раз большей, чем требуемая выходная частота. В момент прерывания соответствующее значение синусоидальной функции может быть вычислено из ряда Маклорена:

$$\sin(X) = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} \dots \frac{(-1)^K \cdot X^{(2 \cdot K + 1)}}{(2 \cdot K + 1)!},$$

где K выбирается достаточно большим для обеспечения требуемой точности. Эти вычисления занимают много времени и ограничивают диапазон выходных частот, которые можно обеспечить. При использовании предварительной записи значений функции в таблице ОМЭВМ в момент прерывания находит соответствующее значение в таблице и выводит его на ЦАП.

В системе команд ОМЭВМ семейства МК48 предусмотрена специальная команда MOVP3 A, @A для выбора данных из таблиц, если таблица помещена в последние 256 байт (3-я страница первого килобайта памяти программ). Эта команда использует исходное содержимое аккумулятора для индексации ячейки третьей страницы памяти программ. Содержимое указанной ячейки считывается и помещается в аккумулятор. Если требуется таблица объемом меньше 256 байт, ее можно разместить на любой странице памяти программ и для нахождения данных в таблице использовать команду MOVP A, @A. Эта команда аналогична предыдущей, но она предполагает, что таблица содержится на текущей странице памяти программ.

В тех случаях, когда требования ко времени, отводимому на поиск в таблице, менее жестки, чем по отношению к объему памяти, можно применять таблицу меньших размеров и пользоваться интерполяцией промежуточных значений функции.

В качестве примера процесса поиска в таблице рассмотрим систему, вариант функциональной схемы которой показан на рис. 1.28. Система содержит три измерителя потока, создающие дифференциальное напряжение, являющееся некоторой функцией потока. В состав системы входит АЦП, имеющий три дифференциальных входа, ОМЭВМ МК1816ВЕ48 и пульт управления с дисплеем. В системе использованы датчики потока, которые имеют нелинейные характеристики и требуют индивидуальной калибровки. Чтобы использовать метод табличного поиска и ОМЭВМ МК1816ВЕ48, можно откалибровать датчики потока, а результаты калибровочных проверок занести в виде таблиц в программируемое ПЗУ ОМЭВМ МК1816ВЕ48.

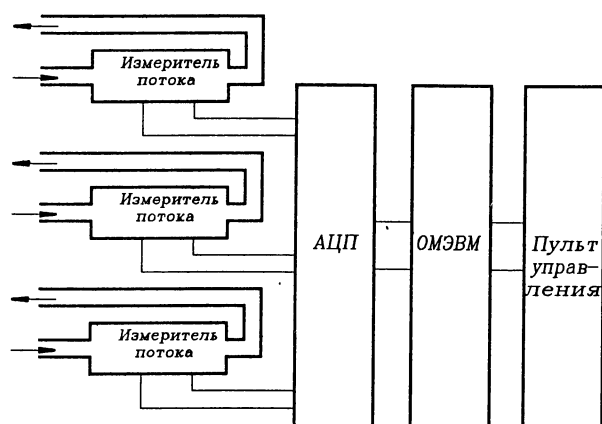


Рис. 1.28. Функциональная схема системы измерения на базе ОМЭВМ

Пример результатов калибровки одного из измерителей потока в виде графика показан на рис. 1.29. Эта же зависимость приведена в табл. 1.11. Напряжение с датчика представлено в шестнадцатеричном виде, что соответствует цифровым выходам АЦП.

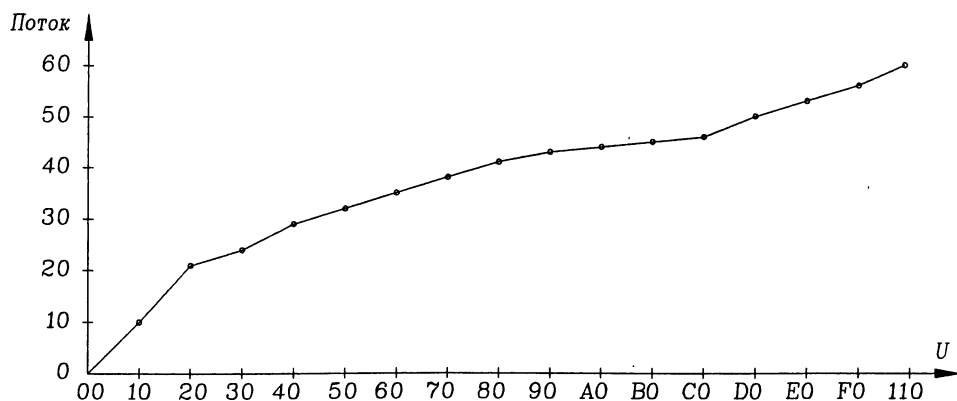


Рис. 1.29. Результаты калибровки измерителя потока

Восемь разрядов независимой переменной (напряжения) можно представить как два 4-разрядных поля. Старшие четыре разряда (7...4) будут использоваться для нахождения одного из табличных значений, младшие (3...0) — для интерполяции между этим значением и значением, которое отыскивается четырьмя старшими

разрядами следующей графы таблицы. Если обозначить старшие четыре разряда символом I, а младшие четыре разряда символом N, то функция интерполяции

$$F(X) = F(I) + N/16 [F(I+1) - F(I)],$$

где X - измеренное напряжение; F(X) — соответствующий поток.

Если, например, измеренное напряжение датчика равно 48Н, значение потока (см. табл. 1.11)

$$F(48) = 30' + 8/16 [34 - 30] = 32.$$

Как видно из формулы, кроме операций сложения и вычитания нужно еще выполнить умножение двух величин и деление их на 16. Умножение выполняется по специальной подпрограмме, а деление на 16 выполняется сдвигом вправо на четыре разряда после операции округления. Для вычисления функций двух переменных применяется интерполяция в двух измерениях. Использование таблиц дает разработчикам мощное средство вычисления функций.

Таблица 1.11

Наименование характеристики	Значение характеристики
Выходное напряжение датчика	00 10 20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0 100
Мощность потока (усл. ед.)	0 10 22 26 30 34 38 40 41 42 43 45 48 49 53 56 63

1.7.3. Реализация приема-передачи последовательного кода.

Последовательную передачу данных кодов КОИ8 можно реализовать на ОМЭВМ. Формат передаваемых данных, показанный на рис. 1.30, содержит стартовый бит (START), восемь бит данных и два стоповых бита (STOP).

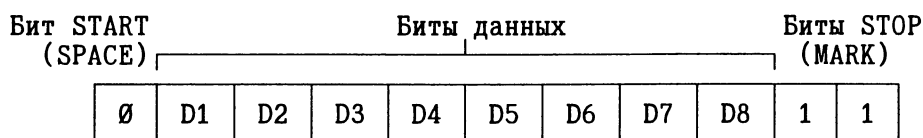


Рис. 1.30. Формат передаваемых данных

Алгоритм программы регистрации принимаемых последовательных импульсов, которая выполняется программными средствами ОМЭВМ, показан на рис. 1.31. Основной задачей этого алгоритма является регистрация каждого бита данных. В начале выполнения стандартной программы биты принимаемых данных регистрируются без интервалов (один за другим), пока на линии не появится импульс MARK. После обнаружения импульса MARK начинается второй цикл, в течение которого схема приема ожидает появления импульса SPACE. Цель этого алгоритма заключается в возможно более точном обнаружении переднего фронта бита START. Этот момент времени будет использован как точка отсчета для регистрации всех последующих битов символа. После обнаружения переднего фронта бита START устанавливается задержка (ожидание) в течение половины длительности приема одного бита (для удобства период следования входного сигнала обозначается через P).

Если на линии еще присутствует импульс SPACE, то бит START считается действительным и устанавливается временная задержка, равная длительности P. По окончании задержки P обрабатывается первый бит данных и устанавливается новая задержка, равная длительности P. Этот процесс повторяется до тех пор, пока не будут зарегистрированы все восемь бит данных. Последний принятый бит необходимо проверить, является ли он действительно битом STOP. Если проверка подтверждает бит STOP, считается, что символ достоверен, если нет, — произошла ошибка приема и, возможно, символ не достоверен.

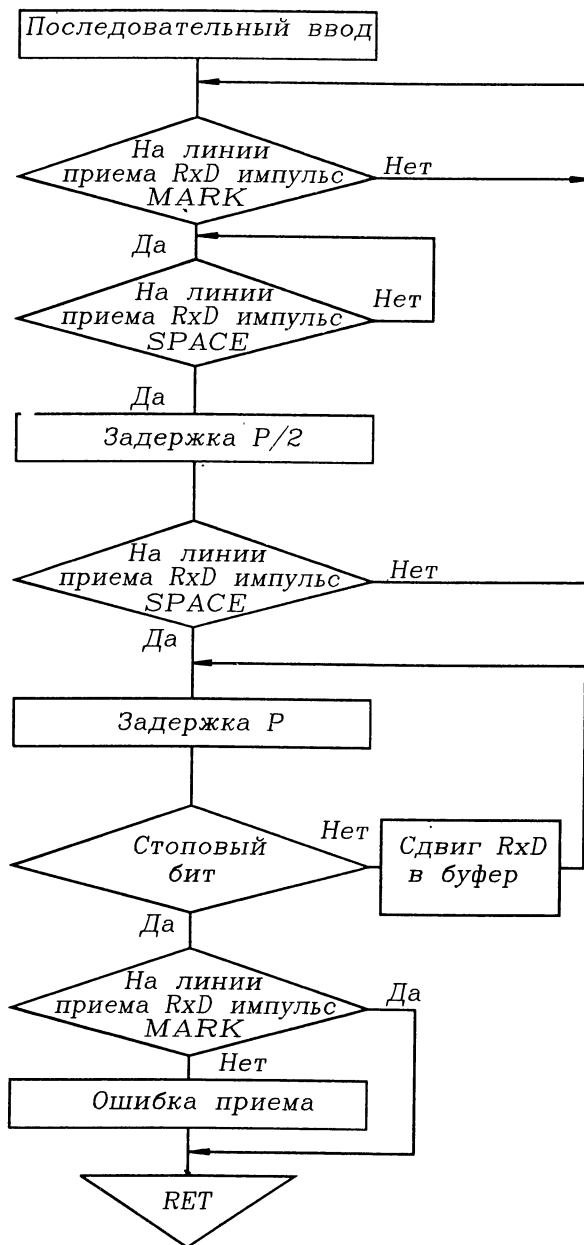


Рис. 1.31. Алгоритм программы регистрации принимаемых последовательных импульсов


```

ENTO CLK           ; разрешение синхроимпульсов
ORL P2,#00100000    ; начало сброса
MOV R2,#DELAY       ; подтверждение сигнала сброса
                    ; на время DELAY
LOOP: DJNZ R2,LOOP   ; проверка сброса
ANL P2,#11011111    ; окончание сброса

```

При работе ИС КР580ВВ51А в режиме синхронной передачи синхронизация обеспечивается модемом. В асинхронном режиме импульсы синхронизации приема и передачи должны вырабатываться отдельными схемами, связанными с ИС КР580ВВ51А.

Если ИС КР580ВВ51А находится в режиме X16 (в этом режиме частота синхроимпульсов приема и передачи в 16 раз больше, чем скорость передачи данных в бод), делением частоты 1,9968 МГц на 13 может быть получена частота синхроимпульсов приема и передачи, соответствующая скорости передачи данных 9600 бод. Дальнейшим делением частоты тактового сигнала можно получить импульсы для синхронизации передачи данных со скоростью 4800, 2400, 1200, 600 и 300 бод. Путем деления скорости передачи в 1200 бод на 11 может быть обеспечена скорость передачи 109,1 бод, которая всего на 1% отличается от стандартной частоты передачи, равной 110 бод.

Однокристалльная микроЭВМ КМ1816ВЕ48 осуществляет связь с ИС КР580ВВ51А в режиме распределения памяти (то есть так, как если бы ИС КР580ВВ51А была внешним ЗУ с произвольной выборкой) с помощью команды MOVX @Rn, A, которая записывает содержимое аккумулятора в ячейку внешнего ЗУ, адресуемую регистром Rn (n=0 или 1), и команды MOVX A, @Rn, которая пересылает данные из внешнего ЗУ в аккумулятор. Так как в ОМЭВМ используется мультиплексная передача данных и адресов по 8-разрядной шине, то для адресации ИС КР580ВВ51А через регистры R0 и R1 потребовался бы внешний регистр с фиксацией. Вместо этого для адресации ИС КР580ВВ51А использован ряд других контактов ОМЭВМ: вход CS ИС КР580ВВ51А соединен с седьмым разрядом порта P2 (P27), а линия C/D ИС КР580ВВ51А — с шестым разрядом порта P2 (P26). Это обеспечило возможность адресовать ИС КР580ВВ51А без использования регистров R0 и R1.

В том случае, если к ОМЭВМ подключается несколько ИС КР580ВВ51А или требуются другие периферийные ИС (например, таймер), целесообразно изменить схему (рис. 1.32) для обеспечения возможности использования регистров R0 и R1 для адресации периферийных устройств.

На рис. 1.33 показана измененная часть схемы с дополнительными элементами, позволяющая адресовать ИС КР580ВВ51А через регистры R0 и R1. Дополнительным элементом является 8-разрядный буферный регистр с фиксацией К580ИР82. В этот буфер загружается адрес с шины с помощью сигнала ALE, выдаваемого ОМЭВМ. В цикле чтения или записи из внешнего ЗУ этот адрес используется для адресации ИС КР580ВВ51А. В рассматриваемом варианте схема КР580ВВ51А выбирается любым адресом, в первом разряде которого записан "логический 0", а выбор режима управления (передачи данных) C/D будет определяться нулевым разрядом адреса, полученного из регистров-указателей R0 и R1.

1.7.4. Прием последовательного кода по прерываниям с использованием таймера ОМЭВМ

Рассмотренный выше метод последовательного ввода (рис. 1.31) требует значительных затрат мощности процессора на работу в цикле для реализации нужной временной задержки.

Использование внутреннего таймера для осуществления временной задержки приводит к значительной экономии времени обработки. Программно таймер устанавливается на заданный интервал и запускается, а процессор переходит к

выполнению других задач. Переполнение таймера приводит к инициализации прерывания, кроме того, информация об окончании данного интервала времени анализируется с помощью программного обеспечения. Алгоритм для схемы с использованием таймера приведен на рис. 1.34. Этот алгоритм аналогичен алгоритму на рис. 1.31 до того момента, где определяется передний фронт стартового бита. В этой точке таймер устанавливается на $1/2$ длительности прохождения одного бита (P), и происходит возврат к прерванной программе. При завершении данного интервала времени пополнение таймера приведет к инициализации прерывания. После обнаружения первого прерывания линия проверяется на состояние "0" (правильный стартовый бит). Если линия находится в состоянии "0", таймер устанавливается на временной интервал (для распознавания первого бита данных), и происходит возврат к программе, которая выполнялась в момент появления прерывания.

При обнаружении следующего прерывания происходит выборка данных, таймер реиницируется, а управление возвращается к выполнявшейся программе. Когда будет обнаружен последний бит (STOP), устанавливается флаг конца приема и происходит возврат к программе, выполнявшейся к моменту проявления пополнения таймера. Периодической проверкой флагов ошибки и конца приема программно можно определить, когда программа приема по прерываниям закончит прием символа. Этот способ реализации последовательного приема имеет следующие недостатки: для определения переднего фронта бита используется программный цикл, что приводит к потере обрабатывающей мощности процессора, несмотря на то, что задержка между приемом битов в данном алгоритме осуществляется таймером; повышаются требования к точности синхронизации при увеличении длины сообщений, поскольку при использовании описанного метода выборки накопленная ошибка может привести к ошибочному приему. Максимально допустимая погрешность синхронизации, при которой обеспечивается достоверный прием 11-разрядного символа в коде КОИ8, определяется из выражения:

$$E_{\max} = 0,5P / 11P = 4,5\%,$$

где P — длительность прохождения одного бита; $11P$ — длительность прохождения 11-разрядного символа.

Допустимая погрешность для 32-разрядного символа будет равна:

$$E_{\max} = 0,5P / 32P = 1,6\%.$$

Поскольку в этих вычислениях не учитывается искажение сигналов, в реальных условиях требуется более высокая стабильность синхроимпульсов или более гибкий алгоритм. Эта проблема усугубляется при возрастании скорости передачи, когда разрешающая способность счетчика (80 мкс при частоте синхронизации 6 МГц) становится сопоставимой с периодом принимаемого сигнала. Эффективное распознавание стартового бита и увеличение точности синхронизации могут быть обеспечены в том случае, если система на базе ОМЭВМ семейства МК48 сможет определить передний фронт бита START непосредственно в момент прихода принимаемых данных (RxD). Достичь этого можно, подав принимаемые данные на вход T1 ОМЭВМ. При этом необходимо настроить таймер/счетчик на режим подсчета внешних событий и загрузить его числом FFH. При организации схемы приема таким образом, передний фронт бита START будет вызывать прерывание по пополнению таймера/счетчика.

1.7.5. Реализация передачи последовательного кода.

Передача последовательного кода принципиально проще, чем прием, так как не требует синхронизации. При передаче последовательного кода требуется использование таймера для организации прерываний со скоростью передачи и выдачи подготовленного к передаче символа на контакты ввода-вывода.

Процесс передачи последовательного кода усложняется, если передача происходит одновременно с приемом. При этом необходимо решить вопрос совместного использования таймера. Одним из возможных решений этого вопроса

может быть использование таймера в качестве основного измерителя временных интервалов, управляющего программно-реализуемыми таймерами.

1.7.6. Программа реализации контроля по четности.

Организация контроля по четности требуется во многих системах связи. В ИС КР580ВВ51А предусмотрена возможность автоматического формирования контрольного разряда четности. При управлении передачей данных с помощью программных средств ОМЭВМ все вычисления, связанные с контролем по четности, должна осуществлять программа. Напомним, что символ является четным, если число единиц в его разрядах четное, и нечетным, если число единиц нечетное. Показанный ниже фрагмент программы можно использовать для вычисления контрольных сумм. В начале программы устанавливается счетчик циклов на восемь и сбрасывается флаг переноса. После этой подготовки идет повторение цикла восемь раз. В течение каждого выполнения в аккумуляторе происходит циклический сдвиг и проверяется последний значащий разряд. Если в этом разряде ноль, флаг переноса инвертируется, если единица — никаких последующих действий не требуется. Так как для 8-разрядного символа четное число нулей означает также и четное число единиц, после завершения всех восьми циклов сдвига разряд переноса устанавливается в "1", если в символе нечетное число единиц, и сбрасывается, если число единиц четное. Поскольку команда RR не связана с переносом, общим результатом выполнения этого цикла программы будет установка в единицу разряда переноса при нечетной контрольной сумме:

```
COUNT EQU R2
      MOV COUNT, #8
      CLR C
LOOP:  RR  A
      JB0 BYTE
      CPL C
      BYTE: DJNZ COUNT, LOOP
      END
```

ГЛАВА 2 ОДНОКРИСТАЛЬНЫЕ МИКРОЭВМ СЕМЕЙСТВА МК51

2.1. Общие сведения об однокристальных микроЭВМ семейства МК51

Восьмиразрядные высокопроизводительные однокристальные микроЭВМ (ОМЭВМ) семейства МК51 выполнены по высококачественной п-МОП технологии (серия 1816) и КМОП технологии (серия 1830).

Использование ОМЭВМ семейства МК51 по сравнению с МК48 обеспечивает увеличение объема памяти команд и памяти данных. Новые возможности ввода-вывода и периферийных устройств расширяют диапазон применения и снижают общие затраты системы. В зависимости от условий использования, быстродействие системы увеличивается минимум в два с половиной раза и максимум в десять раз.

Семейство МК51 включает пять модификаций ОМЭВМ (имеющих идентичные основные характеристики), основное различие между которыми состоит в реализации памяти программ и мощности потребления.

ОМЭВМ КР1816ВЕ51 и КР1830ВЕ51 содержат масочно-программируемое в процессе изготовления кристалла ПЗУ памяти программ емкостью 4096 байт и рассчитаны на применение в массовой продукции. За счет использования внешних микросхем памяти общий объем памяти программ может быть расширен до 64 Кбайт.

ОМЭВМ КМ1816ВЕ751 содержит ППЗУ емкостью 4096 байт со стиранием ультрафиолетовым излучением и удобна на этапе разработки системы при отладке программ, а также при производстве небольшими партиями или при создании систем, требующих в процессе эксплуатации периодической подстройки. За счет использования внешних микросхем памяти общий объем памяти программ может быть расширен до 64 Кбайт.

ОМЭВМ КР1816ВЕ31 и КР1830ВЕ31 не содержат встроенной памяти программ, однако могут использовать до 64 Кбайт внешней постоянной или перепрограммируемой памяти программ и эффективно использоваться в системах, требующих существенно большего по объему (чем 4 Кбайт на кристалле) ПЗУ памяти программ.

Каждая из перечисленных выше микросхем является соответственно аналогом БИС 8051, 80С51, 8751, 8031, 80С31 семейства MCS-51 фирмы Intel (США).

Сравнительные данные микросхем приведены в табл. 2.1.

Каждая ОМЭВМ рассматриваемого семейства содержит встроенное ОЗУ памяти данных емкостью 128 байт с возможностью расширения общего объема оперативной памяти данных до 64 Кбайт за счет использования внешних микросхем ЗУПВ.

Общий объем памяти ОМЭВМ семейства МК51 может достигать 128 Кбайт: 64 Кбайт памяти программ и 64 Кбайт памяти данных.

При разработке на базе ОМЭВМ более сложных систем могут быть использованы стандартные ИС с байтовой организацией, например, серии КР580. В дальнейшем обозначение "МК51" будет общим для всех моделей семейства, за исключением случаев, которые будут оговорены особо.

ОМЭВМ содержат все узлы, необходимые для автономной работы:

- 1) центральный восьмиразрядный процессор;
- 2) память программ объемом 4 Кбайт (только КМ1816ВЕ751, КР1816ВЕ51 и КР1830ВЕ51);
- 3) память данных объемом 128 байт;
- 4) четыре восьмиразрядных программируемых канала ввода-вывода;
- 5) два 16-битовых многорежимных таймера/счетчика;
- 6) систему прерываний с пятью векторами и двумя уровнями;
- 7) последовательный интерфейс;
- 8) тактовый генератор.

Система команд ОМЭВМ содержит 111 базовых команд с форматом 1, 2, или 3 байта.

Таблица 2.1.

Микросхемы	Аналог	Объем внутрен- ней па- мяти про- грамм, байт	Тип памяти про- грамм	Объем внут- ренней памяти данных, байт	Максималь- ная частота следования тактовых сигналов, МГц	Ток потреб- ления, мА
KP1816BE31	8031AH	—	внешн.	128	12,0	150,0
KP1816BE51	8051AH	4К	ПЗУ	128	12,0	150,0
KM1816BE751	8751H	4К	ППЗУ	128	12,0	220,0
KP1830BE31	80C31BH	—	внешн.	128	12,0	18,0
KP1830BE51	80C51BH	4К	ПЗУ	128	12,0	18,0

ОМЭВМ имеет:

- 32 РОН;
- 128 определяемых пользователем программно-управляемых флагов;
- набор регистров специальных функций.

РОН и определяемые пользователем программно-управляемые флаги расположены в адресном пространстве внутреннего ОЗУ данных. Регистры специальных функций (SFR, SPECIAL FUNCTION REGISTERS) с указанием их адресов приведены в таблице 2.1а.

Таблица 2 1а

Обозначение	Наименование	Адрес
* ACC	Аккумулятор	0E0H
* B	Регистр В	0F0H
* PSW	Регистр состояния программы	0D0H
SP	Указатель стека	81H
DPTR	Указатель данных. 2 байта:	
DPL	Младший байт	82H
DPH	Старший байт	83H
* P0	Порт 0	80H
* P1	Порт 1	90H
* P2	Порт 2	0A0H
* P3	Порт 3	0B0H
* IP	Регистр приоритетов прерываний	0B8H
* IE	Регистр разрешения прерываний	0A8H
TMOD	Регистр режимов таймера/счетчика	89H
* TCON	Регистр управления таймера/счетчика	88H
TH0	Таймер/счетчик 0. Старший байт	8CH
TL0	Таймер/счетчик 0. Младший байт	8AH
TH1	Таймер/счетчик 1. Старший байт	8DH
TL1	Таймер/счетчик 1. Младший байт	8BH
* SCON	Управление последовательным портом	98H
SBUF	Буфер последовательного порта	99H
PCON	Управление потреблением	87H

* — регистры, допускающие побитовую адресацию.

Ниже кратко описываются функции регистров, приведенных в таблице 2.1а. Подробно эти регистры рассматриваются в соответствующих разделах настоящего описания.

Аккумулятор. ACC — регистр аккумулятора. Команды, предназначенные для работы с аккумулятором, используют мнемонику "А", например, MOV A, P2. Мнемоника "ACC" используется, к примеру, при побитовой адресации аккумулятора. Так, символическое имя пятого бита аккумулятора при использовании ассемблера ASM51 будет следующим: ACC.5.

Регистр В. Используется во время операций умножения и деления. Для других инструкций регистр В может рассматриваться как дополнительный сверхоперативный регистр.

Регистр состояния программы. Регистр PSW содержит информацию о состоянии программы.

Указатель стека SP. 8-битовый регистр, содержимое которого инкрементируется перед записью данных в стек при выполнении команд PUSH и CALL. При начальном сбросе указатель стека устанавливается в 07H, а область стека в ОЗУ данных начинается с адреса 08H. При необходимости путем переопределения указателя стека область стека может быть расположена в любом месте внутреннего ОЗУ данных микроЭВМ.

Указатель данных. Указатель данных (DPTR) состоит из старшего байта (DPH) и младшего байта (DPL). Содержит 16-битовый адрес при обращении к внешней памяти. Может использоваться как 16-битовый регистр или как два независимых восьмибитовых регистра.

Порт0—Порт3. Регистрами специальных функций P0, P1, P2, P3 являются регистры—"защелки" соответственно портов P0, P1, P2, P3.

Буфер последовательного порта. SBUF представляет собой два отдельных регистра: буфер передатчика и буфер приемника. Когда данные записываются в SBUF, они поступают в буфер передатчика, причем запись байта в SBUF автоматически инициирует его передачу через последовательный порт. Когда данные читаются из SBUF, они выбираются из буфера приемника.

Регистры таймера. Регистровые пары (TH0, TL0) и (TH1, TL1) образуют 16-битовые счетные регистры соответственно таймера/счетчика 0 и таймера/счетчика 1.

Регистры управления. Регистры специальных функций IP, IE, TMOD, TCON, SCON содержат биты управления и биты состояния системы прерываний, таймеров/счетчиков и последовательного порта.

ОМЭВМ при функционировании обеспечивает:

- минимальное время выполнения команд сложения — 1 мкс;
- аппаратное умножение и деление с минимальным временем выполнения команд умножения/деления — 4 мкс

В ОМЭВМ предусмотрена возможность задания частоты внутреннего генератора с помощью кварца, LC-цепочки или внешнего генератора.

Архитектура семейства МК51 несмотря на то, что она основана на архитектуре семейства МК48, все же не является полностью совместимой с ней. В новом семействе имеется ряд новых режимов адресации, дополнительные инструкции, расширенное адресное пространство и ряд других аппаратных отличий. Расширенная система команд обеспечивает побайтовую и побитовую адресацию, двоичную и двоично-десятичную арифметику, индикацию переполнения и определения четности/нечетности, возможность реализации логического процессора.

Важнейшей и отличительной чертой архитектуры семейства МК51 является то, что АЛУ может наряду с выполнением операций над 8-разрядными типами данных манипулировать одноразрядными данными. Отдельные программно-доступные биты могут быть установлены, сброшены или заменены их дополнением, могут пересылаться, проверяться и использоваться в логических вычислениях. Тогда как поддержка простых типов данных (при существующей тенденции к увеличению длины слова) может с первого взгляда показаться шагом назад, это качество делает микроЭВМ семейства МК51 особенно удобными для применений, в которых

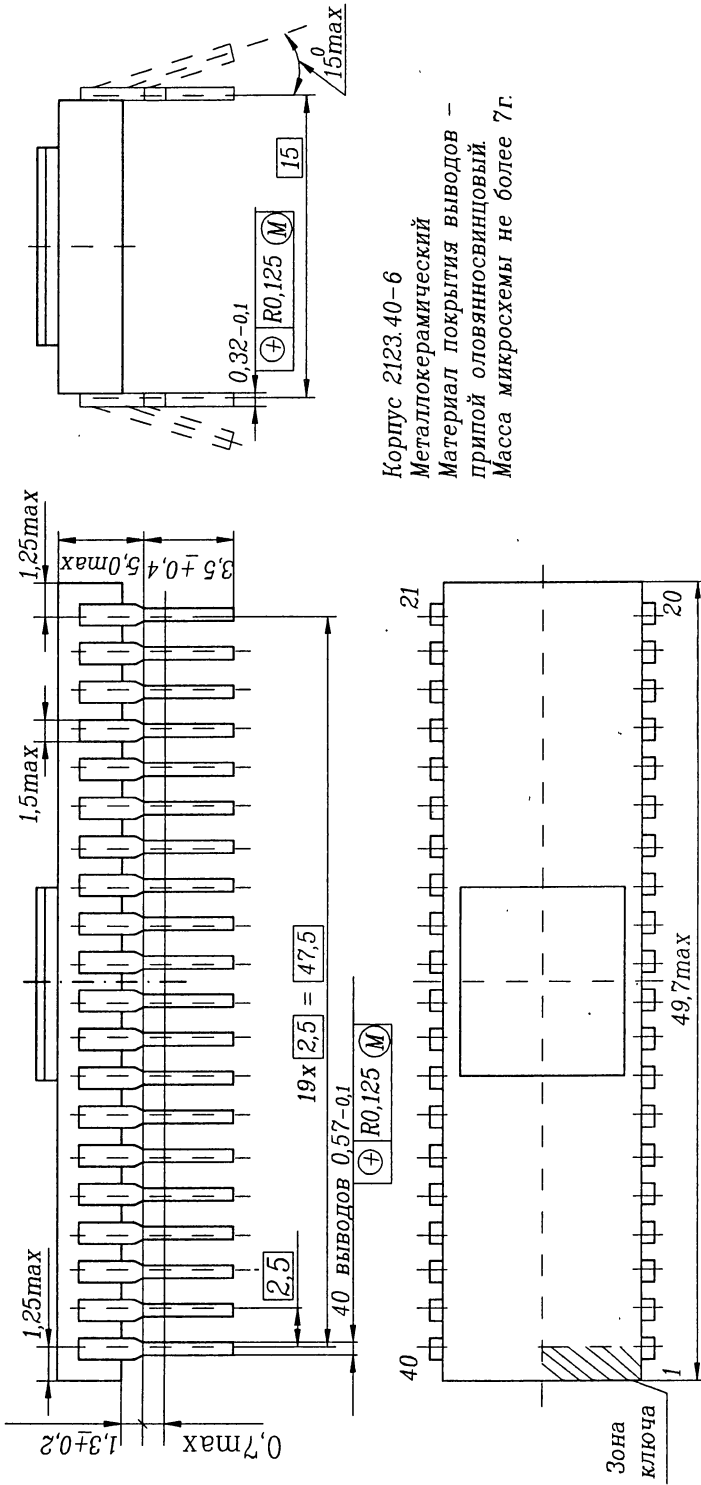


Рис. 2.1а. Конструктивное исполнение КМ1816ВЕ751

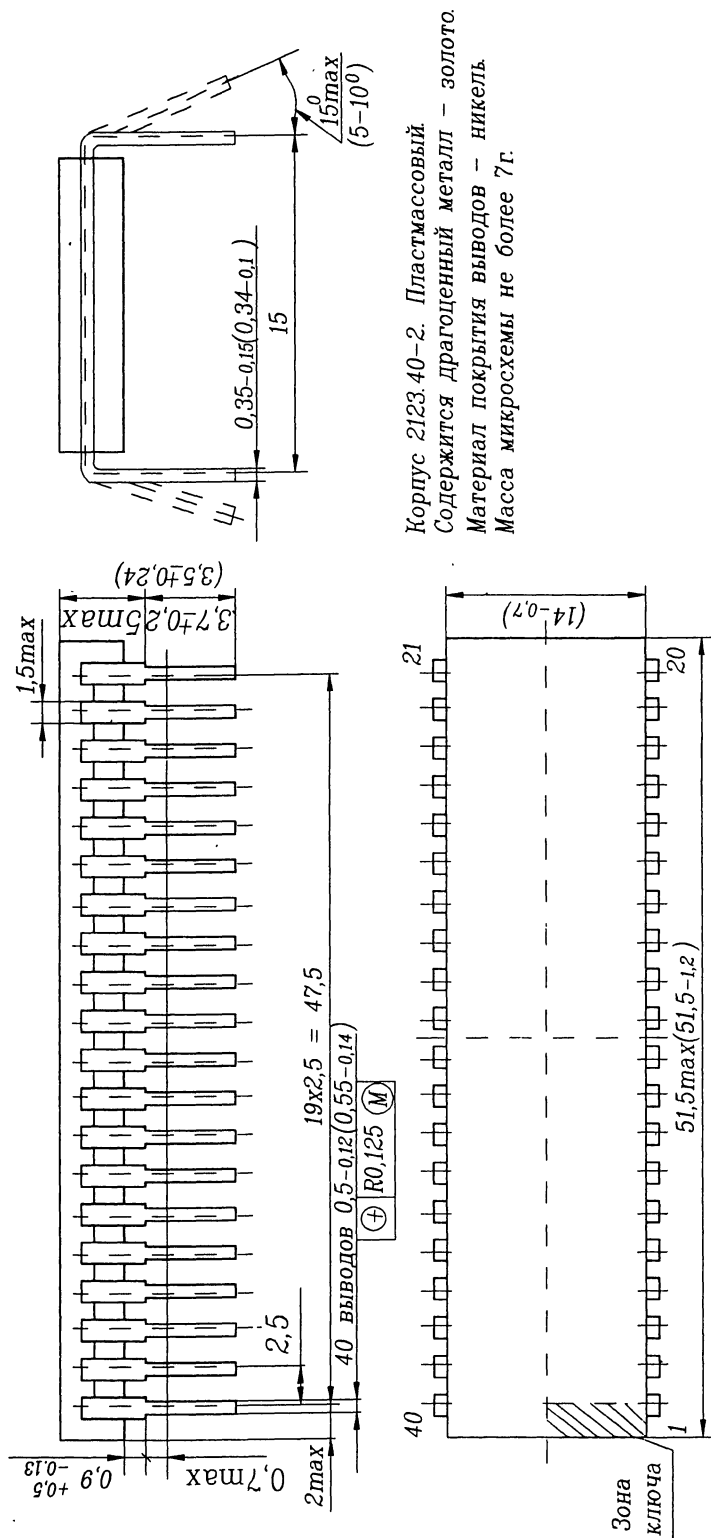


Рис. 2.16. Конструктивное исполнение КР1816ВЕ51, КР1830ВЕ51, КР1830ВЕ31

используются контроллеры. Алгоритмы работы последних по своей сути предполагают наличие входных и выходных булевых переменных, которые сложно реализовать при помощи стандартных микропроцессоров. Все эти свойства в целом называются булевым процессором семейства МК51. Благодаря такому мощному АЛУ набор инструкций микроЭВМ семейства МК51 одинаково хорошо подходит как для применений управления в реальном масштабе времени, так и для алгоритмов с большим объемом данных.

2.2. Структурная организация ОМЭВМ

Микросхемы семейства КМ1816ВЕ751 конструктивно выполнены в металлокерамическом корпусе типа 2123.40-6 с прозрачной для ультрафиолетового излучения крышкой (рис. 2.1а). Остальные рассматриваемые в данном описании ОМЭВМ семейства МК51 конструктивно выполнены в пластмассовых корпусах типа 2123.40-2 (рис. 2.1б). Условное графическое обозначение микросхем показано на рис. 2.1в, назначение выводов приведено в табл. 2.2.

ОМЭВМ, структурная схема которой представлена на рис. 2.2а и рис. 2.2б, состоит из следующих основных функциональных узлов: блока управления, арифметико-логического устройства, блока таймеров/счетчиков, блока последовательного интерфейса и прерываний, программного счетчика, памяти данных и памяти программ. Двусторонний обмен информацией между функциональными блоками осуществляется с помощью внутренней 8-разрядной магистрали данных.

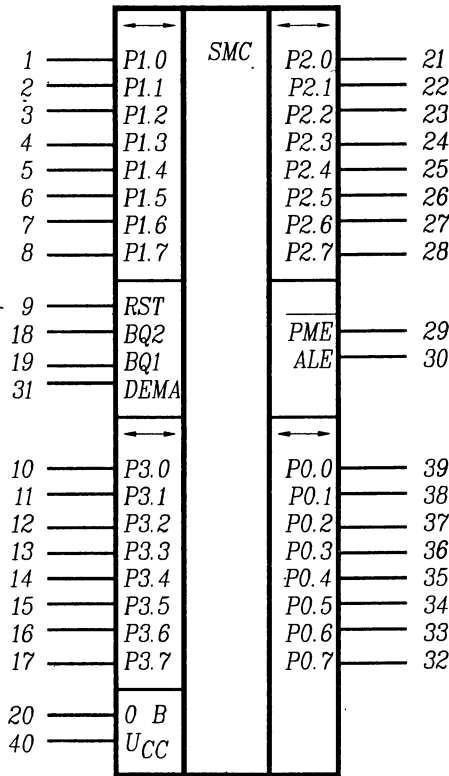


Рис. 2.1в. Условное графическое обозначение

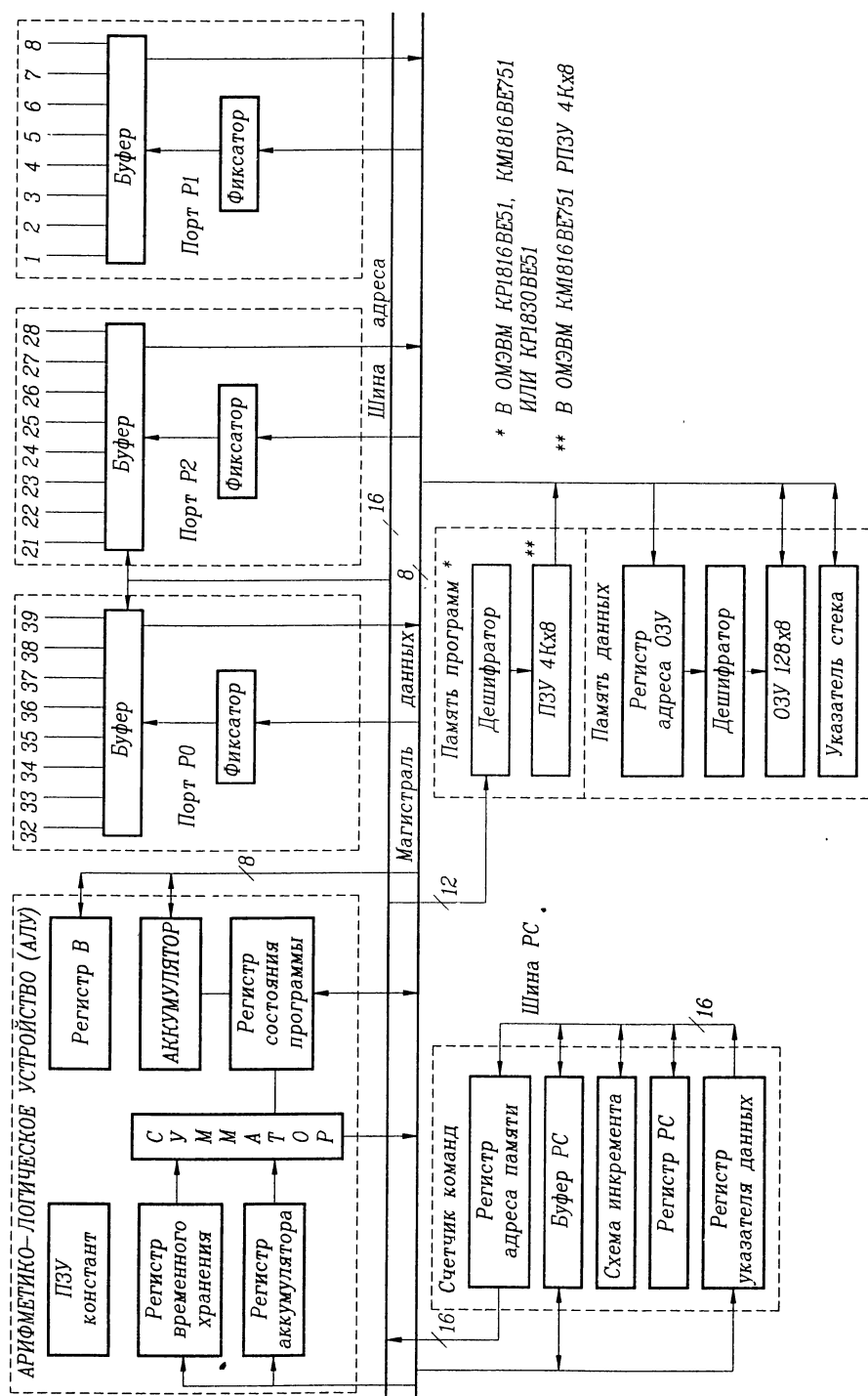


Рис. 2.2а. Структурная схема ОМЭВМ

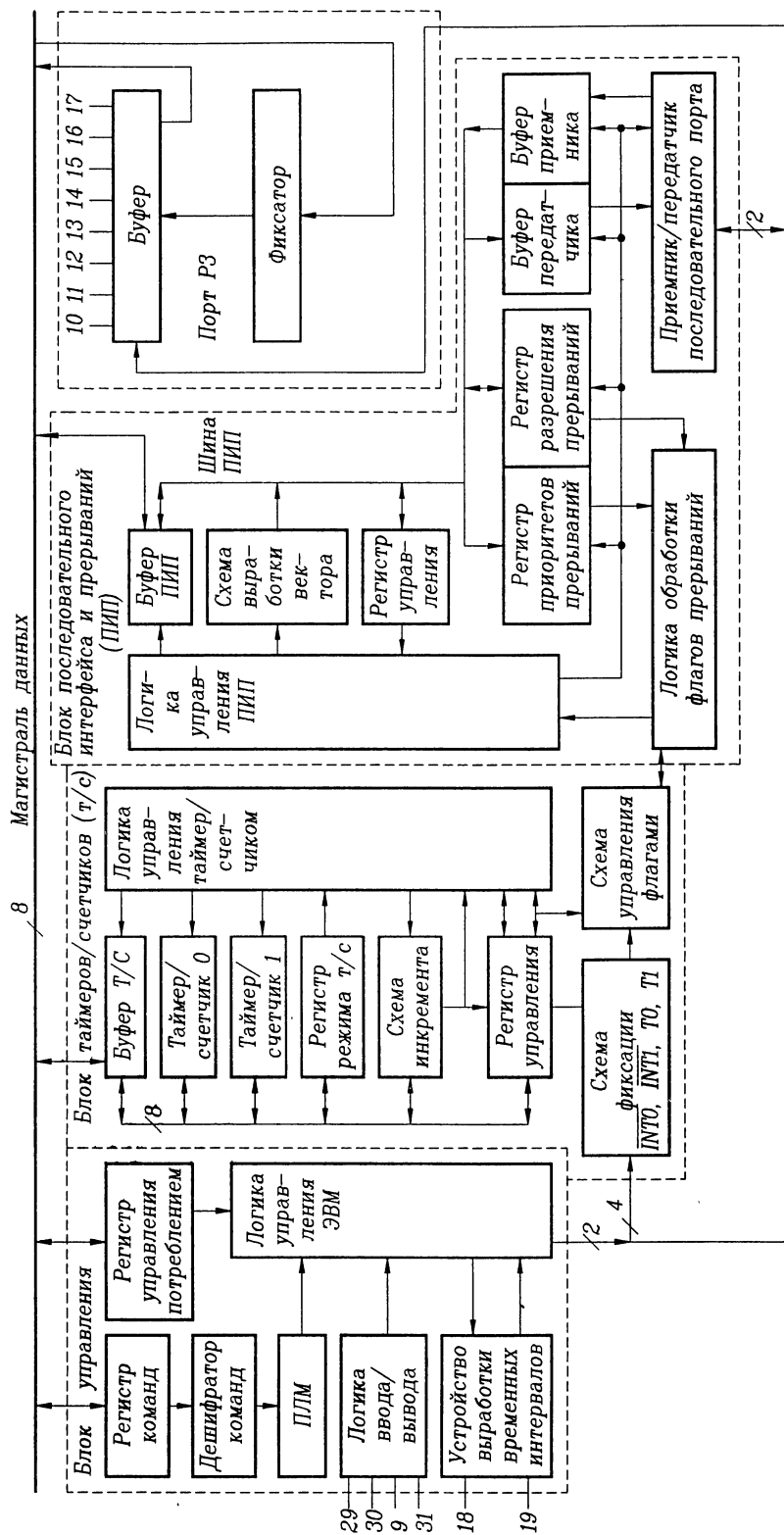


Рис. 2.26. Структурная схема ОМЭВМ

Таблица 2.2

№ вывода	Обозн.	Назначение	Тип
1-8	P1.0-P1.7	8-разрядный двунаправленный порт P1. Вход адреса A0-A7 при проверке внутреннего ПЗУ (РПЗУ).	вход/ выход
9	RST	Сигнал общего сброса. Вывод резервного питания ОЗУ от внешнего источника (для 1816)	вход
10-17	P3.0-P3.7	8-разрядный двунаправленный порт P3 с дополнительными функциями:	вход/ выход
	P3.0	Последовательные данные приемника - RxD	вход
	P3.1	Последовательные данные передатчика - TxD	выход
	P3.2	Вход внешнего прерывания 0 - $\overline{INT0}$	вход
	P3.3	Вход внешнего прерывания 1 - $\overline{INT1}$	вход
	P3.4	Вход таймера/счетчика 0: - T0	вход
	P3.5	Вход таймера/счетчика 1: - T1	вход
	P3.6	Выход стробирующего сигнала при записи во внешнюю память данных: - \overline{WR}	выход
	P3.7	Выход стробирующего сигнала при чтении из внешней памяти данных - \overline{RD}	выход
18 19	BQ2 BQ1	Выходы для подключения кварцевого резонатора.	выход вход
20	0 В	Общий вывод	
21-28	P2.0-P2.7	8-разрядный двунаправленный порт P2. Выход адреса A8-A15 в режиме работы с внешней памятью. В режиме проверки внутреннего ПЗУ выводы P2.0 - P2.6 используются как вход адреса A8-A14. Вывод P2.7 - разрешение чтения ПЗУ: - \overline{E}	вход/ выход
29	\overline{PME}	Разрешение программной памяти	выход
30	ALE	Выходной сигнал разрешения фиксации адреса. При программировании РПЗУ сигнал: - PROG	вход/ выход

Продолжение таблицы 2.2

№ вывода	Обозн.	Назначение	Тип
31	DEMA	Блокировка работы с внутренней памятью. При программировании РПЗУ подается сигнал U_{PR}	вход/ выход
32–39	P0.7–P0.0	8-разрядный двунаправленный порт P0. Шина адреса /данных при работе с внешней памятью. Выход данных D7–D0 в режиме проверки внутреннего ПЗУ (РПЗУ).	вход/ выход
40	U_{CC}	Вывод питания от источника напряжения +5 В	

2.2.1. Блок управления. Синхронизация микроЭВМ. Регистр PCON. Режимы уменьшенного энергопотребления

Блок управления предназначен для выработки синхронизирующих и управляющих сигналов, обеспечивающих координацию совместной работы блоков ОМЭВМ во всех допустимых режимах ее работы.

В состав блока управления входят: устройство выработки временных интервалов, логика ввода-вывода, регистр команд, регистр управления потреблением, дешифратор команд, ПЛМ и логика управления ЭВМ.

Устройство выработки временных интервалов предназначено для формирования и выдачи внутренних синхросигналов фаз, тактов и циклов. Количество машинных циклов определяет продолжительность выполнения команд. Практически все команды ОМЭВМ выполняются за один или два машинных цикла, кроме команд умножения MUL A, B и деления DIV A, B, продолжительность выполнения которых составляет четыре машинных цикла. Машинный цикл имеет фиксированную длительность и содержит шесть состояний S1–S6, каждое из которых по длительности соответствует такту, и, в свою очередь, состоит из двух временных интервалов, определяемых фазами P1 и P2. Длительность фазы равна периоду следования внешнего сигнала BQ, являющегося первичным сигналом синхронизации ОМЭВМ. Сигнал BQ вырабатывается либо встроенным тактовым генератором ОМЭВМ при подключении к ее выводам 18 (BQ2) и 19 (BQ1) кварцевого резонатора или LC-цепочки, либо внешним источником тактовых сигналов.

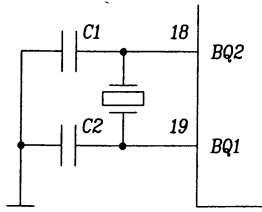
Схема подключения кварцевого резонатора и LC-цепочки к выводам ОМЭВМ BQ2 и BQ1 показана на рис. 2.2в. Схема подключения внешнего источника тактовых сигналов показана на рис. 2.2г. Источник тактовых сигналов должен обеспечивать следующие характеристики внешнего синхросигнала ОМЭВМ:

- длительность низкого уровня сигнала — не менее 20 нс;
- длительность высокого уровня сигнала — не менее 20 нс;
- времена фронтов нарастания и спада сигнала — не более 20 нс.

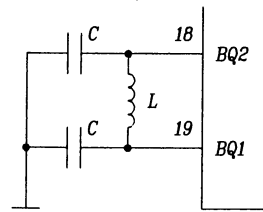
Как видно из рис. 2.2г, внешний синхросигнал для п-МОП ОМЭВМ (серия 1816) подается на вывод 18 (BQ2), а для КМОП ОМЭВМ (серия 1830) — на вывод 19 (BQ1). При этом необходимо обеспечивать требуемые уровни напряжения синхросигнала, значения которых приведены в таблице 2.24.

На рис. 2.2д показаны в общем виде внутренние генераторы п-МОП и КМОП ОМЭВМ семейства МК51. BQ1 и BQ2 являются соответственно входом и выходом инвертирующего усилителя, который может быть включен в режим генератора при подключении к выводам BQ1 и BQ2 резонатора или LC-цепочки (рис. 2.2в).

Показанный на рис. 2.2д сигнал \overline{PD} задается установкой одноименного бита в регистре PCON для перевода ОМЭВМ в режим микропотребления.



$C1, C2$ – емкость $30\text{пФ} \pm 10\text{пФ}$



$$f = \frac{1}{2\pi\sqrt{LC^1}} \quad C^1 = \frac{C+3C_{PP}}{2}$$

$C_{PP} \approx 10\text{пФ}$ – емкость вывода

Рис. 2.2в. Подключение кварцевого резонатора и LC-цепочки



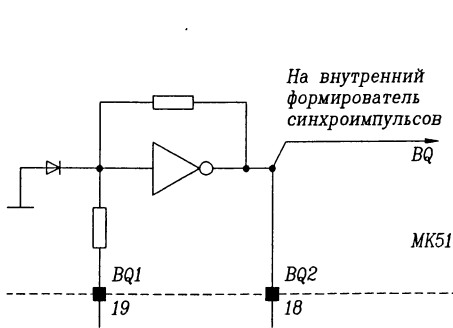
Для ОМЭВМ серии 1816



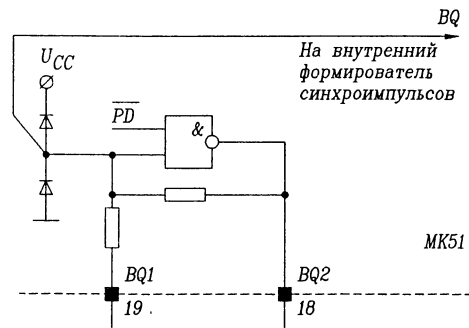
Вывод 18 (BQ2) оставить неподключенным

Для ОМЭВМ серии 1830

Рис. 2.2г. Подключение внешнего источника тактовых сигналов



Серия 1816 (п-МОП ОМЭВМ)



Серия 1830 (КМОП ОМЭВМ)

Рис. 2.2д. Внутренние генераторы ОМЭВМ МК51

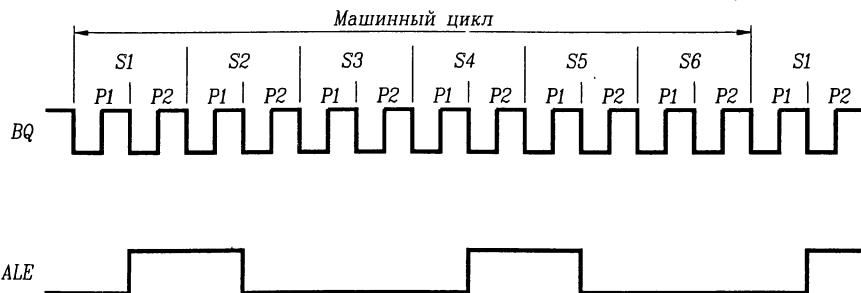


Рис. 2.2е. Диаграмма формирования машинных циклов ОМЭВМ

Рисунок 2.2е иллюстрирует формирование машинных циклов ОМЭВМ. Все машинные циклы ОМЭВМ одинаковы, состоят из 12 периодов сигнала BQ, начинаются фазой S1P1 и заканчиваются фазой S6P2. Дважды за один машинный цикл формируется сигнал ALE. На рис. 2.2з показана диаграмма внешнего синхросигнала ОМЭВМ.

Логика ввода-вывода предназначена для приема и выдачи сигналов, обеспечивающих обмен информацией ОМЭВМ с внешними устройствами через порты ввода-вывода P0—P3.

Регистр команд предназначен для записи и хранения 8-ми разрядного кода операции выполняемой команды, который с помощью дешифратора команд преобразовывается в 24-х разрядный код для ПЛМ, с помощью которой вырабатывается набор микроопераций в соответствии с микропрограммой выполнения команды. Регистр команд программно не доступен.

Конструкция регистра управления потреблением (PCON) определяется технологией изготовления ОМЭВМ: п-МОП или КМОП.

Для варианта изготовления по технологии п-МОП (серия 1816) регистр PCON имеет всего 1 бит, управляющий скоростью передачи последовательного порта SMOD.

Для варианта изготовления по технологии КМОП (серия 1830) обозначение разрядов регистра PCON приведено в таблице 2.3, а назначение разрядов в таблице 2.4.

Для п-МОП и КМОП ОМЭВМ расположение и назначение разряда SMOD идентичны.

Все биты регистра PCON программно доступны по записи ("0" и "1") и чтению.

Функции бита SMOD подробно рассмотрены при описании работы последовательного порта.

Таблица 2.3

Биты	7	6	5	4	3	2	1	0
Обозначение	SMOD	—	—	—	GF1	GF0	PD	IDL

Таблица 2.4

Биты	Наименов.	Назначение битов	Примечание
7	SMOD	Бит удвоения скорости передачи: при установке в "1" — скорость передачи удваивается	При работе последовательного порта
6	—	Резервный	
5	—	Резервный	
4	—	Резервный	
3	GF1	Флаг общего назначения	
2	GF0	Флаг общего назначения	
1	PD	Бит включения режима микропотребления "1" — режим микропотребления	Если в PD и IDL одновременно записана "1", преимущество имеет PD
0	IDL	Бит холостого хода "1" — режим холостого хода	

Биты PCON с номерами 4—6 зарезервированы для дальнейшего расширения семейства МК51. При чтении значение этих разрядов не определено. Программист не должен записывать "1" в эти биты, т. к. они могут использоваться в будущих разработках ОМЭВМ семейства МК51 для задания новых функций. В этом случае пассивное значение битов 4—6 будет "0", а активное — "1".

Биты GF1 и GF0 пользователь может задействовать по своему усмотрению.

В ОМЭВМ семейства МК51, выполненных по КМОП технологии, имеются два режима уменьшенного энергопотребления: режим холостого хода и режим микропотребления. Источником питания в этих режимах является вывод U_{CC} .

Режимы уменьшенного потребления для КМОП ОМЭВМ инициируются установкой битов PD и IDL в регистре PCON. Воздействие этих битов на аппаратуру ОМЭВМ показано на рис. 2.2ж.

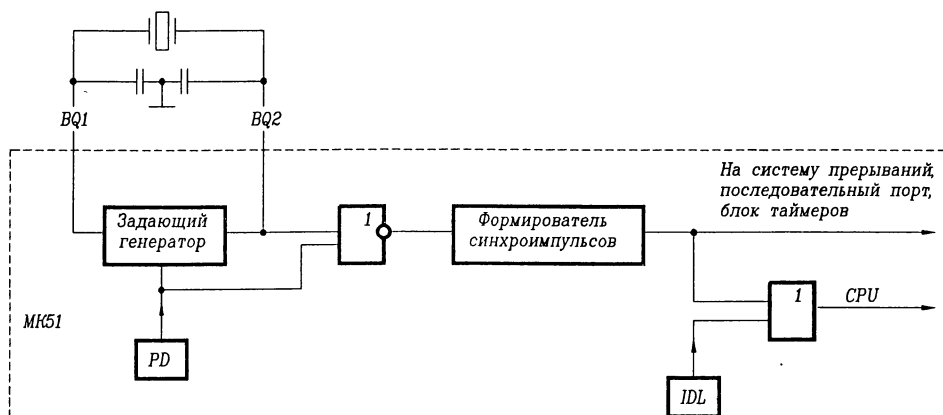


Рис. 2.2ж. Действие битов PD и IDL регистра PCON

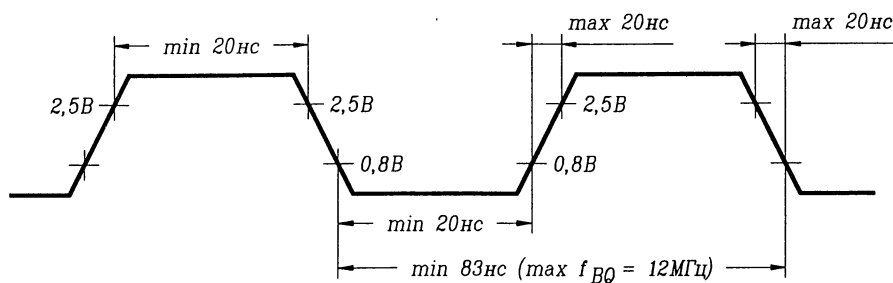


Рис. 2.2з. Диаграмма внешнего синхросигнала ОМЭВМ

Режим холостого хода

Инструкция, которая устанавливает $PCON.\bar{0}=1$ (IDL), является последней инструкцией, выполняемой перед переходом в режим холостого хода. В этом режиме блокируются функциональные узлы центрального процессора (CPU), что и уменьшает энергопотребление. Сохраняются состояния указателя стека, программного счетчика, PSW, аккумулятора и всех других регистров, а также внутреннего ОЗУ данных.

Для окончания режима холостого хода имеются два способа. Активизация любого разрешенного прерывания автоматически приведет к установке $PCON.\bar{0}=0$, оканчивая режим холостого хода. После исполнения команды RETI (выход из подпрограммы обслуживания прерывания) будет исполнена команда, которая следует за командой, переведшей ОМЭВМ в режим холостого хода.

Биты GF0 и GF1 удобно использовать для индикации режима, в котором была вызвана программа обработки прерывания: произошло это при нормальной работе ОМЭВМ или в режиме холостого хода. К примеру, команда, вызывающая режим холостого хода, может также устанавливать один или несколько флагов (GF0, GF1 или каких-либо других). Программа обработки прерывания, проверяя эти флаги, может определить предысторию своего вызова.

Другим способом окончания режима холостого хода является аппаратный сброс по входу RST длительностью не менее двух машинных циклов.

Активный сигнал сброса на выводе RST асинхронно сбрасывает бит IDL (PCON.0). Поскольку тактовый генератор работает, ОМЭВМ сразу после сброса IDL начинает выполнять программу с команды, следующей за командой, вызвавшей режим холостого хода. Между сбросом бита IDL и моментом, когда включится внутренний алгоритм сброса, может пройти до двух машинных циклов выполнения программы. Внутренние аппаратные средства ОМЭВМ блокируют доступ к внутренней памяти данных в течение указанного времени, но не блокируют доступ к портам. Если при этом изменение информации на портах нежелательно, то необходимо следить, чтобы за командой, которая устанавливает бит IDL, не следовала непосредственно команда, записывающая информацию в порт или во внешнюю память данных.

Режим микропотребления

Инструкция, которая устанавливает PCON.1=1 (PD), является последней выполняемой командой перед переходом в режим микропотребления. В этом режиме задающий генератор выключается, прекращая тем самым работу всех узлов ОМЭВМ и сохраняется только содержимое ОЗУ.

Единственным выходом из этого состояния является аппаратный сброс RST.

В этом режиме работы напряжение U_{CC} может быть уменьшено до 2 В и должно быть восстановлено до номинального перед выходом из режима микропотребления.

Сброс следует удерживать в активном состоянии не менее 10 мс (время восстановления работы задающего генератора).

При записи IDL=1 и PD=1 преимущество имеет бит PD.

В табл. 2.4а представлены состояния выводов ОМЭВМ в режимах холостого хода и микропотребления.

Таблица 2.4а. Состояния выводов ОМЭВМ в режимах холостого хода и микропотребления

Режим	Память программ	ALE	PWE	Порт P0	Порт P1	Порт P2	Порт P3
Холостого хода	Внутренняя	1	1	Данные	Данные	Данные	Данные
Холостого хода	Внешняя	1	1	Z	Данные	Адрес	Данные
Микропотребления	Внутренняя	0	0	Данные	Данные	Данные	Данные
Микропотребления	Внешняя	0	0	Z	Данные	Данные	Данные

Режим пониженного потребления для ОМЭВМ серии 1816 (п-МОП)

Во время нормальной работы внутреннее ОЗУ питается от U_{CC} . Однако для ОМЭВМ серии 1816 семейства МК51, если напряжение на выводе RST превышает U_{CC} , оно становится источником питания для ОЗУ. Это реализовано с помощью двух внутренних диодов, с катодов которых берется питание ОЗУ, а аноды подключены соответственно ко входу RST и к выводу питания ОМЭВМ U_{CC} . Необходимо подчеркнуть, что для КМОП ОМЭВМ данный режим отсутствует.

Логика управления ЭВМ в зависимости от режима работы ОМЭВМ вырабатывает необходимый набор управляющих сигналов.

2.2.2. Арифметико-логическое устройство (АЛУ). Регистр PSW

АЛУ представляет собой параллельное восьмиразрядное устройство, обеспечивающее выполнение арифметических и логических операций, а также операции логического сдвига, обнуления, установки и т. п.

АЛУ состоит из регистра аккумулятора, регистра временного хранения, ПЗУ констант, сумматора, дополнительного регистра (регистра В), аккумулятора, регистра состояния программы.

Регистр аккумулятора и регистр временного хранения — восьмиразрядные регистры, предназначенные для приема и хранения операндов на время выполнения операций над ними. Программно не доступны.

ПЗУ констант обеспечивает выработку корректирующего кода при двоично-десятичном представлении данных, кода маски при битовых операциях и кода констант.

Параллельный восьмиразрядный сумматор представляет собой схему комбинационного типа с последовательным переносом, предназначенную для выполнения арифметических операций сложения, вычитания и логических операций сложения, умножения, неравнозначности и тождественности.

Регистр В — восьмиразрядный регистр, используемый во время операций умножения и деления. Для других инструкций он может рассматриваться как дополнительный сверхоперативный регистр.

Аккумулятор представляет собой восьмиразрядный регистр, предназначенный для приема и хранения результата, полученного при выполнении арифметико-логических операций или операций пересылки.

Регистр состояния программы (PSW) предназначен для хранения информации о состоянии АЛУ при выполнении программы. Обозначение разрядов регистра PSW и назначение разрядов приведены соответственно в таблицах 2.5, 2.6.

Флаг переноса CY может устанавливаться и сбрасываться как аппаратными, так и программными средствами. Флаг CY может быть программно прочитан. Аппаратными средствами флаг CY устанавливается, если в старшем бите результата возникает перенос или заем. При выполнении операций умножения и деления флаг CY сбрасывается. Кроме того, флаг CY выполняет функции "булева аккумулятора" в командах, работающих с битами.

Флаг дополнительного переноса AC программно доступен по записи ("0" и "1") и чтению.

Флаги F0, RS1, RS0 программно доступны по записи ("0" и "1") и чтению.

Флаг переполнения OV программно доступен по записи ("0" и "1") и чтению. Устанавливается аппаратно, 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 аппаратно сбрасывается, а в случае деления на ноль устанавливается. При умножении флаг OV аппаратно устанавливается, если результат больше 255.

Флаг P является дополнением содержимого аккумулятора до четности. В 9-разрядном слове, состоящем из 8 разрядов аккумулятора и бита P, всегда содержится четное число единичных битов. В случае, если в аккумуляторе все разряды установлены в "0", флаг P примет нулевое значение. Программно доступен только по чтению.

2.2.3. Блок таймеров/счетчиков. Регистры TMOD и TCON

Таймеры/счетчики (T/C) предназначены для подсчета внешних событий, для получения программно управляемых временных задержек и выполнения времязадающих функций ОМЭВМ.

В состав блока T/C входят:

- 1) два 16-разрядных регистра T/C 0 и T/C 1;
- 2) восьмиразрядный регистр режимов T/C (TMOD);
- 3) восьмиразрядный регистр управления (TCON);
- 4) схема инкремента;

Таблица 2.5

Биты	7	6	5	4	3	2	1	0
Обозначение	CY	AC	F0	RS1	RS0	OV	—	P

Таблица 2.6

Биты	Наименов.	Назначение битов	Доступ к биту
7	CY	Флаг переноса. Изменяется во время выполнения некоторых арифметических и логических инструкций.	аппаратно или программно
6	AC	Флаг дополнительного переноса. Аппаратно устанавливается/сбрасывается во время выполнения инструкций сложения или вычитания для указания переноса или заема в бите 3 при образовании младшего полубайта результата (D0-D3).	аппаратно или программно
5	F0	Флаг 0. Флаг состояния определяемый пользователем.	программно
4	RS1	Указатель банка рабочих регистров	программно
3	RS0	Указатель банка рабочих регистров	программно
	RS1 RS0		
	0 0	Банк 0 с адресами (00H - 07H)	
	0 1	Банк 1 с адресами (08H - 0FH)	
	1 0	Банк 2 с адресами (10H - 17H)	
	1 1	Банк 3 с адресами (18H - 1FH)	
2	OV	Флаг переполнения. Аппаратно устанавливается/сбрасывается во время выполнения арифметических инструкций для указания состояния переполнения	аппаратно или программно
1	—	Резервный. Содержит триггер, доступный по записи ("0" и "1") и чтению, который можно использовать	
0	P	Бит четности. Аппаратно сбрасывается/устанавливается в каждом цикле инструкций для указания четного/нечетного количества разрядов аккумулятора, находящихся в состоянии "1".	аппаратно или программно

5) схема фиксации $\overline{INT0}$, $\overline{INT1}$, $T0$, $T1$;

6) схема управления флагами;

7) логика управления Т/С.

Два 16-разрядных регистра Т/С 0 и Т/С 1 выполняют функцию хранения содержимого счета. Каждый из них состоит из пары восьмиразрядных регистров, соответственно $TH0$, $TL0$ и $TH1$, $TL1$. Причем регистры $TH0$, $TH1$ — старшие, а регистры $TL0$, $TL1$ — младшие 8 разрядов. Каждый из восьмиразрядных регистров имеет свой адрес и может быть использован как РОН, если Т/С не используются (бит $TR0$ для Т/С 0 и бит $TR1$ для Т/С 1 в регистре управления $TCON$ равны "0").

Код величины начального счета заносится в регистры Т/С программно. В процессе счета содержимое регистров Т/С инкрементируется. Признаком окончания счета, как правило, является переполнение регистра Т/С, т. е. переход его содержимого из состояния "все единицы" в состояние "все нули". Все регистры $TH0$, $TH1$, $TL0$, $TL1$ доступны по чтению, и, при необходимости, контроль достижения требуемой величины счета может выполняться программно.

Регистр режимов Т/С ($TMOD$) предназначен для приема и хранения кода, определяющего:

- один из 4-х возможных режимов работы каждого Т/С;
- работу в качестве таймеров или счетчиков;
- управление Т/С от внешнего вывода.

Обозначение разрядов регистра $TMOD$ приведено в таблице 2.7. Назначение разрядов регистра $TMOD$ приведено в таблице 2.8.

Таблица 2.7

Биты	7	6	5	4	3	2	1	0
Обозн.	$GATE1$	$C/T1$	$M1.1$	$M0.1$	$GATE0$	$C/T0$	$M1.0$	$M0.0$

Таблица 2.8

Биты	Наименование	Назначение битов	Примечание															
0-1 4-5	M0-M1	<p>Определяют один из 4-х режимов работы, отдельно для Т/С 1 и Т/С 0</p> <table><tr><th>M1</th><th>M0</th><th>Режим</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>2</td></tr><tr><td>1</td><td>1</td><td>3</td></tr></table>	M1	M0	Режим	0	0	0	0	1	1	1	0	2	1	1	3	Все биты устанавливаются программно; биты 0-3 определяют режим работы Т/С 0, биты 4-7 определяют режим работы Т/С 1.
M1	M0	Режим																
0	0	0																
0	1	1																
1	0	2																
1	1	3																
2,6	C/T 0 C/T 1	Определяют работу в качестве: C/T 0, C/T 1 = 0 - таймера C/T 0, C/T 1 = 1 - счетчика																
3,7	GATE	Разрешает управлять таймером от внешнего вывода (INT0 - для Т/С 0, INT1 - для Т/С 1). GATE = 0 - управление запрещено GATE = 1 - управление разрешено																

При работе в качестве таймера содержимое регистра Т/С инкрементируется в каждом машинном цикле, т. е. Т/С является счетчиком машинных циклов ОМЭВМ. Поскольку машинный цикл состоит из 12 периодов частоты синхронизации ОМЭВМ f_{BQ} , то частота счета в данном случае равна $f_{BQ}/12$.

При работе Т/С в качестве счетчика внешних событий содержимое регистра Т/С инкрементируется в ответ на переход из "1" в "0" сигнала на счетном входе ОМЭВМ. (вывод Т0 для Т/С 0 и вывод Т1 для Т/С 1). Счетные входы аппаратно проверяются в фазе S5P2 каждого машинного цикла. Когда проверки показывают высокий уровень на счетном входе в одном машинном цикле и низкий уровень в другом машинном цикле, регистр Т/С инкрементируется. Новое (инкрементированное) значение заносится в регистр Т/С в фазе S3P1 машинного цикла, непосредственно следующего за тем, в котором был обнаружен переход из "1" в "0" на счетном входе ОМЭВМ. Т. к. для распознавания такого перехода требуется два машинных цикла (24 периода частоты синхронизации ОМЭВМ f_{BQ}), то максимальная частота счета Т/С в режиме счетчика равна $f_{BQ}/24$.

Чтобы уровень сигнала на счетном входе был гарантировано зафиксирован, он должен оставаться неизменным в течение как минимум одного машинного цикла.

Регистр управления (TCON) предназначен для приема и хранения кода управляющего слова. Обозначение разрядов регистра TCON приведено в табл. 2.9. Назначение разрядов регистра TCON приведено в табл. 2.10.

Флаги переполнения TF0 и TF1 устанавливаются аппаратно при переполнении соответствующих Т/С (переход Т/С из состояния "все единицы" в состояние "все нули"). Если при этом прерывание от соответствующего Т/С разрешено, то установка флага TF вызовет прерывание. Флаги TF0 и TF1 сбрасываются аппаратно при передаче управления программе обработки соответствующего прерывания.

Флаги TF0 и TF1 программно доступны и могут быть установлены/сброшены программой. Используя этот механизм, прерывания по TF0 и TF1 могут быть вызваны (установка TF) и отменены (сброс TF) программой.

Флаги IE0 и IE1 устанавливаются аппаратно от внешних прерываний (соответственно входы ОМЭВМ INT0 и INT1) или программно и инициируют вызов программы обработки соответствующего прерывания. Сброс этих флагов выполняется аппаратно при обслуживании прерывания только в том случае, когда прерывание было вызвано по фронту сигнала. Если прерывание было вызвано уровнем сигнала на входе INT0 (INT1), то сброс флага IE должна выполнять программа обслуживания прерывания, воздействуя на источник прерывания для снятия им запроса.

Схема инкремента предназначена:

— для увеличения на 1 в каждом машинном цикле содержимого регистров Т/С 0, Т/С 1, для которых установлен режим таймера и счет разрешен;

— для увеличения на 1 содержимого регистров Т/С 0, Т/С 1, для которых установлен режим счетчика, счет разрешен и на соответствующем входе ОМЭВМ (Т0 для Т/С 0 и Т1 для Т/С 1) зафиксирован счетный импульс.

Схема фиксации INT0, INT1, T0, T1 представляет собой четыре триггера. В каждом машинном цикле в момент S5P2 в них запоминается информация с выводов ОМЭВМ INT0, INT1, T0, T1.

Схема управления флагами вырабатывает и снимает флаги переполнения Т/С и флаги запросов внешних прерываний.

Логика управления Т/С синхронизирует работу регистров Т/С 0 и Т/С 1 в соответствии с запрограммированными режимами работы и синхронизирует работу блока Т/С с работой ОМЭВМ.

Таблица 2.9

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

Таблица 2.10

Биты	Наименование	Назначение битов	Примечание
6 4	TR1 TR0	Биты выключения Т/С, отдельно для Т/С 0 и Т/С 1. TR=0 – выключен, TR=1 – включен.	Биты устанавливаются и сбрасываются программно. Доступны по чтению.
7 5	TF1 TF0	Флаги переполнения Т/С.	Биты сбрасываются и устанавливаются аппаратно и программно. Доступны по чтению.
2 0	IT1 IT0	Биты, определяющие вид прерывания по входам $\overline{INT0}$, $\overline{INT1}$: IT=0 – прерывание по уровню (низкому) IT=1 – прерывание по фронту (переход из "1" в "0")	Биты устанавливаются и сбрасываются программно. Доступны по чтению.
3 1	IE1 IE0	Флаги запроса внешних прерываний по входам $\overline{INT0}$, $\overline{INT1}$	Биты сбрасываются и устанавливаются аппаратно и программно. Доступны по чтению.
			Биты 4,5 относятся к Т/С 0; биты 6, 7 – к Т/С 1. Биты 0,1 определяют внешние прерывания по входу $\overline{INT0}$, биты 2,3 – по входу $\overline{INT1}$.

Режимы работы Т/С

Режим работы каждого Т/С определяется значением битов $M0$, $M1$ в регистре TMOD. Т/С 0 и Т/С 1 имеют четыре режима работы. Режимы работы 0, 1, 2 одинаковы для обоих Т/С; Т/С 0 и Т/С 1 в этих режимах полностью независимы друг от друга. Работа Т/С 0 и Т/С 1 в режиме 3 различна. При этом установка режима 3 в Т/С 0 влияет на режимы работы Т/С 1.

Установка битов $M0=0$, $M1=0$ определяет режим работы 0. Т/С в режиме 0 представляет собой устройство на основе 13-разрядного регистра и функционально совместим с таймером/счетчиком семейства МК48 (восьмиразрядный таймер/счетчик с предделителем на 32).

13-разрядный регистр состоит для Т/С 0 из 8 разрядов регистра TH0 и 5 младших разрядов регистра TL0, а для Т/С 1 — из 8 разрядов регистра TH1 и 5 младших разрядов регистра TL1.

В этом режиме функцию делителя на 32 выполняют регистры TL0, TL1. Они являются программно доступными, но надо помнить, что значащими в режиме 0 являются только пять младших разрядов регистров TL0, TL1. Логика работы в режиме 0 на примере Т/С 1 показана на рис. 2.3. Для Т/С 0 логика работы аналогична. На рис. 2.3—2.6 OSC — источник синхронизации ОМЭВМ (внутренний или внешний). На выходе OSC — частота f_{BQ} . Бит С/Т регистра TMOD определяет

работу Т/С или в качестве таймера ($C/T=0$), или в качестве счетчика ($C/T=1$). Счет начинается при установке бита TR регистра TCON в состояние "1". При необходимости управления счетом извне бит GATE регистра TMOD устанавливается в состояние "1". Тогда при TR=1 счет будет разрешен, если на входе INT0 (для Т/С 0) или INT1 (для Т/С 1) установлено состояние "1" и будет запрещен, если установлено состояние "0". Установка бита TR0 для Т/С 0 и TR1 для Т/С 1 в состояние "0" выключает Т/С независимо от состояния других битов.

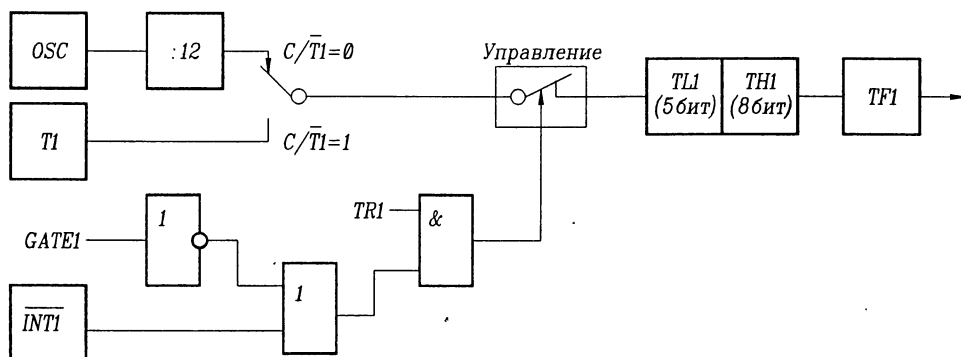


Рис. 2.3 Логика работы Т/С 1 в режиме 0

При переполнении Т/С (переход содержимого регистра Т/С из состояния "все единицы" в состояние "все нули") устанавливается флаг TF0 для Т/С 0 или TF1 для Т/С 1 в регистре TCON.

Установка битов M1=0, M0=1 определяет режим работы 1. Режим 1 аналогичен режиму 0. Отличие состоит в том, что установка режима 1 превращает Т/С в устройство на основе 16-разрядного регистра. Для Т/С 0 регистр состоит из программно доступных пар TL0, TH0, для Т/С 1 из программно доступных пар TL1, TH1. Логика работы в режиме 1 на примере Т/С 1 показана на рис. 2.4.

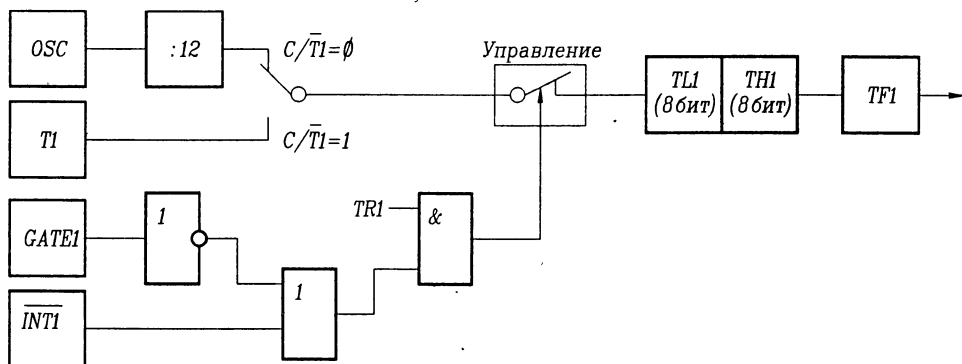


Рис. 2.4. Логика работы Т/С 1 в режиме 1

Установка битов M1=1, M0=0 определяет режим 2. В этом режиме Т/С представляет собой устройство на основе восьмизначного регистра TL0 для Т/С 0 и TL1 для Т/С 1. При каждом переполнении TL0 кроме установки в регистре TCON флага TF0 происходит автоматическая перезагрузка содержимого из TH0 в TL0. Соответственно для Т/С 1 при переполнении TL1 в регистре TCON устанавливается флаг TF1 и происходит перезагрузка TL1 из TH1. Регистры TH0 и TH1 загружаются программно. Перезагрузка TL0 из TH0 и TL1 из TH1 не влияет на содержимое регистров TH0 и TH1. Логика работы Т/С 1 в режиме 2 показана на рис. 2.5. Логика работы Т/С 0 в режиме 2 аналогична. Назначение битов управления TR0, TR1, GATE0, GATE1, C/T 0, C/T 1 такое же как режиме 0.

Установка битов $M1=1$, $M0=1$ определяет режим 3. Т/С 1 в режиме 3 заблокирован и просто сохраняет свой счет (значение кода в регистре Т/С). Эффект такой же, как при установке $TR1=0$.

Т/С 0 в режиме 3 представляет собой два независимых устройства на основе восьмизрядных регистров $TL0$ и $TH0$. Устройство на основе регистра $TL0$ может работать в режиме таймера и в режиме счетчика. За ним сохраняются все биты управления Т/С 0, оно реагирует на воздействия по входам $T0$, $\overline{INT0}$. При переполнении $TL0$ устанавливается флаг $TF0$. Устройство на основе регистра $TH0$ может работать только в режиме таймера. Оно использует бит включения $TR1$, при переполнении $TH0$ выставляет флаг $TF1$. Других битов управления устройство на основе $TH0$ в этом режиме не имеет. Логика работы Т/С 0 в режиме 3 показана на рис. 2.6.

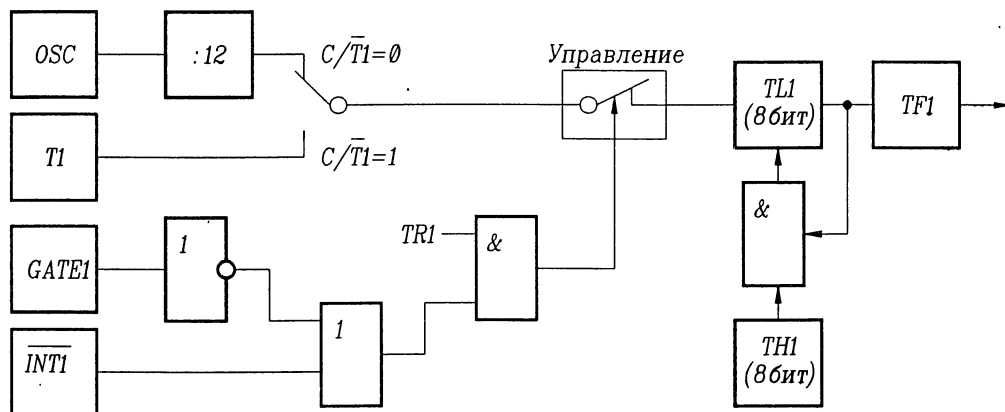


Рис. 2.5. Логика работы Т/С 1 в режиме 2

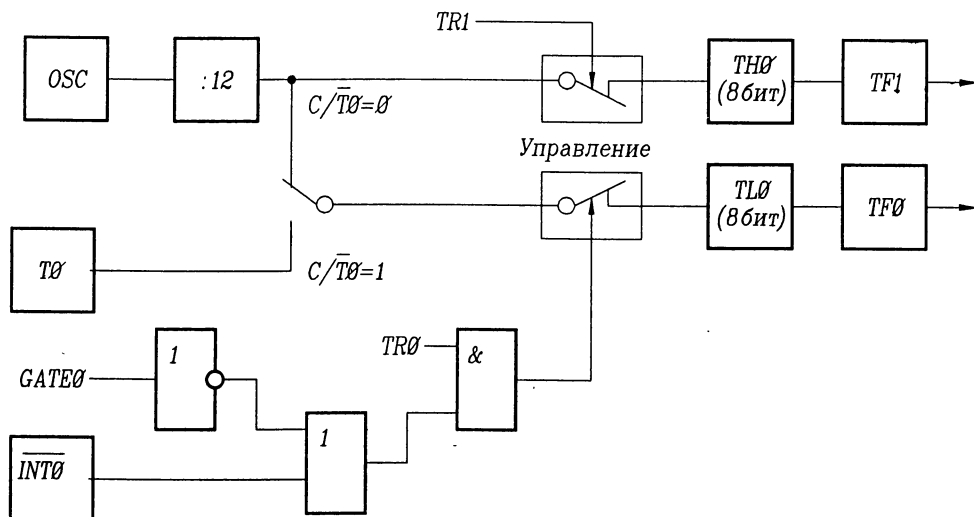


Рис. 2.6. Логика работы Т/С 0 в режиме 3

Установка Т/С 0 в режим 3 лишает Т/С 1 бита включения $TR1$. Поэтому Т/С 1 в режимах 0, 1, 2 при $GATE1=0$ всегда включен и при переполнении в режимах 0 и 1 Т/С 1 обнуляется, а в режиме 2 перезагружается не устанавливая флаг, если Т/С 0 находится в режиме 3. Управление от входов $\overline{INT1}$, $T1$, биты управления $C/T1$, $GATE1$ для Т/С 1 не зависят от режима Т/С 0.

Т/С 1 аппаратно связан с блоком синхронизации последовательного интерфейса (ПИ). При работе в режимах 0, 1, 2 при переполнении Т/С 1 всегда

вырабатывает импульс тактировки ПИ. Поэтому 3-й режим Т/С 0 удобно применять тогда, когда требуется работа ПИ и двух таймеров или ПИ, таймера и счетчика.

Когда Т/С 0 переведен в режим 3, Т/С 1 можно выключить, переведя его также в режим 3, использовать с последовательным портом для выработки импульсов тактировки или в любых других приложениях, не требующих прерывания.

Дополнительная информация. Выключение Т/С с помощью битов TR0, TR1 (сброс этих битов в "0") или с помощью входов ОМЭВМ INT0, INT1 (установка на этих входах логического "0" при GATE=1) не искажает код, находящийся в регистре Т/С. Т/С можно выключить, через произвольное время вновь включить и счет начнется с той величины, которая была в регистре Т/С на момент выключения (если, конечно, после выключения регистр Т/С не перезаписывался)

Новая загрузка Т/С сразу же означает новую величину счета, а старая теряется. Если загрузка произведена при включенном Т/С, счет продолжится с новой величины. Очередность загрузки регистров TL0, TH0, TL1, TH1 — произвольная.

Во всех режимах, кроме режима 2, после переполнения Т/С счет продолжается с величины 00H, если Т/С не выключить с помощью битов TR0, TR1 или входов INT0, INT1.

2.2.4. Блок последовательного интерфейса и прерываний. Регистры SC0N, IP, IE

Блок последовательного интерфейса и прерываний (ПИП) предназначен для организации ввода-вывода последовательных потоков информации и организации системы прерывания программ.

В состав блока ПИП входят: буфер ПИП, логика управления ПИП, регистр управления, буфер передатчика, буфер приемника, приемник/передатчик последовательного порта, регистр приоритетов прерываний, регистр разрешения прерываний, логика обработки флагов прерываний и схема выработки вектора.

Буфер ПИП обеспечивает побайтовый обмен информацией между внутренней магистралью данных и шиной ПИП.

Логика управления ПИП предназначена для выработки сигналов управления, обеспечивающих четыре режима работы последовательного интерфейса, и организации прерывания программ.

Последовательный интерфейс (последовательный порт) МК51 может работать в следующих четырех режимах:

Режим 0. Информация передается и принимается через вход приемника RxD (вывод P3.0 ОМЭВМ). Через выход передатчика TxD (вывод P3.1 ОМЭВМ) выдаются импульсы синхронизации, стробирующие каждый передаваемый или принимаемый бит информации. Формат посылки — 8 бит. Частота приема и передачи — $f_{BQ}/12$, где f_{BQ} — тактовая частота ОМЭВМ.

Режим 1. Информация передается через выход передатчика TxD, а принимается через вход приемника RxD. Формат посылки — 10 бит: старт-бит (ноль), восемь бит данных и стоп-бит (единица). Частота приема и передачи задается Т/С 1.

Режим 2. Информация передается через выход передатчика TxD, а принимается через вход приемника RxD. Формат посылки — 11 бит: старт-бит (ноль), восемь бит данных, программируемый девятый бит и стоп-бит (единица). Передаваемый девятый бит данных принимает значение бита TB8 из регистра специальных функций SC0N. Бит TB8 в регистре SC0N может быть программно установлен в "0" или в "1", или в него, к примеру, можно поместить значение бита P из регистра PSW для повышения достоверности принимаемой информации (контроль по паритету). При приеме девятый бит данных принятой посылки поступает в бит RB8 регистра SC0N. Частота приема и передачи в режиме 2 задается программно и может быть равна $f_{BQ}/32$ или $f_{BQ}/64$.

Режим 3. Режим 3 полностью идентичен режиму 2 за исключением частоты приема и передачи, которая в режиме 3 задается Т/С 1.

Подробно работа последовательного порта описана в разделе 2.3.3.

Регистр управления (SCON) предназначен для приема и хранения кода восьмибитового слова, управляющего последовательным интерфейсом. Обозначение разрядов регистра SCON приведено в таблице 2.11. Все разряды регистра SCON программно доступны по записи ("0" и "1") и чтению.

Разряды SM0, SM1 определяют режим работы ПИП, как указано в табл. 2.12. Остальные биты регистра имеют следующее назначение:

SM2 — разрешение многопроцессорной работы. В режимах 2 и 3 при SM2=1 флаг RI не активизируется, если девятый принятый бит данных равен "0". В режиме 1 при SM2=1 флаг RI не активизируется, если не принят стоп-бит, равный "1". В режиме 0 бит SM2 должен быть установлен в "0".

REN — разрешение приема последовательных данных. Устанавливается и сбрасывается программным обеспечением соответственно для разрешения и запрета приема.

TB8 — девятый бит передаваемых данных в режимах 2 и 3. Устанавливается и сбрасывается программным обеспечением.

RB8 — девятый бит принятых данных в режимах 2 и 3. В режиме 1, если SM2=0, RB8 является принятым стоп-битом. В режиме 0 бит RB8 не используется.

TI — флаг прерывания передатчика. Устанавливается аппаратно в конце времени выдачи 8-го бита в режиме 0 или в начале стоп-бита в других режимах. Сбрасывается программным обеспечением.

RI — флаг прерывания приемника. Устанавливается аппаратно в конце времени приема 8-го бита в режиме 0 или через половину интервала стоп-бита в режимах 1, 2, 3 при SM2=0. При SM2=1 см. описание для бита SM2.

Таблица 2.11

Биты	7	6	5	4	3	2	1	0
Обозначение	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Таблица 2.12

SM0	SM1	Режим	Наименование	Скорость передачи
0	0	0	Сдвигový регистр	$f_{BQ}/12$
0	1	1	8-битовый универсальный асинхронный приемник/передатчик (УАПП)	переменная, задается T/C 1
1	0	2	9-битовый (УАПП)	$f_{BQ}/64$ или $f_{BQ}/32$
1	1	3	9-битовый (УАПП)	переменная, задается T/C 1

Буфер передатчика предназначен для приема с шины ПИП параллельной информации и выдачи ее в виде последовательного потока символов на передатчик последовательного порта.

Буфер приемника предназначен для приема данных в виде последовательного потока символов с последовательного порта, преобразования их в параллельный вид, хранения и выдачи в параллельном виде на внутреннюю шину ПИП.

Буфер приемника и буфер передатчика при программном доступе имеют одинаковое имя (SBUF) и адрес (99H). Если команда использует SBUF как регистр источника, то обращение происходит к буферу приемника. Если команда использует SBUF как регистр назначения, то обращение происходит к буферу передатчика.

Во всех четырех режимах работы последовательного порта передача инициируется любой командой, которая использует SBUF как регистр назначения. Прием в режиме 0 инициируется условием RI=0 и REN=1. В остальных режимах прием инициируется приходом старт-бита, если REN=1.

Приемник/передатчик последовательного порта предназначен для приема последовательного потока символов со входа последовательного порта, выделения данных и выдачи их в буфер приемника, а также для приема последовательных данных с буфера передатчика, преобразования их в последовательный поток символов и выдачи его на выход последовательного порта.

Регистр приоритетов прерываний (IP) предназначен для установки уровня приоритета прерывания для каждого из пяти источников прерываний. Обозначение разрядов регистра IP показано в таблице 2.13, а их назначение указано ниже.

PX0 — установка уровня приоритета прерывания от внешнего источника $\overline{INT0}$.

PT0 — установка уровня приоритета прерывания от T/C 0.

PX1 — установка уровня приоритета прерывания от внешнего источника $\overline{INT1}$.

PT1 — установка уровня приоритета прерывания от T/C 1.

PS — установка уровня приоритета прерывания от последовательного порта.

X — резервный разряд.

Таблица 2.13

Биты	7	6	5	4	3	2	1	0
Обозначение	X	X	X	PS	PT1	PX1	PT0	PX0

Наличие в разряде IP "1" устанавливает для соответствующего источника высокий уровень приоритета, а наличие в разряде IP "0" — низкий уровень приоритета. При чтении резервных разрядов соответствующие линии магистрали данных не определены. Пользователь не должен записывать "1" в резервные разряды, т. к. они зарезервированы под дальнейшее расширение семейства МК51.

Регистр разрешения прерываний (IE) предназначен для разрешения или запрещения прерываний от соответствующих источников. Обозначение разрядов регистра IE показано в таблице 2.14, а их назначение указано ниже.

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

X — резервный разряд.

ES — управление прерыванием от последовательного порта. ES=1 — разрешение. ES=0 — запрещение.

ET1 — управление прерыванием от T/C 1. ET1=1 — разрешение. ET1=0 — запрещение.

EX1 — управление прерыванием от внешнего источника $\overline{INT1}$. EX1=1 — разрешение. EX1=0 — запрещение.

ET0 — управление прерыванием от T/C 0. ET0=1 — разрешение. ET0=0 — запрещение.

EX0 — управление прерыванием от внешнего источника $\overline{INT0}$. EX0=1 — разрешение. EX0=0 — запрещение.

При чтении резервных разрядов соответствующие линии магистрали не определены. Пользователь не должен записывать "1" в резервные разряды, т. к. они зарезервированы под дальнейшее расширение семейства МК51.

Таблица 2.14

Биты	7	6	5	4	3	2	1	0
Обозначение	EA	X	X	ES	ET1	EX1	ET0	EX0

Логика обработки флагов прерываний осуществляет приоритетный выбор запроса прерывания, сброс его флага и инициирует выработку аппаратно реализованной команды перехода на подпрограмму обслуживания прерывания.

Схема выработки вектора прерывания вырабатывает двухбайтовые адреса подпрограмм обслуживания прерывания в зависимости от источника прерываний, которые приведены в таблице 2.15.

Подробно система прерываний описана в разделе 2.3.4.

Таблица 2.15

Источник прерывания	Вектор прерывания
Внешнее прерывание $\overline{INT0}$	0003H
Таймер/счетчик Т/С 0	000BH
Внешнее прерывание $\overline{INT1}$	0013H
Таймер/счетчик Т/С 1	001BH
Последовательный порт	0023H

2.2.5. Счетчик команд. Регистр DPTR

Счетчик команд (PC) предназначен для формирования текущего 16-разрядного адреса программной памяти и 8/16-разрядного адреса внешней памяти данных.

В состав счетчика команд входят 16-разрядные буфер PC, регистр указателя данных DPTR, регистр PC, схема инкремента, регистр адреса памяти.

Буфер PC осуществляет связь между 16-разрядной шиной PC и восьмиразрядной магистралью данных, обеспечивая запись, хранение и коммутацию информации.

Регистр указателя данных (DPTR) предназначен для хранения 16-разрядного адреса внешней памяти данных. Состоит из двух восьмиразрядных регистров DPH и DPL, входящих в блок регистров специальных функций. Они программно доступны и могут использоваться в качестве двух независимых РОН, если нет необходимости в хранении 16-разрядного адреса внешней памяти данных.

В регистре PC хранится текущий 16-разрядный адрес памяти программ.

Схема инкремента увеличивает текущее значение 16-разрядного адреса памяти программ на единицу.

Регистр адреса памяти предназначен для записи и хранения исполнительного 16-разрядного адреса памяти программ или 8/16-разрядного адреса внешней памяти данных, а также для передачи данных на порт P0 при выполнении команд MOVX @Ri, A и MOVX @DPTR, A, обеспечивающих запись данных через порт P0 во внешние устройства.

2.2.6. Порты

Порты P0, P1, P2, P3 являются двунаправленными портами ввода-вывода и предназначены для обеспечения обмена информацией ОМЭВМ с внешними устройствами, образуя 32 линии ввода-вывода. Каждый из портов содержит фиксатор-защелку, который представляет собой восьмиразрядный регистр, имеющий

байтовую и битовую адресацию для установки (сброса) разрядов с помощью программного обеспечения.

Физические адреса фиксаторов $P0$, $P1$, $P2$, $P3$ составляют для:

$P0$ — $80H$, при битовой адресации $80H—87H$;

$P1$ — $90H$, при битовой адресации $90H—97H$;

$P2$ — $A0H$, при битовой адресации $A0H—A7H$;

$P3$ — $B0H$, при битовой адресации $B0H—B7H$.

Помимо работы в качестве обычных портов ввода/вывода линии портов $P0—P3$ могут выполнять ряд дополнительных функций, описанных ниже.

Через порт $P0$:

— выводится младший байт адреса $A0—A7$ при работе с внешней памятью программ и внешней памятью данных;

— выдается из ОМЭВМ и принимается в ОМЭВМ байт данных при работе с внешней памятью (при этом обмен байтом данных и вывод младшего байта адреса внешней памяти мультиплексированы во времени);

— задаются данные при программировании внутреннего ППЗУ и читается содержимое внутренней памяти программ.

Через порт $P1$:

— задается младший байт адреса при программировании внутреннего ППЗУ и при чтении внутренней памяти программ.

Через порт $P2$:

— выводится старший байт адреса $A8—A15$ при работе с внешней памятью программ и внешней памятью данных (для внешней памяти данных — только при использовании команд $MOVX A, @DPTR$ и $MOVX @DPTR, A$, которые вырабатывают 16-разрядный адрес);

— задается старший байт (разряды $A8—A14$) адреса при программировании внутреннего ППЗУ и при чтении внутренней памяти программ.

Каждая линия порта $P3$ имеет индивидуальную альтернативную функцию:

$P3.0$ — RxD , вход последовательного порта, предназначен для ввода последовательных данных в приемник последовательного порта;

$P3.1$ — TxD , выход последовательного порта, предназначен для вывода последовательных данных из передатчика последовательного порта;

$P3.2$ — $\overline{INT0}$, используется как вход 0 внешнего запроса прерывания;

$P3.3$ — $\overline{INT1}$, используется как вход 1 внешнего запроса прерывания;

$P3.4$ — $T0$, используется как вход счетчика внешних событий $T/C 0$;

$P3.5$ — $T1$, используется как вход счетчика внешних событий $T/C 1$;

$P3.6$ — \overline{WR} , строб записи во внешнюю память данных, выходной сигнал, сопровождающий вывод данных через порт $P0$ при использовании команд $MOVX @Ri, A$ и $MOVX @DPTR, A$.

$P3.7$ — \overline{RD} , строб чтения из внешней памяти данных, выходной сигнал, сопровождающий ввод данных через порт $P0$ при использовании команд $MOVX A, @Ri$ и $MOVX A, @DPTR$.

Альтернативная функция любой из линий порта $P3$ реализуется только в том случае, если в соответствующем этой линии разряде фиксатора-защелки содержится "1". В противном случае на линии порта $P3$ будет присутствовать "0".

На рис. 2.7—2.10 показаны функциональные схемы одного вывода в каждом из портов для серии 1816. Для серии 1830 конфигурация портов $P1—P3$ отличается только выходным усилительным каскадом, который представлен на рис. 2.11.

Электрические параметры портов $P0—P3$ приведены в табл. 2.24.

На рис. 2.7—2.10 фиксатор-защелка показан в виде D-триггера, тактируемого внутренним сигналом "Запись в защелку", который вырабатывается при записи данных в порт. Выход триггера Q может быть подключен на внутреннюю шину ОМЭВМ через буфер $B1$ сигналом "Чтение защелки", что обеспечивает возможность программного чтения содержимого фиксатора. Значение сигнала непосредственно на выводе порта может быть программно считано на внутреннюю шину ОМЭВМ через буфер $B2$, управляемый внутренним сигналом

"Чтение выводов". Часть команд ОМЭВМ при чтении порта активизируют сигнал "Чтение защелки", другая часть команд — сигнал "Чтение выводов".

Как показано на рис. 2.7 и 2.9, выходные каскады порта P0 (образованы транзисторами N1, N) и порта P2 (образованы транзисторами N1, N2, N) через мультиплексоры МХ могут подключаться либо к выходу защелок, либо к внутренним шинам "Адрес/данные" и "Адрес". Последнее используется при обращении к внешней памяти. Во время обращения к внешней памяти содержимое защелок порта P2 не изменяется. В аналогичной ситуации в защелки порта P0 всегда автоматически записываются "1" во все разряды.

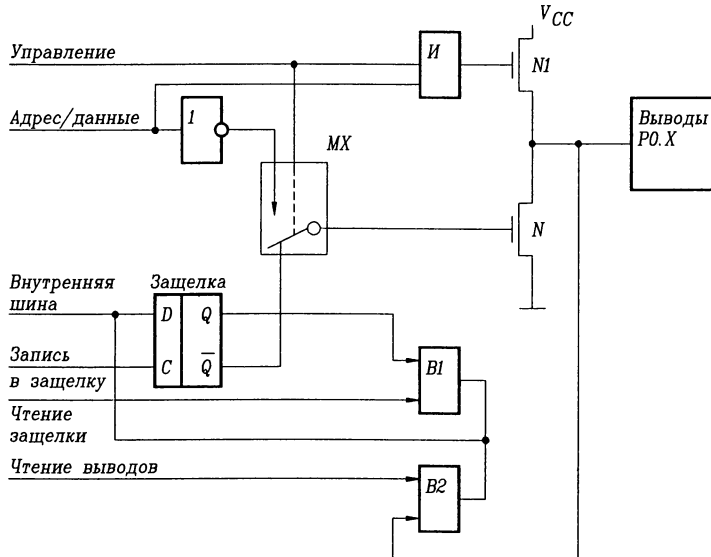


Рис. 2.7. Разряд порта 0

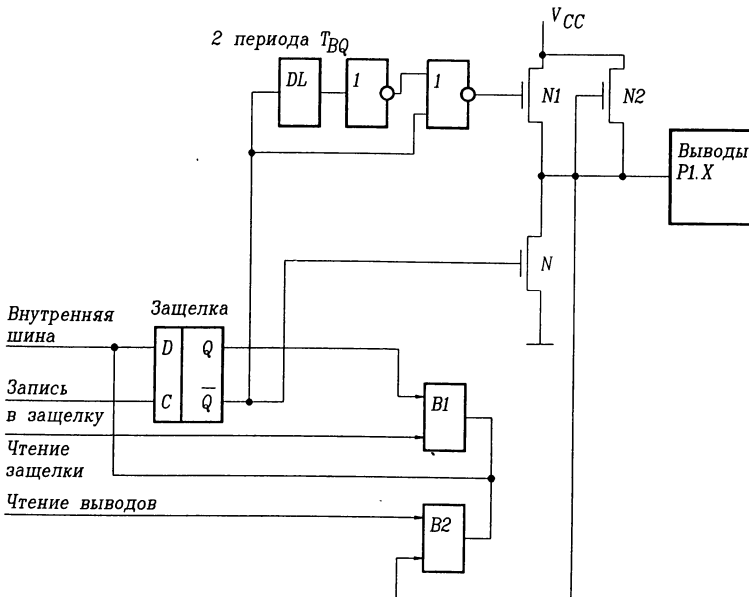


Рис. 2.8. Разряд порта 1

выход. Для использования вывода в качестве входа необходимо, чтобы его защелка содержала "1", которая при этом запирает выходной транзистор N. Из-за наличия внутреннего подтягивающего резистора выводы портов P1, P2, P3 в режиме "оборванный вход" имеют уровень "1". Благодаря этой особенности порты P1, P2, P3 иногда называют "квазидвунправленными".

Порт P0 не имеет внутренних подтягивающих резисторов (рис. 2.7). Транзистор N1 в выходном каскаде выводов порта P0 открыт только когда через эти выводы выдается "1" при обращениях к внешней памяти. Во всех других режимах работы транзистор N1 заперт. Таким образом, в случае использования порта P0 в качестве выходного порта общего назначения, необходимо устанавливать на его выводах внешние подтягивающие резисторы для задания уровня "1". Запись "1" в защелку вывода порта P0 закрывает транзистор N и при отсутствии внешнего подтягивающего резистора переводит вывод в высокоимпедансное состояние. При этом данный вывод может использоваться в качестве входа. Если порт P0 используется в качестве порта ввода/вывода общего назначения, каждый из его выводов может независимо от других работать как вход или как выход. Порт P0 является в чистом виде двунаправленным портом.

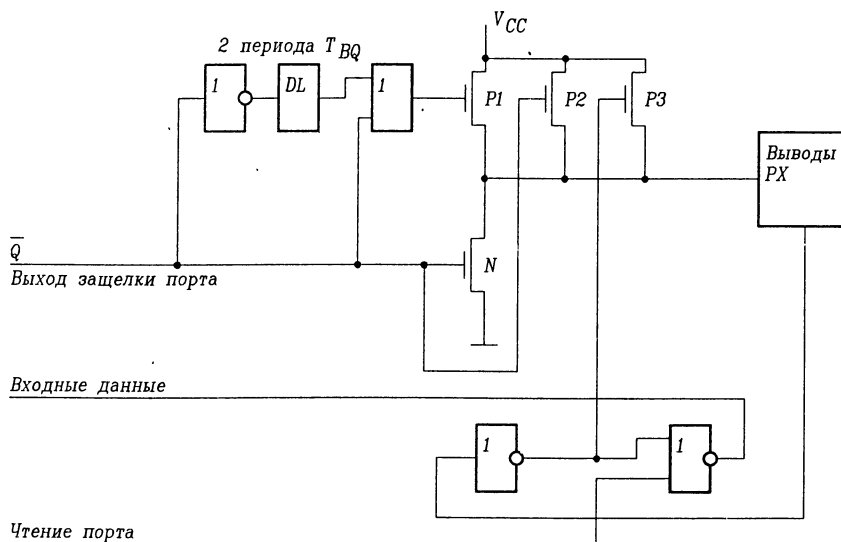


Рис. 2.11. Разряд порта 1—3 (серия 1830)

Все разряды фиксаторов-защелок портов Р0—Р3 по сбросу устанавливаются в "1". Если защелка вывода порта содержит "0", то для настройки данного вывода на ввод необходимо записать в защелку "1".

Процедура записи в порт

При выполнении команды записи в порт новое значение записывается в защелку в фазе S6P2 последнего машинного цикла команды. Однако, новое содержимое защелки выводится непосредственно на выходной контакт порта только в фазе S1P1 следующего машинного цикла.

При переходе выводов портов P1, P2 и P3 из состояния "0" в состояние "1" для уменьшения времени переключения используется дополнительный транзистор N1 (рис. 2.8—2.10), который включается на время, равное двум периодам T_{BQ} тактовой частоты ОМЭВМ f_{BQ} (транзистор N1 открыт в течение фаз S1P1 и S1P2 машинного цикла, в котором происходит смена состояния вывода порта). В открытом состоянии транзистор N1 обеспечивает ток приблизительно в 100 раз больший, чем постоянно открытый транзистор N2. В схемах на рис. 2.8—2.10 выдержка в два периода T_{BQ} обеспечивается элементом DL.

Как уже отмечалось, рассматривавшиеся до настоящего момента схемы на рис. 2.8—2.10 относятся к п-МОП ОМЭВМ семейства МК51 (серия 1816). Аналогичные структуры для КМОП ОМЭВМ семейства МК51 (серия 1830) отличаются только выходным каскадом.

Схема выходного каскада портов P1—P3 для КМОП ОМЭВМ семейства МК51 показана на рис. 2.11. Рассматриваемый выходной каскад содержит три р-канальных МОП транзистора (P1, P2, P3), подключенных к источнику питания, и один п-МОП транзистор, подключенный к общей шине. Можно отметить, что п-МОП транзисторы, используемые в выходных каскадах ОМЭВМ серий 1816 и 1830 открыты, когда на их затворах логическая "1" и закрыты, когда на их затворах логический "0". р-МОП транзисторы, наоборот, открыты, когда на их затворах "0" и закрыты, когда на затворах "1".

Транзистор P1 на рис. 2.11 включается на два периода T_{BQ} для уменьшения времени переключения при переходе вывода порта из состояния "0" в состояние "1". Как только открывается транзистор P1, сигнал с выхода порта через инвертор поступает на затвор транзистора P3 и также открывает его. Этот инвертор и транзистор P3 образуют триггер, который удерживает состояние логической "1" на выходе порта после того, как транзистор P1 закрывается. Транзистор P3 при этом играет роль подтягивающего резистора аналогично N2 на рис. 2.8—2.10. Однако, из рис. 2.11 видно, что если вывод порта находится в состоянии "1", то внешняя помеха, кратковременно устанавливающая "0" на данном выходе, способна запереть транзистор P3 и перевести вывод порта в высокоимпедансное состояние. Для предотвращения подобной ситуации служит транзистор P2, работающий в противофазе с транзистором P3. После исчезновения помехи транзистор P2 через инвертор вновь откроет транзистор P3. Мощность транзистора P2 приблизительно в 10 раз меньше мощности транзистора P3.

Режим "Чтение—Модификация—Запись"

Команды чтения портов ОМЭВМ делятся на две категории: команды, считывающие информацию с выходов защелок, и команды, считывающие информацию непосредственно с внешних контактов выводов порта. Команды, считывающие информацию с выходов защелок, реализуют так называемый режим "Чтение—Модификация—Запись", заключающийся в том, что команда считывает состояние защелки, при необходимости модифицирует полученное значение и записывает результат обратно в защелку.

Ниже приводятся команды, работающие в режиме "Чтение—Модификация—Запись". Во всех случаях, когда операндом и регистром назначения результата является порт или бит порта, команды считывают информацию с выходов защелок, а не с внешних контактов выводов порта.

ANL (логическое И, например, ANL P1,A)

ORL (логическое ИЛИ, например, ORL P2,A)

XRL (логическое ИСКЛЮЧАЮЩЕЕ ИЛИ, например, XRL P3,A)

JBC (переход, если бит = 1 и очистка бита, например, JBC P1.1,LABEL)

CPL (инверсия бита, например, CPL P3.0)

INC (инкремент, например, INC P2)

DEC (декремент, например, DEC P2)

DJNZ (декремент и переход, если не ноль, например, DJNZ P3,LABEL)

MOV PX.Y,C (пересылка бита переноса в бит Y порта X)

CLR PX.Y (очистка бита Y порта X)

SETB PX.Y (установка бита Y порта X)

Не очевидно, что последние три команды в приведенном списке работают в режиме "Чтение—Модификация—запись", однако, это так. Указанные команды считывают с порта весь байт целиком, модифицируют адресуемый бит, после чего записывают полученный новый байт обратно в фиксатор-защелку порта.

Чтение информации с выходов защелок, а не с внешних контактов выводов порта позволяет исключить возможную в ряде случаев неправильную интерпретацию уровня напряжения на выводе порта. К примеру, вывод порта может

использоваться для управления базой p-p-n транзистора. В этом случае, когда в защелку вывода порта записывается "1", транзистор открывается. Если после этого ОМЭВМ прочтает состояние внешнего контакта рассматриваемого вывода порта, то получит значение логического "0", т. к. на контакте в это время присутствует напряжение базы открытого транзистора. Чтение же выхода защелки покажет истинное значение сигнала на выводе порта, т. е. "1".

2.2.7. Память данных

Память данных предназначена для приема, хранения и выдачи информации, используемой в процессе выполнения программы. Память данных, расположенная на кристалле ОМЭВМ, состоит из регистра адреса ОЗУ, дешифратора, ОЗУ и указателя стека.

Регистр адреса ОЗУ предназначен для приема и хранения адреса выбираемой с помощью дешифратора ячейки памяти, которая может содержать как бит, так и байт информации.

ОЗУ представляет собой 128 восьмиразрядных регистров, предназначенных для приема, хранения и выдачи различной информации.

Указатель стека представляет собой восьмиразрядный регистр, предназначенный для приема и хранения адреса ячейки стека, к которой было последнее обращение. При выполнении команд LCALL, ACALL содержимое указателя стека увеличивается на 2. При выполнении команд RET, RETI содержимое указателя стека уменьшается на 2. При выполнении команды PUSH direct содержимое указателя стека увеличивается на 1. При выполнении команды POP direct содержимое указателя стека уменьшается на 1. После сброса в указателе стека устанавливается адрес 07H, что соответствует началу стека с адресом 08H.

В ОМЭВМ предусмотрена возможность расширения памяти данных путем подключения внешних устройств емкостью до 64 Кбайт. При этом обращение к внешней памяти данных возможно только с помощью команд MOVX.

Команды MOVX @Ri, A и MOVX A, @Ri формируют восьмиразрядный адрес, выдаваемый через порт P0. Команды MOVX @DPTR, A и MOVX A, DPTR формируют 16-разрядный адрес, младший байт которого выдается через порт P0, а старший — через порт P2.

Байт адреса, выдаваемый через порт P0, должен быть зафиксирован во внешнем регистре по спаду сигнала ALE, т. к. в дальнейшем линии порта P0 используются как шина данных, через которую байт данных принимается из памяти при чтении или выдается в память данных при записи. При этом чтение стробируется сигналом ОМЭВМ \overline{RD} , а запись — сигналом ОМЭВМ \overline{WR} . При работе с внутренней памятью данных сигналы \overline{RD} и \overline{WR} не формируются.

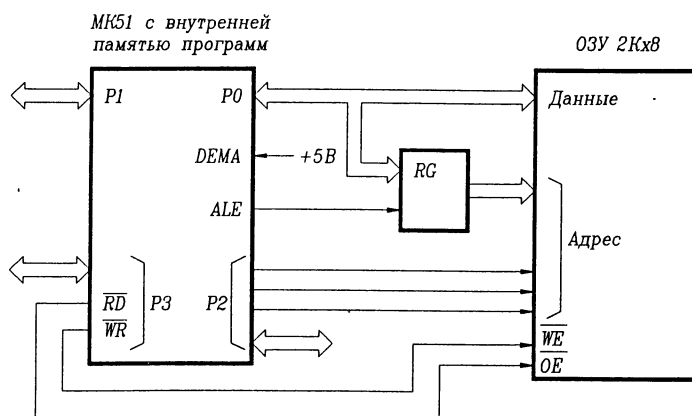
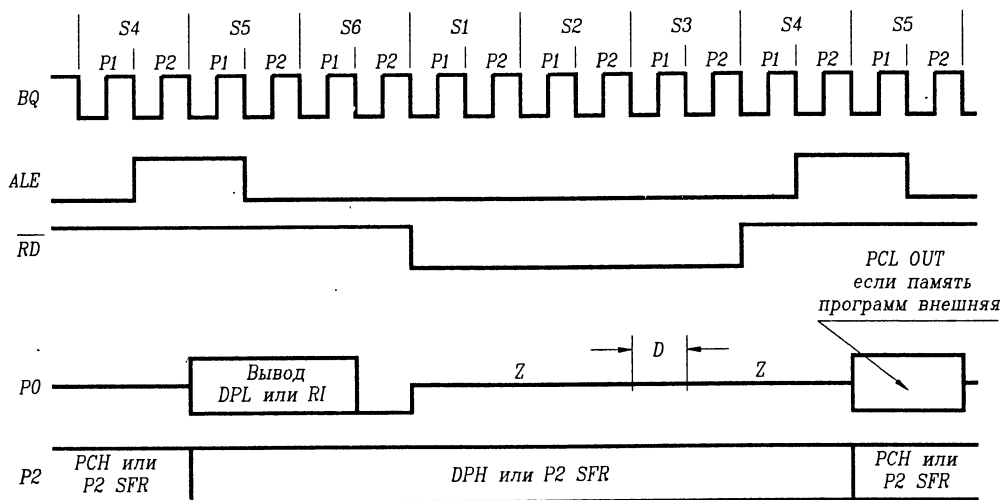


Рис. 2.11а. Страничная организация внешней памяти данных

На рис. 2.11а показана страничная организация внешней памяти данных. Приведенная схема позволяет работать с памятью данных емкостью 2 Кбайт, используя команды типа MOVX @Ri. Порт P0 при этом работает как мультиплексированная шина адрес/данные, а три линии порта P2 адресуют страницы внешнего ОЗУ. Остальные 5 линий порта P2 могут использоваться в качестве линий ввода/вывода.

Подробно организация памяти данных описана в разделе 2.3.5.

На рис. 2.11б и 2.11в соответственно приведены диаграммы циклов чтения и записи при работе ОМЭВМ с внешней памятью данных.



PCL OUT — выдача младшего байта счетчика команд PC

PCH — старший байт счетчика команд PC

DPL, DPH — соответственно младший и старший байты регистра указателя данных DPTR, который используется в качестве регистра косвенного адреса в командах MOVX A, @DPTR и MOVX @DPTR, A

P2 SFR — защелки порта P2

RI — регистры R0 и R1, которые используются в качестве регистров косвенного адреса в командах MOVX A, @RI и MOVX @RI, A

Z — высокоимпедансное состояние

D — период, в течение которого данные с P0 вводятся в ОМЭВМ

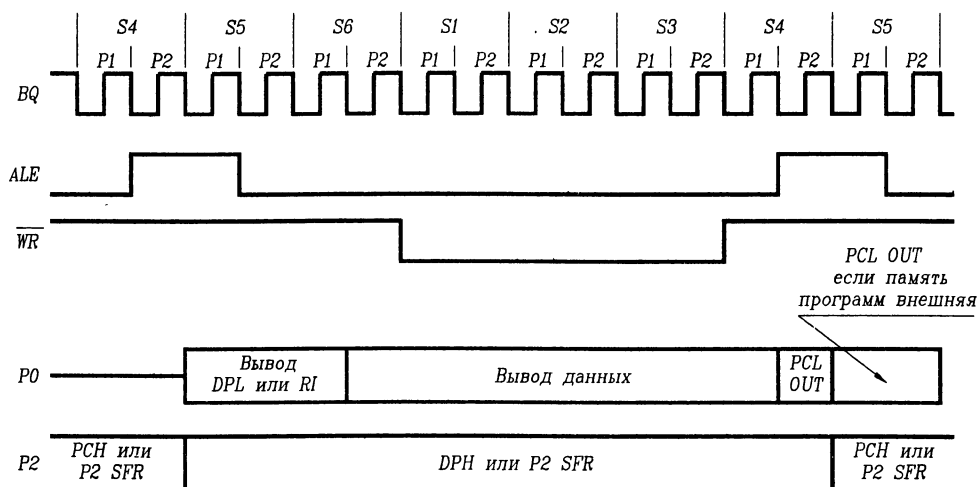
Рис. 2.11б. Цикл чтения из внешней памяти данных

2.2.8. Память программ

Память программ предназначена для хранения программ и имеет отдельное от памяти данных адресное пространство объемом до 64 Кбайт, причем, для микросхем КР1816ВЕ51, КМ1816ВЕ751 и для КР1830ВЕ51 часть памяти программ с адресами 0000H—0FFFH расположена на кристалле ОМЭВМ. Память программ, расположенная на кристалле, состоит из 12-разрядного дешифратора и ПЗУ емкостью 4К*8 бит для микросхем КР1816ВЕ51, КР1830ВЕ51 или ППЗУ с ультрафиолетовым стиранием емкостью 4К*8 бит для КМ1816ВЕ751. Запись программ в ПЗУ происходит во время изготовления кристаллов.

Если на вывод ОМЭВМ DEMA подано напряжение питания U_{CC} , то обращение к внешней памяти программ происходит автоматически при выработке счетчиком команд адреса, превышающего 0FFFH. Если адрес находится в пределах 0000H—0FFFH, обращение происходит к памяти программ, расположенной на кристалле (внутренней памяти программ).

Если на вывод ОМЭВМ DEMA подан "0", внутренняя память программ отключается и начиная с адреса 0000H все обращения выполняются к внешней памяти программ.



PCL OUT — выдача младшего байта счетчика команд PC
 PCH — старший байт счетчика команд PC
 DPL, DPH — соответственно младший и старший байты регистра указателя данных DPTR, который используется в качестве регистра косвенного адреса в командах MOVX A, @DPTR и MOVX @DPTR, A
 P2 SFR — защелки порта P2
 RI — регистры R0 и R1, которые используются в качестве регистров косвенного адреса в командах MOVX A, @RI и MOVX @RI, A

Рис. 2.11в. Цикл записи во внешнюю память данных

Если ОМЭВМ не имеет внутренней памяти программ, ее вывод ДЕМА должен быть подключен к шине 0 В.

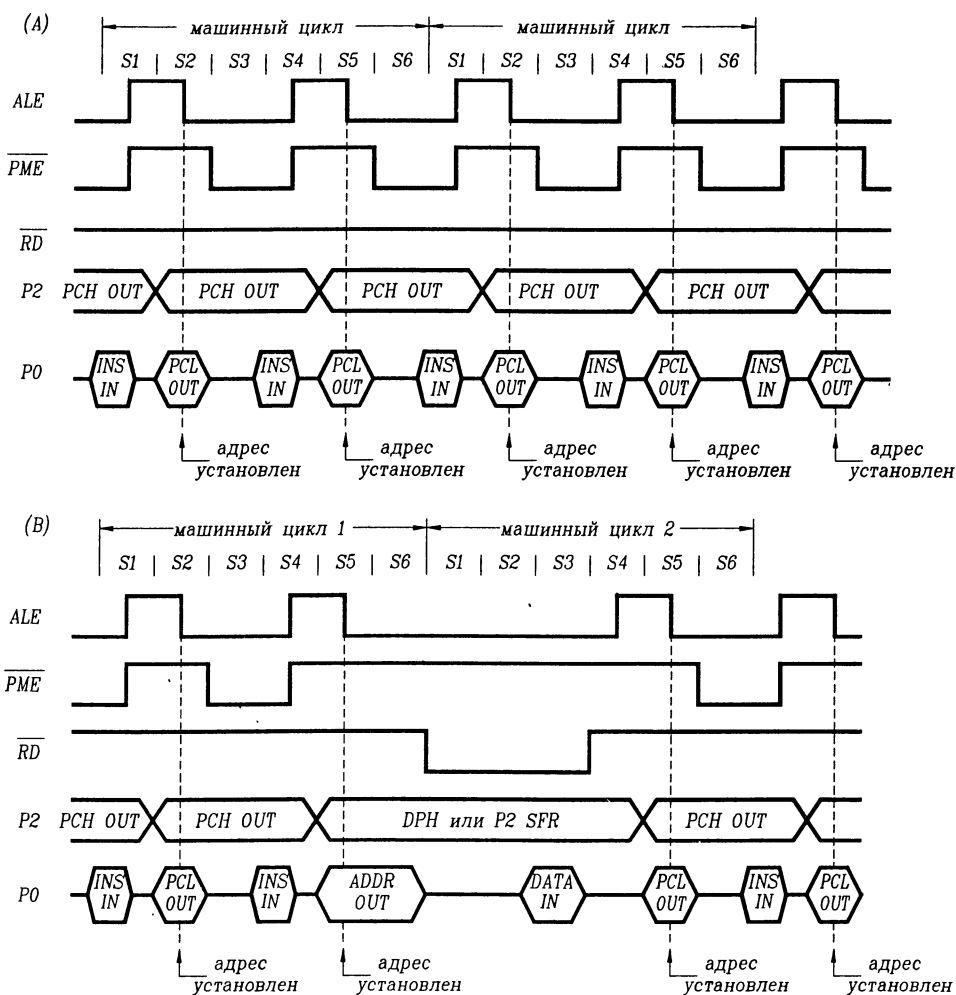
Чтение из внешней памяти программ строится сигналом ОМЭВМ \overline{PME} . При работе с внутренней памятью программ сигнал \overline{PME} не формируется. ОМЭВМ не имеют инструкций и аппаратных средств для программной записи в память программ.

При обращениях к внешней памяти программ всегда формируется 16-разрядный адрес, младший байт которого выдается через порт P0, а старший — через порт P2. При этом байт адреса, выдаваемый через порт P0, должен быть зафиксирован во внешнем регистре по спаду сигнала ALE, т. к. в дальнейшем линии порта P0 используются в качестве шины данных, по которой байт из внешней памяти программ вводится в ОМЭВМ.

На рис. 2.11г показана функциональная схема включения ОМЭВМ МК51 с внешним ППЗУ программ. Порт P0 работает как мультиплексированная шина адрес/данные: выдает младший байт счетчика команд, а затем переходит в высокоимпедансное состояние и ожидает прихода байта из ППЗУ программ. Когда младший байт адреса находится на выходах порта P0, сигнал \overline{ALE} защелкивает его в адресном регистре RG. Старший байт адреса находится на выходах порта P2 в течение всего времени обращения к ППЗУ. Сигнал \overline{PME} разрешает выборку байта из ППЗУ, после чего выбранный байт поступает на порт P0 МК51 и вводится в ОМЭВМ.

Подробно организация памяти программ описана в разделе 2.3.5.

На рис. 2.11д и 2.11е приведены диаграммы, показывающие формирование соответствующих сигналов при работе ОМЭВМ с внешней памятью программ. Как видно из диаграмм, при работе с внешней памятью программ сигнал \overline{PME} формируется дважды в каждом машинном цикле независимо от количества байт в команде. Если второй выбираемый байт в текущей команде не используется, он игнорируется ОМЭВМ. В дальнейшем при переходе к выполнению следующей команды этот байт будет введен вторично.



(A) — без выполнения команды MOVX

(В) — с выполнением команды MOVX

PCL OUT — выдача младшего байта счетчика команд PC

PCN OUT — выдача старшего байта счетчика команд PC

DPH — старший байт регистра указателя данных DPTR, который используется в качестве регистра косвенного адреса в командах MOVX A, @DPTR и MOVX @DPTR, A

P2 SFR — защелки порта P2

INS IN — ввод байта инструкции из памяти программ

ADDR OUT — выдача младшего байта адреса внешней памяти данных из регистров R0, R1 или регистра DPL

Задний фронт ALE стробирует адрес на порте P0. В этот момент адрес гарантированно установлен

Рис. 2.11е. Циклы работы с внешней памятью программ

2.3.1. Режимы работы и начальная установка

ОМЭВМ могут работать в следующих режимах:

- только с внешней памятью программ (все модификации ОМЭВМ);
- только с внутренней памятью программ (только КР1816ВЕ51, КМ1816ВЕ751 и КР1830ВЕ51);
- с внутренней и внешней памятью данных (все модификации ОМЭВМ);

— в режиме программирования внутренней памяти программ (только КМ1816ВЕ751);

— в режиме проверки внутренней памяти программ (только КР1816ВЕ51, КМ1816ВЕ751 и КР1830ВЕ51).

Режим работы устанавливается комбинацией входных и выходных сигналов.

Инициализация (сброс) микросхем осуществляется сигналом RST (активный высокий уровень напряжения) при условии подачи на микросхему внешнего сигнала синхронизации или при подключенном кварце. Вход RST является входом внутреннего триггера Шмитта.

Для того, чтобы сброс микросхемы гарантированно произошел, длительность сигнала высокого уровня на входе RST должна быть не менее двух машинных циклов ОМЭВМ (24 периода частоты синхронизации f_{BQ}). При поступлении внешнего сигнала сброса на вход RST ОМЭВМ формирует сигнал сброса.

Внешний сигнал сброса является асинхронным по отношению к внутренней синхронизации ОМЭВМ. Состояние вывода RST проверяется в фазе S5P2 каждого машинного цикла. После подачи сигнала высокого уровня на вход RST ОМЭВМ продолжает работу в течение времени от 19 до 31 периода частоты f_{BQ} (формируются ALE, P \overline{ME} и т. п.), после чего ALE и P \overline{ME} устанавливаются в "1" и находятся в этом состоянии все время, пока на входе RST присутствует активный сигнал сброса. После подачи на вход RST уровня "0" проходит от 1 до 2 машинных циклов до начала формирования сигналов ALE и P \overline{ME} .

При подаче сигнала сброса на вход RST внутренний алгоритм сброса ОМЭВМ производит следующие действия:

- устанавливает счетчик команд PC и все регистры специальных функций, кроме защелок портов P $\overline{0}$ —P3, указателя стека SP и регистра SBUF, в ноль;

- указатель стека принимает значение равное 07H;

- запрещает все источники прерываний, работу таймеров-счетчиков и последовательного порта;

- выбирает БАНК 0 ОЗУ, подготавливает порты P $\overline{0}$ —P3 для приема данных и определяет выводы ALE и P \overline{ME} как входы для внешней синхронизации;

- в регистрах специальных функций PCON, IP и IE резервные биты принимают случайные значения, а все остальные биты сбрасываются в ноль;

- в регистрах SBUF устанавливаются случайные значения.

- устанавливает фиксаторы-защелки портов P $\overline{0}$ —P3 в "1".

Обобщенные данные по состояниям регистров после сброса адреса указаны в таблице 2.16.

Сигнал сброса на входе RST не влияет на внутреннее ОЗУ данных. После включения питания содержимое ячеек внутреннего ОЗУ данных принимает случайные значения.

На рис. 2.12 показана схема подключения ОМЭВМ для реализации автоматического сброса по включению питания.

Для п-МОП ОМЭВМ автоматический сброс при включении питания U_{CC} может быть реализован подключением входа RST к U_{CC} через конденсатор емкостью 10 мкФ и к шине 0 В через резистор 8,2 К. Для КМОП ОМЭВМ этот резистор не требуется, однако его наличие не принесет вреда. КМОП ОМЭВМ содержат внутренний резистор, включенный между RST и выводом 0 В. Если использовать только внутренний резистор, емкость конденсатора может быть уменьшена до 1 мкФ.

Чтобы при включении питания сброс был гарантированно выполнен, вывод RST должен удерживаться в состоянии высокого уровня в течение времени, достаточного для запуска тактового генератора ОМЭВМ плюс еще минимум два машинных цикла. Время запуска тактового генератора ОМЭВМ зависит от его частоты работы и для 10 МГц кварцевого резонатора составляет в среднем 1 мс, а для 1 МГц кварцевого резонатора — 10 мс.

Представленная на рис. 2.12 цепь сброса при быстром уменьшении напряжения питания вызывает появление на входе RST отрицательного напряжения, которое не является опасным для микросхем вследствие наличия у ОМЭВМ внутренней схемы защиты.

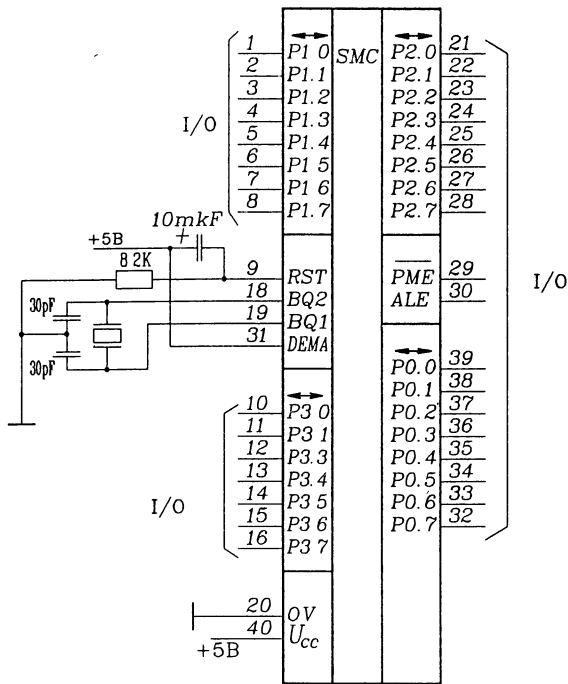


Рис. 2.12. Схема подключения ОМЭВМ для реализации автоматического сброса по включению питания

Таблица 2.16

Регистр	Информация
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0-P3	0FFH
IP	XXX00000B
IE	0XX00000B
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
SCON	00H
SBUF	Неопределенная
PCON	[0XXX0000B для серии 1830 0XXXXXXXXB для серии 1816.

Выводы портов находятся в случайном состоянии до момента запуска тактового генератора ОМЭВМ и только после этого внутренний сигнал сброса записывает "1" в фиксаторы-защелки портов, настраивая их на ввод.

Включение питания без обеспечения гарантированного сброса может привести к тому, что ОМЭВМ начнет выполнение программы с некоторого случайного адреса. Это объясняется тем, что счетчик команд РС не будет сброшен в $\emptyset\emptyset F$.

2.3.1.1. Работа с внешней памятью программ

При работе с внешней памятью программ выдача младших разрядов адреса ($A\emptyset...A7$) осуществляется через порт $P\emptyset$ ($P\emptyset.\emptyset...P\emptyset.7$). При этом адрес фиксируется по сигналу ALE , а команды принимаются по сигналу \overline{PME} . Старшие разряды адреса $A8...A15$ выдаются через $P2$ ($P2.\emptyset...P2.7$).

Временные диаграммы работы ОМЭВМ с внешней памятью программ представлены на рис. 2.13.

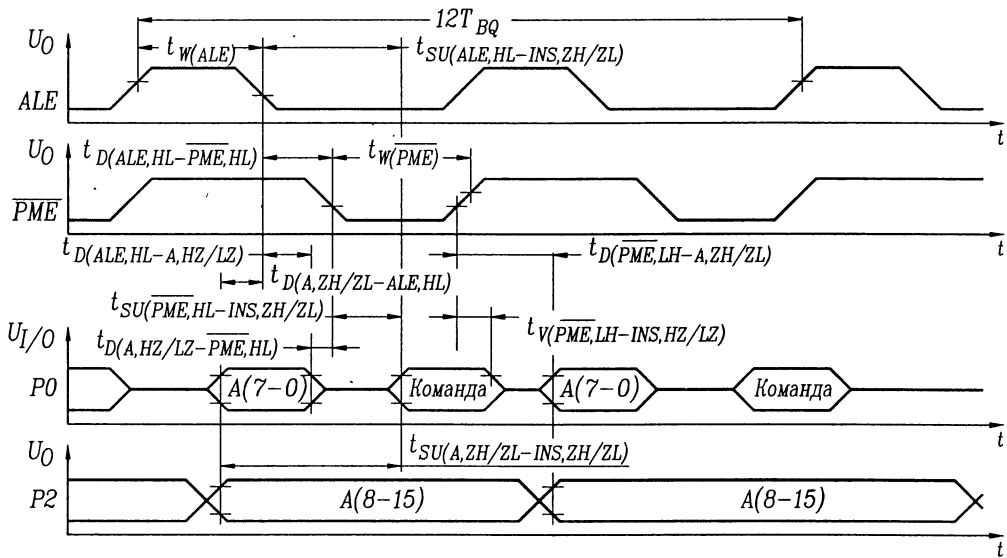


Рис. 2.13. Временные диаграммы работы микросхем с внешней памятью программ

2.3.1.2. Работа с внутренней памятью программ

Режим работы с внутренней памятью программ устанавливается заданием высокого уровня напряжения на выводе DEMA. Выполнение программы, хранящейся в памяти, начинается с команды, расположенной по адресу $\emptyset\emptyset H$, так как счетчик команд РС по сигналу "сброс" обнуляется.

В этом режиме порты $P\emptyset$ и $P2$ можно использовать как порты ввода-вывода, так как адрес/данные памяти программ передаются по внутренней магистрали ОМЭВМ.

Очевидно, что данный режим работы возможен только для ОМЭВМ, имеющих внутреннюю память программ: КМ1816ВЕ751, КР1816ВЕ51, КР1830ВЕ51.

2.3.1.3. Работа с памятью данных

При работе с внутренней памятью данных доступ к внутреннему ОЗУ (128 байт) осуществляется при помощи команд, имеющих операнды типа direct, Rn, @Ri (кроме MOVX).

При подключении внешнего ОЗУ емкостью до 256 байт обмен данными между ОЗУ и ОМЭВМ осуществляется через двунаправленный порт $P\emptyset$ с помощью команд MOVX @Ri, A и MOVX A, @Ri. Для работы с внешним ОЗУ объемом свыше 256 байт

(до 64 Кбайт) используются команды MOVX A, @DPTR и MOVX @DPTR, A . При этом выдача младших разрядов адреса ($A0 \dots A7$) и обмен данными осуществляются через порт $P0$ ($P0.0 \dots P0.7$). Старшие разряды адреса $A8 \dots A15$ выдаются через порт $P2$ ($P2.0 \dots P2.7$). При этом адрес $A7 \dots A0$ фиксируется по спаду сигнала ALE , а прием и выдача данных стробируются сигналами \overline{RD} и \overline{WR} .

Временные диаграммы работы ОМЭВМ с внешней памятью данных представлены на рис. 2.14 (при чтении данных) и 2.15 (при записи данных).

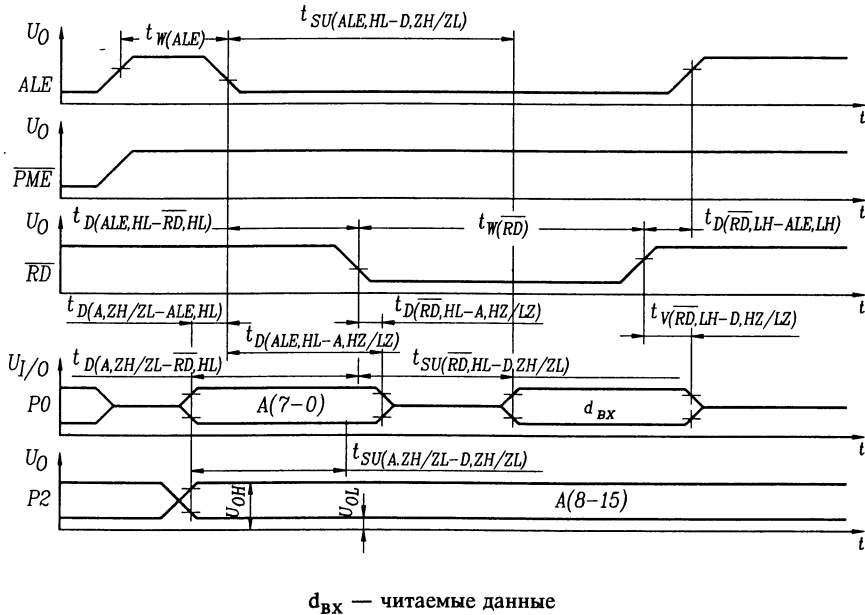


Рис. 2.14. Временные диаграммы работы микросхем при чтении данных из внешней памяти

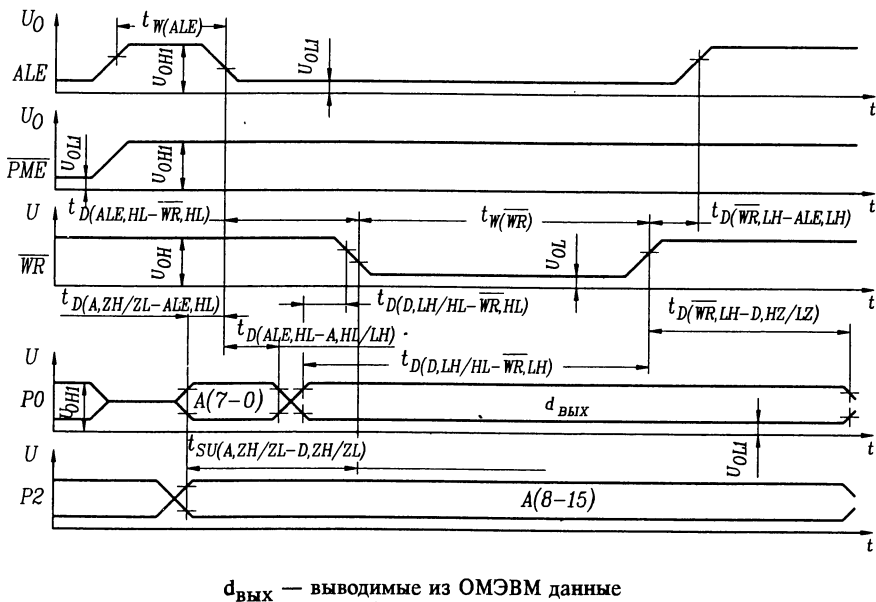


Рис. 2.15. Временные диаграммы работы микросхем при записи данных во внешнюю память

2.3.1.4. Программирование ОМЭВМ КМ1816ВЕ751

Программирование осуществляется на частоте задающего генератора 4—6 МГц.

Генератор необходим для работы внутренней шины ОМЭВМ, по которой происходят пересылки адреса и данных в соответствующие внутренние регистры. В табл. 2.16а приведены режимы работы ОМЭВМ при программировании внутреннего ППЗУ.

Временные диаграммы работы микросхемы при программировании и проверке внутреннего ППЗУ приведены на рис. 2.16а, а значения временных параметров — в табл. 2.25.

На рис. 2.16б, 2.16в и 2.16г показаны схемы подачи сигналов на выводы ОМЭВМ при ее работе в режимах, приведенных в табл. 2.16а. Значения уровней сигналов на рис. 2.16б, 2.16в и 2.16г приведены в табл. 2.24.

Таблица 2.16а. Режимы работы при программировании

Режим	RST	\overline{PME}	ALE	DEMA	P2.7	P2.6	P2.5	P2.4
Программирование	1	0	0*	U_{PR}	1	0	X	X
Проверка	1	0	1	1	0	0	X	X
Программирование бита защиты памяти	1	0	0*	U_{PR}	1	1	X	X

Примечания: "1" — уровень логической единицы на соответствующем выводе ОМЭВМ.

"0" — уровень логического нуля на соответствующем выводе ОМЭВМ.

"X" — произвольный логический уровень.

$U_{PR} = +21 \text{ В} \pm 0.5 \text{ В}$.

* — ALE подается импульсом низкого логического уровня длительностью $50 \text{ мс} \pm 5 \text{ мс}$.

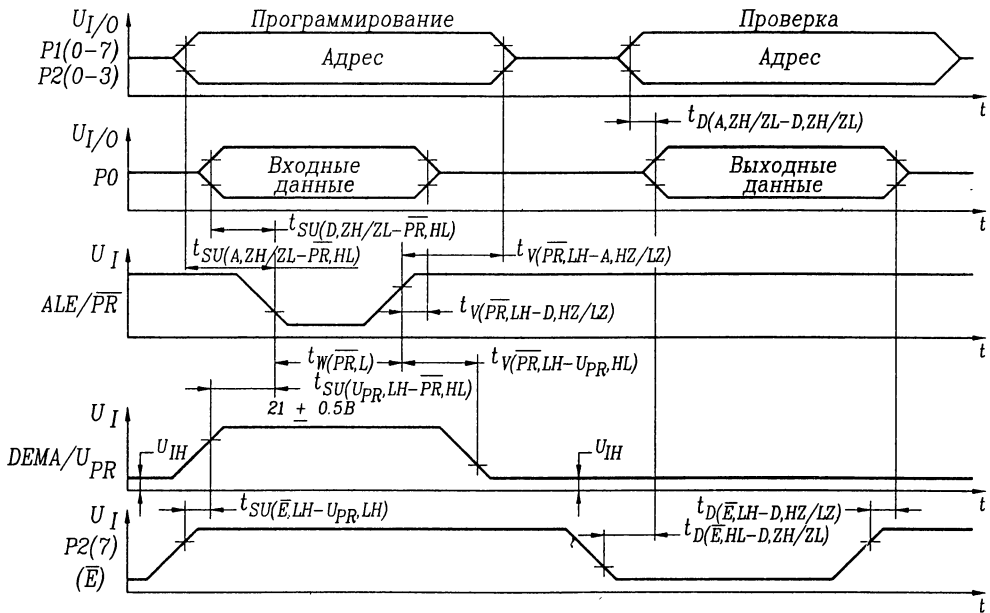


Рис. 2.16а. Временные диаграммы работы микросхем при записи и считывании внутренней программной памяти

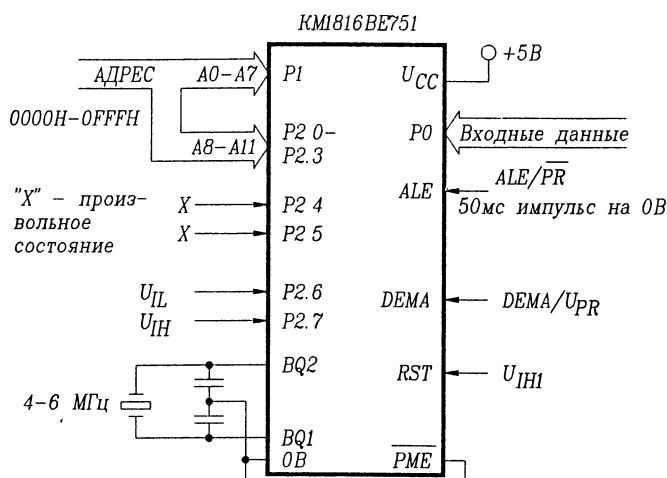


Рис. 2.166. Программирование KM1816BE751

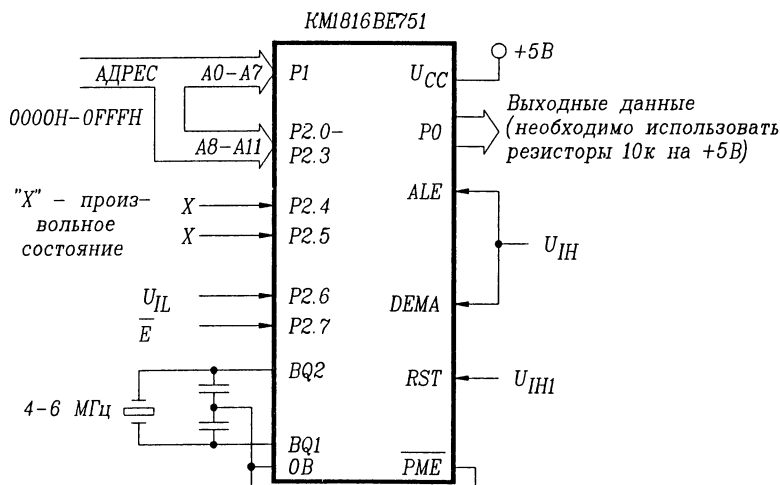


Рис. 2.16в. Проверка ППЗУ

В режиме программирования адрес ячейки ППЗУ, в которую необходимо записать информацию, подается на выводы портов P1 и P2: младшие разряды адреса A0...A7 подаются соответственно на выводы ОМЭВМ P1.0...P1.7, а старшие разряды адреса A8...A11 — соответственно на выводы P2.0...P2.3.

Байт данных, который необходимо записать в адресуемую ячейку ППЗУ, подается на выводы порта P0.

На остальные выводы порта P2, а также на выводы ОМЭВМ RST, $\overline{\text{PME}}$ и DEMA подаются уровни напряжения, приведенные для режима "Программирование" в табл. 2.16а. Программирование ячейки ППЗУ происходит при подаче импульса низкого уровня ALE длительностью $50 \text{ мс} \pm 5 \text{ мс}$ в соответствии с рис. 2.16а. DEMA удерживается в состоянии логической 1 до подачи импульса ALE. Затем напряжение на выводе DEMA повышается до $21 \text{ В} \pm 0,5 \text{ В}$, подается импульс ALE и DEMA вновь возвращается к уровню логической 1.

Необходимо особо подчеркнуть, что даже кратковременное импульсное превышение напряжения на выводе DEMA уровня $21,5 \text{ В}$ может вызвать необратимый отказ микросхемы. Поэтому источник программируемого напряжения U_{PR} должен быть хорошо отрегулирован.

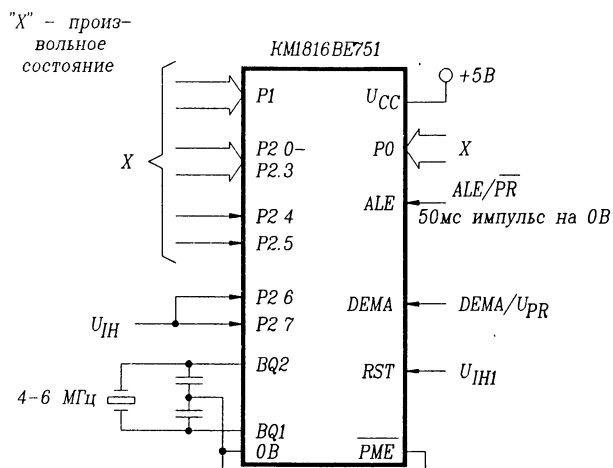


Рис. 2.16г. Программирование бита защиты памяти

Режим проверки содержимого внутренней памяти программ возможен только в том случае, если не запрограммирован бит защиты памяти. В данном режиме возможно чтение внешней по отношению к ОМЭВМ аппаратурой содержимого внутренней памяти программ ОМЭВМ. Чтение возможно либо во время, либо после программирования микросхемы.

Адрес ячейки памяти программ, содержимое которой необходимо прочитать, подается на выходы портов P1 и P2 аналогично режиму программирования. Другие выходы ОМЭВМ должны поддерживаться в состояниях, приведенных для режима "Проверка" в табл. 2.16а и на рис. 2.16в. Содержимое адресуемой ячейки памяти считывается с выходов порта P0 при подаче низкого логического уровня на вывод P2.7. При чтении вывод P2.7 может либо постоянно поддерживаться в состоянии "0", либо использоваться в качестве строб-сигнала чтения с активным низким уровнем.

Для работы в данном режиме необходимо использовать подтягивающие резисторы, включенные между выводами порта P0 и напряжением питания.

Свойство защиты памяти заключается в наличии в составе ППЗУ специального бита, который, будучи запрограммирован, запрещает доступ к внутренней памяти программ любыми внешними по отношению к ОМЭВМ средствами.

Включение микросхемы в режим программирования бита защиты памяти показано на рис. 2.16г. Процедура программирования в данном случае аналогична обычному программированию ППЗУ ОМЭВМ за исключением того, что вывод P2.6 удерживается в состоянии логической 1. Порты P0, P1 и выходы P2.0—P2.3 могут находиться в произвольном состоянии. Остальные выходы ОМЭВМ должны удерживаться в состояниях, приведенных в табл. 1.16а для режима "Программирование бита защиты памяти".

Если бит защиты памяти запрограммирован, его можно очистить только полным стиранием всего ППЗУ. При запрограммированном бите защиты памяти внутренняя память программ не может быть прочитана внешними по отношению к ОМЭВМ средствами, дальнейшее программирование ППЗУ становится невозможным и ОМЭВМ теряет возможность работы с внешней памятью программ. Стирание ППЗУ очищает бит защиты памяти и микросхема полностью восстанавливает свои функциональные возможности: может быть перепрограммирована и может работать с внешней памятью программ.

Стирание ППЗУ выполняется ультрафиолетовым излучением с длиной волны, меньшей 4000 ангстрем. Т. к. солнечный свет, а также лампы дневного света излучают волны в указанном диапазоне, они могут являться причиной порчи содержащейся в ППЗУ ОМЭВМ информации (критическая продолжительность

облучения — 1 неделя на солнце или 3 года в комнате с лампами дневного света). Поэтому рекомендуется закрывать кварцевое окошко микросхемы защитной наклейкой.

Рекомендуется стирание ППЗУ ультрафиолетовым излучением (длина волны 2537 ангстрем) с интегральной дозой как минимум 15 Вт-сек/см². При этом для стирания должно быть достаточным помещение микросхемы под ультрафиолетовую лампу с потоком 12000 мкВт/см² на время 20—30 минут и на расстояние около 2,5 см от лампы до кварцевого окошка микросхемы. После стирания все ячейки ППЗУ находятся в состоянии логической единицы.

2.3.1.5. Проверка внутренней памяти программ

В режиме проверки внутренней памяти контролируется правильность хранящейся в памяти программ информации, записанной в процессе производства микросхем КР1816ВЕ51 и КР1830ВЕ51 или в режиме программирования ИС КМ1816ВЕ751.

Выходы микросхемы выполняют следующие функции:

P2.7 — при подаче напряжения низкого уровня активизирует режим обращения к внутренней памяти для считывания;

P1.0...P1.7, P2.0...P2.3 — организуют подачу адреса A0—A11;

P0.0...P0.7 — организуют выдачу данных для контроля.

Задание режима проверки внутренней памяти программ и схема подачи сигналов на выходы ОМЭВМ в этом режиме полностью идентичны соответственно строке "Проверка" табл. 2.16а и рис. 2.16в.

Временная диаграмма при проверке внутренней памяти программ приведена на рис. 2.17а. Данная диаграмма полностью верна для микросхем КР1816ВЕ51 и КМ1816ВЕ751. Для КР1830ВЕ51 необходима привязка к частоте синхронизации ОМЭВМ: считываемая из внутренней памяти программ этих микросхем информация является истинной на выходах порта P0 в состояниях S5, S6, S1 и ложной в состояниях S2, S3, S4 каждого машинного цикла ОМЭВМ.

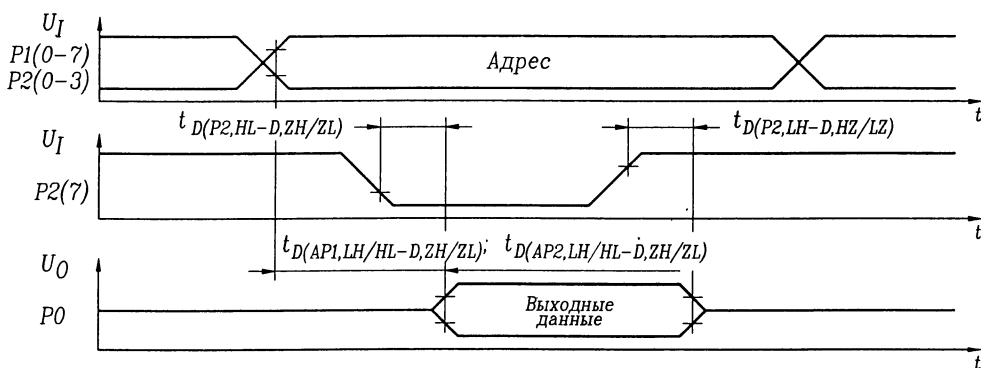


Рис. 2.17а. Временные диаграммы работы микросхем с внутренним ПЗУ

2.3.2. Работа с портами

Порты P1—P3 имеют идентичные характеристики. Данные, записанные в них, статически фиксируются и не изменяются до перезаписи.

В режиме работы с внешней памятью программ порт P2 служит для выдачи сигналов старших разрядов адреса и статически фиксирует их до момента изменения.

Обращения к внешней памяти программ всегда выполняются с использованием 16-разрядного адреса. Когда ОМЭВМ работает с внешней памятью программ, все 8 бит порта P2 задействованы на выдачу старшего байта адреса (старший байт счетчика команд PC) и не могут быть использованы в качестве линий ввода/вывода общего назначения.

Обращения к внешней памяти данных могут выполняться как с использованием 16-разрядного адреса (MOVX @DPTR), так и с использованием 8-разрядного адреса (MOVX @Ri). При использовании 16-разрядного адреса старший байт адреса выдается через порт P2, на линиях которого байт адреса удерживается в течение всего времени цикла записи или чтения.

Если через линию порта P2 выдается разряд адреса, содержащий "1", то напряжение логической "1" формируется мощным транзистором (транзисторы N1 и P1 на рис. 2.8—2.11), который при этом будет открыт в течение всего времени выдачи адреса.

Выдача адреса через порт P2 не влияет на содержимое защелок порта P2. Если порт P2 не задействован на выдачу адреса, то на его выводах выставляется содержимое защелок (регистр P2 в области SFR).

Для перевода любой линии портов P1—P3 на прием входной информации необходимо в соответствующий фиксатор-защелку порта записать "1" с помощью команды выдачи данных. Сигнал RST устанавливает все порты на прием входной информации.

Порт P0 — восьмиразрядный двунаправленный порт с тремя состояниями. Информация, выдаваемая портом P0 с помощью команд выдачи, сопровождается строб-импульсом \overline{WR} .

При записи информации в порт P0 с помощью команд приема вырабатывается строб-импульс \overline{RD} .

Кроме операций ввода-вывода информации предусмотрена возможность выполнения логических операций И, ИЛИ и исключающее ИЛИ непосредственно на фиксаторах портов P0—P3.

В режиме работы с внешней памятью программ порт P0 служит для выдачи младших разрядов адреса памяти программ и приема кодов команд. В режиме работы с внешним ОЗУ данных порт P0 служит для выдачи адреса внешнего ОЗУ данных и приема-выдачи данных при обмене с внешним ОЗУ.

Любую линию ввода-вывода можно проверить с помощью команд условного перехода.

Кроме того, выводы P3.2 и P3.3 можно использовать для внешнего аппаратного прерывания, выводы P3.4 и P3.5 — как входы счетчиков внешних событий для таймеров/счетчиков, а вывод P3.1 можно использовать как выход тактового сигнала в синхронном режиме работы последовательного интерфейса ОМЭВМ. Выводы P3.6 и P3.7 порта P3 служат для выдачи сигналов разрешения соответственно записи (\overline{WR}) и чтения (\overline{RD}) байта внешнего ЗУ данных через порт P0.

При работе с последовательным портом ОМЭВМ линии P3.0 и P3.1 используются, соответственно, как вход и выход последовательного канала.

На рис. 2.176 приведены временные диаграммы работы последовательного канала в режиме сдвигового регистра. В табл. 2.17а даны значения временных параметров для диаграммы на рис. 2.176.

Строго говоря, временные диаграммы на рис. 2.176 и значения в табл. 2.17а нормируются только для KP1830BE31/KP1830BE51. Однако, если судить по аналогам фирмы Intel, рис. 2.176 и табл. 2.17а абсолютно идентичны для всех микросхем серий 1816, 1830 семейства MK51.

Система команд ОМЭВМ позволяет считывать информацию с фиксатора-защелки порта или непосредственно с выхода, в зависимости от кода инструкции.

В инструкциях, в которых порт служит операндом-источником, информация считывается непосредственно с выводов порта, например ADD A, P1: содержимое аккумулятора складывается с информацией на выводах порта P1 и результат заносится в аккумулятор.

Во всех случаях, когда операндом и регистром назначения является порт или бит порта, команды считывают информацию с выходов фиксаторов-защелок, а не с внешних контактов выводов порта. Например, ORL P2, A.

На рис. 2.17в показана диаграмма работы с портами, из которой видно, в какие моменты времени информация вводится с выводов портов в ОМЭВМ и в какие

моменты происходит смена информации на портах, работающих на вывод, при выполнении команд MOV PORT, SRC (SRC — операнд-источник).

2.3.3. Работа с последовательным портом

Последовательный порт ОМЭВМ может использоваться в виде регистра сдвига для расширения ввода-вывода или в качестве универсального асинхронного приемопередатчика (УАПП) с фиксированной или переменной скоростью последовательного обмена и возможностью дуплексного включения (т.е. через последовательный порт можно принимать и передавать данные одновременно). Последовательный порт может принимать очередной байт даже если уже принятый до этого байт не был прочитан из регистра приемника. Однако, если до окончания приема находящийся в регистре приемника байт не будет прочитан, принятый байт теряется. Программный доступ к регистрам приемника и передатчика осуществляется обращением к регистру специальных функций SBUF. При записи в SBUF байт загружается в регистр передатчика, а при чтении SBUF байт читается из регистра приемника.

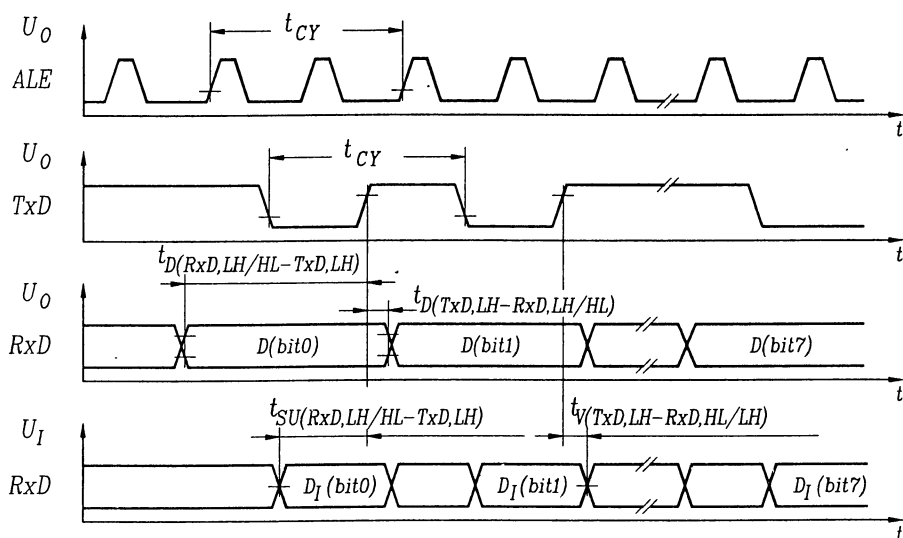
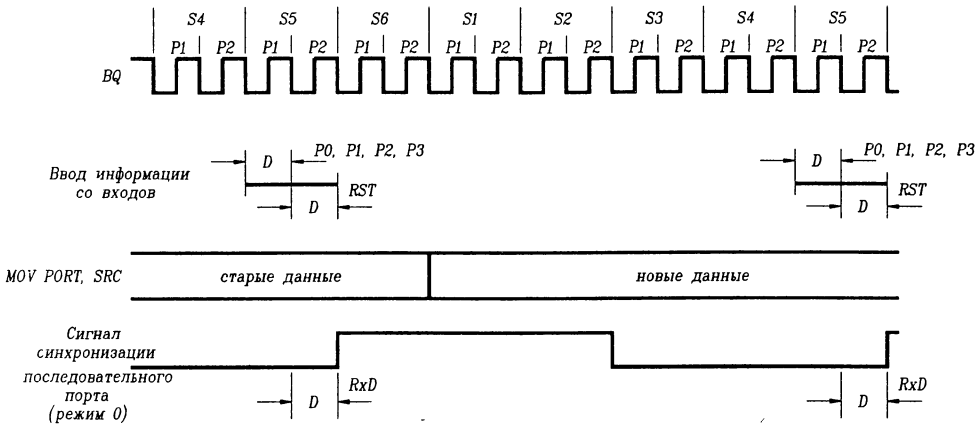


Рис. 2.176. Временные диаграммы работы последовательного канала в режиме сдвигового регистра

Таблица 2.17а. Временные параметры при работе последовательного канала в режиме сдвигового регистра

Ном.	Наименование параметра, единица измерения	Буквенное обозначение	Значение параметров	
			Не менее	Не более
1.	Период следования импульсов тактовых сигналов BQ, нс	$T_{BQ}=t$	83,3	286
2	Время цикла, нс	t_{CY}	12t	-
3	Время задержки сигнала TxD относительно сигнала RxD	$t_D(RxD-TxD)$	10t-133	-
4	Время задержки сигнала RxD относительно сигнала TxD	$t_D(TxD-RxD)$	2t-117	-
5	Время установления сигнала RxD относительно сигнала TxD	$t_{SU}(RxD-TxD)$	-	2t+133
6	Время сохранения сигнала RxD относительно сигнала TxD	$t_V(TxD-RxD)$	0	-



D — период, в течение которого информация вводится с соответствующего входа в ОМЭВМ

Рис. 2.17в. Диаграмма работы с портами

Прием и выдача байта данных начинается с младшего разряда и заканчивается старшим разрядом. Для разрешения приема необходимо установить 1 в разряде REN регистра управления SCON (регистр SCON описан в разделе 2.2.3).

Последовательный порт может быть запрограммирован на один из четырех режимов приема/передачи путем программирования разрядов SM0 и SM1 регистра SCON. Во всех четырех режимах передача инициируется любой командой, которая использует SBUF в качестве регистра назначения (выполняет операцию "Запись в SBUF"). Прием в режиме 0 инициируется одновременным выполнением условий REN=1 и RI=0 (REN и RI — разряды регистра управления SCON). В остальных режимах прием инициируется приходом стартового бита (нулевой уровень) при REN=1.

В режиме 0 последовательный порт работает как восьмиразрядный сдвиговый регистр. При этом 8 бит информации в последовательном коде принимаются и передаются через двунаправленный вывод RxD. На выводе TxD формируется сигнал синхронизации сдвигов.

Скорость (частота) приема/передачи в режиме 0 постоянна и составляет $f_{BQ}/12$, где f_{BQ} — частота синхронизации ОМЭВМ.

Временные диаграммы, иллюстрирующие работу последовательного порта в режиме 0, показаны на рис. 2.18. Все изображенные на рис. 2.18 сигналы за исключением RxD и TxD являются внутренними сигналами ОМЭВМ.

Передача начинается любой командой, которая использует SBUF в качестве регистра назначения (выполняет операцию "запись в SBUF").

При выполнении такой команды в фазе S6P2 вырабатывается внутренний импульс ЗАПИСЬ В SBUF, по которому предназначенный к передаче байт записывается в регистр сдвига передатчика и запускается блок управления передачей. Внутренняя система тактирования ОМЭВМ организована так, что между сигналом ЗАПИСЬ В SBUF и началом передачи проходит один полный машинный цикл, после чего вырабатывается внутренний сигнал ПОСЫЛКА, разрешающий выдачу содержимого регистра сдвига передатчика на выход RxD (вывод P3.0 ОМЭВМ) и импульсов синхронизации сдвига (СИНХР СДВИГ на рис. 2.18) на выход TxD (вывод P3.1 ОМЭВМ). Сигнал СИНХР СДВИГ имеет низкий уровень в состояниях S3, S4 и S5 каждого машинного цикла и высокий уровень в состояниях S6, S1 и S2. В фазе S6P2 каждого машинного цикла, в котором сигнал ПОСЫЛКА активен, формируется внутренний импульс СДВИГ, по которому содержимое регистра сдвига передатчика сдвигается на одну позицию и на выходе RxD выставляется очередной бит передаваемой посылки. Всего формируется восемь импульсов СДВИГ, после чего блок управления передачей снимает сигнал ПОСЫЛКА и устанавливает флаг

прерывания передатчика TI (разряд в регистре SCON). Оба эти действия выполняются в фазе S1P1 10-го машинного цикла после сигнала ЗАПИСЬ В SBUF.

Прием начинается при одновременном выполнении двух условий: REN=1 и RI=0. В фазе S6P2 следующего машинного цикла блок управления приемом вырабатывает внутренний сигнал ПРИЕМ, разрешающий выдачу импульсов СИНХР СДВИГ на выход ОМЭВМ TxD. Импульсы СИНХР СДВИГ меняют свое состояние в фазах S3P1 и S6P1. Биты принимаемой посылки через вход RxD поступают на регистр сдвига приемника. Состояние входа RxD опрашивается в фазе S5P2. В фазе S6P2 каждого машинного цикла, в котором сигнал ПРИЕМ активен, формируется внутренний импульс СДВИГ и содержимое регистра сдвига приемника сдвигается влево на одну позицию. Значение, которое при этом записывается в его крайний правый разряд, является значением сигнала на входе RxD, полученным в фазе S5P2 этого же машинного цикла. Всего формируется восемь импульсов СДВИГ, после чего блок управления приемом формирует сигнал загрузки содержимого регистра сдвига приемника в SBUF. В фазе S1P1 10-го машинного цикла после записи в SCON, сбросившей RI в 0, сигнал ПРИЕМ сбрасывается и устанавливается флаг прерывания приемника RI (бит в регистре SCON).

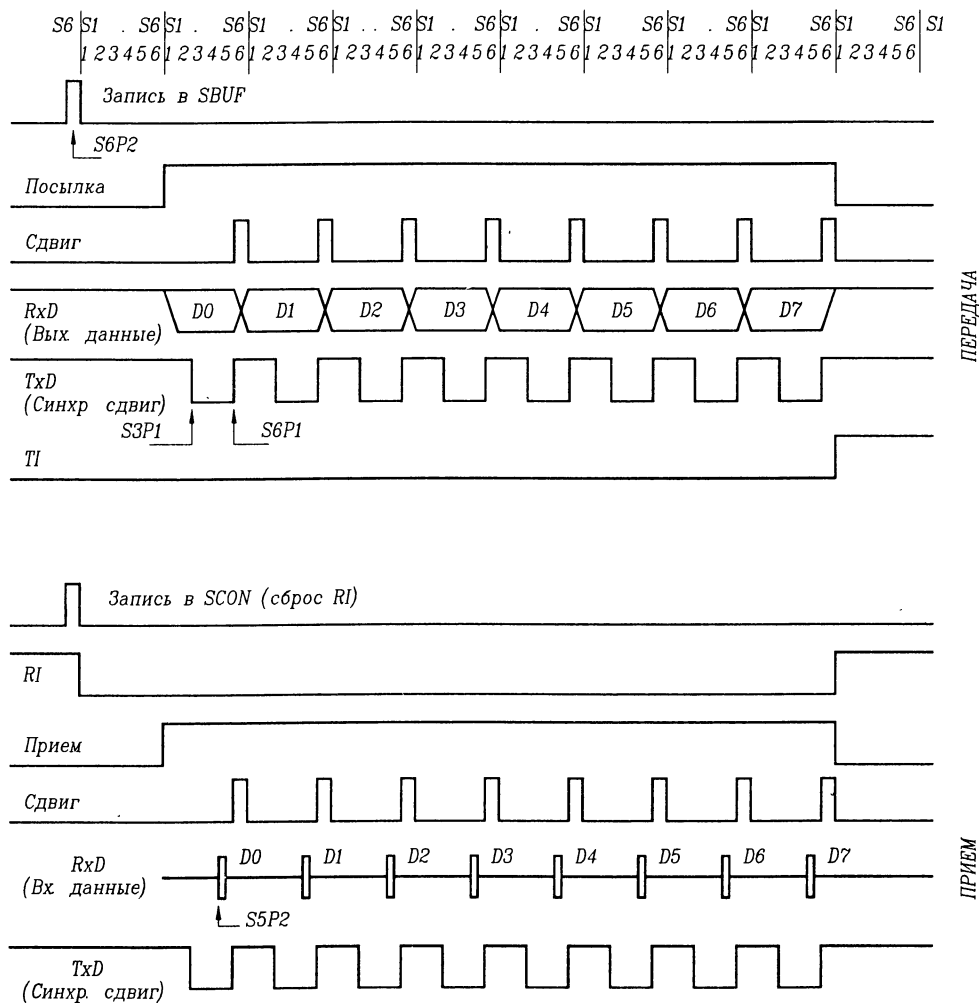


Рис. 2.18. Работа последовательного порта в режиме 0

В режиме 1 прием/передача данных осуществляется в формате восьмиразрядного УАПП. Через TxD передаются, а через RxD принимаются 10 бит: старт-бит (0), 8 бит данных и стоп-бит (1). При приеме стоп-бит заносится в бит RB8 регистра SCON. Скорость (частота) приема/передачи определяется частотой переполнений Таймера/Счетчика 1 f_{OV} .

На рис. 2.19 показаны схема синхронизации и временные диаграммы, иллюстрирующие работу последовательного порта в режиме 1.

В зависимости от состояния бита SMOD регистра PCON частота $f_1 = f_{OV}$ при SMOD=1 и $f_1 = f_{OV}/2$ при SMOD=0. Частота f_1 делится на 16 для получения сигналов синхронизации передачи СИНХР Tx и приема СИНХР Rx.

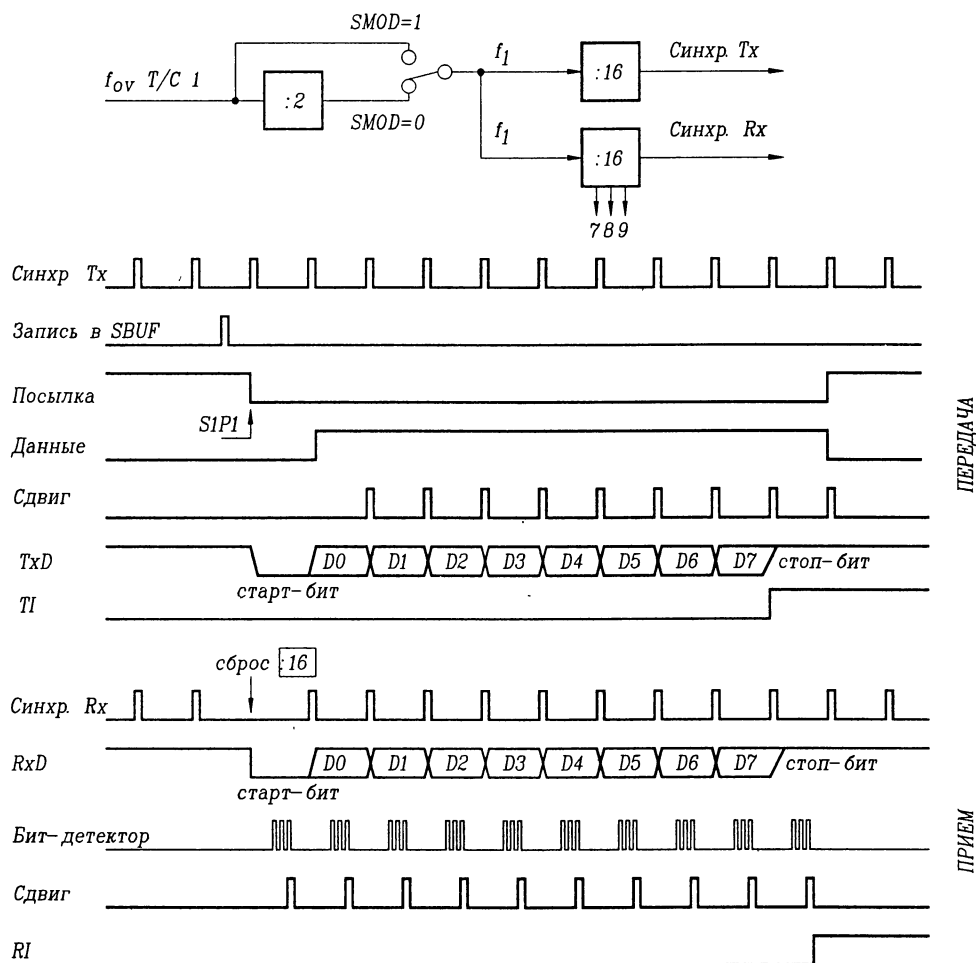


Рис. 2.19. Работа последовательного порта в режиме 1

Передача инициируется любой командой, использующей SBUF в качестве регистра назначения, в который производится запись. Выработываемый при этом внутренний импульс ОМЭВМ ЗАПИСЬ В SBUF загружает предназначенный к передаче байт в младшие 8 разрядов регистра сдвига передатчика и инициирует начало работы блока управления передачей. В режиме 1 регистр сдвига передатчика имеет 9 разрядов и в его 9-й разряд по импульсу ЗАПИСЬ В SBUF заносится "1" (стоп-бит).

Реально передача начинается в фазе S1P1 машинного цикла, следующего за ближайшим после ЗАПИСЬ В SBUF переполнением делителя на 16 в цепи сигнала СИНХР Тх (рис. 2.19). Таким образом, начало передачи синхронизовано делителем на 16, а не импульсом ЗАПИСЬ В SBUF. Период сигнала СИНХР Тх (синхронизация передачи) определяет время, в течение которого выдаваемый бит присутствует на выходе ТхD (время трансляции бита).

Передача начинается установкой активного (для режима 1 низкого) уровня внутреннего сигнала ОМЭВМ ПОСЫЛКА, появление которого вызывает выдачу на выход ТхD уровня старт-бита (ноль). После этого через время трансляции одного бита становится активным внутренний сигнал ОМЭВМ ДАННЫЕ, который разрешает выдачу содержимого регистра сдвига передатчика на выход ТхD (вывод P3.0 ОМЭВМ). При появлении активного сигнала ДАННЫЕ старт-бит на выходе ТхD сменяется битом D0 регистра сдвига передатчика. По окончании времени трансляции бита D0 формируется первый внутренний импульс ОМЭВМ СДВИГ, по которому содержимое регистра сдвига передатчика сдвигается на один разряд и бит D0 на выходе ТхD заменяется битом D1. Всего формируется 9 импульсов СДВИГ, в результате чего на выход ТхD выдаются 8 бит данных и стоп-бит. По окончании выдачи всех бит посылки блок управления передачей устанавливает, как показано на рис. 2.19, флаг прерывания передатчика TI и снимает сигналы ПОСЫЛКА и ДАННЫЕ.

Прием начинается при обнаружении перехода сигнала на входе RxD из "1" в "0". Для отслеживания такого перехода вход RxD аппаратно опрашивается с частотой f_1 (рис. 2.19). Когда переход сигнала на входе RxD из "1" в "0" обнаружен, немедленно сбрасывается счетчик-делитель на 16 в цепи сигнала СИНХР Rx (рис. 2.19), в результате чего происходит совмещение моментов переполнения этого счетчика-делителя (импульсы СИНХР Rx на рис. 2.19) с границами смены битов принимаемой посылки на входе RxD. Шестнадцать состояний счетчика-делителя делят время, в течение которого каждый бит принимаемой посылки присутствует на входе RxD, на 16 фаз, с 1-й по 16-ю для каждого бита. В фазах 7, 8 и 9 специальное устройство ОМЭВМ, бит-детектор, считывает со входа RxD 3 значения принимаемого бита, по мажоритарному принципу "2 из 3-х" выбирает из них одно и подает его на вход регистра сдвига приемника. Блок управления приемом при этом формирует внутренний импульс ОМЭВМ СДВИГ, в результате чего содержимое регистра сдвига приемника сдвигается на один разряд и принятый бит заносится в регистр сдвига приемника. Всего формируется 10 импульсов СДВИГ, а регистр сдвига приемника в режиме 1 является 9-разрядным. Поэтому после 10-го импульса СДВИГ в регистре сдвига приемника находятся биты данных D0—D7 и стоп-бит. После 10-го импульса СДВИГ блок управления приемом загружает данные из регистра сдвига приемника в SBUF, загружает стоп-бит из регистра сдвига приемника в разряд RB8 регистра SCON и устанавливает флаг прерывания приемника RI. Сигнал загрузки SBUF, RB8 и установки RI вырабатывается блоком управления приемом только в том случае, если в момент генерации последнего импульса СДВИГ выполняются следующие условия:

1. RI=0 и
2. Либо SM2=0, либо принятый стоп-бит равен "1".

Если хотя бы одно из этих условий не выполняется, принятая посылка безвозвратно теряется, а флаг RI не устанавливается. Если оба приведенных условия выполнены, стоп-бит поступает в RB8, восемь бит данных поступают в SBUF и устанавливается флаг RI. В это же время, независимо от выполнения приведенных выше условий, последовательный порт вновь начинает отслеживание перехода сигнала из "1" в "0" на входе RxD.

Если мажоритарный отбор при приеме первого бита посылки (старт-бит) показывает ненулевое значение бита, все устройства блока приема сбрасываются и начинается отслеживание следующего перехода сигнала из "1" в "0" на входе RxD. Таким образом обеспечивается защита от сбойных старт-битов.

Режимы 2 и 3 — это режимы 9-разрядного УАПП с постоянной (режим 2) и переменной (режим 3) скоростью обмена. В этих режимах 11 бит передаются/принимаются соответственно через выходы TxD/RxD в следующей последовательности: старт-бит, 9 бит данных, стоп-бит. 9-ый бит данных при передаче определяется содержимым разряда TB8 регистра SCON. При приеме 9-й бит данных заносится в бит RB8 регистра SCON.

Скорость (частота) приема/передачи в режиме 2 программно настраивается на одну из двух возможных величин: $f_{BQ}/32$ и $f_{BQ}/64$, где f_{BQ} — частота синхронизации ОМЭВМ. В режиме 3 скорость (частота) приема/передачи определяется частотой переполнений Таймера/Счетчика 1 f_{OV} .

Различие в скорости (частоте) приема/передачи является единственным отличием между режимом 2 и режимом 3. Во всем остальном эти два режима полностью идентичны.

На рис. 2.20 показаны схема синхронизации и временные диаграммы, иллюстрирующие работу последовательного порта в режимах 2 и 3.

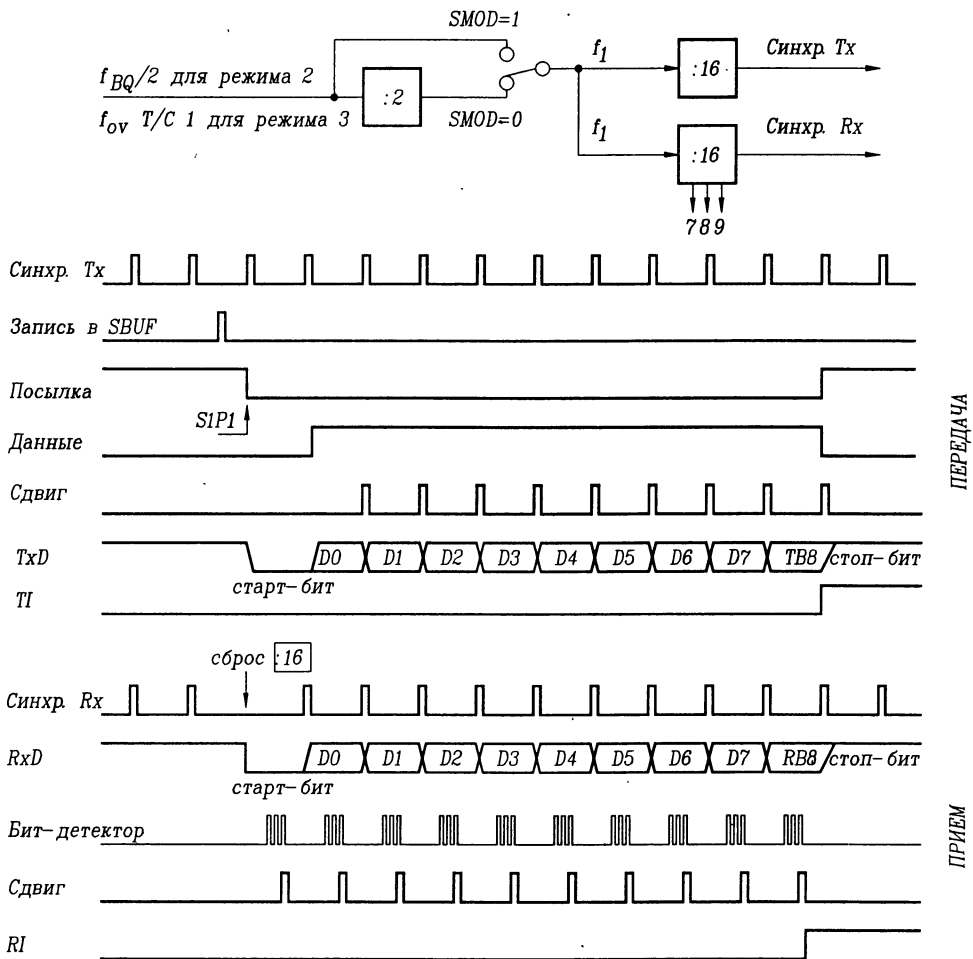


Рис. 2.20. Работа последовательного порта в режимах 2 и 3

Передача инициируется любой командой, использующей SBUF в качестве регистра назначения (выполняющей операцию "Запись в SBUF"). Выработываемый при этом внутренний импульс ОМЭВМ ЗАПИСЬ В SBUF загружает в младшие восемь разрядов регистра сдвига передатчика предназначенный к передаче байт и

инициирует начало работы блока управления передачей. Так же, как и в режиме 1, в режимах 2 и 3 регистр сдвига передатчика имеет 9 разрядов. По импульсу ЗАПИСЬ В SBUF в 9-й разряд регистра сдвига передатчика записывается значение бита TB8 регистра SCON. Передача начинается в фазе S1P1 машинного цикла, следующего за ближайшим после ЗАПИСЬ В SBUF переполнением делителя на 16 в цепи сигнала СИНХР Тх (рис. 2.20). Период сигнала СИНХР Тх (синхронизация передачи) определяет время, в течение которого выдаваемый бит присутствует на выходе ТxD (время трансляции бита). Внутренние сигналы ОМЭВМ ПОСЫЛКА, ДАННЫЕ и СДВИГ по функциональному назначению и формированию в режимах 2 и 3 идентичны этим же сигналам в режиме 1. В отличие от режима 1 в режимах 2 и 3 на выход ТxD выдается девять бит данных: D0—D7 и TB8. После первого импульса СДВИГ в освободившийся 9-й разряд регистра сдвига передатчика заносится "1" (стоп-бит). Всего формируется 9 импульсов СДВИГ, в результате чего все биты регистра сдвига передатчика последовательно выдаются на выход ТxD. По окончании выдачи всех бит посылки блок управления передачей устанавливает, как показано на рис. 2.20, флаг прерывания передатчика TI и снимает сигналы ПОСЫЛКА и ДАННЫЕ.

Прием начинается при обнаружении перехода сигнала на входе RxD из "1" в "0". Работа последовательного порта и блока управления приемом в режимах 2 и 3 полностью идентична режиму 1, включая мажоритарный отбор по принципу "2 из 3-х" значения каждого принимаемого бита с помощью бит-детектора. Регистр сдвига приемника в режимах 2 и 3 является 9-разрядным. Поэтому после 10-го импульса СДВИГ (рис. 2.20) в регистре сдвига приемника находятся 9 бит принятой посылки (обозначены на рис. 2.20 D0—D7 и RB8). После 10-го импульса СДВИГ блок управления приемом загружает биты D0—D7 из регистра сдвига приемника в SBUF, переписывает 9-й разряд регистра сдвига приемника в бит RB8 регистра SCON и устанавливает флаг прерывания приемника RI в регистре SCON. Сигнал загрузки SBUF, RB8 и установки RI вырабатывается блоком управления приемом тогда и только тогда, когда в момент генерации последнего импульса СДВИГ выполняются следующие условия:

1. RI=0 и
2. Либо SM2=0, либо принятый 9-й бит данных равен 0.

Если хотя бы одно из этих условий не выполняется, принятая посылка безвозвратно теряется, а флаг RI не устанавливается. Если оба приведенных условия выполнены, принятый 9-й бит данных поступает в RB8, биты D0—D7 записываются в SBUF и устанавливается флаг RI. Независимо от выполнения приведенных выше условий последовательный порт вновь начинает отслеживание перехода сигнала из "1" в "0" на входе RxD. Особо необходимо отметить, что значение принятого стоп-бита в режимах 2 и 3 не влияет на SBUF, RB8 или RI.

Скорость (частота пересылки битов) последовательного обмена F_n в зависимости от режима работы последовательного порта определяется либо частотой синхронизации ОМЭВМ f_{BQ} (режимы 0 и 2), либо частотой переполнения Таймера/Счетчика 1 f_{QV} (режимы 1 и 3).

В режиме 0 частота пересылки одного бита (скорость последовательного обмена) максимальна. Она постоянна и составляет:

$$F_{n0} = f_{BQ} / 12.$$

При необходимости работать с переменной скоростью используется режим 2 последовательного порта. В этом режиме скорость последовательной передачи зависит от состояния бита SMOD регистра SCON:

$$F_{n2} = (2^{SMOD} / 64) \cdot f_{BQ}.$$

Т.е. при SMOD=0 $F_{n2} = f_{BQ} / 64$, а при SMOD=1 $F_{n2} = f_{BQ} / 32$. По сигналу сброс бита SMOD устанавливается в ноль. Для задания бита SMOD используются команды с байтовой адресацией, например, команда MOV 87H, #80H.

В режимах 1, 3 также имеется возможность изменить скорость последовательной передачи:

$$F_{n1}=F_{n3}=(2^{SMOD}/32) \cdot f_{OV},$$

где f_{OV} — частота переполнений Т/С 1.

Для использования Т/С 1 в качестве источника для задания частот F_{n1} и F_{n3} необходимо:

- 1) запретить прерывания от Т/С 1;
- 2) запрограммировать работу Т/С 1 в качестве таймера или в качестве счетчика, установив при этом для него один из режимов 0, 1 или 2;
- 3) запустить Т/С 1 на счет.

Обычно для синхронизации последовательного порта таймер Т/С 1 включается в режим автозагрузки (режим 2).

В этом случае скорость последовательного обмена определяется по формуле:

$$F_{n1}=F_{n3}=(2^{SMOD} \cdot f_{BQ}) / (32 \cdot 12 \cdot [256 - (TH)]),$$

где (TH) — десятичный код содержимого TH1. Если необходим последовательный обмен с очень низкой скоростью, то можно использовать Т/С 1 в режиме 16-разрядного таймера (режим 1), разрешив при этом прерывание от Т/С 1 с целью перезагрузки TL1/TH1 в подпрограмме обслуживания прерывания.

В табл. 2.17 приведен ряд стандартных скоростей последовательного обмена и то, как они могут быть реализованы с помощью Т/С 1 в режимах 1, 3.

В табл. 2.18 приведена сводная информация по всем четырем режимам работы последовательного порта ОМЭВМ семейства МК51.

Ниже приведен пример программы инициализации последовательного порта для работы на частоте тактового сигнала $f_{BQ}=6$ МГц:

	; инициализация последовательного порта
	; для работы со скоростью 110 бод на
	; частоте тактового сигнала 6 МГц;
INT1: CLR TR1	; останов таймера;
MOV TH1, #72H	; автозагружаемое значение для получения
	; скорости 110 бод;
MOV SCON, #11011100B	; установка режима 9-разрядного УАПП;
MOV SCON, #00100000B	; установка режима автозагрузки
	; таймера 1;
SET TR1	; запуск таймера 1;
	; прием символа от внешнего устройства
	; ввода;
CIN: INB BI, CIN	; ожидание завершения приема;
MOV A, SBUF	; чтение полученного символа;
CLR RI	; очистка флага приема;
	; выдача принятого символа на внешнее
	; устройство ввода;
COUT: INB TI, COUT	; ожидание окончания передачи;
CLR TI	; очистка флага передачи;
MOV SBUF, A	; выдача символа.

Режим 2 и режим 3 последовательного порта позволяют организовать работу ОМЭВМ в многопроцессорных системах, использующих для обмена информацией между ОМЭВМ разделяемый моноканал (коаксиальный кабель, витая пара, оптоволокно). В этих режимах принимается 9 бит данных и 9-й принятый бит записывается в бит RB8 регистра SCON. При этом, если бит SM2 регистра SCON установлен в "1", то после приема посылки флаг прерывания приемника RI будет

установлен только в том случае, если RB8=1. Эту особенность работы последовательного порта в режимах 2 и 3 можно использовать для организации межконтроллерного обмена следующим образом.

Таблица 2.17

Режимы работы последовательного порта	Скорость приема/передачи Кбод	f _{BQ} , МГц	SMOD	Разряды TMOD			TH1	Примечание
				C/T	M1	M0		
Режим 0	Макс: 1000	12	X	X	X	X	X	
Режим 2	Макс: 375	12	1	X	X	X	X	
Режимы 1, 3	62,5	12	1	0	1	0	FFH	
	19,2	11,059	1	0	1	0	FDH	
	9,6	11,059	0	0	1	0	FDH	
	4,8	11,059	0	0	1	0	FAH	
	2,4	11,059	0	0	1	0	E4H	
	1,2	11,059	0	0	1	0	E8H	
	0,1375	11,986	0	0	1	0	18H	
	0,110	6	0	0	1	0	72H	
	0,110	12	0	0	0	1	FEH	
								TL1=EBH

Таблица 2.18

Режим обмена	Вид обмена	Разряды регистра SCON							Скорость передачи	Примечание
		SM0	SM1	SM2	REN	TB8	RB8	ФЛАГ		
0	ПРД	0	0	0	-	-	-	TI	f _{BQ} /12	Для инициализации приема установить: RI=0
	ПРМ			1	RI					
1	ПРД	0	1	-	-	-	-	TI	$\frac{2^{SMOD} \cdot f_{0V}}{32}$	
	ПРМ			0	1		стоп-бит	RI		
				1			1			
							0	-		
2	ПРД	1	0	-	-	9-й бит данных	-	TI	$\frac{2^{SMOD} \cdot f_{BQ}}{64}$	
	ПРМ			0	1	-	9-й бит данных	RI		
				1			1			
							0	-		
3	ПРД	1	1	-	-	9-й бит данных	-	TI	$\frac{2^{SMOD} \cdot f_{0V}}{32}$	
	ПРМ			0	1	-	9-й бит данных	RI		
				1			1			
							0	-		

Когда ведущая ОМЭВМ хочет передать блок данных одной из ведомых ОМЭВМ, она выдает в моноканал посылку с адресом ведомой, которой будет передан блок данных. Адресная посылка отличается от посылки с данными тем, что в адресной посылке 9-й бит данных равен "1", а в посылке с данными — "0". Таким образом, при $SM2=1$ ни одна ведомая ОМЭВМ не будет реагировать на посылку с данными, но все ведомые среагируют на адресную посылку. Проанализировав полученный адрес, адресуемая ОМЭВМ сбрасывает свой бит $SM2$, а остальные оставляют его без изменения и вновь переходят к выполнению прерванной программы. После этого ведущая ОМЭВМ может начинать выдачу в моноканал блока данных, на посылки которого будет реагировать только ОМЭВМ, у которой $SM2=0$.

Бит $SM2$ никак не участвует в работе последовательного порта в режиме 0. В режиме 1 бит $SM2$ может использоваться для контроля правильности принятого стоп-бита: в режиме 1, если $SM2=1$, флаг прерывания приемника RI не будет установлен, если принятый стоп-бит не равен "1".

2.3.4. Структура прерываний

Механизм прерываний в ОМЭВМ позволяет автоматически реагировать на внешние и на внутренние события (переполнение таймеров/счетчиков; завершение последовательного обмена). Алгоритм обработки прерывания при обнаружении запроса прерывания представлен на рис. 2.21. На рис. 2.22 изображены все возможные источники прерывания.

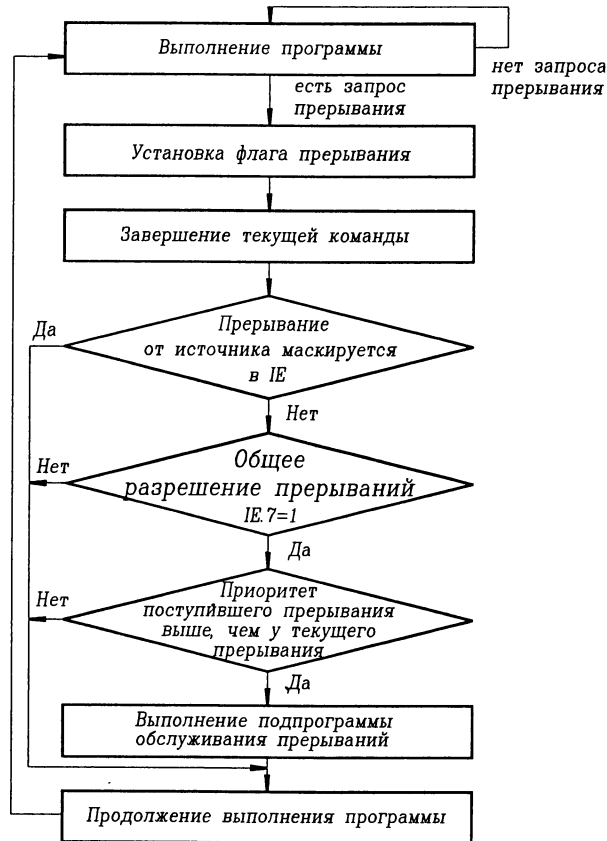
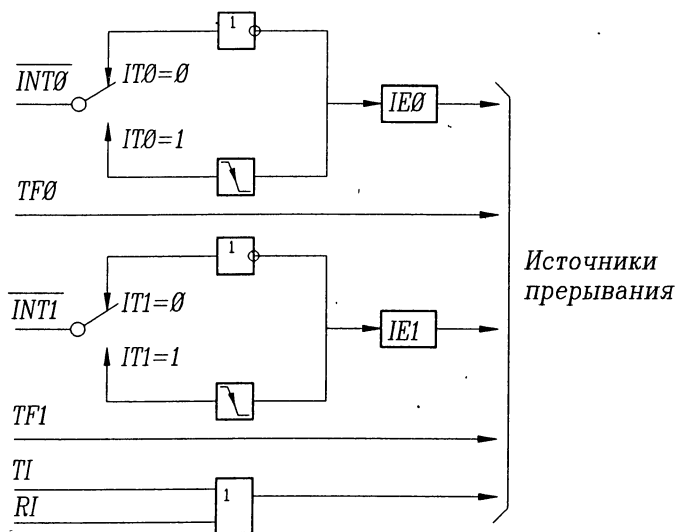


Рис. 2.21. Алгоритм обработки прерывания

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

Чтобы внешнее прерывание по уровню было распознано, необходимо, чтобы низкий уровень на выводе \overline{INTx} удерживался в течение не менее 12 периодов сигнала тактовой частоты ОМЭВМ. Это объясняется тем, что проверка выводов ОМЭВМ $\overline{INT0}$, $\overline{INT1}$ выполняется внутренними аппаратными средствами ОМЭВМ один раз в каждом машинном цикле. В случае внешнего прерывания по фронту флаг IEx будет установлен, если две последовательные проверки входа \overline{INTx} покажут в одном машинном цикле "1", а в следующем "0". Поэтому, если внешнее прерывание активизируется по переходу из состояния высокого уровня в состояние низкого уровня, то минимум одному машинному циклу низкого уровня должен предшествовать минимум один машинный цикл высокого уровня на выводе \overline{INTx} . Если внешнее прерывание активизируется по уровню, запрос должен удерживаться до начала обслуживающей подпрограммы и сниматься до завершения этой подпрограммы для предотвращения повторного обслуживания.



Примечания. $IT0$, $IT1$ — биты управления регистра TCON
 $TF0$, $TF1$, $IE0$, $IE1$ — флаги-признаки в регистре TCON
 TI , RI — флаги-признаки в регистре SCON

Рис. 2.22. Возможные источники прерывания

Прерывания от таймеров/счетчиков вызываются установкой флагов $TF0$ и $TF1$ регистра TCON, которые устанавливаются при переполнении соответствующих регистров таймеров/счетчиков (за исключением режима 3, см. раздел 2.3.3). Очистка флагов $TF0$ и $TF1$ производится внутренней аппаратурой ОМЭВМ при переходе к подпрограмме обслуживания прерывания.

Прерывание от последовательного порта вызывается установкой флага прерывания приемника RI или флага прерывания передатчика TI в регистре SCON. В отличие от всех остальных флагов, RI и TI сбрасываются только программным

путем обычно в пределах подпрограммы обработки прерывания, где определяется, какому из флагов RI или TI соответствует прерывание.

Каждый из перечисленных источников прерываний может быть индивидуально разрешен или запрещен установкой или сбросом соответствующего бита в регистре разрешения прерываний IE. Регистр IE содержит также бит EA, сброс которого в "0" запрещает сразу все прерывания. Необходимым условием прерывания является его разрешение в регистре IE. Формат и описание регистра разрешения прерываний приведены в разделе 2.2.4.

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

```

MAIN:  MOV IE, #00000101B ; разрешение прерывания от INT0, INT1.
        MOV IP, #04H      ; присвоение INT1 старшего приоритета.
        SET EA            ; общее разрешение прерывания.
        MOV P3; #11110011B ; имитация внешних прерываний.
SUBR:  ORG013H           ; переход к подпрограмме обслуживания INT1.

```

В предложенном примере запросы прерывания INT0 и INT1, имеющие различный приоритет, поступают одновременно. При этом обслуживается прерывание с высшим приоритетом.

В случае, когда прерывание по INTx (x=0,1) вызывается уровнем сигнала на соответствующем входе ОМЭВМ, флаг IEx (x=0,1) при переходе к подпрограмме обработки прерывания автоматически сбрасывается, а затем, если соответствующий вывод ОМЭВМ P3.2 или P3.3 все еще находится в состоянии логического "0", вновь устанавливается. Поэтому, в случае, когда прерывание по входам INT0, INT1 вызывается уровнем, программная установка в "1" флагов IE0, IE1 вызовет прерывание, после чего соответствующий флаг IEx (x=0,1) будет автоматически сброшен при переходе к подпрограмме обработки прерывания.

Флаги IE0, IE1, TF0, TF1, RI, TI устанавливаются независимо от того разрешено или нет соответствующее прерывание в регистре IE.

Структура приоритетов прерываний является двухступенчатой. Каждому источнику прерывания может быть индивидуально присвоен один из двух уровней приоритета: высокий или низкий. Выполняется это установкой (высокий уровень приоритета) или сбросом (низкий уровень приоритета) соответствующего бита в регистре приоритетов прерываний IP (описан в разделе 2.2.4). Программа обработки прерывания с низким уровнем приоритета может быть прервана запросом прерывания с высоким уровнем приоритета, но не может быть прервана другим запросом прерывания с низким уровнем приоритета. Программа обработки прерывания с высоким уровнем приоритета не может быть прервана никаким другим запросом прерывания ни от одного из источников. Если два запроса с разными уровнями приоритета приняты одновременно, сначала будет обслужен запрос с высоким уровнем приоритета. Если одновременно приняты запросы с одинаковым уровнем приоритета, обработка их будет производиться в порядке, задаваемом последовательностью внутреннего опроса флагов прерываний. Таким образом, в пределах одного приоритетного уровня существует еще одна структура приоритетов:

<u>Источник</u>	<u>Приоритет внутри уровня</u>
1. IE0	(высший)
2. TF0	
3. IE1	
4. TF1	
5. RI+TI	(низший)

Необходимо особо подчеркнуть, что структура "Приоритет внутри уровня" работает только в тех случаях, когда определяется последовательность обслуживания запросов на прерывания, которые приняты одновременно и при этом имеют одинаковый уровень приоритета.

Общая схема системы прерываний ОМЭВМ приведена на рис. 2.22а.

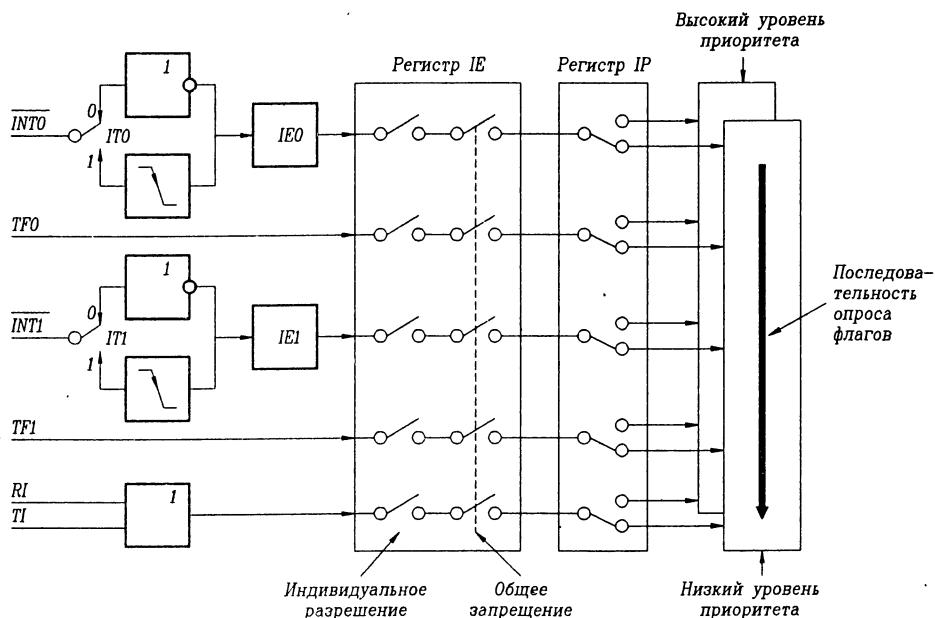


Рис. 2.22а. Система прерываний ОМЭВМ

Обработка прерываний и время отклика. Уровни на выводах $\overline{INT0}$ и $\overline{INT1}$ инвертируются и защелкиваются в флаги прерывания IE0 и IE1 в фазе S5P2 каждого машинного цикла. В фазе S5P2 устанавливаются флаги прерываний последовательного порта RI и TI. Флаги TF0 и TF1 таймеров/счетчиков устанавливаются в фазе S5P2 машинного цикла, в котором происходит переполнение Т/С. Анализ (опрос) флагов выполняется внутренними средствами ОМЭВМ в следующем после установки (защелкивания) флагов машинном цикле (цикл опроса флага). И только после выполнения последнего цикла текущей команды производится аппаратный вызов соответствующей подпрограммы обслуживания, эквивалентный команде LCALL.

Обращение к подпрограмме обслуживания задерживается (блокируется аппаратный LCALL) при выполнении хотя бы одного из следующих условий:

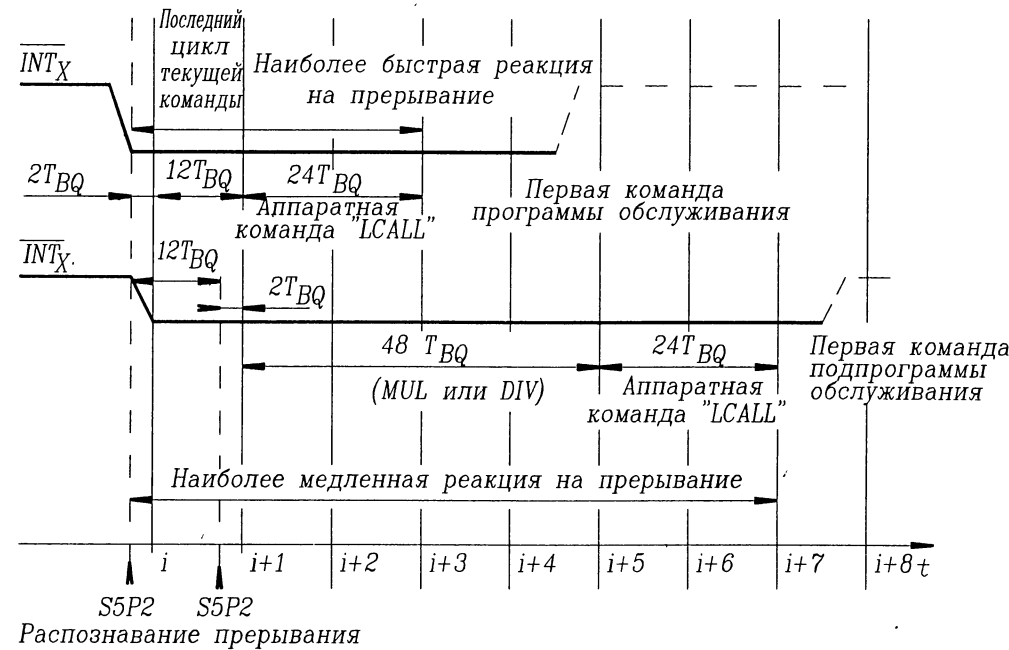
- уже производится обработка прерывания с таким же или высшим приоритетом;
- текущий машинный цикл (цикл опроса флага) не является последним циклом выполняемой команды;
- выполняемая команда текущей программы является командой RETI или любой командой обращения к регистрам IE, IP.

В последнем условии после окончания одной из вышеуказанных команд обязательно выполнится еще минимум одна команда текущей программы перед вызовом подпрограммы обслуживания прерывания.

Флаг прерывания, установленный во время действия блокировки прерывания по одному из трех указанных выше условий и сброшенный до их снятия, не вызовет обслуживания соответствующего запроса прерывания.

На рис. 2.23 изображены случаи наиболее быстрой и медленной реакции на прерывание (на примере внешнего прерывания). От момента фиксации запроса внешнего прерывания до начала выполнения первой команды подпрограммы

обслуживания требуется от 38 до 86 периодов сигнала тактовой частоты ОМЭВМ T_{BQ} (табл. 2.19).



Примечания. 1. i — машинный цикл, равный $12T_{BQ}$
2. $INTx$ — условное обозначение $INT0$ или $INT1$.

Рис. 2.23. Временная диаграмма задержки начала выполнения подпрограммы обслуживания внешнего прерывания

Таблица 2.19

Выполняемые операции	Время отклика на прерывание	
	Наиболее быстрая реакция	Наиболее медленная реакция
1. Завершение цикла, в котором обнаружен запрос прерывания	$2T_{BQ}$	$2T_{BQ}$
2. Окончание текущей или следующей команды	$12T_{BQ}$	$12T_{BQ}$
3. Выполнение команды MUL или DIV	Команды MUL, DIV не используются	$48T_{BQ}$
4. Выполнение аппаратно-реализуемой команды LCALL вызова подпрограммы обслуживания прерывания	$24T_{BQ}$	$24T_{BQ}$
Всего	$38T_{BQ}$	$86T_{BQ}$

Если запрос прерывания с более высоким уровнем приоритета зафиксирован во время аппаратного вызова подпрограммы обслуживания, а именно в фазе S5P2 1-го цикла аппаратной команды LCALL, то по окончании процедуры текущего вызова сразу же начнет выполняться процедура аппаратного вызова по поступившему запросу.

Аппаратно-реализуемая команда LCALL загружает содержимое счетчика команд PC в стек (при этом PSW в стек не записывается), после чего записывает в PC адрес соответствующей подпрограммы обработки прерывания:

Источник прерывания	Адрес подпрограммы (вектор прерывания)
IE0	0003 H
TF0	000B H
IE1	0013 H
TF1	001B H
TI+RI	0023 H

При выполнении аппаратно-реализуемой команды LCALL в ячейку стека с младшим адресом загружаются разряды 0—7 счетчика команд, а в следующую ячейку стека — разряды 8—15 счетчика команд.

Подпрограмма обслуживания прерывания продолжается до выполнения команды RETI. Команда RETI восстанавливает состояние логики прерывания и загружает в счетчик команд PC 2 байта адреса возврата из двух верхних ячеек стека. Восстановление состояния логики прерывания заключается в следующем: при переходе по вектору на подпрограмму обработки прерывания автоматически до выполнения команды RETI независимо от состояния бит регистра IE запрещаются все прерывания с уровнем приоритета, равным уровню приоритета обслуживаемого прерывания, т. е. вложенные прерывания с равными уровнями приоритета невозможны. Команда RETI снимает этот запрет. При использовании команды RET восстанавливается только состояние счетчика команд, т. е. происходит возврат в прерванную программу. Состояние логики прерывания команда RET не меняет, т. е. логика управления обслуживанием прерываний по-прежнему считает, что продолжает обслуживаться прерывание, подпрограмма обработки которого была закончена командой RET.

Организация пошагового режима работы в ОМЭВМ семейства МК51 может быть реализована с использованием особенностей системы прерывания при очень небольших затратах в плане дополнительного программного обеспечения. Как уже отмечалось, при переходе по вектору на подпрограмму обработки прерывания автоматически до выполнения команды RETI запрещаются все прерывания с уровнем приоритета, равным уровню приоритета обслуживаемого прерывания. После выполнения команды RETI обязательно будет выполнена минимум одна команда прерывания программы, после чего возможен следующий переход на обработку прерывания. Использовать эту особенность для организации пошагового режима можно следующим образом:

- запрограммировать одно из внешних прерываний (к примеру, $\overline{INT0}$) на активизацию по уровню;
- закончить подпрограмму обработки прерывания от $\overline{INT0}$ следующей последовательностью команд:

```

LABEL1:  JNB P3.2, LABEL1      ; ожидание "1" на входе P3.2 ( $\overline{INT0}$ )
LABEL2:  JB  P3.2, LABEL2      ; ожидание "0" на входе P3.2 ( $\overline{INT0}$ )
          RETI                 ; возврат и исполнение одной команды

```

- задать на входе P3.2 ($\overline{INT0}$) постоянный уровень "0" с возможностью подачи единичных импульсов, к примеру, с помощью кнопки "ШАГ".

После того, как указанные манипуляции выполнены, будет происходить следующее: выполнится одна команда основной программы, после чего управление будет передано подпрограмме обработки прерывания по $\overline{INT0}$, которая не сможет завершиться до тех пор, пока на входе $\overline{INT0}$ не будет зафиксирован импульс "0"—"1"—"0". После прохождения такого импульса, задаваемого кнопкой "ШАГ" выполнится команда RETI, управление возвратится в основную программу, где будет выполнена одна команда, после чего вновь начнет обрабатываться прерывание по $\overline{INT0}$. Таким образом, одно нажатие кнопки "ШАГ" (один импульс "0"—"1"—"0" на входе $\overline{INT0}$) вызывает выполнение одной команды основной программы.

2.3.5. Организация памяти

Все ОМЭВМ семейства МК51 имеют несколько адресных пространств, функционально и логически разделенных за счет разницы в механизмах адресации и сигналах управления записью и чтением:

- память программ;
- внутренняя память данных;
- внешняя память данных.

Структура адресного пространства ОМЭВМ показана на рис. 2.24. Слева приводятся адреса соответствующих областей памяти.



Рис. 2.24. Пространство памяти ОМЭВМ

Память программ имеет 16-битовую адресную шину, ее элементы адресуются с использованием счетчика команд (PC) или инструкций, которые вырабатывают 16-разрядные адреса.

Память программ доступна только по чтению. ОМЭВМ не имеют команд и управляющих сигналов, предназначенных для записи в память программ. Память программ имеет байтовую организацию и общий объем до 64 Кбайт. Ряд ОМЭВМ (KP1816BE51, KM1816BE751, KP1830BE51) содержат расположенную на кристалле внутреннюю память программ емкостью 4 Кбайт, которая может быть расширена до 64 Кбайт за счет подключения микросхем внешней памяти программ. Внутренняя память программ KP1816BE51 и KP1830BE51 представляет собой ПЗУ,

формируемое при изготовлении ОМЭВМ. Внутренняя память программ КМ1816ВЕ751 является ППЗУ с ультрафиолетовым стиранием.

Таким образом, для ОМЭВМ КР1816ВЕ51, КМ1816ВЕ751 и КР1830ВЕ51 внутренняя и внешняя память программ разделены в соотношении 4 К/60 К.

ОМЭВМ КР1816ВЕ31 и КР1830ВЕ31 не имеют внутренней памяти программ и могут работать только с внешней емкостью до 64 Кбайт.

С точки зрения программиста имеется только один вид памяти программ объемом 64 К. Тот факт, что в ряде ОМЭВМ он образуется комбинацией массивов, находящихся на кристалле и вне его, в соотношении 4 К/60 К для программиста неощутим, так как АЛУ автоматически выбирает байт из соответствующего массива в соответствии с его адресом.

Сигналом, стробирующим выборку и ввод байта из внешней памяти программ в ОМЭВМ является сигнал ОМЭВМ РМЕ. Для ОМЭВМ, содержащих внутреннюю память программ, РМЕ формируется только в том случае, если адрес в счетчике команд превосходит максимальный адрес внутренней памяти программ 0FFFFH (т. е. для выборок из внутренней памяти программ РМЕ не формируется).

Для ОМЭВМ, не имеющих внутренней памяти программ, РМЕ формируется при любом обращении к памяти программ.

ОМЭВМ семейства МК51 имеют внешний вывод DEMA, с помощью которого можно запретить работу внутренней памяти программ, для чего необходимо подать на вывод DEMA "0". При этом внутренняя память программ отключается и, начиная с нулевого адреса, все обращения происходят к внешней памяти программ с формированием сигнала РМЕ. В случае, если DEMA=1, работают и внутренняя и внешняя память программ. Для ОМЭВМ, не имеющих внутренней памяти программ, для нормальной работы всегда необходимо задавать DEMA=0.

Таким образом, доступ к внешней памяти программ осуществляется в двух случаях:

- 1) при действии сигнала DEMA=0 независимо от адреса обращения,
- 2) в любом случае, если программный счетчик (РС) содержит число, большее чем 0FFFFH.

Если центральный процессор осуществляет доступ к внешней памяти программ, сигнал РМЕ вырабатывается дважды во время каждого машинного цикла (исключение составляет команда MOVX) независимо от того, необходим или нет выбираемый байт для текущей команды. При выборке из внешней памяти программ всегда используется 16-битовый адрес, младший байт которого выдается через порт P0, а старший байт — через порт P2 ОМЭВМ. Байт из внешней памяти программ вводится в ОМЭВМ через порт P0, который в этом случае используется как шина адреса/данных в режиме мультиплексирования.

На рис. 2.24а показаны младшие адреса памяти программ, которые, как правило, отводятся под обработку прерываний и начало работы ОМЭВМ после сброса.

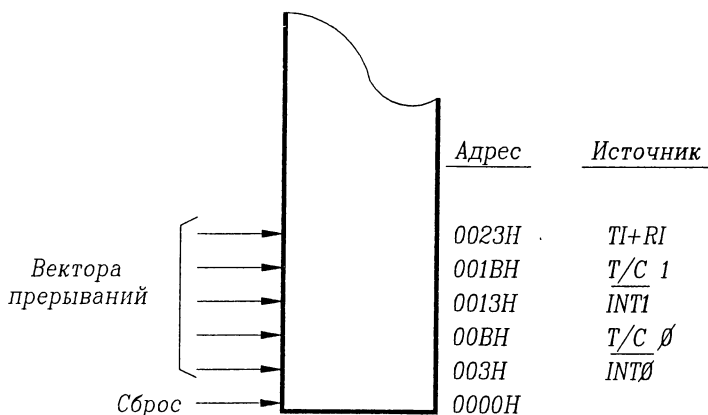


Рис. 2.24а. Младшие адреса памяти программ

Внутренняя память данных ОМЭВМ состоит из двух областей: 128 байт оперативной памяти (ОЗУ) с адресами \emptyset —7FH и области регистров специальных функций, занимающей адреса 80H—FFH. Распределение пространства внутренней памяти данных показано на рис. 2.25. Физически внутреннее ОЗУ данных и область регистров специальных функций являются отдельными устройствами.

Все ячейки внутреннего ОЗУ данных могут адресоваться с использованием прямой и косвенной адресации (режимы адресации описаны в описании системы команд ОМЭВМ). Кроме того, внутреннее ОЗУ данных имеет следующие особенности.

Младшие 32 байта внутреннего ОЗУ данных сгруппированы в 4 банка по 8 регистров в каждом (БАНК0—БАНК3 на рис. 2.25). Команды программы могут обращаться к регистрам, используя их имена R0—R7. Два бита PSW (указатели банка рабочих регистров RS0 и RS1) определяют, с регистрами какого банка производятся манипуляции. Наличие такого механизма работы с ячейками ОЗУ позволяет экономить память программ, т. к. команды, работающие с регистрами R0—R7, короче команд, использующих прямую адресацию.

Следующие после банков регистров внутреннего ОЗУ данных 16 байт (адреса 20H—2FH) образуют область ячеек, к которым возможна побитовая адресация. Набор команд ОМЭВМ семейства МК51 содержит значительное количество инструкций, позволяющих работать с отдельными битами, используя при этом прямую адресацию. 128 бит, составляющих рассматриваемую область внутреннего ОЗУ данных, имеют адреса 00H—7FH и предназначены для работы с такими инструкциями. Битовая адресация ОЗУ показана на рис. 2.26, где в квадратах, символизирующих биты, указаны их адреса.

Обращение к внутреннему ОЗУ данных всегда осуществляется с использованием 8-разрядного адреса. При включении питания содержимое ОЗУ будет иметь случайное значение.



Рис. 2.25. Адресное пространство внутренней памяти данных

Область регистров специальных функций содержит защелки портов, регистры таймеров/счетчиков, регистры управления и т.п. Полный список регистров

специальных функций с их адресами приведен в табл. 2.1а раздела 2.1. Эти регистры допускают только прямую адресацию. Двенадцать байт в области регистров специальных функций допускают как байтовую, так и побитовую адресацию. Список регистров с побитовой адресацией показан на рис. 2.27. Как видно из рисунка, побитовую адресацию допускают те регистры специальных функций, чей адрес заканчивается 000В. Биты в рассматриваемой области регистров специальных функций имеют адреса 80Н—F7Н.

Адрес байта	ст. бит (D7)								мл. бит (D0)	
2FH	7F	7E	7D	7C	7B	7A	79	78		
2EH	77	76	75	74	73	72	71	70		
2DH	6F	6E	6D	6C	6B	6A	69	68		
2CH	67	66	65	64	63	62	61	60		
2BH	5F	5E	5D	5C	5B	5A	59	58		
2AH	57	56	55	54	53	52	51	50		
29H	4F	4E	4D	4C	4B	4A	49	48		
28H	47	46	45	44	43	42	41	40		
27H	3F	3E	3D	3C	3B	3A	39	38		
26H	37	36	35	34	33	32	31	30		
25H	2F	2E	2D	2C	2B	2A	29	28		
24H	27	26	25	24	23	22	21	20		
23H	1F	1E	1D	1C	1B	1A	19	18		
22H	17	16	15	14	13	12	11	10		
21H	0F	0E	0D	0C	0B	0A	09	08		
20H	07	06	05	04	03	02	01	00		
1FH	БАНК 3								R7	
18H									R0	
17H	БАНК 2								R7	
10H									R0	
0FH	БАНК 1								R7	
08H									R0	
07H	БАНК 0								R7	
00									R0	

Рис. 2.26. Битовая адресация ОЗУ

Прямые адреса байтов	Идентифи- каторы программно адресуемых регистров и битов	ст. бит (D7)				мл. бит (D0)			
		F7	F6	F5	F4	F3	F2	F1	F0
0F0H	B								
0E0H	ACC	E7	E6	E5	E4	E3	E2	E1	E0
0D0H	PSW	CV	AC	F0	RS1	RS0	OV	P	
0B8H	IP	D7	D6	D5	D4	D3	D2	D1	D0
0B0H	P3	PT2 PS PT1 PX1 PT0 PX0							
0A8H	IE	—	—	BD	BC	BB	BA	B9	B8
0A0H	P2	B7	B6	B5	B4	B3	B2	B1	B0
98H	SCON	EA		ET2	ES	ET1	EX1	ET0	EX0
90H	P1	AF	—	AD	AC	AB	AA	A9	A8
88H	TCON	A7	A6	A5	A4	A3	A2	A1	A0
80H	P0	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
		9F	9E	9D	9C	9B	9A	99	98
		97	96	95	94	93	92	91	90
		TF1	TR1	TE0	TR0	IE1	IT1	IE0	IT0
		8F	8E	8D	8C	8B	8A	89	88
		87	86	85	84	83	82	81	80

Рис. 2.27. Адреса битов регистров специальных функций

Внешняя память данных формируется дополнительными микросхемами памяти, подключаемыми к ОМЭВМ и может иметь емкость до 64 Кбайт. Пространства внутренней и внешней памяти данных не пересекаются, т.к. доступ к ним осуществляется с помощью разных команд. Для работы с внешней памятью данных существуют специальные команды MOVX, которые не влияют на внутреннюю память данных ОМЭВМ. Таким образом, в системе могут одновременно присутствовать внутренняя память данных с адресами 00H—FFH и внешняя память данных с адресами 0000H—FFFFH. Обращение к ячейкам внешней памяти данных осуществляется только с использованием косвенной адресации по регистрам R0 и R1

активного банка регистров внутреннего ОЗУ (команды типа MOV @Ri) или по регистру специальных функций DPTR (команды типа MOV @DPTR). Соответственно в первом случае будет формироваться 8-разрядный, а во втором случае 16-разрядный адреса внешней памяти данных.

При обращении к внешней памяти данных адрес выводится через порт P0 (младший байт) и порт P2 (старший байт) ОМЭВМ. Обмен байтом данных (запись и чтение) производится через порт P0 ОМЭВМ, т. е. порт P0 используется как шина адреса/данных в режиме мультиплексирования.

Считывание данных из внешней памяти данных в ОМЭВМ производится с помощью выходного сигнала ОМЭВМ \overline{RD} , а запись данных из ОМЭВМ во внешнюю память данных с помощью выходного сигнала ОМЭВМ \overline{WR} .

Каждый тип внешней памяти (память программ, память данных) может быть добавлен независимо от другого и каждый использует те же адреса и шины данных, но различные сигналы управления.

2.4. Система команд

Система команд ОМЭВМ предоставляет большие возможности обработки данных, обеспечивает реализацию логических, арифметических операций, а также управление в режиме реального времени. Реализована побитовая, потетрадная (4 бита), побайтовая (8 бит) и 16-разрядная обработка данных.

БИС семейства МК51 — 8-разрядная ОМЭВМ: ПЗУ, ОЗУ, регистры специального назначения, АЛУ и внешние шины имеют байтовую организацию. Двухбайтовые данные используются только регистром-указателем (DPTR) и счетчиком команд (PC). Следует отметить, что регистр-указатель данных может быть использован как двухбайтовый регистр DPTR или как два однобайтовых регистра специального назначения DPH и DPL. Счетчик команд всегда используется как двухбайтовый регистр.

Набор команд ОМЭВМ имеет 42 мнемонических обозначения (аббревиатур) команд для конкретизации 33 функций этой системы.

Синтаксис большинства команд ассемблерного языка ОМЭВМ состоит из мнемонического обозначения функции, вслед за которым идут операнды, указывающие методы адресации и типы данных. Различные типы данных или режимы адресации определяются установленными операндами, а не изменениями мнемонических обозначений. Например, аббревиатура "MOV" используется восемнадцатью различными командами для обработки трех типов данных (битов, байтов, адресов) в различных адресных пространствах.

Мнемонические обозначения функций однозначно связаны с конкретными комбинациями способов адресации и типами данных. Всего в системе команд возможно 111 таких сочетаний. В табл. 2.20 представлен перечень команд, упорядоченных по алфавиту, а в табл. 2.21 — упорядоченных по кодам.

Таблица 2.20

Мнемоника	Код	Кол-во байт	Кол-во циклов	Страница в описании
ACALL addr11	11	2	2	188
	31	2	2	
	51	2	2	
	71	2	2	
	91	2	2	
	B1	2	2	
	D1	2	2	
	F1	2	2	

Продолжение таблицы 2.20

Мнемоника	Код	Кол-во байт	Кол-во циклов	Страница в описании
ADD A,R0	28	1	1	188
A,R1	29	1	1	188
A,R2	2A	1	1	
A,R3	2B	1	1	
A,R4	2C	1	1	
A,R5	2D	1	1	
A,R6	2E	1	1	
A,R7	2F	1	1	
ADD A,#data	24	2	1	189
ADD A,direct	25	2	1	189
ADD A,@R0	26	1	1	189
A,@R1	27	1	1	189
ADDC A,R0	38	1	1	190
A,R1	39	1	1	
A,R2	3A	1	1	
A,R3	3B	1	1	
A,R4	3C	1	1	
A,R5	3D	1	1	
A,R6	3E	1	1	
A,R7	3F	1	1	
ADDC A,#data	34	2	1	190
ADDC A,direct	35	2	1	190
ADDC A,@R0	36	1	1	190
A,@R1	37	1	1	190
AJMP addr11	01	2	2	190
	21	2	2	
	41	2	2	
	61	2	2	
	81	2	2	
	A1	2	2	
	C1	2	2	
	E1	2	2	
ANL A,R0	58	1	1	191
A,R1	59	1	1	
A,R2	5A	1	1	
A,R3	5B	1	1	
A,R4	5C	1	1	
A,R5	5D	1	1	
A,R6	5E	1	1	
A,R7	5F	1	1	
ANL A,#data	54	2	1	191
ANL A,direct	55	2	1	191
ANL A,@R0	56	1	1	191
A,@R1	57	1	1	
ANL direct,A	52	1	1	192
ANL direct,#data	53	3	2	192
ANL C,bit	82	2	2	192
ANL C,/bit	B0	2	2	192
CJNE A,#data,addr	B4	3	2	193

Продолжение таблицы 2.20

Мнемоника	Код	Кол-во байт	Кол-во циклов	Страница в описании
CJNE A, direct, addr	B5	3	2	193
CJNE R0, #data, addr	B8	3	2	193
R1, #data, addr	B9	3	2	
R2, #data, addr	BA	3	2	
R3, #data, addr	BB	3	2	
R4, #data, addr	BC	3	2	
R5, #data, addr	BD	3	2	
R6, #data, addr	BE	3	2	
R7, #data, addr	BF	3	2	
CJNE @R0, #data, addr	B6	3	2	194
@R1, #data, addr	B7	3	2	
CLR A	E4	1	1	194
CLR bit	C2	2	1	194
CLR C	C3	1	1	194
CPL A	F4	1	1	195
CPL bit	B2	2	1	195
CPL C	B3	1	1	195
DA A	D4	1	1	195
DEC A	14	1	1	196
DEC direct	15	2	1	197
DEC R0	18	1	1	196
R1	19	1	1	
R2	1A	1	1	
R3	1B	1	1	
R4	1C	1	1	
R5	1D	1	1	
R6	1E	1	1	
R7	1F	1	1	
DEC @R0	16	1	1	197
@R1	17	1	1	
DIV AB	84	1	4	197
DJNZ R0, addr	D8	2	2	198
R1, addr	D9	2	2	
R2, addr	DA	2	2	
R3, addr	DB	2	2	
R4, addr	DC	2	2	
R5, addr	DD	2	2	
R6, addr	DE	2	2	
R7, addr	DF	2	2	
DJNZ direct, addr	D5	3	2	198
INC A	04	1	1	199
INC direct	05	2	1	199
INC DPTR	A3	1	2	199
INC R0	08	1	1	199
R1	09	1	1	
R2	0A	1	1	
R3	0B	1	1	
R4	0C	1	1	
R5	0D	1	1	

Продолжение таблицы 2.20

Мнемоника	Код	Кол-во байт	Кол-во циклов	Страница в описании
INC R6	0E	1	1	199
INC R7	0F	1	1	199
INC @R0	06	1	1	199
INC @R1	07	1	1	
JB bit,addr	20	3	2	200
JBC bit,addr	10	3	2	200
JC addr	40	2	2	200
JMP @A+DPTR	73	1	2	201
JNB bit,addr	30	3	2	201
JNC addr	50	2	2	202
JNZ addr	70	2	2	202
JZ addr	60	2	2	202
LCALL addr16	12	3	2	203
LJMP addr16	02	3	2	203
MOV A,direct	E5	2	1	204
MOV A,R0	E8	1	1	204
MOV A,R1	E9	1	1	
MOV A,R2	EA	1	1	
MOV A,R3	EB	1	1	
MOV A,R4	EC	1	1	
MOV A,R5	ED	1	1	
MOV A,R6	EE	1	1	
MOV A,R7	EF	1	1	
MOV A,@R0	E6	1	1	204
MOV A,@R1	E7	1	1	
MOV A,#data	74	2	1	204
MOV bit,C	92	2	2	206
MOV C,bit	A2	2	1	206
MOV direct,A	F5	2	1	205
MOV direct,#data	75	3	2	205
MOV direct,direct	85	3	2	205
MOV direct,R0	88	2	2	205
MOV direct,R1	89	2	2	
MOV direct,R2	8A	2	2	
MOV direct,R3	8B	2	2	
MOV direct,R4	8C	2	2	
MOV direct,R5	8D	2	2	
MOV direct,R6	8E	2	2	
MOV direct,R7	8F	2	2	
MOV direct,@R0	86	2	2	205
MOV direct,@R1	87	2	2	
MOV DPTR,#data16	90	3	2	207
MOV R0,A	F8	1	1	204
MOV R1,A	F9	1	1	
MOV R2,A	FA	1	1	
MOV R3,A	FB	1	1	
MOV R4,A	FC	1	1	
MOV R5,A	FD	1	1	
MOV R6,A	FE	1	1	
MOV R7,A	FF	1	1	

Продолжение таблицы 2.20

Мнемоника	Код	Кол-во байт	Кол-во циклов	Страница в описании
MOV @R0,A	F6	1	1	206
@R1,A	F7	1	1	
MOV R0,#data	78	2	1	205
R1,#data	79	2	1	
R2,#data	7A	2	1	
R3,#data	7B	2	1	
R4,#data	7C	2	1	
R5,#data	7D	2	1	
R6,#data	7E	2	1	
R7,#data	7F	2	1	
MOV @R0,#data	76	2	1	206
@R1,#data	77	2	1	
MOV R0,direct	A8	2	2	205
R1,direct	A9	2	2	
R2,direct	AA	2	2	
R3,direct	AB	2	2	
R4,direct	AC	2	2	
R5,direct	AD	2	2	
R6,direct	AE	2	2	
R7,direct	AF	2	2	
MOV @R0,direct	A6	2	2	206
@R1,direct	A7	2	2	
MOVC A,@A+DPTR	93	1	2	207
MOVC A,@A+PC	83	1	2	207
MOVX A,@DPTR	E0	1	2	208
MOVX A,@R0	E2	1	2	208
A,@R1	E3	1	2	
MOVX @DPTR,A	F0	1	2	208
MOVX @R0,A	F2	1	2	208
@R1,A	F3	1	2	
MUL AB	A4	1	4	209
NOP	00	1	1	209
ORL A,R0	48	1	1	210
A,R1	49	1	1	
A,R2	4A	1	1	
A,R3	4B	1	1	
A,R4	4C	1	1	
A,R5	4D	1	1	
A,R6	4E	1	1	
A,R7	4F	1	1	
ORL A,direct	45	2	1	210
ORL A,#data	44	2	1	210
ORL A,@R0	46	1	1	210
A,@R1	47	1	1	
ORL C,bit	72	2	2	211
ORL C,/bit	A0	2	2	211
ORL direct,A	42	2	1	210
ORL direct,#data	43	3	2	210
POP direct	D0	2	2	211
PUSH direct	C0	2	2	212

Продолжение таблицы 2.20

Мнемоника	Код	Кол-во байт	Кол-во циклов	Страница в описании
RET	22	1	2	212
RETI	32	1	2	212
RL A	23	1	1	213
RLC A	33	1	1	213
RR A	03	1	1	213
RRC A	13	1	1	214
SETB bit	D2	2	1	214
SETB C	D3	1	1	214
SJMP addr	80	2	2	214
SUBB A,R0	98	1	1	215
A,R1	99	1	1	
A,R2	9A	1	1	
A,R3	9B	1	1	
A,R4	9C	1	1	
A,R5	9D	1	1	
A,R6	9E	1	1	
A,R7	9F	1	1	
SUBB A,#data	94	2	1	216
SUBB A,direct	95	2	1	215
SUBB A,@R0	96	1	1	216
A,@R1	97	1	1	
SWAP A	C4	1	1	216
XCH A,R0	C8	1	1	216
A,R1	C9	1	1	
A,R2	CA	1	1	
A,R3	CB	1	1	
A,R4	CC	1	1	
A,R5	CD	1	1	
A,R6	CE	1	1	
A,R7	CF	1	1	
XCH A,direct	C5	2	1	217
XCH A,@R0	C6	1	1	217
A,@R1	C7	1	1	
XCHD A,@R0	D6	1	1	217
A,@R1	D7	1	1	
XRL A,R0	68	1	1	217
A,R1	69	1	1	
A,R2	6A	1	1	
A,R3	6B	1	1	
A,R4	6C	1	1	
A,R5	6D	1	1	
A,R6	6E	1	1	
A,R7	6F	1	1	
XRL A,direct	65	2	1	218
XRL A,#data	64	2	1	218
XRL A,@R0	66	1	1	218
A,@R1	67	1	1	
XRL direct,A	62	2	1	218
XRL direct,#data	63	3	2	218

Таблица 2.21

Код	Мнемоника	Кол-во байт	Кол-во циклов	Страница в описании
00	NOP	1	1	209
01	AJMP addr	2	2	190
02	LJMP addr	3	2	203
03	RR A	1	1	213
04	INC A	1	1	199
05	INC direct	2	1	199
06	INC @R0	1	1	199
07	INC @R1	1	1	199
08	INC R0	1	1	199
09	INC R1	1	1	199
0A	INC R2	1	1	199
0B	INC R3	1	1	199
0C	INC R4	1	1	199
0D	INC R5	1	1	199
0E	INC R6	1	1	199
0F	INC R7	1	1	199
10	JBC bit,addr	3	2	200
11	ACALL addr	2	2	188
12	LCALL addr	3	2	203
13	RRC A	1	1	214
14	DEC A	1	1	196
15	DEC direct	2	1	197
16	DEC @R0	1	1	197
17	DEC @R1	1	1	197
18	DEC R0	1	1	196
19	DEC R1	1	1	196
1A	DEC R2	1	1	196
1B	DEC R3	1	1	196
1C	DEC R4	1	1	196
1D	DEC R5	1	1	196
1E	DEC R6	1	1	196
1F	DEC R7	1	1	196
20	JB bit,addr	3	2	200
21	AJMP addr	2	2	190
22	RET	1	2	212
23	RL A	1	1	213
24	ADD A,#data	2	1	189
25	ADD A,direct	2	1	189
26	ADD A,@R0	1	1	189
27	ADD A,@R1	1	1	189
28	ADD A,R0	1	1	188
29	ADD A,R1	1	1	188
2A	ADD A,R2	1	1	188
2B	ADD A,R3	1	1	188
2C	ADD A,R4	1	1	188
2D	ADD A,R5	1	1	188
2E	ADD A,R6	1	1	188
2F	ADD A,R7	1	1	188
30	JNB bit,addr	3	2	201

Продолжение таблицы 2.21

Код	Мнемоника	Кол-во байт	Кол-во циклов	Страница в описании
31	ACALL addr	2	2	188
32	RETI	1	2	212
33	RLC A	1	1	213
34	ADDC A,#data	2	1	190
35	ADDC A,direct	2	1	190
36	ADDC A,@R0	1	1	190
37	ADDC A,@R1	1	1	190
38	ADDC A,R0	1	1	190
39	ADDC A,R1	1	1	190
3A	ADDC A,R2	1	1	190
3B	ADDC A,R3	1	1	190
3C	ADDC A,R4	1	1	190
3D	ADDC A,R5	1	1	190
3E	ADDC A,R6	1	1	190
3F	ADDC A,R7	1	1	190
40	JC addr	2	2	200
41	AJMP addr	2	2	190
42	ORL direct,A	2	1	210
43	ORL direct,#data	3	2	210
44	ORL A,#data	2	1	210
45	ORL A,direct	2	1	210
46	ORL A,@R0	1	1	210
47	ORL A,@R1	1	1	210
48	ORL A,R0	1	1	210
49	ORL A,R1	1	1	210
4A	ORL A,R2	1	1	210
4B	ORL A,R3	1	1	210
4C	ORL A,R4	1	1	210
4D	ORL A,R5	1	1	210
4E	ORL A,R6	1	1	210
4F	ORL A,R7	1	1	210
50	JNC addr	2	2	202
51	ACALL addr	2	2	188
52	ANL direct,A	2	1	192
53	ANL direct,#data	3	2	192
54	ANL A,#data	2	1	191
55	ANL A,direct	2	1	191
56	ANL A,@R0	1	1	191
57	ANL A,@R1	1	1	191
58	ANL A,R0	1	1	191
59	ANL A,R1	1	1	191
5A	ANL A,R2	1	1	191
5B	ANL A,R3	1	1	191
5C	ANL A,R4	1	1	191
5D	ANL A,R5	1	1	191
5E	ANL A,R6	1	1	191
5F	ANL A,R7	1	1	191
60	JZ addr	2	2	202
61	AJMP addr	2	2	190

Продолжение таблицы 2.21

Код	Мнемоника	Кол-во байт	Кол-во циклов	Страница в описании
62	XRL direct,A	2	1	218
63	XRL direct,#data	3	2	218
64	XRL A,#data	2	1	218
65	XRL A,direct	2	1	218
66	XRL A,@R0	1	1	218
67	XRL A,@R1	1	1	218
68	XRL A,R0	1	1	217
69	XRL A,R1	1	1	217
6A	XRL A,R2	1	1	217
6B	XRL A,R3	1	1	217
6C	XRL A,R4	1	1	217
6D	XRL A,R5	1	1	217
6E	XRL A,R6	1	1	217
6F	XRL A,R7	1	1	217
70	JNZ addr	2	2	202
71	ACALL addr	2	2	188
72	ORL C,bit	2	2	211
73	JMP @A+DPTR	1	2	201
74	MOV A,#data	2	1	204
75	MOV direct,#data	3	2	205
76	MOV @R0,#data	2	1	206
77	MOV @R1,#data	2	1	206
78	MOV R0,#data	2	1	205
79	MOV R1,#data	2	1	205
7A	MOV R2,#data	2	1	205
7B	MOV R3,#data	2	1	205
7C	MOV R4,#data	2	1	205
7D	MOV R5,#data	2	1	205
7E	MOV R6,#data	2	1	205
7F	MOV R7,#data	2	1	205
80	SJMP addr	2	2	214
81	AJMP addr	2	2	190
82	ANL C,bit	2	2	192
83	MOVC A,@A+PC	1	2	207
84	DIV AB	1	4	197
85	MOV direct,direct	3	2	205
86	MOV direct,@R0	2	2	205
87	MOV direct,@R1	2	2	205
88	MOV direct,R0	2	2	205
89	MOV direct,R1	2	2	205
8A	MOV direct,R2	2	2	205
8B	MOV direct,R3	2	2	205
8C	MOV direct,R4	2	2	205
8D	MOV direct,R5	2	2	205
8E	MOV direct,R6	2	2	205
8F	MOV direct,R7	2	2	205
90	MOV DPTR,#data	3	2	207
91	ACALL addr	2	2	188
92	MOV bit,C	2	2	206

Продолжение таблицы 2.21

Код	Мнемоника	Кол-во байт	Кол-во циклов	Страница в описании
93	MOVC A,@A+DPTR	1	2	207
94	SUBB A,#data	2	1	216
95	SUBB A,direct	2	1	216
96	SUBB A,@R0	1	1	216
97	SUBB A,@R1	1	1	216
98	SUBB A,R0	1	1	215
99	SUBB A,R1	1	1	215
9A	SUBB A,R2	1	1	215
9B	SUBB A,R3	1	1	215
9C	SUBB A,R4	1	1	215
9D	SUBB A,R5	1	1	215
9E	SUBB A,R6	1	1	215
9F	SUBB A,R7	1	1	215
A0	ORL C,/bit	2	2	211
A1	AJMP addr	2	2	190
A2	MOV C,bit	2	1	206
A3	INC DPTR	1	2	199
A4	MUL AB	1	4	209
A5	—	—	—	—
A6	MOV @R0,direct	2	2	206
A7	MOV @R1,direct	2	2	206
A8	MOV R0,direct	2	2	205
A9	MOV R1,direct	2	2	205
AA	MOV R2,direct	2	2	205
AB	MOV R3,direct	2	2	205
AC	MOV R4,direct	2	2	205
AD	MOV R5,direct	2	2	205
AE	MOV R6,direct	2	2	205
AF	MOV R7,direct	2	2	205
B0	ANL C,/bit	2	2	192
B1	ACALL addr	2	2	188
B2	CPL bit	2	1	195
B3	CPL C	1	1	195
B4	CJNE A,#data,addr	3	2	193
B5	CJNE A,direct,addr	3	2	193
B6	CJNE @R0,#data,addr	3	2	194
B7	CJNE @R1,#data,addr	3	2	194
B8	CJNE R0,#data,addr	3	2	193
B9	CJNE R1,#data,addr	3	2	193
BA	CJNE R2,#data,addr	3	2	193
BB	CJNE R3,#data,addr	3	2	193
BC	CJNE R4,#data,addr	3	2	193
BD	CJNE R5,#data,addr	3	2	193
BE	CJNE R6,#data,addr	3	2	193
BF	CJNE R7,#data,addr	3	2	193
C0	PUSH direct	2	2	212
C1	AJMP addr	2	2	190
C2	CLR bit	2	1	194
C3	CLR C	1	1	194

Продолжение таблицы 2.21

Код	Мнемоника	Кол-во байт	Кол-во циклов	Страница в описании
C4	SWAP A	1	1	216
C5	XCH A,direct	2	1	217
C6	XCH A,@R0	1	1	217
C7	XCH A,@R1	1	1	217
C8	XCH A,R0	1	1	216
C9	XCH A,R1	1	1	216
CA	XCH A,R2	1	1	216
CB	XCH A,R3	1	1	216
CC	XCH A,R4	1	1	216
CD	XCH A,R5	1	1	216
CE	XCH A,R6	1	1	216
CF	XCH A,R7	1	1	216
D0	POP direct	2	2	211
D1	ACALL addr	2	2	188
D2	SETB bit	2	1	214
D3	SETB C	1	1	214
D4	DA A	1	1	195
D5	DJNZ direct,addr	3	2	198
D6	XCHD A,@R0	1	1	217
D7	XCHD A,@R1	1	1	217
D8	DJNZ R0,addr	2	2	198
D9	DJNZ R1,addr	2	2	198
DA	DJNZ R2,addr	2	2	198
DB	DJNZ R3,addr	2	2	198
DC	DJNZ R4,addr	2	2	198
DD	DJNZ R5,addr	2	2	198
DE	DJNZ R6,addr	2	2	198
DF	DJNZ R7,addr	2	2	198
E0	MOVX A,@DPTR	1	2	208
E1	AJMP addr	2	2	190
E2	MOVX A,@R0	1	2	208
E3	MOVX A,@R1	1	2	208
E4	CLR A	1	1	194
E5	MOV A,direct	2	1	204
E6	MOV A,@R0	1	1	204
E7	MOV A,@R1	1	1	204
E8	MOV A,R0	1	1	204
E9	MOV A,R1	1	1	204
EA	MOV A,R2	1	1	204
EB	MOV A,R3	1	1	204
EC	MOV A,R4	1	1	204
ED	MOV A,R5	1	1	204
EE	MOV A,R6	1	1	204
EF	MOV A,R7	1	1	204
F0	MOVX @DPTR,A	1	2	208
F1	ACALL addr	2	2	188
F2	MOVX @R0,A	1	2	208
F3	MOVX @R1,A	1	2	208
F4	CPL A	1	1	195
F5	MOV direct,A	2	1	205

Продолжение таблицы 2.21

Код	Мнемоника	Кол-во байт	Кол-во циклов	Страница в описании
F6	MOV @R0, A	1	1	206
F7	MOV @R1, A	1	1	206
F8	MOV R0, A	1	1	204
F9	MOV R1, A	1	1	204
FA	MOV R2, A	1	1	204
FB	MOV R3, A	1	1	204
FC	MOV R4, A	1	1	204
FD	MOV R5, A	1	1	204
FE	MOV R6, A	1	1	204
FF	MOV R7, A	1	1	204

В машинном коде команда занимает один, два или три байта.

Команды выполняются за один, два или четыре (умножение и деление) машинных цикла.

При частоте тактового генератора, равной 12 МГц, одноцикловые команды выполняются за 1 мкс, двухцикловые — за 2 мкс и т.д.

Из 111 типов команд 64 выполняются за 1 мкс (12 тактов), 45 команд — за 2 мкс (24 такта) и две команды — умножение и деление (MUL, DIV) выполняются за 4 мкс (48 тактов).

В системе команд ОМЭВМ семейства МК51 по сравнению с системой команд ОМЭВМ типа МК48 отсутствуют специальные команды ввода-вывода, управления таймерами/счетчиками и др. Эти функции реализуются, например, как частные случаи обращения к регистрам специального назначения с помощью прямой адресации:

MOV direct, A;

Все команды условных переходов осуществляются относительно содержимого счетчика команд с адресом перехода, вычисляемым ЦПУ во время выполнения команды.

Трехбайтовые команды перехода и вызова LCALL, LJMP (с 16-разрядным адресом) позволяют осуществлять переход и обращение по любому адресу адресного пространства памяти программ емкостью 64 Кбайт. Если необходим переход в пределах области памяти программ 2 К, то можно использовать команды перехода и вызова с 11-разрядным адресом (ACALL, AJMP). Переход внутри участка памяти, определяемый 8-разрядной величиной смещения, осуществляется по команде SJMP.

В табл. 2.22 приведены конструкции, влияющие на установку флагов.

В табл. 2.23 приведены обозначения и символы, используемые в системе команд.

Систему команд ОМЭВМ условно можно разбить на пять групп:

- арифметические команды;
- логические команды с байтовыми переменными;
- команды передачи данных;
- команды битового процессора;
- команды ветвления программ и передачи управления ОМЭВМ.

2.4.1 Арифметические команды

В наборе команд ОМЭВМ имеются следующие арифметические операции: сложение, сложение с учетом флага переноса, вычитание с заемом, инкрементирование, декрементирование, сравнение, десятичная коррекция, умножение и деление.

Таблица 2.22. Команды, влияющие на установку флагов

Мнемоника	Флаги			Мнемоника	Флаги		
	C	OV	AC		C	OV	AC
ADD	x	x	x	CLR C	Ø		
ADDC	x	x	x	CPL C	x		
SUBB	x	x	x	ANL C, bit	x		
MUL	Ø	x		ANL C, /bit	x		
DIV	Ø	x		ORL C, bit	x		
DA	x			ORL C, /bit	x		
RRC	x			MOV C, /bit	x		
RLC	x			CJNE	x		
SETB C	1						

Примечания:

1. X - флаг равен 0 или 1.
2. Операции над SFR с адресом байта 208 или с адресами битов 209—215 (т.е. PSW или бит PSW) также влияют на установку флагов.

Таблица 2.23

Обозначение, символ	Н а з н а ч е н и е
A	Аккумулятор
Rr	Регистры текущего выбранного банка регистров.
r	Номер загружаемого регистра, указанного в команде.
direct	Прямо адресуемый 8-битовый внутренний адрес ячейки данных, который может быть ячейкой внутреннего ОЗУ данных (0-127) или SFR (128-255).
@Rr	Косвенно адресуемая 8-битовая ячейка внутреннего ОЗУ данных.
data 8	8-битовое непосредственное данные, входящее в код операции (КОП).
data 16	16-битовое непосредственное данные, входящее КОП.

Продолжение таблицы 2.23

Обозначение, символ	Н а з н а ч е н и е
data H	Старшие биты (15–8) непосредственных 16-битовых данных.
data L	Младшие биты (7–0) непосредственных 16-битовых данных.
addr 11	11-битовый адрес назначения.
addr 16	16-битовый адрес назначения.
addr L	Младшие биты адреса назначения.
disp 8	8-битовый байт смещения со знаком.
bit	Бит с прямой адресацией, адрес которого содержит КОП, находящийся во внутреннем ОЗУ данных или SFR.
a15,a14...a0	Биты адреса назначения.
(X)	Содержимое элемента X
((X))	Содержимое по адресу, хранящемуся в элементе X
(X)[M]	Разряд M элемента X
(X)[M1–M2]	Группа разрядов M1–M2 элемента X
+ – * / AND OR XOR /X	Операции: сложения вычитания умножения деления логического умножения (операция И) логического сложения (операция ИЛИ) сложения по модулю 2 (операция "Исключающее ИЛИ") инверсия элемента X

В АЛУ производятся действия над целыми числами без знака. В двухоперандных операциях: сложение (ADD), сложение с переносом (ADDC) и вычитание с заемом (SUBB) аккумулятор является первым операндом и принимает результат операции. Вторым операндом может быть рабочий регистр выбранного банка рабочих регистров, регистр внутренней памяти данных с косвенно-регистровой и прямой адресацией или байт непосредственных данных. Указанные операции влияют на флаги: переполнения, переноса, промежуточного переноса и флаг четности в слове состояния процессора (PSW).

Использование разряда переноса позволяет многократно повысить точность при операциях сложения (ADDC) и вычитания (SUBB).

Выполнение операций сложения и вычитания с учетом знака может быть осуществлено с помощью программного управления флагом переполнения (OV) регистра PSW. Флаг промежуточного переноса (AC) обеспечивает выполнение арифметических операций в двоично-десятичном коде.

Операции инкрементирования и декрементирования на флаги не влияют.

Операции сравнения не влияют ни на операнд назначения, ни на операнд источника, но они влияют на флаг переноса.

Существуют три арифметические операции, которые выполняются только на аккумуляторе: две команды проверки содержимого аккумулятора А (JZ, JNZ), и команда десятичной коррекции при сложении двоично-десятичных кодов.

При операции умножения содержимое аккумулятора А умножается на содержимое регистра В и результат размещается следующим образом: младший байт в регистре В, старший — в регистре А.

В случае выполнения операции деления целое от деления помещается в аккумулятор А, остаток от деления — в регистр В.

2.4.2 Логические команды с байтовыми переменными.

Система команд ОМЭВМ позволяет реализовать логические операции: "И", "ИЛИ", "ИСКЛЮЧАЮЩЕЕ ИЛИ" на регистре-аккумуляторе (А) и байте-источнике. Вторым операндом (байтом-источником) при этом может быть рабочий регистр в выбранном банке рабочих регистров; регистр внутреннего ОЗУ, адресуемый с помощью косвенно-регистровой адресации; прямоадресуемые ячейки внутреннего ОЗУ и регистры специального назначения; непосредственная величина.

Указанные логические операции могут быть реализованы на любом прямоадресуемом регистре внутреннего ОЗУ или регистре специального назначения с использованием в качестве второго операнда содержимого аккумулятора А или непосредственных данных.

Существуют логические операции, которые выполняются только на аккумуляторе: сброс и инвертирование всех восьми разрядов А; циклический сдвиг влево и вправо; циклический сдвиг влево и вправо с учетом флага переноса; обмен местами старшей и младшей тетрад (ниблов) внутри аккумулятора.

2.4.3 Команды передачи данных

Таблицы символов (кодов), зашитые в ПЗУ программы могут быть выбраны с помощью команд передачи данных с использованием косвенной адресации. Байт константы может быть передан в аккумулятор из ячейки памяти программ, адресуемой суммой базового регистра (PC или DPTR) и индексного регистра (содержимого А). Это обеспечивает, например, удобное средство реализации алгоритма преобразования кода ASCII в семисегментный код.

Любая ячейка 256-байтового блока внешнего ОЗУ данных может быть выбрана с использованием косвенно-регистровой адресации через регистры указатели R0 или R1 (выбранного банка рабочих регистров).

Ячейка внутри адресного пространства 64 Кбайт внешнего ОЗУ также может быть выбрана с использованием косвенно-регистровой адресации через регистр-указатель данных DPTR.

Команды передачи между прямоадресуемыми регистрами позволяют заносить величину из порта в ячейку внутреннего ОЗУ без использования рабочих регистров или аккумулятора.

В логическом процессоре любой прямоадресуемый бит может быть помещен в бит переноса и наоборот.

Содержимое аккумулятора может быть обменено с содержимым рабочих регистров (выбранного банка) и с содержимым адресуемых с помощью косвенно-регистровой адресации ячеек внутреннего ОЗУ, а также с содержимым прямо-адресуемых ячеек внутреннего ОЗУ и с содержимым регистров специального назначения.

Младший нибл (разряды 3—0) содержимого аккумулятора, может быть обменен с младшим ниблом содержимого ячеек внутреннего ОЗУ, выбираемых с помощью косвенно-регистровой адресации.

2.4.4 Команды битового процессора.

Битовый процессор является частью архитектуры ОМЭВМ семейства МК51 и его можно рассматривать как независимый процессор побитовой обработки. Битовый процессор выполняет свой набор команд, имеет свое побитово-адресуемое ОЗУ и свой ввод-вывод.

Команды, оперирующие с битами, обеспечивают прямую адресацию 128 битов (0—127) в шестнадцати ячейках внутреннего ОЗУ (ячейки с адресами 20H—2FH) и прямую побитовую адресацию регистров специального назначения, адреса которых кратны восьми:

P0(80H), TCON(88H), P1(90H), SCON(98H), P2(A0H), IE(A8H), P3(B0H), IP(B8H), PSW(D0H), A(E0H), B(F0H).

Каждый из отдельно адресуемых битов может быть установлен в "1", сброшен в "0", инвертирован, проверен. Могут быть реализованы переходы: если бит установлен; если бит не установлен; переход, если бит установлен, со сбросом этого бита; бит может быть перезаписан в (из) разряда переноса. Между любым прямоадресуемым битом и флагом переноса могут быть произведены логические операции "И", "ИЛИ", где результат заносится в разряд флага переноса. Команды побитовой обработки обеспечивают реализацию сложных функций комбинаторной логики и оптимизацию программ пользователя.

2.4.5 Команды ветвления и передачи управления

Адресное пространство памяти программ ОМЭВМ не имеет страничной организации, что позволяет свободно перемещать фрагменты программы внутри адресного пространства, при этом не требуется перезасылка (изменение) номера страницы.

Перемещение отдельных фрагментов программы обеспечивает возможность использования перемещаемых программных модулей различными программами.

Команды 16-разрядных переходов и вызовов подпрограмм позволяют осуществлять переход в любую точку адресного пространства памяти программ объемом 64 Кбайт.

Команды 11-разрядных переходов и вызовов подпрограмм обеспечивают переходы внутри программного модуля емкостью 2 Кбайт. В системе команд имеются команды условных и безусловных переходов относительно начального адреса следующей команды в пределах от (PC)–128 до (PC)+127. Команды проверки отдельных разрядов позволяют осуществлять условные переходы по состоянию "0" или "1" прямоадресуемых битов. Команды проверки содержимого аккумулятора (на ноль / не ноль) позволяют осуществлять условные переходы по содержимому А.

Косвенно-регистровые переходы в системе команд ОМЭВМ обеспечивают ветвление относительно базового регистра (содержимого DPTR или PC) со смещением, находящимся в аккумуляторе А.

2.4.6. Способы адресации операндов

Существуют следующие способы адресации операндов-источников:

- регистровая адресация;
- прямая адресация;
- косвенно-регистровая адресация;
- непосредственная адресация;
- косвенная адресация по сумме базового и индексного регистра.

Первые три способа используются также для адресации операнда назначения. Указанные пять способов адресации, используемые в различных сочетаниях, обеспечивают 21 режим адресации.

Многие команды содержат поля: "приемник", "источник", которые определяют тип данных, метод адресации и участвующие операнды.

Для команд, не выполняющих операции перезаписи, операнд назначения является и операндом-источником.

Большое количество команд включает операнды, расположенные во внутреннем ОЗУ данных ОМЭВМ. Выбор адресного пространства памяти программ или внешней

памяти данных в качестве второго операнда определяется командной мнемоникой (если только второй операнд не является непосредственной величиной).

Адресуемая область внутреннего ОЗУ данных определяется способом адресации и величиной адреса. Например, обращение к регистрам специального назначения может быть выполнено только с помощью прямой адресации.

2.4.6.1 Регистровая адресация

Регистровая адресация используется для обращения к восьми рабочим регистрам выбранного банка рабочих регистров (эти же регистры могут быть выбраны с помощью прямой адресации и косвенно-регистровой адресации как обычные ячейки внутреннего ОЗУ данных).

Регистровая адресация используется также для обращения к регистрам А, В, АВ (сдвоенному регистру), DPTR и к флагу переноса С. Использование регистровой адресации позволяет получать двухбайтовый эквивалент трехбайтовых команд прямой адресации.

2.4.6.2 Прямая адресация

Прямая байтовая адресация используется для обращения к ячейкам внутренней памяти (ОЗУ) данных (0—127) и к регистрам специального назначения.

Прямая побитовая адресация используется для обращения к отдельно адресуемым 128 битам, расположенным в ячейках с адресами 20Н—2FH и к отдельно адресуемым битам регистров специального назначения.

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

2.4.6.2 Косвенно-регистровая адресация

Косвенно-регистровая адресация используется для обращения к ячейкам внутреннего ОЗУ данных. В качестве регистров-указателей используется регистры R0, R1 выбранного банка регистров.

В командах PUSH и POP используется содержимое указателя стека (SP).

Косвенно-регистровая адресация используется также для обращения к внешней памяти данных. В этом случае с помощью регистров-указателей R0 и R1 (выбранного банка рабочих регистров) выбирается ячейка из блока в 256 байт внешней памяти данных. Номер блока предварительно задается содержимым порта P2.

16-разрядный указатель данных (DPTR) может быть использован для обращения к любой ячейке адресного пространства внешней памяти данных объемом до 64 Кбайт.

2.4.6.4 Непосредственная адресация

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

2.4.6.5 Косвенно-регистровая адресация по сумме базового и индексного регистров.

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

2.5 Описание машинных команд

Описание каждой приводимой машинной команды ОМЭВМ типа МК51 состоит из предложения языка ассемблера, кода, длины команды в байтах, времени ее выполнения, алгоритма и примера.

При описании операций используются обозначения, приведенные в табл. 2.23.

Команда ACALL <addr 11>

Команда "абсолютный вызов подпрограммы" вызывает безусловно подпрограмму, размещенную по указанному адресу. При этом счетчик команд увеличивается на 2 для получения адреса следующей команды, после чего полученное 16-битовое значение PC помещается в стек (сначала следует младший байт), и содержимое указателя стека также увеличивается на два. Адрес перехода получается с помощью конкатенации старших бит увеличенного содержимого счетчика команд, битов старшего байта команды и младшего байта команды.

Ассемблер: ACALL <метка>

Код:

A10 A9 A8 1 0 0 0 1

A7 A6 A5 A4 A3 A2 A1 A0

Время: 2 цикла

Алгоритм:

```
(PC):=(PC)+2
(SP):=(SP)+1
((SP)):= (PC[7-0])
(SP):=(SP)+1
((SP)):= (PC[15-8])
(PC[10-0]):=A10A9A8 II A7A6A5A4A3A2A1A0,
где II – знак конкатенации (сцепление)
```

Пример:

```
;ДО ВЫПОЛНЕНИЯ КОМАНДЫ ACALL
; (SP)=07H
;метка MT1 соответствует адресу: 0345H,
;т.е. (PC)=0345H
ACALL MT1 ;расположена по адресу 028DH, т.е.
; (PC)=028DH
;ПОСЛЕ ВЫПОЛНЕНИЯ КОМАНДЫ
; (SP)=09H, (PC)=0345H,
;03H [08]=8FH, 03H [09]=02H.
```

Команда ADD A, <байт-источник>

Эта команда ("сложение") складывает содержимое аккумулятора A с содержимым байта-источника, оставляя результат в аккумуляторе. При появлении переносов из разрядов 7 и 3, устанавливаются флаги переноса (C) и дополнительного переноса (AC) соответственно, в противном случае эти флаги сбрасываются. При сложении целых чисел без знака флаг переноса "C" указывает на появление переполнения. Флаг переполнения (OV) устанавливается, если есть перенос из бита 6 и нет переноса из бита 7, или есть перенос из бита 7 и нет — из бита 6, в противном случае флаг OV сбрасывается. При сложении целых чисел со знаком флаг OV указывает на отрицательную величину, полученную при суммировании двух положительных операндов или на положительную сумму для двух отрицательных операндов.

Для команды сложения разрешены следующие режимы адресации байта-источника:

- 1) регистровый;
- 2) косвенно-регистровый;
- 3) прямой;
- 4) непосредственный.

1) Ассемблер: ADD A, Rn ; где n=0-7

Код:

0 0 1 0 1 rrr

, где rrr=000-111

- Время: 1 цикл
 Алгоритм: $(A) := (A) + (R_n)$, где $n = 0-7$
 $C := X$, $OV := X$, $AC := X$, где $X = (0 \text{ или } 1)$
 Пример: ; $(A) = C3H$, $(R6) = AAH$
 ADD A, R6 ; $(A) = 6DH$, $(R6) = AAH$
 ; $(AC) = 0$, $(C) = 1$, $(OV) = 1$
- 2) Ассемблер: ADD A, @Ri ; где $i = 0, 1$
- Код:

0 0 1 0 0 1 1 i

 , где $i = 0, 1$
- Время: 1 цикл
 Алгоритм: $(A) := (A) + ((R_i))$, где $i = 0, 1$
 $C := X$, $OV := X$, $AC := X$, где $X = (0 \text{ или } 1)$
 Пример: ; $(A) = 95H$, $(R1) = 31H$, $(O3Y [31]) = 4CH$
 ADD A, @R1 ; $(A) = E1H$, $(O3Y [31]) = 4CH$,
 ; $(C) = 0$, $(AC) = 1$, $(OV) = 0$
- 3) Ассемблер: ADD A, <direct>
- Код:

0 0 1 0 0 1 0 1

direct address

- Время: 1 цикл
 Алгоритм: $(A) := (A) + (\text{direct})$
 $C := X$, $OV := X$, $AC := X$, где $X = (0 \text{ или } 1)$
 Пример: ; $(A) = 77H$, $(O3Y [90]) = FFH$
 ADD A, 90H ; $(A) = 76H$, $(O3Y [90]) = FFH$,
 ; $(C) = 1$, $(OV) = 0$, $(AC) = 1$
- 4) Ассемблер: ADD A, <#data>
- Код:

0 0 1 0 0 1 0 0

#data

- Время: 1 цикл
 Алгоритм: $(A) := (A) + \#data$
 $C := X$, $OV := X$, $AC := X$, где $X = (0 \text{ или } 1)$
 Пример: ; $(A) = 09H$
 ADD A, #0D3H ; $(A) = DCH$,
 ; $(C) = 0$, $(OV) = 0$, $(AC) = 0$

Команда ADDC A, <байт-источник>

Эта команда ("сложение с переносом") одновременно складывает содержимое байта-источника, флаг переноса и содержимое аккумулятора A, оставляя результат в аккумуляторе. При этом флаги переноса и дополнительного переноса устанавливаются, если есть перенос из бита 7 или бита 3, и сбрасываются в противном случае. При сложении целых чисел без знака флаг переноса указывает на переполнение. Флаг переполнения (OV) устанавливается, если имеется перенос бита 6 и нет переноса из бита 7 или есть перенос из бита 7 и нет — из бита 6, в противном случае OV сбрасывается. При сложении целых чисел со знаком OV указывает на отрицательную величину, полученную при суммировании двух положительных операндов или на положительную сумму от двух отрицательных операндов.

Для этой команды разрешены следующие режимы адресации байта-источника:

- 1) регистровый;
- 2) косвенно-регистровый;
- 3) прямой;

4) непосредственный.

1) Ассемблер: `ADDC A,Rn ; где n=0-7`

Код:

0 0 1 1 1 rrr

, где rrr=000-111

Время: 1 цикл

Алгоритм: $(A) := (A) + (C) + (Rn)$
 $(C) := X, (AC) := X, (OV) := X$, где $X = (0 \text{ или } 1)$

Пример: ; (A)=B2H, (R3)=99,
 ; (C)=1
`ADDC A,R3 ; (A)=4CH, (R3)=99,`
 ; (C)=1, (AC)=0, (OV)=1

2) Ассемблер: `ADDC A,@Ri ; где i=0,1`

Код:

0 0 1 1 0 1 1 i

, где i=0,1

Время: 1 цикл

Алгоритм: $(A) := (A) + (C) + ((Ri))$
 $(C) := X, (AC) := X, (OV) := X$, где $X = (0 \text{ или } 1)$

Пример: ; (A)=D5H, (R0)=3AH,
 ; (03Y[3A])=1AH, (C)=1
`ADDC A,@R0 ; (A)=F0H, (03Y[3A])=1AH,`
 ; (C)=0, (AC)=1, (OV)=0

3) Ассемблер: `ADDC A,<direct>`

Код:

0 0 1 1 0 1 0 1

direct address

Время: 1 цикл

Алгоритм: $(A) := (A) + (C) + (\text{direct})$
 $(C) := X, (AC) := X, (OV) := X$, где $X = (0 \text{ или } 1)$

Пример: ; (A)=11H, (03Y[80])=DFH, (C)=1
`ADDC A,80H ; (A)=F1H, (C)=0, (AC)=1, (OV)=0`

4) Ассемблер: `ADDC A,#data`

Код:

0 0 1 1 0 1 0 0

#data

Время: 1 цикл

Алгоритм: $(A) := (A) + (C) + (\text{direct})$
 $(C) := X, (AC) := X, (OV) := X$, где $X = (0 \text{ или } 1)$

Пример: ; (A)=55H, (C)=0
`ADDC A,#55H ; (A)=AAH, (C)=0, (AC)=0, (OV)=1`

Команда AJMP <addr11>

Команда "абсолютный переход", передает управление по указанному адресу, который получается при конкатенации пяти старших бит счетчика команд PC (после увеличения его на два), 7—5 битов кода операции и второго байта команды. Адрес перехода должен находиться внутри одной страницы объемом 2 Кбайт памяти программы, определяемой пятью старшими битами счетчика команд.

Ассемблер: `AJMP <метка>`

Код:

A10 A9 A8 0 0 0 0 1

A7 A6 A5 A4 A3 A2 A1 A0

Время: 2 цикла
 Алгоритм: $(PC[15-0]) := (PC[15-0]) + 2,$
 $(PC[10-0]) := \langle \text{addr11} \rangle$
 Пример: ; (PC)=028FH
 ; Метке MT2 соответствует адрес 034AH
 AJMP MT2 ; (PC)=034AH

Команда ANL <байт-назначения>, <байт-источник>

Команда "логическое "И" для переменных-байтов" выполняет операцию логического "И" над битами указанных переменных и помещает результат в байт-назначения. Эта операция не влияет на состояние флагов.

Два операнда обеспечивают следующие комбинации шести режимов адресации:

— байтом назначения является аккумулятор (A):

- 1) регистровый;
- 2) прямой;
- 3) косвенно-регистровый;
- 4) непосредственный;
- байтом назначения является прямой адрес (direct):
- 5) прямой аккумуляторный;
- 6) непосредственный (байт-источник равен константе).

- 1) Ассемблер: ANL A, Rn ; где n=0-7

Код:

0 1 0 1 1 rrr

 , где rrr=000-111

Время: 1 цикл
 Алгоритм: $(A) := (A) \text{ AND } (Rn)$
 Пример: ; (A)=FEH, (R2)=C5H
 ANL A, R2 ; (A)=C4H, (R2)=C5H

- 2) Ассемблер: ANL A, <direct>

Код:

0 1 0 1 0 1 0 1

direct address

Время: 1 цикл
 Алгоритм: $(A) := (A) \text{ AND } (\text{direct})$
 Пример: ; (A)=A3H, (PSW)=86H
 ANL A, PSW ; (A)=82H, (PSW)=86H

- 3) Ассемблер: ANL A, @Ri ; где i=0,1

Код:

0 1 0 1 0 1 1 i

 , где i=0,1

Время: 1 цикл
 Алгоритм: $(A) := (A) \text{ AND } ((Ri))$
 Пример: ; (A)=BCH, (03Y[35])=47H, (R0)=35H
 ANL A, @R0 ; (A)=04H, (03Y[35])=47H

- 4) Ассемблер: ANL A, #data

Код:

0 1 0 1 0 1 0 0

#data8

Время: 1 цикл
 Алгоритм: $(A) := (A) \text{ AND } \#data$

Пример: ; (A)=36H
 ANL A, #0DDH ; (A)=14H

5) Ассемблер: ANL <direct>, A

Код:

0 1 0 1 0 0 1 0

direct address

Время: 1 цикл

Алгоритм: (direct):=(direct) AND (A)

Пример: ; (A)=55H, (P2)=AAH
 ANL P2, A ; (P2)=00H, (A)=55H

6) Ассемблер: ANL <direct>, #data

Код:

0 1 0 1 0 0 1 1

direct address

#data8

Время: 2 цикла

Алгоритм: (direct):=(direct) AND #data

Пример: ; (P1)=FFH
 ANL P1, #73H ; (P1)=73H

Примечание. Если команда "ANL" применяется для изменения содержимого порта, то значение, используемое в качестве данных порта, будет считываться из "защелки" порта, а не с выводов БИС.

Команда ANL C, <бит источника>

Команда "логическое "И" для переменных-битов", выполняет операцию логического "И" над указанными битами. Если бит-источник равен "0", то происходит сброс флага переноса, в противном случае флаг переноса не изменяет текущего значения. "/" перед операндом в языке ассемблера указывает на то, что в качестве значения используется логическое отрицание адресуемого бита, однако сам бит источника при этом не изменяется. На другие флаги эта команда не влияет.

Для операнда-источника разрешена только прямая адресация к битам.

1) Ассемблер: ANL C, <bit>

Код:

1 0 0 0 0 0 1 0

bit address

Время: 2 цикла

Алгоритм: (C):=(C) AND (bit)

Пример: ; (C)=1, P1[0]=0
 ANL C, P1.0 ; (C)=0, P1[0]=0

2) Ассемблер: ANL C, </bit>

Код:

1 0 1 1 0 0 0 0

bit address

Время: 2 цикла

Алгоритм: (C):=(C) AND (/bit)

Пример: ; (C)=1, (AC)=0
 ANL C, /AC ; (C)=1, (AC)=0

Команда CJNE <байт назначения>, <байт источник>, <смещение>

Команда "сравнение и переход, если не равно" сравнивает значения первых двух операндов и выполняет ветвление, если операнды не равны. Адрес перехода

(ветвления) вычисляется при помощи сложения значения (со знаком), указанного в последнем байте команды, с содержимым счетчика команд после увеличения его на три.

Флаг переноса "С" устанавливается в "1", если значение целого без знака <байта назначения> меньше, чем значение целого без знака <байта источника>, в противном случае перенос сбрасывается (если значения операндов равны, флаг переноса сбрасывается). Эта команда не оказывает влияния на операнды.

Операнды, стоящие в команде, обеспечивают комбинации четырех режимов адресации:

— если байтом-назначения является аккумулятор:

- 1) прямой
- 2) непосредственный,

— если байтом-назначения является любая ячейка ОЗУ с косвенно-регистровой или регистровой адресацией:

- 3) непосредственный к регистровому
- 4) непосредственный к косвенно-регистровому

1) Ассемблер: CJNE A,<direct>,<метка>

Код:

1 0 1 1 0 1 0 1

direct address

rel8

Время: 2 цикла

Алгоритм: (PC):=(PC)+3,
если (direct)<(A) то (PC):=(PC)+<rel8>, C:=0
если (direct)>(A), то (PC):=(PC)+<rel8>, C:=1

Пример: ;(A)=97H, (P2)=F0H, (C)=0

CJNE A,P2,MT3

MT3: CLR A ;(A)=97H, (P2)=F0H, (C)=1
;Адрес, соответствующий метке
;MT3 вычисляется, как
;(PC):=(PC)+3+(rel8)

2) Ассемблер: CJNE A,#data,<метка>

Код:

1 0 1 1 0 1 0 0

#data8

rel8

Время: 2 цикла

Алгоритм: (PC):=(PC)+3,
если #data<(A), то (PC)+<rel8>, C:=0
если #data8>(A), то (PC):=(PC)+<rel8>, C:=1

Пример: ;(A)=FCH, (C)=1

CJNE A,#0BFH,MT4

MT4: INC A ;(A)=FDH, C=0,
;(PC):=(PC)+3+(rel8)

3) Ассемблер: CJNE Rn,#data,<метка> ; где n=0-7

Код:

1 0 1 1 1 rrr

#data8

rel8

Время: 2 цикла

Алгоритм: (PC):=(PC)+3,
если #data<(Rn), то (PC):=(PC)+<rel8>, C:=0
если #data8>(Rn), то (PC)+<rel8>, C:=1

Пример: ;(R7)=80H, (C)=0

CJNE R7,#81H,MT5

MT5: NOP

;(R7)=80H, (C)=1,
;(PC):=(PC)+3+(rel8)

4) Ассемблер: CJNE @Ri,#data,<метка> ; где i=0,1

Код:

1 0 1 1 0 1 1 i

#data8

rel8

Время: 2 цикла

Алгоритм: (PC):=(PC)+3,
если #data<((Ri)), то (PC)+<rel8>, C:=0
если #data8>((Ri)), то (PC)+<rel8>, C:=1

Пример:

;(R0)=41H, (C)=1, (03Y[41])=57H
CJNE @R0,#29H,MT6

MT6: DEC R0

;(03Y[41])=57H, (C)=0,
;(PC):=(PC)+3+(rel8)

Команда CLR A

Команда "сброс аккумулятора" сбрасывает (обнуляет) содержимое аккумулятора A. На флаги команда не влияет.

Ассемблер: CLR A

Код:

1 1 1 0 0 1 0 0

Время: 1 цикл

Алгоритм: (A):=0

Пример:

;(A)=6DH, (C)=0, (AC)=1
CLR A ;(A)=00H, (C)=0, (AC)=1

Команда CLR <bit>

Команда "сброс бита" сбрасывает указанный бит в нуль. Эта команда работает с флагом переноса "C" или любым битом с прямой адресацией.

1) Ассемблер: CLR C

Код:

1 1 0 0 0 0 1 1

Время: 1 цикл

Алгоритм: (C):=0

Пример:

;(C)=1
CLR C ;(C)=0

2) Ассемблер: CLR <bit>

Код:

1 1 0 0 0 0 1 0

bit address

Время: 1 цикл

Алгоритм: (bit):=0

Пример:

;(P1)=5EH (01011110B)
CLR P1.3 ;(P1)=56H (01010110B)

Команда CPL A

Команда "инверсия аккумулятора" каждый бит аккумулятора инвертирует (изменяет на противоположный). Биты, содержащие "единицы", после этой команды будут содержать "нули", и наоборот. На флаги эта операция не влияет.

Ассемблер: CPL A

Код:

1	1	1	1	0	1	0	0
---	---	---	---	---	---	---	---

Время: 1 цикл

Алгоритм: (A):=/(A)

Пример: ;(A)=65H (01100101B)
CPL A ;(A)=9AH (10011010B)

Команда CPL <bit>

Команда "инверсия бита" инвертирует (изменяет на противоположное значение) указанный бит. Бит, который был "единицей", изменяется в "нуль" и наоборот. Команда CPL может работать с флагом переноса или с любым прямо адресуемым битом. На другие флаги команда не влияет.

1) Ассемблер: CPL <bit>

Код:

1	0	1	1	0	0	1	0
---	---	---	---	---	---	---	---

bit address

Время: 1 цикл

Алгоритм: (bit):=/(bit)

Пример: ;(P1)=39H (00111001B)
CPL P1.1
CPL P1.3 ;(P1)=33H (00110011B)

2) Ассемблер: CPL C

Код:

1	0	1	1	0	0	1	1
---	---	---	---	---	---	---	---

Время: 1 цикл

Алгоритм: (C):=/(C)

Пример: ;(C)=0, (AC)=1, (OV)=0
CPL C ;(C)=1, (AC)=1, (OV)=0

Примечание: Если эта команда используется для изменения информации на выходе порта, значение, используемое как исходные данные, считывается из "защелки" порта, а не с выводов БИС.

Команда DA A

Команда "десятичная коррекция аккумулятора для сложения" упорядочивает 8-битовую величину в аккумуляторе после выполненной ранее команды сложения двух переменных (каждая в упакованном двоично-десятичном формате). Для выполнения сложения может использоваться любая из типов команд ADD или ADDC. Если значение битов 3—0 аккумулятора (A) превышает 9 (XXXX 1010—XXXX 1111) или, если флаг AC равен "1", то к содержимому (A) прибавляется 06, получая соответствующую двоично-десятичную цифру в младшем полубайте. Это внутреннее побитовое сложение устанавливает флаг переноса, если перенос из поля младших четырех бит распространяется через все старшие биты, а в противном случае — не изменяет флаг переноса. Если после этого флаг переноса равен "1", или если значение четырех старших бит (7—4) превышает 9 (1010 XXXX - 1111 XXXX), значения этих старших бит увеличивается на 6,

создавая соответствующую двоично-десятичную цифру в старшем полубайте. И снова при этом флаг переноса устанавливается, если перенос получается из старших битов, но не изменяется в противном случае. Таким образом, флаг переноса указывает на то, что сумма двух исходных двоично-десятичных переменных больше чем 100. Эта команда выполняет десятичное преобразование с помощью сложения 06, 60, 66 с содержимым аккумулятора в зависимости от начального состояния аккумулятора и слова состояния программы (PSW).

Ассемблер: DA A

Код: 1 1 0 1 0 1 0 0

Время: 1 цикл

Алгоритм: если $((A[3-0]) > 9$ или $(AC)=1$), то $A[3-0] := A[3-0] + 6$
если $((A[7-4]) > 9$ или $(C)=1$), то $A[7-4] := A[7-4] + 6$

Примеры: а) ;(A)=56H, (R3)=67H, (C)=1

ADDC A, R3

DA A ;(A)=24H, (R3)=67H, (C)=1

б) ;(A)=30H, (C)=0

ADD A, #99H

DA A ;(A)=29, (C)=1

Примечание: Команда DA A не может просто преобразовать шестнадцатичное значение в аккумуляторе в двоично-десятичное представление и не применяется, например, для десятичного вычитания.

Команда DEC <байт>

Команда "декремент" производит вычитание "1" из указанного операнда. Начальное значение 00H перейдет в 0FFH. Команда DEC не влияет на флаги. Этой командой допускается четыре режима адресации операнда:

- 1) к аккумулятору
- 2) регистровый
- 3) прямой
- 4) косвенно-регистровый

1) Ассемблер: DEC A

Код: 0 0 0 1 0 1 0 0

Время: 1 цикл

Алгоритм: $(A) := (A) - 1$

Пример: ;(A)=11H, (C)=1, (AC)=1

DEC A ;(A)=10H, (C)=1, (AC)=1

2) Ассемблер: DEC Rn ; где n=0-7

Код: 0 0 0 1 1 rrr , где rrr=000-111

Время: 1 цикл

Алгоритм: $(Rn) := (Rn) - 1$

Пример: ;(R1)=7FH,
;(O3Y[7F])=40H, (O3Y[7F])=00H

DEC @R1

DEC R1

DEC @R1 ;(R1)=7EH,
;(O3Y[7F])=3FH, (O3Y[7F])=FFH

3) Ассемблер: DEC <direct>

Код:

0 0 0 1 0 1 0 1

direct address

Время: 1 цикл

Алгоритм: (direct):=(direct)-1

Пример: ;(SCON)=A0H, (C)=1, (AC)=1
 DEC SCON ;(SCON)=9FH, (C)=1, (AC)=1

4) Ассемблер: DEC @Ri ; где i=0,1

Код:

0 0 0 1 0 1 1 i

Время: 1 цикл

Алгоритм: ((Ri)):=((Ri))-1

Пример: ;(R1)=7FH,
 ;(03Y[7F])=40H, (03Y[7F])=00H
 DEC @R1
 DEC R1
 DEC @R1 ;(R1)=7EH,
 ;(03Y[7F])=3FH, (03Y[7F])=FFH

Примечание: Если эта команда используется для изменения информации на выходе порта, значение, используемое как исходные данные, считывается из "защелки" порта, а не с выводов БИС.

Команда DIV AB

Команда "деление" делит 8-битовое целое без знака из аккумулятора А на 8-битовое целое без знака в регистре В. Аккумулятору присваивается целая часть частного (старшие разряды), а регистру В — остаток. Флаги переноса (C) и переполнения (OV) сбрасываются. Если (A) < (B), то флаг дополнительного переноса (AC) не сбрасывается. Флаг переноса сбрасывается в любом случае.

Ассемблер: DIV AB

Код:

1 0 0 0 0 1 0 0

Время: 4 цикла

Алгоритм: (A):=((A)/(B))[15-8],
 (B):=((A)/(B))[7-0]

Пример: Пусть аккумулятор содержит число 251 (0FBH или 11111011B), а регистр В — число 18 (12H или 00010010B). После выполнения команды DIV AB в аккумуляторе будет число 13 (0DH или 00001101B), а в регистре В — число 17 (11H или 00010001B), т.к. 251=(13*18)+17. Флаги C и OV будут сброшены.

Примечание. Если В содержит 00, то после команды DIV содержимое аккумулятора А и регистра В будут не определены. Флаг переноса сбрасывается, а флаг переполнения устанавливается в "1".

Команда DJNZ <байт>, <смещение>

Команда "декремент и переход, если не равно нулю" выполняет вычитание "1" из указанной ячейки и осуществляет ветвление по вычисляемому адресу, если

результат не равен нулю. Начальное значение $00H$ перейдет в $0FFH$. Адрес перехода (ветвления) вычисляется сложением значения смещения (со знаком), указанного в последнем байте команды, с содержимым счетчика команд, увеличенным на длину команды DJNZ. На флаги эта команда не влияет и допускает следующие режимы адресации:

1) регистровый

2) прямой

1) Ассемблер: DJNZ Rn, <метка> ; где $n=0-7$

Код:

1 1 0 1 1 rrr

rel8

Время: 2 цикла

Алгоритм: $(PC) := (PC) + 2$,
 $(Rn) := (Rn) - 1$,
 если $((Rn) > 0$ или $(Rn) < 0)$, то $(PC) := (PC) + \langle rel8 \rangle$

Пример: ; (R2)=08H, (P1)=FFH (11111111B)

LAB4: CPL P1.7

DJNZ R2, LAB4 ; (R2)={07-00}

; Эта последовательность команд переключает P1.7

; восемь раз и приводит к появлению четырех импульсов

; на выводе БИС, соответствующем биту P1.7.

2) Ассемблер: DJNZ <direct>, <метка>

Код:

1 1 0 1 0 1 0 1

direct address

rel8

Время: 2 цикла

Алгоритм: $(PC) := (PC) + 3$,
 $(direct) := (direct) - 1$,
 если $((direct) > 0$ или $(direct) < 0)$, то
 $(PC) := (PC) + \langle rel8 \rangle$

Пример: ; (03Y[40])=01H, (03Y[50])=80H,

; (03Y[60])=25H

DJNZ 40H, LAB1 ; (03Y[40]) := 00H

DJNZ 50H, LAB2 ; (03Y[50]) := 7FH

DJNZ 60H, LAB3 ; (03Y[60]) := 25H

LAB1: CLR A

LAB2: DEC R1 ; осуществился переход на
 ; метку LAB2

Примечание: Если команда DJNZ используется для изменения выхода порта, значение, используемое как операнд, считывается из "защелки" порта, а не с выводов БИС.

Команда INC <байт>

Команда "инкремент" выполняет прибавление "1" к указанной переменной и не влияет на флаги. Начальное значение $0FFH$ перейдет в $00H$. Эта команда допускает четыре режима адресации:

1) к аккумулятору

2) регистровый

3) прямой

4) косвенно-регистровый

1) Ассемблер: INC A

Код: 0 0 0 0 0 1 0 0

Время: 1 цикл

Алгоритм: $(A) := (A) + 1$

Пример: ; (A)=1FH, (AC)=0
INC A ; (A)=20H, (AC)=0

2) Ассемблер: INC Rn ; где n=0-7

Код: 0 0 0 0 1 rrr , где rrr=000-111

Время: 1 цикл

Алгоритм: $(Rn) := (Rn) + 1$

Пример: ; (R4)=FFH, (C)=0, (AC)=0
INC R4 ; (R4)=00H, (C)=0, (AC)=0

3) Ассемблер: INC <direct>

Код: 0 0 0 0 0 1 0 1 direct address

Время: 1 цикл

Алгоритм: $(direct) := (direct) + 1$

Пример: ; (O3Y[43])=22H
INC 43H ; (O3Y[43])=23H

4) Ассемблер: INC @Ri ; где i=0,1

Код: 0 0 0 0 0 1 1 i , где i=0,1

Время: 1 цикл

Алгоритм: $((Ri)) := ((Ri)) + 1$

Пример: ; (R1)=41H, (O3Y[41])=4FH, (AC)=0
INC @R1 ; (R1)=41H, (O3Y[41])=50H, (AC)=0

Примечание. При использовании команды INC для изменения содержимого порта, величина, используемая как операнд, считывается из "защелки" порта, а не с выводов БИС.

Команда INC DPTR

Команда "инкремент указателя данных" выполняет инкремент (прибавление "1") содержимого 16-битового указателя данных (DPTR). Прибавление "1" осуществляется к 16 битам, причем переполнение младшего байта указателя данных (DPL) из FFH в 00H приводит к инкременту старшего байта указателя данных (DPH). На флаги эта команда не влияет.

Ассемблер: INC DPTR

Код: 1 0 1 0 0 0 1 1

Время: 2 цикла

Алгоритм: $(DPTR) := (DPTR) + 1$

Пример: ;(DPH)=12H, (DPL)=FEN
 INC DPTR
 INC DPTR
 INC DPTR ;(DPH)=13H, (DPL)=01H

Команда JB <bit>, <rel8>

Команда "переход, если бит установлен" выполняет переход по адресу ветвления, если указанный бит равен "1", в противном случае выполняется следующая команда. Адрес ветвления вычисляется с помощью прибавления относительного смещения со знаком в третьем байте команды (rel8) к содержимому счетчика команд после прибавления к нему 3. Проверяемый бит не изменяется. Эта команда на флаги не влияет.

Ассемблер: JB (bit), <метка>

Код:	0 0 1 0 0 0 0 0	bit address	rel8
------	-----------------	-------------	------

Время: 2 цикла

Алгоритм: (PC):=(PC)+3,
 если (bit)=1, то (PC):=(PC)+<rel8>

Пример: ;(A)=96H (10010110B)
 JB ACC.2, LAB5 ;эта команда обеспечивает переход
 ;на метку LAB5

LAB5: INC A

Команда JBC <bit>, <rel8>

Команда "переход, если бит установлен и сброс этого бита", выполняет ветвление по вычисляемому адресу, если бит равен "1". В противном случае выполняется следующая за JBC команда. В любом случае указанный бит сбрасывается. Адрес перехода вычисляется сложением относительного смещения со знаком в третьем байте команды (rel8) и содержимого счетчика команд после прибавления к нему 3. Эта команда не влияет на флаги.

Ассемблер: JBC (bit), <метка>

Код:	0 0 0 1 0 0 0 0	bit address	rel8
------	-----------------	-------------	------

Время: 2 цикла

Алгоритм: (PC):=(PC)+3,
 если (bit)=1, то (bit):=0, (PC):=(PC)+<rel8>

Пример: (A)=76H (0111 0110B)
 JBC ACC.3, LAB6 ; Перехода на LAB6 нет, т.к.
 ; (A[3]) = 0
 JBC ACC.2, LAB7 ; (A)=72H (0111 0010B) и переход
 ; на адрес, соответствующий
 ; метке LAB7.

Примечание. Если эта команда используется для проверки бит порта, то значение, используемое как операнд, считывается из "защелки" порта, а не с вывода БИС.

Команда JC <rel8>

Команда "переход, если перенос установлен" выполняет ветвление по адресу, если флаг переноса равен "1", в противном случае выполняется следующая команда. Адрес ветвления вычисляется с помощью сложения относительного

смещения со знаком во втором байте команды (rel8) и содержимого счетчика команд, после прибавления к нему 2. Эта команда на флаги не влияет.

Ассемблер: JC <метка>

Код:

0 1 0 0 0 0 0 0

rel8

Время: 2 цикла

Алгоритм: (PC):=(PC)+2,
если (C)=1, то (PC):=(PC)+<rel8>

Пример: ; (C)=0
JC LAB8 ; нет перехода на метку LAB8
CPL C ; (C):=1
LAB8: JC LAB9 ; переход на метку LAB9, т.к. (C)=1
LAB9: NOP

Команда JMP @A+DPTR

Команда "косвенный переход" складывает 8-битовое содержимое аккумулятора без знака с 16-битовым указателем данных (DPTR) и загружает полученный результат в счетчик команд, содержимое которого является адресом для выборки следующей команды. 16-битовое сложение выполняется по модулю 2^{16} , перенос из младших восьми бит распространяется на старшие биты программного счетчика. Содержимое аккумулятора и указателя данных не изменяется. Эта команда на флаги не влияет.

Ассемблер: JMP @A+DPTR

Код:

0 1 1 1 0 0 1 1

Время: 2 цикла

Алгоритм: (PC):=(A)[7-0]+(DPTR)[15-0]

Пример: ; (PC)=034EH, (A)=86H, (DPTR)=0329H
JMP @A+DPTR ; (PC)=03AFH, (A)=86H, (DPTR)=0329H

Команда JNB <bit>,<rel8>

Команда "переход, если бит не установлен" выполняет ветвление по адресу, если указанный бит равен "нулю", в противном случае выполняется следующая команда. Адрес ветвления вычисляется с помощью сложения относительного смещения со знаком в третьем байте команды (rel8) и содержимого счетчика команд после прибавления к нему 3. Проверяемый бит не изменяется. Эта команда на флаги не влияет.

Ассемблер: JNB (bit),<метка>

Код:

0 0 1 1 0 0 0 0

bit address

rel8

Время: 2 цикла

Алгоритм: (PC):=(PC)+3,
если (bit)=0, то (PC):=(PC)+<rel8>

Пример: ; (P2)=CAH (11001010B),
 ; (A)=56H (0101 0110B)
 JNB P1.3, LAB10 ; нет перехода на LAB10
 JNB ACC.3, LAB11 ; переход на метку LAB11

LAB11: INC A

Команда JNC <rel8>

Команда "переход, если перенос не установлен" выполняет ветвление по адресу, если флаг переноса равен нулю, в противном случае выполняется следующая команда. Адрес ветвления вычисляется с помощью сложения относительного смещения со знаком во втором байте команды (rel8) и содержимого счетчика команд после прибавления к нему 2. Флаг переноса не изменяется. Эта команда на другие флаги не влияет.

Ассемблер: JNC <метка>

Код:

0 1 0 1 0 0 0 0

rel8

Время: 2 цикла

Алгоритм: (PC):=(PC)+2,
 если (C)=0, то (PC):=(PC)+<rel8>

Пример: ; (C)=1
 JNC LAB12 ; нет перехода на LAB12
 CPL C
 LAB12: JNC LAB13 ; переход на метку LAB13

Команда JNZ <rel8>

Команда "переход, если содержимое аккумулятора не равно нулю" выполняет ветвление по адресу, если хотя бы один бит аккумулятора равен "1", в противном случае выполняется следующая команда. Адрес ветвления вычисляется сложением относительного смещения со знаком во втором байте команды (rel8) и содержимого счетчика команд (PC) после прибавления к нему 2. Содержимое аккумулятора не изменяется. Эта команда на флаги не влияет.

Ассемблер: JNZ <метка>

Код:

0 1 1 1 0 0 0 0

rel8

Время: 2 цикла

Алгоритм: (PC):=(PC)+2,
 если (A)≠0 то (PC):=(PC)+<rel8>

Пример: ; (A)=00H
 JNC LAB14 ; нет перехода на LAB14
 INC A
 LAB14: JNZ LAB15 ; переход на метку LAB15
 LAB15: NOP

Команда JZ <rel8>

Команда "переход, если содержимое аккумулятора равно "0" выполняет ветвление по адресу, если все биты аккумулятора равны "0", в противном случае выполняется следующая команда. Адрес ветвления вычисляется сложением относительного смещения со знаком во втором байте команды (rel8) и

содержимым счетчика команд после прибавления к нему 2. Содержимое аккумулятора не изменяется. Эта команда на флаги не влияет.

Ассемблер: JZ <метка>

Код:

0 1 1 0 0 0 0 0

rel8

Время: 2 цикла

Алгоритм: (PC):=(PC)+2,
если (A)=0, то (PC):=(PC)+<rel8>

Пример: ;(A)=01H

JZ LAB16 ;нет перехода на LAB16

DEC A

LAB16: JZ LAB17 ;переход на метку LAB17

...

LAB17: CLR A

Команда LCALL <addr16>

Команда "длинный вызов" вызывает подпрограмму, находящуюся по указанному адресу. По команде LCALL к счетчику команд (PC) прибавляется 3 для получения адреса следующей команды и после этого полученный 16-битовый результат помещается в СТЕК (сначала следует младший байт, за ним — старший), а содержимое указателя СТЕКа (SP) увеличивается на 2. Затем старший и младший байты счетчика команд загружаются соответственно вторым и третьим байтами команды LCALL. Выполнение программы продолжается командой, находящейся по полученному адресу. Подпрограмма, следовательно, может начинаться в любом месте адресного пространства памяти программ объемом до 64 Кбайт. Эта команда на флаги не влияет.

Ассемблер: LCALL <метка>

Код:

0 0 0 1 0 0 1 0

addr[15-8]

addr[7-0]

Время: 2 цикла

Алгоритм: (PC):=(PC)+3
(SP):=(SP)+1
((SP)):=<PC[7-0]>
(SP):=(SP)+1
((SP)):=<PC[15-8]>
(PC):=<addr[15-0]>

Пример: ;(SP)=07H,
;метке PRN соответствует адрес 1234H,
;по адресу 0126H находится команда
;LCALL
LCALL PRN ;(SP)=09H, (PC)=1234H,
;(03Y[08])=26H, (03Y[09])=01H

Команда LJMP <addr16>

Команда "длинный переход" выполняет безусловный переход по указанному адресу, загружая старший и младший байты счетчика команд (PC) соответственно вторым и третьим байтами, находящимися в коде команды. Адрес перехода, таким образом, может находиться по любому адресу пространства памяти программ в 64 Кбайт. Эта команда на флаги не влияет.

Ассемблер: LJMP <метка>

Код:

0 0 0 0 0 0 1 0

addr[15-8]

addr[7-0]

Время: 2 цикла

Алгоритм: (PC):=<addr[15-0]>

Команда MOV <байт-назначения>, <байт-источника>

Команда "переслать переменную-байт" пересылает переменную-байт, указанную во втором операнде, в ячейку, указанную в первом операнде. Содержимое байта источника не изменяется. Эта команда на флаги и другие регистры не влияет. Команда "MOV" допускает 15 комбинаций адресации байта-источника и байта-назначения.

1) Ассемблер: MOV A,Rn ; где n=0-7

Код:

1 1 1 0 1 rrr

, где rrr=000-111

Время: 1 цикл

Алгоритм: (A):=(Rn)

Пример:

; (A)=F0H, (R4)=93H
MOV A,R4 ; (A)=93H, (R4)=93H

2) Ассемблер: MOV A,<direct>

Код:

1 1 1 0 0 1 0 1

direct address

Время: 1 цикл

Алгоритм: (A):=(direct)

Пример:

; (A)=93H, (03Y[40])=10H, (R0)=40H
MOV A,40H ; (A)=10H, (03Y[40])=10H, (R0)=40H

3) Ассемблер: MOV A,@Ri ; где i=0,1

Код:

1 1 1 0 0 1 1 i

Время: 1 цикл

Алгоритм: (A):=((Ri))

Пример:

; (A)=10H, (R0)=41H, (03Y[41])=0CAH
MOV A,@R0 ; (A)=CAH, (R0)=41H, (03Y[41])=0CAH

4) Ассемблер: MOV A,#data

Код:

0 1 1 1 0 1 0 0

#data8

Время: 1 цикл

Алгоритм: (A):=<#data8>

Пример:

; (A)=C9H (11001001B)
MOV A,#37H ; (A)=37H (00110111B)

5) Ассемблер: MOV Rn,A ; где n=0-7

Код:

1 1 1 1 1 rrr

, где rrr=000-111

Время: 1 цикл

Алгоритм: (Rn):=(A)

- Пример: ; (A)=38H, (R0)=42H
MOV R0,A ; (A)=38H, (R0)=38H
- 6) Ассемблер: MOV Rn,<direct> ; где n=0-7
- Код:

1 0 1 0 1 rrr

direct address

- Время: 2 цикла
Алгоритм: (Rn):=(direct)
Пример: ; (R0)=39H, (P2)=0F2H
MOV R0,P2 ; (R0)=F2H
- 7) Ассемблер: MOV Rn,#data ; где n=0-7
- Код:

0 1 1 1 1 rrr

#data8

- Время: 1 цикл
Алгоритм: (Rn):=<#data8>
Пример: ; (R0)=0F5H
MOV R0,#49H ; (R0)=49H
- 8) Ассемблер: MOV <direct>,A
- Код:

1 1 1 1 0 1 0 1

direct address

- Время: 1 цикл
Алгоритм: (direct):=(A)
Пример: ; (P0)=FFH, (A)=4BH
MOV P0,A ; (P0)=4BH, (A)=4BH
- 9) Ассемблер: MOV <direct>,Rn ; где n=0-7
- Код:

1 0 0 0 1 rrr

direct address

,
где rrr=000-111
- Время: 2 цикла
Алгоритм: (direct):=(Rn)
Пример: ; (PSW)=C2H, (R7)=57H
MOV PSW,R7 ; (PSW)=57H, (R7)=57H
- 10) Ассемблер: MOV <direct>,<direct>
- Код:

1 0 0 0 0 1 0 1

direct address

direct address

- Время: 2 цикла
Алгоритм: (direct):=(direct)
Пример: ; (03Y[45])=33H, (03Y[48])=0DEH
MOV 48H,45H ; (03Y[45])=33H, (03Y[48])=33H
- 11) Ассемблер: MOV <direct>,@Ri ; где i=0,1
- Код:

1 0 0 0 0 1 1 i

direct address

- Время: 2 цикла
Алгоритм: (direct):=((Ri))
Пример: ; (R1)=49H, (03Y[49])=0E3H
MOV 51H,@R1 ; (03Y[51])=0E3H, (03Y[49])=0E3H
- 12) Ассемблер: MOV <direct>,#data

- Код:

0 1 1 1 0 1 0 1

direct address

#data8

- Время: 2 цикла
 Алгоритм: (direct):=<#data8>
 Пример: ;(03Y[5F])=9BH
 MOV 5FH,#07H ;(03Y[5F])=07H
- 13) Ассемблер: MOV @Ri,A ; где i=0,1
- Код:

1 1 1 1 0 1 1 i

 , где i=0,1
- Время: 1 цикл
 Алгоритм: ((Ri)):= (A)
 Пример: ;(R1)=48H, (03Y[48])=75H, (A)=0BDH
 MOV @R1,A ;(03Y[48])=0BDH
- 14) Ассемблер: MOV @Ri,<direct> ; где i=0,1
- Код:

1 0 1 0 0 1 1 i

direct address

- Время: 2 цикла
 Алгоритм: ((Ri)):= (direct)
 Пример: ;(R0)=51H, (03Y[51])=0E3H, (P0)=0ACH
 MOV @R0,P0 ;(03Y[51])=0ACH
- 15) Ассемблер: MOV @Ri,#data ; где i=0,1
- Код:

0 1 1 1 0 1 1 i

- Время: 1 цикл
 Алгоритм: ((Ri)):=<#data8>
 Пример: ;(03Y[7E])=67H, (R1)=7EH
 MOV @R1, #0A9H ;(03Y[7E])=0A9H, (R1)=7EH

Команда MOV <бит назначения>, <бит источника>

Команда "переслать бит данных" битовую переменную, указанную во втором байте, копирует в разряд, который указан в первом операнде. Одним из операндов должен быть флаг переноса C, а другим может быть любой бит, к которому возможна прямая адресация.

- 1) Ассемблер: MOV C,<bit>

Код:

1 0 1 0 0 0 1 0

bit address

Время: 1 цикл
 Алгоритм: (C):=(bit)
 Пример: ;(C)=0, (P3)=D5H (11010101B)
 MOV C,P3.0 ;C:=1
 MOV C,P3.3 ;C:=0
 MOV C,P3.7 ;C:=1

- 2) Ассемблер: MOV <bit>,C

Код:

1 0 0 1 0 0 1 0

bit address

Время: 2 цикла
 Алгоритм: (bit):=(C)

Пример: ; (C)=1, (P0)=20H (00100000B)
 MOV P0.1,C
 MOV P0.2,C
 MOV P0.3,C ; (C)=1, (P0)=2EH (00101110B)

Команда MOV DPTR,#data16

Команда "загрузить указатель данных 16-битовой константой" загружает указатель данных DPTR 16-битовой константой, указанной во втором и третьем байтах команды. Второй байт команды загружается в старший байт указателя данных (DPH), а третий байт — в младший байт указателя данных (DPL). Эта команда на флаги не влияет и является единственной командой, которая одновременно загружает 16 бит данных.

Ассемблер: MOV DPTR,#<data16>

Код:

1 0 0 1 0 0 0 0

#data[15-8]

#data[7-0]

Время: 2 цикла

Алгоритм: (DPTR):=#data[15-0],
 причем DPH:=#data[15-8], DPL:=#data[7-0]

Пример: ; (DPTR)=01FDH
 MOV DPTR,#1234H ; (DPTR)=1234H,
 ; (DPH)=12H, (DPL)=34H

Команда MOVC A,@A+(<R16>)

<R16> — 16-разрядный регистр.

Команда "переслать байт из памяти программ" загружает аккумулятор байтом кода или константой из памяти программы. Адрес считываемого байта вычисляется как сумма 8-битового исходного содержимого аккумулятора без знака и содержимого 16-битового регистра. В качестве 16-битового регистра может быть:

- 1) указатель данных DPTR;
- 2) счетчик команд PC.

В случае, когда используется PC, он увеличивается до адреса следующей команды перед тем, как его содержимое складывается с содержимым аккумулятора. 16-битовое сложение выполняется так, что перенос из младших восьми бит может распространяться через старшие биты. Эта команда на флаги не влияет.

1) Ассемблер: MOVC A,@A+DPTR

Код:

1 0 0 1 0 0 1 1

Время: 2 цикла

Алгоритм: (A):=((A)+(DPTR))

Пример: ; (A)=1BH, (DPTR)=1020H,
 ; (ПЗУ[103B])=48H,
 MOVC A,@A+DPTR ; (A)=48H, (DPTR)=1020H

2) Ассемблер: MOVC A,@A+PC

Код:

1 0 0 0 0 0 1 1

Время: 2 цикла

Алгоритм: (A):=((A)+(PC))

Пример: ; (A)=FAH, (PC)=0289,
 ; (ПЗУ[0384])=9BH
 MOVC A,@A+PC ; (A)=9BH, (PC)=028AH

Команда MOVX <байт приемника>, <байт источника>

Команда "переслать во внешнюю память (из внешней памяти) данных" пересылает данные между аккумулятором и байтом внешней памяти данных. Имеется два типа команд, которые отличаются тем, что обеспечивают 8-битовый или 16-битовый косвенный адрес к внешнему ОЗУ данных.

В первом случае содержимое R0 или R1 в текущем банке регистров обеспечивает 8-битовый адрес, который мультиплексируется с данными порта P0. Для расширения дешифрации ввода-вывода или адресации небольшого массива ОЗУ достаточно восьми бит адресации. Если применяются ОЗУ, немного больше чем 256 байт, то для фиксации старших битов адреса можно использовать любые другие выходы портов, которые переключаются командой, стоящей перед командой MOVX.

Во втором случае, при выполнении команды MOVX указатель данных DPTR генерирует 16-битовый адрес. Порт P2 выводит старшие восемь бит адреса (DPH), а порт P0 мультиплексирует младшие 8 бит адреса (DPL) с данными. Эта форма является эффективной при доступе к большим массивам данных (до 64К байт), так как для установки портов вывода не требуется дополнительных команд.

1) Ассемблер: MOVX A,@Ri ; где i=0,1

Код:

1	1	1	0	0	0	1	i
---	---	---	---	---	---	---	---

Время: 2 цикла

Алгоритм: (A):=((Ri))

Пример: ;(A)=32H, (R0)=83H, ячейка
;внешнего ОЗУ по адресу 83H
;содержит B6H
MOVX A,@R0 ;(A)=B6H, (R0)=83H

2) Ассемблер: MOVX A,@DPTR

Код:

1	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

Время: 2 цикла

Алгоритм: (A):=((DPTR))

Пример: ;(A)=5CH, (DPTR)=1ABEH,
;ячейка внешнего ОЗУ по адресу
;1ABEH содержит 72H
MOVX A,@DPTR ;(A)=72H, (DPTR)=2ABEH

3) Ассемблер: MOVX @Ri,A ; где i=0,1

Код:

1	1	1	1	0	0	1	i
---	---	---	---	---	---	---	---

Время: 2 цикла

Алгоритм: ((Ri)):= (A)

Пример: ;(A)=95H, (R1)=FDH,
;ячейка внешнего ОЗУ с адресом
;FDH содержит 00
MOVX @R1,A ;(A)=95H, (R1)=FDH,
;ячейка внешнего ОЗУ с адресом
;FDH содержит 95H

4) Ассемблер: MOVX @DPTR,A

Код:

1111	0000
------	------

Время: 2 цикла

Алгоритм: $((DPTR)) := (A)$

Пример: ; (A)=97H, (DPTR)=1FFFH,
; ячейка внешнего ОЗУ с адресом
; 1FFFH содержит 00
MOVX @DPTR, A ; (A)=97H,
; ячейка внешнего ОЗУ с адресом
; 1FFFH содержит 97H

Команда MUL AB

Команда "умножение" умножает 8-битовые целые числа без знака из аккумулятора и регистра В. Старший байт 16-битового произведения помещается в регистр В, а младший — в аккумулятор А. Если результат произведения больше, чем 0FFH(255), то устанавливается флаг переполнения (OV), в противном случае он сбрасывается. Флаг переноса всегда сбрасывается.

Ассемблер: MUL AB

Код:

1	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---

Время: 4 цикла

Алгоритм: $(A)[7-0] = (A) * (B)$,
 $(B)[15-8] = (A) * (B)$

Пример: а) ; (A)=50H (50H=80 DEC), (C)=1,
; (B)=0A0H (A0H=160 DEC), (OV)=0
MUL AB ; (A)=00H, (B)=32H, (C)=0, (OV)=1
б) ; (A)=2HH, (OV)=1, (B)=06H, (C)=1
MUL AB ; (A)=0D8H, (B)=00H, (OV)=0, (C)=0

Команда NOP

Команда "нет операции" выполняет холостой ход и не влияет на регистры и флаги, кроме как на счетчик команд (PC).

Ассемблер: NOP

Код:

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

Время: 1 цикл

Алгоритм: $(PC) := (PC) + 1$

Пример: Пусть требуется создать отрицательный выходной импульс на порте P1[6] длительностью 3 цикла. Это выполнит следующая последовательность команд:

```
CLR P1.6 ; P1[6] := 0
NOP
NOP
NOP
SETB P1.6 ; P1[6] := 1
```

Команда ORL <байт назначения>, <байт источника>

Команда "логическое "ИЛИ" для переменных-байтов" выполняет операцию логического "ИЛИ" над битами указанных переменных, записывая результат в байт назначения. Эта команда на флаги не влияет. Допускается шесть комбинаций режимов адресации:

— если байтом назначения является аккумулятор:

1) регистровый

- 2) прямой
- 3) косвенно-регистровый
- 4) непосредственный
- если байтом назначения является прямой адрес:
- 5) к аккумулятору
- 6) к константе

1) Ассемблер: ORL A,Rn ; где n=0-7

Код:

0 1 0 0 1 rrr

 , где rrr=000-111

Время: 1 цикл

Алгоритм: (A):=(A) OR (Rn),
где OR – операция логического "ИЛИ"

Пример: ;(A)=15H, (R5)=6CH
ORL A,R5 ;(A)=7DH, (R5)=6CH

2) Ассемблер: ORL A,<direct>

Код:

0 1 0 0 0 1 0 1

direct address

Время: 1 цикл

Алгоритм: (A):=(A) OR (direct)

Пример: ;(A)=84H, (PSW)=C2H
ORL A,PSW ;(A)=C6H, (PSW)=C2H

3) Ассемблер: ORL A,@Ri ; где i=0,1

Код:

0 1 0 0 0 1 1 i

Время: 1 цикл

Алгоритм: (A):=(A) OR ((Ri))

Пример: ;(A)=52H, (R0)=6DH, (03Y[6D])=49H
ORL A,@R0 ;(A)=5BH, (03Y[6D])=49H

4) Ассемблер: ORL A,#<data>

Код:

0 1 0 0 0 1 0 0

#data8

Время: 1 цикл

Алгоритм: (A):=(A) OR #<data>

Пример: ;(A)=F0H
ORL A,#0AH ;(A)=FAH

5) Ассемблер: ORL (direct),A

Код:

0 1 0 0 0 0 1 0

direct address

Время: 1 цикл

Алгоритм: (direct):=(direct) OR (A)

Пример: ;(A)=34H, (IP)=23H
ORL IP,A ;(IP)=37H, (A)=34H

6) Ассемблер: ORL (direct),#<data>

Код:	<div>0 1 0 0 0 0 1 1</div>	direct address	#data8
Время:	2 цикла		
Алгоритм:	(direct):=(direct) OR #<data>		
Пример:	<div>;(P1)=00H</div> <div>ORL P1,#0C4H ;(P1)=11000100B (C4H)</div>		

Примечание. Если команда используется для работы с портом, величина, используемая в качестве исходных данных порта, считывается из "защелки" порта, а не с выводов БИС.

Команда ORL C, <бит источника>

Команда "логическое "ИЛИ" для переменных-битов" устанавливает флаг переноса C, если булева величина равна логической "1", в противном случае устанавливает флаг C в "0". Косая дробь ("/") перед операндом на языке ассемблера указывает на то, что в качестве операнда используется логическое отрицание значения адресуемого бита, но сам бит источника не изменяется. Эта команда на другие флаги не влияет.

1) Ассемблер: ORL C, <bit>

Код:	<div>0 1 1 1 0 0 1 0</div>	bit address
Время:	2 цикла	
Алгоритм:	(C):=(C) OR (bit)	
Пример:	<div>;(C)=0, (P1)=53H (01010011B)</div> <div>ORL C, P1.4 ;(C)=1, (P1)=53H (01010011B)</div>	

2) Ассемблер: ORL C, /<bit>

Код:	<div>1 0 1 0 0 0 0 0</div>	bit address
Время:	2 цикла	
Алгоритм:	(C):=(C) OR /(bit)	
Пример:	<div>;(C)=0, (03Y[25])=39H (00111001B)</div> <div>ORL C, /2A ;(C)=1, (03Y[25])=39H (00111001B)</div>	

Команда POP <direct>

Команда "чтение из стека" считывает содержимое ячейки, которая адресуется с помощью указателя стека, в прямо адресуемую ячейку ОЗУ, при этом указатель стека уменьшается на единицу.

Эта команда не воздействует на флаги и часто используется для чтения из стека промежуточных данных.

Ассемблер: POP <direct>

Код:	<div>1 1 0 1 0 0 0 0</div>	direct address
Время:	2 цикла	
Алгоритм:	<div>(direct):=((SP)),</div> <div>(SP):=(SP)-1</div>	

Пример: ; (SP)=32H, (DPH)=01, (DPL)=ABH,
 ; (03Y[32])=12H, (03Y[31])=56H,
 ; (03Y[30])=20H
 POP DPH
 POP DPL ; (SP)=30H, (DPH)=12H, (DPL)=56H,
 ; (03Y[32])=12H, (03Y[31])=56H
 POP SP ; (SP)=20H, (03Y[30])=20H

Команда PUSH <direct>

Команда "запись в стек" увеличивает указатель стека на единицу и после этого содержимое указанной прямо адресуемой переменной копируется в ячейку внутреннего ОЗУ, адресуемого с помощью указателя стека. На флаги эта команда не влияет и используется для записи промежуточных данных в стек.

Ассемблер: PUSH <direct>

Код:

1 1 0 0 0 0 0 0

direct address

Время: 2 цикла

Алгоритм: (SP):=(SP)+1,
 ((SP)):=(<direct>)

Пример: ; (SP)=09H, (DPTR)=1279H
 PUSH DPL
 PUSH DPH ; (SP)=0BH, (DPTR)=1279H,
 ; (03Y[0A])=79H, (03Y[0B])=12H,

Команда RET

Команда "возврат из подпрограммы" последовательно выгружает старший и младший байты счетчика команд из стека, уменьшая указатель стека на 2. Выполнение основной программы обычно продолжается по адресу команды, следующей за ACALL или LCALL. На флаги эта команда не влияет.

Ассемблер: RET

Код:

0 0 1 0 0 0 1 0

Время: 2 цикла

Алгоритм: (PC)[15-8]:=((SP)),
 (SP):=(SP)-1,
 (PC)[7-0]:=((SP)),
 (SP):=(SP)-1

Пример: ; (SP)=0DH, (03Y[0C])=93H, (03Y[0D])=02H
 RET ; (SP)=0BH, (PC)=0293H

Команда RETI

Команда "возврат из прерывания" выгружает старший и младший байты счетчика команд из стека и устанавливает "логику прерываний", разрешая прием других прерываний с уровнем приоритета, равным уровню приоритета только что обработанного прерывания. Указатель стека уменьшается на 2. Слово состояния программы (PSW) не восстанавливается автоматически. Выполнение основной программы продолжается с команды, следующей за командой, на которой произошел переход к обнаружению запроса на прерывание. Если при выполнении команды RETI обнаружено прерывание с таким же или меньшим уровнем

приоритета, то одна команда основной программы успевает выполниться до обработки такого прерывания.

Ассемблер: RETI

Код:

0	0	1	1	0	0	1	0
---	---	---	---	---	---	---	---

Время: 2 цикла

Алгоритм: $(PC)[15-8] := ((SP))$,
 $(SP) := (SP) - 1$,
 $(PC)[7-0] := ((SP))$,
 $(SP) := (SP) - 1$

Пример: ; $(SP) = 0BH$, $(O3Y[0A]) = 2AH$, $(O3Y[0B]) = 03H$,
 ; $(PC) = YUYUYN$, где $Y = 0-FH$
 RETI ; $(SP) = 09H$, $(PC) = 032AH$

Команда RL A

Команда "сдвиг содержимого аккумулятора влево", сдвигает восемь бит аккумулятора на один бит влево, бит 7 засылается на место бита 0. На флаги эта команда не влияет.

Ассемблер: RL A

Код:

0	0	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Время: 1 цикл

Алгоритм: $(A[N+1]) := (A[N])$, где $N = 0-6$
 $(A[0]) := (A[7])$

Пример: ; $(A) = 0D5H$ (11010101B), $(C) = 0$
 RL A ; $(A) = 0ABH$ (10101011B), $(C) = 0$

Команда RLC A

Команда "сдвиг содержимого аккумулятора влево через флаг переноса" сдвигает восемь бит аккумулятора и флаг переноса влево на один бит. Содержимое флага переноса помещается на место бита 0 аккумулятора, а содержимое бита 7 аккумулятора переписывается в флаг переноса. На другие флаги эта команда не влияет.

Ассемблер: RLC A

Код:

0	0	1	1	0	0	1	1
---	---	---	---	---	---	---	---

Время: 1 цикл

Алгоритм: $(A[N+1]) := (A[N])$, где $N = 0-6$
 $(A[0]) := (C)$
 $(C) := (A[7])$

Пример: ; $(A) = 56H$ (01010110B), $(C) = 1$
 RLC A ; $(A) = 0ADH$ (10101101B), $(C) = 0$

Команда RR A

Команда "сдвиг содержимого аккумулятора вправо" сдвигает вправо на один бит все восемь бит аккумулятора. Содержимое бита 0 помещается на место бита 7. На флаги эта команда не влияет.

Ассемблер: RR A

Код:

0 0 0 0 0 0 1 1

Время: 1 цикл

Алгоритм: $(A[N]) := (A[N+1])$, где $N=0-6$
 $(A[7]) := (A[0])$ Пример: ;(A)=0D6H (11010110B), (C)=1
RR A ;(A)=6BH (01101011B), (C)=1**Команда RRC A**

Команда "сдвиг содержимого аккумулятора вправо через флаг переноса" сдвигает восемь бит аккумулятора и флаг переноса на один бит вправо. Бит 0 перемещается в флаг переноса, а исходное содержимое флага переноса помещается в бит 7. На другие флаги эта команда не влияет.

Ассемблер: RRC A

Код:

0 0 0 1 0 0 1 1

Время: 1 цикл

Алгоритм: $(A[N]) := (A[N+1])$, где $N=0-6$
 $(A[7]) := (C)$,
 $(C) := (A[0])$ Пример: ;(A)=95H (10010101B), (C)=0
RRC A ;(A)=4AH (01001010B), (C)=1**Команда SETB <бит>**

Команда "установить бит" устанавливает указанный бит в "1".

Адресуется:

- 1) к флагу переноса (C);
- 2) к биту с прямой адресацией.

1) Ассемблер: SETB C

Код:

1 1 0 1 0 0 1 1

Время: 1 цикл

Алгоритм: $(C) := 1$ Пример: ;(C)=0
SETB C ;(C)=1

2) Ассемблер: SETB (bit)

Код:

1 1 0 1 0 0 1 0

bit address

Время: 1 цикл

Алгоритм: $(bit) := 1$ Пример: ;(P2)=38H (00111000B)
SETB P2.0
SETB P2.7 ;(P2)=B9H (10111001B)**Команда SJMP <метка>**

Команда "короткий переход" выполняет безусловное ветвление в программе по указанному адресу. Адрес ветвления вычисляется сложением смещения со знаком во втором байте команды с содержимым счетчика команд после прибавления к нему 2.

Таким образом, адрес перехода должен находиться в диапазоне от 128 байт, предшествующих команде, до 127 байт, следующих за ней.

Ассемблер: SJMP <метка>

Код:

1 0 0 0 0 0 0 0

rel8

Время: 2 цикла

Алгоритм: (PC):=(PC)+2,
(PC):=(PC)+(rel8)

Пример:

; (PC)=0418H,
; метка MET1 соответствует адресу 039AH
SJMP MET1 ; (PC)=039AH, где (rel8)=80H=-128 DEC
SJMP MET2 ; (PC)=041AH, где метка MET2 соответ-
; ствует адресу 041AH,
; (rel8)=7DH=+125 DEC

Команда SUBB A, <байт источника>

Команда "вычитание с заемом" вычитает указанную переменную вместе с флагом переноса из содержимого аккумулятора, засылая результат в аккумулятор. Эта команда устанавливает флаг переноса (заема), если при вычитании для бита 7 необходим заем, в противном случае флаг переноса сбрасывается. Если флаг переноса установлен перед выполнением этой команды, то это указывает на то, что заем необходим при вычитании с увеличенной точностью на предыдущем шаге, поэтому флаг переноса вычитается из содержимого аккумулятора вместе с операндом источника. (AC) устанавливается, если заем необходим для бита 3 и сбрасывается в противном случае. Флаг переполнения (OV) устанавливается, если заем необходим для бита 6, но его нет для бита 7, или есть для бита 7, но нет для бита 6.

При вычитании целых чисел со знаком (OV) указывает на отрицательное число, которое получается при вычитании отрицательной величины из положительной, или положительное число, которое получается при вычитании положительного числа из отрицательного.

Операнд источника допускает четыре режима адресации:

- 1) регистровый
- 2) прямой
- 3) косвенно-регистровый
- 4) непосредственный (к константе).

1) Ассемблер: SUBB A, Rn ; где n=0-7

Код:

1 0 0 1 1 rrr

, где r=000-111

Время: 1 цикл

Алгоритм: (A):=(A)-(C)-(Rn);
(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример:

; (A)=C9H, (R2)=54H, (C)=1
SUBB A, R2 ; (A)=74H, (R2)=54H, (C)=0,
; (AC)=0, (OV)=1

2) Ассемблер: SUBB A, <direct>

Код:

1 0 0 1 0 1 0 1

direct address

Время: 1 цикл

Алгоритм: (A):=(A)-(C)-(direct);
(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример: ; (A)=97H, (B)=25H, (C)=0
 SUBB A,B ; (A)=72H, (B)=25H, (C)=0,
 ; (AC)=0, (OV)=1

3) Ассемблер: SUBB A,@Ri ; где i=0,1

Код:

1 0 0 1 0 1 1 i

Время: 1 цикл

Алгоритм: (A):=(A)-(C)-((Ri));
 (C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример: ; (A)=49H, (C)=1, (R0)=33H,
 ; (03Y[33])=68H
 SUBB A,R0 ; (A)=E0H, (C)=1, (AC)=0, (OV)=0

4) Ассемблер: SUBB A,#data

Код:

1 0 0 1 0 1 0 0

#data8

Время: 1 цикл

Алгоритм: (A):=(A)-(C)-(#data8);
 (C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример: ; (A)=0BEH, (C)=0
 SUBB A,#3FH ; (A)=7FH, (C)=0, (AC)=1, (OV)=1

Команда SWAP A

Команда "обмен тетрадами внутри аккумулятора" осуществляет обмен между младшими четырьмя и старшими четырьмя битами аккумулятора (между старшей и младшей тетрадами).

Эта команда может рассматриваться так же, как команда четырехбитового циклического сдвига. На флаги эта команда не влияет.

Ассемблер: SWAP A

Код:

1 1 0 0 0 1 0 0

Время: 1 цикл

Алгоритм: (A[3-0]):=(A[7-4]), (A[7-4]):=(A[3-0])

Пример: ; (A)=0D7H (11010111B)
 SWAP A ; (A)=7DH (01111101B)

Команда XCH A,<байт>

Команда "обмен содержимого аккумулятора с переменной-байтом" осуществляет обмен содержимого аккумулятора с содержимым источника, указанным в команде. Операнд источника может использовать следующие режимы адресации:

- 1) регистровый
- 2) прямой
- 3) косвенно-регистровый.

1) Ассемблер: XCH A,Rn ; где n=0-7

Код:

1 1 0 0 1 rrr

, где rrr=000-111

Время: 1 цикл

Алгоритм: (A):=(Rn), (Rn):=(A)

Пример: ; (A)=3CH, (R4)=15H
XCH A, R4 ; (A)=15H, (R4)=3CH

2) Ассемблер: XCH A, <direct>

Код:

1 1 0 0 0 1 0 1

direct address

Время: 1 цикл

Алгоритм: (A):=(direct), (direct):=(A)

Пример: ; (A)=0FEH, (P3)=0DAH

XCH A, P3 ; (A)=0DAH, (P3)=0FEH

3) Ассемблер: XCH A, @Ri, где i=0,1

Код:

1 1 0 0 0 1 1 i

Время: 1 цикл

Алгоритм: (A):=((Ri)), ((Ri)):= (A)

Пример: ; (R1)=39H, (03Y[39])=44H, (A)=0BCH

XCH A, @R1 ; (03Y[39])=0BCH, (A)=44H

Команда XCHD A, @Ri

Команда "обмен тетрадой" выполняет обмен младшей тетрады (биты 3—0) аккумулятора с содержимым младшей тетрады (биты 3—0) ячейки внутреннего ОЗУ, косвенная адресация к которой производится с помощью указанного регистра. На старшие биты (биты 7—4) эта команда не влияет (так же, как и на флаги).

Ассемблер: XCHD A, @Ri ; где i=0,1

Код:

1 1 0 1 0 1 1 i

Время: 1 цикл

Алгоритм: (A[3-0]):=((Ri[3-0])),
((Ri[3-0])):=(A[3-0])

Пример: ; (R0)=55H, (A)=89H, (03Y[55])=0A2H

XCHD A, @R0 ; (A)=82H, (03Y[55])=0A9H

Команда XRL <байт назначения>, <байт источника>

Команда "логическое ИСКЛЮЧАЮЩЕЕ ИЛИ" для переменных-байтов" выполняет операцию "ИСКЛЮЧАЮЩЕЕ ИЛИ" над битами указанных переменных, записывая результат в байт назначения. На флаги эта команда не влияет.

Допускается шесть режимов адресации:

— байтом назначения является аккумулятор:

- 1) регистровый
- 2) прямой
- 3) косвенно-регистровый
- 4) непосредственный

— байтом назначения является прямой адрес:

- 5) к аккумулятору
- 6) к константе.

1) Ассемблер: XRL A, Rn ; где n=0-7

Код:

0 1 1 0 1 rrr, где rrr=000-111

- Время: 1 цикл
 Алгоритм: (A):=(A) XOR (Rn)
 Пример: ;(A)=C3H, (R6)=0AAH
 XRL A,R6 ;(A)=69H, (R6)=0AAH
- 2) Ассемблер: XRL A,<direct>
- Код:

0 1 1 0 0 1 0 1

direct address

- Время: 1 цикл
 Алгоритм: (A):=(A) XOR (direct)
 Пример: ;(A)=0FH, (P1)=0A6H
 XRL A,P1 ;(A)=A9H, (P1)=0A6H
- 3) Ассемблер: XRL A,@Ri ; где i=0,1
- Код:

0 1 1 0 0 1 1 i

- Время: 1 цикл
 Алгоритм: (A):=(A) XOR ((Ri))
 Пример: ;(A)=55H, R1=77H, (03Y[77])=5AH
 XRL A,@R1 ;(A)=0FH, (03Y[77])=5AH
- 4) Ассемблер: XRL A,#data
- Код:

0 1 1 0 0 1 0 0

#data8

- Время: 1 цикл
 Алгоритм: (A):=(A) XOR <data>
 Пример: ;(A)=0C3H
 XRL A,#0F5H ;(A)=36H
- 5) Ассемблер: XRL <direct>,A
- Код:

0 1 1 0 0 0 1 0

direct address

- Время: 1 цикл
 Алгоритм: (direct):=(direct) XOR (A)
 Пример: ;(A)=31H, (P1)=82H
 XRL P1,A ;(A)=31H, (P1)=B3H
- 6) Ассемблер: XRL <direct>,#data
- Код:

0 1 1 0 0 0 1 1

direct address

#data8

- Время: 2 цикла
 Алгоритм: (direct):=(direct) XOR #data
 Пример: ;(IP)=65H
 XRL IP,#65H ;(IP)=00H

Примечание. Если эта команда используется для работы с портами, то значение, используемое в качестве операнда, считывается из "защелки" порта, а не с выводов БИС.

2.6. Электрические параметры

Электрические параметры микросхем в диапазоне температур от минус 10°C до 70°C приведены в табл. 2.24 (статические параметры) и табл. 2.25 (динамические параметры).

Предельные значения электрических режимов эксплуатации приведены в табл. 2.26.

Таблица 2.24. Электрические параметры микросхем в диапазоне температур от -10°C до 70°C

N п/п	ПАРАМЕТРЫ	БУКВЕННОЕ ОБОЗНАЧЕНИЕ	ЗНАЧЕНИЕ ПАРАМЕТРОВ					
			КР1816ВЕ31 КР1816ВЕ51		КР1816ВЕ751		КР1830ВЕ31 КР1830ВЕ51	
			минимал.	максимал.	минимал.	максимал.	минимал.	максимал.
1	2	3	4	5	6	7	8	9
1.	Напряжение питания, В	U_{CC}	4,5	5,5	4,75	5,25	4,5	5,5
2.	Напряжение программирующего питания высокого уровня, В	U_{PR}	-	-	20,5	21,5	-	-
3.	Выходное напряжение высокого уровня сигналов: P1(0-7), P2(0-7), P3(0-7), B P0(0-7), ALE, PWE, В	U_{OH}	2,4	-	2,4	-	2,4	-
		U_{OH1}	2,4	-	2,4	-	2,4	-
4.	Выходное напряжение низкого уровня сигналов: P1(0-7), P2(0-7), P3(0-7), B P0(0-7), ALE, PWE, В	U_{OL}	-	0,45	-	0,45	-	0,45
		U_{OL1}	-	0,45	-	0,45	-	0,45
5.	Выходной ток высокого уровня сигналов: P1(0-7), P2(0-7), P3(0-7), P0(0-7), ALE, PWE, мА	I_{OH}	-0,08	-	-0,08	-	-0,08	-
		I_{OH1}	-0,4	-	-0,4	-	-0,4	-
6.	Выходной ток низкого уровня сигналов: P1(0-7), P2(0-7), P3(0-7), P0(0-7), ALE, PWE, мА	I_{OL}	-	1,6	-	1,6	-	1,6
		I_{OL1}	-	3,2	-	2,4	-	3,2
7.	Входное напряжение высокого уровня сигналов, кроме RST и BQ2*, В	U_{IH}	2,0	-	2,0	-	2,0	-
8.	Входное напряжение высокого уровня сигналов RST и BQ2*, В	U_{IH1}	2,5	$U_{CC}+0,5$	2,5	$U_{CC}+0,5$	3,8	$U_{CC}+0,5$
9.	Входное напряжение низкого уровня, В	U_{IL}	-0,5	0,8	-0,5	0,8	-0,5	0,8

Продолжение таблицы 2.24

1	2	3	4	5	6	7	8	9
10	Входной ток высокого уровня сигналов: RST ^{**} , мкА DEMA, мкА	I_{IH} I_{IHL1}	- -	$ -500,0 $ -	- -	$ -500,0 $ $ -500,0 $	- -	$ -500,0 $ -
11	Входной ток низкого уровня сигналов: P1(0-7), P2(0-7), P3(0-7), мкА BQ2 [*] , мА DEMA, мА	I_{IL} I_{IL1} I_{IL2}	- - -	$ -800 $ $ -2,5 $ -	- - -	$ -500,0 $ $ -2,5 $ $ -15 $	- - -	$ -50,0 $ - -
12	Ток утечки сигналов, мкА	I_L	-10,0	10,0	-10,0	10,0	-10,0	10,0
13	Выходной ток в состоянии "выключено", мкА	I_{OZ}	-10,0	10,0	-10,0	10,0	-10,0	10,0
14	Ток потребления, мА	I_{CC}	-	150,0	-	220,0	-	18,0
15	Ток потребления в режиме резервного питания, мА	I_{CCS}	-	10,0	-	-	-	-
16	Ток потребления в режиме хранения содержимого регистров спецфункций (в режиме холостого хода), мА	I_{CCS1}	-	-	-	-	-	4,2
17	Ток потребления в режиме хранения содержимого ОЗУ (в режиме микропотребления), мкА	I_{CCS2}	-	-	-	-	-	50,0
18	Ток потребления по выводу U_{PR} при программировании, мА	I_{PR}	-	-	-	30,0	-	-
19	Внутреннее сопротивление в цепи "сброс", кОм	R_{RST}	-	-	-	-	40,0	125,0
20	Емкость входа/выхода, пФ	$C_{I/O}$	-	20,0	-	20,0	-	20,0
21	Входная емкость, пФ	C_I	-	10,0	-	10,0	-	10,0

Примечания к таблице 2.24.

^{*} — для серии КР1830 — BQ1.

^{**} — в серии КР1830 значение приведено для сигналов P1(0-7), P2(0-7), P3(0-7).

Таблица 2.25. Электрические параметры в диапазоне температур от -10°C до 70°C и $U_{CC}=5\text{В}\pm 5\%$

N п/п	НАИМЕНОВАНИЕ ПАРАМЕТРА, ЕДИНИЦА ИЗМЕРЕНИЯ	БУКВЕННОЕ ОБОЗНАЧЕНИЕ	ЗНАЧЕНИЕ ПАРАМЕТРОВ	
			КР1816ВЕ51 КМ1816ВЕ751 КР1830ВЕ51 КР1816ВЕ31 КС1816ВЕ751 КР1830ВЕ31	
			не менее	не более
1	2	3	4	5
ДИНАМИЧЕСКИЕ ПАРАМЕТРЫ				
1.	Период следования импульсов тактовых сигналов BQ1(BQ2), нс	$T_{BQ}=t$	83,3	286
2.	Время фронта нарастания сигнала BQ1 (BQ2), нс	$t_{BQ, LH}$	-	20
3.	Время фронта спада сигнала BQ1 (BQ2), нс	$t_{BQ, HL}$	-	20
4.	Минимальное время цикла, нс	$t_{cy \min}$	12t	-
5.	Длительность сигнала ALE, нс	$tw(ALE)$	2t-40	-
6.	Длительность сигнала \overline{PME} , нс	$tw(\overline{PME})$	3t-60	-
7.	Длительность сигнала \overline{RD} , нс	$tw(\overline{RD})$	6t-100	-
8.	Длительность сигнала \overline{WR} , нс	$tw(\overline{WR})$	6t-100	-
9.	Время задержки сигнала ALE относительно сигналов адреса A(7-0), A(8-15), нс	$td(A, ZH/ZL-ALE, HL)$	t-30	-
10.	Время задержки сигналов адреса A(7-0) относительно сигнала ALE, нс	$td(ALE, HL-A, HZ/LZ)$	t-35	-
11.	Время задержки сигнала \overline{PME} относительно сигнала ALE, нс	$td(ALE, HL-\overline{PME}, HL)$	t-25	-
12.	Время задержки сигналов адреса A(7-0) относительно сигнала \overline{PME} , нс	$td(\overline{PME}, LH-A, ZH/ZL)$	t-8	-
13.	Время задержки сигнала \overline{PME} относительно сигналов адреса A(7-0), A(8-15), нс	$td(A, HZ/LZ-\overline{PME}, HL)$	0	-
14.	Время задержки сигнала \overline{RD} относительно сигнала ALE, нс	$td(ALE, HL-\overline{RD}, HL)$	3t- 50	3t+50
15.	Время задержки сигнала \overline{WR} относительно сигнала ALE, нс	$td(ALE, HL-\overline{WR}, HL)$	3t-50	3t+50
16.	Время задержки сигнала \overline{RD} относительно сигналов адреса A(7-0), A(8-15), нс	$td(A, ZH/ZL-\overline{RD}, HL)$	4t-130	-
17.	Время задержки сигнала \overline{WR} относительно сигналов адреса A(7-0), A(8-15), нс	$td(A, ZH/ZL-\overline{WR}, HL)$	4t-130	-

Продолжение таблицы 2.25

1	2	3	4	5	
18.	Время задержки сигнала \overline{WR} относительно сигналов данных, нс	$td(D, LH/HL-\overline{WR}, HL)$ $td(D, LH/HL-\overline{WR}, LH)$	$t-70$ $7t-150$	- -	
19.	Время задержки сигналов данных $D(7-0)$ относительно сигнала \overline{WR} , нс	$td(\overline{WR}, LH-D, HZ/LZ)$	$t-50$	-	
20.	Время задержки сигналов сигналов данных $D(7-0)$ относительно сигнала \overline{RD} , нс	$td(\overline{RD}, HL-A, HZ/LZ)$	-	0	
21.	Время задержки сигнала ALE относительно сигнала \overline{WR} , нс	$td(\overline{WR}, LH-ALE, LH)$	$t-50$	$t+50$	
22.	Время задержки сигнала ALE относительно сигнала \overline{RD} , нс	$td(\overline{RD}, LH-ALE, LH)$	$t-50$	$t+50$	
23.	Время сохранения сигналов INS относительно сигнала \overline{PME} , нс	$tv(\overline{PME}, LH-INS, HZ/LZ)$	0	$t-20$	
24.	Время сохранения сигналов данных $D(0-7)$ относительно сигнала \overline{RD} , нс	$tv(\overline{RD}, LH-D, HZ/LZ)$	-	$2t-70$	
25.	Время установления сигн. INS относительно сигнала ALE, нс	$tsu(ALE, HL-INS, ZH/ZL)$	-	$4t-150$	
26.	Время установления сигн. INS относительно сигнала \overline{PME} , нс	$tsu(\overline{PME}, HL-INS, ZH/ZL)$	-	$3t-150$	
27.	Время установления сигн. INS относительно сигналов адреса $A(7-0)$, нс	$tsu(A, ZH/ZL-INS, ZH/ZL)$	-	$5t-150$	
28.	Время установления сигналов данных $D(0-7)$ относительно сигнала \overline{RD} , нс	$tsu(\overline{RD}, HL-D, ZH/ZL)$	-	$5t-165$	
29.	Время установления сигналов данных $D(0-7)$ относительно сигнала ALE, нс	$tsu(ALE, HL-D, ZH/ZL)$	-	$8t-150$	
30.	Время установления сигналов данных $D(0-7)$ относительно сигналов адреса $A(7-0)$, нс	$tsu(A, ZH/ZL-D, ZH/ZL)$	-	$9t-165$	
ДИНАМИЧЕСКИЕ ПАРАМЕТРЫ ПРИ ПРОГРАММИРОВАНИИ И ПРОВЕРКЕ РПЗУ ПРИ $t=(25\pm 5)^\circ C$					
N п/п	Наименование сигнала	Буквенное обозначение	1816BE751	1816BE51	1830BE51
1	2	3	4	5	6
31.	Время задержки сигналов данных $D(0-7)$ относительно сигналов адреса $A(7-0)$, нс	$td(A, ZH/ZL-D, ZH/ZL)$	не более 48t	-	-
32.	Время задержки сигналов данных $D(0-7)$ относительно сигнала \overline{E} , нс	$td(\overline{E}, HL-D, ZH/ZL)$ $td(\overline{E}, LH-D, HZ/LZ)$	не более 48t	-	-

Продолжение таблицы 2.25

1	2	3	4	5	6
33.	Длительность сигнала \overline{PR} низкого уровня, мс	$tw(PR, L)$	45–55	–	–
34.	Время установления адреса относительно сигнала \overline{PR} , нс	$tsu(A, ZH/ZL-PR, HL)$	не менее 48t	–	–
35.	Время сохранения адресного сигнала относительно сигнала \overline{PR} , нс	$tv(PR, LH-A, HZ/LZ)$	не менее 48t	–	–
36.	Время установления данных $D(0-7)$ относительно сигнала \overline{PR} , нс	$tsu(D, ZH/ZL-PR, HL)$	не менее 48t	–	–
37.	Время сохранения сигнала данных $D(0-7)$ относительно сигнала \overline{PR} , нс	$tv(PR, LH-D, HZ/LZ)$	не менее 48t	–	–
38.	Время установления сигнала U_{PR} относительно \overline{PR} , мкс	$tsu(U_{PR}, LH-PR, HL)$	не менее 10	–	–
39.	Время сохранения сигнала U_{PR} относительно сигнала \overline{PR} , мкс	$tv(PR, LH-U_{PR}, HL)$	не менее 10	–	–
40.	Время установления сигнала U_{PR} относительно сигнала \overline{E} , нс	$tsu(\overline{E}, LH-U_{PR}, LH)$	не менее 48t	–	–
41.	Частота следования импульсов BQ , МГц	f_{BQ}	4 – 6	4 – 6	4 – 6
42.	Время задержки сигналов данных $D(7-0)$ относительно сигналов адреса $AP1(0-7)$, $AP2(0-3)$ при проверке внутреннего ПЗУ	$td(AP1, LH/HL-D, ZH/ZL)$ $td(AP2, LH/HL-D, ZH/ZL)$	–	не более 48t	не более 48t
43.	Время задержки сигналов данных $D(7-0)$ относительно сигнала разрешения $P2(7)$ при проверке внутреннего ПЗУ	$td(P2, HL-D, ZH/ZL)$ $td(P2, LH-D, HZ/LZ)$	–	не более 48t	не более 48t

Примечания к таблице 2.25.

1. Для микросхем КР1830ВЕ51 (ВЕ31) $t = T_{BQ1}$. Для микросхем КР1816ВЕ51 (ВЕ31) и КМ1816ВЕ751, КС1816ВЕ751 $t = T_{BQ2}$.

2. Временные диаграммы представлены на рис. 2.13–2.17.

3. Погрешность измерения временных параметров ± 10 нс.

4. Динамические параметры приведены для емкостей нагрузки:

— по выводам $P0$, ALE , \overline{PME} — 100 пф;

— по остальным выводам — 80 пф.

5. Символ "LH" ("HL") обозначает переход сигнала из состояния низкого (высокого) уровня в состояние высокого (низкого) уровня. Символ "ZH" ("ZL") обозначает переход сигнала из высокоимпедансного состояния в состояние высокого (низкого) уровня. Символ "HZ" ("LZ") обозначает переход сигнала высокого (низкого) уровня в высокоимпедансное состояние.

2.7. Примеры применения

Представленные ниже материалы состоят из трех разделов:

- Примеры программирования микроЭВМ семейства МК51;
- Программные методы сопряжений периферийных устройств;
- Схемы включения ОМЭВМ.

Таблица 2.26. Предельные значения электрических режимов эксплуатации

N п/ п	НАИМЕНОВАНИЕ	БУКВЕННОЕ ОБОЗНАЧЕНИЕ	Н О Р М А					
			KP1816BE31 KP1816BE51		KP1816BE751		KP1830BE31 KP1830BE51	
			не менее	не более	не менее	не более	не менее	не более
1	2	3	4	5	6	7	8	9
1.	Напряжение питания, В	U_{CC}	-0,51	7,0	-0,51	7,0	-0,51	7,0
2.	Напряжение программирующего питания высокого уровня, В	U_{PR}	-	-	-	23,0	-	-
3.	Выходное напряжение на выводах, В	U_I	-0,51	7,0	-0,51	7,0	-0,51	7,0
4.	Выходной ток низкого уровня для выводов $P0$, ALE , \overline{PME} , мА остальных выводов, мА	I_{OL1}	-	5,0	-	3,5	-	5,0
		I_{OL}	-	3,0	-	2,0	-	3,0
5.	Выходной ток высокого уровня для выводов $P0$, ALE , \overline{PME} , мА остальных выводов, мА	I_{OH1}	-0,8	-	-0,4	-	-0,8	-
		I_{OH}	-0,2	-	-0,08	-	-0,2	-
6.	Емкость нагрузки для выводов $P0$, ALE , \overline{PME} , мА остальных выводов, мА	C_{L1}	-	500,0	-	500,0	-	500,0
		C_L	-	100,0	-	100,0	-	100,0
7.	Количество циклов перепрограммирования	$N_{цикл.}$				25		

Первый раздел содержит примеры использования программного обеспечения для составления программ общего пользования. Некоторые программы относятся к программам арифметических вычислений двойной точности и методам просмотра таблиц.

Во втором разделе представлены программы, обслуживающие порты ввода-вывода микроЭВМ, последовательный канал и таймер/счетчики. В этом разделе с другими программами рассматриваются вопросы, связанные с изменением конфигурации портов ввода-вывода, организации задержки программным способом и передачи через последовательный порт строк символов наряду.

Третий раздел состоит из схем включения микроЭВМ семейства МК51 с периферийными устройствами и схемами ЗУ.

2.7.1. Примеры программирования микроЭВМ

2.7.1.1. Программа преобразования системы счисления

Команда деления может быть использована для перевода числа из одной системы счисления в другую. Программа BINBCD выполняет преобразование целого двоичного 8-разрядного числа без знака, содержащегося в регистре A (значение в интервале 0—255), в трехзначное число двоично-кодированного формата BCD (два байта). Число сотен возвращается в поле переменной HUND, а числа десятков и

единиц возвращаются в двоично-десятичном упакованном коде в поле другой переменной TENONE.

```

;BINBCD    Выполняет преобразование 8-разрядного двоичного
            числа, хранящегося в регистре, в трехзначное
            число в двоично-десятичном упакованном формате.
            Число сотен размещено слева в поле переменной
            HUND, а числа десятков и единиц размещены в
            переменной TENONE.

HUND       DATA    21H
TENONE     DATA    22H          ;Разделить на 100 для
                                   ;определения числа сотен

BEGIN:     MOV      B,#100
            DIV      AB          ;Разделить остаток на 10
            MOV      HUND,A      ;для определения числа
            MOV      A,#10       ;сотен слева
            XCH      A,B         ;Цифра десятков в A
            DIV      AB          ;Остаток - цифра единиц
            SWAP     A           ;В A цифры в двоично-деся-
            ADD      A,B         ;тичном упакованном формате
            MOV      TENONE,A
            RET

```

Команда деления может быть также использована для отделения подполей данного в аккумуляторе. Например, выполняя деление упакованного двоично-десятичного данного на 16 можно отделить два полубайта, старшие разряды в аккумуляторе, а младшие (остаток) — в регистре B. Каждое данное выравнено вправо и, следовательно, может обрабатываться отдельно. В приведенном примере два упакованных двоично-десятичных числа хранятся в аккумуляторе, выполняется отделение каждого числа, затем вычисляется их произведение и возвращается в упакованном двоично-десятичном формате в аккумулятор.

```

;DISBCD    Выполняет распаковку двух упакованных двоично-
            десятичных чисел, полученных в аккумуляторе,
            и возвращает их произведение в аккумулятор
            также в двоично-десятичном упакованном формате

BEGIN:     MOV      B,#10H       ;Исходное значение де-
            DIV      AB          ;лится на 16. В регист-
                                   ;рах A и B хранятся вы-
                                   ;деленные цифры (каждая
                                   ;выравнена вправо)
            MUL      AB          ;Регистр A содержит про-
                                   ;изведение в двоичном
                                   ;формате (0-99(десятич-
                                   ;ное)=0-63H)

            MOV      B,#10       ;Произведение делится
            DIV      AB          ;на 10. Регистр A содер-
                                   ;жит цифру десятков,
                                   ;B содержит остаток
            SWAP     A           ;Цифры в упакованном
            ORL      A,B         ;формате
            RET

```

2.7.1.2. Арифметические операции двойной точности

Команды ADDC и SUBB учитывают предыдущее состояние флажка переноса (заема), что позволяет производить вычисления с двойной точностью путем повторения операции над последующими старшими байтами операнда. Если входными данными для операции является строка целых чисел без знака, то флажок переноса установится после завершения операции при переполнении (для ADDC) или

при исчезновении значащих разрядов (для SUBB). Для отрицательных данных, представленных дополнительным кодом, старший разряд старшего байта исходных данных содержит знак строки, поэтому флажок переполнения (OV) будет указывать на переполнение или исчезновение значащих разрядов.

```

;SUBSTR  Вычитает строку, указанную регистром R1, из строки,
          указанной регистром R0, с точностью, указанной ре-
          гистром R2. После выполнения операции проверяется
          переполнение результата.
SUBSTR:  CLR      C      ;Заем=0
BEGIN:   MOV      A,@R0   ;Загрузка байта уменьшаемого
          SUBB     A,@R1   ;Вычитание байта
          MOV      @R0,A   ;Запоминание байта разности
          INC      R0      ;Установка указателей на следующее
          INC      R1      ;поле
          DJNZ     R2,BEGIN;Выполнение цикла до завершения
                           ;операции
;        После завершения цикла проверяется ситуация пере-
;        полнения в последней итерации.
          JNB      OV,OK
;        ..... ;Программа восстановления старших
;        ;разрядов
OK:      RET

```

2.7.1.3. Последовательности просмотра таблиц

Обе версии команды MOVC используются как часть трехшаговой последовательности для осуществления доступа к таблицам в памяти ПЗУ. При использовании версии DPTR инструкции MOVC следует загрузить указатель данных (DPTR) значением начального адреса таблицы; загрузить аккумулятор индексом (координатой входа); выполнить команду MOVC A,@A+DPTR. Значения DPH и DPL могут быть вычислены или модифицированы с помощью стандартных арифметических инструкций. Версия PC инструкции MOVC используется для "локальных" таблиц и имеет то преимущество, что она не изменяет значения указателя данных. Это делает ее полезной для программ обработки прерываний или других ситуаций, в которых значение DPTR служит для других целей.

Последовательность просмотра состоит из трех шагов: загрузки аккумулятора индексом; восстановления адреса инструкции просмотра таблицы для начала просмотра путем добавления смещения к аккумулятору; выполнения инструкции MOV A, @A+PC. Ниже приводится задача, когда в одномерной памяти программы хранятся большие многомерные таблицы, содержащие комбинации двоичных разрядов, нелинейные калибровочные параметры и т.д. Для выборки данных из таблицы переменные, представляющие индексы элементов, должны быть преобразованы в адрес памяти для записи. Для матрицы с размерностью (M x N), начальным адресом STRT и соответствующими индексами I и J адрес элемента (I, J) определяется по формуле:

$$\text{Адрес записи} = [\text{STRT} + (\text{N} \times \text{I}) + \text{J}]$$

Подпрограмма MATRX1 может осуществлять доступ к записи любого массива, содержащего до 255 элементов (в примере массив размерностью 11x21 содержит 231 элемент). Записи в таблицу объявляются при помощи директивы Data Byte (DB) и будут содержаться в объектном коде как часть самой программы. В более общем случае может быть использована подпрограмма MATRX2, обеспечивающая доступ к таблицам неограниченных размеров путем комбинации инструкции MUL, арифметических операций двойной точности и версии PC инструкции MOVC. Единственное ограничение накладывается на значение индекса, которое находится в пределах 0—255.

```

; MATRX1  Загружает константу в аккумулятор из двумерной
;         таблицы, размещенной в памяти команд, при помо-
;         щи инструкции просмотра MOVC A,@A+PC. Общее
;         число записей меньше 255 - в данном примере
;         используется матрица 11 x 21.
;         Адрес записи определяется по формуле:
;         [(STRT) = (21 x I) + (J)]
I      EQU      R6      ;Первая координата записи (0-10)
J      DATA    23H     ;Вторая координата записи (0-20)
MATRX1: MOV      A,I
        MOV      B,#21
        MUL      AB      ;(21 x I)
        ADD      A,J      ;Добавить смещение внутри строки
;         Предусматривается байт между инструкцией MOVC и
;         записью (0,0).
        INC      A
        MOVC     A,@A+PC ;Запись 0,0
        RET
BASE1:  DB1      1       ;Запись 0,1
        DB2      2
;         ..... ;Запись 0,20
        DB       21      ;Запись 1,0
        DB       22
;         ..... ;Запись 1,20
;         DB       42
;         .....
;         DB       231    ;Запись 10,20
MATRX2: MOV      A,I      ;Загрузка первой координаты
        MOV      B,#N
        MUL      AB
        ADD      A,#LOW   ;Сложить с 16-разрядным
                           ;базовым адресом
        MOV      DPL,A
        MOV      A,B
        ADDC     A,#HIGH
        MOV      DPH,A    ;DPTR=(STRT)+(I x N)
        MOV      A,J
        MOVC     A,@A+DPTR ;Добавить индекс и выбрать
                           ;байт
        RET

```

2.7.1.4. Сохранение состояния центрального процессора во время прерываний

При распознавании запроса прерывания управление программой переходит к соответствующей сервисной подпрограмме путем выполнения центральным процессором команды длинного вызова (LCALL). Адрес программы сохраняется в стеке. После завершения сервисной подпрограммы инструкция RETI возвращает процессор в основную программу в точку прерывания. Программа обработки прерывания не должна изменять значения переменных, используемых в основной программе, для возможности продолжения правильной работы последней. Инструкции PUSH и POP обеспечивают сохранение регистров в стеке.

```

LOC      EQU      $      ;Запоминание счетчика
                           ;адреса
                           ORG      0003H ;Начальный адрес про-
                           LJMP     SERV ;граммы прерывания
                           ..... ;Выполнение подпрограммы
                           ORG      LOC   ;Восстановление счетчика

```

```

SERV:      PUSH    PSW      ; адреса
           PUSH    ACC      ; Запоминание регистра PSW
           PUSH    B
           PUSH    DPL
           PUSH    DPH
           MOV     PSW, #1000B ; Выбор регистрового банка 1
;           .....          ; Тело подпрограммы
           POP     DPH      ; Восстановление регистров
           POP     DPL      ; в обратном порядке
           POP     B
           POP     ACC
           POP     PSW
           RETI            ; Возврат к основной программе

```

2.7.1.5. Условный переход с N-ветвлениями

Обычно в ситуациях условного перехода с N-ветвлениями не все возможные адреса известны во время ассемблирования. Из множества небольших программ выбирается одна в соответствии со значением индекса переменной, который определяется во время выполнения программы. Наиболее эффективным путем решения этой проблемы является использование инструкции `MOVC` и инструкции перехода с косвенной адресацией, которая использует небольшую таблицу смещений ПЗУ для указателей начальных адресов соответствующих подпрограмм.

Инструкция `JMP @A+DPTR` выполняет переход с косвенной адресацией по адресу, определяемому в процессе выполнения программы. По этой инструкции выполняется сложение без знака 8-разрядного содержимого аккумулятора с содержимым 16-разрядного указателя данных (аналогично тому, как действует инструкция `MOVC A, @A+DPTR`). Результат сложения загружается в счетчик команд и используется в качестве адреса для последующих выборок инструкций. 16-разрядное сложение выполняется так, что перенос из младшего байта распространяется на старший байт. При этом содержимое аккумулятора и `DPTR` сохраняется.

Ниже приводится пример подпрограммы, в которой байт памяти ЗУ считывается в аккумулятор из одной из четырех возможных областей памяти, определяемой содержимым переменной `MEMSEL`. Адрес считываемого байта определяется содержимым `R0`. Эта подпрограмма может использоваться для терминалов, когда четыре различных модели принтеров используют один и тот же код памяти ПЗУ, но при этом используются различные типы (и размеры) буферной памяти при различных скоростях и режимах работы.

```

MEMSEL EQU R3
BRANCH: MOV A, MEMSEL
        MOV DPTR, #TBL
        MOVC A, @A+DPTR
        JMP @A+DPTR
TBL:    DB MSP0-TBL
        DB MSP1-TBL
        DB MSP2-TBL
        DB MSP3-TBL
MSP0:   MOV A, @R0      ; Чтение внутренней памяти
        RET
MSP1:   MOVB A, @R0     ; Чтение 256 байт внешней
        RET            ; памяти ЗУ
MSP2:   MOV DPL, R0     ; Чтение 64К внешней памяти
        MOV DPH, R1     ; ЗУ
        MOVB A, @DPTR
        RET
MSP3:   MOV A, R1       ; Чтение 4К байт внешней

```

ANL	A, #07H	; памяти ЗУ
ANL	P1, #11111000B	
ORL	P1, A	
MOVX	A, @R0	
RET		

Для использования этого подхода необходимо, чтобы размер таблицы адресов перехода плюс длина возможных программ не превышала 256 байт. Таблица адресов переходов и программы могут размещаться в любом месте памяти команд и не зависят от 256-байтной структуры страниц памяти программ.

2.7.1.6. Последовательная передача параметров

Если программа вызывается часто, то наиболее эффективным программным способом передачи параметров является последовательный. Константы в этом случае являются частью программного кода и следуют непосредственно за инструкцией вызова подпрограммы. По адресу возврата в стеке программа определяет место расположения параметров для последующего считывания. Например, подпрограмма LINE выполняет сложение 16-разрядной упакованной двоично-десятичной константы с двухбайтной двоично-десятичной переменной, хранящейся во внутренней памяти ЗУ, и запоминает сумму в другом двухбайтном буфере. В подпрограмму должны передаваться константа и оба адреса буферов. Вместо использования для этих целей четырех рабочих регистров используются 4 байта в памяти программ, следующих за командой вызова подпрограммы. В приведенном ниже примере подпрограмма осуществляет сложение десятичной константы 1234 с содержимым адреса внутренней памяти 56H и записывает результат в буфер по адресу 78H. Подпрограмма LINE определяет, из какой точки был осуществлен ее вызов путем извлечения из стека адреса возврата в старший и младший байты указателя данных. Инструкция MOVC затем считывает параметры из памяти программы по мере необходимости.

CALL	LINE	
DW	1234H	;BCD-константа
DB	56H	;Адрес строки операнда
DB	78H	;Адрес строки результата
;	Продолжение программы
LINE:	POP	DPH ;Извлечение адреса возврата в DPTR
	POP	DPL
	MOV	A, #2 ;Индекс строк источника параметров
	MOVC	A, @A+DPTR ;Извлечение адреса строки источника
	MOV	R0, A
	MOV	A, #3 ;Индекс строк приемника параметров
	MOVC	A, @A+DPTR ;Извлечение адреса строки приемника
	MOV	R1, A
	MOV	A, #1 ;Индекс мл.байта 16-разр. константы
	MOVC	A, @A+DPTR ;Извлечение мл. байта константы
	ADD	A, @R0 ;Вычисление мл. байта суммы
	DA	A ;Десятичная коррекция
	MOV	@R1, A ;Запись в буфер
	INC	R0
	INC	R1
	CLR	A ;Индекс ст. байта равен нулю
	MOVC	A, @A+DPTR ;Извлечение ст. байта константы
	ADDC	A, @R0
	DA	A ;Десятичная коррекция
	MOV	@R1, A ;Запись в буфер
	MOV	A, #4 ;Индекс для продолжения программы
	JMP	@A+DPTR ;Переход к инструкции, следующей ;за списком параметров.

Следует учесть, что инструкции ACALL и LCALL не используются в программе, поскольку каждая из них продвигает адрес в стек. В то же время инструкция MOVC имеет доступ ко всем 64 К байтам адресного пространства памяти программ микроЭВМ.

2.7.2. Программные методы сопряжения периферийных устройств

2.7.2.1. Переконфигурация портов ввода-вывода

Порты ввода-вывода часто должны выполнять функции приема/передачи данных, формат которых не соответствует 8-разрядному байту. Пусть, например, требуется три пятипроводных порта с фиксацией на выходе (обозначим их X, Y, Z), которые планируется отобразить на контакты портов P1 и P2. При таком планировании остается один свободный контакт P2.7, который может использоваться для тестирования, для ввода данных или для управления выводом программным путем. При этом разряды порта Z расположены в обратном порядке по отношению к порту P2. Ниже приведена табличка соответствия выводов портов:

Порт Z:	PZ0	PZ1	PZ2	PZ3	PZ4
P2.7	P2.6	P2.5	P2.4	P2.3	P2.2
Порт Y:	PY4	PY3	PY2	PY1	PY0
	P2.1	P2.0	P1.7	P1.6	P1.5
Порт X:	PX4	PX3	PX2	PX1	PX0
	P1.4	P1.3	P1.2	P1.1	P1.0

Для компенсации такого несовпадения контактов соединения могут быть выполнены программным путем.

```

PX      DATA    20H
PY      DATA    21H
PZ      DATA    22H
;
OUTPX:  ANL      A,#00011111B ;Сброс разрядов ACC.7-ACC.5
        MOV      PX,A
        ACALL    OUTP1        ;Обновление по порту 1
        RET
;
OUTPY:  MOV      PY,A
        ACALL    OUTP1
        ACALL    OUTP2        ;Обновление по порту 2
        RET
;
OUTPZ:  MOV      PZ,A
        ACALL    OUTP2
        RET
;
OUTP1:  MOV      A,PY
        SWAP     A
        RL       A            ;Сдвиг влево на 5 бит
        ANL      A,#11100000B
        ORL      A,PX
        MOV      P1,A        ;Включение разрядов PX
        RET
;
OUTP2:  MOV      C,PZ.0        ;Загрузка CY битом P2.6
        RLC      A            ;и сдвиг в аккумуляторе
        MOV      C,PZ.1        ;Загрузка CY битом P2.5
        RLC      A            ;и сдвиг в аккумуляторе
        ;

```

MOV	C,PZ.4	;Загрузка СУ битом P2.2
RLC	A	;и сдвиг в аккумуляторе
MOV	C,PY.4	;Загрузка СУ битом P2.1
RLC	A	
MOV	C,PY.3	;Загрузка СУ битом P2.0
RLC		
SETB	ACC.7	;Ввод с P2.7
MOV	P2,A	
RET		

Каждая подпрограмма (OUTPX, OUTPY, OUTPZ) вызывается для вывода выравненных вправо в аккумуляторе данных, при этом значение разрядов ACC.7—ACC.5 несущественно.

2.7.2.2. Сопряжение с расширителем ввода-вывода KP580BP43

Квазидвунаправленная структура портов микроЭВМ позволяет осуществлять ввод данных через любой контакт ввода-вывода, вывод данных с фиксации, или использовать этот контакт в качестве тестового или строб-импульса, управляемого программно. Примером такого сопряжения является интерфейс базового процессора с расширителем ввода/вывода KP580BP43 (рис. 2.28). Протокол сопряжения при помощи простого программного обеспечения приведен ниже.

;	Ввод данных с расширителя ввода-вывода, подклю-
;	ченного к контактам P23—P20. Контакты P25 и P24
;	имитируют CS и PROG. Контакты P27—P26 используются
;	как входы. Код порта считывается из ACC.1—ACC.0.
PROG	BIT P2.4 ;Описание контакта
BP43:	ORL A,#11010000B ;Установка PROG и контак-
	;тов, используемых как вх.
	MOV P2,A ;Вывод кода порта и кода
	;операции
	CLR PROG ;
	ORL P2,00001111B ;Установка младших контактов
	;для ввода.
	MOV A,P2 ;Считывание данных порта
	ORL P2,#00110000B ;Установка PROG и CS

2.7.2.3. Программная организация задержки

В большинстве применений микроЭВМ семейства МК51 предполагается управление синхронизацией по выводу данных. Например, программно генерируемый выходной строб-импульс должен составлять 50 мкс. Инструкция DJNZ может ввести задержку в сегмент программы, добавляя за каждую итерацию два машинных цикла, две инструкции могут задержать выполнение программы генерации импульса WR:

CLR	WR
MOV	R2,#24
DJNZ	R2,\$
SETB	WR

Знак "\$" в примере является специальным символом адреса текущей инструкции и может использоваться для исключения меток из строк текста программы.

2.7.2.4. Конфигурация последовательного порта и таймера

Для задания конфигурации последовательного порта микроЭВМ семейства МК51 последовательный порт и слова управления таймером должны быть инициализированы соответствующими значениями.

Предположим, что микроЭВМ будет взаимодействовать со стандартным терминалом, выполняющим передачу данных со скоростью 2400 бод (бит в сек).

Каждый символ передается последовательностью семи информационных бит, бита четности и одного стоп-бита. Таким образом, скорость передачи символов составит 2400 бод/9 бит, то есть приблизительно 265 символов в секунду.

В данном примере для простоты подпрограммам передачи и приема данных предпочтение отдано программному опросу состояния, а не прерываниям.

Последовательный порт должен быть инициализирован в режиме 8-разрядного универсального асинхронного приемопередатчика (УАПП) ($SM0=0$, $SM1=1$), позволяющем принимать все сообщения ($SM2=0$, $REN=1$). Флаг, указывающий на то, что регистр передачи свободен, будет искусственно устанавливаться для того, чтобы позволить программно определять доступность регистра вывода. Эта операция выполняется с помощью инструкции, помеченной меткой SPINIT.

Таймер 1 будет использоваться в режиме автозагрузки в качестве генератора скорости передачи данных. Чтобы получить скорость передачи данных 2400 бод, необходимо, чтобы таймер вырабатывал сигнал синхронизации с частотой 1:32 по отношению к частоте внутренней синхронизации, что соответствует 13 машинным циклам. Таймер должен перезагружать значение —13 или 0F3H (десятичное со знаком или шестнадцатичное).

```
;      Инициализация последовательного порта для
;      8-разрядного режима УАПП и установка флага
;      готовности передачи.
SPINIT: MOV     SCON,#01010010B
;      Инициализация Таймера 1 для автозагрузки
;      с частотой 32х2400 Гц.
TINIT:  MOV     TCON,#1101001B
        MOV     TH1,#-13
        SETB    TR1
```

2.7.2.5. Простые драйверы для последовательного ввода-вывода

Подпрограмма SPOUT управляет передачей символа, поступающего в нее через аккумулятор. Прежде всего программа вычисляет бит четности, вводит его в состав информационного байта, ожидает разрешения передачи, затем выводит символ и возвращает управление вызывающей программе.

Подпрограмма SPIN также проста. Она ожидает приема, устанавливает флаг переноса в случае ошибки четности и возвращает маскированный 7-разрядный код в аккумулятор.

```
; SPOUT  Добавляет разряд четности к аккумулятору и
;         осуществляет передачу при наличии готовности
;         последовательного порта..
SPOUT:  MOV     C,P
        CPL     C
        MOV     ACC.7,C
        JNB     TI,$
        CLR     TI
        MOV     SBUF,A
        RET

;
;      SPIN  Вводит очередной символ из последовательного
;            порта и устанавливает перенос в случае
;            ошибки четности.
SPIN:   JNB     RI,$
        CLR     RI
        MOV     A,SBUF
        MOV     C,P
        CPL     C
        ANL     A,#7FH
        RET
```

2.7.2.6. Передача символьной строки через последовательный порт

При передаче символов через последовательный порт в устройство вывода (например, принтер с форматом ASCII) может возникнуть необходимость в выводе служебных сообщений, включающих сообщения об ошибках, диагностику или инструкции для оператора. Эти символьные строки просто описываются при помощи директивы DB.

```

CR      EQU      0DH      ;Код возврата каретки (ASCII)
LF      EQU      0AH      ;Код перевода строки
ESC     EQU      1BH      ;Код операции ESC
;
;      .....
CALL    STRING
DB      CR,LF      ;Установка новой строки
DB      "HALLO!"    ;Сообщение
DB      ESC         ;Завершение
;
STRING: POP      DPH      ;Загрузка первого символа
        POP      DPL
STR1:   CLR      A
        MOVC     A,@A+DPTR ;Выборка первого символа
STR2:   JNB      TI,$      ;Ожидание готовности
        CLR      TI
        MOV      SBUF,A    ;Выдача символа
        INC      DPTR      ;Сдвиг указателя
        CLR      A
        MOVC     A,@A+DPTR ;Выборка следующего символа
        CJNE     A,#ESC,STR2 ;Цикл чтения
        MOV      A,#1
        JMP      @A+DPTR    ;Возврат к программе
;      ; после чтения

```

2.7.2.7. Чтение содержимого таймера/счетчика

Встречаются ситуации, когда желательно иметь возможность считывания текущего состояния таймера без нарушения процесса синхронизации. Все регистры таймеров/счетчиков микроЭВМ могут читаться или записываться в процессе работы, однако для этого надо соблюдать меры предосторожности.

Предположим, что подпрограмма RDTIME должна возвращать 16-разрядное значение в регистры R1 и R0, отражающее содержимое таймера 0. Существует опасность, что при считывании двух половин числа переполнение младшего регистра может перейти на содержимое старшего регистра, так что две возвращаемых половины слова окажутся "не в фазе". Решение этой проблемы заключается в том, чтобы читать сначала старший байт, а затем — младший, после чего выдать подтверждение того, что старший байт не изменился. Если изменение имело место, то следует повторить процесс сначала.

```

RDTIME: MOV      A,TH0      ;Выборка содержимого таймера T0
        MOV      R0,TL0
        CJNE     A,TH0,RDTIME
        MOV      R1,A
        RET

```

2.7.3. Схемы включения ОМЭВМ

Схемы включения периферийных устройств и ЗУ с микроЭВМ семейства МК51 приведены на рис. 2.28—2.33.

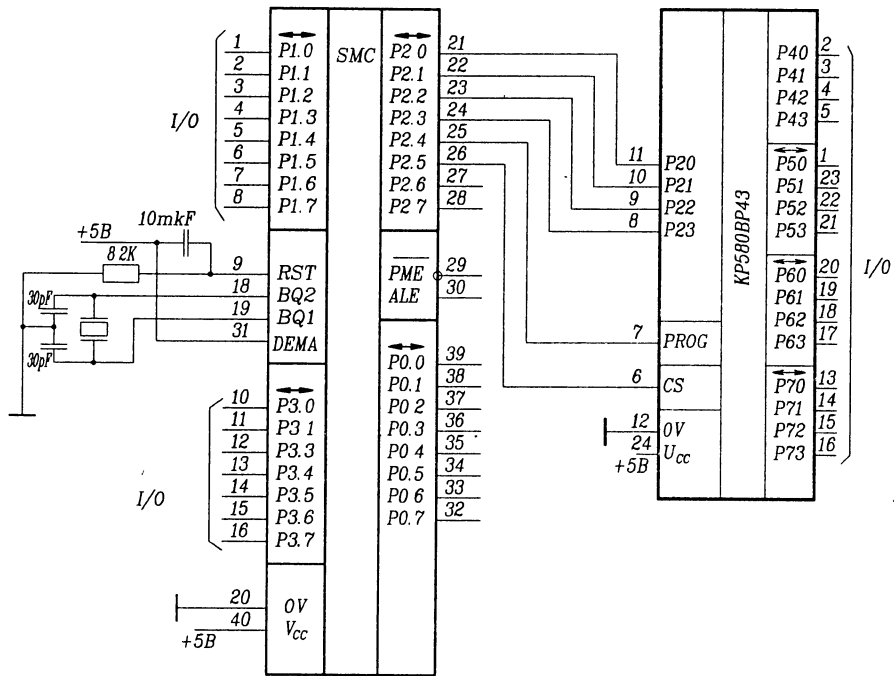


Рис. 2.28. Схема расширения портов ввода-вывода с использованием микросхемы KP580BP43

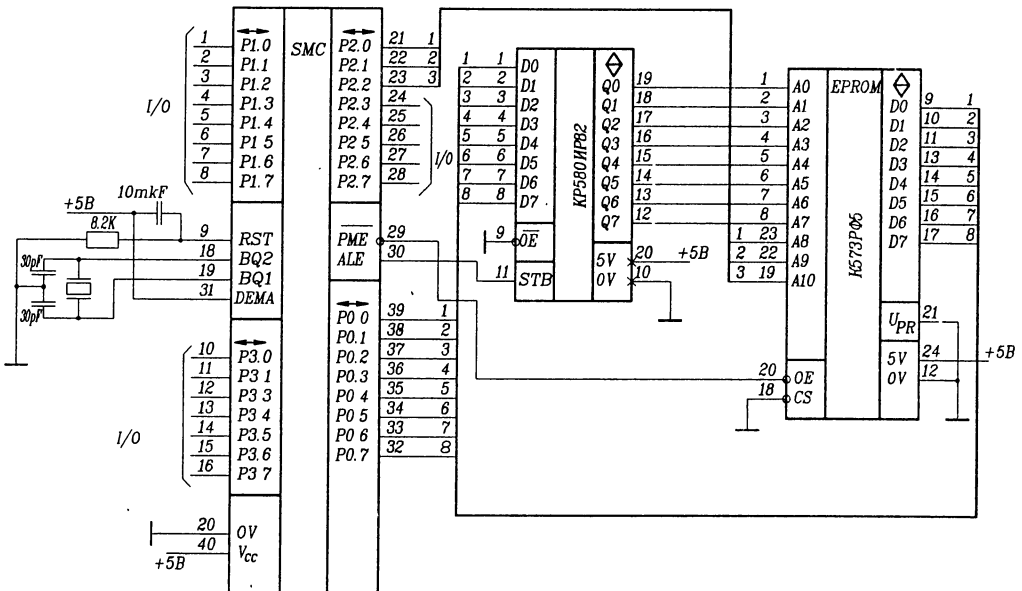


Рис. 2.29. Схема подключения внешней памяти программ с использованием ИС K573PФ5

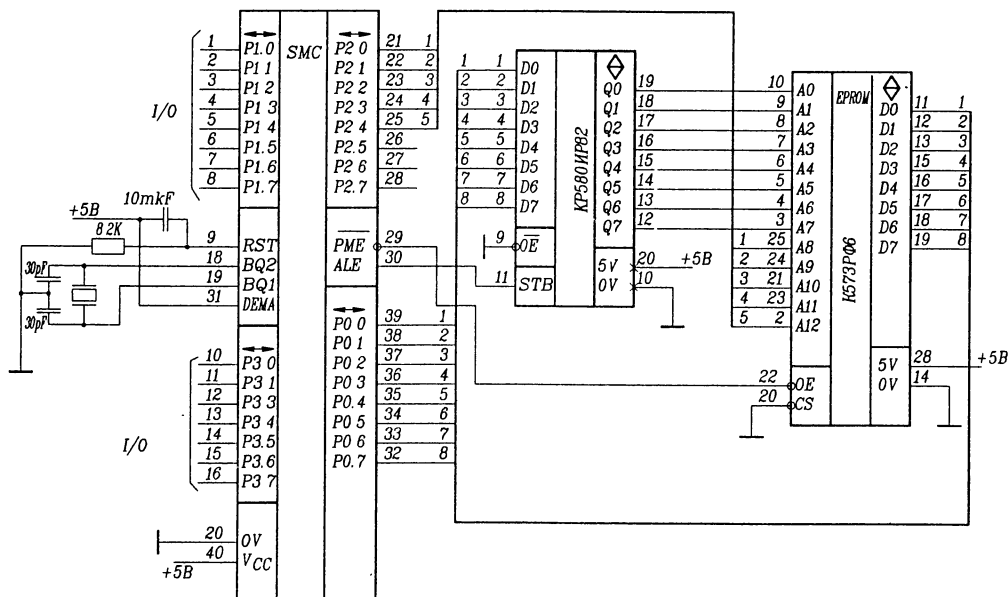


Рис. 2.30. Схема подключения внешней памяти программ с использованием ИС K573PΦ6

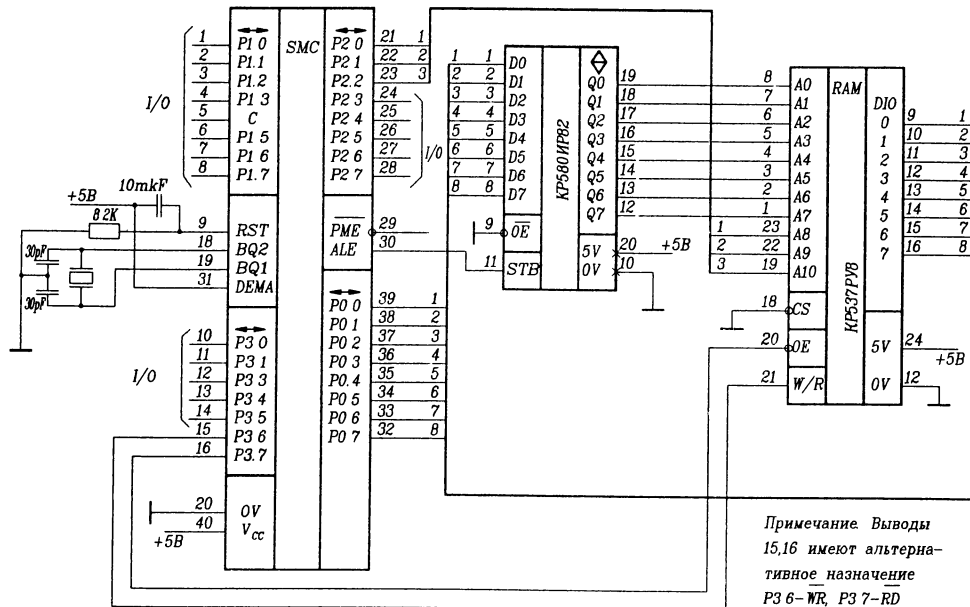
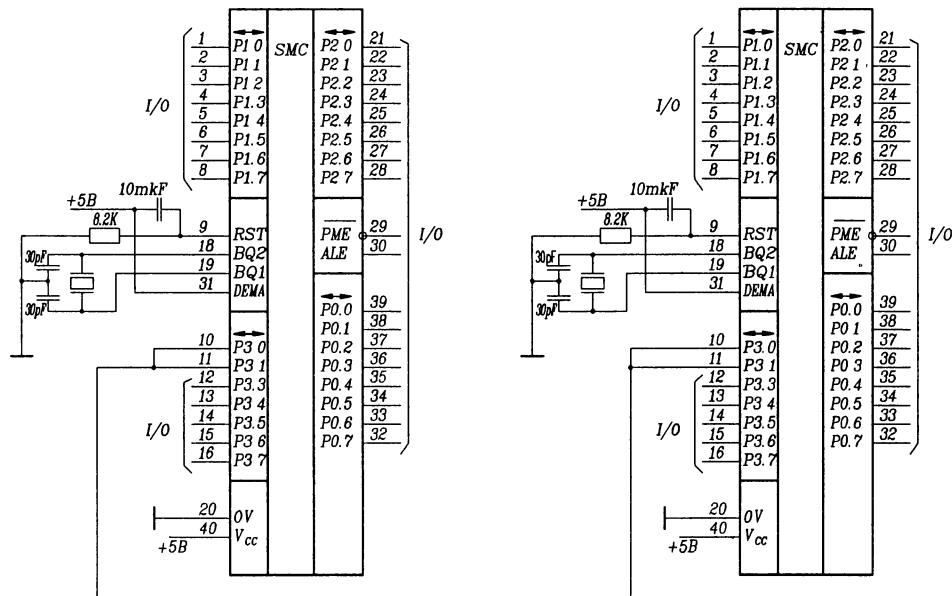
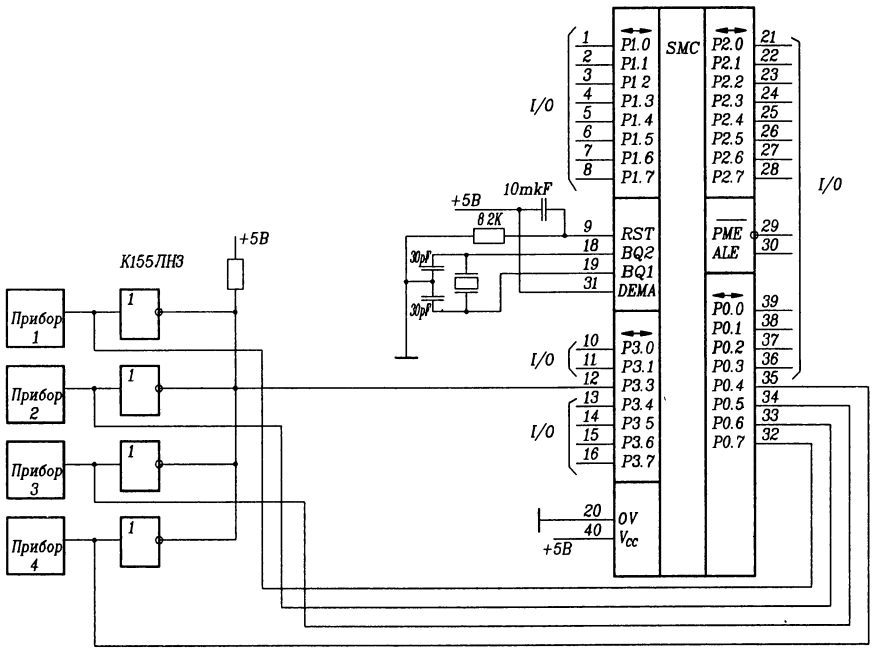


Рис. 2.31. Схема подключения внешней памяти данных с использованием ИС KP537PY8



Примечание. Выводы 10, 11 имеют альтернативное назначение: P3.0 — RxD, P3.1 — TxD

Рис. 2.32. Схема организации полудуплексного последовательного канала с использованием пары ОМЭВМ



Примечание. Вывод 12 имеет назначение INT0

Рис. 2.33. Схема увеличения числа источников прерывания ОМЭВМ

ГЛАВА 3 УСЛОВИЯ ЭКСПЛУАТАЦИИ

Микросхемы должны быть устойчивы к механическим и климатическим воздействиям по ГОСТ.18725-83, в том числе:

- линейное ускорение — 5000 (500)м/с² (g);
- пониженная рабочая температура среды — минус 10 °С;
- повышенная рабочая температура среды — + 70 °С;
- повышенная предельная температура среды — + 85 °С;
- изменения температуры среды — от минус 60 °С до + 85 °С.

Предельно допустимая температура верхней со стороны монтажа поверхности корпуса (теплоотвода) микросхем — + 80 °С.

Наработка микросхем 50000 ч., а в следующих облегченных режимах : отклонении напряжения питания от номинального не более 1% и нормальных климатических условиях — 60000 ч.

Интенсивность отказов в течение наработки не более $1.0 \cdot 10^{-6}$ 1/ч.

Гамма-процентный срок сохраняемости 12 лет.

Температура пайки (235 ± 5) °С, расстояние от корпуса до места пайки не менее 1,0 мм, продолжительность пайки 2,5 с. Микросхемы выдерживают воздействие тепла, возникающего при температуре пайки (260 ± 5) °С.

Предельные значения электрических режимов ОМЭВМ семейства МК48 приведены в табл. 1.10, а семейства МК51 — в табл. 2.26.

Типовые зависимости основных электрических параметров микросхем КМ1816ВЕ48, КР1816ВЕ35 представлены на рис. 3.1—3.8; микросхем КР1816ВЕ39, КР1816ВЕ49 — на рис. 3.9—3.16; микросхем КР1830ВЕ35, КР1830ВЕ48 — на рис. 3.17—3.29; микросхем КР1816ВЕ31, КР1816ВЕ51, КМ1816ВЕ751 — на рис. 3.30—3.48; микросхем КР1830ВЕ31, КР1830ВЕ51 на рис. 3.49—3.63.

При эксплуатации микросхем необходимо принимать меры для защиты от воздействия статического электричества. Допустимое значение статического потенциала — 200 В.

Установку и извлечение микросхем из контактных приспособлений при ремонте аппаратуры необходимо производить при отсутствии напряжений на выводах микросхем. Особенно внимательно необходимо устанавливать в контактные приспособления микросхемы КМ1816ВЕ48, КМ1816ВЕ751 при программировании, так как напряжения, используемые при программировании, могут вывести из строя неправильно установленную микросхему.

В непосредственной близости (не более 50 мм) от микросхем рекомендуется устанавливать по цепям питания фильтрующие конденсаторы типа КМ, емкостью 0,022 — 0,15 мкФ.

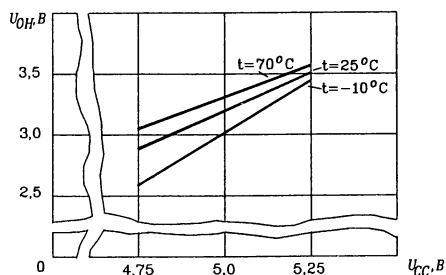


Рис. 3.1. Типовая зависимость $U_{OH}=F(U_{CC})$

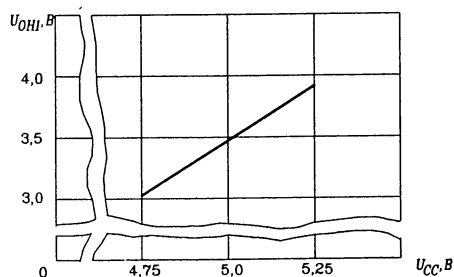


Рис. 3.2. Типовая зависимость $U_{OH}=F(U_{CC})$ при $t=70\text{ }^{\circ}\text{C}$, $25\text{ }^{\circ}\text{C}$, $-10\text{ }^{\circ}\text{C}$

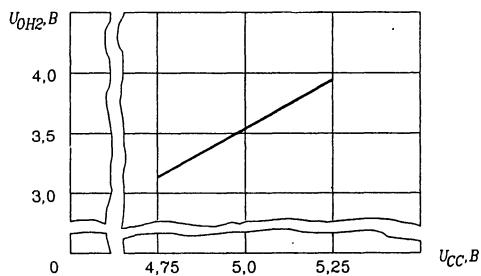


Рис. 3.3. Типовая зависимость $U_{OH2}=F(U_{CC})$ при $t=70^{\circ}\text{C}$, 25°C , -10°C

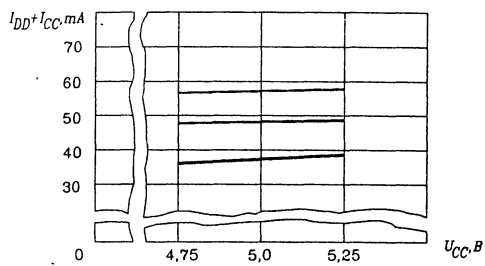


Рис. 3.7. Типовая зависимость $I_{DD}+I_{CC}=F(U_{CC})$

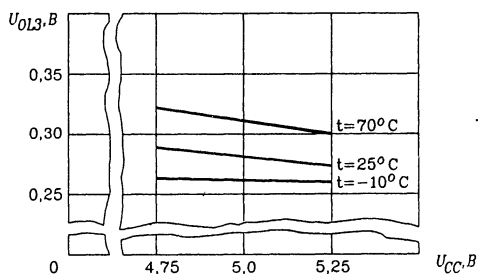


Рис. 3.4. Типовая зависимость $U_{OL3}=F(U_{CC})$

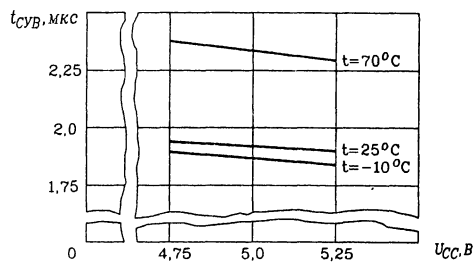


Рис. 3.8. Типовая зависимость $t_{cyb}=F(U_{CC})$

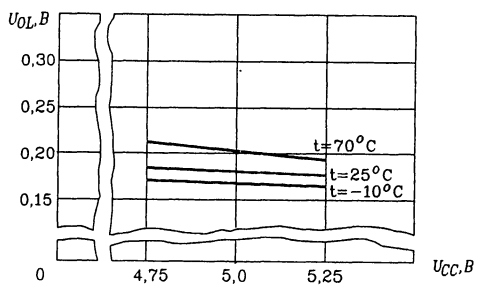


Рис. 3.5. Типовая зависимость $U_{OL}=F(U_{CC})$

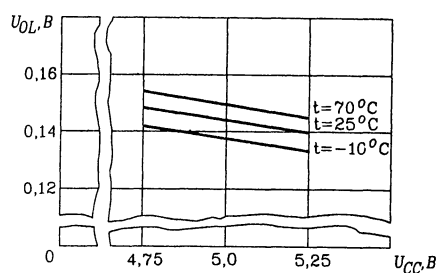


Рис. 3.9. Типовая зависимость $U_{OL}=F(U_{CC})$

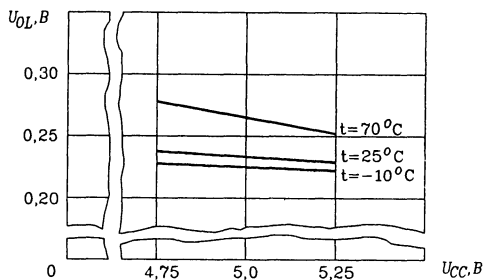


Рис. 3.6. Типовая зависимость $U_{OL}=F(U_{CC})$

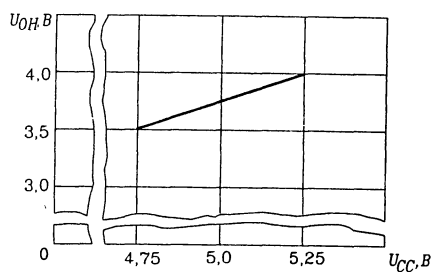


Рис. 3.10. Типовая зависимость $U_{OH}=F(U_{CC})$ при $t=70^{\circ}\text{C}$, 25°C , -10°C

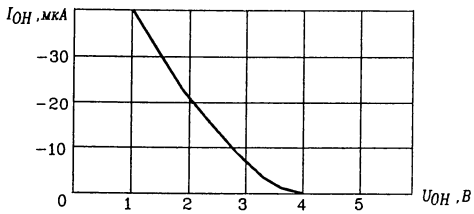


Рис. 3.11. Типовая зависимость $I_{OH}=F(U_{OH})$ при $t=25^\circ C$

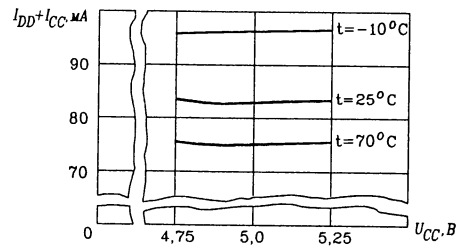


Рис. 3.15. Типовая зависимость $I_{DD}+I_{CC}=F(U_{CC})$

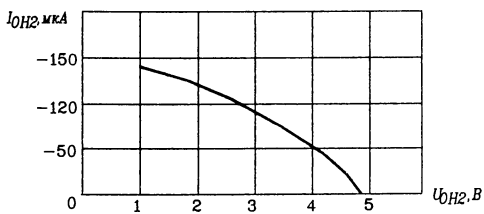


Рис. 3.12. Типовая зависимость $I_{OH2}=F(U_{OH2})$ при $t=25^\circ C$

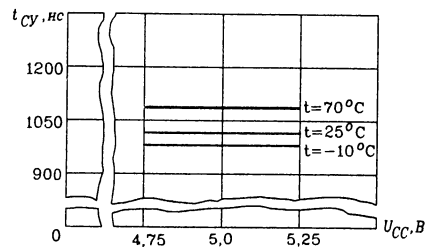


Рис. 3.16. Типовая зависимость $t_{CY}=F(U_{CC})$ при $f=11\text{ МГц}$

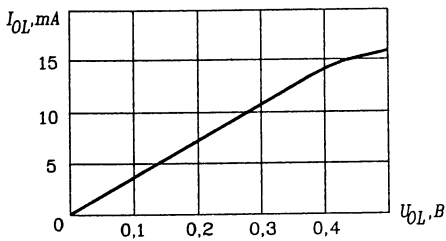


Рис. 3.13. Типовая зависимость $I_{OL}=F(U_{OL})$ при $t=25^\circ C$

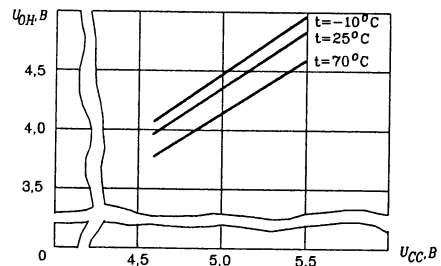


Рис. 3.17. Типовая зависимость $U_{OH}=F(U_{CC})$

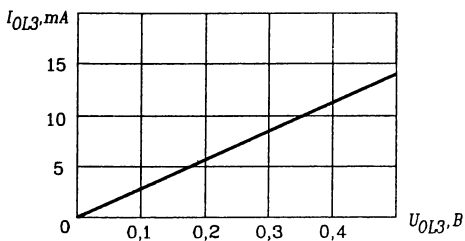


Рис. 3.14. Типовая зависимость $I_{OL3}=F(U_{OL3})$ при $t=25^\circ C$

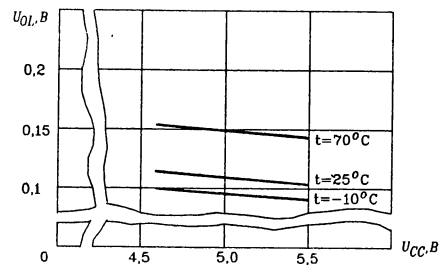


Рис. 3.18. Типовая зависимость $U_{OL}=F(U_{CC})$

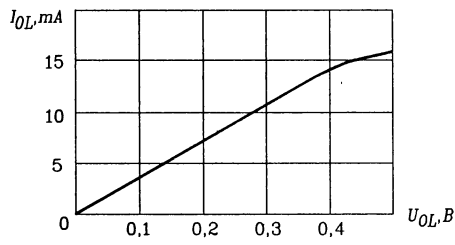


Рис. 3.19. Типовая зависимость $I_{OL}=F(U_{OL})$ при $t=25^\circ\text{C}$

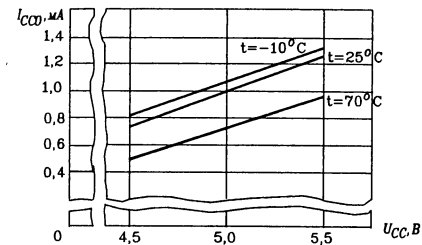


Рис. 3.23. Типовая зависимость $I_{CC0}=F(U_{CC})$ при $f=1\text{ МГц}$

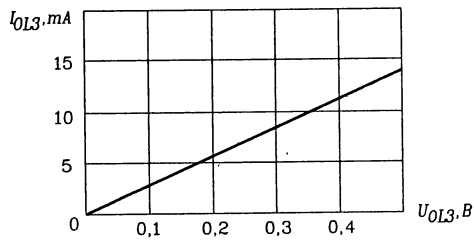


Рис. 3.20. Типовая зависимость $I_{OL3}=F(U_{OL3})$ при $t=25^\circ\text{C}$

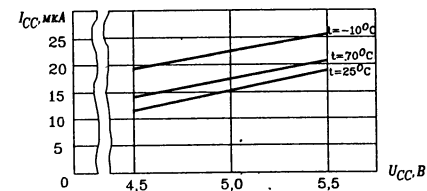


Рис. 3.24. Типовая зависимость $I_{CC}=F(U_{CC})$

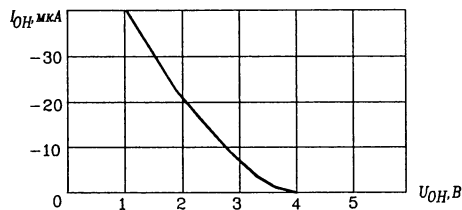


Рис. 3.21. Типовая зависимость $I_{OH}=F(U_{OH})$ при $t=25^\circ\text{C}$

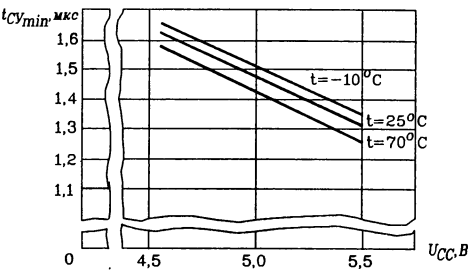


Рис. 3.25. Типовая зависимость $t_{cymin}=F(U_{CC})$

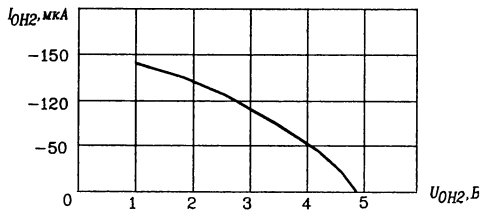


Рис. 3.22. Типовая зависимость $I_{OH2}=F(U_{OH2})$ при $t=25^\circ\text{C}$

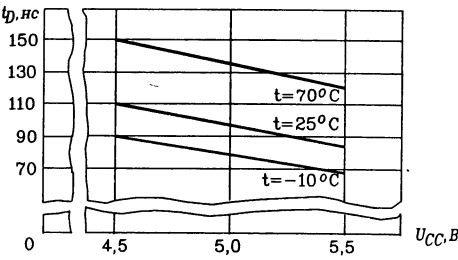


Рис. 3.26. Типовая зависимость $t_{D(ALE,HL-ADB,HZ/LZ)}=F(U_{CC})$

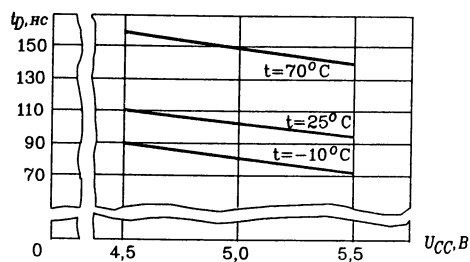


Рис. 3.27. Типовая зависимость $t_{D(WR, LH-DB, HZ/LZ)} = F(U_{CC})$

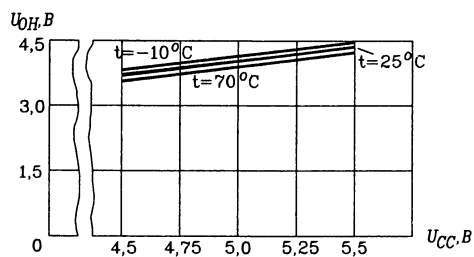


Рис. 3.31. Типовая зависимость $I_{OH} = F(U_{CC})$

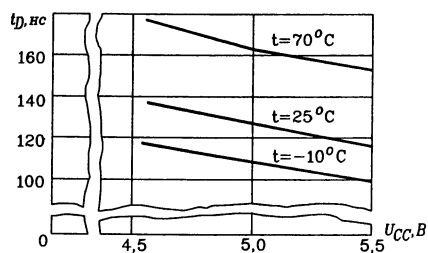


Рис. 3.28. Типовая зависимость $t_{D(ADB, ZH/ZL-ALE, HL)} = F(U_{CC})$

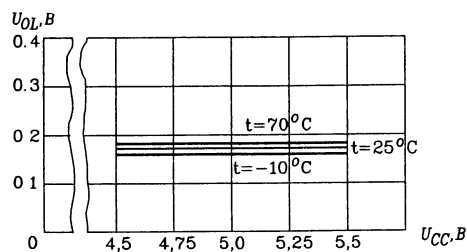


Рис. 3.32. Типовая зависимость $U_{OL} = F(U_{CC})$

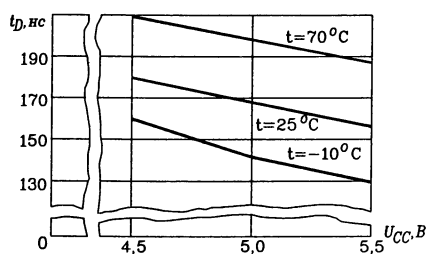


Рис. 3.29. Типовая зависимость $t_{D(ADB, ZH/ZL-WR, HL)} = F(U_{CC})$

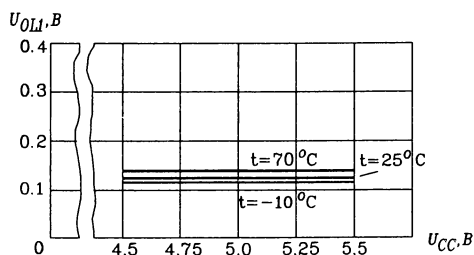


Рис. 3.33. Типовая зависимость $U_{OL} = F(U_{CC})$

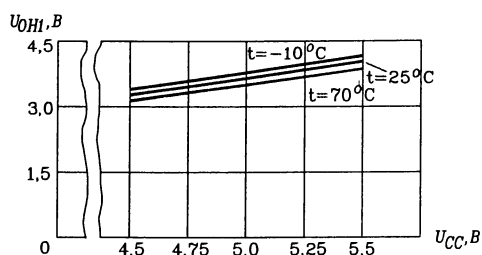


Рис. 3.30. Типовая зависимость $U_{OH} = F(U_{CC})$

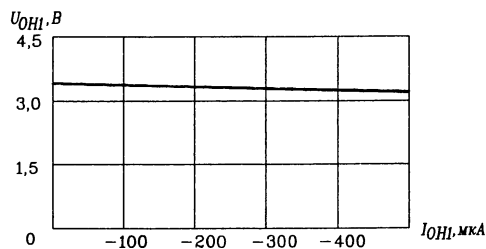


Рис. 3.34. Типовая зависимость $U_{OH} = F(I_{OH})$ при $U_{CC} = 4.5$ В, $t = 25^\circ\text{C}$

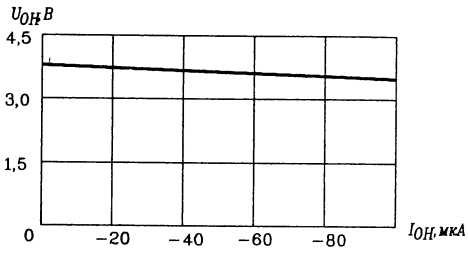


Рис. 3.35. Типовая зависимость $U_{OH}=F(I_{OH})$ при $U_{CC}=4,5$ В, $t=25$ °C

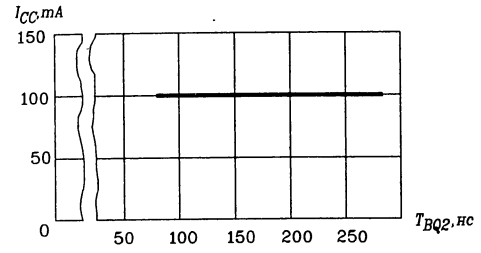


Рис. 3.39. Типовая зависимость $I_{CC}=F(T_{BQ2})$ при $U_{CC}=5,5$ В, $t=25$ °C

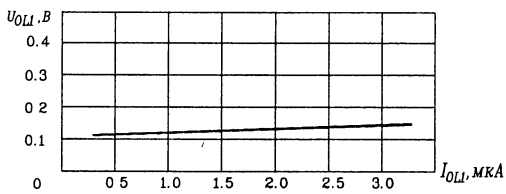


Рис. 3.36. Типовая зависимость $U_{OL}=F(I_{OL})$ при $U_{CC}=4,5$ В, $t=25$ °C

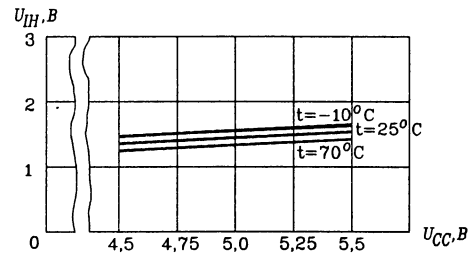


Рис. 3.40. Типовая зависимость $U_{IH}=F(U_{CC})$

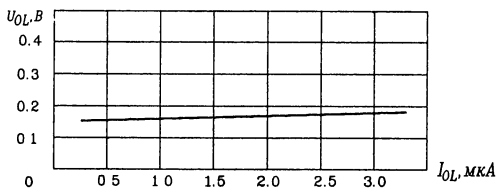


Рис. 3.37. Типовая зависимость $U_{OL}=F(I_{OL})$ при $U_{CC}=4,5$ В, $t=25$ °C

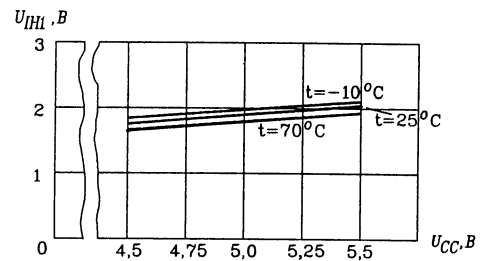


Рис. 3.41. Типовая зависимость $U_{IH1}=F(U_{CC})$

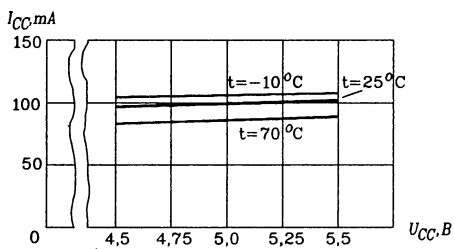


Рис. 3.38. Типовая зависимость $I_{CC}=F(U_{CC})$ при $T_{BQ2}=280$ нс

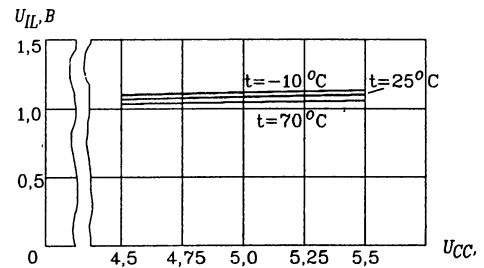


Рис. 3.42. Типовая зависимость $U_{IL}=F(U_{CC})$

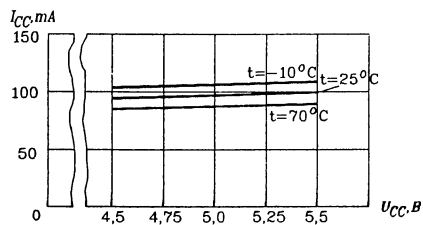


Рис. 3.43. Типовая зависимость $I_{CC}=F(U_{CC})$ при $T_{BQ2}=80$ нс

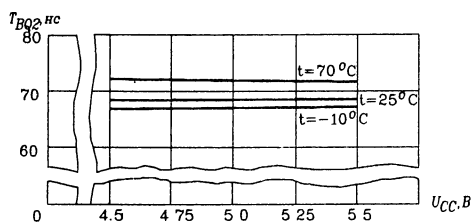


Рис. 3.44. Типовая зависимость $T_{BQ2}=F(U_{CC})$

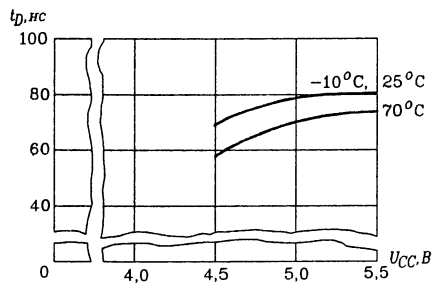


Рис. 3.45. Типовая зависимость $t_{D(A,ZH/ZL-ALE,HL)}=F(U_{CC})$

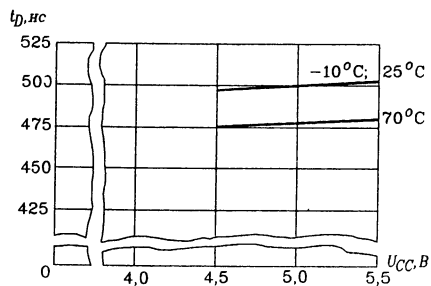


Рис. 3.46. Типовая зависимость $t_{D(D,LH/HL-WR,LH)}=F(U_{CC})$

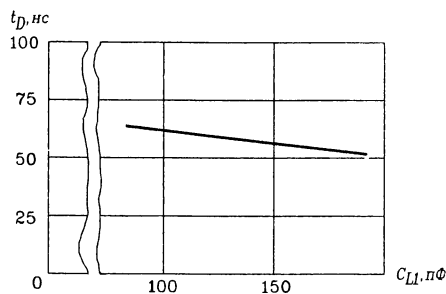


Рис. 3.47. Типовая зависимость $t_{D(A,ZH/ZL-ALE,HL)}=F(C_{L1})$

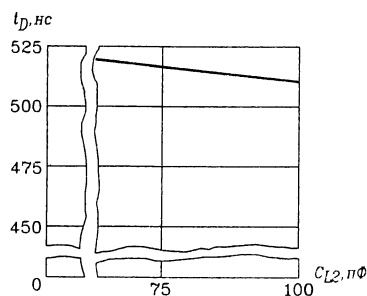


Рис. 3.48. Типовая зависимость $t_{D(D,LH/HL-WR,LH)}=F(C_{L2})$

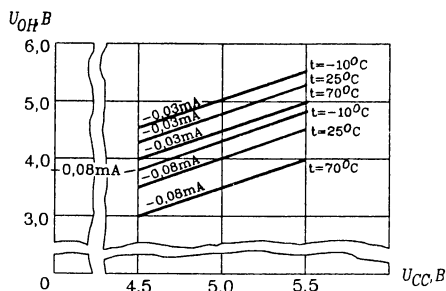


Рис. 3.49. Типовая зависимость $U_{OH}=F(U_{CC}, I_{OH}, t^\circ)$

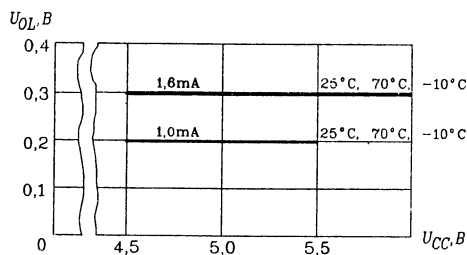


Рис. 3.50. Типовая зависимость $U_{OL}=F(U_{CC}, I_{OL}, t^\circ)$

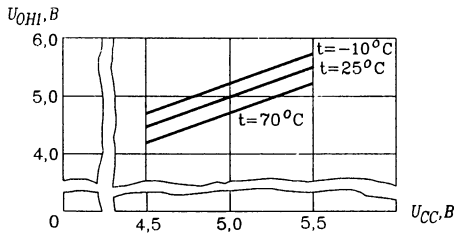


Рис. 3.51. Типовая зависимость $U_{OH1}=F(U_{CC}, t^\circ)$

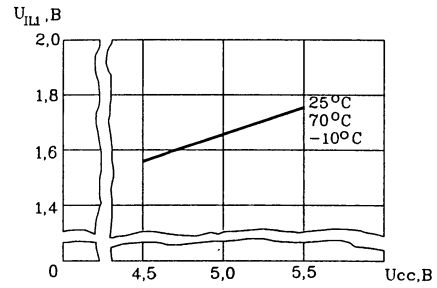


Рис. 3.55. Типовая зависимость $U_{OL1}=F(U_{CC}, t^\circ)$

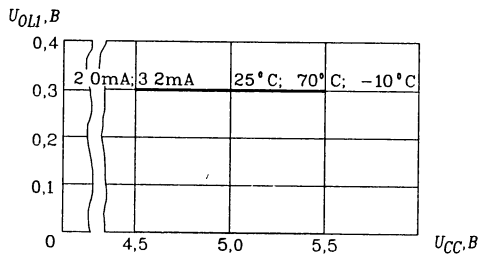


Рис. 3.52. Типовая зависимость $U_{OL1}=F(U_{CC}, I_{OL}, t^\circ)$

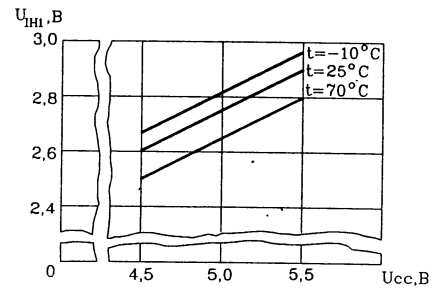


Рис. 3.56. Типовая зависимость $U_{OH1}=F(U_{CC}, t^\circ)$

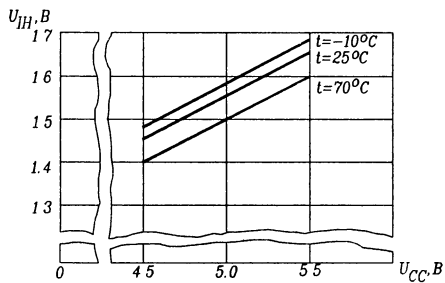


Рис. 3.53. Типовая зависимость $U_{OH1}=F(U_{CC}, t^\circ)$

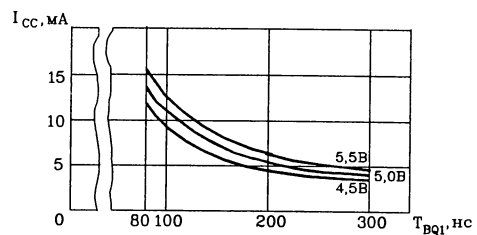


Рис. 3.57. Типовая зависимость $I_{CC}=F(U_{CC}, U_{BE1})$ при $t=25^\circ\text{C}$

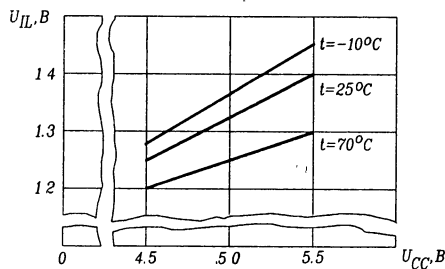


Рис. 3.54. Типовая зависимость $U_{OL1}=F(U_{CC}, t^\circ)$

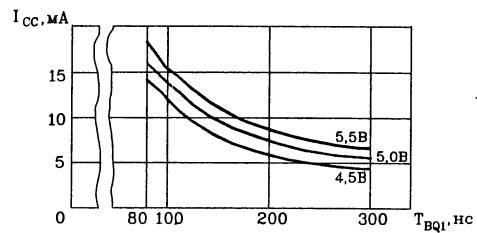


Рис. 3.58. Типовая зависимость $I_{CC}=F(U_{CC}, U_{BE1})$ в диапазоне температур от минус 10°C до 70°C

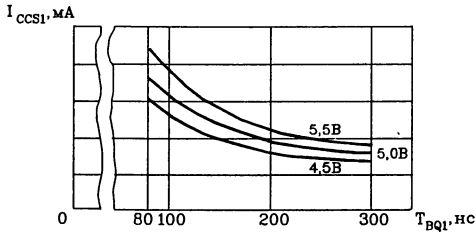


Рис. 3.59. Типовая зависимость $I_{CCS1}=F(U_{CC}, T_{BQ1})$ в диапазоне температур от минус 10°C до 70°C

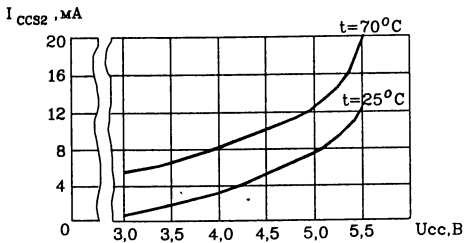


Рис. 3.60. Типовая зависимость $I_{CCS2}=F(U_{CC}, t^{\circ})$

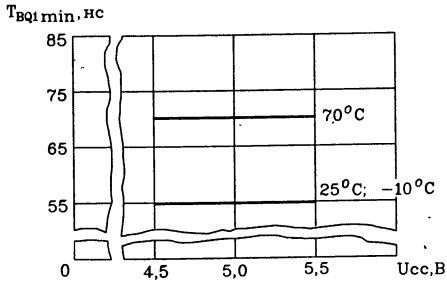


Рис. 3.61. Типовая зависимость $T_{BQ1min}=F(U_{CC}, t^{\circ})$

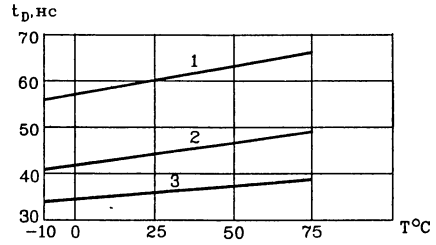


Рис. 3.62. Типовые зависимости при $U_{CC}=4,5$ В, $C_{L1}=100$ нФ:

1. $t_{D(P0,ZH/ZL)}=F(t^{\circ})$
2. $t_{D(P0,ZH1/ZL1)}=F(t^{\circ})$
3. $t_{D(P0H1L2/L1H2)}=F(t^{\circ})$

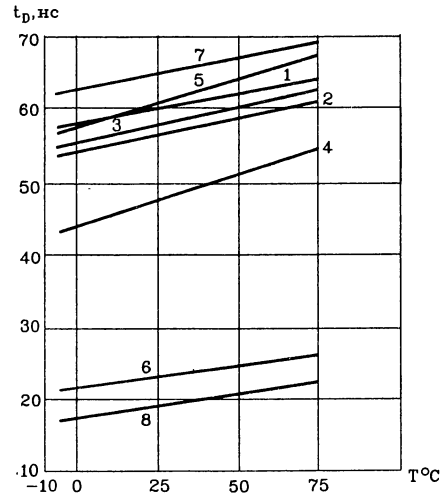


Рис. 3.63. Типовые зависимости при $U_{CC}=4,5$ В, $C_{L1}=100$ нФ, $C_{L2}=80$ нФ:

1. $t_{D(RD,HL)}=F(t^{\circ})$
2. $t_{D(PME,HL)}=F(t^{\circ})$
3. $t_{D(WR,HL)}=F(t^{\circ})$
4. $t_{D(ALE,HL)}=F(t^{\circ})$
5. $t_{D(RD,LH)}=F(t^{\circ})$
6. $t_{D(PME,LH)}=F(t^{\circ})$
7. $t_{D(WR,LH)}=F(t^{\circ})$
8. $t_{D(ALE,LH)}=F(t^{\circ})$

ГЛАВА 4

КМОП ОМЭВМ КМ1830ВЕ751/КМ1830ВЕ753

Микросхемы однокристалльных микроЭВМ КМ1830ВЕ751/КМ1830ВЕ753 принадлежат к семейству МК51 и выполнены по высококачественной КМОП технологии.

Микросхемы КМ1830ВЕ753 идентичны микросхемам КМ1830ВЕ751. Отличием является только емкость ППЗУ: КМ1830ВЕ753 содержат на кристалле ППЗУ с ультрафиолетовым стиранием емкостью 8 Кбайт, а КМ1830ВЕ751 — 4 Кбайт. Ближайшим зарубежным аналогом КМ1830ВЕ753 является микросхема 8753Н фирмы AMD, выполненная по nМОП технологии.

Корпус и назначение выводов микросхем КМ1830ВЕ751/КМ1830ВЕ753 те же, что и у описанных в разделе 2 микросхем КМ1816ВЕ751.

Поскольку микросхемы КМ1830ВЕ751/КМ1830ВЕ753 во всем, кроме ППЗУ идентичны, то в дальнейшем приводится их единое описание с указанием в нужных местах различий между ними.

В данном разделе приводятся только отличия микросхем КМ1830ВЕ751/КМ1830ВЕ753 от микросхем КР1830ВЕ51, подробное описание которых содержится в разделе 2 и которым можно пользоваться при работе с КМ1830ВЕ751/КМ1830ВЕ753 с учетом рассматриваемых ниже дополнений и изменений.

Микросхемы КМ1830ВЕ751/КМ1830ВЕ753 имеют следующие особенности:

- специальный режим эмуляции (ONCE MODE, "on-circuit emulation");
- дополнительные средства защиты памяти программ, расположенной на кристалле: два бита защиты памяти и шифровальную таблицу;
- алгоритм программирования укороченными импульсами.

4.1. Специальный режим эмуляции

Данный режим позволяет выполнять тестирование и отладку систем, использующих КМ1830ВЕ751/КМ1830ВЕ753, без удаления последних из платы. Для активизации режима эмуляции необходимо выполнить следующую последовательность действий:

1. При высоких логических уровнях на RST и \overline{PME} подать уровень лог. 0 на вывод ALE.

2. Удерживать ALE в лог. 0 при деактивации сигнала сброса на RST (подаче на RST уровня лог. 0), после чего перестать удерживать ALE в состоянии лог. 0.

Во время нахождения микросхемы в режиме эмуляции выходы порта P0 переходят в высокоимпедансное состояние, а выходы портов P1—P3 и выходы ALE, \overline{PME} находятся в состоянии лог. 1, которое обеспечивается с помощью внутренних высокоомных резисторов. Тракт генератора остается активным. В то время, когда микросхема КМ1830ВЕ751/КМ1830ВЕ753 находится в режиме эмуляции, для управления системой можно использовать внешний эмулятор или тестовый процессор. Нормальная работы схемы возобновляется после подачи нормального сигнала сброса на вход RST.

4.2. Защита внутренней памяти программ

Микросхемы КМ1830ВЕ751/КМ1830ВЕ753 содержат два механизма защиты внутренней памяти программ от несанкционированного доступа извне: проверку зашифрованного содержимого памяти программ и биты защиты памяти программ.

4.2.1. Шифровальная таблица

КМ1830ВЕ751/КМ1830ВЕ753 имеют 32-байтную шифровальную таблицу, располагающуюся во внутреннем УФППЗУ и программируемую пользователем. Эта таблица (или область памяти) может использоваться для шифрования байтов внутренней памяти программ, выбираемых внешними средствами из УФППЗУ во время проверки его содержимого.

Всякий раз, когда в режиме проверки адресуется байт внутреннего УФППЗУ, 5 бит адреса используются для адресации шифровальной таблицы. Появляющийся на внешних выводах микросхемы байт является результатом выполнения операции "ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ" над байтом, выбранным из УФППЗУ программ и байтом, выбранным из шифровальной таблицы. Зная содержимое шифровальной таблицы, можно выполнить дешифрацию и получить истинное содержимое внутренней памяти программ.

Незапрограммированные байты шифровальной таблицы содержат FFH. Таким образом, если шифровальную таблицу оставить незапрограммированной, то в режиме проверки внутренне УФППЗУ будет считываться истинное значение его содержимого. Заметим попутно, что незапрограммированные байты УФППЗУ внутренней памяти программ также содержат FFH.

4.2.2. Биты защиты памяти программ

Кроме шифровальной таблицы микросхемы КМ1830ВЕ751/КМ1830ВЕ753 содержат на кристалле два бита защиты внутреннего УФППЗУ, каждый из которых может быть запрограммирован (З) или оставлен незапрограммированным (Н) для получения следующих дополнительных свойств:

Бит 1	Бит 2	Дополнительные свойства
Н	Н	нет дополнительных свойств
З	Н	– команда, выбранная из внешней памяти, не имеет доступа к внутренней памяти программ (актуально для команды MOVС); – дальнейшее программирование невозможно
Н	З	– зарезервировано под дальнейшее использование
З	З	– команда, выбранная из внешней памяти, не имеет доступа к внутренней памяти программ (актуально для MOVС); – дальнейшее программирование невозможно; – режим проверки внутренней памяти программ запрещен

Если Бит 1 запрограммирован, логический уровень на выводе DEMA опрашивается и фиксируется во внутренней защелке во время сброса. Если питание микросхемы включается без формирования сигнала сброса, запоминается случайное значение и сохраняется до активизации сигнала сброса. Для правильной работы микросхемы необходимо, чтобы значение логического уровня на выводе DEMA совпадало с состоянием защелки.

4.3. Программирование

Микросхемы КМ1830ВЕ751/КМ1830ВЕ753 программируются с помощью специального алгоритма ("Алгоритм программирования укороченными импульсами"), отличающегося от более ранних модификаций величиной программирующего напряжения U_{PR} , а также длительностью и количеством программирующих импульсов на выводе ALE/PR.

В табл. 4.1 показаны логические уровни на выводах микросхем при программировании памяти программ, шифровальной таблицы и бит защиты памяти.

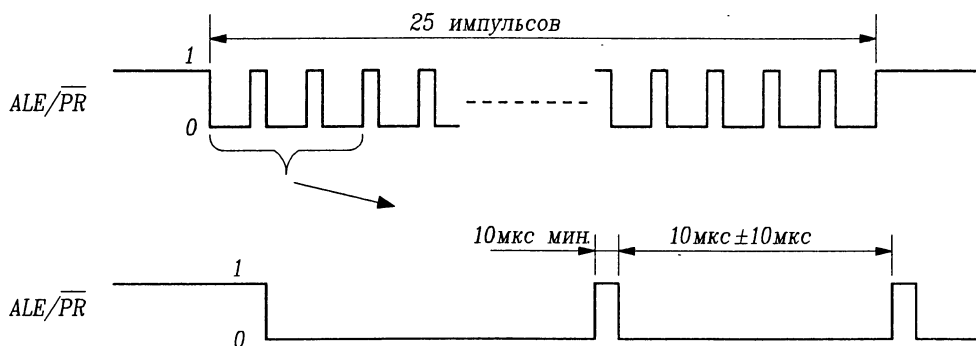


Рис. 4.2. Диаграмма импульсов программирования

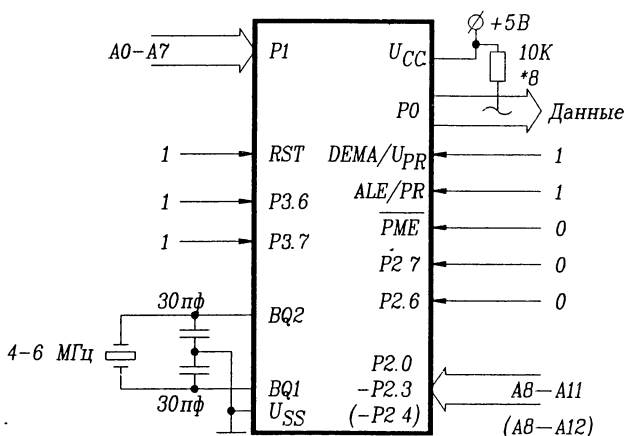


Рис. 4.3. Подключение микросхем в режиме проверки внутренней памяти программ

Временные диаграммы работы микросхем КМ1830ВЕ751, КМ1830ВЕ753 при записи и считывании внутренней памяти программ полностью соответствуют диаграммам, приведенным на рис. 2.16а в разделе 2 настоящего описания для микросхем КМ1816ВЕ751, с той лишь разницей, что один программирующий импульс на выводе ALE/ \overline{PR} заменяется серией из 25 импульсов в соответствии с рис. 4.2. Все временные параметры, кроме $t_{W(PR,L)}$, в рассматриваемом режиме работы для КМ1830ВЕ751, КМ1830ВЕ753 идентичны приведенным в табл. 2.25 для КМ1816ВЕ751. На вход DEMA/ U_{PR} при программировании микросхем КМ1830ВЕ751, КМ1830ВЕ753 подается $U_{PR}=12,75 \text{ В} \pm 0,25 \text{ В}$.

Работа в режимах записи и считывания (проверки) внутренней памяти программ возможна только при наличии тактовой синхронизации (4—6 МГц), которая необходима потому, что в данных режимах микросхемы осуществляют внутренние передачи адресов и данных.

Адрес ячейки УФППЗУ программ, которую необходимо запрограммировать, подается на выводы портов P1 и P2, как показано на рис. 4.1. Байт, который требуется записать в адресуемую ячейку УФППЗУ, подается на выводы порта P0. Состояния выводов RST, \overline{PME} , выводов портов P2 и P3, должно соответствовать состояниям для этих выводов, приведенным в табл. 4.1 для режима "Программирование памяти программ". После того, как состояния всех выводов микросхемы установлены, на вывод ALE/ \overline{PR} подаются 25 импульсов низкого уровня в соответствии с диаграммами на рис. 4.2.

Для программирования шифровальной таблицы используется тот же алгоритм, что и для программирования УФППЗУ программ, но выводы микросхемы должны находиться в состояниях, приведенных в табл. 4.1 для режима "Программирование шифровальной таблицы". При этом шифровальная таблица расположена в адресном пространстве от 0 до 1FH. Не следует забывать, что после того, как шифровальная таблица запрограммирована, в режиме проверки из микросхемы будут считываться только зашифрованные данные.

Для программирования бита защиты памяти программ используется тот же алгоритм, что и для программирования УФППЗУ программ и ячеек шифровальной таблицы, но выводы микросхемы должны находиться в состояниях, приведенных в табл. 4.1 для режимов "Программирование Бита 1 защиты памяти программ" или "Программирование Бита 2 защиты памяти программ". После того, как один из бит защиты памяти программ запрограммирован, дальнейшее программирование внутренней памяти программ и шифровальной таблицы становится невозможным. Однако, при этом другой бит защиты памяти может быть запрограммирован. В отличие от микросхем KM1816BE751, для KM1830BE751, KM1830BE753 программирование бита защиты памяти не влияет на их способность работать с внешней памятью программ.

Необходимо отметить, что вход DEMA/U_{PR} очень чувствителен к превышениям максимально допустимого напряжения программирования U_{PR}=13 В. Даже "иголки" выше указанного уровня могут привести к необратимому отказу микросхемы.

4.4. Чтение внутренней памяти программ

Если Бит 2 защиты памяти программ не запрограммирован, расположенная на кристалле внутренняя память программ может быть считана внешними по отношению к микросхеме однокристалльной микроЭВМ средствами. Адрес считываемой ячейки УФППЗУ подается на выводы портов P1 и P2, как показано на рис. 4.3. Остальные выводы микросхемы необходимо установить в состояния, приведенные в табл. 4.1 для режима "Проверка памяти программ". Содержимое адресуемой ячейки УФППЗУ программ считывается с выводов порта P0. При этом на выводах порта P0 необходимы внешние подтягивающие резисторы на +5 В.

Если шифровальная таблица запрограммирована, данные, считываемые с выводов порта P0, являются результатом операции "ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ" над адресуемым байтом УФППЗУ программ и адресуемым байтом шифровальной таблицы (адресуется пятью младшими разрядами адреса). Пользователь должен знать содержимое шифровальной таблицы для правильного декодирования содержимого внутренней памяти программ. Непосредственно содержимое шифровальной таблицы не может быть прочитано.

Все приведенные в табл. 4.1 режимы допустимы как по одиночке, так и в произвольной последовательности.

4.5. Стирание УФППЗУ

Стирание содержимого внутреннего УФППЗУ начинается при облучении кристалла светом с длиной волны менее 4000 Ангстрем. Поскольку солнечный свет и лампы дневного света содержат волны в указанном диапазоне, облучение кристалла этими источниками в течение длительного времени (около 1 недели под солнцем или около 3 лет в комнате с лампами дневного света) может вызвать искажение содержащейся в УФППЗУ информации. Если эксплуатация микросхем предполагает возможность такого облучения, необходимо закрывать кварцевое окошко на корпусе микросхемы каким-либо светонепроницаемым экраном.

Для стирания УФППЗУ рекомендуется облучение микросхемы ультрафиолетовым излучением (длина волны около 2537 Ангстрем) с интегральной дозой по меньшей мере 15 Вт-сек/см². Облучение УФППЗУ под ультрафиолетовой лампой с потоком 12000 мкВт/см² с расстояния 2,5 см в течение 30 минут, как правило, является достаточным.

После стирания все ячейки УФППЗУ, включая область шифровальной таблицы, находятся в состоянии лог. 1, а биты защиты памяти — в незапрограммированном состоянии.

4.6. Электрические параметры

В табл. 4.2 приведены статические параметры микросхем КМ1830ВЕ751, КМ1830ВЕ753. Необходимо отметить, что на момент подготовки данного описания для микросхем КМ1830ВЕ751, КМ1830ВЕ753 еще не были выполнены в полном объеме все измерения электрических параметров. Вследствие этого параметры приводятся по микросхеме 87С51 ф. Intel. Параметры микросхем КМ1830ВЕ751, КМ1830ВЕ753 могут незначительно отличаться от американского аналога.

Что касается динамических параметров, то для рассматриваемых микросхем можно использовать информацию, приведенную в разделе 2 для микросхем КР1830ВЕ51.

Таблица 4.2. Статические параметры в диапазоне температур от 0°C до +70°C и $U_{CC}=5\text{ В}\pm 10\%$

Обозн.	Параметр	Мин.	Тип ⁽¹⁾	Макс.	Ед. изм.	Примечание
U_{IL}	Входное напряжение низкого уровня (кроме DEMA)	-0,5	-	$0,2U_{CC}-0,1$	В	-
U_{IL1}	Входное напряжение низкого уровня для вывода DEMA	0	-	$0,2U_{CC}-0,3$	В	-
U_{IH}	Входное напряжение высокого уровня (кроме BQ1, RST)	$0,2U_{CC}+0,9$	-	$U_{CC}+0,5$	В	-
U_{IH1}	Входное напряжение высокого уровня для выводов BQ1, RST	$0,7U_{CC}$	-	$U_{CC}+0,5$	В	-
U_{OL}	Выходное напряжение низкого уровня (порты P1, P2, P3) ⁽⁷⁾	-	-	0,45	В	$I_{OL}=1,6\text{ мА}^{(2)}$
U_{OL1}	Выходное напряжение низкого уровня (порт P0; ALE; PWE) ⁽⁷⁾	-	-	0,45	В	$I_{OL}=3,2\text{ мА}^{(2)}$
U_{OH}	Выходное напряжение высокого уровня (порты P1, P2, P3; ALE, PWE)	2,4	-	-	В	$I_{OH}=-60\text{ мкА}$
		$0,75U_{CC}$	-	-	В	$I_{OH}=-25\text{ мкА}$
		$0,9U_{CC}$	-	-	В	$I_{OH}=-10\text{ мкА}$
U_{OH1}	Выходное напряжение высокого уровня (порт P0 в режиме внешней шины)	2,4	-	-	В	$I_{OH}=-800\text{ мкА}$
		$0,75U_{CC}$	-	-	В	$I_{OH}=-300\text{ мкА}$
		$0,9U_{CC}$	-	-	В	$I_{OH}=-80\text{ мкА}^{(3)}$
I_{IL}	Входной ток логического нуля (порты P1, P2, P3)	-	-	-50	мкА	$U_I=0,45\text{ В}$
I_{TL}	Ток перехода "лог. 1 – лог. 0" (порты P1, P2, P3)	-	-	-650	мкА	(4)
I_{LI}	Входной ток утечки (порт P0)	-	-	± 10	мкА	$U_I=U_{IL}$ или U_{IH}
R_{RST}	Внутренний резистор между выводом RST и выводом 0 В	50	-	300	кОм	-
I_{CC}	Ток потребления	-	-	-	-	(6)
	Активный режим, 12 МГц ⁽⁵⁾	-	11,5	25	мА	
	Idle режим, 12 МГц ⁽⁵⁾	-	1,3	4	мА	
	Режим микropотребления	-	3	50	мкА	

Продолжение таблицы 4.2

Обозн.	Параметр	Мин.	Тип ⁽¹⁾	Макс.	Ед. изм.	Примечание
C _{IO}	Емкость выводов	–	–	10	пФ	–
C _L	Емкость нагрузки для выводов порта P0, ALE и PME	–	–	100	пФ	–
C _{I1}	Емкость нагрузки (кроме выводов P0, ALE, PME)	–	–	80	пФ	–

Примечания к табл. 4.2:

1. Типовые значения параметров (колонка "Тип" в таблице) основаны на измерении ограниченного числа образцов приборов, взятых из ранее произведенных партий, и не гарантируются. Приведенные типовые значения получены при комнатной температуре и $U_{CC}=5$ В.

2. Емкостная нагрузка на выводах портов P0 и P2 может стать причиной появления ложных шумовых импульсов, особенно сильно влияющих на напряжения U_{OL} , U_{OL1} выводов ALE и выводов портов P1 и P3. Шум появляется вследствие разряда емкостей внешней шины на выводы портов P0 и P2 при изменении состояния на них из лог. 1 в лог. 0 во время операций на шине. В худшем случае (емкостная нагрузка больше 100 пФ), шумовые импульсы на выводе ALE могут превысить 0,8 В. В таких случаях желательно пропустить сигнал ALE через триггер Шмитта или использовать защелку адреса с триггером Шмитта на стробирующем входе.

3. Емкостная нагрузка на выводах портов P0 и P2 может вызвать кратковременное падение напряжения U_{OH} на выводах ALE и PME ниже $0,9U_{CC}$ во время стабилизации бит адреса.

4. Выводы портов P1, P2, P3 становятся источниками тока перехода, когда состояние на них меняется внешними средствами из лог. 1 в лог. 0. Ток перехода достигает максимума при входном напряжении 2 В.

5. I_{CCMAX} на других частотах вычисляется:

Активный режим: $I_{CCMAX}=0,94FREQ+13,71$

Idle режим: $I_{CCMAX}=0,14FREQ+2,31$,

где FREQ — частота синхронизации в МГц. I_{CCMAX} дан в мА. См. рис. 4.4.

6. Минимальное напряжение питания в режиме микропотребления — $U_{CC}=2$ В.

7. В статическом состоянии ток I_{OL} должен быть внешне ограничен следующим образом:

Максимальный I_{OL} на один вывод — 10 мА

Максимальный I_{OL} на восьмьбитный порт:

Порт P0 — 26 мА

Порты P1, P2 и P3 — 15 мА

Максимальный общий I_{OL} на все выводы, работающие в режиме выхода — 71 мА

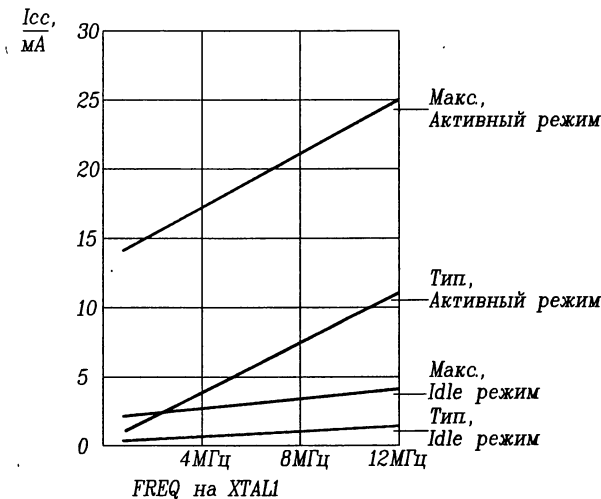


Рис. 4.4. Зависимость тока потребления I_{CC} от частоты синхронизации FREQ

ГЛАВА 5

ХАРАКТЕРИСТИКИ АНАЛОГОВ ФИРМЫ INTEL

В данном разделе приводятся статические параметры микросхем семейства MCS-51 фирмы Intel: 8031АН/8051АН/8751Н, 80С31ВН/80С51ВН. Во всем остальном, включая временные диаграммы и динамические параметры, при работе с указанными микросхемами можно в полном объеме использовать описание соответственно микросхем КР1816ВЕ31/КР1816ВЕ51/КМ1816ВЕ751 и КР1830ВЕ31/КР1830ВЕ51.

Статические параметры, приведенные в разделе 4 настоящего описания для микросхем КМ1830ВЕ751, полностью соответствуют аналогичным параметрам 87С51 фирмы Intel. При работе с последними можно в полном объеме использовать описание, временные параметры, приведенные для микросхем КМ1830ВЕ751.

Информация о микросхемах фирмы Intel приводится по каталогу фирмы Intel "8-BIT EMBEDDED CONTROLLER HANDBOOK" за 1991 год.

Фирма Intel улучшает и видоизменяет свои изделия. К примеру, у микросхем 87С51 по каталогу "EMBEDDED CONTROLLERS AND PROCESSORS", VOLUME 1, 1992 г. добавлен третий бит защиты памяти и расширена до 64 Кбайт шифровальная таблица. Согласно каталогу, такие модифицированные микросхемы имеют индекс "А" в конце маркировки, расположенной на верхней стороне корпуса.

Поэтому, если Вы собираетесь работать с микросхемами фирмы Intel, желательно связаться с представителями этой фирмы для уточнения характеристик имеющихся у Вас приборов.

Тем не менее хотим подчеркнуть, что приведенное в настоящем томе техническое описание в полной мере применимо для микросхем фирмы Intel 8031АН, 8051АН, 8751Н, 80С31ВН, 80С51ВН, 87С51 без индекса "А" в конце маркировки, а если не использовать указанные приборы в экстремальных режимах, то к ним также в полной мере применимы и все приведенные статические и динамические параметры.

Таблица 5.1. Соответствие обозначений выводов ОМЭВМ серий 1816, 1830 семейства МК51 и микросхем фирмы Intel семейства MCS-51, выполненных в 40-выводных корпусах типа DIP

Номер вывода	Обозначение		Номер вывода	Обозначение	
	МК51	МCS-51		МК51	МCS-51
1	P1.0	P1.0	21	P2.0	A8
2	P1.1	P1.1	22	P2.1	A9
3	P1.2	P1.2	23	P2.2	A10
4	P1.3	P1.3	24	P2.3	A11
5	P1.4	P1.4	25	P2.4	A12
6	P1.5	P1.5	26	P2.5	A13
7	P1.6	P1.6	27	P2.6	A14
8	P1.7	P1.7	28	P2.7	A15
9	RST	RST	29	PME	PSEN
10	P3.0	P3.0 RxD	30	ALE	ALE/PROG*
11	P3.1	P3.1 TxD	31	DEMA	EA/V _{pp}
12	P3.2	P3.2 INT0	32	P0.7	P0.7 AD7
13	P3.3	P3.3 INT1	33	P0.6	P0.6 AD6
14	P3.4	P3.4 T0	34	P0.5	P0.5 AD5
15	P3.5	P3.5 T1	35	P0.4	P0.4 AD4
16	P3.6	P3.6 WR	36	P0.3	P0.3 AD3
17	P3.7	P3.7 RD	37	P0.2	P0.2 AD2
18	BQ2	XTAL2	38	P0.1	P0.1 AD1
19	BQ1	XTAL1	39	P0.0	P0.0 AD0
20	0B	VSS	40	UCC	VCC

* — Только для микросхем, имеющих внутреннее УФППЗУ (EPROM)

Таблица 5.2. Статические параметры микросхем 8031АН/8051АН, 8751Н фирмы Intel при температуре окружающей среды $T_A=0^{\circ}\text{C} \rightarrow +70^{\circ}\text{C}$; $V_{CC}=5\text{В} \pm 10\%$; $V_{SS}=0\text{ В}$

Обозначение	Параметр	Мин.	Макс.	Единица измерения	Режим измерения
V_{IL}	Входное напряжение низкого уровня (кроме вывода \overline{EA} для 8751Н)	-0,5	0,8	В	
V_{IL1}	Входное напряжение низкого уровня вывода \overline{EA} для 8751Н	0	0,7	В	
V_{IH}	Входное напряжение высокого уровня (кроме XTAL2; RST)	2,0	$V_{CC}+0,5$	В	
V_{IH1}	Входное напряжение высокого уровня для XTAL2; RST	2,5	$V_{CC}+0,5$	В	XTAL1= V_{SS}
V_{OL}	Выходное напряжение низкого уровня (порты 1, 2, 3)		0,45	В	$I_{OL}=1,6\text{ мА}$
V_{OL1}	Выходное напряжение низкого уровня (порт 0; ALE; PSEN)				
	8751Н		0,60 0,45	В В	$I_{OL}=3,2\text{ мА}$ $I_{OL}=2,4\text{ мА}$
	8031АН/8051АН		0,45	В	$I_{OL}=3,2\text{ мА}$
V_{OH}	Выходное напряжение высокого уровня (порты 1, 2, 3; ALE, PSEN)	2,4		В	$I_{OH}=-80\text{ мкА}$
V_{OH1}	Выходное напряжение высокого уровня (порт 0 в режиме внешней шины)	2,4		В	$I_{OH}=-400\text{ мкА}$
I_{IL}	Входной ток логического нуля (порты 1, 2, 3; RST)		-500	мкА	$V_{IN}=0,45\text{ В}$
I_{IL1}	Входной ток логического нуля вывода \overline{EA} для 8751Н		-15	мА	$V_{IN}=0,45\text{ В}$
I_{IL2}	Входной ток логического нуля (XTAL2)		-3,2	мА	$V_{IN}=0,45\text{ В}$
I_{LI}	Входной ток утечки (порт 0) 8751Н 8031АН/8051АН		± 100 ± 10	мкА мкА	$0,45 < V_{IN} < V_{CC}$ $0,45 < V_{IN} < V_{CC}$
I_{IH}	Ток логической единицы вывода \overline{EA} для 8751Н		500	мкА	$V_{IN}=2,4\text{ В}$
I_{CL}	Ток потребления 8031АН/8051АН 8751Н		125 250	мА мА	Все выходы отсоединены от нагрузки; $\overline{EA}=V_{CC}$
C_{IO}	Емкость выводов		10	пФ	Частота измерен. 1МГц

Примечания к таблице 5.2:

1. Емкостная нагрузка на линиях портов 0 и 2 может вызвать паразитные шумовые импульсы, особенно сильно воздействующие на напряжения V_{OL} , V_{OL1} вывода ALE и выводов портов 1 и 3. Шум появляется из-за разряда емкостей в выводы порта 0 и порта 1 при переходе напряжений на этих выводах из "1" в "0" во время операций на внешней шине. В наихудшем случае (емкость нагрузки больше 100 пФ) шумовой импульс на выводе ALE может превышать 0,8 В. В таких случаях

желательно сигнал ALE пропустить через триггер Шмитта или использовать триггерную защелку с триггером Шмитта на стробирующем входе.

2. Рекомендуемая емкость нагрузки, не более:

для выводов порта 0, ALE и PSEN: 100 пФ

для остальных выводов: 80 пФ.

3. Предельные значения параметров:

Температура окружающей среды: 0°C—+70°C

Напряжение на выводе \overline{EA}/V_{PP} относительно V_{SS} -0,5 В—21,5 В

Напряжение на других выводах относительно V_{SS} -0,5 В—+7 В

Рассеиваемая мощность 1,5 Вт.

Превышение предельных значений параметров может вызвать отказ микросхемы.

Таблица 5.3. Статические параметры микросхем 80C31BH/80C51BH фирмы Intel при температуре окружающей среды $T_A=0^\circ\text{C}$ —+70°C; $V_{CC}=5\text{В}\pm 20\%$; $V_{SS}=0\text{ В}$

Обозн.	Параметр	Мин.	Тип. (3)	Макс.	Ед. изм.	Режим измерения
V_{IL}	Входное напряжение низкого уровня (кроме вывода \overline{EA})	-0,5		$0,2V_{CC}-0,1$	В	
V_{IL1}	Входное напряжение низкого уровня для вывода \overline{EA}	-0,5		$0,2V_{CC}-0,3$	В	
V_{IH}	Входное напряжение высокого уровня (кроме XTAL1; RST)	$0,2V_{CC}+0,9$		$V_{CC}+0,5$	В	
V_{IH1}	Входное напряжение высокого уровня для XTAL1; RST	$0,7V_{CC}$		$V_{CC}+0,5$	В	
V_{OL}	Выходное напряжение низкого уровня(6) для портов 1, 2, 3			0,45	В	$I_{OL}=1,6\text{ мА}(1)$
V_{OL1}	Выходное напряжение низкого уровня(6) для порта 0; ALE; PSEN	2,4				
V_{OH}	Выходное напряжение высокого уровня для портов 1, 2, 3; ALE, PSEN	2,4				$I_{OH}=-60\text{ мкА}; V_{CC}=5\text{ В}\pm 10\%$
		$0,75V_{CC}$				$I_{OH}=-25\text{ мкА}$
		$0,9V_{CC}$				$I_{OH}=-10\text{ мкА}$
V_{OH1}	Выходное напряжение высокого уровня для порта 0 в режиме внешней шины	2,4			В	$I_{OH}=-800\text{ мкА}; V_{CC}=5\text{ В}\pm 10\%$
		$0,75V_{CC}$				$I_{OH}=-300\text{ мкА}$
		$0,9V_{CC}$				$I_{OH}=-80\text{ мкА}(2)$
I_{IL}	Входной ток логического нуля для портов 1, 2, 3			-50	мкА	$V_{IN}=0,45\text{ В}$
I_{TL}	Ток перехода из "1" в "0" для портов 1, 2, 3			-650	мкА	$V_{IN}=2\text{ В}$
I_{LI}	Входной ток утечки для порта 0, \overline{EA}			± 10	мкА	$0,45 < V_{IN} < V_{CC}$
R_{RST}	Резистор между входом RST и выводом V_{SS}	50		150	кОм	

Продолжение таблицы 5.3

Обозн.	Параметр	Мин.	Тип. (3)	Макс.	Ед. изм.	Режим измерения
C _{IO}	Емкость выводов			10	пФ	Частота измерения 1 МГц; T _A =25°C
I _{CC}	Ток потребления: Активный режим, 12 МГц(4) Idle режим, 12 МГц(4) Режим микрopotребления (Power Down Mode)		11 1,7 5	20 5 50	мА мА мкА	(5)

Примечания к таблице 5.3:

1. Емкостная нагрузка на линиях портов 0 и 2 может вызвать паразитные шумовые импульсы, особенно сильно воздействующие на напряжения V_{OL}, V_{OL1} вывода ALE и выводов портов 1 и 3. Шум появляется из-за разряда емкостей в выводы порта 0 и порта 1 при переходе напряжений на выводах этих портов из "1" в "0" во время операций на внешней шине. В наихудшем случае (емкость нагрузки больше 100 пФ) шумовой импульс на выводе ALE может превышать 0,8 В. В таких случаях желательно сигнал ALE пропустить через триггер Шмитта или использовать триггерную защелку с триггером Шмитта на стробирующем входе.

2. Емкостная нагрузка на линиях портов 0 и 2 может быть причиной эффекта, заключающегося в том, что напряжение V_{OH} на выводах ALE и PSEN кратковременно падает ниже величины 0,9V_{CC} во время стабилизации бит адреса.

3. "Типовые" значения основаны на ограниченном количестве промеренных образцов микросхем, взятых из ранее произведенных партий, и не являются гарантированными. Приведенные значения получены при комнатной температуре и V_{CC}=5 В.

4. I_{CCMAX} на других частотах вычисляется:

Активный режим:

$$I_{CCMAX}=1,47FREQ+2,35$$

Idle режим:

$$I_{CCMAX}=0,33FREQ+1,05,$$

где FREQ — частота синхронизации в МГц. I_{CCMAX} дан в мА. См. рис. 5.1.

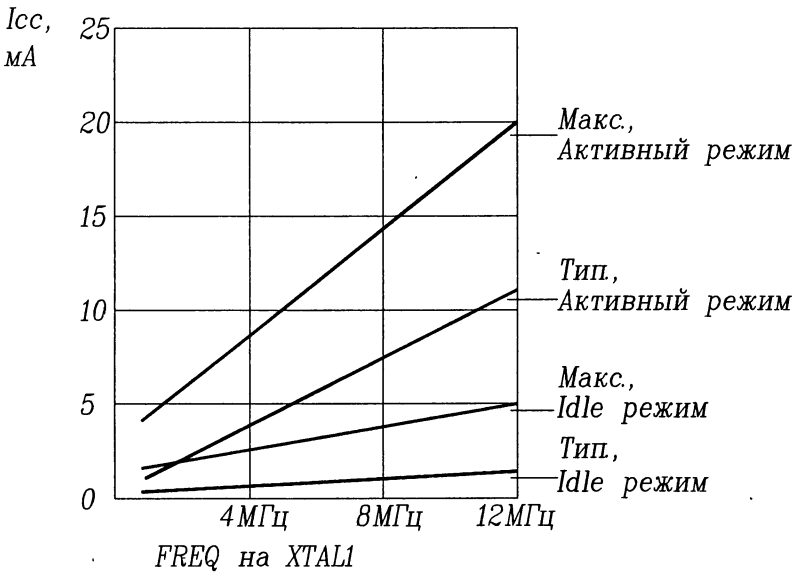


Рис. 5.1. Зависимость I_{CC} от частоты.

5. На рис. 5.2—5.4 показаны схемы измерения I_{CC} . Минимальное значение V_{CC} в режиме микропотребления составляет 2 В.

6. В статическом состоянии ток I_{OL} должен быть внешне ограничен следующими значениями:

Максимальный I_{OL} на один вывод порта: 10 мА

Максимальный I_{OL} на 8-битный порт:

— Порт 0: 26 мА

— Порты 1, 2 и 3: 15 мА

Максимальный общий I_{OL} на все выходные линии: 71 мА

7. Рекомендуемая емкость нагрузки, не более:

для выводов порта 0, ALE и \overline{PSEN} : 100 пФ

для остальных выводов: 80 пФ

8. Предельные значения параметров:

Температура окружающей среды: 0°C—+70°C

Напряжение на любом выводе относительно V_{SS} : -0,5 В— $V_{CC}+0,5$ В

Напряжение на V_{CC} относительно V_{SS} : -0,5 В—+6,5 В

Максимальный ток I_{OL} на один контакт ввода/вывода: 15 мА

Рассеиваемая мощность (значение основано на максимальной температуре и тепловом сопротивлении корпуса) 1,0 Вт.

Превышение предельных значений параметров может вызвать отказ микросхемы.

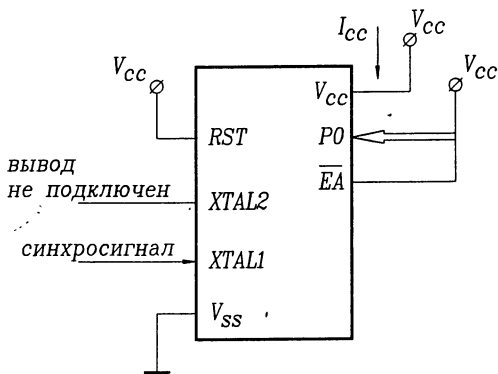


Рис. 5.2. Схема измерения I_{CC} . Активный режим. Все другие выходы никуда не подключены.

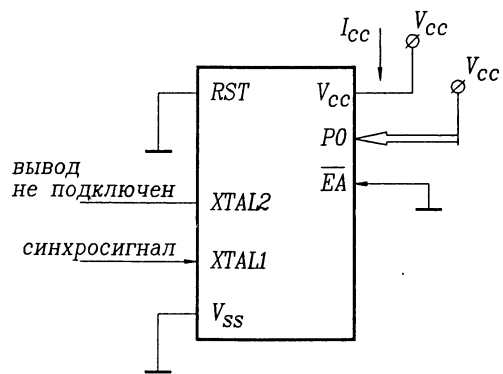


Рис. 5.3. Схема измерения I_{CC} . Idle режим. Все другие выходы не подключены.

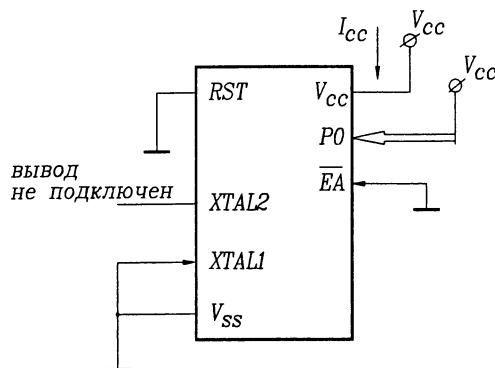


Рис. 5.4. Схема измерения I_{CC} . Режим микропотребления. Все другие выходы никуда не подключены. $V_{CC}=2$ В—6 В

ГЛАВА 6

БИТОВЫЙ ПРОЦЕССОР ОМЭВМ СЕМЕЙСТВА МК51

Пригодность архитектуры каждого компьютера для конкретного класса задач определяется тем, насколько его система команд соответствует задачам, которые должны быть выполнены. Поэтому для дискретного управления в реальном масштабе времени наличие в системе команд операций непосредственно над битами приводит к созданию более производительных систем и программ обработки входной и выходной двоичной информации. С этой целью в ОМЭВМ семейства МК51 введены специальные средства, называемые битовым процессором, которые поддерживают прямые логические операции с отдельными битами и операции их тестирования и позволяют использовать однобитовые переменные в логических операциях. Цель данного приложения — показать и объяснить применение битового процессора ОМЭВМ семейства МК51.

6.1. Общие сведения

В систему команд ОМЭВМ семейства МК51 введены специальные команды для выполнения операций с битовыми переменными. Имеется 17 таких команд, которые перечислены в табл. 6.1.

Эти команды в зависимости от выполняемой функции могут быть одно-, двух- или трехбайтные. Те из них, которые оперируют с флагом переноса, имеют однобайтный код или код, за которым следует байт смещения, использующийся для вычисления адреса условного перехода (рис. 6.1а). В более обобщенных командах битовых операций после кода добавляется байт адреса прямоадресуемого бита, образуя двух- или трехбайтные команды (рис. 6.1б). На рис. 6.1 для справки приведены коды этих команд.

С помощью указанных команд можно обращаться непосредственно к 128 битам внутреннего ОЗУ и к 83 битам одиннадцати восьмиразрядных регистров ОМЭВМ.

Таблица 6.1

Мнемоническое обозначение	Описание команды	Число байтов	Число циклов
SETB C	Установка флага переноса	1	1
SETB bit	Установка бита	2	1
CLR C	Сброс флага переноса	1	1
CLR bit	Сброс бита	2	1
CPL C	Инверсия флага переноса	1	1
CPL bit	Инверсия бита	2	1
MOV C, bit	Пересылка бита во флаг переноса	2	1
MOV bit, C	Пересылка флага переноса в бит	2	2
ANL C, bit	"Логическое И" бита и флага переноса	2	2
ANL C, /bit	"Логическое И" инверсии бита и флага переноса	2	2
ORL C, bit	"Логическое ИЛИ" бита и флага переноса	2	2
ORL C, /bit	"Логическое ИЛИ" инверсии бита и флага переноса	2	2
JC rel8	Переход, если флаг переноса установлен	2	2
JNC rel8	Переход, если флаг переноса сброшен	2	2
JB bit, rel8	Переход, если бит установлен	3	2
JNB bit, rel8	Переход, если бит сброшен	3	2
JBC bit, rel8	Переход, если бит установлен, и сброс этого бита	3	2

ОБОЗНАЧЕНИЯ:

C — *флаг переноса*;

bit — *128 программно-доступных битов, любой I/O вывод, бит управления или состояния*;

/bit — *128 программно-доступных битов, любой I/O вывод, бит управления или состояния, взятые с инверсией*;

rel8 — *байт относительного смещения (условный переход осуществляется в диапазоне от -128 до +127 байтов относительно адреса первого байта следующей команды)*.

Код команды		Код команды:
SETB C		11010011B (D3H)
CLR C		11000011B (C3H)
CPL C		10110011B (B3H)
Код команды	Смещение	Код команды:
JC	<rel8>	01000000B (40H)
JNC	<rel8>	01010000B (50H)

а) Команды проверки и управления флагом переноса

Код команды	Адрес бита	Код команды:
SETB	<bit>	11010010B (D2H)
CLR	<bit>	11000010B (C2H)
CPL	<bit>	10110010B (B2H)
MOV C,	<bit>	10100010B (A2H)
MOV	<bit>, C	10010010B (92H)
ANL C,	<bit>	10000010B (82H)
ANL C,	</bit>	10110000B (B0H)
ORL C,	<bit>	01110010B (72H)
ORL C,	</bit>	10100000B (A0H)

Код команды	Адрес бита	Смещение	Код команды:
JB	<bit>,	<rel8>	00100000B (20H)
JNB	<bit>,	<rel8>	00110000B (30H)
JBC	<bit>,	<rel8>	00010000B (10H)

б) Команды проверки и операций с битами

Рис. 6.1. Форматы команд операций над битами

6.2. Прямая адресация битов

В зависимости от значения байта адреса прямоадресуемый бит выбирается из двух групп битов. Значения адреса от 0 (00H) до 127 (7FH) определяют биты в 16-байтном блоке внутреннего ОЗУ между адресами 20H и 2FH (см. рис. 6.2а). Они пронумерованы последовательно от младшего бита младшего байта к старшему биту старшего байта. Адреса битов от 128 (80H) до 255 (0FFH) соответствуют битам регистров специальных функций. Адреса этих битов вычисляются иначе, чем адреса ячеек ОЗУ: пять старших битов адреса совпадают с собственными адресами регистров, а три младших бита адреса идентифицируют позицию бита в пределах регистра (см. рис. 6.2б).

Хотя МК51 имеет 20 регистров специальных функций, побитовый доступ обеспечен только для 11 (PSW, ACC, B, P0, P1, P2, P3, TCON, SCON, IE, IP). У 6 из них (PSW, P3, TCON, SCON, IE, IP) разряды имеют собственные символические имена. На рис. 6.3 приведены символическое обозначение, название и позиция битов, являющихся разрядами этих регистров. Эти 33 индивидуально адресуемых бита позволяют контролировать состояние процессора и всех внутренних аппаратно реализованных функции (таймер/счетчик, последовательный порт, логика прерываний и т.д.).

Команды общего назначения, адресуемые непосредственно к битам, могут обращаться к ним (в том числе и к биту переноса), используя соответствующую мнемонику: CY, AC, F0 и т.д.

К битам всех 11 регистров также можно обратиться, используя соответствующее позиционное обозначение: PSW.1, ACC.2, B.3, P0.4 и т.д.

Регистр ACC (аккумулятор) и регистр B относятся к байтовой арифметике, но их отдельные биты могут использоваться в программе в качестве произвольных 16 флагов. В сумме со 128 ячейками ОЗУ это дает 144 битовые ячейки общего назначения для хранения переменных или программных флагов.

Адрес байта ОЗУ	Адреса битов								Адрес регистра	Адреса битов								Обозначение регистра
7								0	7									0
7FH									0FFH									
2FH	7F	7E	7D	7C	7B	7A	79	78										
2EH	77	76	75	74	73	72	71	70	0F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
2DH	6F	6E	6D	6C	6B	6A	69	68										
2CH	67	66	65	64	63	62	61	60	0E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
2BH	5F	5E	5D	5C	5B	5A	59	58										
2AH	57	56	55	54	53	52	51	50	0D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
29H	4F	4E	4D	4C	4B	4A	49	48										
28H	47	46	45	44	43	42	41	40	0B8H	—	—	—	BC	BB	BA	B9	B8	IP
27H	3F	3E	3D	3C	3B	3A	39	38										
26H	37	36	35	34	33	32	31	30	0B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
25H	2F	2E	2D	2C	2B	2A	29	28										
24H	27	26	25	24	23	22	21	20	0A8H	AF	—	—	AC	AB	AA	A9	A8	IE
23H	1F	1E	1D	1C	1B	1A	19	18										
22H	17	16	15	14	13	12	11	10	0A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
21H	0F	0E	0D	0C	0B	0A	09	08										
20H	07	06	05	04	03	02	01	00	98H	9F	9E	9D	9C	9B	9A	99	98	SCON
1FH	Банк 3																	
18H																		
17H	Банк 2								90H	97	96	95	94	93	92	91	90	P1
10H																		
0FH	Банк 1								88H	8F	8E	8D	8C	8B	8A	89	88	TCON
08H																		
07H	Банк 0								80H	87	86	85	84	83	82	81	80	P0
00																		

а) Адреса битов ОЗУ

б) Адреса битов специальных регистров

Рис. 6.2. Адреса прямоадресуемых битов

Все 32 вывода портов P0-P3 могут индивидуально адресоваться как вход, выход или вход/выход в любой комбинации. Любой вывод может служить программно реализуемым стробом, тестовым входом или последовательным каналом ввода/вывода.

Заметим, что пять регистров с потенциальной возможностью побитового доступа (0C0H, 0C8H, 0D8H, 0E8H и 0F8H) зарезервированы для будущего расширения возможностей МК51. Обращение к зарезервированным регистрам может привести к непредсказуемым результатам. Также не имеют доступа пять битов регистров IE и IP: IE.5, IE.6, IP.5-IP.7. Они не могут использоваться в программе в качестве флагов.

Для обработки битов регистров ОЗУ, не имеющих побитового доступа, могут быть использованы логические операции с байтами.

Регистр слова состояния программы PSW (рис. 6.3а) содержит биты флагов и состояния процессора, включая флаг переноса. Операции обработки байтов, воздействуя на регистр PSW, могут таким образом влиять и на бит переноса.

Регистр PSW							
7				0			
CY	AC	F0	RS1	RS0	OV	—	P
Символическое обозначение CY	Позиционное обозначение PSW. 7	Название и назначение Флаг переноса Устанавливается или сбрасывается при выполнении некоторых арифметических и логических команд					
AC	PSW. 6	Флаг дополнительного переноса Устанавливается или сбрасывается при выполнении команд сложения и вычитания для указания на заем или перенос для (из) бита 3					
F0	PSW. 5	Флаг 0 Устанавливается, сбрасывается и проверяется программно как флаг пользователя					
RS1 RS0	PSW. 4 PSW. 3	Биты 0 и 1 выбора банка ОЗУ Устанавливаются и сбрасываются программно для указания используемого банка ячеек ОЗУ RS1=0, RS0=0 – банк 0 (00H–07H) RS1=0, RS0=1 – банк 1 (08H–0FH) RS1=1, RS0=0 – банк 2 (10H–17H) RS1=1, RS0=1 – банк 3 (18H–1FH)					
OV	PSW. 2	Флаг переполнения Устанавливается и сбрасывается аппаратно при выполнении арифметических команд для указания возникновения переполнения					
— P	PSW. 1 PSW. 0	Не используется Флаг четности Устанавливается и сбрасывается аппаратно в каждом машинном цикле для указания четности количества битов "1" результата в аккумуляторе					

Рис. 6.3а. Организация регистра слова состояния программы PSW

Регистр P3							
7				0			
RD	WR	T1	T0	INT1	INT0	TXD	RXD
Символическое обозначение RD	Позиционное обозначение P3. 7	Название и назначение Выход управления чтением данных Вырабатывается отрицательный импульс при чтении из внешней памяти данных					
WR	P3. 6	Выход управления записью данных Вырабатывается отрицательный импульс при записи во внешнюю память данных					
T1	P3. 5	Внешний вход таймера/счетчика 1 или проверяемый вход					
T0	P3. 4	Внешний вход таймера/счетчика 0 или проверяемый вход					
INT1	P3. 3	Вход аппаратного прерывания 1 Прерывание вызывается низким уровнем или отрицательным фронтом сигнала					
INT0	P3. 2	Вход аппаратного прерывания 0 Прерывание вызывается низким уровнем или отрицательным фронтом сигнала					
TXD	P3. 1	Выход данных последовательного порта в асинхронном режиме Тактовый выход в режиме сдвигового регистра					
RXD	P3. 0	Вход данных последовательного порта в асинхронном режиме Выход данных в режиме сдвигового регистра					

Рис. 6.3б. Организация порта ввода/вывода P3 (альтернативные функции)

Регистр IE'			7					0		
			EA	—	—	ES	ET1	EX1	ET0	EX0
Символическое обозначение	Позиционное обозначение	Название и назначение								
EA	IE.7	Бит разрешения прерываний								
		Программно сбрасывается для запрета всех прерываний и устанавливается для разрешения прерываний в зависимости от состояния битов IE.4–IE.0								
—	IE.6	Не используется								
—	IE.5	Не используется								
ES	IE.4	Бит управления прерываниями последовательного порта								
		Программно устанавливается/сбрасывается для разрешения/запрета прерываний по флагам TI и RI								
ET1	IE.3	Бит разрешения прерывания от таймера 1								
		Программно устанавливается и сбрасывается для разрешения/запрета прерываний от таймера/счетчика 1								
EX1	IE.2	Бит разрешения внешнего прерывания 1								
		Программно устанавливается/сбрасывается для разрешения/запрета аппаратного прерывания INT1								
ET0	IE.1	Бит разрешения прерывания от таймера 0								
		Программно устанавливается/сбрасывается для разрешения/запрета прерываний от таймера/счетчика 0								
EX0	IE.0	Бит разрешения внешнего прерывания 0								
		Программно устанавливается/сбрасывается для разрешения/запрета аппаратного прерывания INT0								

Рис. 6.3д. Организация регистра разрешения прерываний IE

Регистр IP							
7							0
—	—	—	PS	PT1	PX1	PT0	PX0
Символическое обозначение	Позиционное обозначение	Название и назначение					
—	IP.7	Не используется					
—	IP.6	Не используется					
—	IP.5	Не используется					
PS	IP.4	Бит управления приоритетом прерываний последовательного порта					
		Программно устанавливается/сбрасывается для присвоения высокого/низкого приоритета прерываниям последовательного порта					
PT1	IP.3	Бит управления приоритетом прерывания от таймера 1					
		Программно устанавливается/сбрасывается для присвоения высокого/низкого приоритета прерываниям от таймера/счетчика 1					
PX1	IP.2	Бит управления приоритетом внешнего прерывания 1					
		Программно устанавливается/сбрасывается для присвоения высокого/низкого приоритета аппаратному прерыванию INT1					
PT0	IP.1	Бит управления приоритетом прерывания от таймера 0					
		Программно устанавливается/сбрасывается для присвоения высокого/низкого приоритета прерываниям от таймера/счетчика 0					
PX0	IP.0	Бит управления приоритетом внешнего прерывания 0					
		Программно устанавливается/сбрасывается для присвоения высокого/низкого приоритета аппаратному прерыванию INT0					

Рис. 6.3е. Организация регистра приоритетов прерываний IP

6.3. Система команд

Команды битового процессора представлены в табл. 6.1 и на рис. 6.1.

Управление состоянием

Двухбайтные команды SETB, CLR и CPL устанавливают, обнуляют или логически дополняют до 2 (для битового операнда эквивалентно инверсии) адресуемые биты или флаги за один машинный цикл. Команды SETB и CLR осуществляют загрузку бита константой 1 или 0 соответственно. Однобайтная версия этих команд выполняют те же операции над битом переноса (C).

В языке ассемблера МК51 адрес бита указывается одним из трех способов:

- числом или выражением, соответствующим прямому адресу бита;
- названием или адресом регистра, содержащего данный бит, и позицией бита в регистре (0-7), разделенными точкой;
- для определенных битов (рис. 6.3) указанием символического имени.

Биты также могут обозначаться произвольными именами. директивой ассемблера "BIT". Например, 5-й бит регистра PSW может быть очищен любой из четырех команд:

```
USR_FLG BIT PSW.5      ; Описание символа пользователя
;
;      CLR 0D5H        ; Абсолютная адресация
;      CLR PSW.5       ; Использование точечного оператора
;      CLR F0          ; Использование собственного имени бита
;      CLR USR_FLG     ; Символ пользователя.
```

Пересылка данных

Двухбайтные команды MOV могут передавать любой адресуемый бит в перенос (C) за один машинный цикл или копировать перенос в любой бит за два цикла. Бит может быть перемещен из ячейки в ячейку с помощью комбинации этих двух команд.

Флаг переноса используется как одноразрядный регистр-аккумулятор для логических операций. Значение переноса можно сохранить командами PUSH и POP.

Логические операции

Команды ANL и ORL выполняют операции логических "И" и "ИЛИ" соответственно между битом переноса (C) и любым другим битом и помещают результат в перенос. Наличие символа "/" перед операндом источника указывает на использование логического дополнения до 2 адресуемого бита, но это не влияет на операнд источника.

Команды проверки битов

Команды условных переходов JC rel8 (переход по переносу) и JNC rel8 (переход по отсутствию переноса) проверяют состояние флага переноса и, если перенос равен 1 или 0 соответственно, передают управление команде, расположенной по адресу перехода. Он вычисляется как сумма адреса команды, следующей за командой перехода, и байта смещения (rel8).

Команды JB bit,rel8 (переход по установленному биту) и JNB bit,rel8 (переход по сброшенному биту) проверяют состояние бита, байт адреса которого (bit) указывается после кода команды, и выполняют аналогичный предыдущему условный переход.

Команда JBC bit,rel8 сочетает операции условного перехода по состоянию бита и его очистки. Она сначала в зависимости от состояния указанного бита выполняет условный переход, вычисляя его по байту смещения (rel8), а затем очищает этот бит.

Все команды условного перехода используют относительную адресацию (относительно программного счетчика) и выполняются за два машинных цикла. Последний байт инструкции представляет собой код смещения со знаком в интервале от -128 до +127. При выполнении команд условных переходов процессор прибавляет это значение к инкрементированному значению программного счетчика,

чтобы вычислить исполнительный адрес перехода. Другими словами при смещении 00H условный переход осуществится на следующую команду.

Участок программы или подпрограммы, который содержит относительные переходы только на близкие адреса, будет иметь машинные коды, не зависящие от их расположения в памяти. Ассемблированная подпрограмма может быть перемещена в памяти без необходимости модифицировать программу или вычисления новых адресов переходов. Для облегчения реализации такой гибкости программ существует команда SJMP (короткий переход), которая использует относительную адресацию. В языке ассемблера МК51 в командах короткого перехода вместо смещения указывается метка. Ассемблер МК51 автоматически вычисляет необходимое смещение для получения заданного адреса или метки. Кроме того программист получит сообщение об ошибке в случае выхода смещения за допустимый диапазон значений.

Взаимодействие с другими командами

На флаг переноса также воздействуют команды, перечисленные в табл. 6.2. Флаг переноса может быть "прокручен" через аккумулятор командами сдвига и изменен командами арифметических и логических операций.

Таблица 6.2

Мнемоническое обозначение	Описание команды	Число байтов	Число циклов
ADD A,Rn	Сложение содержимого аккумулятора и регистра	1	1
ADD A,direct	Сложение содержимого аккумулятора и байта	2	1
ADD A,@Ri	Сложение содержимого аккумулятора и ячейки ОЗУ, адрес которой находится в регистре	1	1
ADD A,#data	Сложение содержимого аккумулятора и непосредственных данных	2	1
ADDC A,Rn	Сложение с переносом содержимого аккумулятора и регистра	1	1
ADDC A,direct	Сложение с переносом содержимого аккумулятора и байта	2	1
ADDC A,@Ri	Сложение с переносом содержимого аккумулятора и ячейки ОЗУ, адрес которой находится в регистре	1	1
ADDC A,#data	Сложение с переносом содержимого аккумулятора и непосредственных данных	2	1
SUBB A,Rn	Вычитание с заемом содержимого регистра из содержимого аккумулятора	1	1
SUBB A,direct	Вычитание с заемом байта из содержимого аккумулятора	2	1
SUBB A,@Ri	Вычитание с заемом содержимого ячейки ОЗУ, адрес которой находится в регистре, из содержимого аккумулятора	1	1
SUBB A,#data	Вычитание с заемом непосредственных данных из содержимого аккумулятора	2	1
MUL AB	Перемножение содержимого аккумулятора и регистра В	1	4
DIV AB	Деление содержимого аккумулятора на содержимое регистра В	1	4
DA A	Десятичная коррекция содержимого аккумулятора	1	1
RLC A	Сдвиг содержимого аккумулятора влево через флаг переноса	1	1
RRC A	Сдвиг содержимого аккумулятора вправо через флаг переноса	1	1
CJNE A,direct,rel8	Сравнение содержимого аккумулятора и байта и переход, если не равно	3	2
CJNE A,#data,rel8	Сравнение содержимого аккумулятора и непосредственных данных и переход, если не равно	3	2
CJNE Rn,#data,rel8	Сравнение непосредственных данных и содержимого регистра и переход, если не равно	3	2
CJNE @Ri,#data,rel8	Сравнение непосредственных данных и ячейки ОЗУ, адрес которой находится в регистре, и переход, если не равно	3	2

ОБОЗНАЧЕНИЯ:

A — регистр-аккумулятор;

B — регистр В;

Rn — регистр Rn текущего банка данных, где n=0...7;

@Ri — адрес ячейки ОЗУ, записанный в регистр Ri, где i=0,1;

direct — прямой адрес байта;

#data — непосредственные данные;

rel8 — байт относительного смещения (условный переход осуществляется в диапазоне от -128 до +127 байтов относительно адреса первого байта следующей команды).

6.4. Применение битового процессора

Реализация последовательного порта программным способом

ОМЭВМ семейства МК51 могут программно принимать и передавать последовательные данные с использованием системы команд битового процессора. Поскольку любой вывод порта может служить последовательным входом или выходом, одновременно можно организовать несколько последовательных линий связи.

На рис. 6.4 показаны алгоритмы приема и передачи байтов данных. Программа будет обращаться к этому алгоритму 8 раз, синхронизируясь старт-битом, тактовым сигналом, программной задержкой или прерыванием от таймера. Данные принимаются путем опроса входного вывода, присвоения биту переноса состояния этого вывода, сдвига бита переноса в буфер данных и сохранения полученного значения в ОЗУ. Передача происходит путем сдвига содержимого буфера передаваемого байта через перенос и выдачи состояния флага переноса на выходной вывод.

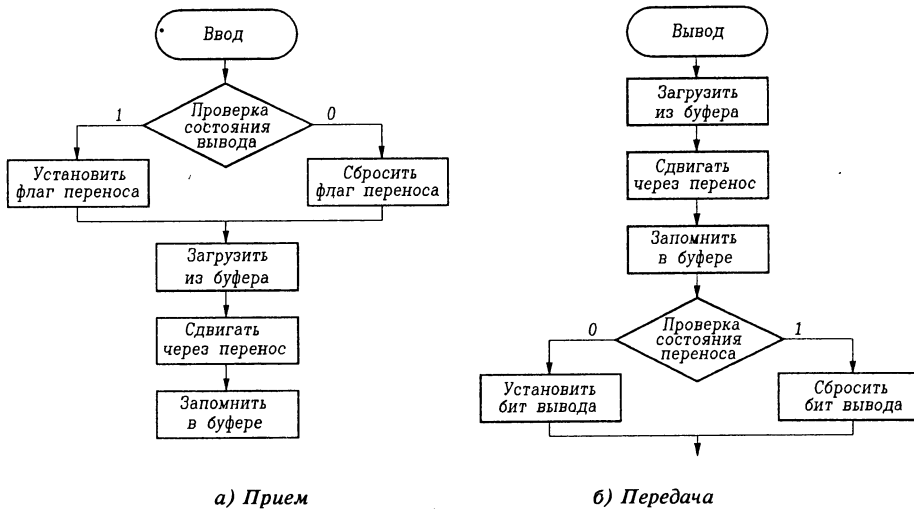


Рис. 6.4. Алгоритмы последовательного порта ввода/вывода

На рис. 6.5 приведены программы, реализующие показанные на рис. 6.4 алгоритмы приема и передачи для трех различных микропроцессоров: 8085, 8048 и 8051. Программная реализация последовательного порта на МК51 наиболее эффективна.

Решение уравнений комбинаторной логики

На рис. 6.6 показана реализация на ТТЛ-элементах функции шести переменных от U до Z, которая является решением уравнения

$$Q = (U * (V + W)) + (X * \bar{Y}) + \bar{Z}.$$

Уравнения такого рода решаются с помощью карт Карно или аппарата алгебры булевой логики. Для сравнения выполним эту функцию тремя способами, ограничиваясь тремя подмножествами системы команд МК51.

Предположим, что U и V являются входными выводами некоторого порта, W и X — биты состояния двух периферийных контроллеров, а Y и Z — программные флаги, ранее установленные в программе. Окончательный результат должен выдаваться на выходной вывод порта.

Первые два способа решения основываются на алгоритме, показанном на рис. 6.7, который может быть выполнен практически любым микропроцессором. Выполнение программы идет с проверками и ветвлением, до тех пор пока не будет получен результат, который выдается на порт.

8085	8048	8051
IN SERPORT ANI MASK JZ LO CMC LO: LXI HL, SERBUF MOV A, M RR MOV M, A	CLR C JNTØ LO CPL C LO: MOV RØ, #SERBUF MOV A, @RØ RRC A MOV @RØ, A	MOV C, SERPIN MOV A, SERBUF RRC A MOV SERBUF, A
8 команд	7 команд	4 команды
14 байт	9 байт	7 байт
56 состояний	9 циклов	4 цикла
19 мкс	22,5 мкс	4 мкс

а) Подпрограмма ввода

8085	8048	8051
LXI HL, SERBUF MOV A, M RR MOV M, A IN SERPORT JC HI LO: ANI NOT MASK JMP CNT HI: ORI MASK CNT: OUT SERPORT	MOV RØ, #SERBUF MOV A, @RØ RRC A MOV @RØ, A JC HI ANL SERPRT, #NOT MASK JMP CNT HI: ORL SERPRT, #MASK CNT:	MOV A, SERBUF RRC A MOV SERBUF, A MOV SERPIN, C
1Ø команд	8 команд	4 команды
2Ø байт	13 байт	7 байт
72 состояния	11 циклов	5 циклов
24 мкс	27,5 мкс	5 мкс

б) Подпрограмма вывода

Рис. 6.5. Подпрограммы последовательного ввода/вывода


```

OUTBUF DATA 22H ; карта состояния выходов
;
TESTV: MOV A,P2
      ANL A,#00000100B
      JNZ TESTU
      MOV A,TCON
      ANL A,#00100000B
      JZ TESTX
TESTU: MOV A,P1
      ANL A,#00000010B
      JNZ SETQ
TESTX: MOV A,TCON
      ANL A,#00001000B
      JNZ TESTZ
      MOV A,20H
      ANL A,#00000001B
      JZ SETQ
TESTZ: MOV A,21H
      ANL A,#00000010B
      JZ SETQ
CLRQZ: MOV A,OUTBUF
      ANL A,#11110111B
      JMP OUTQ
SETQ:  MOV A,OUTBUF
      ORL A,#00001000B
OUTQ:  MOV OUTBUF,A
      MOV P3,A

```

```

;      Пример 16. Использование команд проверки битов
;
;      ;BFUNC2 вычисляет логическую функцию 6 переменных путем прямого
;      ;опроса каждого бита. Биты обозначены в соответствии с
;      ;символами, использованными в алгоритме.
;      ;      (Используется возможность проверки битов)
;
U      BIT P1.1
V      BIT P2.2
W      BIT TF0
X      BIT IE1
Y      BIT 20H.0
Z      BIT 21H.1
Q      BIT P3.3
;
TEST_V: JB V,TEST_U
      JNB W,TEST_X
TEST_U: JB U,SET_Q
TEST_X: JNB X,TEST_Z
      JNB Y,SET_Q
TEST_Z: JNB Z,SET_Q
CLR_Q: CLR Q
      JMP NXTTST
SET_Q: SETB Q
NXTTST: ... ; продолжение программы

```

Пример 1в. Использование битового процессора

; FUNC3 вычисляет логическую функцию 6 переменных с
; использованием возможностей битового процессора МК51.

```
MOV C,V
ORL C,W ; Выход вентилей ИЛИ
ANL C,U ; Выход верхнего вентилей И
MOV F0,C ; Сохранение промежуточного состояния
MOV C,X
ANL C,/Y ; Выход нижнего вентилей И
ORL C,F0 ; Использование ранее вычисленного значения
ORL C,/Z ; Использование последней входной переменной
MOV Q,C ; Вывод результата
```

Функции автомобильной панели

Рассмотрим всемирно известную модель — сигналы поворота автомобиля. Представим себе положения рычага поворотов на колонке рулевого управления как однополюсного трехходового шарнирного переключателя. В центральном положении все контакты разомкнуты. В верхнем или нижнем положениях контакты замыкают цепи соответствующих ламп-мигалок передних и задних указателей поворота и индикаторов поворота на панели. Когда замыкается аварийный переключатель, зажигаются все шесть ламп. Мигание обеспечивается термо-механическим реле. Нажатие тормозной педали включает немигающий задний свет, если в это время не происходит поворот, при котором сохраняется мигание заднего указателя поворота. Все эти ситуации сведены в таблице 6.3. Кроме этого каждая из внешних сигнальных ламп имеет второе назначение — ослабленный свет габаритных указателей на стоянке.

Таблица 6.3

Входные сигналы				Выходные сигналы			
Педаль тормоза	Аварийный переключатель	Переключатель левого поворота	Переключатель правого поворота	Левая передняя лампа и указатель на перед. панели	Правая передняя лампа и указатель на перед. панели	Левая задняя лампа	Правая задняя лампа
0 0 0	0 0 0	0 0 1	0 1 0	выкл выкл мигание	выкл мигание выкл	выкл выкл мигание	выкл мигание выкл
0 0 0	1 1 1	0 0 1	0 1 0	мигание мигание мигание	мигание мигание мигание	мигание мигание мигание	мигание мигание мигание
1 1 1	0 0 0	0 0 1	0 1 0	выкл выкл мигание	выкл мигание выкл	вкл вкл мигание	вкл мигание вкл
1 1 1	1 1 1	0 0 1	0 1 0	мигание мигание мигание	мигание мигание мигание	вкл вкл мигание	вкл мигание вкл

Применение ОМЭВМ семейства МК51 позволяет сделать простую, надежную и удобную в эксплуатации систему контроля и управления электрическим оборудованием автомобиля. В схеме, показанной на рис. 6.8, используются пять выводов для пяти входных переменных: выбор левого поворота, выбор правого поворота, тормозная педаль нажата, включен аварийный сигнал, включены габаритные огни, — и шесть выводов для шести выходных переменных: передние и задние левые и правые указатели поворота и индикаторы поворота на панели управления. Микрокомпьютер выполняет функции контроля и управления, изменяя выходные сигналы по прошествии времени или при изменении условий на входах.

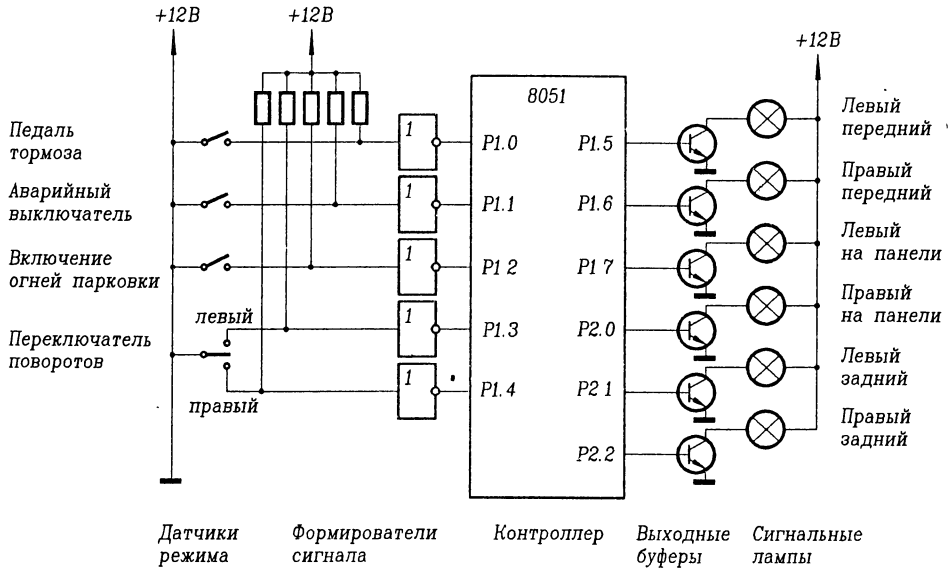


Рис. 6.8. Применение ОМЭВМ для контроля и управления лампами автомобиля

В данном примере сегменты полной программы представлены в той последовательности, в какой они расположены в программе.

Описание входов			(Положительная логика)
;			
;			
BRAKE	BIT	P1.0	; Педаль тормоза нажата
EMERG	BIT	P1.1	; Аварийный сигнал включен
PARK	BIT	P1.2	; Включение габаритных огней
L_TURN	BIT	P1.3	; Включен левый поворот
R_TURN	BIT	P1.4	; Включен правый поворот
;			
Описание выходов			(Положительная логика)
;			
;			
L_FRNT	BIT	P1.5	; Передний указатель левого поворота
R_FRNT	BIT	P1.6	; Передний указатель правого поворота
L_DASH	BIT	P1.7	; Индикатор левого поворота на панели
R_DASH	BIT	P2.0	; Индикатор правого поворота на панели
L_REAR	BIT	P2.1	; Задний указатель левого поворота
R_REAR	BIT	P2.2	; Задний указатель правого поворота
;	

Преимущества символической адресации проявятся на дальнейших этапах разработки. Очень вероятно, что потребуются введение дополнительных входных или выходных сигналов, и произвольное обозначение вывода, присвоенное на раннем этапе разработки, не окажется оптимальным. В данном случае для переназначения выводов входа/выхода достаточно сделать изменения в исходных обозначениях переменных. Ассемблер ASM51 автоматически согласует адреса и символические обозначения с переобозначенными переменными.

```

; SUB_DIV DATA 20H ; Делитель частоты прерываний
HI_FREQ BIT SUB_DIV.0 ; Бит генератора высокой частоты
LO_FREQ BIT SUB_DIV.7 ; Бит генератора низкой частоты
;
; ORG 0000H
; JMP INIT
;
; ORG 100H
INIT: MOV TMOD,#00000001B ; Таймер 0 в режим 1
      MOV TL0,#0 ; Инициализация регистров таймера
      MOV TH0,#-16
      MOV SUB_DIV,#244 ; Деление частоты прерываний на 244
      SETB ET0 ; Разрешение прерывания от таймера
      SETB EA ; Общее разрешение всех прерываний
      SETB TR0 ; Старт таймера
;      ... ; Продолжение основной программы

```

Таймер 0 (один из двух внутренних таймеров/счетчиков) может заменить термо-механическое реле мигания. Во время инициализации системы он устанавливается в режим 1 (установка младшего бита регистра управления режимом таймера TMOD). В этом случае младший байт таймера (TL0) инкрементируется в каждом машинном цикле (1 мкс), а переполнение и инкрементирование старшего байта таймера (TH0) происходит каждые 256 мкс. Поскольку прерывание от таймера 0 разрешено, каждый раз при переполнении TH0 будет возникать аппаратное прерывание.

Для организации программного делителя частоты прерываний потребуется восьмибитовая переменная-счетчик (SUB_DIV) в ОЗУ с битовой адресацией. Младший бит счетчика переключается достаточно быстро, этой частотой будем модулировать свет на стоянке (HI_FREQ). Бит 7 счетчика (LO_FREQ) будем использовать для мигания указателей поворота и сигнала аварии (частота изменения примерно 1 Гц).

```

ORG 000BH ; Вектор прерывания таймера 0
MOV TH0,#-16 ; Обслуживание прерывания таймера 0
PUSH PSW
PUSH ACC
PUSH B
DJNZ SUB_DIV,T0SERV
MOV SUB_DIV,#244

```

Загружая в TH0 число -16, получим прерывание через 4096 мкс. Подпрограмма обслуживания прерывания перезагружает старший байт таймера 0 на следующий интервал, сохраняет содержимое регистров, декрементирует регистр с переменной SUB_DIV. При начальной загрузке в регистр SUB_DIV числа 244 и декрементировании его в каждом цикле до обнуления пройдет 0,999 секунды.

Проверка состояния входов, выполнение вычислений и обновление состояния выходов (основа алгоритма) могут выполняться либо как часть подпрограммы обслуживания прерывания, либо в конце цикла выполнения всей программы.

Следует также отметить, что для снижения интенсивности свечения габаритных огней необходимо по крайней мере несколько десятков переключений в секунду, чтобы не ощущалось мерцание света. Для этого сделаем соответствующую вставку в подпрограмму обслуживания прерывания таймера 0.

DIM	BIT	PSW.1	; Назначение флага временного хранения
;	
	MOV	C,PARK	; Выполнить логическое умножение сигнала
	ANL	C,HI_FREQ	; включения габаритных огней PARK и
			; высокочастотного сигнала HI_FREQ
	MOV	DIM,C	; Сохранить результат в переменной
			; временного хранения DIM

Выражение PARK AND H_FREQ, отвечающее за приглушение света на стоянке, влияет на четыре из шести выходных переменных. Поэтому сначала вычисляем это выражение, затем временно сохраним его в ячейке DIM. PSW содержит два флага общего назначения: F0 и PSW.1. Поскольку PSW сохраняется на время выполнения подпрограммы обслуживания прерывания и затем восстанавливается, можно использовать любой из этих битов для временного хранения.

Теперь сформируем и выдадим сигнал левого поворота на панель.

;	MOV	C,L_TURN	; Установить перенос, если выбран левый
			; поворот
	ORL	C,EMERG	; или включен сигнал аварии
	ANL	C,LO_FREQ	; Выполнить логическое умножение
			; с сигналом низкой частоты LO_FREQ
	MOV	L_DASH,C	; Выдать результат на индикатор панели

Для формирования левого переднего сигнала поворота необходимо только добавить функцию паркового освещения, хранящуюся в ячейке DIM. Но при этом заметим, что функция, которая находится во флаге переноса, понадобится также для сигнала управления задними фонарями, поэтому сохраним текущее значение в F0.

;	MOV	F0,C	; Сохранить результат в F0
	ORL	C,DIM	; Сложить с функцией включения
			; габаритных огней
	MOV	L_FRNT,C	; Выдать сигнал поворота на левую
			; переднюю лампу

Наконец, сигнал заднего левого поворота должен также включиться, когда нажата тормозная педаль, если только он не был включен в процессе поворота.

```

;
MOV   C,BRAKE      ; Выполнить логическое умножение сигнала
ANL   C,/L_TURN    ;   нажатия тормозной педали и сигнала
                        ;   включения левого поворота
ORL   C,FØ         ; Сложить с функцией, хранящейся в FØ,
ORL   C,DIM         ; и функцией включения габаритных огней
MOV   L_REAR,C      ; Выдать сигнал на левую заднюю лампу

```

Аналогичные операции надо выполнить для всех правых указателей поворота.

```

;
MOV   C,R_TURN      ; Установить перенос, если выбран правый
                        ;   поворот
ORL   C,EMERG        ; или включен сигнал аварии
ANL   C,LO_FREQ      ; Выполнить логическое умножение
                        ;   с сигналом низкой частоты LO_FREQ
MOV   R_DASH,C       ; Выдать результат на индикатор панели,
MOV   FØ,C           ; сохранив его в FØ
ORL   C,DIM          ; Сложить с функцией включения
                        ;   габаритных огней
MOV   R_FRNT,C       ; Выдать сигнал поворота на правую
                        ;   переднюю лампу
MOV   C,BRAKE        ; Выполнить логическое умножение сигнала
ANL   C,/R_TURN      ;   нажатия тормозной педали и сигнала
                        ;   включения правого поворота
ORL   C,FØ           ; Сложить с функцией, хранящейся в FØ,
ORL   C,DIM          ; и функцией включения габаритных огней
MOV   R_REAR,C       ; Выдать сигнал на правую заднюю лампу

```

Внимательный читатель может заметить, что простая перестановка шагов могла бы исключить несколько команд из каждой последовательности.

Так как теперь все шесть ламп находятся в требуемом состоянии, можно вернуться из подпрограммы обработки прерывания и закончить программу. Потребуется только восстановить состояние, запомненное в начале подпрограммы.

```

POP   B              ; Восстановление регистров процессора
POP   ACC
POP   PSW
RETI

```

Уточнения к программе

Свечение нитей ламп накаливания, вообще говоря, нелинейно. При скважности 50% сигнала HI_FREQ может не получиться желаемой интенсивности. Но если потребуется, возможно установить скважность 25%, 75% и т.д., что легко достигается операциями И и ИЛИ с дополнительными битами SUD_DIV. Например,

биты 0-2 могут обеспечить семь вариантов скважности сигнала с частотой 30 Гц, как показано в табл. 6.4. Потребуется только изменить ту часть программы, где устанавливается переменная DIM.

Таблица 6.4

Биты SUB_DIV								Скважность						
7	6	5	4	3	2	1	0	12,5%	25%	37,5%	50%	62,5%	75%	87,5%
X	X	X	X	X	0	0	0	нет	нет	нет	нет	нет	нет	нет
X	X	X	X	X	0	0	1	нет	нет	нет	нет	нет	нет	да
X	X	X	X	X	0	1	0	нет	нет	нет	нет	нет	да	да
X	X	X	X	X	0	1	1	нет	нет	нет	нет	да	да	да
X	X	X	X	X	1	0	0	нет	нет	нет	да	да	да	да
X	X	X	X	X	1	0	1	нет	нет	да	да	да	да	да
X	X	X	X	X	1	1	0	нет	да	да	да	да	да	да
X	X	X	X	X	1	1	1	да	да	да	да	да	да	да

```
MOV    C,SUB_DIV.1 ; Начнем со скважности 50%
ANL    C,SUB_DIV.0 ; Уменьшим до 25%
ORL    C,SUB_DIV.2 ; Восстановим заполнение до 62,5%
MOV    DIM,C       ; Результат в DIM
```

Буферизации каждого вывода, показанной на рис. 6.8, бывает недостаточно, когда для большого числа выходов требуются уровни выше TTL. Простое решение получается с использованием последовательного порта МК51 и сдвигового регистра (рис. 6.9). Записывая байт в буфер данных (SBUF) последовательного порта, работающего в режиме 0, на выходе "RXD" получаем эти данные в виде последовательности битов с импульсной синхронизацией на выходе "TXD". Эти выходы связаны со сдвиговым регистром, в который загружается байт данных. Каскадирование сдвиговых регистров дает увеличение числа выходов. Скорость передачи данных в режиме 0 расширения ввода/вывода составляет 1 мегабайт или 8 мкс на байт. Этот режим устанавливается после сброса по умолчанию, поэтому специальной инициализации не требуется.

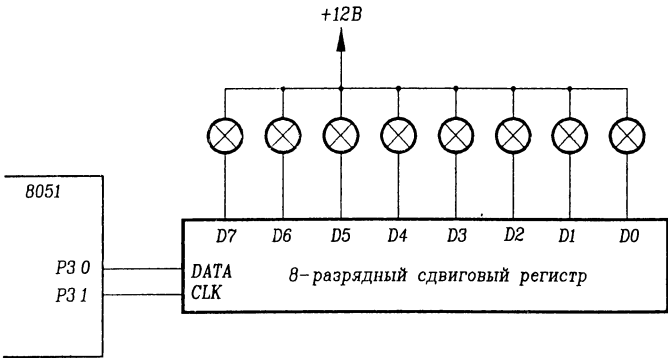


Рис. 6.9. Использование последовательного порта для расширения вывода

В программе для этого случая можно использовать регистр В в качестве "карты", соответствующей различным выходным функциям. Вместо выходных выводов программа манипулирует с битами регистра В. После выполнения всех вычислений содержимое регистра В передается через последовательный порт в сдвиговый регистр. Используя символическую адресацию к битам легко

переназначить выходные переменные. Для этого необходимо только изменить описание назначения битов.

L_FRNT	BIT	B.0	; Передний указатель левого поворота
R_FRNT	BIT	B.1	; Передний указатель правого поворота
L_DASH	BIT	B.2	; Индикатор левого поворота на панели
R_DASH	BIT	B.3	; Индикатор правого поворота на панели
L_REAR	BIT	B.4	; Задний указатель левого поворота
R_REAR	BIT	B.5	; Задний указатель правого поворота

После вычисления выходных переменных содержимое регистра В передается в сдвиговый регистр через последовательный порт:

```
MOV    SBUF,B    ; Загрузка буфера и передача
```

С помощью битового процессора можно легко выполнить дополнительные функции — обнаружение дефектов в цепях ламп. На рис. 6.10 показана такая схема.

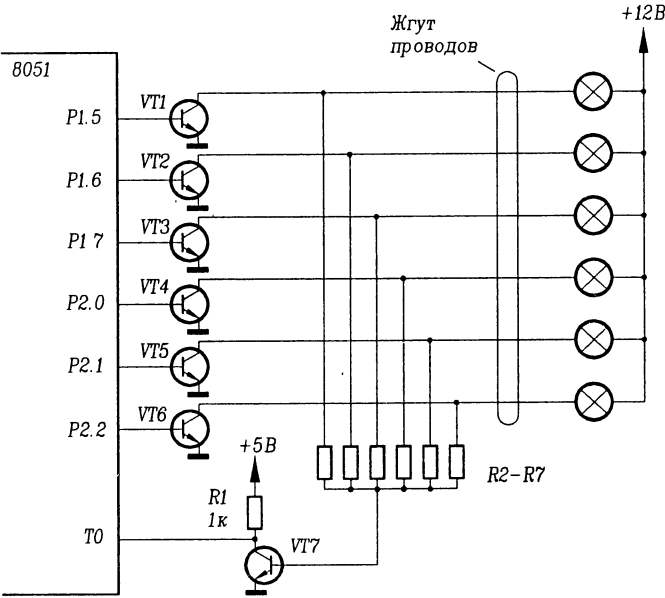


Рис. 6.10. Использование входа T0 для обнаружения неисправностей

Предположим, включены все лампы кроме одной, т.е. все коллекторы кроме одного заземлены. Через выключенную лампу, если нет обрыва цепи, от источника напряжения 12В в базу транзистора VT7 течет ток, ограниченный одним из резисторов R2-R7, чтобы лампа не светилась. Это вызовет низкий уровень напряжения на входе T0. Наличие высокого уровня явится признаком неисправности в цепи данной лампы. Это дает возможность проверить программным способом исправность электрической цепи каждой лампы.

Программно такую проверку можно выполнять один раз в секунду в момент, когда счетчик SUB_DIV перезагружается в подпрограмме обслуживания прерывания.

```

        DJNZ SUB_DIV,T0SERV
        MOV SUB_DIV,#244      ; Перезагрузка счетчика
        ORL P1,11100000B     ; Высокий уровень на выходы
        ORL P2,00000111B     ; управления лампами
;
; Если на T0 высокий уровень, перейти на программу, выполняемую
; при обнаружении дефекта электрической цепи, если низкий -
; продолжить выполнение
;
        CLR L_FRNT           ; Выключение лампы
        JB T0,FAULT          ; T0 должен быть низким
        SETB L_FRNT          ; Включение лампы
        CLR L_DASH
        JB T0,FAULT
        SETB L_DASH
        CLR L_REAR
        JB T0,FAULT
        SETB L_REAR
        CLR R_FRNT
        JB T0,FAULT
        SETB R_FRNT
        CLR R_DASH
        JB T0,FAULT
        SETB R_DASH
        CLR R_REAR
        JB T0,FAULT
        SETB R_REAR
;
        JB T0,T0SERV         ; Обход программы обработки дефекта
;
FAULT:   ...                 ; Программа, выполняемая в случае
; обнаружения электрического дефекта
T0SERV:  ...                 ; Продолжение обработки прерывания

```

Полностью листинг ассемблированной программы представлен в конце приложения. Не считая деклараций и комментариев, программа состоит из 67 выражений, которые ассемблируются в 150 байтов объектного кода. Каждое обращение к подпрограмме обслуживания требует 67 мкс плюс для электрической проверки 32 мкс один раз в секунду. Если выполнять электрическую проверку каждые 4 мс, производительность основной программы снизится менее чем на 2%.

С применением МК51 появляются новые возможности. Программно можно сделать так, что частота мигания аварийного света будет произвольной или выше, чем сигналов указателей поворота. Сигналы поворота могут накладываться на мигание аварийных сигналов. Если не жалеть ламп, то можно сделать гирлянду заднего света.

Управление сложными системами

Выходы портов ввода/вывода могут быть произвольно назначены, если общее число функций входа или выхода не превышает 32. Этого числа выводов

недостаточно для применения МК51 в системах с большим числом входных и выходных переменных. Расширить число входов можно использованием сканирования клавиатуры и датчиков.

На рис. 6.11 показана схема контроллера средней сложности со следующими характеристиками:

- 64 датчика входной переменной;
- 12 выходных сигналов;
- два внешних прерывания по приоритету;
- использование быстродействующего последовательного канала связи;
- часы реального времени и времени дня.

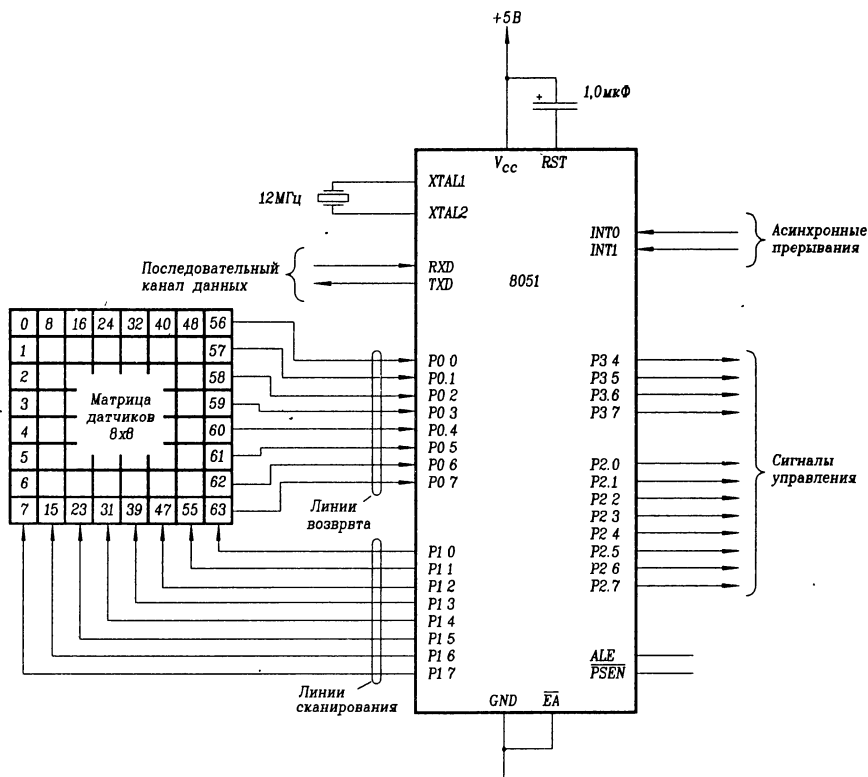


Рис. 6.11.

Входные переменные

64 входных датчика организованы в виде матрицы 8x8 элементов. Сигналы на выводах порта 1 последовательно сканируют столбцы матрицы, а через порт 0 считывается состояние каждого датчика в этом столбце. Считанные данные запоминаются в 8-байтовом блоке ОЗУ с побитовой адресацией, который после завершения сканирования всех столбцов представляет собой внутреннюю карту текущего состояния всех датчиков, используемую программой.

Последовательный порт МК51 настроен как 9-разрядный универсальный асинхронный приемопередатчик со скоростью передачи 17000 байтов в секунду. Девятый бит позволяет различать байты адреса и данных. Последовательный порт можно настроить таким образом, чтобы распознавать байты с установленным 9 битом (битом адреса), автоматически игнорируя все остальные байты.

Аппаратные прерывания INT0 и INT1 настроены соответственно на высокий приоритет со срабатыванием по срезу входного сигнала и низкий приоритет со срабатыванием по низкому уровню входного сигнала. На остальные 12 выходов портов ввода/вывода выдаются сигналы управления (уровни TTL).

Существует несколько способов реализации матрицы датчиков. На рис. 6.12а приведена схема, в которой каждый из 64 датчиков представляет собой простой переключатель, включенный последовательно с диодом, что разрешает несколько одновременных замыканий контактов на одной линии возврата. Линии сканирования порта P1 обеспечивают восемь сигналов сканирования столбцов матрицы активных высоким уровнем. Линии возврата (отклика) в тех рядах, где замкнут контакт, считываются как логические единицы. При незамкнутых контактах линии возврата подключены к общей шине через резисторы 40 кОм и считываются логическими нулями. Номинал резистора должен выбираться из расчета, чтобы сигналы на всех линиях возврата превышали порог логической единицы даже в худшем случае, когда в выбранном столбце все контакты замкнуты. Поскольку порты имеют выходы с открытым коллектором и высокоимпедансные МОП-входы, они хорошо согласуются с матрицей переключателей.

Схемы на рис. 6.12б, в, г являются вариациями на эту тему. Если входные сигналы должны быть электрически изолированы от компьютерной схемы в составе оборудования из-за помех, на линиях сканирования и возврата можно использовать оптроны, как показано на рис. 6.12б. Для других схем предполагается, что входные сигналы уже имеют уровни ТТЛ. На рис. 6.12в показаны трехстабильные буферы, которые опрашиваются низким уровнем сигнала сканирования. На рис. 6.12г показаны мультиплексоры "один из восьми", которые выбирают один из восьми входов для каждой линии возврата и опрашиваются тремя сигналами порта P1, закодированными в двоичном коде, освобождая остальные пять выводов порта P1 для других функций управления.

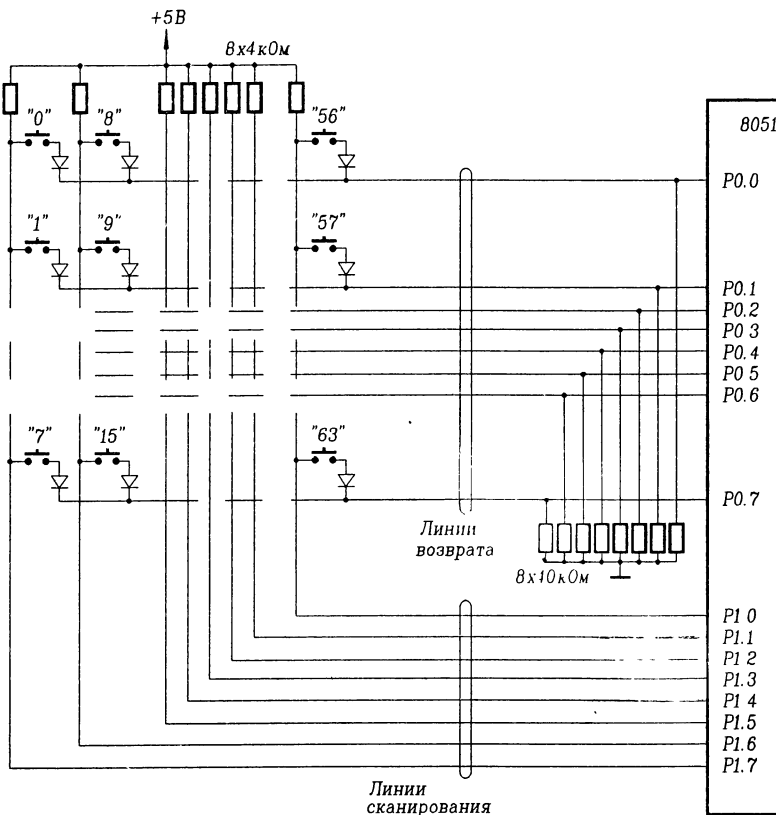


Рис. 6.12а. Использование кнопочных переключателей

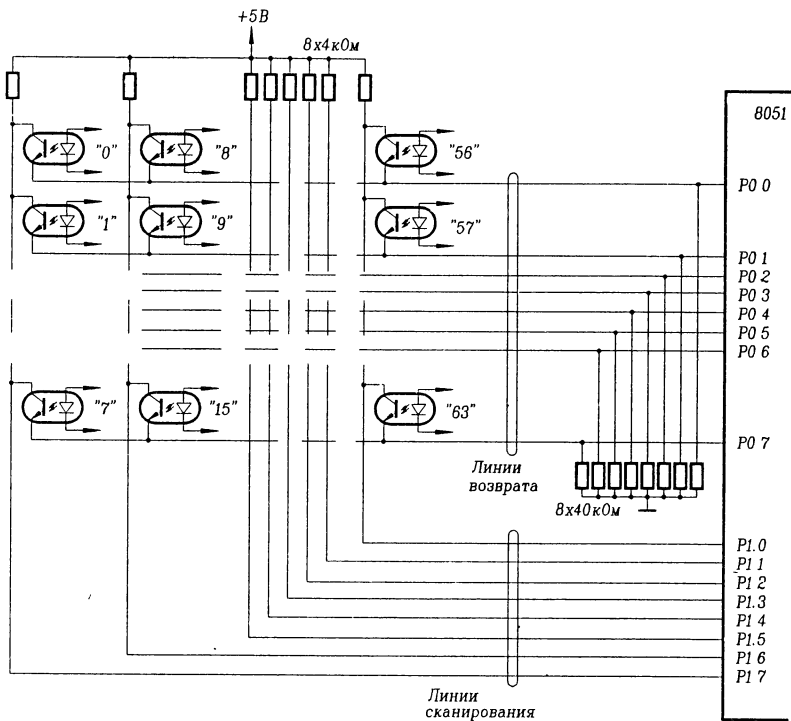


Рис. 6.12б. Использование оптронов

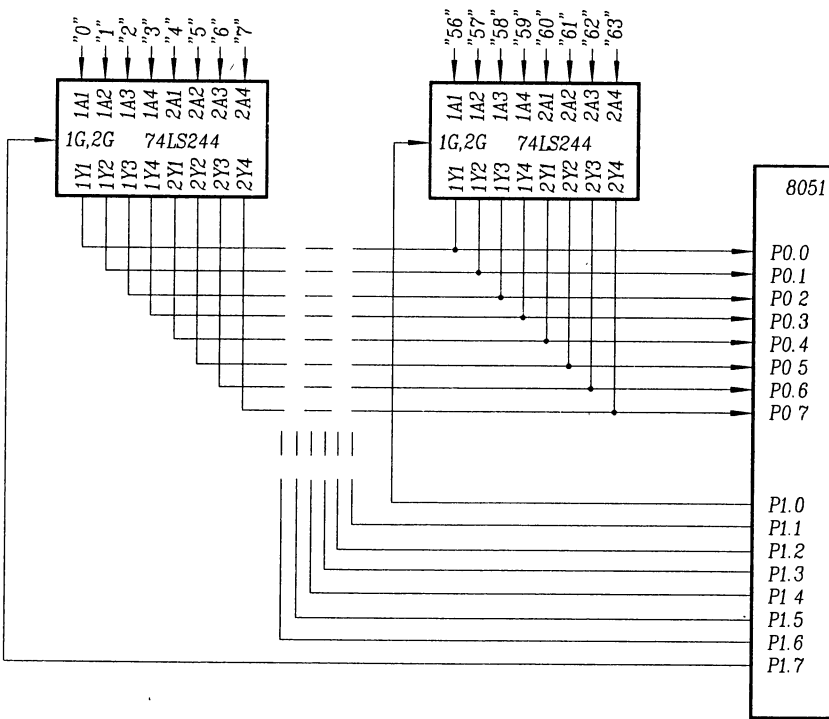


Рис. 6.12в. Использование шинных формирователей с тремя состояниями

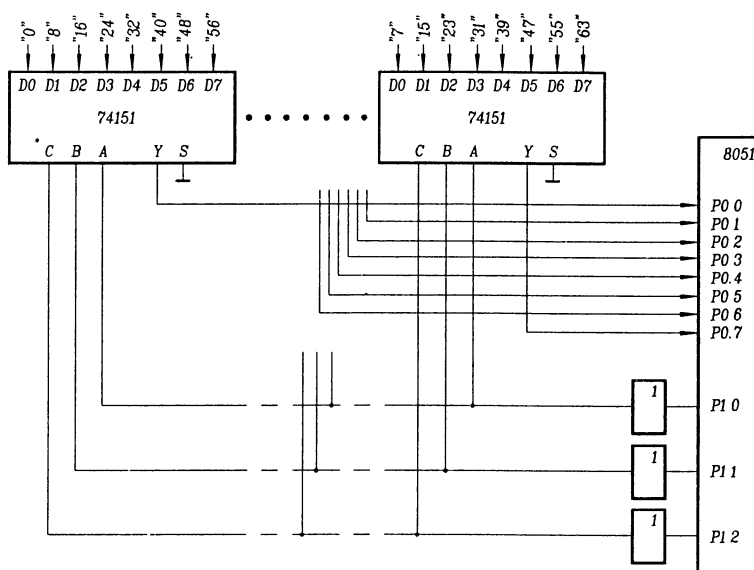


Рис. 6.12г. Использование мультиплексоров

Для исходной схемы (рис. 6.12а), на рис. 6.13 показан алгоритм сканирования матрицы датчиков. В области ОЗУ с битовой адресацией расположены две заполненные битами карты: одна отображает текущее состояние датчиков (адреса 0-63) и другая — предыдущее (адреса 64-127). Если возникает необходимость, программа может устранять дрейф контактов переключателей путем сравнения каждого бита с его предыдущим значением.

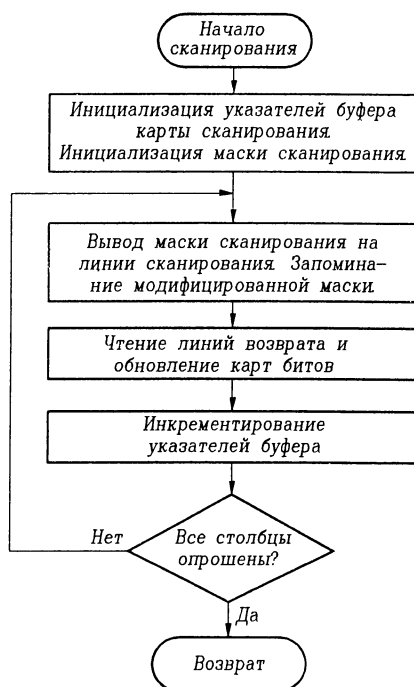


Рис. 6.13. Алгоритм сканирования матрицы датчиков

Программа, приведенная примере 2, выполняет алгоритм сканирования. Для карт битов используется положительная логика. Каждый столбец выбирается установкой одного бита в поле нулей. Что происходит после того как датчики опрошены, зависит от конкретного применения.

Пример 2

INPUT_SCAN:			; Подпрограмма чтения текущего состояния
			; 64 датчиков с сохранением в ячейках
			; ОЗУ с адресами 20H–27H
	MOV	R0, #20H	; Инициализация указателей R0 и R1
	MOV	R1, #28H	; для карт битов
	MOV	A, #80H	; Установка старшего бита аккумулятора
SCAN:	MOV	P1, A	; Возбуждение одной линии сканирования
	RR	A	; Сдвиг на следующую линию сканирования
	MOV	R2, A	; Запоминание текущей позиции
	MOV	A, P0	; Чтение линий возврата
	XCH	A, @R0	; Переключение на предыдущие биты
	MOV	@R1, A	; Сохранение предыдущего состояния
	INC	R0	; Смещение указателей
	INC	R1	
	MOV	A, R2	; Выбор следующей линии опроса
	JNB	ACC.7, SCAN	; Цикл для считывания всех 8 столбцов
	RET		

Выходные переменные

В примере 3 вычисляется выходная переменная, являющаяся комбинационной логической функцией нескольких входных переменных. Программа активизирует выход 2 порта P2, когда контакты 12, 23 и 34 замкнуты, а 45 и 56 разомкнуты.

Пример 3

```

; SET P2.2 = (12)*(23)*(34)*(/45)*(/56)
MOV    C, 12
ANL    C, 23
ANL    C, 34
ANL    C, /45
ANL    C, /56
MOV    P2.2, C

```

Промежуточные переменные

В рассматриваемом примере все 128 битов ОЗУ с битовой адресацией заняты картами входных переменных, но аккумулятор, регистр слова состояния программы PSW и регистр В обеспечивают еще 18 дополнительных флагов для промежуточных переменных.

Пусть переключатели 0-3 управляют блокировкой системы. Замыкание любого из них приводит к подавлению сигналов на некоторых выходах. При выполнении программы функция блокировки может перевычисляться при каждой активизации выхода, или она может быть вычислена однажды и сохранена (как показано в примере 4).

Пример 4

```

CALL  INPUT_SCAN
MOV   C,0           ; Вычисление F = (0)+(1)+(2)+(3)
ORL   C,1
ORL   C,2
ORL   C,3
MOV   F0,C          ; Запоминание во флаге F0
...
ANL   C,/F0
MOV   P2.0,C        ; Подавление выхода P2.0
...
ANL   C,/F0
MOV   P2.1,C        ; Подавление выхода P2.1
...

```

Функция защелки

Устройство типа защелки (триггер, реле) может быть переведено в состояние включено/выключено двумя соответствующими входными сигналами. В этом состоянии оно остается, пока не перейдет в противоположное состояние. Программно реализовать функции такого устройства очень просто. В предыдущем случае условие блокировки должно сохраняться и оставаться активным до тех пор, пока не будет нажата кнопка снятия блокировки. Любой флаг или адресуемый бит может использоваться в качестве программной защелки (см. пример 5).

Пример 5

```

; L_SET устанавливает флаг F0, если C=1
L_SET:  ORL   C,F0
        MOV   F0,C
        ...
; L_RSET сбрасывает флаг F0, если C=1
L_RSET:  CPS   C
        ANL   C,F0
        MOV   F0,C
        ...

```

Функция временной задержки

Устройство временной задержки не разрешает реакцию на входной сигнал до тех пор, пока не пройдет определенное время с того момента, как он появился или пропал. Эта задержка может моделироваться подпрограммой прерывания, которая вызывается одним из двух таймеров/счетчиков. Процедура, следующая за подпрограммой, зависит от деталей данной функции: возможны паузы из-за установления или неустановления сигналов на входах. Если подпрограмма прерывания выполняется каждые 10 мс, программа примера 6 будет очищать промежуточную переменную, установленную в основной программе, через две секунды.

Пример 6

```

JNB   USR_FLG,NXTTST
DJNZ  DLAȳ_COUNT,NXTTST

```

```

CLR    USR_FLG
MOV    DLAY_COUNT, #200
NXTTST: ...

```

Последовательный интерфейс для удаленного процессора

Когда обнаруживаются условия аварийности, определяемые некоторыми комбинациями входных сигналов (как в приведенном выше примере введения в программу функции блокировки), контроллер должен немедленно выключить аварийное устройство и/или обратиться к центральному процессору через последовательный порт. Байты, содержащие коды аварии, могут быть переданы в центральный компьютер. Например, содержимое обеих карт битов полностью может быть передано со скоростью 17 Кбайт/сек в центральный процессор для дальнейшего анализа меньше чем за одну миллисекунду. Если центральный процессор посчитает причину уважительной, у него есть возможность сигнализировать об аварии другим удаленным процессорам и указать, какая последовательность закрытия системы должна быть выполнена.

Время отклика

Отличие между последовательными схемами и программируемыми контроллерами (если рассматривать их как "черный ящик") — время реакции на изменения состояния входов. В первом случае изменение на входе немедленно начинает распространяться по системе, изменяя состояние на выходах в пределах миллисекунд. Программная реализация во втором случае действует последовательно. Изменение на входе не будет обнаружено, пока не наступит момент опроса входов и не произойдет изменения на выходе пока не выполнится соответствующая часть программы. По этой причине решающее значение имеет скорость вычислений логических функций.

Команды битового процессора выполняются за 1-2 мкс (минимальное время исполнения для большинства микроконтроллеров). Программа сканирования матрицы из 64 датчиков и все соответствующие вычисления потребуют около миллисекунды.

Таким образом быстродействие битового процессора часто намного выше, чем требуется для системы. Поэтому можно использовать остальное время на вычисление параметров обратной связи, статистическую обработку и сортировку данных, выполнение системной диагностики и т.д.

Дополнительные функции

Используя команды битового процессора можно синтезировать много полезных функций короткими последовательностями команд.

Исключающее ИЛИ. Хотя операция Исключающее ИЛИ не включена в систему команд битового процессора, операции Исключающее ИЛИ и Исключающее ИЛИ-НЕ легко реализовать с помощью команд условного перехода и логического дополнения флага переноса или любого другого бита.

```

; Наложение функции Исключающее ИЛИ на флаг переноса
; с использованием F0 в качестве входной переменной
XOR_F0: JNB    F0, XORCNT          ; JB для Исключающее ИЛИ-НЕ
        CPL    C
XORCNT:  ...

```

ХСН. Содержимое бита переноса и некоторых других битов может быть изменено с помощью аккумулятора. Биты можно сдвинуть в аккумулятор и одновременно выдвинуть из него командой сдвига через флаг переноса, однако это будет изменять его содержимое.

```
; Обмен между флагом переноса и флагом пользователя
XCHBIT:  RLC    A
          MOV    C,USR_FLG
          RRC    A
          MOV    USR_FLG,C
          RLC    A
```

Расширение битовой адресации. МК51 может прямо адресовать 144 бита общего назначения для всех команд, представленных на рис. 6.16. Подобные операции могут быть распространены на любой бит в любом месте на кристалле с некоторой потерей эффективности.

Логические операции И, ИЛИ и Исключающее ИЛИ выполняются над байтовыми переменными с использованием шести различных режимов адресации, один из которых позволяет источнику служить непосредственной маской, а приемником является прямоадресуемый байт. Любой бит может быть установлен, очищен или дополнен с помощью трехбайтовой двухцикловой команды, если маска накладывается на все биты кроме одного, который должен быть очищен или установлен.

Байтовые переменные, содержимое регистров и ячеек косвенно адресуемого ОЗУ могут пересылаться в регистр с битовым доступом (обычно это аккумулятор). Наряду с пересылкой биты могут быть проверены при выполнении условного перехода. Таким образом можно моделировать режимы адресации битов при операциях с байтовыми операндами.

Четность байтов или групп битов. Четность содержимого аккумулятора всегда находится в бите Р регистра PSW, откуда ее можно переслать в перенос для дальнейшей обработки. Применение корректирующих кодов Хэмминга и другие аналогичные приемы требуют вычисления четности для групп изолированных битов. Это можно сделать с помощью логического дополнения флага переноса или собирания этих битов в аккумулятор с последующей проверкой флага четности.

Многобайтный сдвиг и циклические коды. Хотя последовательный порт МК51 может принимать и передавать восьми- и девятибитовые данные, некоторые протоколы требуют более длинных битовых последовательностей. Большинство внешних ЗУ большой емкости и протоколов последовательной связи используют циклические коды для проверки целостности данных. Функции, обычно вычисляемые последовательно с помощью аппаратных решений, использующих сдвиговые регистры и вентили Исключающего ИЛИ, могут быть выполнены программно. Например, поскольку при приеме многобайтных данных каждый бит попадает в перенос, соответствующие разряды регистра, содержащего циклический код, могут логически дополняться в зависимости от значения переноса. В конце приема содержимое этого регистра может быть сравнено с циклическим кодом, принятым по последовательному каналу.

ISIS-II MCS-51 MACRO ASSEMBLER V1.0
 OBJECT MODULE PLACED IN : F0:CAR.HEX
 ASSEMBLER INVOKED BY : f1:asm51 car.src date(328)

LOC JBJ LINE SOURCE

```

1  :$XREF TITLE
2  :*****
3  :
4      Программа выполнения функций
5      управления и контроля
6      электрическим оборудованием
7      автомобиля:
8      - включение сигналов поворота
9      - включение мигания аварийных огней
10     - включение задних огней при нажатии педали тормоза
11     - включение габаритных огней на стоянке
12     - проверка работоспособности электрооборудования
13 :
14 :*****
15 :
16     Описание входных сигналов:
17     (Все входы представлены положительной логикой.
18     Когда контакт замкнут, на входе высокий уровень.)
19 :
0090 20 BRAKE  BIT   P1.0 ; Педаль тормоза нажата
0091 21 EMERG  BIT   P1.1 ; Включение аварийного мигания
0092 22 PARK  BIT   P1.2 ; Включены габаритные огни на стоянке
0093 23 L_TURN BIT   P1.3 ; Левый поворот включен
0094 24 R_TURN BIT   P1.4 ; Правый поворот включен
25 :
26     Описание выходных сигналов:
27     (Все выходы представлены положительной логикой.
28     Лампа включается, когда на выходе высокий уровень.)
29 :
0095 30 L_FRNT  BIT   P1.5 ; Передний указатель левого поворота
0096 31 R_FRNT  BIT   P1.6 ; Передний указатель правого поворота
0097 32 L_DASH  BIT   P1.7 ; Индикатор левого поворота на панели
00A0 33 R_DASH  BIT   P2.0 ; Индикатор правого поворота на панели
00A1 34 L_REAR  BIT   P2.1 ; Задний указатель левого поворота
00A2 35 R_REAR  BIT   P2.2 ; Задний указатель правого поворота
36 :
00A3 37 S_FAIL  BIT   P2.3 ; Индикатор дефекта в электрических цепях ламп
38 :
39     Назначение внутренних переменных
40 :
0020 41 SUB_DIV DATA 20H ; Делитель скорости прерывания
0000 42 HI_FREQ BIT   SUB_DIV.0 ; Бит генератора высокой частоты
0007 43 LO_FREQ BIT   SUB_DIV.7 ; Бит генератора низкой частоты
44 :
00D1 45 DIM      BIT   PSW.1 ; Флаг парковых огней
46 :
47 :*****
48 $EJECT
49     ORG 0000H ; Вектор сброса
0000 020040 50 L JMP INIT
51 :
000B 52     ORG 000BH ; Вектор прерывания таймера 0
000B 758CF0 53 MOV TH0, #-16 ; Загрузка старшего байта таймера
000E C0D0 54 PUSH PSW ; Сохранение регистров для дальнейшего использования
0010 0154 55 AJMP UPDATE ; (продолжение в подпрограмме)
56 :
0040 57     ORG 040H
0040 758A00 58 INIT: MOV TL0, #0 ; Загрузка младшего байта таймера
0043 758CF0 59 MOV TH0, #-16 ; Загрузка старшего байта таймера (дает период 4 мс)
0046 758961 60 MOV TMOD, #01100001B ; Режим 8-разрядного автоматически перезагружаемого
61 ; счетчика для таймера 1, для таймера 0 выбран режим 16-разрядного таймера
0049 7520F4 62 MOV SUB_DIV, #244 ; Частота прерываний 1Гц
004C D2A9 63 SETB ETO ; Использование переполнения таймера 0 для прерываний
004E D2AF 64 SETB EA ; Разрешение всех прерываний
0050 D28C 65 SETD TR0 ; сохраняется до переполнения
0052 80AE 66 SJMP $ ; Начало выполнения основной программы
0054 D52038 69 UPDATE: DJNZ SUB_DIV, TOSERV ; Выполнение теста 1 раз в секунду
0057 7520F4 70 MOV SUB_DIV, #244 ; Обеспечение задержки на 1 сек и проход по
71 ; по программе тестирования электрики:
005A 4390E0 72 ORL P1, #11100000B ; Установление высокими
005D 43A007 73 ORL P2, #00000111B ; управляющих выходов
0060 C295 74 CLR L_FRNT ; Выключение лампы L_FRNT
0062 20B428 75 JB T0, FAULT ; Опрос T0
0065 D295 76 SETB L_FRNT ; Включение лампы L_FRNT
0067 C297 77 CLR L_DASH ; Повторить для L_DASH
0069 20B421 78 JB T0, FAULT
006C B297 79 SETB L_DASH

```

```

006E C2A1 80 CLR L_REAR ; L_REAR
0070 20B41A 81 JB T0, FAULT
0073 D2A1 82 SETB L_REAR
0075 C296 83 CLR R_FRNT ; R_FRNT
0077 20B413 84 JB T0, FAULT
007A D296 85 SETB R_FRNT
007C C2A0 86 CLR R_DASH ; R_DASH
007E 20B40C 87 JB T0, FAULT
0081 D2A0 88 SETB R_DASH
0083 C2A2 89 CLR R_REAR ; R_REAR
0085 20B405 90 JB T0, FAULT
0088 D2A2 91 SETB R_REAR
92 ;
93 ; При заземленных коллекторах T0 должен быть высоким
94 ; Если да, то продолжать программу прерывания
95 ;
008A 20B402 96 JB T0, TOSERV
008D B2A3 97 FAULT: CPL S_FAULT ; Программа при обнаружении дефекта в электрике
98 ; (1 раз за секунду)
99 ;
100 ; Продолжение программы прерывания:
101 ;
102 ; 1) Вычислить слабую интенсивность
103 ; при включенных парковых лампах
008F A201 104 TOSERV: MOV C, SUB_DIV.1 ; Скважность 50%
0091 8200 105 ANL C, SUB_DIV.0 ; Уменьшить до 25%
0093 7202 106 ORL C, SUB_DIV.2 ; Восстановить до 62.5%
0095 8292 107 ANL C, PARK ; Перемножить с функцией габаритных огней
108 MOV DIM, C ; Сохранить во временной переменной DIM
109 ;
110 ; 2) Вычислить и выдать сигнал
111 ; на индикатор левого поворота на панели
0099 A293 112 MOV C, L_TURN ; Установление переноса, если включен поворот
009B 7291 113 ORL C, EMERG ; или аварийный сигнал
009D 8207 114 ANL C, LO_FREQ ; Если да, включить сигнал 1Гц
009F 9297 115 MOV L_DASH, C ; и выдать на панель
116 ;
117 ; 3) Вычислить и выдать сигнал переднего
118 ; левого поворота.
00A1 92D5 119 MOV F0, C ; Сохранить функцию в F0
00A3 72D1 120 ORL C, DIM ; и выдать сигнал на переднюю
00A5 9295 121 MOV L_FRNT, C ; лампу левого поворота
122 ;
123 ; 4) Вычислить и выдать сигнал заднего
124 ; левого поворота
00A7 A290 125 MOV C, BRAKE ; Выполнить умножение нажатия тормоза
126 ANL C, /L_TURN ; и включения поворота
00AB 72D5 127 ORL C, F0 ; Включение переменной F0 (панель)
00AD 72D1 128 ORL C, DIM ; и функции парковых огней
00AF 92A1 129 MOV L_REAR, C ; Выдача сигнала поворота
130 ;
131 ; 5) Повторить все для правых
132 ;
00B1 A294 133 MOV C, R_TURN ; Установить перенос, если поворот
00B3 7291 134 ORL C, EMERG ; или аварийность
00B5 8207 135 ANL C, LO_FREQ ; Если да, включить сигнал 1Гц
00B7 92A0 136 MOV R_DASH, C ; и выдать на панель
00B9 92D5 137 MOV F0, C ; Сохранить в F0
00BB 72D1 138 ORL C, DIM ; Учесть функцию парковых огней
00BD 9296 139 MOV R_FRNT, C ; и выдать сигнал поворота
00BF A290 140 MOV C, BRAKE ; Выполнить умножение нажатия тормоза
00C1 B094 141 ANL C, /R_TURN ; и включения поворота
00C3 72D5 142 ORL C, F0 ; Включение временной переменной F0 (панель)
00C5 72D1 143 ORL C, DIM ; и функции паркового света
00C7 92A2 144 MOV R_REAR, C ; и выдача сигнала поворота
145 ;
146 ; Восстановление регистров и возврат
147 ;
00C9 D0D0 148 POP PSW ; Восстановление PSW
00CB 32 149 RETI ; и возвращение из
150 ; подпрограммы обработки прерывания
151 END

```

XREF SYMBOL TABLE LISTING

NAME	TYPE	VALUE AND REFERENCES
BRAKE	N BSEG	0090H 20# 125 140
DIM	N BSEG	00D1H 45# 108 120 128 138 143

EA	N BSEG	00AFH	64
EMERG	N BSEG	0091H	21# 113 134
ETO	N BSEG	00A9H	63
FO	N BSEG	00D5H	119 127 137 142
FAULT	L BSEG	008DH	75 78 81 84 87 90 97#
HI_FREQ	N BSEG	0000H	42#
INIT	L BSEG	0040H	50 58#
L_DASH	N BSEG	0097H	32# 77 79 115
L_FRNT	N BSEG	0095H	30# 74 76 121
L_REAR	N BSEG	00A1H	34# 80 82 129
L_TURN	N BSEG	0093H	23# 112 126
LO_FREQ	N BSEG	0007H	43# 114 135
P1	N BSEG	0090H	20 21 22 23 24 30 31 32
P2	N BSEG	00A0H	33 34 35 37
P1	N BSEG	0090H	72
P2	N BSEG	00A0H	73
PARK	N BSEG	0092H	22# 107
PSW	N BSEG	00D0H	45 54 148
R_DASH	N BSEG	00A0H	33# 86 88 136
R_FRNT	N BSEG	0096H	31# 83 85 139
R_REAR	N BSEG	00A2H	35# 89 91 144
R_TURN	N BSEG	0094H	24# 133 141
S_FAIL	N BSEG	00A3H	37# 97
SUB_DIV	N BSEG	0020H	41# 42 43 62 69 70 104 105 106
TO	N BSEG	00B4H	75 78 81 84 87 90 96
TOSERV	L BSEG	008FH	69 96 104#
TH0	N BSEG	008CH	53 59
TL0	N BSEG	008AH	58
TMOD	N BSEG	0089H	60
TRO	N BSEG	008CH	65
UPDATE	L BSEG	0054H	55 69#

ASSEMBLY COMPLETE, NO ERRORS FOUND

ГЛАВА 7 АНАЛОГИ, ПРОИЗВОДИМЫЕ В РОССИИ И БЕЛАРУСИ

7.1. КМОП БИС 8-разрядной микроЭВМ ЭКР/КР/КА1835ВЕ49(39)

Однокристалльная 8-разрядная микроЭВМ (МКЭ), выполненная по высококачественной КМОП технологии, является функциональным аналогом микросхем фирмы Intel и отечественных микросхем 1816ВЕ39/1816ВЕ49.

МКЭ предназначена для управления конвейерами, металлорежущими станками, технологическим оборудованием, роботами, манипуляторами, может использоваться в контрольно-измерительной и бытовой технике.

В частности, серийно выпускаемые микросхемы МКЭ ЭКР/КР/КА1835ВЕ49-XXX (где XXX — порядковый номер кодировки встроенного ПЗУ программ) применяются в качестве контроллеров:

- а) клавиатуры портативной ПЭВМ "Электроника МС1504" ("ПК300");
- б) клавиатуры ПЭВМ РС АТ/ХТ (101/102 и 84 клавиши);
- в) манипуляторов "Мышь" (3,58 МГц и 6 МГц);
- г) телефонных аппаратов I и II классов;
- д) устройств охранной сигнализации;
- е) дозиметров;
- ж) многоканальной портативной радиостанции;
- з) телевизоров 6-го поколения для организации режима "кадр в кадре" (PIR).

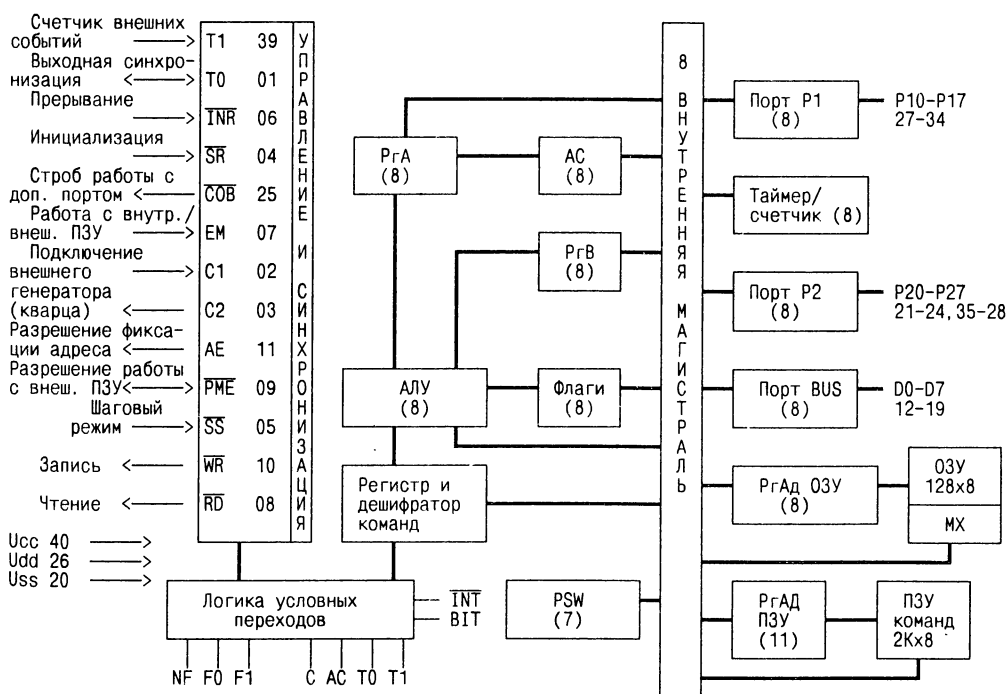


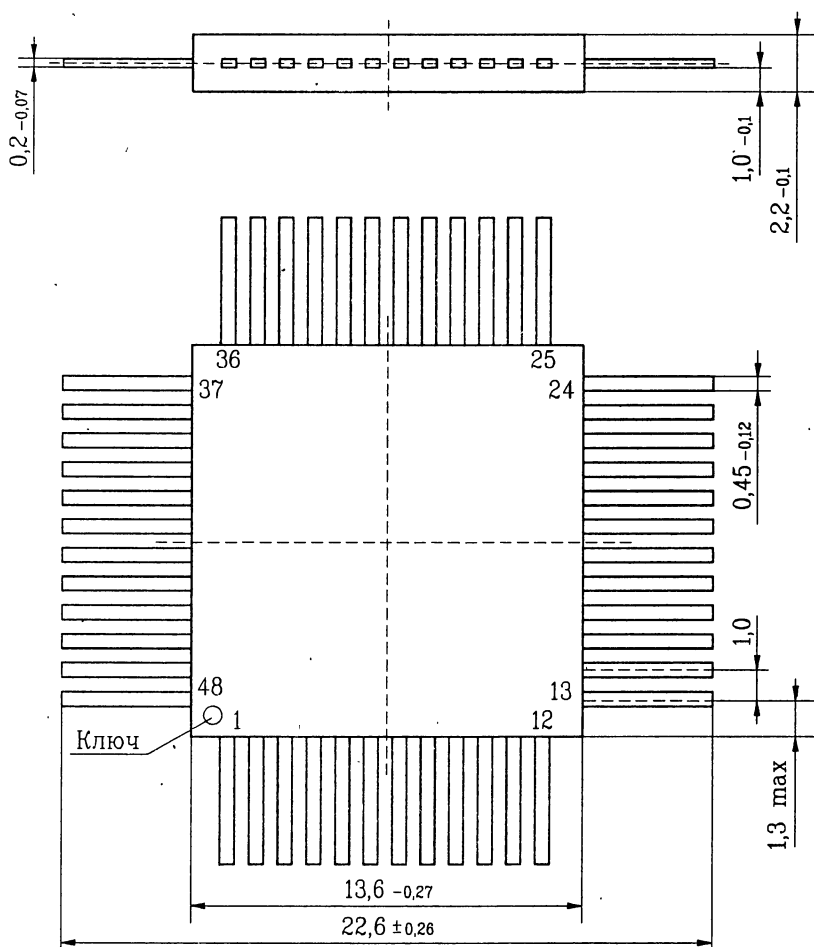
Рис. 7.1. Структурная схема БИС ЭКР/КР/КА1835ВЕ49

Микросхема имеет в своем составе все атрибуты небольшой микроЭВМ: 8-разрядное арифметическо-логическое устройство; ПЗУ программ емкостью 2 Кбайт; регистровое ОЗУ данных емкостью 128 байт; устройство управления; 8-битный программируемый таймер/счетчик событий; программно-управляемые схемы

ввода/вывода (27 линий). Организация МКЭ и ее система команд допускают в случае необходимости расширение функционально-логических возможностей МКЭ. С использованием внешних дополнительных БИС адресное пространство может быть расширено до 4 Кбайт. Архитектура МКЭ обеспечивает возможность прямой адресации внешнего ОЗУ емкостью 256 байт. Структурная схема МКЭ представлена на рис. 7.1. Основу структуры образует внутренняя двунаправленная разделяемая шина, которая связывает между собой все устройства БИС.

Конструктивно микросхемы выполнены в корпусах типа 4222.48-2 (КА1835ВЕ39/49), 2123.40-5 (КР1835ВЕ39/49, шаг выводов 2,5 мм) и 2123.40-С (ЭКР1835ВЕ39/49, шаг выводов 2,54 мм). На рис. 7.2 приведен чертеж корпуса типа 4222.48-2. Чертеж корпуса 2123.40-5 соответствует приведенному на рис. 1.16 (стр. 14), а корпуса 2123.40-С на рис. 8.3 (стр. 302).

Расположение выводов микросхемы приведено на рис. 7.3, а назначение выводов — в таблице 7.1.



Корпус 4222.48-2. Содержится драгоценный металл – золото.
Материал покрытия выводов – оловянно-свинцовый припой.
Масса микросхемы – не более 3 г.

Рис. 7.2. Корпус типа 4222.48-2

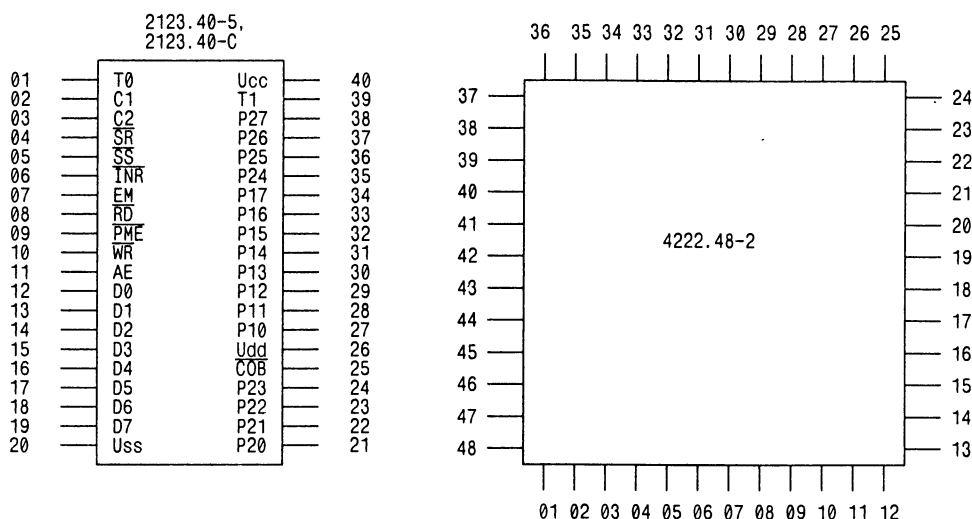


Рис. 7.3. Расположение выводов БИС ЭКР/КР/КА1835ВЕ49

Таблица 7.1

Номер вывода КА1835ВЕ49	Номер вывода КР1835ВЕ49 ЭКР1835ВЕ49	Назначение
01	—	—
02	06	Вход внешнего прерывания, \overline{INR}
03	07	Вход разрешения работы с внутренней/внешней памятью EM
04	08	Выход сигнала "Чтение", \overline{RD}
05	09	Вход/выход разрешения работы с памятью программы PME
06	10	Выход сигнала "Запись", \overline{WR}
07	11	Выход разрешения фиксации адреса AE
08	12	Вход/выход 0 разряда порта данных 0
09	13	Вход/выход 1 разряда порта данных 1
10	14	Вход/выход 2 разряда порта данных 2
11	15	Вход/выход 3 разряда порта данных 3
12, 13	—	—
14	16	Вход/выход 4 разряда порта данных 4
15	17	Вход/выход 5 разряда порта данных 5
16	18	Вход/выход 6 разряда порта данных 6
17	19	Вход/выход 7 разряда порта данных 7
18	20	Общий вывод, U_{SS}
19	21	Вход/выход 0 разряда порта P2 P20
20	22	Вход/выход 1 разряда порта P2 P21
21	23	Вход/выход 2 разряда порта P2 P22
22	24	Вход/выход 3 разряда порта P2 P23
23	25	Выход управления дополнительной шиной COB
24, 25	—	—
26	26	Вход информационный микропотребления, U_{dd}

Продолжение таблицы 7.1

Номер вывода КА1835ВЕ49	Номер вывода КР1835ВЕ49 ЭКР1835ВЕ49	Назначение
27	27	Вход/выход 0 разряда порта P1 P10
28	28	Вход/выход 1 разряда порта P1 P11
29	29	Вход/выход 2 разряда порта P1 P12
30	30	Вход/выход 3 разряда порта P1 P13
31	31	Вход/выход 4 разряда порта P1 P14
32	32	Вход/выход 5 разряда порта P1 P15
33	33	Вход/выход 6 разряда порта P1 P16
34	34	Вход/выход 7 разряда порта P1 P17
35	35	Вход/выход 4 разряда порта P2 P24
36, 37	—	—
38	36	Вход/выход 5 разряда порта P2 P25
39	37	Вход/выход 6 разряда порта P2 P26
40	38	Вход/выход 7 разряда порта P2 P27
41	39	Тестовый вход T1
42	40	Вывод питания от источника напряжения U_{CC}
43	01	Тестовый вход/выход T0
44	02	Вход подключения внешнего тактового генератора C1
45	03	Вход подключения внешнего тактового генератора C2
46	04	Вход установки в "0", \overline{SR}
47	05	Вход пошагового выполнения программ, \overline{SS}
48	—	—

Основные характеристики БИС МКЭ:

- объем внутреннего ПЗУ программ — 2 Кбайт (для ВЕ49);
- объем внутреннего ОЗУ — 128 байт;
- разрядность магистрали данных — 8 бит;
- рабочая частота — 0–8 МГц;
- напряжение питания — $5 \text{ В} \pm 10\%$;
- статический ток потребления — не более 50 мкА;
- напряжение питания при хранении информации (в статическом режиме) — 2,0–2,5 В;
- динамический ток потребления — не более 1 мА/МГц;
- диапазон рабочих температур — от -10°C до $+70^\circ\text{C}$.

Для низкочастотных (в т.ч. телефонных) применений при $F_c < 1 \text{ МГц}$ напряжение питания может быть снижено до 3.0–3,5 В, при этом динамический ток потребления не превышает 600 мкА.

7.2. КМОП БИС 8-разрядной микроЭВМ КР1835ВЕ51(31)

Однокристалльная 8-разрядная микроЭВМ КР1835ВЕ51(31) выполнена по КМОП-технологии с кремниевыми затворами, является функциональным аналогом микросхем 80C51(31) фирмы Intel и отечественных микросхем КР1830ВЕ51(31). Количество элементов в схеме — 73000.

Микросхема предназначена для использования в системах обработки цифровой информации.

Конструктивно микросхемы выполнены в корпусах типа 2123.40-С с дюймовым шагом выводов.

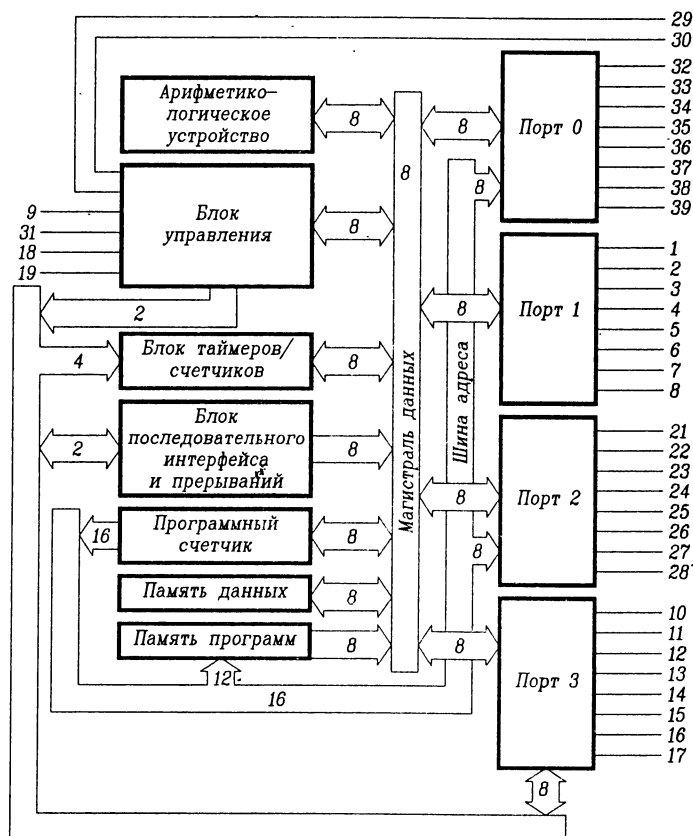


Рис. 7.4. Схема электрическая структурная KP1835BE51(31)

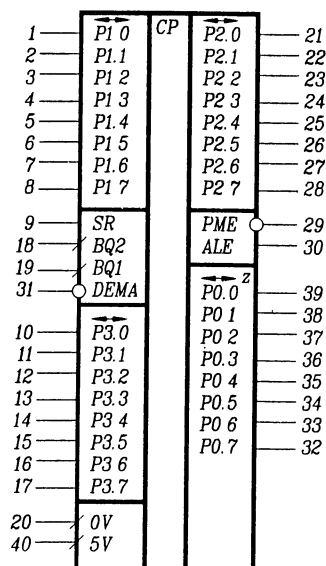


Рис. 7.5. Условное графическое обозначение KP1835BE51(31)

Основные функциональные параметры:

- Количество регистров общего назначения — 32;
- Разрядность регистров общего назначения — 8;
- Количество каналов обмена — 4;
- Частота следования импульсов тактовых сигналов — 3.5–12 МГц;
- Разрядность адреса — 16;
- Разрядность данных — 8;
- Формат команд — 1, 2, 3 байта;
- Количество (базовых) команд — 111;
- Время выполнения команд:
 - сложения регистр-регистр — 1.0 мкс;
 - сложения регистр-память — 2.0 мкс;
 - умножения/деления — 4.0 мкс;
- Объем адресуемой памяти команд — 64 Кбайт;
- Объем внутренней памяти команд — 4 Кбайт (для КР1835ВЕ51);
- Объем адресуемой памяти данных — 64 Кбайт;
- Объем внутренней памяти данных — 128 байт;
- Количество векторов прерывания — 5;
- Количество уровней прерывания — 2;
- Формат данных — бит, нибл, байт, два байта;
- Скорость обмена данных в последовательном канале ввода/вывода — 110–375000 бит/с;
- Количество таймеров-счетчиков — 2.

7.3. Однокристалльный эмулятор для микроЭВМ КР(ЭКР)1830ВЕ51

КМОП БИС однокристалльного эмулятора (ОЭ) предназначена для отладки кодировок ПЗУ в реальном устройстве на этапе его разработки (макетирования), в котором при массовом (серийном) производстве предполагается использовать микросхему ЭКР1830ВЕ51-XXX с масочно запрограммированным ПЗУ.

ОЭ функционально включает в себя микроЭВМ 1830ВЕ51 со всеми внешними выводами, а также дополнительные управляющие, адресные и информационные выходы для подключения внешнего ППЗУ программ (вместо внутреннего масочного ПЗУ) емкостью 4К*8 бит. Это позволяет разработчику устройства полностью использовать возможности всех портов ввода/вывода микросхемы 1830ВЕ51.

Конструктивно БИС ОЭ оформлена в 64-выводном планарном корпусе с 4-х сторонним расположением выводов с шагом 1 мм. Назначение выводов приведено в таблице 7.2.

Временная диаграмма функционирования ОЭ в режиме эмуляции приведена на рис. 7.6. На вывод АЕ подается напряжение высокого уровня для отключения внутренней памяти программ. Сигналом RST осуществляется сброс микросхемы при условии подачи на выводы XTAL1 и XTAL2 внешнего сигнала синхронизации. На выходы A11—A0 устанавливается начальный адрес 00H, сопровождаемый сигналом TP. Прием команды/данных осуществляется через входы D0—D7 при сопровождении сигнала BUS после снятия сигнала RST. Сигналы L2, H2 указывают на выдачу следующего истинного адреса команды/данных.

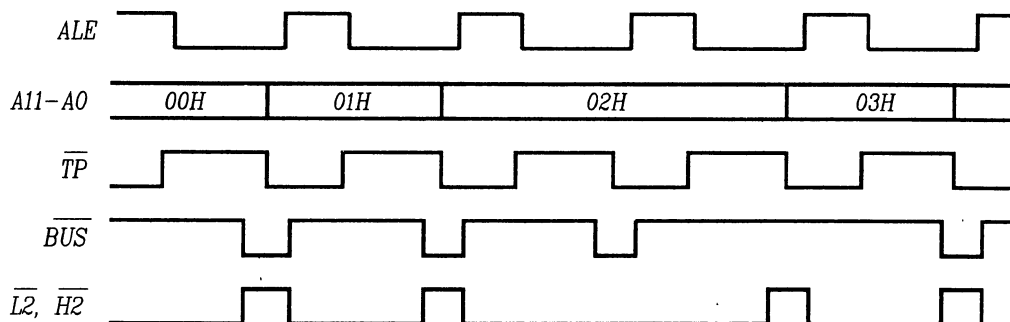


Рис. 7.6. Диаграмма функционирования ОЭ в режиме эмуляции

Таблица 7.2

№ выв.	Обозначение выводов	Назначение выводов	Тип вывода
01-03	A5-A7	Выходы адреса A5-A7	выход
14-17	A8-A11	Выходы адреса A8-A11	выход
51-63	A0-A4	Выходы адреса A0-A4	выход
04	L2	Строб фиксации младшей части адреса	выход
05-12	P1.0-P1.7	8-разрядный двунаправленный порт P1	вх./вых.
13	RST	Сигнал общего сброса	вход
18-22	P3.0-P3.4	8-разрядный двунаправленный порт P3	вх./вых.
26-28	P3.5-P3.7		
23	H2	Строб фиксации старшей части адреса	выход
24-25	D7-D6	Входы данных D7-D6	вход
31-33	D5-D3	Входы данных D5-D3	вход
35-37	D2-D0	Входы данных D2-D0	вход
29	XTAL2	Выводы для подключения кварцевого резонатора или внешнего генератора	выход
30	XTAL1		вход
34	GND	Общий вывод	
38-45	P2.0-P2.7	8-разрядный двунаправленный порт P2	вх./вых.
46	PSEN	Разрешение программной памяти	выход
47	ALE	Сигнал разрешения фиксации адреса	выход
48	TP	Выход квитирования адреса A11-A0	выход
49	BUS	Выход стробирующего сигнала при записи команд/данных	выход
50	AE	Блокировка работы с внутр. памятью	вход
51-58	P0.7-P0.0	8-разрядный двунаправленный порт P0	вх./вых.
64	UCC	Питание + 5 В	

7.4. Однокристалльные микроЭВМ ЭКР1830ВЕ31М/51М.

Микросхемы ЭКР1830ВЕ31М/51М разработаны по высококачественной КМОП технологии с проектными нормами 1,2 мкм, что позволило увеличить объем встроенных ОЗУ и ПЗУ микроЭВМ до 256*8 бит и 16К*8 бит соответственно. Количество транзисторов на кристалле около 200 000.

Электрические характеристики, система команд, назначение выводов, конструктивное исполнение и наличие внутреннего ПЗУ соответствует микросхемам ЭКР1830ВЕ31/51. Прямой зарубежный аналог отсутствует. Косвенным зарубежным аналогом (по объему встроенных ОЗУ и ПЗУ) является микросхема 83C51FB фирмы Intel.

С целью сокращения сроков и повышения качества разработки и отладки кодировок ПЗУ для микроЭВМ ЭКР1830ВЕ51М проектируется микросхема однокристалльного эмулятора. Конструктивное исполнение — 64-выводной планарный корпус с 4-х сторонним расположением выводов (аналогично однокристалльному эмулятору для микросхемы ЭКР1830ВЕ51).

7.5. Однокристалльная микроЭВМ К(Р)1850ВЕ48/50

Основные параметры микросхемы:

Тип корпуса — 2123.40-6;

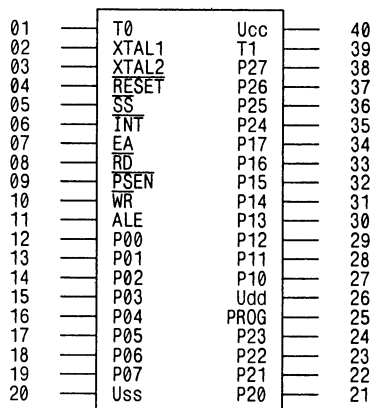
K1850 — металл-керамический;

KP1850 — 40-выводной пластмассовый;

Тактовая частота — 1-6 МГц;

Напряжение питания — 5 В;

Уровни входных и выходных сигналов соответствуют уровням ТТЛ.



Тип ИС	ОЗУ, байт	ПЗУ, байт	Аналог ф. Intel
BE35	64	-	8035
BE48	64	1K	8048
BE39	128	-	8039
BE40	256	-	8040
BE50	256	4K	8050

Рис. 7.7. Расположение выводов микроЭВМ К(Р)1850BE48/50

Возможно расширение функциональных возможностей путем подключения ИС серии 580.

В НИИ "Восток" разработаны микросхемы — однокристальные микроЭВМ — аналоги ИС фирмы Intel (США) и ИС серии 1816. Система команд полностью соответствует командам для семейства микроЭВМ 8048 фирмы Intel (США). ИС имеют в своем составе АЛУ, 8-разрядный таймер, три 8-разрядных порта ввода-вывода. Система прерываний обслуживает прерывания от двух источников (INT, таймер). Возможна работа с внешней программной памятью 2 Кбайт, внешним ОЗУ 256 байт.

7.6. Однокристальная КМОП микроЭВМ К(Р)1850BEC48/50

Основные параметры микросхемы:

Тип корпуса:

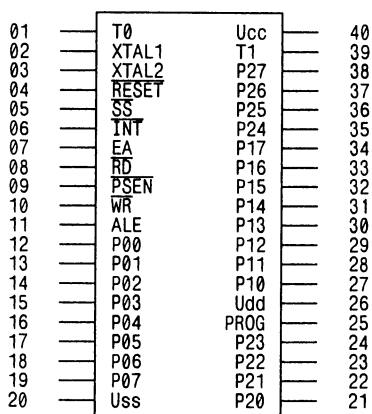
K1850 — металло-керамический 2123.40-6;

KP1850 — 40-выводной пластмассовый;

Потребляемая мощность — 5 мВт/1 МГц;

Тактовая частота — 0–11 МГц;

Напряжение питания — 5 В.



Тип ИС	ОЗУ, байт	ПЗУ, байт	Аналог ф. Intel
BEC35	64	-	80C35
BEC48	64	1K	80C48
BEC39	128	-	80C39
BEC40	256	-	80C40
BEC50	256	4K	80C50

Рис. 7.8. Расположение выводов микроЭВМ К(Р)1850BEC48/50

Уровни входных и выходных сигналов соответствуют уровням ТТЛ.

Возможно расширение функциональных возможностей путем подключения ИС серии 580.

В НИИ "Восток" разрабатываются микросхемы — однокристальные микроЭВМ в КМОП-исполнении — аналоги ИС фирмы Intel (США) и ИС серии 1816. Система команд полностью соответствует командам для семейства микроЭВМ 8048 фирмы Intel (США). ИС имеют в своем составе АЛУ, 8-разрядный таймер, три 8-разрядных порта ввода-вывода. Система прерываний обслуживает прерывания от двух источников (INT, таймер). Возможна работа с внешней программной памятью 2 Кбайт, внешним ОЗУ 256 байт.

7.7. Однокристальная микроЭВМ К(Р)1850ВЕ651

Основные параметры микросхемы:

Тип корпуса:

К1850 — металло-керамический 2123.40-6;

КР1850 — 40-выводной пластмассовый;

Потребляемая мощность — 0.2–0.6 Вт;

Тактовая частота — 3.5–12.0 МГц;

Напряжение питания — 5 В;

Уровни входных, выходных сигналов соответствуют уровням ТТЛ.

Возможно расширение функциональных возможностей путем подключения ИС серии 580.

01	—	P10	Ucc	—	40
02	—	P11	P00	—	39
03	—	P12	P01	—	38
04	—	P13	P02	—	37
05	—	P14	P03	—	36
06	—	P15	P04	—	35
07	—	P16	P05	—	34
08	—	P17	P06	—	33
09	—	RESET	P07	—	32
10	—	P30	EA	—	31
11	—	P31	ALE	—	30
12	—	P32	PSEN	—	29
13	—	P33	P27	—	28
14	—	P34	P26	—	27
15	—	P35	P25	—	26
16	—	P36	P24	—	25
17	—	P37	P23	—	24
18	—	XTAL2	P22	—	23
19	—	XTAL1	P21	—	22
20	—	Uss	P20	—	21

Тип ИС	ОЗУ, байт	ПЗУ, байт	Аналог ф. Intel
ВЕ651	128	32К	
ВЕ631	128	—	
ВЕ31*	128	—	8031

* — вариант ИС без интерфейса I²C

Рис. 7.9. Расположение выводов микроЭВМ К(Р)1850ВЕ651

В НИИ "Восток" разработаны микросхемы — однокристальные микроЭВМ — аналоги ИС фирмы Intel (США) и ИС серии 1816. Система команд полностью соответствует командам для семейства микроЭВМ 8051 фирмы Intel (США). ИС имеют в своем составе АЛУ, два 16-разрядных таймера, четыре 8-разрядных порта ввода-вывода, последовательный порт типа RS232, интерфейс I²C. Система прерываний обслуживает прерывания от пяти источников (INT0, INT1, таймеры, последовательный порт). Возможна работа с внешней программной памятью 64 Кбайт, внешним ОЗУ 64 Кбайт.

ГЛАВА 8

КМОП МИКРОКОНТРОЛЛЕРЫ ЭКР1847ВГ6. СЕМЕЙСТВО UPI-42

8.1. Общие сведения

Одновременно с внедрением МП 8088, 8086, 80186, 80286 широкое распространение получили программируемые периферийные контроллеры, предназначенные для расширения возможностей главного процессора (ГП) в различных областях.

Программируемые периферийные контроллеры содержат регистры, программируемые главным процессором в ходе начальной загрузки системы. Эти управляющие регистры позволяют настроить периферийный контроллер на работу в различных режимах. Настройки для периферийного контроллера находятся в основной памяти системы и передаются в управляющие регистры всякий раз, когда требуется изменение режима работы.

Программируемые периферийные контроллеры разрабатываются для решения сугубо специальных задач. Отдельные кристаллы разрабатываются для средств связи, устройств параллельного ввода-вывода, клавиатуры, устройств синхронизации и т. д.

С созданием универсального периферийного интерфейсного микроконтроллера (УПИМК), делается шаг к созданию интеллектуального контроллера, полностью программируемого пользователем. Это законченный однокристалльный микроконтроллер, который может быть напрямую связан с информационной шиной главного процессора. Приборы типа схемы управления принтером, устройства сканирования клавиатуры и т. п. выполненные на базе УПИМК могут быть полностью автономными, взаимодействуя с ГП только для передачи данных.

Микросхема ЭКР1847ВГ6 является одним из УПИМК и имеет следующие отличительные особенности:

- 8-разрядный процессор;
- 8-разрядная шина данных интерфейсных регистров;
- 2К*8 ПЗУ программ;
- 128*8 ОЗУ данных;
- счетчик интервалов времени/событий;
- 2 8-разрядных ТТЛ совместимых порта ввода-вывода;
- встроенный генератор тактовой частоты
- 8 МГц — максимальная рабочая частота.

БИС ЭКР1847ВГ6 является функционально-конструктивным аналогом 80С42 фирмы NEC и 8042 фирмы Intel, а по системе команд, назначению выводов и применению совместима с ИС семейства UPI-42 фирмы Intel. БИС ЭКР1847ВГ6 выполнена по КМОП-технологии, что позволило повысить эффективность и производительность при относительно низкой стоимости.

УПИМК ориентированы на преимущественное применение в системах управления различных периферийных устройств, что нашло отражение в их структуре и функциональных характеристиках. 8-разрядный процессор имеет минимальное время цикла 1,875 мкс при тактовой частоте 8 МГц и систему двух одноуровневых прерываний. Его система команд состоит из более чем 90 инструкций. Большинство команд — однобайтные и одноцикловые и совсем мало двухбайтовых. Система команд оптимизирована для работы с битами и операциями ввода-вывода. Предусмотрены специальные команды для арифметических операций в двоичном и двоично-десятичном кодах, логических операций над отдельными разрядами аккумулятора и портов ввода-вывода, табличного просмотра программ, для счетчиков циклов и программ с ветвлением по N-пути. Все вышесказанное повышает эффективность работы микроконтроллера при выполнении алгоритмов управления при сравнительно низкой эффективности вычислительных операций.

В обычной конфигурации УПИМК согласуются с системной шиной как обычные микропроцессорные периферийные устройства (рис. 8.1). Главный процессор и УПИМК образуют слабо связанную мультипроцессорную систему с прямой связью.

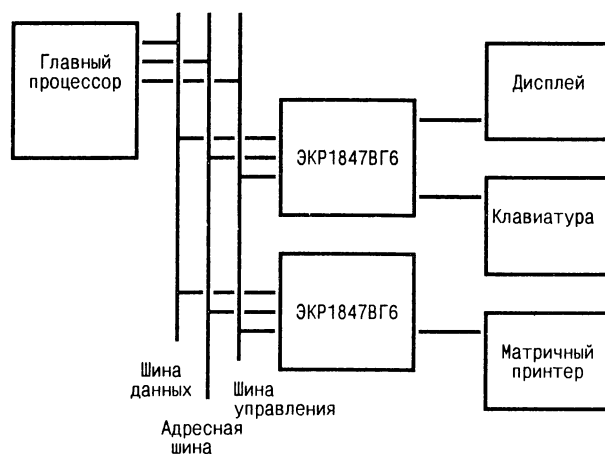


Рис. 8.1

Общие ресурсы представляют собой 3 адресуемых регистра, имеющиеся в ИС КР1847ВГ6 (в дальнейшем УПИМК):

- DBBIN - (Data Bus Buffer Input) регистр входного буфера информационной шины;
- DBBOUT - (Data Bus Buffer Output) регистр выходного буфера информационной шины;
- STATUS - регистр состояния.

Главный процессор может считывать данные из DBBOUT или записывать их в DBBIN. Состояние DBBOUT и DBBIN плюс состояние, выбранное пользователем, подается в STATUS. Главный процессор может считывать STATUS в любое время. При заполнении регистра DBBIN автоматически происходит прерывание процессора микроконтроллера (если оно разрешено) по заполнению входного буфера IBF (Input Buffer Full).

Поскольку УПИМК включает в себя законченный микроконтроллер с программной памятью, памятью данных и процессором, он может использоваться как "универсальный" контроллер. Разработчик может программировать УПИМК для управления печатью, лентопротяжным механизмом, каналами последовательной и параллельной связи и т. п. Наличие двух 8-разрядных портов ввода-вывода P1 и P2 обеспечивает возможность обмена информацией с периферией. УПИМК может также использоваться для автономной арифметической обработки данных или других задач, не требующих большого быстродействия.

С точки зрения архитектурных особенностей регистры общего назначения (РОН), косвенная адресация, программные прерывания и асинхронный ввод-вывод, страничная адресация памяти не представляют принципиально новых архитектурных решений. Конструктивной особенностью УПИМК является принцип сведения к минимуму количества разбросанных по кристаллу регистров, реализация которого повлекла за собой образование 8-уровневого стека и регистров общего назначения на общем адресном пространстве ОЗУ как сверхоперативной памяти.

Программист, работающий с УПИМК ЭКР1847ВГ6, располагает двумя программно переключаемыми банками регистров общего назначения, каждый из которых содержит по 8 регистров и расположен на общем адресном пространстве ОЗУ. Нулевой банк РОН0 занимает адреса 0...7, первый банк РОН1 занимает адреса 24...31. Регистры активного в данный момент банка РОН прямо адресуются большим количеством инструкций.

Кроме того, все ячейки ОЗУ, включая стек, адресуются косвенно с использованием нулевого или первого регистров банка РОН.

Ячейки ОЗУ с адресами 8...23 могут использоваться в качестве стека с глубиной 8, каждому уровню которого соответствуют два байта. При обращении к

подпрограмме один байт представляет младшие разряды адреса возврата, второй — содержит четыре старших разряда адреса возврата, а остальные полбайта сохраняют четыре старших разряда слова состояния программы.

При объеме ОЗУ 128*8 разрядов пользователь имеет два 8-разрядных регистровых банка с прямой адресацией регистров и 96*8-разрядную резидентную память данных с косвенной адресацией, если используется вся глубина стека, или 128*8-разрядную память данных с косвенной адресацией, если используется только один регистровый банк и не используется стек.

8-разрядные ТТЛ-совместимые порты ввода-вывода P1 и P2 обладают сходными возможностями в части фиксации. Данные статически присутствуют на выводах порта и могут быть изменены только с новой выдачей. Каждая из 16 линий портов может действовать независимо как вход или выход в зависимости от полученной команды. Порт 1 работает как статический порт независимо от режима использования УПИМК. Порт 2 является многофункциональным. Кроме работы как и порт 1 в режиме ввода-вывода, 4 младших разряда порта 2 могут использоваться как интерфейс для расширителя ввода-вывода KP580BP43, что обеспечивает получение четырех дополнительных 4-разрядных портов. 4 старших разряда могут использоваться при реализации внешних прерываний для ЦП и организации прямого доступа к памяти.

При помощи команд ввода-вывода адресуются все порты УПИМК, а также дополнительные порты ввода-вывода, которые можно реализовать на микросхеме расширителя ввода-вывода. Предусмотрена возможность выполнения логических операций И и ИЛИ над содержимым дополнительных портов и младшими четырьмя разрядами аккумулятора. Кроме двух 8-разрядных портов ввода-вывода на кристалле имеются два вывода тестирования, которые могут программно проверяться, и один из которых можно также использовать как вход счетчика внешних событий.

Входы тестирования позволяют программам разветвляться без необходимости загрузки данных через порт в аккумулятор.

Характерной особенностью архитектуры УПИМК является наличие на кристалле 8-разрядного таймера-счетчика, предназначенного для организации прерывания по его переполнению. Таймер также можно использовать для формирования комплексных временных последовательностей и в качестве счетчика внешних событий с переходом к подпрограмме обслуживания прерывания по заданному числу событий.

При возникновении прерывания УПИМК переходит на выполнение программы обработки прерывания, начальный адрес которой фиксирован в памяти программ. Адрес возврата и слово состояния программы (частично) заносятся во внутренний стек и восстанавливаются после окончания выполнения подпрограммы. Команда RETR разрешает дальнейшие прерывания.

В дополнение к такому значительному количеству линий ввода-вывода УПИМК ЭКР1847ВГ6 обладает мощным набором команд, ориентированных на внешний обмен.

Внутренняя синхронизация, с делением частоты на три по отношению к внешней, вырабатывает тактировку внутренних состояний. Сигнал внутренней тактировки с делением частоты на пять поступает на счетчик машинных циклов.

8.2 Функциональное описание

Микросхема ЭКР1847ВГ6 конструктивно выполнена в 40-выводном пластмассовом DIP-корпусе 2123.40-С с дюймовым шагом.

Условное графическое обозначение микросхемы показано на рис. 8.2, чертеж корпуса — на рис. 8.3, назначение выводов — в таблице 8.1.

ИС ЭКР1847ВГ6 (УПИМК) — программируемый периферийный контроллер, спроектированный для работы в iAPX-86, iAPX-88, MCS-85, MCS-80, MCS-51 и MCS-48 системах. Структурная схема УПИМК, показанная на рис. 8.4, является архитектурой дешевого 1-кристального микроконтроллера с памятью программ, памятью данных, процессором, вводом-выводом, таймером-счетчиком и тактовым генератором в одном 40-выводном корпусе.

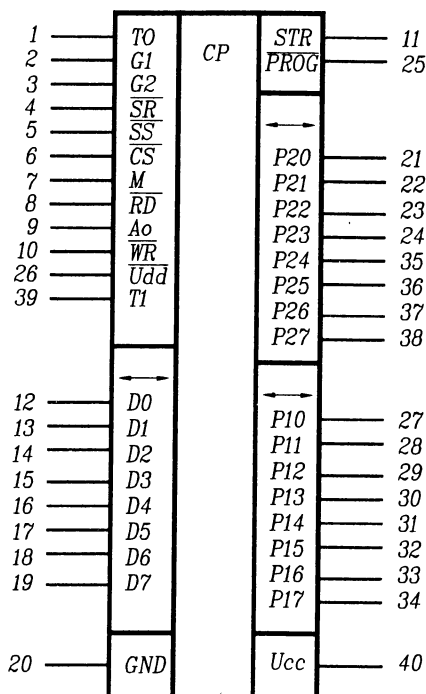


Рис. 8.2. Условное графическое обозначение

8.2.1. Процессор

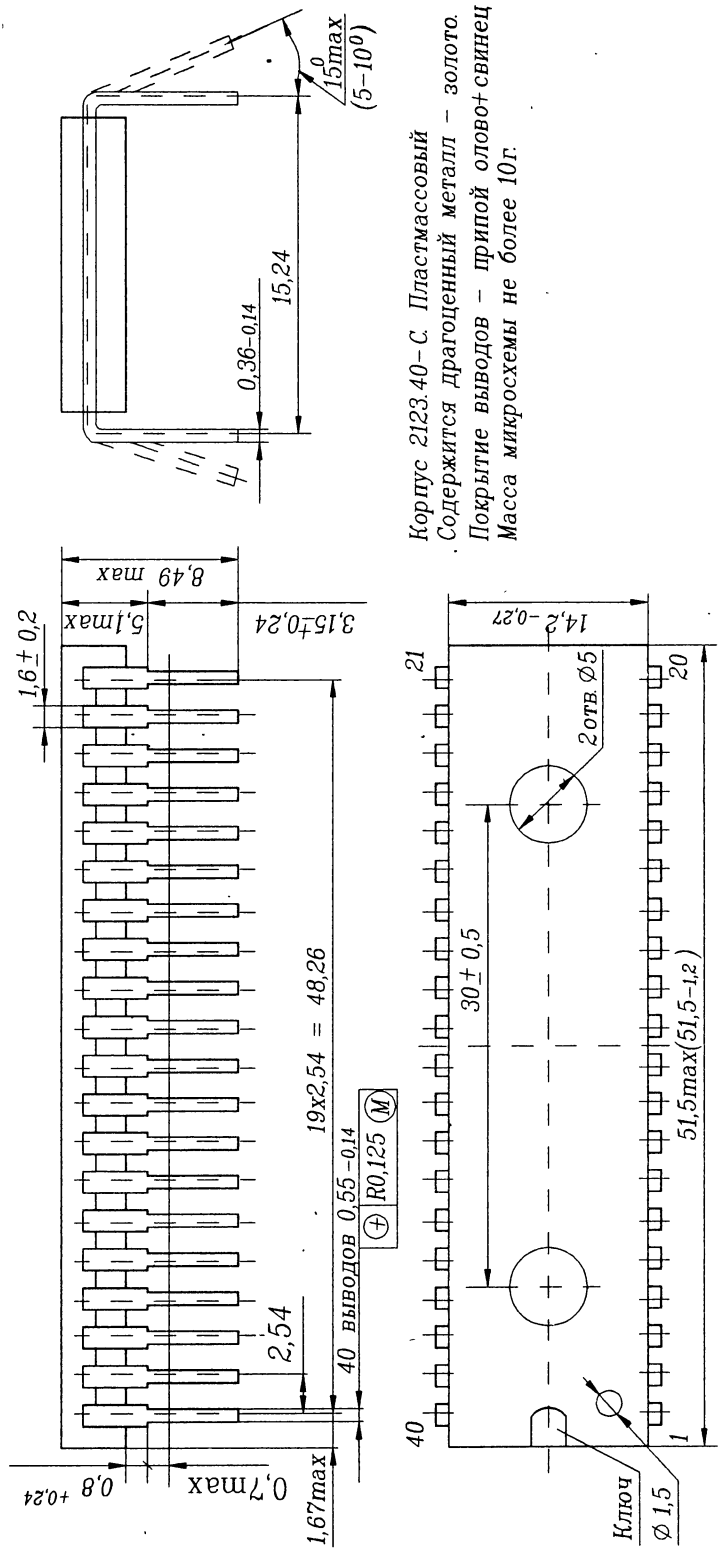
Процессор УПИМК выполняет манипулирование базовыми данными и управляет потоком данных всего однокристального микроконтроллера через внутреннюю 8-битовую информационную шину. В состав процессора входят 8-разрядное арифметико-логическое устройство (АЛУ), дешифратор команд, счетчик команд, регистр состояния программы и схема условных переходов.

АЛУ состоит из собственно арифметико-логического устройства, аккумулятора, регистра временного хранения и схемы десятичной коррекции аккумулятора. АЛУ выполняет следующие операции:

- суммирование с переносом и без переноса;
- И, ИЛИ и Исключающее ИЛИ;
- инкремент и декремент;
- битовое дополнение;
- циклический сдвиг влево или вправо;
- SWAP (перестановку старшего и младшего ниблов);
- десятичную коррекцию двоично-десятичных чисел.

Аккумулятор представляет собой 8-разрядный регистр, предназначенный для записи и хранения данных подаваемых с внутренней шины. Это наиболее важный регистр процессора. Через аккумулятор в большинстве случаев проходят данные из портов ввода-вывода в память и обратно. Он является основным источником данных для АЛУ. Данные из аккумулятора объединяются в АЛУ с данными от других источников, находящихся на внутренней шине (таких, например, как регистры или порты ввода-вывода). Результаты обработки данных в АЛУ могут передаваться на внутреннюю шину или обратно в аккумулятор.

Регистр временного хранения (Регистр В) представляет собой 8-разрядный регистр предназначенный для записи и хранения второго операнда при выполнении операций в АЛУ.



Корпус 2123.40-С Пластмассовый
Содержится драгоценный металл – золото.
Покрываете выводы – припой олово+свинец
Масса микросхемы не более 10г.

Рис. 8.3. Конструктивное исполнение микросхемы КР1847ВГ6

Таблица 8.1

Номер вывода	Обозначение (Intel)	Обозначение ЭКР1847ВГ6	Назначение	Тип
1	TEST0	T0	Тестовый вход T0 используется при выполнении команд условного перехода JT0 и JNT0.	Вх.
2	XTAL1	G1	Выводы для подключения кварцевого резонатора или LC-цепи.	Вх.
3	XTAL2	G2		Вых.
4	RESET	SR	Сброс.	Вх.
5	SS	SS	Пошаговое выполнение команд.	Вх.
6	CS	CS	Выбор кристалла.	Вх.
7	EA	M	Режим работы микросхемы.	Вх.
8	RD	RD	Чтение регистров DBBOUT, STATUS.	Вх.
9	Ao	Ao	Выборка команды/данных.	Вх.
10	WR	WR	Запись в регистр DBBIN.	Вх.
11	SYNC	STR	Строб цикла.	Вых.
12...19	D0...D7 (BUS)	D0...D7	Шина данных	Вх./вых.
20	VSS	GND	Общий.	
21...24, 35...38	P20...P27	P20...P27	Порт P2: 8 двунаправленных линий ввода-вывода. 4 младших бита (P20...P23) используются как интерфейс расширителя ввода-вывода KP580BP43. 4 старших бита (P24...P27) могут использоваться (с помощью программного обеспечения) для организации внешних прерываний и прямого доступа к памяти.	Вх./вых.
25	PROG	PROG	Выход строба для расширителя ввода-вывода.	Вых.
26	VDD	U _{dd}	Микропотребление.	Вх.
27...34	P10...P17	P10...P17	Порт P1: 8 двунаправленных линий ввода-вывода.	Вх./вых.
39	TEST1	T1	Тестовый вход T1 используется при выполнении команд условного перехода JT1 и JNT1 и как вход счетчика внешних событий после выполнения команды STRT CNT.	Вх.
40	VCC	U _{cc}	Напряжение питания.	

Схема десятичной коррекции предназначена для обработки данных, представленных в двоично-десятичном коде.

Если операции, такие как суммирование или сдвиг, занимают больше 8 битов, флаг переноса (CARRY — CY) используется как индикатор. Аналогично, в процессе десятичной коррекции и других двоично-десятичных операций флаг дополнительного переноса (AUXILIARY CARRY — AC) может быть установлен и использован далее. Эти флаги есть часть слова состояния программы (PSW).

Дешифратор команд представляет собой программируемую логическую матрицу (ПЛИМ). В процессе выборки команды операционный код каждой программной инструкции хранится и декодируется дешифратором команд. Дешифратор формирует управляющие сигналы, осуществляющие выполнение этой команды.

Счетчик команд (СК) предназначен для формирования текущего адреса местонахождения команды в памяти программ. УПИМК ЭКР1847ВГ6 имеет 11-битовый счетчик команд (СК), который обеспечивает прямую адресацию 2048 ячеек ПЗУ. Счетчик команд всегда содержит адрес следующей выполняемой команды и обычно он последовательно увеличивается на единицу при выборке каждой последующей команды.

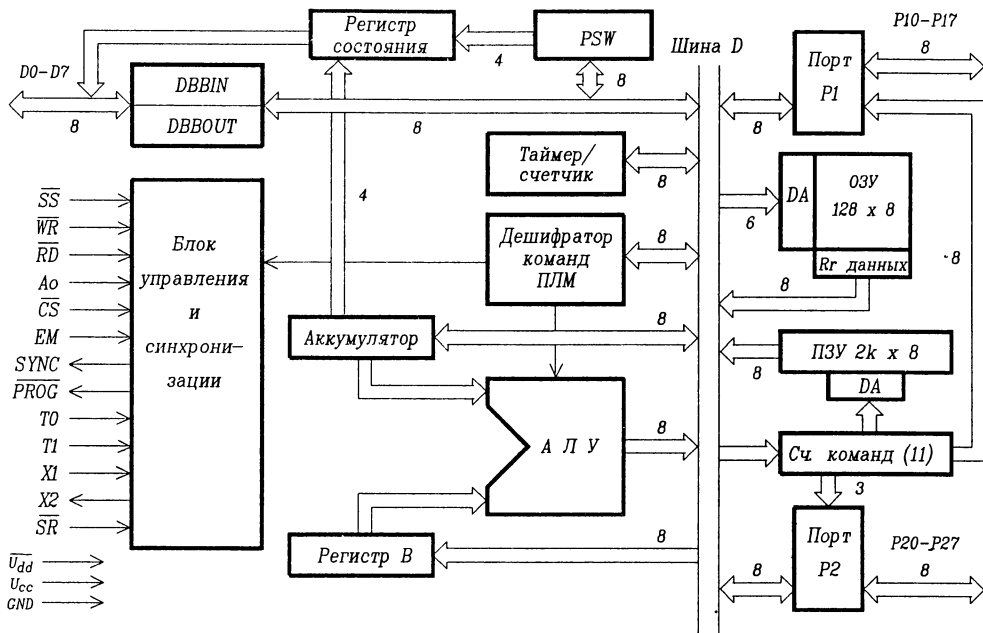


Рис. 8.4. Структурная схема УПИМК

При передаче управления от основной программы к подпрограмме или программе прерывания содержимое СК изменяется на указанное в программе значение. При этом текущее значение СК сохраняется в стеке, чтобы после окончания выполнения подпрограммы продолжить выполнение основной программы. Счетчик программ устанавливается в начальное (нулевое) состояние сигналом сброса \overline{SR} .

Слово состояния программ (PSW) представляет собой 8-разрядный регистр (рис. 8.5), который используется для хранения основной информации о выполнении программы. Помимо 3-разрядного указателя стека PSW включает в себя следующие флаги: CY — флаг переноса; AC — флаг дополнительного переноса; F0 — флаг пользователя (флаг 0); BS — флаг выбора банка регистров.

Сохранение в стеке					Указатель стека		
CY	AC	F0	BS	—	S2	S1	S0
7	6	5	4	3	2	1	0

Рис. 8.5. Формат PSW

PSW может быть загружено в аккумулятор или из аккумулятора командами $MOV\ A, PSW$ и $MOV\ PSW, A$ соответственно. Возможность непосредственной записи PSW позволяет легко восстанавливать состояние устройства после режима уменьшенного энергопотребления.

4 старших разряда PSW (4, 5, 6 и 7) сохраняются в стеке с каждым вызовом подпрограммы или вектора прерывания. После возвращения в основную программу эти биты восстанавливаются по команде $RETR$ и не восстанавливаются по команде RET .

Разряды PSW определяются следующим образом:

— биты 0—2 — разряды S0, S1, S2 указателя стека;

- бит 3 — не используется;
- бит 4 — выбор банка РОН:
 - 0 - 0-ой банк
 - 1 - 1-ый банк

— бит 5 — разряд флага F0. Этот флаг может быть установлен или очищен командами CPL F0 и CLR F0. Команды условного перехода могут проверять его установку. Он также может быть использован во время передачи данных на внешнюю шину.

— бит 6 — дополнительный перенос (AC — Auxillary Carry). Этот флаг может устанавливаться при выполнении команды ADD и используется при десятичной коррекции командой DA A.

— Бит 7 — перенос (CY — Carry). Этот флаг устанавливается если в результате выполнения предыдущей команды произошло переполнение аккумулятора.

Схема условных переходов предназначена для формирования сигналов управления ветвлением программы при выполнении команд условных переходов.

В таблице 8.2 приведен список внутренних состояний, которые могут быть проверены и условий по которым осуществляется переход.

Таблица 8.2

Устройство	Мнемоника		Условие перехода (Переход если:)
Аккумулятор	JZ	адрес	Все биты = 0
	JNZ	адрес	Любой бит не ноль
Бит аккумулятора	JBx	адрес	Бит "x" = 1
Флаг переноса	JC	адрес	Флаг переноса = 1
	JNC	адрес	Флаг переноса = 0
Флаг пользователя	JF0	адрес	Флаг F0 = 1
	JF1	адрес	Флаг F1 = 1
Флаг таймера	JTF	адрес	Флаг таймера = 1
Тестовый вход T0	JT0	адрес	T0 = 1
	JNT0	адрес	T0 = 0
Тестовый вход T1	JT1	адрес	T1 = 1
	JNT1	адрес	T1 = 0
Флаг входн. буфера	JNIBF	адрес	Флаг IBF = 0
Флаг вых. буфера	JOBF	адрес	Флаг OBF = 1

8.2.2. Память программ

Память программ предназначена для хранения и считывания команд, которые поступают в процессор и управляют процессом обработки информации. Микросхема ЭКР1847ВГ6 имеет 2048 8-битовых слов резидентного ПЗУ программной памяти. Каждая из этих ячеек памяти прямо адресуема 11-битовым счетчиком программ.

Все выборки из внутренней памяти не сопровождаются никакими внешними сигналами, генерируемыми микроконтроллером, кроме сигнала STR, который вырабатывается независимо от режима использования УПИМК и является идентификатором машинного цикла.

Память, расположенная на кристалле микросхемы, занимает адресное пространство от 000H до 7FFH. Карта распределения памяти программ представлена на рис. 8.6.

Память программ делится на страницы емкостью по 256 байт в каждой.

Для доступа к памяти программ как к таблицам данных служат команды обращения к текущей странице памяти программ MOVР и к третьей странице MOVР3, которые позволяют считывать байт из программной памяти в аккумулятор.

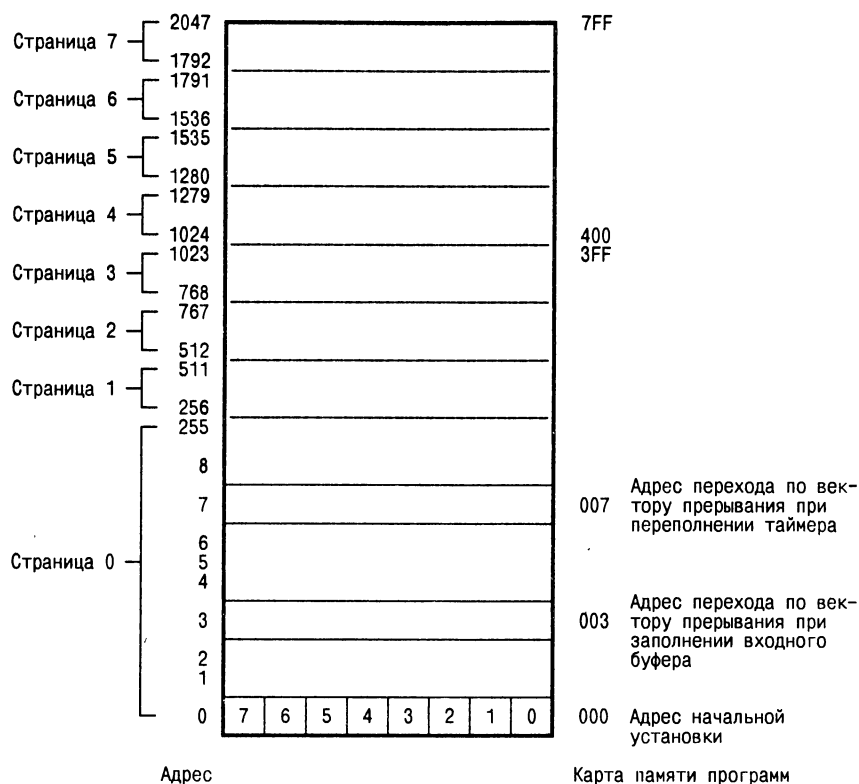


Рис. 8.6. Карта программной памяти

В памяти программ имеются три ячейки специального назначения:

— адрес 0 - адрес, по которому выполняется первая выборка инструкции по сигналу сброса \overline{SR} ;

— адрес 3 - начальный адрес подпрограммы, вызываемой по заполнению входного буфера (IBF — Input Buffer Full) при условии, что прерывание разрешено;

— адрес 7 - начальный адрес подпрограммы, вызываемой по переполнению таймера-счетчика при условии, что прерывание разрешено.

При подаче на систему сигнала сброса \overline{SR} выполнение программы начинается по адресу 0. Обычно команды в памяти программ выполняются последовательно. Нормальная последовательность команд может быть нарушена при поступлении прерывания по IBF или по таймеру, а также когда встречается команда вызова подпрограммы или перехода. Прерывание по IBF (если оно разрешено) автоматически переносит управление на ячейку с адресом 3, а прерывание по таймеру — на ячейку с адресом 7.

Все команды условного и безусловного перехода ограничены областью текущей страницы емкостью 256 байт (т.е. изменяют только биты 0...7 счетчика команд). Если команда перехода находится в ячейке 255 на странице, это указывает что адресат находится на следующей странице.

8.2.3. Память данных

Память данных предназначена для записи, хранения и считывания данных, получаемых в процессе обработки информации. Память данных, состоящая из 128 ячеек ОЗУ, разбита на два банка регистров общего назначения (РОН) с адресами от 00H до 07H — банк РОН 0 и с адресами от 18H до 1FH — банк РОН 1. Карта распределения памяти данных изображена на рис. 8.7. Переключение банков осуществляется программным путем с помощью команд SEL RB0, SEL RB1. Сигнал сброса \overline{SR} автоматически выбирает регистры банка РОН 0. Когда выбран банк

РОН 0, обращения к R0—R7 производятся в соответствии с адресами 0...7 в коде команды. Если выполняется команда SEL RB1 (Select Register Bank 1 — Выбор банка RОН1) и после нее идет команда обращения к R0—R7, то выбираются адреса 24...31.

Если банк 1 не используется, регистры 24...31 могут использоваться как дополнительное ОЗУ данных.

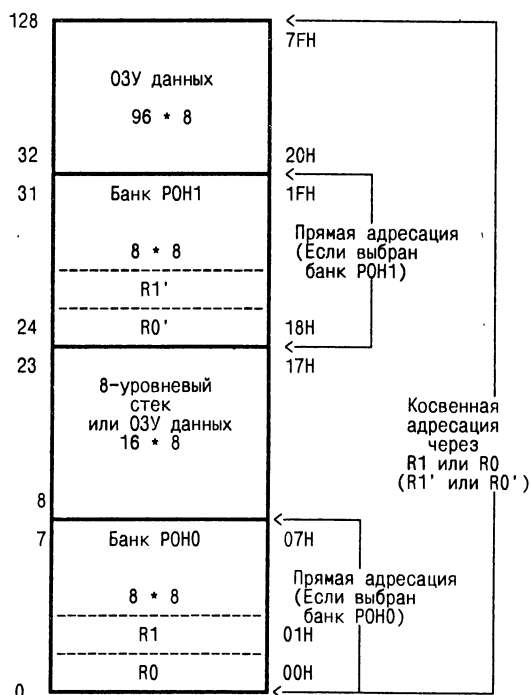


Рис. 8.7. Карта распределения памяти данных

Ячейки ОЗУ с адресами от 20Н до 7FH используются только как ОЗУ данных. Восьмиуровневый 16-разрядный стек с адресами от 08Н до 17Н адресуется указателем стека из PSW. Организация стека показана на рис. 8.8 (адрес ОЗУ приведен в десятичном коде). Кроме того, с использованием косвенной адресации ячейки стека могут адресоваться как ОЗУ данных. Размер доступной памяти ОЗУ может изменяться в зависимости от количества занятых ячеек в стеке и количества используемых рабочих регистров.

Для записи и выборки данных из ОЗУ применяются два вида адресации: прямая и косвенная (регистровая). Независимо от типа адресации три младших разряда кода команды указывают один из восьми регистров РОН R0...R7 с учетом принадлежности к ранее выбранному банку регистров. При использовании команд с прямой адресацией указанный регистр является источником или приемником данных, а при использовании команд с косвенной адресацией указанный регистр содержит адрес данных (в качестве регистров косвенного адреса используются только R0 и R1). С помощью косвенной адресации можно адресоваться к любой ячейке памяти данных. Программист по своему усмотрению может заносить данные для хранения в любые ячейки-регистры банков РОН, стек, а также имеет доступ к любой из ячеек ОЗУ посредством косвенной адресации.

Стек допускает 8 уровней вложения стандартной подпрограммы, т.е. подпрограмма может вызвать следующую подпрограмму, которая может вызвать третью, и так до восьми раз. Неиспользованные адреса стека могут быть задействованы под ОЗУ данных. Каждый неиспользованный уровень вложения подпрограммы обеспечивает два дополнительных адреса ОЗУ данных.

Указатель стека		Адрес ОЗУ
111		23
		22
110		21
		20
101		19
		18
100		17
		16
011		15
		14
010		13
		12
001		11
		10
000	PSW(4-7)	9
	СК(8-10)	
	СК(4-7)	8
	СК(0-3)	

Рис. 8.8. Организация стека

Стек состоит из 16 адресов памяти данных. Эти адреса (8...23) используются для хранения 11 бит счетчика команд и 4 бит слова состояния программы (PSW).

При прерывании или переходе к подпрограмме происходит сохранение содержимого СК в одной из 8 регистровых пар стека.

3-битовый указатель стека, являющийся частью слова состояния программы (PSW) определяет стековую пару, которая используется в данный момент. Указатель стека сбрасывается сигналом \overline{SR} в $\overline{00H}$, что соответствует 8 и 9 адресу ОЗУ.

Первое прерывание или вызов подпрограммы передает содержание СК и PSW в ячейки с адресами 8 и 9 в формате, показанном на рис. 8.8. Указатель стека автоматически осуществляет приращение адреса на 1, тем самым указывая на адреса 10 и 11 при следующем вызове подпрограммы.

Число вложений подпрограмм внутри подпрограмм может достигать 8 уровней без переполнения стека. При переполнении самый ранний сохраненный адрес (ячейки с адресами 8 и 9) будет утерян и заменен на новый, а указатель стека изменится с $\overline{07H}$ на $\overline{00H}$. Таким же образом будет изменяться содержимое разрядов от $\overline{00H}$ до $\overline{07H}$ в указателе стека при дальнейшем его переполнении.

Подпрограмма должна оканчиваться командой RET или RETR. Каждая из этих команд автоматически осуществляет отрицательное приращение указателя стека и передает содержимое соответствующей пары регистров ОЗУ в счетчик команд (СК).

8.2.4. Устройство генератора и синхронизации

Внутренний генератор УПИМК вырабатывает сигналы, обеспечивающие управление выполнением команд. Генератор реализован на кристалле микроконтроллера за исключением источника опорной частоты, в качестве которого может быть использован кварцевый резонатор, LC-цепочка или внешний источник синхроимпульсов.

Схема синхронизации состоит из генератора, счетчика стадий и счетчика циклов (рис. 8.9). На рис. 8.10 показан процесс выполнения одноцикловой команды по стадиям (S1...S5).

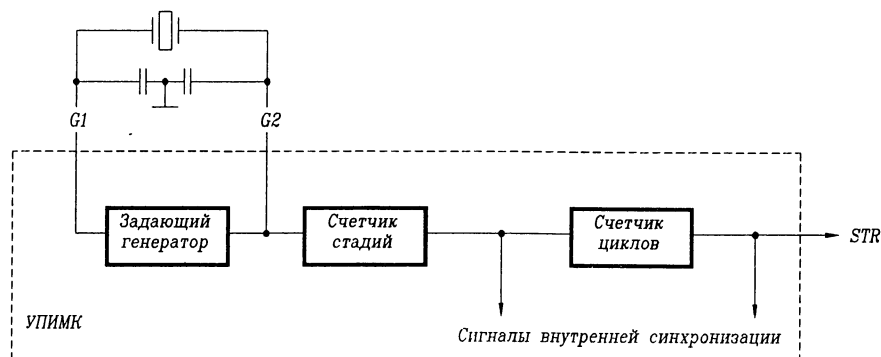


Рис. 8.9. Схема синхронизации

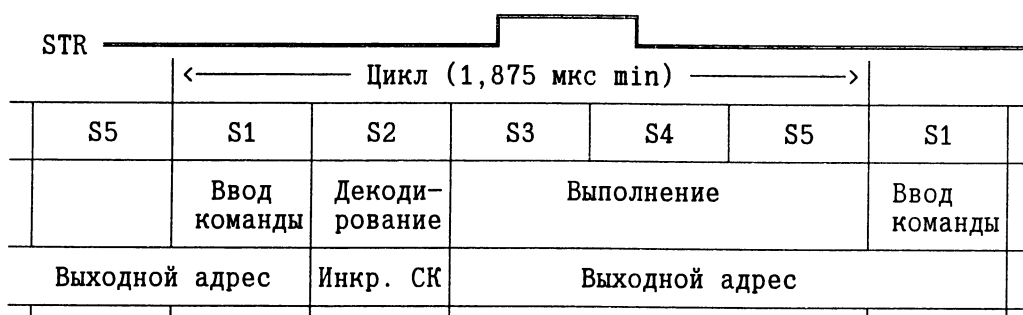


Рис. 8.10. Тактовое выполнение команды

Выводы G1 и G2 УПИМК (см. таблицу 8.1) являются входом и выходом каскада большого усиления. Кварц или LC-цепочка, подключенные между G1 и G2, обеспечивают обратную связь и фазовый сдвиг для схемы генератора. Рекомендуемые схемы подключения кварцевого резонатора и LC-цепочки показаны на рис. 8.11. В зависимости от значения опорной частоты емкости конденсаторов при использовании кварцевого резонатора подбираются опытным путем в пределах 0...30 пф.

Тактовая частота для работы устройства вырабатывается путем деления на 3 счетчиком стадий основной частоты генератора.

Командный цикл состоит из 5 стадий, как показано на рис. 8.10 и в таблице 8.3. Совмещение операций адресации и выполнения команд, показанное на рис. 8.10, способствует быстрому выполнению команды.

Выходной сигнал счетчика стадий делится на 5 в счетчике циклов. Полученный таким образом сигнал называется STR, он определяет машинный цикл и его можно снять с вывода STR. Этот сигнал можно использовать для синхронизации работы внешней схемы и как сигнал генератора тактов. Он также используется для пошаговой синхронизации.

Подключение кварцевого резонатора в качестве источника опорной частоты обеспечивает высокую скорость работы микроконтроллера и стабильность частоты задающего генератора. Рекомендуемая частота кварцевого резонатора для ЭКР1847ВГ6 — 1...8 МГц. Если стабильность частоты и высокая скорость работы не являются решающим фактором, то вместо кварцевого резонатора можно использовать LC-цепочку (рис. 8.11). В этом случае при $C=20$ пф и $L=130$ мкГн обеспечивается частота генерации 3 МГц, а при $L=45$ мкГн — 5 МГц. В общем случае частота генерации вычисляется по формуле, приведенной на рис. 8.11.

Таблица 8.3. Диаграмма выполнения команд

ЦИКЛ 1					
КОМАНДЫ	S1	S2	S3	S4	S5
IN A, P	Выборка команды	Инкрем.СК	—	Инкремент Т/С	—
OUTL P, A	Выборка команды	Инкрем.СК	—	Инкремент Т/С	Вывод на порт
ANL P, DATA	Выборка команды	Инкрем.СК	—	Инкремент Т/С	Чтение порта
ORL P, DATA	Выборка команды	Инкрем.СК	—	Инкремент Т/С	Чтение порта
MOVD A, P	Выборка команды	Инкрем.СК	Выдача КОП и адреса	Инкремент Т/С	—
MOVD P, A	Выборка команды	Инкрем.СК	Выдача КОП и адреса	Инкремент Т/С	Вывод данных на P2 (0...3)
ANLD P, A	Выборка команды	Инкрем.СК	Выдача КОП и адреса	Инкремент Т/С	Вывод данных
ORLD P, A	Выборка команды	Инкрем.СК	Выдача КОП и адреса	Инкремент Т/С	Вывод данных
J(условие перехода)	Выборка команды	Инкрем.СК	Определение условия перехода	Инкремент Т/С	—
MOV STS, A	Выборка команды	Инкрем.СК	—	Инкремент Т/С	Обновление Status (4...7)
IN A, DBB	Выборка команды	Инкрем.СК	—	Инкремент Т/С	—
OUT DBB, A	Выборка команды	Инкрем.СК	—	Инкремент Т/С	Вывод на порт
STRT T STRT CNT	Выборка команды	Инкрем.СК	—	—	Старт счетчика
STOP TCNT	Выборка команды	Инкрем.СК	—	—	Останов счетчика
EN I	Выборка команды	Инкрем.СК	—	Разрешение прерывания	—
DIS I	Выборка команды	Инкрем.СК	—	Запрещение прерывания	—
EN DMA	Выборка команды	Инкрем.СК	—	DMA разрешено DRQ очищено	—
EN FLAGS	Выборка команды	Инкрем.СК	—	Вывод OBF, TBF на порт разрешен	—

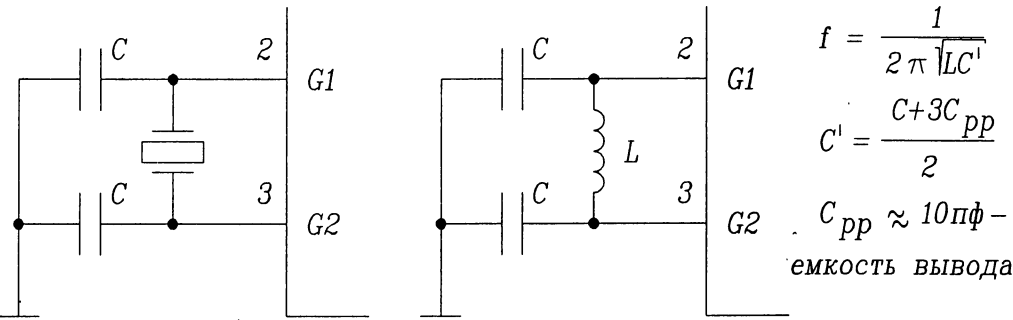


Рис. 8.11. Схемы подключения кварцевого резонатора и LC-цепочки

Продолжение таблицы 8.3

ЦИКЛ 2					
КОМАНДЫ	S1	S2	S3	S4	S5
IN A, P	—	Чтение порта	—	—	—
OUTL P, A	—	—	—	—	—
ANL P, DATA	Выборка непоср. данных	—	Инкрем.СК	Вывод на порт	—
ORL P, DATA	Выборка непоср. данных	—	Инкрем.СК	Вывод на порт	—
MOVD A, P	—	Чтение P2 (0...3)	—	—	—
MOVD P, A	—	—	—	—	—
D P, A	—	—	—	—	—
ORLD P, A	—	—	—	—	—
J(условие перехода)	Выборка непоср. данных	—	Обновление СК	—	—

Принятые сокращения: Т/С — таймер-счетчик, СК — счетчик команд, КОП код операции.

В качестве источника опорной частоты может быть также использован внешний тактовый сигнал, при этом следует иметь ввиду, что входной сигнал не является ТТЛ совместимым. Частота сигнала может находиться в пределах 0 — 8 МГц, а сам сигнал подается на вывод G1. Для согласования ТТЛ сигналов со входом G1 микроконтроллера вводится буферно-резисторная цепь (рис. 8.12) с тем, чтобы уровень логической "1" был не ниже 3,8 вольт. Аналогичным образом можно согласовать ТТЛ сигналы со входами \overline{CS} , \overline{RD} , \overline{WR} , A_0 .

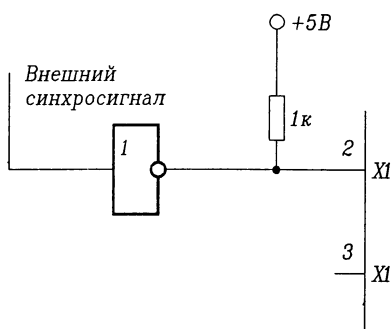


Рис. 8.12. Подключение внешнего источника тактового сигнала

8.2.5. Таймер-счетчик (Т/С)

УПИМК имеет резидентный 8-разрядный таймер-счетчик, имеющий несколько режимов работы, выбираемых программно. В режиме таймера он может вырабатывать точные интервалы времени от 60 мкс. до 15.36 мс. В режиме счетчика могут подсчитываться и использоваться для управления ходом программы такие внешние события как количество переключений различных устройств или число поступивших тактов.

Структурная схема таймера-счетчика показана на рис. 8.13. 8-разрядный регистр используется для подсчета импульсов либо от внутреннего генератора либо от внешнего источника. Счетчик может предварительно устанавливаться командой MOV T, A и читаться командой MOV A, T. Эти команды передают содержимое аккумулятора в счетчик и наоборот. Счетчик останавливается сигналом сброса \overline{SR} и командой STOP TCNT и остается в состоянии останова до поступления команды пуска таймера START T или пуска счетчика START CNT. После запуска, достигнув максимального значения (FFH) счетчик переполняется в ноль и продолжает увеличение до тех пор, пока не поступит сигнал сброса \overline{SR} или команда останова STOP TCNT.

В результате переполнения происходит установка флага таймера (TF) и вырабатывается запрос прерывания. Состояние флага переполнения может быть проверено командой условного перехода JTF. Флаг сбрасывается после выполнения команды JTF или поступления сигнала \overline{SR} .

Переполнение таймера фиксируется RS-триггером (рис. 8.14) и суммируется схемой ИЛИ с запросом прерывания от входного буфера (IBF). Прерывание по таймеру может быть разрешено или запрещено независимо от прерывания по IBF командами EN TCNT и DIS TCNT. Если прерывание разрешено, то переполнение таймера вызывает обращение к подпрограмме в ячейке 7 памяти программ, в которой находится начальный адрес программы обслуживания таймера. Если прерывания по таймеру и по IBF наступают одновременно, то источником прерывания определяется IBF и обращение будет к ячейке 3. Прерывание по таймеру фиксируется триггером, ожидая пока обслуживается входной регистр шины данных (DBBIN — Data Bus Buffer INput) и сразу же распознается после возвращения из программы обслуживания IBF. Состояние ожидания прерывания по таймеру сбрасывается в момент вызова программы обработки прерывания по таймеру, а также командой DIS TCNT и сигналом сброса \overline{SR} .

Режим счетчика. Команда STRT CNT соединяет вывод T1 со входом счетчика (рис. 8.13) и разрешает счет. Эта команда не очищает счетчик. Счетчик инкрементируется при переходе сигнала на входе T1 от высокого уровня к низкому. Минимально допустимая длительность сигналов низкого и высокого уровней составляет соответственно t_{cu} и $t_{cu}/5$ (1,875 мкс и 375 нс при $f_{G1} = 8$ МГц); t_{cu} — длительность машинного цикла, рассчитываемая по формуле: $t_{cu} = 15/f_{G1}$, где f_{G1} — частота генератора тактовых импульсов УПИМК. Максимальная частота счета — один импульс счета за три командных цикла (178 КГц при $f_{G1} = 8$ МГц). Минимальная частота счета не ограничена.

Режим таймера. Команда STRT T соединяет внутренний задающий генератор со входом счетчика и разрешает счет. Входной сигнал для счетчика получается из сигнала STR делением его на 32 (рис. 8.13). Эта команда не очищает регистр таймера. Интервалы времени, получаемые таким образом, имеют значения от 60 мкс до 15,36 мс при $f_{G1} = 8$ МГц. Интервалы более 15,36 мс могут быть получены путем программного накопления нескольких переполнений счетчика. Для получения интервалов длительностью менее 60 мкс на вход T1 можно подавать синхроимпульсы нужной частоты и работать со счетчиком в режиме подсчета событий. Внешней синхрочастотой может быть сигнал STR деленный на три и более.

Вход T1 счетчика. Вывод T1 — многофункциональный. Он автоматически определяется как тестируемый вход при поступлении сигнала сброса \overline{SR} и может быть проверен командами условного перехода.

В другом режиме вывод T1 используется как вход 8-разрядного счетчика (рис. 8.13). Команда STRT CNT управляет внутренним переключателем, который соединяет T1 через детектор перепада уровня с 8-разрядным внутренним счетчиком. Эта команда не препятствует проверке T1 посредством команд условного перехода.

В режиме счетчика состояние T1 проверяется в каждом командном цикле. После определения на входе T1 высокого уровня переход на низкий уровень вызывает увеличение содержимого счетчика на 1. Время перехода сигнала T1 из "1" в "0" — не более t_{G1} , где $t_{G1} = 1/f_{G1}$.

Счетчик останавливается командой STOP TCNT (Stop Timer/Counter). После выполнения этой команды вывод T1 функционирует как тестируемый вход.

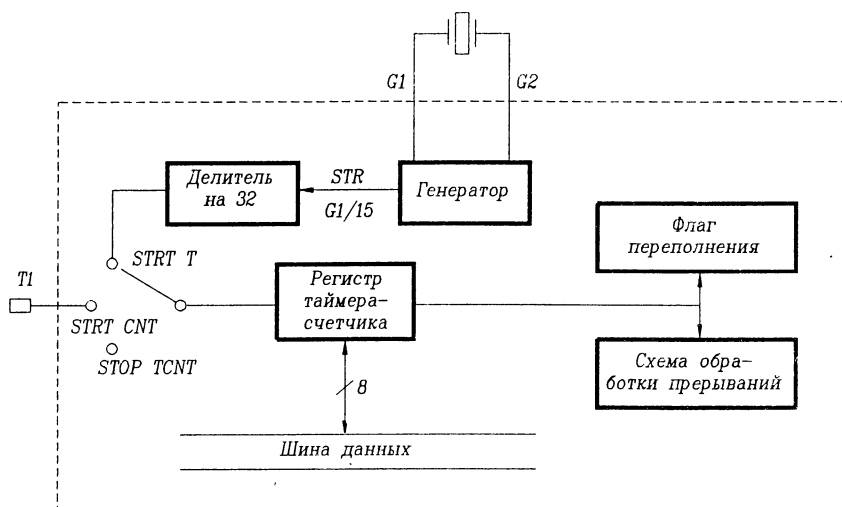


Рис. 8.13. Структурная схема таймера-счетчика

ЭКР1847ВГ6 имеет два тестируемых входа — $T\emptyset$ и $T1$. В нормальном режиме состояние каждого из этих выводов может быть непосредственно проверено с помощью следующих команд перехода:

- $JT\emptyset$ — переход, если $T\emptyset = 1$;
- $JNT\emptyset$ — переход, если $T\emptyset = 0$;
- $JT1$ — переход, если $T1 = 1$;
- $JNT1$ — переход, если $T1 = 0$;

Входы $T\emptyset$ и $T1$ — ТТЛ совместимые. Состояние внешнего сигнала подсоединенного к одному из этих выводов определяется во время выполнения команды условного перехода. Путь выполнения программы будет изменяться в зависимости от состояния проверяемого сигнала.

8.2.6. Система прерываний

УПИМК имеет следующие внутренние прерывания:

- прерывание по заполнению входного буфера (IBF);
- прерывание по переполнению таймера.

IBF-прерывание вызывает обращение по адресу $\emptyset3$ памяти программ, а прерывание по таймеру — по адресу $\emptyset7$. IBF-прерывание разрешается командой $EN I$ и запрещается командой $DIS I$. Прерывание по переполнению таймера/счетчика разрешается и запрещается командами $EN TCNTI$ и $DIS TCNTI$ соответственно.

На рис. 8.14 показана организация внутренних прерываний. Запрос на прерывание по IBF вырабатывается всякий раз, когда оба сигнала \overline{WR} и \overline{CS} находятся в низком состоянии (на рис. 8.14 показаны инверсные значения этих сигналов). Запрос прерывания по IBF сбрасывается только при обращении к программе обслуживания IBF, так как команда $DIS I$ не снимает ожидание прерывания по IBF.

Если IBF-прерывание разрешено и происходит запрос прерывания по IBF, то процесс выполнения прерывания начнется, как только завершит выполнение текущая команда. При этом будет иметь место следующая последовательность событий:

- обращение к ячейке памяти программ с адресом $\emptyset3$;
- содержимое счетчика программ и разряды 4—7 слова состояния программы (PSW) записываются в стек;
- указатель стека увеличивается на 1.

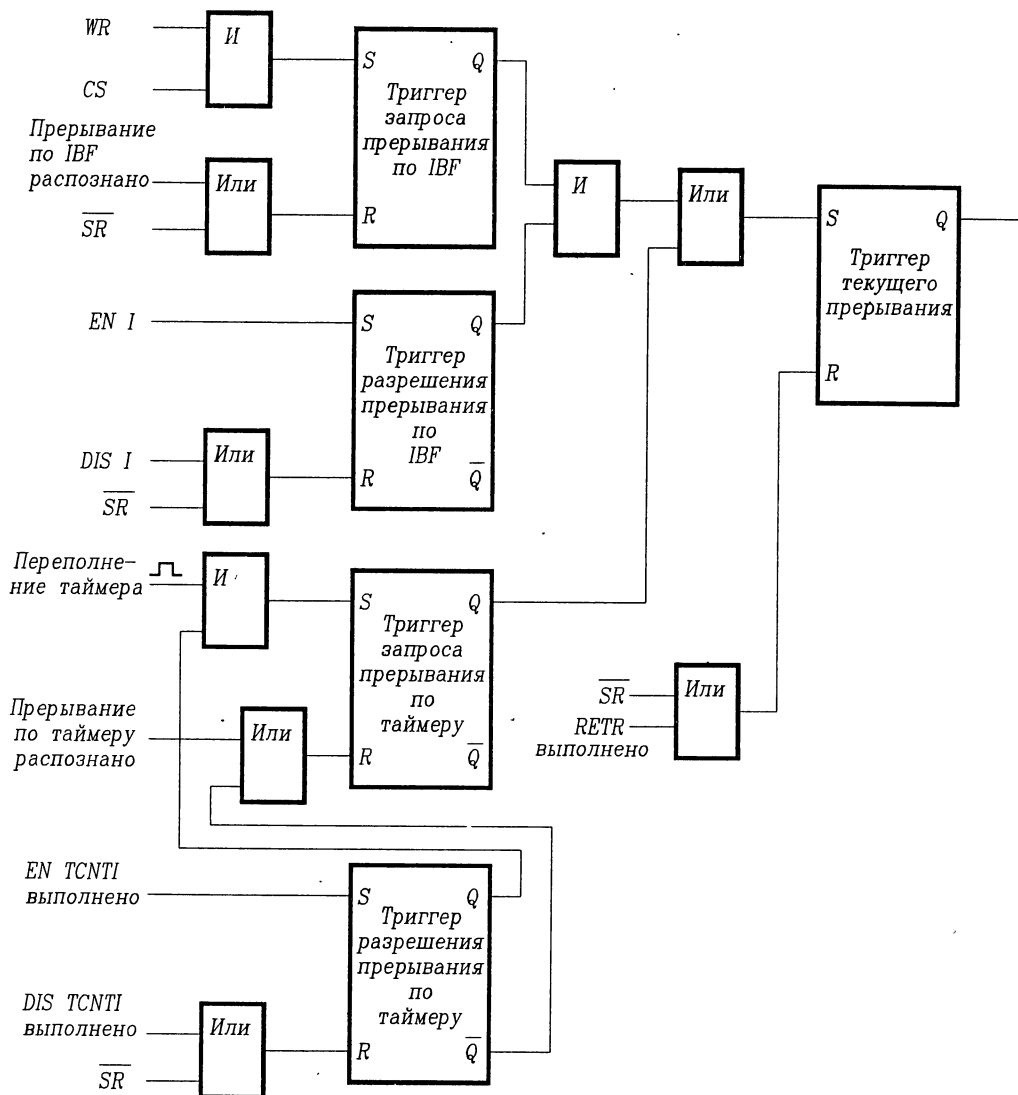


Рис. 8.14. Структурная схема организации прерываний.

Ячейка 03 памяти программ содержит начальный адрес области программ, где хранится программа обслуживания IBF-прерывания. Завершает процедуру обслуживания прерывания команда $RETR$, которая передает управление основной программе. Эта команда восстанавливает программный счетчик и 4—7 разряды PSW , обеспечивая тем самым восстановление рабочего банка регистров общего назначения $POH0$ или $POH1$. Команда $RETR$ также разрешает обработку нового прерывания.

Прерывание по переполнению таймера разрешается командой $EN\ TCNTI$ и запрещается командой $DIS\ TCNTI$. Если прерывание разрешено, то оно происходит при переполнении регистра таймера-счетчика. При этом происходит обращение к ячейке памяти 07 и далее процесс проходит аналогично случаю прерывания по IBF.

Время ожидания обслуживания прерывания складывается из времени выполнения текущей команды, времени распознавания прерывания и обращения к адрес-вектору. Максимальное время задержки составляет 7 циклов, минимальное — 4 цикла.

Последовательность выполнения. Прерывания могут разрешаться или запрещаться программно с помощью команд EN I, DIS I, EN TCNTI, DIS TCNTI. Сигнал \overline{SR} также запрещает прерывания. Запрос прерывания должен быть снят перед выполнением команды RETR, иначе программа обработки будет запущена вновь. Поэтому низкие уровни на входах \overline{WR} и \overline{CS} также не должны держаться дольше, чем продолжительность выполнения программы обработки прерывания.

Система прерываний УПИМК одноуровневая. При обнаружении любого прерывания все последующие запросы прерываний будут сохраняться триггером, но не активизироваться до выполнения команда RETR, снова разрешающей обработку прерываний. Это происходит в начале второго цикла команды RETR. Если прерывания по IBF и по таймеру поступают одновременно, то прерывание по IBF распознается и выполняется первым, а прерывание по таймеру ожидает выполнения команды RETR.

Внешние прерывания. Прерывание от внешнего источника может быть получено с помощью таймера-счетчика, работающего в режиме счетчика внешних событий. Предварительно счетчик устанавливается в FFH и выполняется команда EN TCNTI. При первом изменении уровня на входе T1 с высокого на низкий происходит переполнение счетчика и выполняется прерывание по таймеру. Как и в предыдущем случае, если прерывание по IBF наступает во время обслуживания прерывания по таймеру, оно останется в ожидании до тех пор, пока не завершит работу выполняемая подпрограмма обслуживания прерывания, обязательно заканчивающаяся командой RETR.

8.2.7. Прерывания головной системы и DMA

Используя команду EN FLAGS можно создать два внешних прерывания для главного процессора. Эта команда задействует две линии ввода-вывода порта P2: P24 и P25 (рис. 8.15). P24 становится выходом запроса прерывания по OBF (Output Buffer Full — выходной буфер заполнен) для головной системы, P25 — выходом прерывания по незаполнению входного буфера. Эти прерывания отражают внутреннее состояние флага OBF и инвертированного флага IBF — \overline{IBF} . Они запрещаются записью "0" в выходы P24 и P25 и вновь разрешаются записью "1". Изначально прерывания разрешены, так как после включения микросхемы выходы портов P1 и P2 находятся в состоянии "1". Действие команды EN FLAGS отменяется только аппаратным сбросом \overline{SR} .

Для управления DMA* (Direct Memory Access — прямой доступ к памяти) используются выходы P26 и P27 порта P2, которые становятся доступны по команде EN DMA. P26 становится линией запроса DMA — DRQ (DMA Request), а P27 — линией подтверждения DMA — \overline{DACK} (DMA ACKnowledge). Запрос DMA вызывается программной записью "1" в P26. Контроллер DMA передает данные в регистр данных DBBIN используя сигнал \overline{DACK} , который действует как сигнал выбора кристалла. Действие команды EN DMA отменяется только сигналом сброса \overline{SR} .

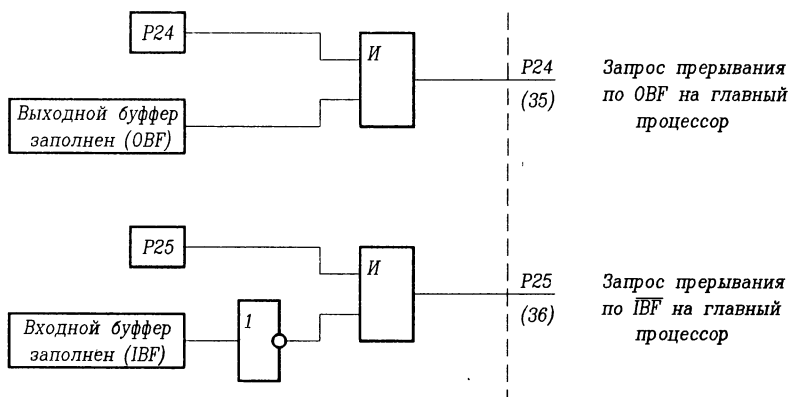
8.2.8. Сброс

Вход сброса \overline{SR} используется для внутренней инициализации микроконтроллера. Подключение емкости 1 мкФ между входом \overline{SR} и землей обеспечивает автоматическую инициализацию при включении (рис. 8.16). Для этого внутри УПИМК имеется резистор сопротивлением 100 Ом для зарядки емкости и триггер Шмидта, генерирующий сигнал инициализации.

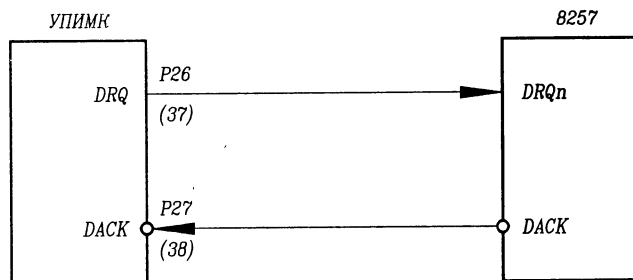
При автоматической инициализации по включению сигнал \overline{SR} должен находиться в состоянии низкого уровня по крайней мере 10 мс для стабилизации переходных процессов. При использовании внешнего формирователя, сигнал \overline{SR} должен находиться в состоянии низкого уровня не менее 8 командных циклов. На рис. 8.16 показана схема подключения ТТЛ источника внешнего сброса. Входной сигнал сброса \overline{SR} выполняет следующие функции:

* В ИС ЭКР1847ВГ6 клавиатурного контроллера команда EN DMA не используется.

- запрещает прерывания;
- сбрасывает программный счетчик в "0";
- очищает указатель стека;
- очищает регистр состояния STATUS и флаги;
- очищает таймер и флаг таймера;
- останавливает таймер;
- выбирает банк PОН0;
- устанавливает Порты 1 и 2 в режим входа.



а)



б)

Рис. 8.15. Организация внешних прерываний а), и DMA б)

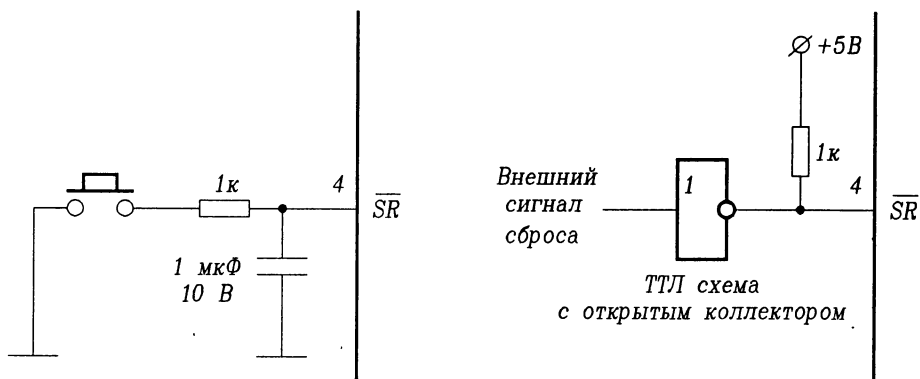


Рис. 8.16. Организация внешнего сброса

8.2.9. Буфер шины данных

Два 8-разрядных буферных регистра шины данных DBBIN и DBBOUT необходимы для временного хранения команд и данных, проходящих между УПИМК главным процессором (ГП). Данные передаются или принимаются через буферные регистры, под управлением команд ввода или вывода ГП. При этом используются четыре управляющих сигнала, поступающие от главного процессора:

- Ao — адресный вход, указывающий что передается: команда или данные;
- \overline{CS} — выбор кристалла;
- \overline{RD} — строб чтения;
- \overline{WR} — строб записи.

Передача может быть реализована с использованием программного управления УПИМК или без него посредством разрешения или запрещения внутреннего прерывания УПИМК. Внутри микроконтроллера данные передаются между буферными регистрами шины данных и аккумулятором под программным управлением асинхронно работе внешнего процессора. Это позволяет программным средствам УПИМК управлять периферийными задачами независимо от главного процессора, который в это время может обслуживать основную систему.

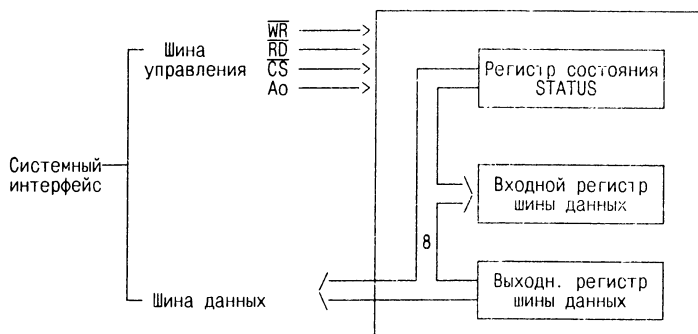


Рис. 8.17. Регистры буфера шины данных

На рис. 8.17 показана внутренняя конфигурация регистров буфера шины данных. Данные хранятся в двух 8-разрядных буферных регистрах DBBIN и DBBOUT. Эти регистры доступны внешнему процессору с помощью сигналов \overline{WR} и \overline{RD} . Шина данных — это двунаправленная шина с тремя состояниями, которая может быть непосредственно подсоединена к 8-разрядной микропроцессорной системе. Четыре линии управления (\overline{WR} , \overline{RD} , \overline{CS} , Ao), используются внешним процессором для передачи данных в регистр DBBIN и чтения регистра DBBOUT.

Входной регистр шины данных DBBIN записывается главным процессором и читается в аккумулятор с помощью программного обеспечения УПИМК. Этот регистр не может быть прочитан главным процессором.

Выходной регистр шины данных DBBOUT записывается с помощью программного обеспечения УПИМК и читается главным процессором. Этот регистр не может быть считан в аккумулятор.

8-разрядный регистр состояния STATUS используется для отображения состояний регистров буфера шины данных (рис. 8.18). Восемь флагов состояний определены следующим образом:

— OBF (Output Buffer Full — заполнение выходного буфера). Этот флаг автоматически устанавливается, когда в регистр DBBOUT загружены данные и очищается, когда главный процессор читает этот регистр данных.

— IBF (Input Buffer Full — заполнение входного буфера). Этот флаг устанавливается, когда главный процессор записывает какое-либо значение в регистр DBBIN и очищается, когда УПИМК передает содержимое этого регистра данных в свой аккумулятор.

— F0. Это флаг общего применения, который может быть установлен или очищен программными средствами УПИМК. Флаг используется для передачи информации о состоянии УПИМК главному процессору.

— F1 — команда/данные. Этот флаг устанавливается в состояние линии A0, когда главный процессор записывает значение в регистр DBBIN. Флаг F1 может быть также очищен или установлен программными средствами УПИМК.

— ST4...ST7. Эти биты состояния определяются пользователем с помощью команды MOV STS, A.

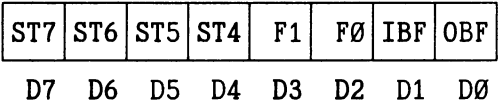


Рис. 8.18 Состояние шины данных при чтении регистра STATUS

Таблица 8.4. Управление передачей данных

CS	RD	WR	Ao	
0	0	1	0	Чтение DBBOUT
0	0	1	1	Чтение STATUS
0	1	0	0	Запись данных в DBBIN
0	1	0	1	Запись команды в DBBIN
1	X	X	X	Запрещение обращения к шине данных

Чтение данных. В таблице 8.4 показано состояние входных сигналов, соответствующее чтению DBBOUT. Когда CS, Ao и RD находятся в низком уровне, содержимое регистра DBBOUT выводится на линии D0—D7 шины данных, а флаг OBF очищается.

Главный процессор использует сигналы CS, Ao, WR и RD для управления передачей данных между регистром DBBOUT и основной системой. Следующие операции производятся под управлением головного процессора.

Чтение регистра STATUS. В таблице 8.4 показана логическая последовательность требуемая для чтения регистра состояния. Когда CS и RD находятся в низком уровне, а Ao в высоком, содержимое 8-разрядного регистра состояния выводится на линии D0—D7 шины данных.

Запись данных. Когда CS и WR находятся в состоянии низкого уровня, содержимое системной шины данных записывается в DBBIN, устанавливается флаг IBF и генерируется прерывание, если оно разрешено.

Запись команд. Во время выполнения любой команды записи (таблица 8.4) состояние входа Ao записывается в регистр состояния в область флага F1. Этот бит используется для указания содержащейся в DBBIN информации — команда это (Ao=1) или данные (Ao=0).

Временные диаграммы режимов чтения и записи представлены на рис. 8.19а и 8.19б. Числовые значения указанные в диаграммах не зависят от тактовой частоты УПИМК, т. е. остаются неизменными во всем допустимом диапазоне частот (0...8 МГц) работы микроконтроллера.

8.2.10. Системный интерфейс 8080

На рис. 8.20 показано соединение УПИМК со стандартной шиной системы 8080. Линии данных D0...D7 образуют двунаправленный порт с тремя состояниями, непосредственно соединенный с системной шиной данных. Шинный интерфейс УПИМК обеспечивает нагрузочную способность 400 мкА, достаточную для малых

систем. Для больших систем необходимо подключение через шинные формирователи.

Четыре сигнала управления обеспечивают передачу данных и состояния:

— \overline{WR} — сигнал записи. Используется при передаче данных из системной шины в регистр DBBIN. \overline{WR} устанавливает флаг F1 регистра состояний.

— \overline{RD} — сигнал чтения. Используется при передаче данных из регистра DBBOUТ или регистра состояний на системную шину данных.

— \overline{CS} — выбор кристалла. Разрешает подключение УПИМК к общей шине.

— A_0 вход адресации. Производит выбор регистра DBBOUТ или STATUS, содержимое которого будет выдано на системную шину по сигналу \overline{RD} . Сигнал A_0 также используется для установки флага F1 в регистре состояний во время действия сигнала \overline{RD} .

Сигналы \overline{WR} и \overline{RD} являются стандартными сигналами управления периферийными устройствами в системе MCS-80 и используются для синхронизации передачи данных между системной шиной и периферийными устройствами.

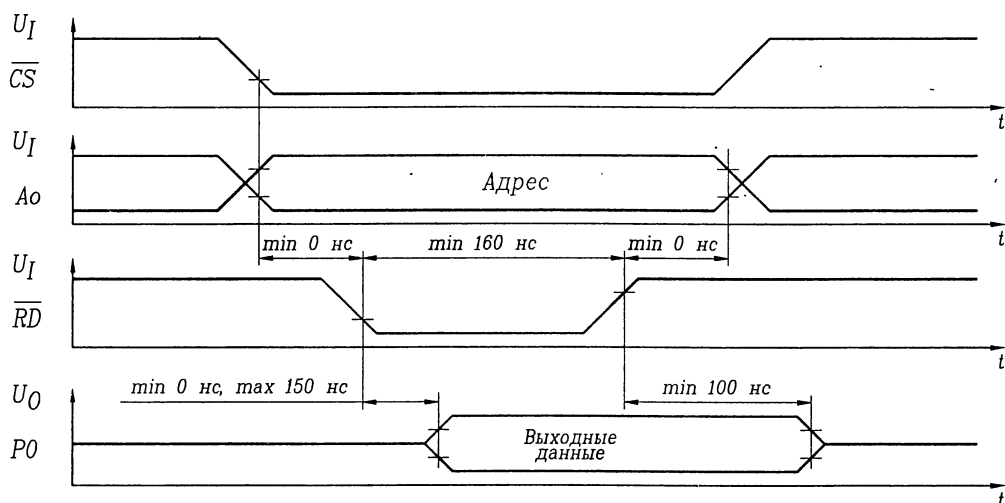


Рис. 8.19а. Чтение регистров DBBOUТ, STATUS

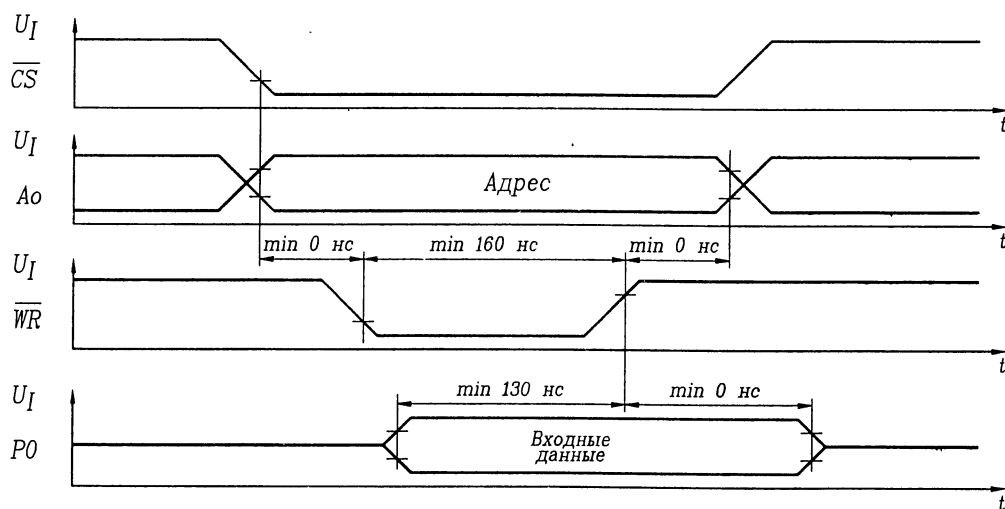


Рис. 8.19б. Запись данных/команд в регистр DBBIN

Сигналы \overline{CS} и A_0 вырабатываются адресной шиной головной системы. В системах с небольшим количеством устройств ввода-вывода можно использовать прямое подключение сигналов адресной шины (на рис. 8.20 линии A_0 и A_1 непосредственно соединены со входами A_0 и \overline{CS}).

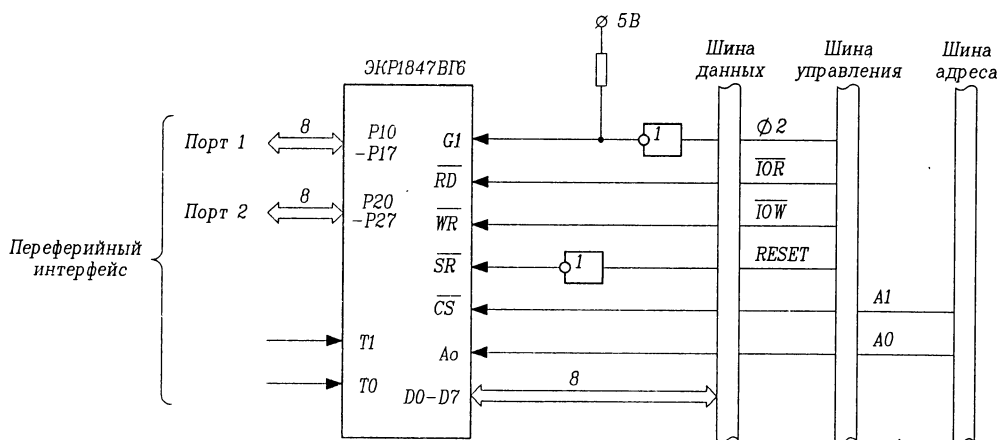


Рис. 8.20. Интерфейс системной шины 8080

8.2.11. Примеры работы УПИМК в интерфейсах систем 8085АН, 8086, 8088, 8048

Интерфейс 8085АН

На рис. 8.21 показан пример использования УПИМК в системе 8085АН. Система 8085АН использует мультиплексированную шину адреса/данных. В процессе ввода-вывода 8 старших адресных линий $A_8...A_{15}$ содержат тот же адрес, что и младшие 8 линий $A_0...A_7$ адреса/данных, поэтому при декодировании адреса используются только 8 старших линий. ИС декодера 8205 обеспечивает декодирование адреса как для УПИМК так и для ИС 8237. При передаче данных между УПИМК и 8085АН происходит подтверждение связи, для чего используются две линии порта P2 (P_{26} и P_{27}) в режиме DMA (посредством команды EN DMA). ИС 8237 управляет передачей данных по шине. Использование вывода P_{24} в режиме прерывания главного процессора по OVF позволяет известить 8085АН о завершении передачи, для чего используется вход прерывания RST 5.5.

Интерфейс 8088

УПИМК может быть использован в системе 8088, работающей в минимальном режиме (рис. 8.22). Две 8-битовых схемы-защелки 8282 производят демultipлексирование шины адреса и данных. Шина адреса — 20-разрядная. 16 младших разрядов шины адреса используются только для ввода-вывода, обеспечивая область адресации до 64К. Выбор УПИМК осуществляется декодером 8205 подачей сигнала \overline{CS} . Линия шины адреса A_0 соединена с соответствующим входом УПИМК для выбора регистра.

Интерфейс 8086

На рис. 8.23 показан пример использования УПИМК с системой 8086 в максимальном режиме. Демultipлексирование шины адреса и данных производится тремя схемами-защелки 8282, обеспечивающими разделение шины адреса и шины данных. Шина адреса — 20-разрядная, шина данных — 16 разрядная. Декодирование сигналов управления производится схемой 8288. Вход \overline{CS} используется для прямого выбора УПИМК. Данные передаются по линиям $D_0...D_7$.

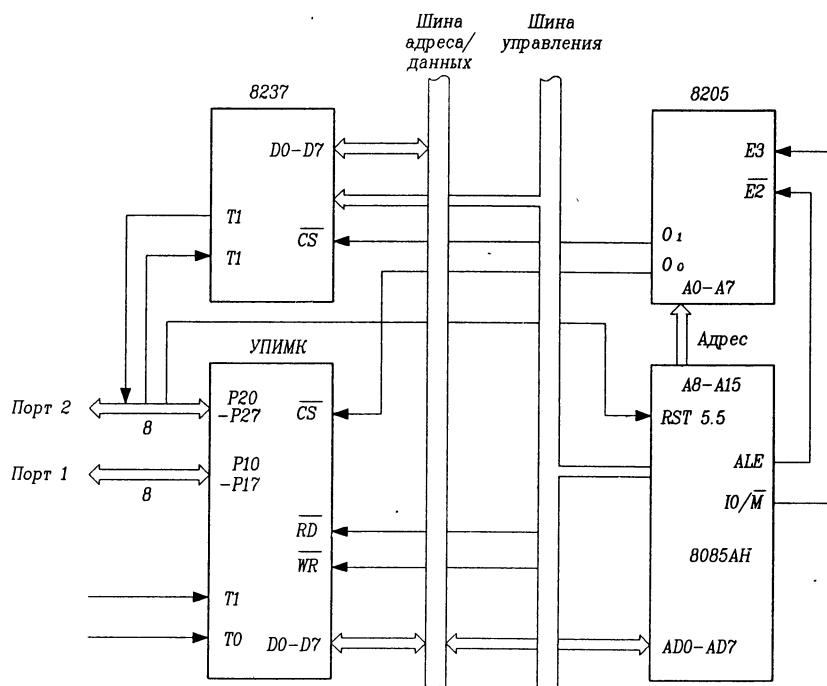


Рис. 8.21. Интерфейс системы 8085АН

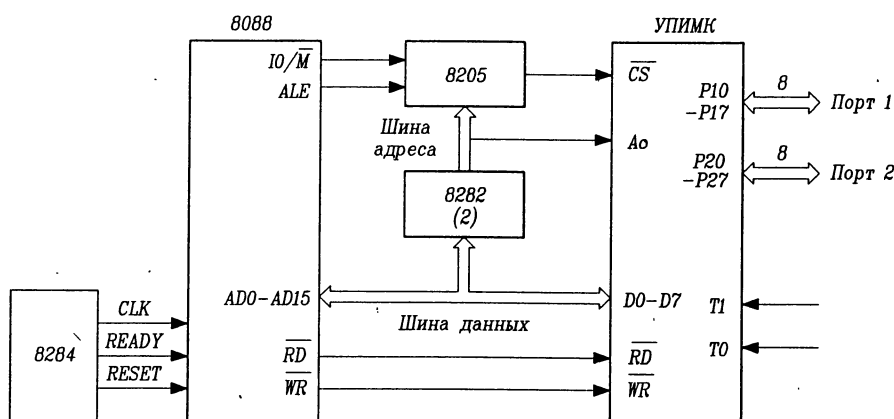


Рис. 8.22. Интерфейс системы 8088

Интерфейс 8048

Использование УПИМК в итерфейсе системы 8048 показано на рис. 8.24. Сигналы \overline{RD} и \overline{WR} ИС 8048 полностью соответствуют аналогичным сигналам УПИМК и соединены с ними напрямую. На рис. 8.25 показана распределенная процессорная система с возможностью подсоединений до 7 УПИМК к одному главному процессору 8048.

В этой конфигурации Порт 0 процессора 8048 используется в качестве шины данных. Порт 2 используется для выбора одного из семи УПИМК во время пересылки данных. Программирование УПИМК на решение отдельных изолированных задач в совокупности с параллельной работой позволяют увеличить производительность системы.

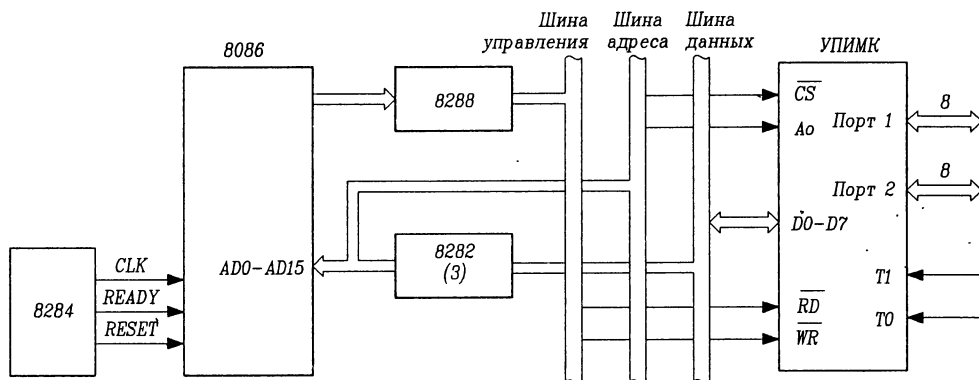


Рис. 8.23. Интерфейс системы 8086

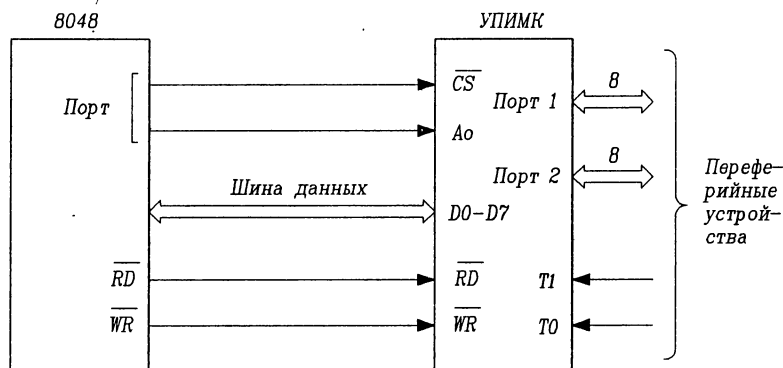


Рис. 8.24. Интерфейс системы 8048

Протокол обмена

При обмене информацией между УПИМК и главным процессором происходит следующая последовательность действий:

1. Главный процессор (ГП) читает регистр состояния STATUS (\overline{RD} , \overline{CS} , $A_0 = 0, 0, 1$) в процессе опроса или в ответ на запрос прерывания по \overline{IBF} или по OVF .

2. Если DBBIN регистр УПИМК пуст (флаг $IBF = 0$), ГП записывает слово в регистр DBBIN (\overline{WR} , \overline{CS} , $A_0 = 0, 0, 1$ или $0, 0, 0$). Если $A_0 = 1$, то это слово — команда и флаг $F1$ устанавливается в "1". Если $A_0 = 0$, то это данные и флаг $F1$ устанавливается в "0".

3. Если DBBOUT регистр УПИМК заполнен (флаг $OVF = 1$), ГП читает слово из регистра DBBOUT (\overline{RD} , \overline{CS} , $A_0 = 0, 0, 0$).

4. УПИМК распознает IBF (заполнение входного буфера) через прерывание по \overline{IBF} или через инструкцию $JNIBF$. Входные данные или команда обрабатываются в зависимости от состояния флага $F1$. Флаг IBF сбрасывается. Возвращение к шагу 1.

5. УПИМК распознает значение флага $OVF = 0$ (через инструкцию $JOBF$). Следующее слово заносится в регистр DBBOUT. Флаг OVF устанавливается в "1". Возвращение к шагу 1.

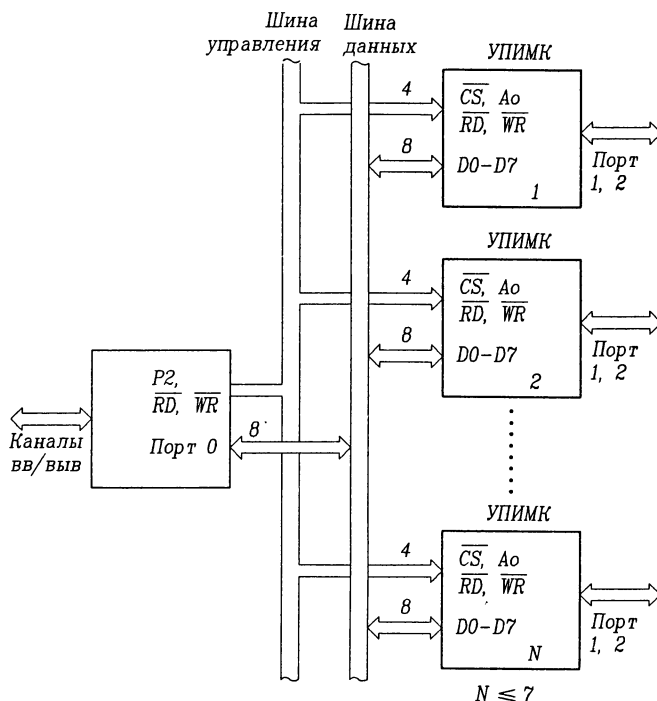


Рис. 8.25. Распределенная процессорная система

8.2.12. Каналы ввода-вывода

УПИМК имеет 16 каналов ввода-вывода, которые объединены в два 8-битовых TTL совместимых порта: Порт P1 и Порт P2. Под управлением программы эти каналы могут служить как для ввода так и для вывода данных.

Порты P1, P2 в режиме вывода обладают возможностью фиксации данных в так называемых триггерах-защелках. Эти данные статически присутствуют на выводах порта и могут быть изменены только новой выдачей по команде OUTL. Каждая выдача сопровождается занесением данных в защелку порта.

В состоянии ввода входная информация не изменяет состояния защелок. При использовании портов P1 и P2 в качестве входов необходимо до подачи входной информации линии портов установить в состояние высокого уровня, выдав на порт байт единиц. В это состояние выводы портов устанавливаются также после подачи сигнала \overline{SR} . Возможна произвольная смешанная настройка линий портов P1 и P2, когда одни линии порта работают на ввод, а другие — на вывод. Для настройки линии на режим ввода необходимо в триггер-защелку этой линии записать "1". Вводимые данные должны присутствовать на линиях порта до тех пор, пока не будут программно прочитаны. Электрическая схема одной из линий портов P1 и P2 показана на рис. 8.26.

Устройство B1 предназначено для передачи содержимого защелки (D-триггер) на внутреннюю шину данных для дальнейшей модификации по командам ORL PR, #data или ANL PR, #data.

Устройство B2 обеспечивает передачу входной информации порта на внутреннюю шину данных при выполнении команд, осуществляющих ввод информации с выводов порта.

Устройство И обеспечивает включение транзистора VT1 на время $t_{su}/6$ при изменении содержимого защелки (D-триггер) с "0" на "1" для формирования фронта нарастания сигнала на выводах порта. После выключения транзистора VT1

уровень логической единицы поддерживается на выходе порта с помощью транзистора VT3. Сопротивление открытого транзистора VT1 составляет приблизительно 500 Ом, сопротивление открытого транзистора VT3 — около 50 кОм. Время t_{cy} определяется по следующей формуле:

$$t_{cy} = 15/f_{G1},$$

где f_{G1} — частота тактовых сигналов УПИМК, МГц.

Кроме операций ввода-вывода информации, предусмотрена возможность выполнения логических операций И, ИЛИ непосредственно на портах P1 и P2 с помощью команд ANL PR, #data; ORL PR, #data.

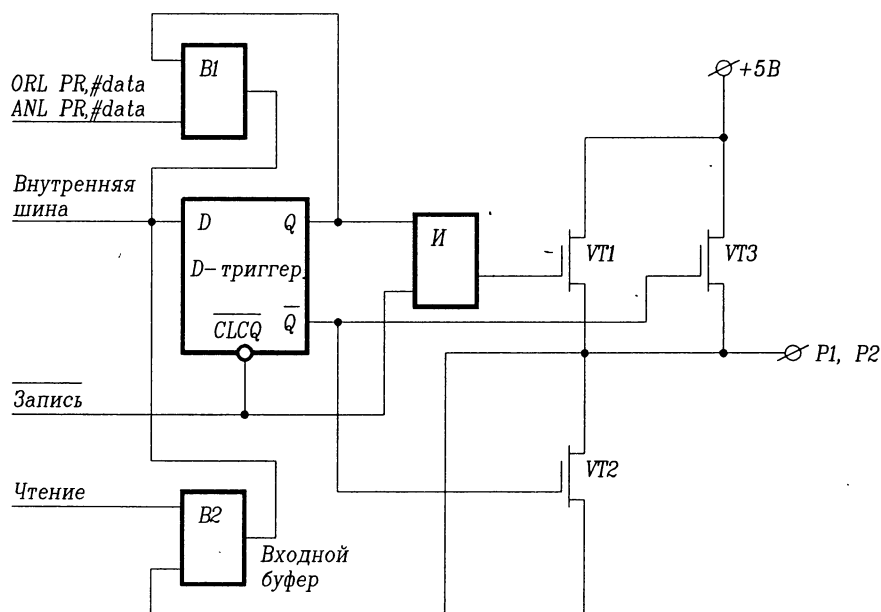


Рис. 8.26. Электрическая схема одной из линий портов P1 и P2

4 младших разряда Porta P2 (P20...P23) могут быть использованы как интерфейс для расширителя ввода-вывода KP580BP43. Таким способом можно получить до 28 и более каналов ввода-вывода. Дополнительные каналы представляют собой четыре 4-битовых порта: Порт 4, 5, 6 и 7. Стробирование данных при работе с дополнительным портом осуществляется сигналом \overline{PROG} . Входные данные для резидентного порта должны быть поданы в состояние машинного цикла, когда считывание возможно, то есть между соседними сигналами STR при выполнении команды ввода IN.

Схема соединения УПИМК с расширителем портов — ИС KP580BP43 показана на рис. 8.27. Расширитель портов KP580BP43 содержит четыре 4-битовых двунаправленных порта P4, P5, P6, P7, которые являются расширением резидентной системы ввода/вывода УПИМК. Обращения к портам P4—P7 производятся по командам MOVD, ANLD и ORLD, выполняемым совместно ОМЭВМ и расширителем. Каждая из указанных команд реализует обмен между УПИМК и KP580BP43. Обмен производится по линиям P20...P23 и синхронизируется выходным сигналом \overline{PROG} . Каждый обмен состоит из двух пересылок 4-разрядных полубайтов, первый из которых содержит код управляющего слова и адрес порта P4—P7, а второй — 4 бита данных (рис. 8.29). Переход сигнала \overline{PROG} из состояния

высокого уровня в состояние низкого уровня указывает, что на выводах P20...P23 находятся код управляющего слова и адрес порта, а переход сигнала \overline{PROG} из состояния низкого уровня в состояние высокого уровня указывает на то, что на выводах P20...P23 находятся данные.

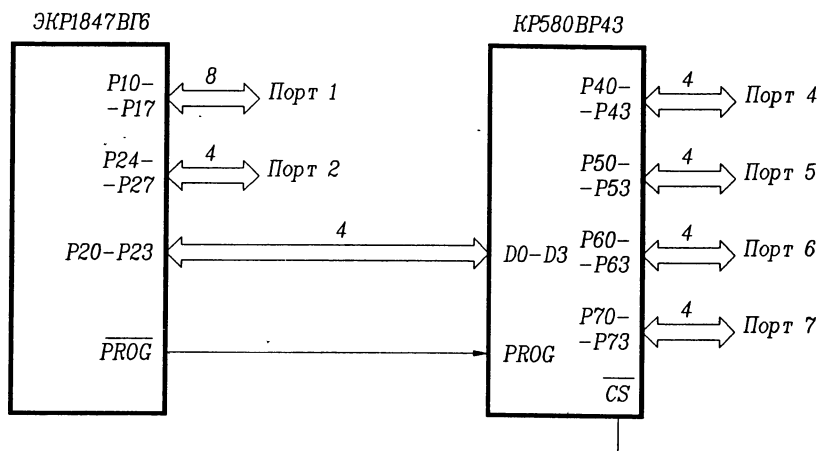


Рис. 8.27. Схема соединения УПИМК с расширителем портов — ИС КР580ВР43

Изменить состояние сигналов на портах P4—P7, работающих в режиме вывода, можно следующими способами: командой MOVD PD, A загрузить в требуемый порт новое значение; командой ORLD PD, A произвести логическое сложение порта с аккумулятором с загрузкой результата в порт; командой ANLD PD, A произвести логическое умножение порта и аккумулятора с загрузкой результата в порт. Логические операции выполняет КР580ВР43, а не УПИМК.

Для настройки порта расширителя на ввод данных необходимо выполнить для него команду ввода MOVD A, PD. Если до выполнения этой команды порт находился в режиме вывода, то результат первого ввода является неопределенным, а все последующие команды ввода будут давать правильный результат. При выполнении операции ввода порт настраивается на ввод и его выходы переключаются в высокоимпедансное состояние.

При включении питания все порты ИС КР580ВР43 переходят в высокоимпедансное состояние, а линии P20...P23 настраиваются на ввод.

Временные диаграммы работы УПИМК с дополнительными портами приведены на рис. 8.28а и рис. 8.28б.

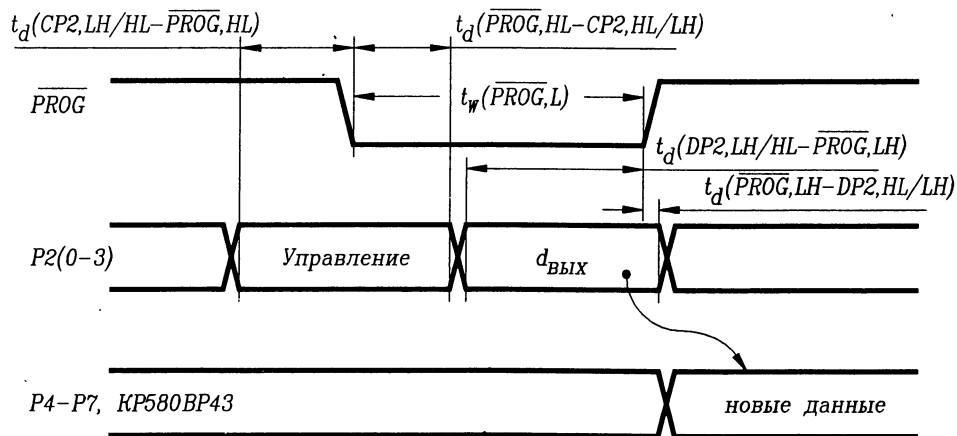


Рис. 8.28а. Временные диаграммы вывода данных при работе с КР580ВР43

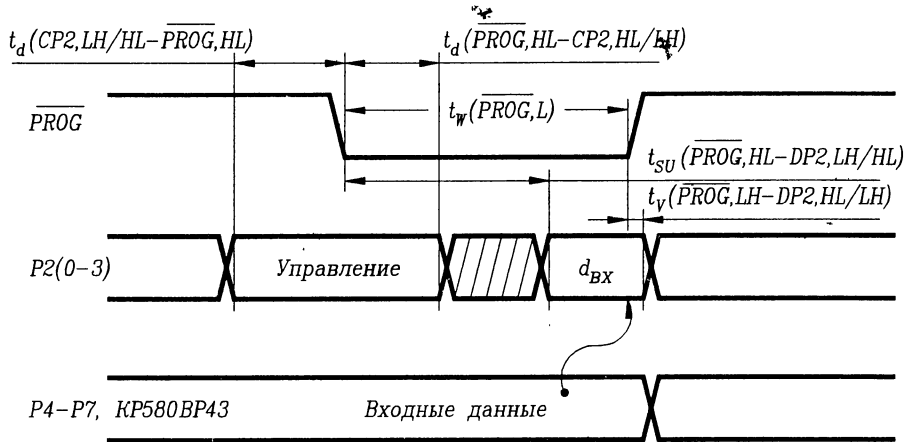


Рис. 8.28б. Временные диаграммы ввода данных при работе с KP580BP43

Рис. 8.28а иллюстрирует вывод данных $d_{\text{вых}}$ из УПИМК на порты P4—P7 расширителя портов KP580BP43.

На рис. 8.28б показан ввод данных с портов P4—P7 расширителя портов KP580BP43 в УПИМК ($d_{\text{вх}}$).

На рис. 8.30 приведены временные диаграммы работы портов P1 и P2. Смена информации на портах P1 и P2 "Данные порта" — "Новые данные" соответствует выполнению команд OUTL P1, A и OUTL P2, A.

3	2	1	Ø	3	2	1	Ø
I	I	A	A	d	d	d	d

Код управляющего слова

Данные

II	AA
00 Чтение	00 Порт 4
01 Запись	01 Порт 5
10 И	10 Порт 6
11 ИЛИ	11 Порт 7

Рис. 8.29. Формат пересылок при обменах УПИМК с KP50BP43

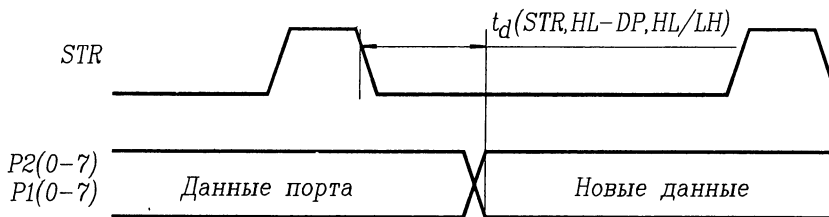


Рис. 8.30. Временные диаграммы работы портов P1 и P2

8.3. Режимы работ

8.3.1. Режим пошагового выполнения команд

В микроконтроллере ЭКР1847ВГ6 предусмотрена возможность организации работы в пошаговом режиме, который используется в процессе отладки и проверки памяти программ. В этом режиме при выполнении программы процессор можно остановить на каждой команде. В режиме останова адрес следующей команды, которая должна быть выбрана, выведен на порт P1 и 3 младших разряда порта P2.

При организации пошагового режима выполняются следующие операции:

1. В микроконтроллер поступает запрос на останов путем подачи напряжения низкого уровня на вывод \overline{SS} . Вывод \overline{SS} не должен находиться в состоянии низкого уровня пока сигнал STR остается в состоянии высокого уровня.

2. Процессор УПИМК останавливается на этапе выборки следующей команды; при этом завершается выполнение текущей команды. Если выполняется двухцикловая команда, то перед остановом выполняются оба цикла команды.

3. Микроконтроллер подтверждает, что он находится в режиме останова путем установки сигнала STR в состояние высокого уровня. В этом состоянии (которое можно сохранить сколь угодно долго) 11-битовый адрес следующей команды, которая будет выбираться после текущей, должен присутствовать на выводах порта P1 и трех младших разрядах порта P2.

4. Для того чтобы выйти из режима останова, на выводе \overline{SS} необходимо установить высокий уровень, что обеспечит выборку следующей команды. Процессор подтверждает выход из состояния останова путем установки низкого уровня сигнала STR.

5. Для того чтобы обеспечить останов на следующей команде, устанавливается низкий уровень сигнала \overline{SS} , как только уровень сигнала STR станет низким. Если уровень сигнала \overline{SS} остается высоким, УПИМК обрабатывает программу в обычном динамическом режиме.

На рис. 8.31 показан пример реализации пошагового режима, а на рис. 8.32 — диаграмма работы УПИМК в режиме пошагового выполнения программы. Для включения пошагового режима необходимо соответствующий переключатель установить в положение "Пошаговый режим". Кнопкой "шаг" обеспечивается переход на следующую команду.

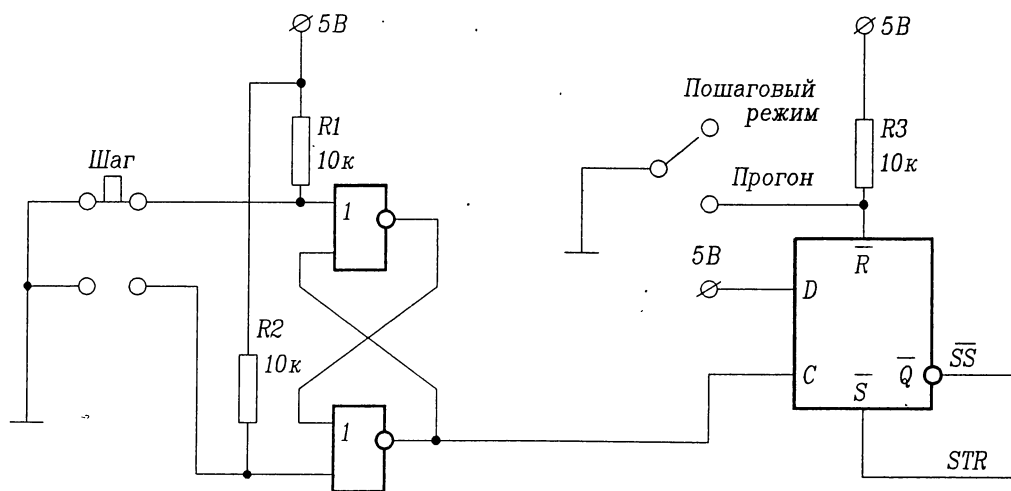


Рис. 8.31. Пример реализации пошагового режима

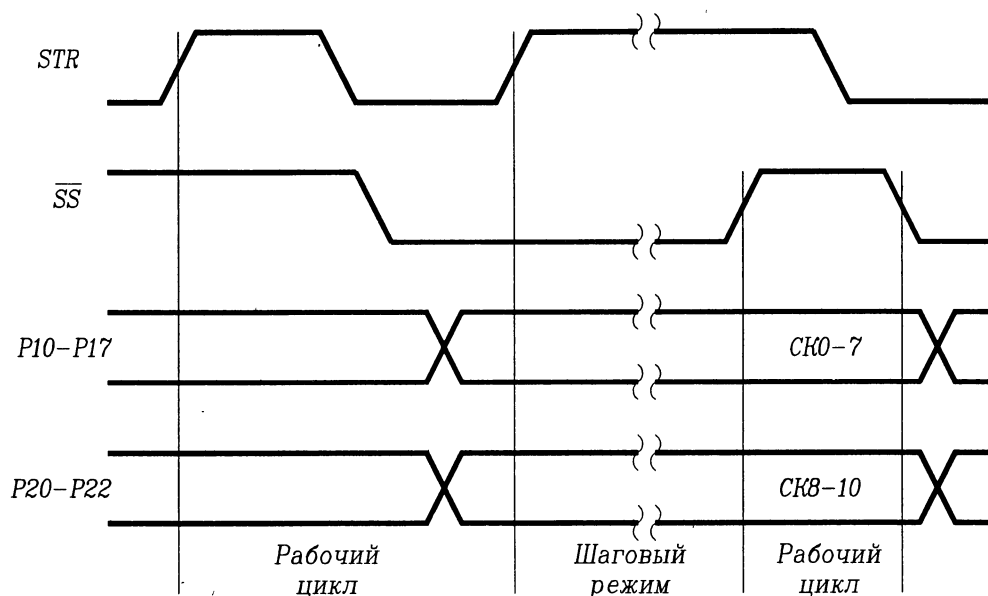


Рис. 8.32. Диаграмма работы УПИМК в режиме пошагового выполнения программы

8.3.2. Режим внешнего доступа

УПИМК имеет режим внешнего доступа, который устанавливает процессор в режим тестирования. Этот режим позволяет пользователю отключать внутреннюю программную память и работать с внешней памятью. Режим внешнего доступа полезен при отладке и тестировании, поскольку он позволяет пользователю непосредственно проверять работу процессора. Он используется только для тестирования, так как в этом режиме задействуются линии D0—D7, порт P1 (P10—P17) и порт P2 (P20—P22).

Этот режим обеспечивается путем подачи напряжения от источника +5 В на вывод М. 11-разрядное содержимое программного счетчика выдается на выходы P10—P17 порта P1 и P20—P22 порта P2 после того как сигнал STR установится в состояние высокого уровня. P10 является наименьшим значащим битом. Требуемый код операции должен находиться на линиях D0—D7 перед первой стадией S1. Во время стадии S1 код операции выбирается из D0—D7 и выполняется также как и при выборке из внутренней памяти.

Возможность чтения и записи регистров буфера шины данных в этом режиме остается, но только когда по линиям D0—D7 не происходит выборки кода команды. Эта возможность остается пока сигнал STR находится в состоянии высокого уровня.

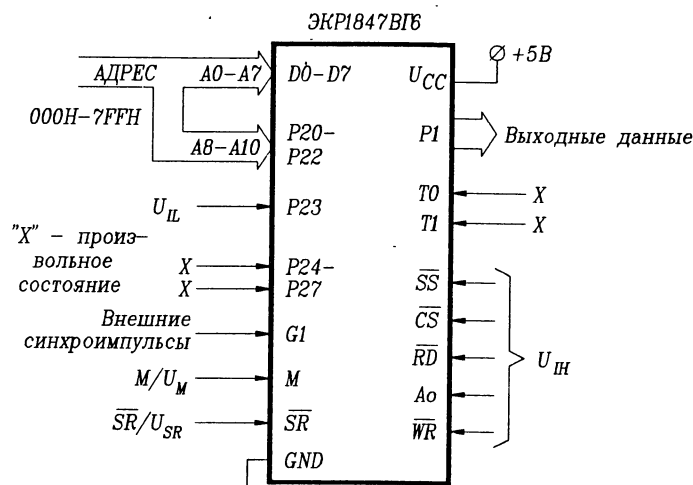
8.3.3. Проверка внутренней памяти программ

Режим проверки внутреннего ПЗУ используется при контроле правильности изготовления микросхем. В этом режиме контролируется правильность хранящейся в памяти программ информации, закодированной в процессе производства микросхемы ЭКР1847ВГ6.

Выходы микросхемы выполняют следующие функции:

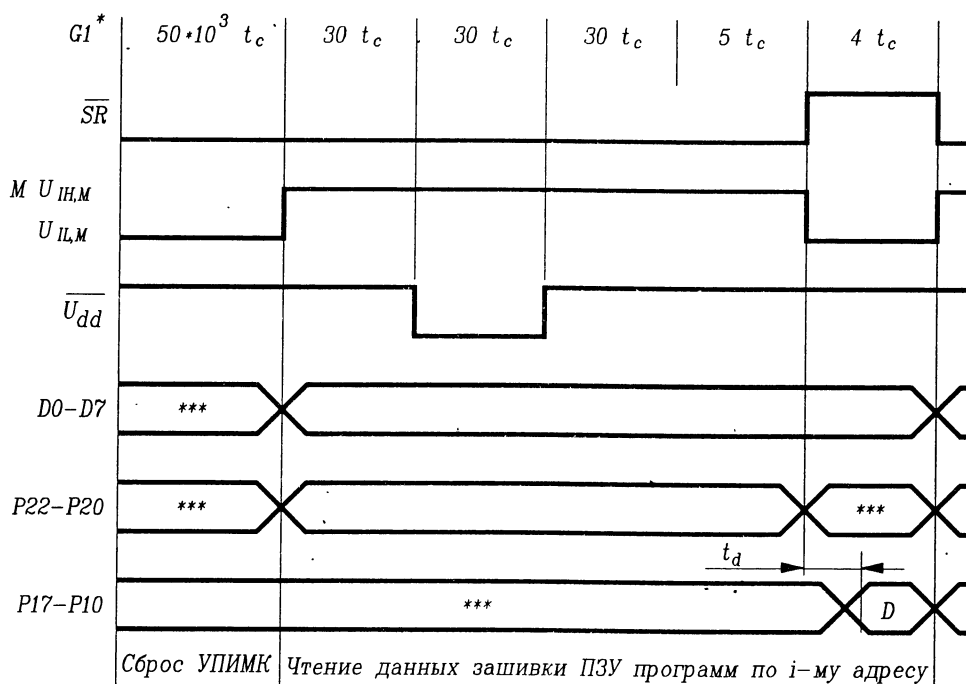
- D0...D7, P20...P22 - организуют подачу адреса A0—A10;
- P10...P17 - организуют выдачу данных для контроля.

Схема подачи сигналов на выходы УПИМК в режиме проверки внутренней памяти программ приведена на рис. 8.33, временная диаграмма — на рис. 8.34.



Параметры входных сигналов и адреса: $U_{IH} = U_{CC}$, $U_{IL} = 0$ В

Рис. 8.33. Режим проверки внутренней памяти программ



* — внешние синхроимпульсы с параметрами: $t_c = 1$ мкс, $t_{WH} = t_{WL}$, $t_{HL} = t_{LH}$ не более 100 нс, $U_{IL} = 0$ В, $U_{IH} = U_{CC}$
 $U_{IH,M} = 10,5$ В, $U_{IL,M} = 5$ В
 Параметры входных сигналов и адреса: $U_{IH} = U_{CC}$, $U_{IL} = 0$ В
 *** — произвольное состояние выводов порта
 D — данные зашивки ПЗУ по i-му адресу

Рис. 8.34. Временная диаграмма работы микросхемы в режиме проверки памяти программ (контроль данных по i -му адресу)

В качестве примера длительность внешних синхроимпульсов t_c выбрана равной 1 мкс (тактовая частота — 1 МГц). Для других значений t_c (в допустимом диапазоне частот 0...8 МГц) указанные временные значения других параметров соответственно увеличиваются или уменьшаются прямо пропорционально новому значению t_c .

В первый момент после включения, длящийся $50 \cdot 10^3 t_c$ происходит начальная установка схемы. В течение следующих $30 t_c$ значение 11-разрядного адреса A_i , установленного на выводах D7—D0 (разряды адреса A7—A0) и P22—P20 (разряды адреса A10—A8) записывается во внутренний регистр адреса ПЗУ. Во время следующих $30 t_c$, в течение которых сигнал \bar{U}_{dd} устанавливается в состояние низкого уровня, происходит синхронизация внутренних сигналов схемы к сигналам выдачи данных. Следующие $30 t_c$ — подготовка выдачи данных. Два последних временных периода длительностью $5 t_c$ и $4 t_c$ соответствуют режиму выдачи данных ПЗУ на выводы порта P1.

Для работы ИС ЭКР1847ВГ6 в режиме проверки внутренней памяти программ необходима привязка к частоте синхронизации УПИМК: считываемая из внутреннего ПЗУ информация является истинной на выводах порта P1 в течение одной стадии ($3t_c$) после $t_d = 4t_c(t_{G1/LH} + t_{SR/LH}, t_{M/HL})$, т.е. через $4t_c$ после перехода сигналов \bar{SR} и \bar{M} соответственно в состояние высокого и низкого уровня.

8.3.3. Режим микропотребления

УПИМК может быть приведен в режим микропотребления аппаратным или программным способом. В первом случае на вывод \bar{U}_{dd} необходимо подать напряжение низкого уровня. При этом работа задающего генератора блокируется и УПИМК удерживается в состоянии сброса. В нем производится установка всех узлов в такое же состояние как и при подаче внешнего сигнала \bar{SR} . В этом режиме сохраняется содержимое внутреннего ОЗУ, аккумулятора и старших разрядов PSW.

При подаче напряжения высокого уровня на вывод \bar{U}_{dd} УПИМК переходит из режима микропотребления в режим нормальной работы и начинает выполнение программы с нулевого адреса.

Для перехода УПИМК в режим микропотребления программным способом необходимо выполнить команду IDL. При выполнении команды IDL выполнение программы прекращается, внутренние фазовые и тактовые сигналы не формируются, состояние всех узлов УПИМК сохраняется. Для вывода УПИМК из этого состояния необходимо на выводы \bar{CS} и \bar{WR} подать напряжение низкого уровня длительностью не менее $2t_{cy}$ ($t_{cy} = 15/f_{G1}$, где f_{G1} — частота внутреннего тактового генератора УПИМК). Если в УПИМК перед этим было установлено разрешение на прерывание, то начнется выполнение программы обработки внешнего прерывания. В противном случае будет возобновлено выполнение программы с команды, следующей за командой IDL.

8.4. Система команд

Система команд УПИМК (ЭКР1847ВГ6 и UPI-42) совместима по кодам операции с системой МК48 (MCS-48) за исключением команд работы с внешней памятью программ и памятью данных, а также дополнительных команд работы с шиной данных. Из 93 команд 67 — однобайтовые. В двухбайтовых командах первый байт несет информацию о коде операции, второй байт представляет собой непосредственные данные или младшие разряды адреса следующей команды. Большинство команд (55) выполняется за один машинный цикл; 38 команд, в том числе 11 однобайтовых, выполняются за два машинных цикла. Выполнение двухбайтных команд за два цикла связано с дополнительным обращением к памяти программ. Однобайтовые команды в большинстве выполняются за один цикл, однако команды, требующие обращения к внешним устройствам, а также команды RET и RETR выполняются за два цикла.

Система команд УПИМК эффективно выполняет однобитовые операции, используемые при управлении различных внешних устройств. Специальные инструкции позволяют индивидуально устанавливать или очищать отдельные

Продолжение таблицы 8.5

ПЕРЕЧЕНЬ КОМАНД МИКРОЭВМ СЕМЕЙСТВА UPI-42 И ЭКР1847ВГ6			
МНЕМОНИКА	КОД	ВРЕМЯ (в циклах)	Страница в описании
ADD A,REG	01101REG	1	342
ADD A,@R	0110000R	1	342
ADD A,#DATA	00000011 DDDDDDDD	2	342
ADDC A, REG	01111REG	1	343
ADDC A, @R	0111000R	1	343
ADDC A,#DATA	00010011 DDDDDDDD	2	344
ANL A, REG	01011REG	1	344
ANL A, @R	0101000R	1	345
ANL A,#DATA	01010011 DDDDDDDD	2	345
ANL PR,#DATA	100110PR DDDDDDDD	2	345
ANLD PD, A	100111PD	2	346
CALL ADDR	AAA10100 AAAAAAAA	2	346
CLR A	00100111	1	347
CLR C	10010111	1	347
CLR F0	10000101	1	347
CLR F1	10100101	1	347
CPL A	00110111	1	348
CPL C	10100111	1	348
CPL F0	10010101	1	348
CPL F1	10110101	1	348
DA A	01010111	1	348
DEC A	00000111	1	349
DEC REG	11001REG	1	349
DIS I	00010101	1	350
DIS TCNTI	00110101	1	350

Продолжение таблицы 8.5

МНЕМОНИКА	КОД	ВРЕМЯ (в циклах)	Страница в описании
DJNZ REG, ADDR	11101REG AAAAAAAA	2	350
* EN DMA	11100101	1	351
EN FLAGS	11110101	1	351
EN I	00000101	1	351
EN TCNTI	00100101	1	352
** IDL	00000001	1	352
IN A, DBB	00100010	1	352
IN A, PR	000010PR	2	352
INC A	00010111	1	353
INC REG	00011REG	1	353
INC @R	0001000R	1	353
JBBIT ADDR	BIT10010 AAAAAAAA	2	354
JC ADDR	11110110 AAAAAAAA	2	354
JF0 ADDR	10110110 AAAAAAAA	2	355
JF1 ADDR	01110110 AAAAAAAA	2	355
JMP ADDR	AAA00100 AAAAAAAA	2	356
JMPP @A	10110011	2	356
JNC ADDR	11100110 AAAAAAAA	2	357
JNIBF ADDR	11010110 AAAAAAAA	2	357
JNT0 ADDR	00100110 AAAAAAAA	2	358
JNT1 ADDR	01000110 AAAAAAAA	2	358
JNZ ADDR	10010110 AAAAAAAA	2	359
J0BF ADDR	10000110 AAAAAAAA	2	359

Продолжение таблицы 8.5

МНЕМОНИКА	КОД	ВРЕМЯ (в циклах)	Страница в описании
JTF ADDR	00010110 AAAAAAAA	2	360
JT0 ADDR	00110110 AAAAAAAA	2	360
JT1 ADDR	01010110 AAAAAAAA	2	361
JZ ADDR	11000110 AAAAAAAA	2	361
MOV A, #DATA	00100011 DDDDDDDD	2	362
MOV A, PSW	11000111	1	364
MOV A, REG	11111REG	1	362
MOV A, @R	1111000R	1	363
MOV A, T	01000010	1	364
MOV PSW, A	11010111	1	364
MOV REG, A	10101REG	1	363
MOV REG, #DATA	10111REG DDDDDDDD	2	362
MOV @R, A	1010000R	1	362
MOV @R, #DATA	1011000R DDDDDDDD	2	363
MOV STS, A	10010000	1	365
MOV T, A	01100010	1	364
MOVD A, PD	000011PD	2	365
MOVD PD, A	001111PD	2	365
MOVP A, @A	10100011	2	365
MOVP3 A, @A	11100011	2	366
NOP	00000000	1	366
ORL A, REG	01001REG	1	366
ORL A, @R	0100000R	1	366
ORL A, #DATA	01000011 DDDDDDDD	2	367
ORL PR, #DATA	100010PR DDDDDDDD	2	367

Продолжение таблицы 8.5

МНЕМОНИКА	КОД	ВРЕМЯ (в циклах)	Страница в описании
ORLD PD, A	100011PD	2	367
OUT DBB, A	00000010	1	368
OUTL PR, A	001110PR	2	368
RET	10000011	2	368
RETR	10010011	2	368
RL A	11100111	1	369
RLC A	11110111	1	369
RR A	01110111	1	369
RRC A	01100111	1	370
SEL RB0	11000101	1	370
SEL RB1	11010101	1	370
STOP TCNT	01100101	1	371
STRT CNT	01000101	1	371
STRT T	01010101	1	371
SWAP A	01000111	1	371
XCH A, REG	00101REG	1	372
XCH A, @R	0010000R	1	372
XCHD A, @R	0011000R	1	372
XRL A, REG	11011REG	1	373
XRL A, @R	1101000R	1	373
XRL A, #DATA	11010011 DDDDDDDD	2	373

* Только для микросхем серии UPI-42.

** Только для микросхем КР1847ВГ6.

Команды можно условно разделить на следующие группы:

- команды работы с аккумулятором;
- команды ввода-вывода;
- команды работы с регистрами;
- команды перехода;
- команды обращения к подпрограмме;
- команды работы с флагами;
- команды пересылки данных;
- команды работы с таймером/счетчиком;
- команды управления.

Группа команд работы с аккумулятором включает команды сложения и логические команды (И, ИЛИ, исключающее ИЛИ), с помощью которых производятся операции сложения и логические операции над содержимым аккумулятора и вторым операндом. В качестве второго операнда могут использоваться данные регистров общего назначения, содержимое внутренней памяти данных и непосредственные данные. Кроме этого, в команды работы с аккумулятором включены: сдвиг вправо и влево, увеличение и уменьшение на 1, дополнение, очистка, обмен ниблами (полубайтами). Арифметические операции и операции сдвига могут также производиться с использованием переноса C. Для задач, связанных с обработкой двоично-десятичных чисел, имеется команда десятичной коррекции содержимого аккумулятора.

Группа команд ввода-вывода осуществляет обмен информацией между портами УПИМК и аккумулятором, а также производит логические операции над содержимым портов и непосредственными данными.

Команды работы с регистрами производят две операции: инкремент и декремент содержимого регистров и инкремент содержимого внутренней памяти данных.

Группа команд перехода предназначена для организации ветвления программ и включает группу команд безусловных переходов и группу команд условных переходов. В качестве условий анализируются входные сигналы T0, T1, состояние флагов F0, F1, входного и выходного буфера шины данных OBF и IBF, флага таймера-счетчика, триггера переноса, содержимого аккумулятора и отдельных его разрядов.

Команды обращения к подпрограмме включают в себя: переход к подпрограмме и две команды выхода из подпрограммы, причем одна из них осуществляет выход с восстановлением содержимого регистра состояния программы (PSW). Эту команду рекомендуется использовать для организации выхода из подпрограммы обработки прерываний. При использовании команды RET не происходит снятия блокировки со входа схемы обработки режима прерывания.

Группа команд работы с флагами включает два типа команд: установка в ноль флагов F0, F1, C и установка их инверсии.

Группа команд пересылки данных осуществляет передачу данных между аккумулятором, регистрами общего назначения и внутренней памятью данных, передачу непосредственных данных в вышеперечисленные элементы, передачу данных между аккумулятором и регистром состояния программы (PSW) или регистром состояния STATUS (в последнем случае передаются только биты 4—7), пересылку данных между аккумулятором и внешней памятью данных, обмен данными между аккумулятором и регистрами общего назначения или внутренней памятью данных, а также пересылку данных из памяти программ в аккумулятор.

В группе команд работы с таймером-счетчиком имеется две команды пересылки данных между таймером-счетчиком и аккумулятором, три команды запуска и остановки таймера-счетчика, четыре команды разрешения и запрета прерывания по таймеру-счетчику.

Команды управления включают команды разрешения и запрета IBF-прерывания, команды по выбору банков РОН, команду разрешения ввода тактовых импульсов на вывод T0 и команду "Нет операции".

Перечень команд УПИМК, систематизированных по мнемосодам, приведен в табл. 8.6.

Перечень команд, систематизированных по кодам операций — в табл. 8.7.

Коды команд в таблицах 8.6 и 8.7 приведены в шестнадцатеричном формате.

Таблица 8.6

МНЕМОНИКА	КОД	МНЕМОНИКА	КОД	МНЕМОНИКА	КОД
ADD A, R0	68	CALL ADDRESS		INC R6	1E
A, R1	69	(PAGE 6)	D4	R7	1F
A, R2	6A	(PAGE 7)	F4	@R0	10
A, R3	6B	CLR A	27	@R1	11
A, R4	6C	C	97	JB0 ADDR	12
A, R5	6D	F0	85	JB1 ADDR	32
A, R6	6E	F1	A5	JB2 ADDR	52
A, R7	6F	CPL A	37	JB3 ADDR	72
A, @R0	60	C	A7	JB4 ADDR	92
A, @R1	61	F0	95	JB5 ADDR	B2
A, #DATA	03	F1	B5	JB6 ADDR	D2
ADDC A, R0	78	DA A	57	JB7 ADDR	F2
A, R1	79	DEC A	07	JC ADDR	F6
A, R2	7A	R0	C8	JF0 ADDR	B6
A, R3	7B	R1	C9	JF1 ADDR	76
A, R4	7C	R2	CA	JMP ADDR	
A, R5	7D	R3	CB	(PAGE 0)	04
A, R6	7E	R4	CC	(PAGE 1)	24
A, R7	7F	R5	CD	(PAGE 2)	44
A, @R0	70	R6	CE	(PAGE 3)	64
A, @R1	71	R7	CF	(PAGE 4)	84
A, #DATA	13	DIS I	15	(PAGE 5)	A4
ANL A, R0	58	TCNTI	35	(PAGE 6)	C4
A, R1	59	DJNZ R0, ADDR	E8	(PAGE 7)	E4
A, R2	5A	R1, ADDR	E9	JMPP @A	B3
A, R3	5B	R2, ADDR	EA	JNC ADDR	E6
A, R4	5C	R3, ADDR	EB	JNIBF ADDR	D6
A, R5	5D	R4, ADDR	EC	JNT0 ADDR	26
A, R6	5E	R5, ADDR	ED	JNT1 ADDR	46
A, R7	5F	R6, ADDR	EE	JNZ ADDR	96
A, @R0	50	R7, ADDR	EF	JOBFB ADDR	86
A, @R1	51	*EN DMA	E5	JTF ADDR	16
A, #DATA	53	EN I	05	JT0 ADDR	36
P1, #DATA	99	FLAGS	F5	JT1 ADDR	56
P2, #DATA	9A	TCNTI	25	JZ ADDR	C6
ANLD P4, A	9C	**IDL	01	MOV A, #DATA	23
P5, A	9D	IN A, DBB	22	A, PSW	C7
P6, A	9E	IN A, P1	09	A, R0	F8
P7, A	9F	A, P2	0A	A, R1	F9
CALL ADDRESS		INC A	17	A, R2	FA
(PAGE 0)	14	R0	18	A, R3	FB
(PAGE 1)	34	R1	19	A, R4	FC
(PAGE 2)	54	R2	1A	A, R5	FD
(PAGE 3)	74	R3	1B	A, R6	FE
(PAGE 4)	94	R4	1C	A, R7	FF
(PAGE 5)	B4	R5	1D	A, @R0	F0

Продолжение таблицы 8.6

МНЕМОНИКА	КОД	МНЕМОНИКА	КОД	МНЕМОНИКА	КОД
MOV A, @R1	F1	ORL A, #DATA	43		
A, T	42	P1, #DATA	89		
PSW, A	D7	P2, #DATA	8A		
R0, A	A8	ORLD P4, A	8C		
R1, A	A9	P5, A	8D		
R2, A	AA	P6, A	8E		
R3, A	AB	P7, A	8F		
R4, A	AC	OUT DBB, A	02		
R5, A	AD	OUTL P1, A	39		
R6, A	AE	P2, A	3A		
R7, A	AF	RET	83		
R0, #DATA	B8	RETR	93		
R1, #DATA	B9	RL A	E7		
R2, #DATA	BA	RLC A	F7		
R3, #DATA	BB	RR A	77		
R4, #DATA	BC	RRC A	67		
R5, #DATA	BD	SEL RB0	C5		
R6, #DATA	BE	RB1	D5		
R7, #DATA	BF	STOP TCNT	65		
@R0, A	A0	STRT CNT	45		
@R1, A	A1	T	55		
@R0, #DATA	B0	SWAP A	47		
@R1, #DATA	B1	XCH A, R0	28		
STS, A	90	A, R1	29		
T, A	62	A, R2	2A		
MOVD A, P4	0C	A, R3	2B		
A, P5	0D	A, R4	2C		
A, P6	0E	A, R5	2D		
A, P7	0F	A, R6	2E		
P4, A	3C	A, R7	2F		
P5, A	3D	A, @R0	20		
P6, A	3E	A, @R1	21		
P7, A	3F	XCHD A, @R0	30		
MOV P A, @A	A3	A, @R1	31		
MOV P3 A, @A	E3	XRL A, R0	D8		
NOP	00	A, R1	D9		
ORL A, R0	48	A, R2	DA		
A, R1	49	A, R3	DB		
A, R2	4A	A, R4	DC		
A, R3	4B	A, R5	DD		
A, R4	4C	A, R6	DE		
A, R5	4D	A, R7	DF		
A, R6	4E	A, @R0	D0		
A, R7	4F	A, @R1	D1		
A, @R0	40	A, #DATA	D3		
A, @R1	41				

Таблица 8.7

КОД	МНЕМОНИКА	КОД	МНЕМОНИКА	КОД	МНЕМОНИКА
00	NOP	2E	XCH A, R6	5C	ANL A, R4
01	**IDL	2F	XCH A, R7	5D	ANL A, R5
02	OUTL DBB, A	30	XCHD A, @R0	5E	ANL A, R6
03	ADD A, #DATA	31	XCHD A, @R1	5F	ANL A, R7
04	JMP (PAGE 0)	32	JB1 ADDR	60	ADD A, @R0
05	EN I	33	—	61	ADD A, @R1
06	—	34	CALL (PAGE 1)	62	MOV T, A
07	DEC A	35	DIS TCNTI	63	—
08	—	36	JT0 ADDR	64	JMP (PAGE 3)
09	IN A, P1	37	CPL A	65	STOP TCNT
0A	IN A, P2	38	—	66	—
0B	—	39	OUTL P1, A	67	RRC A
0C	MOVD A, P4	3A	OUTL P2, A	68	ADD A, R0
0D	MOVD A, P5	3B	—	69	ADD A, R1
0E	MOVD A, P6	3C	MOVD P4, A	6A	ADD A, R2
0F	MOVD A, P7	3D	MOVD P5, A	6B	ADD A, R3
10	INC @R0	3E	MOVD P6, A	6C	ADD A, R4
11	INC @R1	3F	MOVD P7, A	6D	ADD A, R5
12	JB0 ADDR	40	ORL A, @R0	6E	ADD A, R6
13	ADDC A, #DATA	41	ORL A, @R1	6F	ADD A, R7
14	CALL (PAGE 0)	42	MOV A, T	70	ADDC A, @R0
15	DIS I	43	ORL A, #DATA	71	ADDC A, @R1
16	JTF ADDR	44	JMP (PAGE 2)	72	JB3 ADDR
17	INC A	45	STRT CNT	73	—
18	INC R0	46	JNT1 ADDR	74	CALL (PAGE 3)
19	INC R1	47	SWAP A	75	—
1A	INC R2	48	ORL A, R0	76	JF1 ADDR
1B	INC R3	49	ORL A, R1	77	RR A
1C	INC R4	4A	ORL A, R2	78	ADDC A, R0
1D	INC R5	4B	ORL A, R3	79	ADDC A, R1
1E	INC R6	4C	ORL A, R4	7A	ADDC A, R2
1F	INC R7	4D	ORL A, R5	7B	ADDC A, R3
20	XCH A, @R0	4E	ORL A, R6	7C	ADDC A, R4
21	XCH A, @R1	4F	ORL A, R7	7D	ADDC A, R5
22	IN A, DBB	50	ANL A, @R0	7E	ADDC A, R6
23	MOV A, #DATA	51	ANL A, @R1	7F	ADDC A, R7
24	JMP (PAGE 1)	52	JB2 ADDR	80	—
25	EN TCNTI	53	ANL A, #DATA	81	—
26	JNT0 ADDR	54	CALL (PAGE 2)	82	—
27	CLR A	55	STRT T	83	RET
28	XCH A, R0	56	JT1 ADDR	84	JMP (PAGE 4)
29	XCH A, R1	57	DA A	85	CLR F0
2A	XCH A, R2	58	ANL A, R0	86	JOBFB ADDR
2B	XCH A, R3	59	ANL A, R1	87	—
2C	XCH A, R4	5A	ANL A, R2	88	—
2D	XCH A, R5	5B	ANL A, R3	89	ORL P1, #DATA

Продолжение таблицы 8.7

КОД	МНЕМОНИКА	КОД	МНЕМОНИКА	КОД	МНЕМОНИКА
8A	ORL P2, #DATA	B6	JFØ ADDR	E2	—
8B	—	B7	—	E3	MOV P3 A, @A
8C	ORLD P4, A	B8	MOV RØ, #DATA	E4	JMP (PAGE 7)
8D	ORLD P5, A	B9	MOV R1, #DATA	E5	*EN DMA
8E	ORLD P6, A	BA	MOV R2, #DATA	E6	JNC ADDR
8F	ORLD P7, A	BB	MOV R3, #DATA	E7	RL A
9Ø	MOV STS, A	BC	MOV R4, #DATA	E8	DJNZ RØ, ADDR
91	—	BD	MOV R5, #DATA	E9	DJNZ R1, ADDR
92	JB4 ADDR	BE	MOV R6, #DATA	EA	DJNZ R2, ADDR
93	RETR	BF	MOV R7, #DATA	EB	DJNZ R3, ADDR
94	CALL (PAGE 4)	CØ	—	EC	DJNZ R4, ADDR
95	CPL FØ	C1	—	ED	DJNZ R5, ADDR
96	JNZ ADDR	C2	—	EE	DJNZ R6, ADDR
97	CLR C	C3	—	EF	DJNZ R7, ADDR
98	—	C4	JMP (PAGE 6)	FØ	MOV A, @RØ
99	ANL P1, #DATA	C5	SEL RBØ	F1	MOV A, @R1
9A	ANL P2, #DATA	C6	JZ ADDR	F2	JB7 ADDR
9B	—	C7	MOV A, PSW	F3	—
9C	ANLD P4, A	C8	DEC RØ	F4	CALL (PAGE 7)
9D	ANLD P5, A	C9	DEC R1	F5	EN FLAGS
9E	ANLD P6, A	CA	DEC R2	F6	JC ADDR
9F	ANLD P7, A	CB	DEC R3	F7	RLC A
AØ	MOV @RØ, A	CC	DEC R4	F8	MOV A, RØ
A1	MOV @R1, A	CD	DEC R5	F9	MOV A, R1
A2	—	CE	DEC R6	FA	MOV A, R2
A3	MOV P A, @A	CF	DEC R7	FB	MOV A, R3
A4	JMP (PAGE 5)	DØ	XRL A, @RØ	FC	MOV A, R4
A5	CLR F1	D1	XRL A, @R1	FD	MOV A, R5
A6	—	D2	JB6 ADDR	FE	MOV A, R6
A7	CPL C	D3	XRL A, #DATA	FF	MOV A, R7
A8	MOV RØ, A	D4	CALL (PAGE 6)		
A9	MOV R1, A	D5	SEL RB1		
AA	MOV R2, A	D6	JNIBF ADDR		
AB	MOV R3, A	D7	MOV PSW, A		
AC	MOV R4, A	D8	XRL A, RØ		
AD	MOV R5, A	D9	XRL A, R1		
AE	MOV R6, A	DA	XRL A, R2		
AF	MOV R7, A	DB	XRL A, R3		
BØ	MOV @RØ, #DATA	DC	XRL A, R4		
B1	MOV @R1, #DATA	DD	XRL A, R5		
B2	JB5 ADDR	DE	XRL A, R6		
B3	JMPP @A	DF	XRL A, R7		
B4	CALL (PAGE 5)	EØ	—		
B5	CPL F1	E1	—		

* Только для микросхем серии UPI-42.

** Только для микросхем ЭКР1847ВГ6.

8.5. Описание машинных команд

Описание каждой машинной команды состоит из формата, кода, алгоритма и времени ее выполнения.

В части "Формат" приведено содержимое полей "Мнемокод" и "Аргументы" машинной команды при ее записи в формате предложения языка ассемблера.

В части "Код" приведен двоичный код машинной команды.

В части "Алгоритм" приведено символическое описание операции, выполняемое машинной командой. При описании операции используются обозначения, приведенные в таблице 8.8.

Таблица 8.8

Обозначение	Операция
(X)	Содержимое элемента X
((X))	Содержимое по адресу, хранящемуся в элементе X
X[M]	Разряд M элемента X
X[M1-M2]	Группа разрядов M1-M2 элемента X
+	Операции: сложения
-	вычитания
AND	логического умножения
OR	логического сложения
XOR	сложения по модулю 2
NOT	инверсии
>	больше
=	равно
<>	неравно
X <= Y	пересылки содержимого элемента Y в элемент X
X <=> Y	обмена содержимых элементов X и Y
X : Y	присоединения старшей части, хранящейся в элементе X, к младшей части, хранящейся в элементе Y
.	Операторы: Условные
если..., то...	
если..., то...	
иначе...	
для N от... до.	цикла
TMP	Промежуточный регистр
DBB	Буфер шины данных
SP	Указатель стека
PC	Счетчик команд
	Код:
B	двоичный
H	шестнадцатеричный

В части "Время" приведено количество циклов УПИМК, требуемых для выполнения машинной команды. Время цикла определяется по следующей формуле: $t_{cy} = 15/f_{G1}$, t_{cy} — время цикла, мкс; f_{G1} — частота генератора тактовых сигналов УПИМК, МГц.

Команда ADD A, <Reg>

По команде ADD A, <Reg> содержимое рабочего регистра <Reg> складывается с содержимым аккумулятора A. Результат вычисления записывается в аккумулятор.

При переносе из 3-го разряда АЛУ триггер признака вспомогательного переноса AC устанавливается в состояние "1", при отсутствии переноса — в состояние "0". При переносе из 7-го разряда АЛУ триггер признака переноса C устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

Формат: ADD A, <Reg>

имя рабочего регистра R0...R7

Код:

0 1 1 0 1	R e g
-----------	-------

код рабочего регистра 000B...111B

Алгоритм: (AC): (TMP[0-3]) <= (A[0-3])+(Reg[0-3])
(C): (A) <= (A)+(Reg)

Время: 1 цикл

Пример: МЕТКА ADD A, R0 ; (A) <= (A)+(R0)

Команда ADD A, @<R>

По команде ADD A, @<R> содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, складывается с содержимым аккумулятора A. Результат вычисления записывается в аккумулятор.

При переносе из 3-го разряда АЛУ триггер признака вспомогательного переноса AC устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

При переносе из 7-го разряда АЛУ триггер признака переноса C устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

Формат: ADD A, @<R>

имя рабочего регистра R0...R1

Код:

0 1 1 0 0 0 0	R
---------------	---

код рабочего регистра 0B...1B

Алгоритм: (AC): (TMP[0-3]) <= (A[0-3]) <= A[0-3] + ((R)[0-3])
(C): (A) <= (A) + ((R))

Время: 1 цикл

Команда ADD A, #<DATA>

По команде ADD A, #<DATA> непосредственные данные, определяемые элементом <DATA>, складываются с содержимым аккумулятора. Результат вычисления записывается в аккумулятор.

При переносе из 3-го разряда АЛУ триггер признака вспомогательного переноса AC устанавливается в состояние "1", при отсутствии переноса — состояние "0".

При переносе из 7-го разряда АЛУ триггер признака переноса С устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

Формат: ADD A, #<DATA>

выражение для непосредственных данных

Код:

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

DATA

непосредственные данные 00H...FFH

Алгоритм: (AC): (TMP[0-3]) <= (A[0-3]) + DATA[0-3]

(C): (A) <= (A) + DATA

Время: 2 цикла

Пример: CONST EQU 17 ; CONST=11H

NEXT: MOV A, #15
ADD A, #CONST

; (A) <= 0FH
; (A) <= 0FH + 11H = 20H
; (AC) <= 1, (C) <= 0

Команда ADDC A, <Reg>

По команде ADDC A, <Reg> содержимое триггера признака переноса С складывается с содержимым аккумулятора А. Затем к содержимому аккумулятора прибавляется содержимое рабочего регистра <Reg>. Результат вычисления записывается в аккумулятор.

При переносе из 3-го разряд АЛУ в случае выполнения первой или второй операции сложения триггер признака вспомогательного переноса АС устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

При переносе из 7-го разряд АЛУ в случае выполнения первой или второй операции сложения триггер признака переноса С устанавливается в состояние "1", при отсутствии переноса — в состояние "0"

Формат: ADDC A, <Reg>

имя рабочего регистра R0...R7

Код:

0	1	1	1	1	Reg
---	---	---	---	---	-----

код рабочего регистра 000B...111B

Алгоритм: (AC): (TMP[0-3]) <= (A[0-3]) + (REG[0-3]) + (C)

(C): (A) <= (A) + (Reg) + (C)

Время: 1 цикл

Команда ADDC A, @<R>

По команде ADDC A, @<R> содержимое триггера признака переноса С складывается с содержимым аккумулятора А. Затем к содержимому аккумулятора прибавляется содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>. Результат вычислений записывается в аккумулятор.

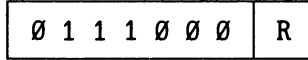
При переносе из 3-го разряда АЛУ в случае выполнения первой или второй операции сложения триггер признака вспомогательного переноса АС устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

При переносе из 7-го разряда АЛУ в случае выполнения первой или второй операции сложения триггер признака переноса С устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

Формат: ADDC A,@<R>

имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

Алгоритм: (AC): (TMP[0-3]) <= (A[0-3] + ((R)[0-3])) + (C)
 (C): (A) <= (A) + ((R)) + C

Время: 1 цикл

Команда ADDC A,#<DATA>

По команде ADDC A,#<DATA> содержимое признака переноса С складывается с содержимым аккумулятора А. Затем к содержимому аккумулятора прибавляются непосредственные данные, определяемые элементом <DATA>. Результат вычисления записывается в аккумулятор.

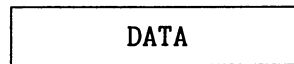
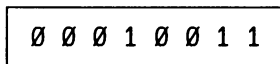
При переносе из 3-го разряда АЛУ в случае выполнения первой или второй операции сложения триггер признака вспомогательного переноса АС устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

При переносе из 7-го разряда АЛУ в случае выполнения первой или второй операции сложения триггер признака переноса С устанавливается в состояние "1", при отсутствии переноса — в состояние "0".

Формат: ADDC A,#<DATA>

выражение для непосредственных данных

Код:



непосредственные данные
 00H...FFH

Алгоритм: (AC): (TMP[0-3]) <= (A[0-3] + DATA[0-3]) + (C)
 (C): (A) <= (A) + DATA + C

Время: 2 цикла

Пример: ; (A) = 0FCH, (C) = 1
 NEXT: ADDC A, #3 ; (A) <= 0FCH + 03H + 1 = 00H
 ; (C) <= 1

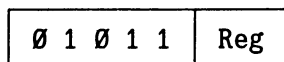
Команда ANL A,<Reg>

По команде ANL A,<Reg> содержимое рабочего регистра <Reg> поразрядно логически умножается на содержимое аккумулятора А. Результат вычисления записывается в аккумулятор.

Формат: ANL A,<Reg>

имя рабочего регистра R0...R7

Код:



|
код рабочего регистра 000B...111B

Алгоритм: $(A) \leftarrow (A) \text{ AND } (REG)$

Время: 1 цикл

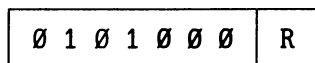
Команда ANL A,@<R>

По команде ANL A,@<R> содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, поразрядно логически умножаются на содержимое аккумулятора A. Результат вычисления записывается в аккумулятор.

Формат: ANL A,@<R>

|
имя рабочего регистра R0...R1

Код:



|
код рабочего регистра 0B...1B

Алгоритм: $(A) \leftarrow (A) \text{ AND } ((R))$

Время: 1 цикл

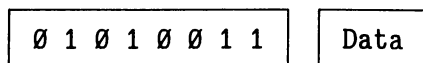
Команда ANL A,#<DATA>

По команде ANL A,#<DATA> непосредственные данные, определяемые элементом <DATA>, поразрядно логически умножаются на содержимое аккумулятора A. Результат вычисления записывается в аккумулятор.

Формат: ANL A,#<DATA>

|
выражение для непосредственных данных

Код:



|
непосредственные данные 00H...FFH

Алгоритм: $(A) \leftarrow (A) \text{ AND DATA}$

Время: 2 цикла

Пример: ;(A)=00001111B
 МЕТКА: ANL A,#01011100B ;(A)=00001100B

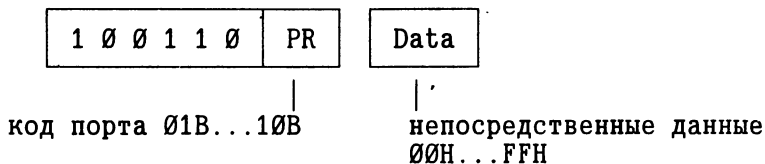
Команда ANL <PR>,#<DATA>

По команде ANL <PR>,#<DATA> непосредственные данные, определяемые элементом <DATA>, поразрядно логически умножаются на содержимое порта <PR>. Результат вычисления записывается в резидентный порт.

Формат: ANL <PR>,#<DATA>

|
выражение для непосредственных данных

Код:

Алгоритм: $(PR) \leftarrow (PR) \text{ AND } DATA$

Время: 2 цикла

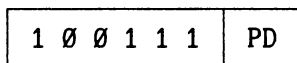
Команда ANLD <PD>, A

По команде ANLD <PD>, A содержимое дополнительного (четырёхразрядного) порта поразрядно логически умножается на содержимое 0...3 разряд аккумулятора A. Результат вычисления записывается в дополнительный порт.

Формат: ANLD <PD>, A

имя дополнительного порта P4...P7

Код:



код дополнительного порта 00B...11B

Алгоритм: $(PD) \leftarrow (PD) \text{ AND } (A[0-3])$

Время: 2 цикла

Команда CALL <ADR 11>

По команде CALL <ADR 11> осуществляется вызов подпрограммы следующим образом:

— содержимое счетчика команд PC (то есть полный 11-разрядный адрес следующей команды) и разрядов 4...7 слова состояния PSW записывается в вершину стека;

— содержимое указателя стека SP увеличивается на единицу;

— в разряды 0...10 счетчика команд записывается длинный адрес подпрограммы, определяемый элементом <ADR 11>;

Команда CALL не должна размещаться в ячейках 2046...2047.

Формат: CALL <ADR 11>

выражение для длинного (11- битного) адреса
ячейки банка памяти программ

Код:



ADR 11

длинный (11- битный) адрес ячейки
банка памяти программ 000H...7FFH

Алгоритм: $((SP)) \leftarrow (PC:PSW[4-7])$
 $(SP) \leftarrow (SP)+1$
 $(PC) \leftarrow ADR11$

Время: 2 цикла

Пример: МЕТКА: CALL ПРОЦЕД

 CALL ПРОЦЕД

ПРОЦЕД: MOV R0, #5

 . . .

 RET

Команда CLR A

По команде CLR A все разряды аккумулятора устанавливаются в состояние "0".

Формат: CLR A

Код:

0	0	1	0	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: (A) <= 0

Время: 1 цикл

Команда CLR C

По команде CLR C триггер признака переноса C устанавливается в состояние "0".

Формат: CLR C

Код:

1	0	0	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: (C) <= 0

Время: 1 цикл

Команда CLR F0

По команде CLR F0 триггер бита условия (флага) F0 устанавливается в состояние "0".

Формат: CLR F0

Код:

1	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: (F0) <= 0

Время: 1 цикл

Команда CLR F1

По команде CLR F1 триггер бита условия (флага) F1 устанавливается в состояние "1".

Формат: CLR F1

Код:

1	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: (F1) <= 0

Время: 1 цикл

Команда CPL A

По команде CPL A состояние всех разрядов аккумулятора A изменяется на противоположное.

Формат: CPL A

Код:

0	0	1	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: $(A) \leftarrow \text{NOT } (A)$

Время: 1 цикл

Пример: ;(A)=01010011B

МЕТКА: CPL A ;(A)=10101100B

Команда CPL C

По команде CPL C состояние триггера признака переноса C изменяется на противоположное.

Формат: CPL C

Код:

1	0	1	0	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: $(C) \leftarrow \text{NOT } (C)$

Время: 1 цикл

Команда CPL F0

По команде CPL F0 состояние триггера бита условия (флага) F0 изменяется на противоположное.

Формат: CPL F Формат: CPL F0

Код:

1	0	0	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: $(F0) \leftarrow \text{NOT } (F0)$

Время: 1 цикл

Команда CPL F1

По команде CPL F1 состояние триггера бита условия (флага) F1 изменяется на противоположное.

Формат: CPL F1

Код:

1	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: $(F1) \leftarrow \text{NOT } (F1)$

Время: 1 цикл

Команда DA A

По команде DA A результат двоичного сложения упакованных двоично-десятичных чисел в аккумуляторе A преобразуется в упакованное двоично-

десятичное число следующим образом. Если число в младших четырех разрядах аккумулятора больше девяти или триггер признака вспомогательного переноса АС установлен в состояние "1", то к содержимому аккумулятора прибавляется число 6; если после этого число в старших четырех разрядах аккумулятора больше девяти или триггер признака переноса С установлен в состояние "1", то к содержимому аккумулятора прибавляют число 60H.

При переносе из 7-го разряда АЛУ триггер признака переноса С устанавливается в состояние "1", в противном случае — в "0".

Формат: DA A

Код:

0	1	0	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: если $(A[0-3]) > 9$ OR $(AC)=1$
 то $(C): (A) \leq (A)+6$
 если $(A[4-7]) > 9$ OR $(C)=1$
 то $(C): (A) \leq (A)+60H$

Время: 1 цикл

Пример: NUMB1 EQU 51H ; NUMB1=51H
 NUMB2 EQU 19H ; NUMB2=19H

```

; 51+19=70
BEGIN: MOV A, #NUMB1   ; (A) <= 51H
      ADD A, #NUMB2   ; (A) <= 51H+19H=6AH
      DA A           ; (A) <= 6AH+6=70H

```

Команда DEC A

По команде DEC A содержимое аккумулятора A уменьшается на единицу.

Формат: DEC A

Код:

0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: $(A) \leq (A)-1$

Время: 1 цикл

Пример: ; (A) <= 12H
 МЕТКА: DEC A ; (A) <= 12H-1=11H

Команда DEC <Reg>

По команде DEC <Reg> содержимое рабочего регистра <Reg> уменьшается на единицу.

Формат: DEC <Reg>

имя рабочего регистра R0...R7

Код:

1	1	0	0	1	Reg
---	---	---	---	---	-----

код рабочего регистра 000B...111B

Алгоритм: $(\langle \text{Reg} \rangle) \leq (\langle \text{Reg} \rangle)-1$

Время: 1 цикл

Команда DIS I

По команде DIS I триггер разрешения IBF-прерываний устанавливается в состояние "0", тем самым запрещаются прерывания по заполнению входного буфера (IBF). При этом установка сигналов \overline{WR} и \overline{CS} не вызывает IBF-прерывания, однако запрос прерывания фиксируется в ожидании выполнения команды EN I (разрешение IBF-прерываний).

Флаг IBF устанавливается и очищается независимо от запроса IBF-прерывания, так что протокол установления связи может работать нормально.

Формат: DIS I

Код:

0 0 0 1 0 1 0 1

Алгоритм: (I) <= 0

Время: 1 цикл

Команда DIS TCNTI

По команде DIS TCNTI триггер разрешения внутренних прерываний TCNTI устанавливается в состояние "0", тем самым запрещаются внутренние прерывания от таймера-счетчика.

Формат: DIS TCNTI

Код:

0 0 1 1 0 1 0 1

Алгоритм: (TCNTI) <= 0

Время: 1 цикл

Команда DJNZ <Reg>, <ADR8>

По команде DJNZ <Reg>, <ADR8> осуществляется ветвление в программе следующим образом:

- содержимое рабочего регистра <Reg> уменьшается на единицу;
- если содержимое рабочего регистра <Reg> не равно нулю, то управление передается в точку назначения, адрес которой определен элементом <ADR8>, в противном случае осуществляется переход к следующей команде.

Если код первого байта команды DJNZ <Reg>, <ADR8> размещается в предпоследней или в последней ячейках страницы памяти программ, то точка назначения такой команды должна находиться на следующей странице. Данное замечание справедливо для всех команд условного перехода.

Формат: DJNZ <Reg>, <ADR8>

|
| выражение для короткого (8-битного)
| адреса памяти программ
| имя рабочего регистра R0...R7

Код:

1 1 0 0 1	Reg	ADR 8
-----------	-----	-------

|
код рабочего регистра
000B...111B

|
короткий (8-битный) адрес
ячейки внутри страницы
памяти программ 00H-0FFH

Алгоритм: (REG) <= (REG) - 1
 если (REG) <> 0
 то (PC[0-7]) <= ADR8,
 иначе (PC) <= (PC) + 2

Время: 2 цикла

Пример: MOV R0, #50 ; (R0) <= 50
 MOV R3, #6 ; (R3) <= 6
 ЦИКЛ: INC @R0 ; ((R0)) <= ((R0)) + 1
 INC R0 ; (R0) <= R0 + 1
 DJNZ R3, ЦИКЛ ; (R3) <= (R3) - 1
 ; если (R3) <> 0, то перейти на
 ; метку "цикл"

Команда EN DMA

Команда EN DMA имеется только в системе команд микросхем серии UPI-42.

По команде EN DMA разрешается квитирование DMA с использованием вывода P26 в качестве линии запроса DMA (DRQ — DMA Request) и вывода P27 в качестве линии подтверждения DMA ($\overline{\text{DACK}}$ — DMA ACKnowledge). Установление линии $\overline{\text{DACK}}$ вызывает принудительную внутреннюю установку сигналов $\overline{\text{CS}}$ и A0 в состояние низкого уровня и очищает DRQ.

Формат: EN DMA

Код:

1	1	1	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: (P26) <= (DRQ)
 (P27) <= ($\overline{\text{DACK}}$)

Время: 1 цикл

Команда EN FLAGS

По команде EN FLAGS выводы порта P24 и P25 становятся соответственно линиями отображения состояния флага OBF и инвертированного флага IBF — $\overline{\text{IBF}}$. Для правильной работы перед выполнением инструкции EN FLAGS необходимо записать "1" в выводы P24 и P25. Запись "0" в выводы P24 и P25 запрещает отображение флагов OBF и $\overline{\text{IBF}}$ по этим выводам.

Формат: EN FLAGS

Код:

1	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: (P24) <= (OBF)
 (P25) <= NOT (IBF)

Время: 1 цикл

Команда EN I

По команде DIS I триггер разрешения IBF-прерываний устанавливается в состояние "1", тем самым разрешаются прерывания по заполнению входного буфера (IBF). IBF-прерывание вырабатывается при установке сигналов $\overline{\text{WR}}$ и $\overline{\text{CS}}$ в состояние низкого уровня.

Формат: EN I

Код:

0 0 0 0 0 1 0 1

Алгоритм: $(I) < = 1$

Время: 1 цикл

Команда EN TCNTI

По команде EN TCNTI триггер разрешения внутренних прерываний TCNTI устанавливается в состояние "1", тем самым разрешаются прерывания от таймера-счетчика (то есть по переносу из 7-го разряда таймера-счетчика управление будет передано подпрограмме обслуживания прерывания с начальным адресом 007H).

Формат: EN TCNTI

Код:

0 0 1 0 0 1 0 1

Алгоритм: $(TCNTI) < = 1$

Время: 1 цикл

Команда IDL

Команда IDL имеется только в системе команд микросхем ЭКР1847ВГ6.

По команде IDL устанавливается режим микропотребления. При этом выполнение программы прекращается, внутренние фазовые и тактовые сигналы не формируются, сохраняется состояние всех функциональных узлов микросхемы (ОЗУ, портов, аккумулятора, PSW и др.), т. е. получается "спящий режим".

Выход из состояния микропотребления осуществляется с помощью сигналов \overline{CS} и \overline{WR} (подача уровня "0").

Формат: IDL

Код:

0 0 0 0 0 0 0 1

Время: 1 цикл

Команда IN A, DBB

По команде IN A, DBB данные из регистра DBBIN пересылаются в аккумулятор и флаг заполнения входного буфера (IBF) устанавливается в "0".

Формат: IN A, DBB

Код:

0 0 1 0 0 0 1 0

Алгоритм: $(A) < = (DBB)$
 $(IBF) < = 0$

Время: 1 цикл

Команда IN A, <PR>

По команде IN A, <PR> содержимое порта <PR> пересылается в аккумулятор A.

Формат: IN A, <PR>

|
имя порта P1...P2

Код:

0 0 0 0 1 0	Pr
-------------	----

код порта 01B...10B

Алгоритм: $(A) \leq (PR)$

Время: 2 цикла

Команда INC A

По команде INC A содержимое аккумулятора A увеличивается на единицу.

Формат: INC A

Код:

0 0 0 1 0 1 1 1

Алгоритм: $(A) \leq (A) + 1$

Время: 1 цикл

Команда INC <Reg>

По команде INC <Reg> содержимое рабочего регистра <Reg> увеличивается на единицу.

Формат: INC <Reg>

имя рабочего регистра R0...R7

Код:

0 0 0 1 1	Reg
-----------	-----

код рабочего регистра 000B...111B

Алгоритм: $(\langle \text{Reg} \rangle) \leq (\langle \text{Reg} \rangle) + 1$

Время: 1 цикл

Команда INC @<R>

По команде INC @<R> содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, увеличивается на единицу.

Формат: INC @<R>

имя рабочего регистра R0...R1

Код:

0 0 0 1 0 0 0	R
---------------	---

код рабочего регистра 0B...1B

Алгоритм: $((R)) \leq ((R)) + 1$

Время: 1 цикл

Пример: ;(R1)=40H, (40H)=05H
 МЕТКА : INC @R1 ;(40H) <= 05H + 1 = 06H

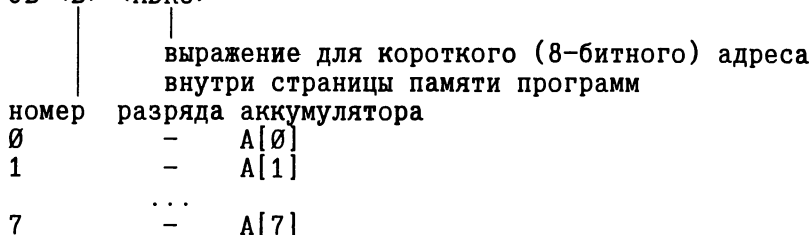
Команда JB <ADR8>

По команде JB <ADR8> осуществляется ветвление в программе следующим образом:

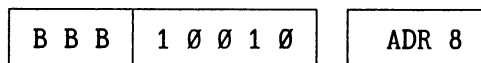
— если разряд аккумулятора установлен в состояние "1", то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JB <ADR8>



Код:



код номера разряда
аккумулятора:

000B — A[0]

001B — A[1]

...

111B — A[7]

короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (A[BBB])=1,
 то (PC[0-7])<=ADR8,
 иначе (PC)<=(PC)+2

Время: 2 цикла

Пример: IN A,P1 ; (A)<=(P1)
 JB4 CONT ; если (A[4])=1, то перейти
 ; на метку "CONT"

Команда JC <ADR8>

По команде JC <ADR8> осуществляется ветвление в программе следующим образом:

— если триггер признака переноса C установлен в состояние "1", то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JC <ADR8>

|
выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

1 1 1 1 0 1 1 0

ADR 8

короткий (8-битный) адрес ячейки внутри
страницы памяти программ 00H-FFH

Алгоритм: если (C)=1,
 то (PC[0-7])<=ADR8,
 иначе (PC)<=(PC)+2

Время: 2 цикла

Команда JF0 <ADR8>

По команде JF0 <ADR8> осуществляется ветвление в программе следующим образом:

— если триггер бита условия (флага) F0 установлен в состояние "1", то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JF0 <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

1 0 1 1 0 1 1 0

ADR 8

короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (F0)=1,
 то (PC[0-7])<=ADR8,
 иначе (PC)<=(PC)+2

Время: 2 цикла

Команда JF1 <ADR8>

По команде JF1 <ADR8> осуществляется ветвление в программе следующим образом:

— если триггер бита условия (флага) F1 установлен в состояние "1", то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JF1 <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

0 1 1 1 0 1 1 0

ADR 8

короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (F1)=1,
) то (PC[0-7])<=ADR8,
 иначе (PC)<=(PC)+2

Время: 2 цикла

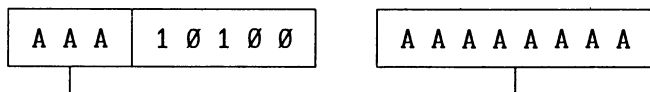
Команда JMP <ADR11>

По команде JMP <ADR11> осуществляется безусловный прямой переход в точку назначения, адрес которой определен элементом <ADR11> (в разряды 0...10 счетчика команд PC записывается длинный адрес точки назначения, указанный в коде команды).

Формат: JMP <ADR11>

выражение для длинного (11-битного) адреса
ячейки банка памяти программ

Код:



ADR 11

длинный (11-битный) адрес ячейки
банка памяти программ 000H...7FFH

Алгоритм: (PC[0-10])<=ADR11
 (PC[11]) <=(DBF)

Время: 2 цикла

Пример: МЕТКА: JMP +5 ;перейти на +5
 JMP MET ;перейти на "MET"

Команда JMPP @A

По команде JMPP @A осуществляется безусловный косвенный переход в точку назначения, адрес которой содержится в ячейке текущей страницы памяти программ, адресуемой аккумулятором А (в разряды 0...7 счетчика команд PC записывается содержимое ячейки памяти программ, адрес которой указан в аккумуляторе).

Формат: JMPP @A

Код:

1 0 1 1 0 0 1 1

Алгоритм: (PC[0-7])<=((A))

Время: 2 цикла

Команда JNC <ADR8>

По команде JNC <ADR8> осуществляется ветвление в программе следующим образом:

— если триггер признака переноса C установлен в состояние "0", то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JNC <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

1 1 1 0 0 1 1 0

ADR 8

короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (C)=0,
то (PC[0-7])<=ADR8,
иначе (PC)<=(PC)+2

Время: 2 цикла

Команда JNIBF <ADR8>

По команде JNIBF <ADR8> осуществляется ветвление в программе следующим образом:

— если триггер бита условия (флага) IBF установлен в состояние "0", то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JNIBF <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

1 1 0 1 0 1 1 0

ADR 8

короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (IBF)=0,
то (PC[0-7])<=ADR8,
иначе (PC)<=(PC)+2

Время: 2 цикла

Команда JNT0 <ADR8>

По команде JNT0 <ADR8> осуществляется ветвление в программе следующим образом:

— если сигнал на выводе T0 равен нулю, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JNT0 <ADR8>

|
выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

0 0 1 0 0 1 1 0

ADR 8

|
короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (T0)=0,
 то (PC[0-7])<=ADR8,
 иначе (PC)<=(PC)+2

Время: 2 цикла

Команда JNT1 <ADR8>

По команде JNT1 <ADR8> осуществляется ветвление в программе следующим образом:

— если сигнал на выводе T1 равен нулю, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JNT1 <ADR8>

|
выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

0 1 0 0 0 1 1 0

ADR 8

|
короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (T1)=0,
 то (PC[0-7])<=ADR8,
 иначе (PC)<=(PC)+2

Время: 2 цикла

Команда JNZ <ADR8>

По команде JNZ <ADR8> осуществляется ветвление в программе следующим образом:

— если содержимое аккумулятора не равно нулю, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JNZ <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

1 0 0 1 0 1 1 0

ADR 8

короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (A) <> 0,
то (PC[0-7]) <= ADR8,
иначе (PC) <= (PC) + 2

Время: 2 цикла

Команда JOBF <ADR8>

По команде JOBF <ADR8> осуществляется ветвление в программе следующим образом:

— если триггер бита условия (флага) OBF установлен в состояние "1", то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JOBF <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

1 0 0 0 0 1 1 0

ADR 8

короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (OBF) = 1,
то (PC[0-7]) <= ADR8,
иначе (PC) <= (PC) + 2

Время: 2 цикла

Команда JTF <ADR8>

По команде JTF <ADR8> осуществляется ветвление в программе следующим образом:

— если триггер переполнения таймера-счетчика TF установлен в состояние "1" (был перенос из 7-го разряда таймера-счетчика), то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JTF <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

0 0 0 1 0 1 1 0

ADR 8

короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (TF) = 1,
 то (PC[0-7]) <= ADR8,
 иначе (PC) <= (PC) + 2
 (TF) <= 0

Время: 2 цикла

Команда JT0 <ADR8>

По команде JT0 <ADR8> осуществляется ветвление в программе следующим образом:

— если сигнал на выводе T0 равен единице, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JT0 <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

0 0 1 1 0 1 1 0

ADR 8

короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (T0) = 1,
 то (PC[0-7]) <= ADR8,
 иначе (PC) <= (PC) + 2

Время: 2 цикла

Команда JT1 <ADR8>

По команде JT1 <ADR8> осуществляется ветвление в программе следующим образом:

— если сигнал на выводе T1 равен единице, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JT1 <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

0 1 0 1 0 1 1 0

ADR 8

короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (T1)=1,
 то (PC[0-7])<=ADR8,
 иначе (PC)<=(PC)+2

Время: 2 цикла

Команда JZ <ADR8>

По команде JZ <ADR8> осуществляется ветвление в программе следующим образом:

— если содержимое аккумулятора равно нулю, то управление передается в точку назначения, адрес которой определен элементом <ADR8> (в разряды 0...7 счетчика команд PC записывается короткий адрес точки назначения, указанный в коде команды).

— в противном случае осуществляется переход к следующей команде (содержимое счетчика увеличивается на 2).

Формат: JZ <ADR8>

выражение для короткого (8-битного) адреса
внутри страницы памяти программ

Код:

1 1 0 0 0 1 1 0

ADR 8

короткий (8-битный)
адрес ячейки внутри
страницы памяти
программ 00H-FFH

Алгоритм: если (A)=0,
 то (PC[0-7])<=ADR8,
 иначе (PC)<=(PC)+2

Время: 2 цикла

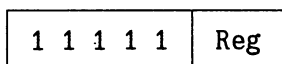
Команда MOV A, <Reg>

По команде MOV A, <Reg> содержимое рабочего регистра <Reg> пересылается в аккумулятор A.

Формат: MOV A, <Reg>

имя рабочего регистра R0...R7

Код:



код рабочего регистра 000B...111B

Алгоритм: <A> <= <Reg>

Время: 1 цикл

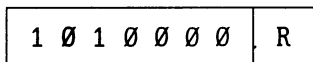
Команда MOV @R, A

По команде MOV @R, A содержимое аккумулятора A пересылается в ячейку внутренней памяти данных, адресуемую рабочим регистром <R>.

Формат: MOV @R, A

имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

Алгоритм: ((R)) <= <A>

Время: 1 цикл

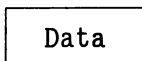
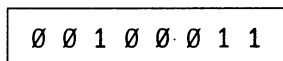
Команда MOV A, #<Data>

По команде MOV A, #<Data> непосредственные данные, определяемые элементом <Data>, пересылаются в аккумулятор A.

Формат: MOV A, #<Data>

выражение для непосредственных данных

Код:



непосредственные данные 00H...FFH

Алгоритм: <A> <= Data

Время: 2 цикла

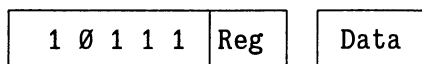
Команда MOV <Reg>, #<Data>

По команде MOV <Reg>, #<Data> непосредственные данные, определяемые элементом <Data> пересылаются в рабочий регистр <Reg>.

Формат: MOV <Reg>, #<Data>

имя рабочего регистра R0...R7

Код:



код рабочего регистра
000B...111B

непосредственные данные 00H...FFH

Алгоритм: (Reg) <= Data

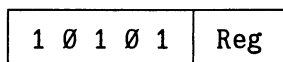
Команда MOV <Reg>, A

По команде MOV <Reg>, A содержимое аккумулятора пересылается в рабочий регистр <Reg>.

Формат: MOV <Reg>, A

имя рабочего регистра R0...R7

Код:



код рабочего регистра 000B...111B

Алгоритм: (Reg) <= <A>

Время: 1 цикл

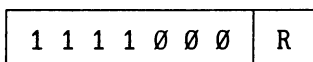
Команда MOV A, @<R>

По команде MOV A, @<R> содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, пересылается в аккумулятор A.

Формат: MOV A, @<R>

имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

Алгоритм: <A> <= ((R))

Время: 1 цикл

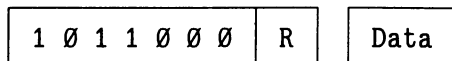
Команда MOV @<R>, #<Data>

По команде MOV @<R>, #<Data> непосредственные данные, определяемые элементом <Data>, пересылаются в ячейку внутренней памяти данных, адресуемую рабочим регистром <R>.

Формат: MOV @<R>, #<Data>

выражение для непосредственных данных
имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

непосредственные данные
00H...FFH

Алгоритм: ((R)) <= DATA

Время: 2 цикла

Команда MOV A, PSW

По команде MOV A, PSW содержимое регистра слова состояния программы PSW пересылается в аккумулятор A.

Формат: MOV A, PSW

Код:

1	1	0	0	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: (A[0-2]) <= (SP)
 (A[3]) <= 1
 (A[4]) <= (BS)
 (A[5]) <= (F0)
 (A[6]) <= (AC)
 (A[7]) <= <C>

Время: 1 цикл

Команда MOV PSW, A

По команде MOV PSW, A содержимое аккумулятора A пересылается в регистр слова состояния программы PSW.

Формат: MOV PSW, A

Код:

1	1	0	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: (SP) <= (A[0-2])
 PSW[3] <= 1
 (BS) <= (A[4])
 (F0) <= (A[5])
 (AC) <= (A[6])
 <C> <= (A[7])

Время: 1 цикл

Команда MOV A, T

По команде MOV A, T содержимое таймера-счетчика T пересылается в аккумулятор A.

Формат: MOV A, T

Код:

0	1	0	0	0	0	1	0
---	---	---	---	---	---	---	---

Алгоритм: <A> <= (T)

Время: 1 цикл

Команда MOV T, A

По команде MOV T, A содержимое аккумулятора A пересылается в таймер-счетчик T.

Формат: MOV T, A

Код:

0	1	1	0	0	0	1	0
---	---	---	---	---	---	---	---

Алгоритм: (T) <= A

Время: 1 цикл

Команда MOV STS, A

По команде MOV STS, A содержимое разрядов 4...7 аккумулятора A пересылается в регистр состояния STATUS.

Формат: MOV STS, A

Код:

1	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---

Алгоритм: STS[4-7] <= (A[4-7])

Время: 1 цикл

Команда MOVD A, <PD>

По команде MOVD A, <PD> содержимое дополнительного (4-разрядного) порта <PD> пересылается в 0...3 разряды аккумулятора A. Разряды 4...7 аккумулятора устанавливаются в состояние "0".

Формат: MOVD A, <PD>

имя дополнительного порта P4...P7

Код:

0	0	0	0	1	1
---	---	---	---	---	---

PD

код дополнительного порта 00B...11B

Алгоритм: (A[0-3]) <= (PD)
(A[4-7]) <= 0

Время: 2 цикла

Команда MOVD <PD>, A

По команде MOVD <PD>, A содержимое аккумулятора A пересылается в дополнительный (4-разрядный) порт <PD>.

Формат: MOVD <PD>, A

имя дополнительного порта P4...P7

Код:

0	0	1	1	1	1
---	---	---	---	---	---

PD

код дополнительного порта 00B...11B

Алгоритм: (PD) <= (A[0-3])

Время: 2 цикла

Команда MOVP A, @A

По команде MOV A, @A содержимое ячейки текущей страницы памяти программ, адресуемое аккумулятором A, пересылается в аккумулятор.

Формат: MOVP A, @A

Код:

1	0	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Алгоритм: (A) <= ((PC[8-11]:A))

Время: 2 цикла

Пример: CONST: DB 'ABCDEF'

NEXT: MOV A, #LOW(CONST)
MOVP A, @A ; (A) <= 'A'

Команда MOVP3 A, @A

По команде MOVP3 A, @A содержимое ячейки третьей страницы памяти программ, адресуемое аккумулятором A, пересылается в аккумулятор.

Формат: MOVP3 A, @A

Код:

1	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Алгоритм: (A) <= ((3:A))

Время: 2 цикла

Команда NOP

По команде NOP выполняется холостая операция (содержимое счетчика команд увеличивается на единицу).

Формат: NOP

Код:

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

Алгоритм: (PC) <= (PC) + 1

Время: 1 цикл

Команда ORL A, <Reg>

По команде ORL A, <Reg> содержимое рабочего регистра <Reg> поразрядно логически складывается с содержимым аккумулятора A. Результат вычисления записывается в аккумулятор.

Формат: ORL A, <Reg>

имя рабочего регистра R0...R7

Код:

0	1	0	0	1	Reg
---	---	---	---	---	-----

код рабочего регистра 000B...111B

Алгоритм: (A) <= (A) OR (REG)

Время: 1 цикл

Команда ORL A, @<R>

По команде ORL A, @<R> содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, поразрядно логически складывается с содержимым аккумулятора A. Результат вычисления записывается в аккумулятор.

Формат: ORL A, @<R>

имя рабочего регистра R0...R1

Код:

0	1	0	0	0	0	0	R
---	---	---	---	---	---	---	---

код рабочего регистра 0B...1B

Алгоритм: $(A) \leq (A) \text{ OR } ((R))$

Время: 1 цикл

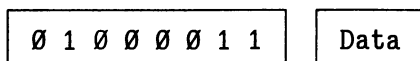
Команда ORL A, #<DATA>

По команде ORL A, #<DATA> непосредственные данные, определяемые элементом <DATA>, поразрядно логически складываются с содержимым аккумулятора A. Результат вычисления записывается в аккумулятор.

Формат: ORL A, #<DATA>

выражение для непосредственных данных

Код:



непосредственные данные 00H...FFH

Алгоритм: $(A) \leq (A) \text{ OR DATA}$

Время: 2 цикла

Пример:

; (A)=10101100B
 МЕТКА: ORL A, #0FH ; DATA=00001111B
 ; (A)=10101111B

Команда ORL <PR>, #<DATA>

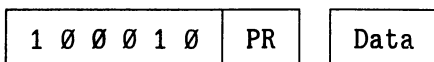
По команде ORL <PR>, #<DATA> непосредственные данные, определяемые элементом <DATA>, поразрядно логически складываются с содержимым порта <PR>. Результат вычисления записывается в резидентный порт.

Формат: ORL <PR>, #<DATA>

имя порта
P1...P2

выражение для непосредственных данных

Код:



код порта 01B...10B

непосредственные данные 00H...FFH

Алгоритм: $(PR) \leq (PR) \text{ OR DATA}$

Время: 2 цикла

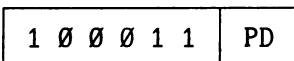
Команда ORLD <PD>, A

По команде ORLD <PD>, A содержимое A поразрядно логически складывается с содержимым дополнительного (4-разрядного) порта <PD>. Результат вычисления записывается в дополнительный порт.

Формат: ORLD <PD>, A

имя дополнительного порта P4...P7

Код:



код дополнительного порта 00B...11B

Алгоритм: $(PD) \leq (PD) \text{ OR } (A[0-3])$

Время: 2 цикла

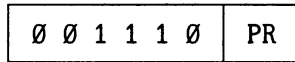
Команда OUTL <PR>, A

По команде OUTL <PR>, A содержимое аккумулятора A пересылается в порт <PR>. Содержимое порта остается без изменений до последующей операции записи в порт.

Формат: OUTL <PR>, A

|
имя порта P1...P2

Код:



|
код порта 01B...10B

Алгоритм: (PR) <= (A)

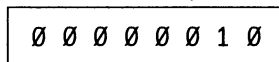
Время: 2 цикла

Команда OUT DBB, A

По команде OUT DBB, A содержимое аккумулятора A пересылается в выходной регистр шины данных DBBUOT, и флаг OBF устанавливается в "1".

Формат: OUT DBB, A

Код:



Алгоритм: (DBB) <= (A)
(OBF) <= 1

Время: 1 цикла

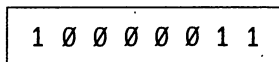
Команда RET

По команде RET осуществляется возврат из подпрограммы без восстановления содержимого регистра слова состояния программы PSW следующим образом:

- содержимое указателя стека SP уменьшается на единицу;
- полный адрес возврата из подпрограммы, находящийся в вершине стека, пересылается в счетчик PC;
- содержимое разрядов 4...7 регистра слова состояния программы не изменяется.

Формат: RET

Код:



Алгоритм: (SP) <= (SP) - 1
(PC) <= ((SP) [0-11])

Время: 2 цикла

Команда RETR

По команде RETR осуществляется возврат из подпрограммы с восстановлением содержимого регистра слова состояния программы PSW следующим образом:

- содержимое указателя стека SP уменьшается на единицу;
- полный адрес возврата из подпрограммы, находящийся в вершине стека, пересылается в счетчик PC;
- система прерываний разблокируется;
- содержимое разрядов 4...7 регистра слова состояния программы восстанавливается по содержимому стека.

Формат: RETR

Код:

1	0	0	1	0	0	1	1
---	---	---	---	---	---	---	---

Алгоритм: $(SP) \leq (SP) - 1$
 $(PC) \leq ((SP)[0-11])$
 $(PSW[4-7]) \leq ((SP)[12-15])$

Время: 2 цикла

Команда RL A

По команде RL A содержимое аккумулятора A циклически сдвигается влево на один двоичный разряд.

Формат: RL A

Код:

1	1	1	0	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: для N от 6 до 0,
 $(A[N+1]) \leq (A[N])$
 $(A[0]) \leq (A[7])$

Время: 1 цикл

Пример: ; (A) = 11110000B
 МЕТКА: RL A ; (A) <= 11100001B

Команда RLC A

По команде RLC A содержимое аккумулятора A и триггера признака переноса C циклически сдвигается влево на один двоичный разряд.

Формат: RLC A

Код:

1	1	1	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: $(TMP) \leq C$
 $(C) \leq (A[7])$
 для N от 6 до 0,
 $(A[N+1]) \leq (A[N])$
 $(A[0]) \leq (TMP)$

Время: 1 цикл

Пример: ; (A) = 11110000B, (C) = 0
 МЕТКА: RLC A ; (A) <= 11100000B, (C) = 1

Команда RR A

По команде RR A содержимое аккумулятора A циклически сдвигается вправо на один двоичный разряд.

Формат: RR A

Код:

0	1	1	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: для N от 6 до 0,
 $(A[N]) \leq (A[N+1])$
 $(A[7]) \leq (A[0])$

Время: 1 цикл

Команда RRC A

По команде RRC A содержимое аккумулятора A и триггера признака переноса C циклически сдвигается вправо на один двоичный разряд.

Формат: RRC A

Код:

1	1	1	1	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: (TMP) <= C
 (C) <= (A[0])
 для N от 0 до 6,
 (A[N]) <= (A[N+1])
 (A[7]) <= (TMP)

Время: 1 цикл

Команда SEL RB0

По команде SEL RB0 триггер переключения банка рабочих регистров BS устанавливается в состояние "0" (то есть производится переключение на нулевой банк рабочих регистров RB0). После выполнения данной команды рабочим регистрам R0...R7 будет соответствовать внутренняя память данных с адресами 0...7.

Формат: SEL RB0

Код:

1	1	0	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: (BS) <= 0

Время: 1 цикл

Команда SEL RB1

По команде SEL RB1 триггер переключения банка рабочих регистров BS устанавливается в состояние "1" (то есть производится переключение на первый банк рабочих регистров RB1). После выполнения данной команды рабочим регистрам R0...R7 будет соответствовать внутренняя память данных с адресами 24...31.

Формат: SEL RB1

Код:

1	1	0	1	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: (BS) <= 1

Время: 1 цикл

Пример:

```

ORG 3
JNI INIT
ORG 10H
INIT: MOV R7, A
      SEL RB1
      MOV R7, #0FAH

      SEL RB0
      MOV A, R7
      RETR

```

Команда STOP TCNT

По команде STOP TCNT таймер-счетчик TCNT останавливается (прекращается инкрементирование). Запуск таймера-счетчика осуществляется следующей командой STRT CNT или STRT T.

Формат: STOP TCNT

Код:

0	1	1	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: останов таймера-счетчика

Время: 1 цикл

Команда STRT CNT

По команде STRT CNT таймер-счетчик подключается к выводу T1 и запускается в качестве счетчика внешних событий (то есть по сигналу на выводе T1 содержимое таймера-счетчика увеличивается на единицу).

Формат: STRT CNT

Код:

0	1	0	0	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: запуск таймера-счетчика в качестве счетчика

Время: 1 цикл

Команда STRT T

По команде STRT T таймер-счетчик подключается к генератору тактовых сигналов и запускается в качестве внутреннего таймера.

Содержимое таймера-счетчика увеличивается на единицу через каждые 32 машинных цикла. Отсчет циклов начинается непосредственно после выполнения команды STRT T.

Формат: STRT T

Код:

0	1	0	1	0	1	0	1
---	---	---	---	---	---	---	---

Алгоритм: запуск таймера-счетчика в качестве таймера

Время: 1 цикл

Команда SWAP A

По команде SWAP A 0...3 и 4...7 разряды аккумулятора A обмениваются содержимым.

Формат: SWAP A

Код:

0	1	0	0	0	1	1	1
---	---	---	---	---	---	---	---

Алгоритм: $(A[4-7]) \leftrightarrow (A[0-3])$

Время: 1 цикл

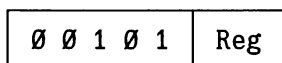
Команда XCH A, <Reg>

По команде XCH A, <Reg> рабочий регистр <Reg> и аккумулятор A обмениваются содержимым.

Формат: XCH A, <Reg>

имя рабочего регистра R0...R7

Код:



код рабочего регистра 000B...111B

Алгоритм: (A) <=> (REG)

Время: 1 цикл

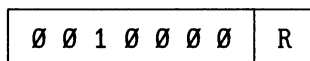
Команда XCH A, @<R>

По команде XCH A, @<R> ячейка внутренней памяти данных, адресуемой рабочим регистром <R>, и аккумулятор A обмениваются содержимым.

Формат: XCH A, @<R>

имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

Алгоритм: <A> <=> ((R))

Время: 1 цикл

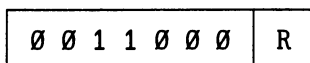
Команда XCHD A, @<R>

По команде XCHD A, @<R> 0...3 разряды ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, и 0...3 разряды аккумулятора A обмениваются содержимым (содержимое 4...7 разрядов ячейки памяти и аккумулятора не изменяются).

Формат: XCHD A, @<R>

имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

Алгоритм: (A[0-3]) <=> ((R)[0-3])

Время: 1 цикл

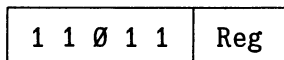
Команда XRL A, <Reg>

По команде XRL A, <Reg> содержимое рабочего регистра <Reg> и содержимое аккумулятора A поразрядно складываются по модулю 2. Результат вычисления записывается в аккумулятор.

Формат: XRL A, <Reg>

имя рабочего регистра R0...R7

Код:



код рабочего регистра 000B...111B

Алгоритм: $(A) \leftarrow (A) \text{ XOR } (\text{REG})$

Время: 1 цикл

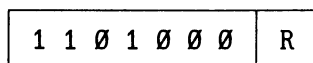
Команда XRL A, @<R>

По команде XRL A, @<R> содержимое ячейки внутренней памяти данных, адресуемой рабочим регистром <R>, и содержимое аккумулятора A поразрядно складываются по модулю 2. Результат вычисления записывается в аккумулятор.

Формат: XRL A, @<R>

имя рабочего регистра R0...R1

Код:



код рабочего регистра 0B...1B

Алгоритм: $(A) \leftarrow (A) \text{ XOR } (R)$

Время: 1 цикл

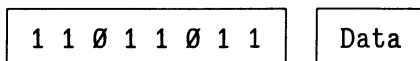
Команда XRL A, #<Data>

По команде XRL A, #<Data> непосредственные данные, определяемые элементом <DATA>, и содержимое аккумулятора A поразрядно складываются по модулю 2. Результат вычисления записывается в аккумулятор.

Формат: XRL A, #<Data>

выражение для непосредственных данных

Код:



непосредственные данные 00H...FFH

Алгоритм: $(A) \leftarrow (A) \text{ XOR } \text{DATA}$

Время: 2 цикла

Пример:

МЕТКА : XRL A, #0FH

; (A) = 0 1 0 1 1 1 0 0 B
; DATA = 0 0 0 0 1 1 1 1 B
; (A) = 0 1 0 1 0 0 1 1 B

8.6. Электрические параметры

Электрические параметры микросхем ЭКР1847ВГ6 в диапазоне температур от минус 10°C до 70°C приведены в табл. 8.9 (статические параметры) и табл. 8.10 (динамические параметры).

Предельные значения электрических режимов эксплуатации приведены в табл. 8.11.

Таблица 8.9. Электрические параметры микросхем ЭКР1847ВГ6 в диапазоне температур от минус 10°C до 70°C

N п/п	ПАРАМЕТРЫ	БУКВЕННОЕ ОБОЗНАЧЕНИЕ	ЗНАЧЕНИЕ ПАРАМЕТРОВ	
			ЭКР1847ВГ6	
			мин.	макс.
1	Напряжение питания, В	U_{CC}	4,5	5,5
2	Выходное напряжение, В – высокого уровня	U_{OH}	2,4	–
	– низкого уровня	U_{OL}	–	0,45
3	Выходной ток высокого уровня, мА – сигналов D0–D7	I_{OH}	–0,4	–
	– сигнала G2	I_{OH1}	–0,025	–
	– остальных сигналов	I_{OH2}	–0,04	–
4	Выходной ток низкого уровня, мА – сигналов D0–D7	I_{OL}	–	2,5
	– сигнала G2	I_{OL1}	–	0,05
	– остальных сигналов	I_{OL2}	–	1,6
5	Входное напряжение высокого уровня, В – сигналов G1, CS, RD, Ao, WR	U_{IH1}	3,7	U_{CC}
	– сигнала M при чтении ПЗУ	$U_{IH,M}$	10,0	11,0
	– остальных сигналов	U_{IH}	2,4	U_{CC}
6	Входное напряжение низкого уровня, В – всех сигналов	U_{IL}	0	0,8
	– сигнала M при чтении ПЗУ	$U_{IL,M}$	4,5	5,5
7	Ток утечки на входах, мкА – сигналов SR, SS	U_{LI}	10	100
	– остальных сигналов	U_{LI1}	–10	10
8	Выходной ток в состоянии "выключено", мкА – сигналов D0–D7	U_{OZ}	–10	10
	– остальных сигналов	U_{OZ1}	10	100
9	Ток потребления статический, мкА	I_{CCS}	–	20
	динамический, мА/МГц	I_{CCd}		1
10	Емкость входа/выхода, пф	$C_{I/O}$	–	10
11	Входная емкость, пф	C_I	–	5

Таблица 8.10. Электрические параметры в диапазоне температур от минус 10°C до 70°C при $U_{CC} = 5 \text{ В} \pm 10\%$

N п/п	НАИМЕНОВАНИЕ ПАРАМЕТРА, ЕДИНИЦА ИЗМЕРЕНИЯ	БУКВЕННОЕ ОБОЗНАЧЕНИЕ	ЗНАЧЕНИЕ ПАРАМЕТРОВ	
			ЭКР1847ВГ6	
			не менее	не более
ДИНАМИЧЕСКИЕ ПАРАМЕТРЫ				
1.	Clock Period Период следования импульсов тактового сигнала G1, нс	$T_{G1}=t$	125	1000
			t	
2.	PROG Pulse Width Длительность сигнала PROG, нс	$t_w(\overline{PROG}, L)$	1080	-
			10, 5t-230	
3	Port Control Setup to PROG Время задержки сигнала PROG относительно сигналов управления дополнительным портом P2(0-3), нс	$t_D(CP2, LH/HL-\overline{PROG}, HL)$	110	-
			1, 5t-80	
4	Port Control Hold to PROG Время задержки сигналов управления дополнительным портом P2(0-3) относительно сигнала PROG, нс	$t_D(\overline{PROG}, HL-CP2, HL/LH)$	100	-
			1, 5t-90	
5	PROG to P2 Input Valid Время установления сигналов данных дополнительного порта P2(0-3) относительно сигнала PROG, нс	$t_{SU}(\overline{PROG}, HL-DP2, LH/HL)$	-	1000
			9t-120	
6	Input Data Hold from PROG Время сохранения сигналов данных дополнительного порта P2(0-3) относительно сигнала PROG, нс	$t_V(\overline{PROG}, LH-DP2, HL/LH)$	0	70
			0t+70	
7	Output Data Setup Время задержки сигнала PROG относительно сигнала данных для дополнительного порта P2(0-3), нс	$t_D(DP2, LH/HL-\overline{PROG}, LH)$	470	-
			6t-280	
8	Output Data Hold Время задержки сигналов данных для дополнительного порта P2(0-3) относительно сигнала PROG, нс	$t_D(\overline{PROG}, LH-DP2, HL/LH)$	100	-
			1, 5t-90	
9	Port Output from STR Время задержки сигналов данных портов P1, P2 относительно сигнала STR, нс	$t_D(STR, HL-DP, HL/LH)$	-	660
			4, 5t+100	

- Примечания:
1. Символ "LH" ("HL") обозначает переход сигнала из состояния низкого (высокого) уровня в состояние высокого (низкого) уровня.
 2. Черта между символами "HL/LH", "LH/HL" означает, что параметр имеет одинаковое значение для обоих переходов.
 3. Значения временных параметров приведены для емкостной нагрузки по входам-выходам $C_{LI/O} < 50$ пФ и емкостной нагрузки по выходам $C_L < 50$ пФ.
 4. Значения временных параметров указаны при частоте генератора 8 МГц.

Таблица 8.11. Предельные значения электрических режимов эксплуатации

N п/п	НАИМЕНОВАНИЕ ПАРАМЕТРА, ЕДИНИЦА ИЗМЕРЕНИЯ	БУКВЕННОЕ ОБОЗНАЧЕНИЕ	Н О Р М А	
			ЭКР1847ВГ6	
			не менее	не более
1	Напряжение питания, В	U_{CC}	—	6,0
2	Входное напряжение высокого уровня, В	U_{IH}	—	$U_{CC}+0,3$
3	Входное напряжение низкого уровня, В	U_{IL}	—0,3	—
4	Время фронта нарастания и спада входных сигналов, нс, для $f_{G1} = 8$ МГц	t_{HL}, t_{LH}	—	30
5	Емкость нагрузки, пФ	C_L	—	200

8.7. Применение УПИМК

На основе УПИМК можно легко проектировать различные интерфейсы периферийных устройств малой и средней скорости. Следующие примеры показывают некоторые типичные применения УПИМК.

На рис. 8.35 показана блок-схема клавиатуры использующая УПИМК и расширитель портов ввода-вывода, который используется для сканирования 128-клавишной матрицы. Устройство состоит из схемы сканирования клавиатурной матрицы, схемы обработки одновременного нажатия нескольких клавиш, таблицы ПЗУ, буфера символов типа FIFO (первым вошел — первым вышел) и выходов используемых для индикации состояния некоторых контрольных клавиш и других специальных функций.

Сопряжение с клавиатурной матрицей обеспечивается при помощи порта P1 и портов P4—P7. Определение нажатия клавиши происходит путем сканирования клавиатурной матрицы, которое происходит под управлением программы сканирования. Процесс сканирования матрицы происходит следующим образом: в каждый момент времени последовательно выбирается один из восьми рядов активизацией линий порта P1; затем при выбранном ряде последовательно просматриваются все 16 столбцов матрицы активизацией линий портов P4—P7 для определения замкнутой клавиши. Каждая точка матрицы имеет свой собственный бинарный код, который используется для адресации ПЗУ когда определено замыкание клавиши. ПЗУ содержит таблицу с используемыми кодами (такими как ASCII, EBCDIC и др.), которые соответствуют каждой клавише. При определении замыкания соответствующий этой клавише код из таблицы ПЗУ сохраняется в буфере FIFO в ОЗУ данных для передачи главному процессору (ГП). Для предотвращения ложных срабатываний замыкание клавиши считается действительным только тогда, когда оно обнаружено двумя последовательными сканированиями. Только тогда код клавиши загружается в буфер FIFO.

Другим типичным применением УПИМК является интерфейс матричного принтера, блок-схема которого показана на рис. 8.36. Два порта УПИМК позволяют использовать до 16 линий ввода-вывода для управления механизмом принтера. Используя таймер-счетчик можно получить временные последовательности для управления положением печатающей головки, перевода строк, возврата каретки и других необходимых функций. Внутреннее ПЗУ программ обеспечивает генерацию символов формата 5 x 7, 7 x 9 или другого необходимого формата. ОЗУ данных может быть использовано в качестве буфера FIFO для символов предназначенных к печати. При этом ГП может загрузить блок данных в буфер на высокой скорости и возвратиться к выполнению другой задачи, а УПИМК будет выдавать символ из буфера со скоростью, которую допускает данный принтер.

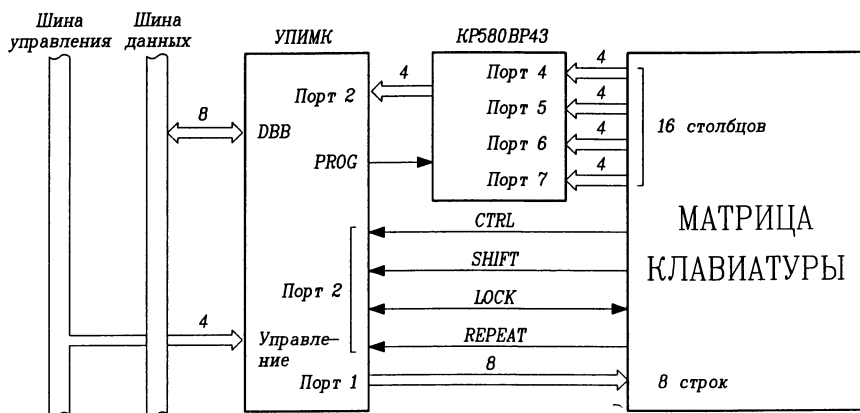


Рис. 8.35. Блок-схема клавиатуры, выполненная на базе УПИМК

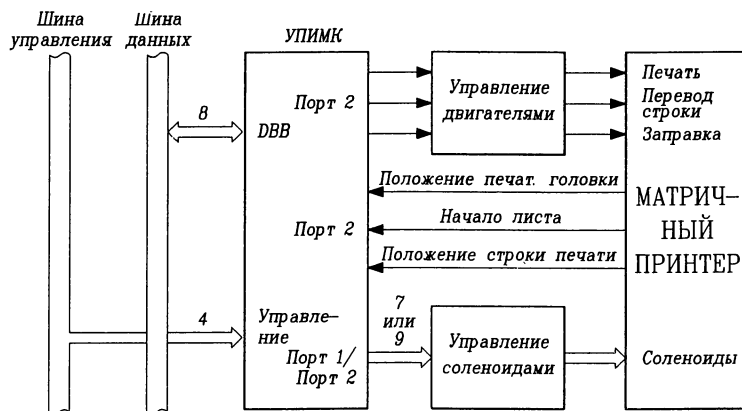


Рис. 8.36. Блок-схема интерфейса матричного принтера

С небольшими изменениями УПИМК можно использовать для других типов принтеров, таких как: барабанный, ленточный, ромашковый и со сферической головкой.

На рис. 8.37 показано еще одно возможное применение УПИМК — интерфейс накопителя на магнитной ленте. УПИМК осуществляет как сервоуправление мотором, так и управление формирователем сигналов записи/считывания. Сервоуправление мотором обеспечивает контроль скорости в форме моностабильных импульсов, ширина которых пропорциональна требуемой скорости. При воспроизведении предварительно записанные на ленту синхроимпульсы используются встроенным таймером для формирования необходимых импульсов контроля скорости. Данные с ленты последовательно поступают в блок формирователя сигналов записи/считывания, преобразовываются УПИМК в 8-битные слова и затем передаются на ГП. При скорости ленты 25,4 см/сек УПИМК может поддерживать скорость обмена до 8 Кбит/сек. Для записи данных УПИМК использует две входных линии формирователя сигналов записи/считывания, который управляет направлением магнитного потока в головке записи. 4 линии контролируют состояние лентопотяжного механизма, в частности: "конец ленты", "кассета установлена", "занято" и "запись разрешена". Все сигналы управления вырабатываются двумя портами УПИМК.

Наиболее просто на основе УПИМК можно реализовать универсальный интерфейс ввода-вывода. Он состоит (рис. 8.38) из 12 линий параллельного порта ввода-вывода и линий последовательного порта RS232C осуществляющих полную дуплексную передачу данных со скоростью до 1200 бод. Этот тип устройства может

быть использован главным процессором как интерфейс для широкого спектра периферийных устройств и каналов последовательной передачи данных.

В данном примере порт P1 используется только в качестве каналов ввода-вывода. Порт P2 используется как в качестве каналов ввода-вывода (линии P20...P23), так и в качестве специальных линий последовательного порта RS232C, таких как RTS (request to send — запрос на передачу данных) (P24), CTS (clear to send — готовность для передачи данных) (P25), прерывание (P26) и передача данных (P27).

Двунаправленная структура портов УПИМК позволяет легко получить интерфейс параллельного порта ввода-вывода. При начальной инициализации сигналом \overline{SR} все линии порта устанавливаются в "1" и фиксируются.

4 линии порта P2 функционируют подобно линиям ввода-вывода порта P1. Сигнал RTS генерируется портом P2 при помощи программного обеспечения когда УПИМК содержит последовательность данных для передачи. Сигнал CTS контролирует передачу данных. Линия порта P2 может также программно генерировать прерывание ГП, когда УПИМК готов принять или передать байт данных. При необходимости линии P24 и P25 могут быть использованы для создания прерываний по OBF и IBF при помощи команды EN FLAGS.

Для приема последовательных данных используется вывод T0. Формирователи A0 и A1 обеспечивают согласование технических требований передачи данных по RS232C.

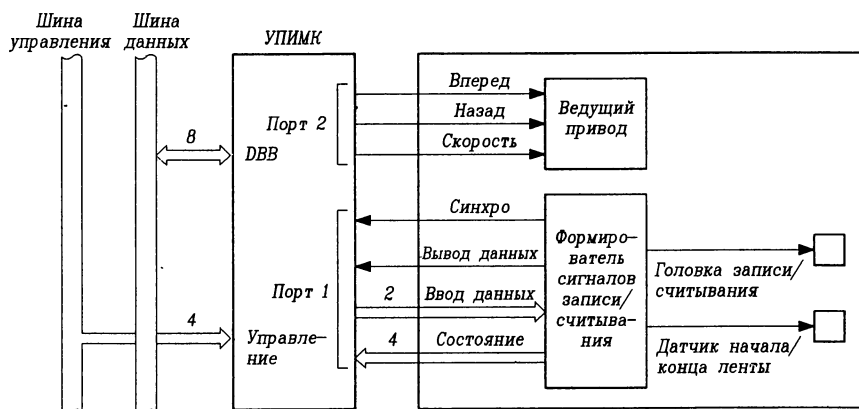


Рис. 8.37. Интерфейс накопителя на магнитной ленте

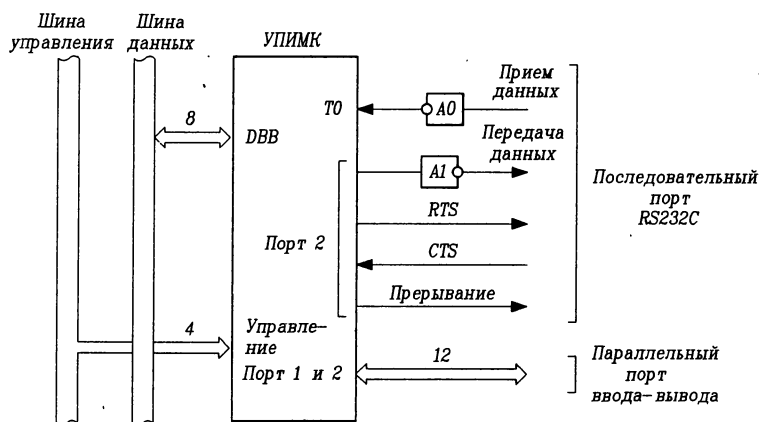


Рис. 8.38. Универсальный интерфейс ввода-вывода

8.8. Аналоги фирмы Intel

Универсальные периферийные интерфейсные микроконтроллеры (УПИМК) серии UPI-42/42АН выполнены по n-канальной МОП технологии и по системе команд, назначению выводов и архитектуре совместимы с ИС ЭКР1847ВГ6. Различия состоят в размере ОЗУ и типе ПЗУ программ, кроме того микросхемы серии UPI-42АН имеют, в отличие от UPI-42, дополнительные режимы работ (сигнатурный и синхронный).

Серия UPI-42/42АН имеет следующий состав:

- 8742 - микроконтроллер с УФППЗУ;
- 8742АН - микроконтроллер с ПЗУ с одноразовой прошивкой (ППЗУ);
- 8042АН - микроконтроллер с ПЗУ масочного типа;
- 8242АН - ИС 8042АН с фирменной прошивкой Phoenix Technologies Ltd.

клавиатурного контроллера для АТ-совместимых систем.

Отличительные особенности микросхем даны в таблице 8.12, режимы работ и коды доступа для микросхем серии UPI-42АН — в таблице 8.13.

Информация о микросхемах фирмы Intel приводится по каталогу ф. Intel "PERIPHERAL COMPONENTS", 1992 г. Фирма Intel постоянно улучшает и видоизменяет свои изделия. Поэтому, если Вы собираетесь работать с микросхемами фирмы Intel, желательно связаться с представителями этой фирмы для уточнения характеристик имеющихся у Вас приборов.

Таблица 8.12

UPI-42	UPI-42АН
МАКСИМАЛЬНАЯ ТАКТОВАЯ ЧАСТОТА	
12 МГц	12,5 МГц
ОЗУ И ПЗУ ПРОГРАММ	
2К * 8 УФППЗУ	2К * 8 ПЗУ (8042АН) 2К * 8 ППЗУ (8742АН)
128 * 8 ОЗУ	256 * 8 ОЗУ
РЕЖИМ ПРОГРАММИРОВАНИЯ	
$V_{dd} = 21 \text{ В}$ $I_{dd} = 50 \text{ мА}$ $E_A = 18 \text{ В}$ $V_{PH} = 18 \text{ В}$ $T_{PW} = 50 \text{ мсек}$	$12,5 \text{ В}$ 30 мА $12,5 \text{ В}$ $5,5 \text{ В}$ 1 мсек
ДОПОЛНИТЕЛЬНОЕ НАЗНАЧЕНИЕ ВЫВОДОВ	
(T0) Вывод T0 дополнительно используется при программировании и проверке УФППЗУ.	Вывод T0 дополнительно используется при программировании и проверке ППЗУ. Он также используется в синхронном режиме работы.
(RESET) Используется при программировании и проверке УФППЗУ.	Используется при программировании и проверке ППЗУ.
(SS) Используется совместно с выводом SYNC для реализации пошагового режима в процессе программирования УФППЗУ. В обычном режиме работы должен находиться в состоянии +5 В.	Используется совместно с выводом SYNC для реализации пошагового режима в процессе программирования ППЗУ. В обычном режиме работы должен находиться в состоянии +5 В. Также используется для реализации синхронного режима работы путем установления потенциала +12,5 В.
(EA) Используется при проверке УФППЗУ	Используется при проверке ПЗУ и ППЗУ
Порт P1 (P10–P17) 8-битовый двунаправленный порт ввода/вывода.	Порт P1 (P10–P17) 8-битовый двунаправленный порт ввода/вывода. Используется при доступе к сигнатурному ряду и биту защиты.

Таблица 8.13. Режимы работ микросхем серии UPI-42АН

Режим работы		Сигналы управления						Шина данных		Код доступа															
										Порт 2			Порт 1												
		T0	RST	SS	EA	PROG	V _{DD}	V _{CC}	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
Режим программирования		0	0	1	HV	1	V _{DDH}	V _{CC}	Адрес		Адрес	a ₀ a ₁ X X X X X X X													
		0	1	1	HV	STB	V _{DDH}	V _{CC}	Ввод данных		Адрес														
Режим проверки		0	0	1	HV	1	V _{CC}	V _{CC}	Адрес		Адрес	a ₀ a ₁ X X X X X X X													
		1	1	1	HV	1	V _{CC}	V _{CC}	Вывод данных		Адрес														
Режим синхронизации		STB _H	0	HV	0	X	V _{CC}	V _{CC}	X X X X X X X X		X X X	X X X X X X X X													
Сигнатурный режим	Прогр.	0	0	1	HV	1	V _{DDH}	V _{CC}	Адрес*		0 0 0	0 1 1 1 1 X X 1													
		0	1	1	HV	STB	V _{DDH}	V _{CC}	Ввод данных		0 0 0														
	Пров.	0	0	1	HV	1	V _{CC}	V _{CC}	Адрес*		0 0 0														
		1	1	1	HV	1	V _{CC}	V _{CC}	Вывод данных		0 0 0														
Защитный бит/байт	Прогр.	0	0	1	HV	1	V _{DDH}	V _{CC}	Адрес		0 0 0														
		0	1	1	HV	STB	V _{DDH}	V _{CC}	Ввод данных		0 0 0														
	Пров.	0	0	1	HV	1	V _{CC}	V _{CC}	Адрес		0 0 0														
		1	1	1	HV	1	V _{CC}	V _{CC}	Вывод данных		0 0 0														

* - см. табл. 8.17.
HV - высокий уровень сигнала.
V_{DDH} - высокий уровень сигнала V_{DD} (см. табл. 8.16).
STB - строб.
STB H - строб высокого уровня.
Прогр. - режим программирования.
Пров. - режим проверки.
a₀ = 0 или 1, a₁ = 0 или 1, уровни a₀ и a₁ обязательно должны быть одинаковы.

Электрические параметры микросхем семейства UPI-42/42АН в диапазоне температур от 0°С до 70°С приведены в табл. 8.14 (статические параметры).
Предельные значения электрических режимов эксплуатации приведены в таблице 8.15, статические параметры в режиме программирования ППЗУ — в таблице 8.16.

Таблица 8.14. Электрические параметры микросхем семейства UPI-42/42АН в диапазоне температур от 0°C до 70°C, $V_{CC} = V_{DD} = +5 В \pm 10\%$

N п/ п	ПАРАМЕТРЫ	БУКВЕННОЕ ОБОЗНАЧЕНИЕ	ЗНАЧЕНИЕ ПАРАМЕТРОВ		
			UPI-42/42АН		РЕЖИМ ИЗМЕРЕНИЙ
			мин.	макс.	
1	Входное напряжение низкого уровня, В – сигналов XTAL1, XTAL2, RESET – остальных сигналов	V_{IL1} V_{IL}	-0,5 -0,5	0,6 0,8	
2	Входное напряжение высокого уровня, В – сигналов XTAL1, RESET – сигнала XTAL2 – остальных сигналов	V_{IH1} V_{IH2} V_{IH}	3,5 2,2 2,0	V_{CC} V_{CC} V_{CC}	
3	Выходное напряжение низкого уровня, В – сигналов D0-D7 – сигналов P10-P17, P20-P27, SYNC – сигнала PROG	V_{OL} V_{OL1} V_{OL2}	- - -	0,45 0,45 0,45	$I_{OL} = 2,0 \text{ мА}$ $I_{OL} = 1,6 \text{ мА}$ $I_{OL} = 1,0 \text{ мА}$
4	Выходное напряжение высокого уровня, В – сигналов D0-D7 – остальных сигналов	V_{OH} V_{OH}	2,4 2,4	- -	$I_{OH} = -400 \text{ мкА}$ $I_{OH} = -50 \text{ мкА}$
5	Входной ток утечки, мкА – сигналов T0, T1, RD, WR, CS, AO, EA	I_{IL}	-10	10	$V_{SS} \leq V_{IN} \leq V_{CC}$
6	Входной ток утечки, мкА – сигналов D0-D7 в состоянии 'выключено'	I_{OFL}	-10	10	$V_{SS} + 0,45 \leq V_{OUT} \leq V_{CC}$
7	Входной ток низкого уровня, мА – сигналов P10-P17 – сигналов RESET, SS	I_{LI} I_{LI1}	- -	0,3 0,2	$V_{IL} = 0,8 \text{ В}$ $V_{IL} = 0,8 \text{ В}$
8	Ток потребления, мА – по выводу V_{DD} – суммарный – в режиме пониженного энергопотребления (только для UPI-42АН)	I_{DD} $I_{DD}+I_{CC}$ $I_{DD}Standby$	- - -	20,0 135,0 20,0	Типов. значение 8 мА Типов. значение 80 мА Типов. значение 8 мА
9	Входной ток утечки, мкА – сигналов P10-P17, P20-P27	I_{IH}		100	$V_{IN} = V_{CC}$
10	Входная емкость, пФ	C_{IN}	-	10	$T_A = 25^\circ\text{C}$
12	Емкость входа/выхода, пФ	C_{IO}	-	20	$T_A = 25^\circ\text{C}$

Таблица 8.15. Предельные значения электрических режимов эксплуатации

N п/п	НАИМЕНОВАНИЕ ПАРАМЕТРА, ЕДИНИЦА ИЗМЕРЕНИЯ	Н О Р М А	
		UPI-42/42AH	
		не менее	не более
1	Напряжение питания, В	-	7,0
2	Входное напряжение высокого уровня, В	-	7,0
3	Входное напряжение низкого уровня, В	-0,5	-
4	Температура режима эксплуатации, °C	0	70
5	Температура хранения, °C	-65	150
6	Рассеиваемая мощность, Вт	-	1,5

Таблица 8.16. Электрические параметры микросхем серии UPI-42/42AH в режиме программирования ППЗУ при температуре 25°C ±5°C, V_{CC}* = 6 В ±0,25 В, V_{DD} = 21 В ±0,5 В (для UPI-42), V_{DD} = 12,5 В ±0,5 В (для UPI-42AH)

ПАРАМЕТРЫ	БУКВЕННОЕ ОБОЗНАЧЕНИЕ	ЗНАЧЕНИЕ ПАРАМЕТРОВ			
		UPI-42		UPI-42AH	
		мин.	макс.	мин.	макс.
Напряжение программирования высокого уровня, В - сигнала V _{DD}	V _{DDH}	20,5	21,5	12	13**
- сигнала PROG	V _{PH}	17,5	18,5	2,0	5,5
Напряжение низкого уровня, В - сигнала V _{DD}	V _{DDL}	4,75	5,25	4,75	5,25
- сигнала PROG	V _{PL}	V _{CC} -0,5	V _{CC}	-0,5	0,8
- сигнала EA	V _{EAL}		5,25	-0,5	5,25
Входное напряжение высокого уровня, В (для UPI-42AH) - сигнала EA***	V _{EAH}			12	13
Напряжение программирования и проверки высокого уровня, В (для UPI-42) - сигнала EA	V _{EAH}	17,5	18,5		
Ток потребления высокого уровня, В - по выводу V _{DD}	I _{DD}		30,0		50
- по выводу EA	I _{EA}		1,0		1,0
- по выводу PROG (для UPI-42)	I _{PROG}		1,0		

* напряжение V_{CC} должно подаваться одновременно или раньше V_{DD} и должно сниматься одновременно или после V_{DD}.
** напряжение свыше 13 В, подаваемое на вывод V_{DD} может вывести микросхему из строя.
*** напряжение V_{EAH} должно подаваться на вывод EA до V_{DDH} и сниматься после V_{DDL}.

Сигнатурный режим (только для UPI-42АН)

Микросхемы серии UPI-42АН имеют 32 байта памяти ППЗУ, содержащие дополнительную информацию, записанную изготовителем или пользователем в процессе программирования ППЗУ. 32 байта памяти распределены следующим образом:

Тест кода/контрольной суммы — занимает до 25 байтов и содержит информацию о проверке внутренних узлов схемы, которые не могут быть проверены в режиме внешнего доступа. Тест кода/контрольной суммы имеется в микросхемах с ПЗУ и ППЗУ.

Сигнатура фирмы Intel — содержит информацию о производителе и точное название устройства, что позволяет автоматизировать процесс идентификации микросхемы. Ячейка 10Н содержит код изготовителя (для фирмы Intel код 89Н), ячейка 11Н — код устройства (код 43Н соответствует микросхеме 8042АН, код 42Н — 8742АН). Код 44Н используется для устройств с битом защиты, установленным ф. Intel. Имеется в микросхемах с ПЗУ и ППЗУ.

Сигнатура пользователя — позволяет пользователю записать свой собственный двухбитовый код версии прошивки ППЗУ в процессе программирования ППЗУ.

Сигнатура проверки — содержит информацию о проверке устройства. Используется при контроле качества продукции в процессе изготовления.

Бит защиты — служит для определения наличия бита защиты в запрограммированном ПЗУ.

Сигнатурный режим доступен при установке $P10 = 0$, $P11-P17 = 1$ в процедурах программирования и проверки. Адреса ячеек памяти, используемых в этом режиме и их содержимое приведены в таблице 8.17.

Таблица 8.17.

	Адрес		Тип устройства	Количество байт
Тест кода/контрольной суммы	0 16Н	0FH 1EH	ПЗУ/ППЗУ	25
Сигнатура фирмы Intel	10Н	11Н	ПЗУ/ППЗУ	2
Сигнатура пользователя	12Н	13Н	ППЗУ	2
Сигнатура проверки	14Н	15Н	ПЗУ/ППЗУ	2
Бит защиты	1FH		ППЗУ	1

Режим пониженного энергопотребления (только для UPI-42АН)

Режим пониженного энергопотребления реализован только в ИС серии UPI-42АН. В этом режиме питание может быть снято со всех элементов схемы кроме ОЗУ, которое должно находиться в режиме микропотребления. При этом содержимое ОЗУ остается неизменным, а энергопотребление может быть снижено до 10—15% от нормального. Вся схема UPI-42АН кроме ОЗУ питается от источника +5 В через вывод V_{CC} . ОЗУ питается через отдельный вывод V_{DD} . В нормальном состоянии выводы V_{CC} и V_{DD} подсоединены к одному источнику +5 В. Для сохранности данных в ОЗУ сигнал RESET должен быть установлен в низкое состояние перед переходом в режим микропотребления. После этого можно снять питание с вывода V_{CC} , оставив его только на выводе V_{DD} . Рекомендуется следующая последовательность действий (рис. 8.39):

1. Возможное отключение питания предварительно определяется специальной схемой, которая формирует сигнал "Отказ питания". Чтобы успеть сохранить все

необходимые данные в ОЗУ этот сигнал должен выдаваться до снятия питания с вывода V_{CC} .

2. Сигнал "Отказ питания" прерывает работу процессора (например через прерывание по таймеру) и вызывает программу обслуживания по "Отказу питания".

3. Эта программа сохраняет данные и PSW в ОЗУ, и также может выдавать команду на резервирование питания на выводе V_{DD} и сообщать внешней системе о завершении работы программы по "Отказу питания".

4. Внешней схемой подается сигнал низкого уровня на вывод \overline{SR} , чем обеспечивается сохранение данных во внутреннем ОЗУ. Сигнал \overline{SR} должен находиться в состоянии низкого уровня до тех пор, пока напряжение на выводе V_{CC} не достигнет нулевого потенциала.

Выход из режима микропотребления обеспечивается подачей на вывод V_{CC} напряжения питания, при этом конденсатор емкостью 1 мкФ соединенный с выводом RESET обеспечит импульс для начальной инициализации.

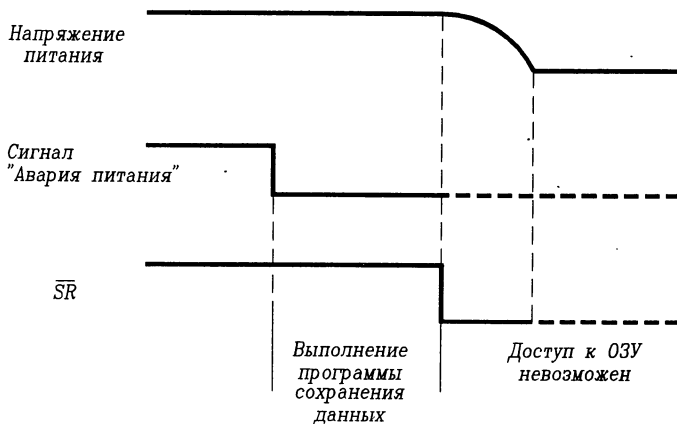
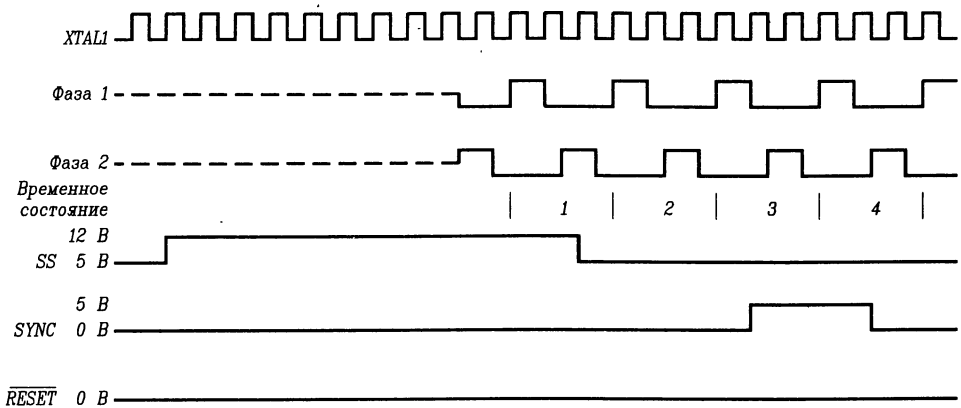


Рис. 8.39. Диаграмма сигналов в режиме микропотребления

Режим синхронизации (только для UPI-42АН)

Наличие режима синхронизации облегчает разработку мультиконтроллерных систем благодаря возможности установки устройства в известную фазу и временное состояние. Режим синхронизации может быть также использован при автоматической проверке оборудования для быстрой и эффективной синхронизации между тестером и тестируемым устройством.

Устройство переводится в синхронный режим подачей напряжения высокого уровня (+12 В) на вывод \overline{SS} . Сигнал $T\emptyset$ устанавливается в состояние высокого уровня (+5 В) не позднее чем через 4 такта $X1$ (частоты тактового генератора, подаваемой на вывод XTAL1) после сигнала \overline{SS} . Сигнал $T\emptyset$ должен находиться в состоянии высокого уровня не менее 4 тактов $X1$, необходимых для сброса предварительного делителя частоты и генерации временного состояния (см. разд. 8.2.4). Сигнал $T\emptyset$ может быть сброшен в период нахождения сигнала $X1$ в состоянии низкого уровня. Два такта спустя одновременно с переходом $X1$ в состояние высокого уровня устройство переходит во временное состояние 1, фазу 1. После этого сигнал \overline{SS} может быть сброшен в состояние низкого уровня (+5 В) через 4 такта после сигнала $T\emptyset$. Для возвращения в режим нормального выполнения команд необходимо через $5t_{cy}$ (75 тактов $X1$) установить сигнал RESET в состояние высокого уровня. Диаграмма работы в режиме синхронизации представлена на рис. 8.40.



- Примечания.
- 1. Длительность сигнала SYNC, $t_{\text{SYNC}} = 3,5 X1$.
 - 2. Длительность сигнала RESET, $t_{\text{RS}} = 4t_{\text{cy}} (60 X1)$.
 - 3. Переход сигнала T0 из состояния низкого уровня в высокое и наоборот должно происходить в период низкого состояния сигнала XTAL1.

Рис. 8.40. Диаграмма работы в режиме синхронизации

Приложение 1.

Использование микросхем КР1810ВК56, КМ1821РУ55, КМ1821РЕ55, К573РФ10 совместно с микроЭВМ семейств МК48 и МК51

Существуют несколько микросхем, особенно удобных для использования совместно с однокристальными микроЭВМ семейств МК48 и МК51 при необходимости расширения функциональных возможностей последних. К таким микросхемам относятся КР1810ВК56, КМ1821РУ55, КМ1821РЕ55, К573РФ10. Все эти микросхемы объединяет наличие внутренней защелки адреса, вследствие чего для стыковки их с микроЭВМ семейств МК48 и МК51 не требуется никаких дополнительных схем.

КМ1821РУ55 является статическим ОЗУ с портами ввода/вывода и таймером. Изготавливается по КМОП технологии в 40-выводном металлокерамическом корпусе типа 2123.40-6 с двусторонним расположением выводов. Микросхема содержит статическое ОЗУ с организацией 256×8 , три порта ввода/вывода, два из которых (А и В) восьмиразрядные и один (С) — шестиразрядный и 14-разрядный двоичный счетчик/таймер.

Порты ввода/вывода обеспечивают режимы простого и стробируемого ввода/вывода. Задание режимов работы портов осуществляется программно путем записи в регистр команд микросхемы управляющего слова определенного формата. В стробируемом режиме ввод или вывод осуществляется только через порты А и В, порт С при этом обеспечивает прием и выдачу сигналов управления.

Все разряды портов А и В используются для ввода или вывода параллельно, т. е. установка направления передачи индивидуально для каждого разряда невозможна. Сказанное в полной мере относится также и к порту С, если он не используется для формирования сигналов управления при работе портов А и В в стробируемом режиме.

Таймер представляет собой 14-разрядный вычитающий счетчик, подсчитывающий импульсы, подаваемые на тактовый вход таймера С, и формирующий на выходе \bar{Q} импульсы или последовательности импульсов в зависимости от заданного режима работы.

Аналогом микросхемы КМ1821РУ55 является изготавливаемая по nМОП технологии микросхема 8155 фирмы Intel.

На рис. 1 показана возможная схема включения микросхемы КМ1821РУ55 совместно с однокристальными микроЭВМ семейств МК48 и МК51.

При сигнале лог. 1 на входе $\text{IO}/\bar{\text{M}}$ выполняются обращения к портам ввода/вывода и регистрам команд и состояния микросхемы КМ1821РУ55, а при сигнале лог. 0 — к внутреннему ОЗУ. Запись и чтение информации производится соответственно по сигналам $\bar{\text{WR}}$ и $\bar{\text{RD}}$. МикроЭВМ обращается к ресурсам КМ1821РУ55 в режиме работы с внешней памятью данных.

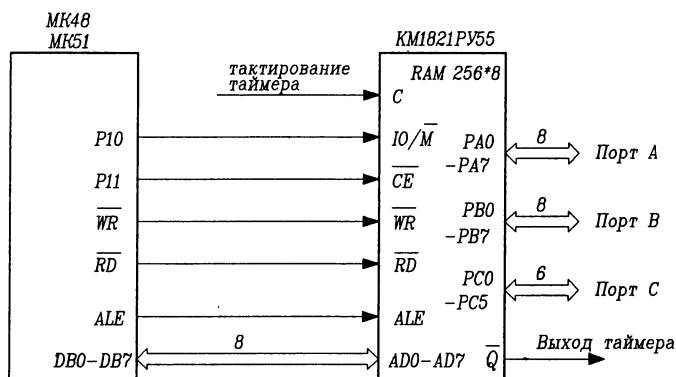


Рис. 1. Схема включения МК48, МК51 совместно с БИС КМ1821РУ55

КМ1821РЕ55 и К573РФ10 являются соответственно масочным ПЗУ и ППЗУ с ультрафиолетовым стиранием с организацией 2 К*8 с портами ввода/вывода. За исключением типа памяти указанные микросхемы идентичны. Изготавливаются в металлокерамических 40-выводных корпусах с двусторонним расположением выводов (типа DIP).

Кроме ППЗУ (УФППЗУ) микросхемы содержат два восьмиразрядных порта ввода/вывода, независимо программируемых на ввод или вывод по каждому разряду.

Аналогами микросхем КМ1821РЕ55 и К573РФ10 являются соответственно микросхемы 8355 и 8755 фирмы Intel, изготавливаемые по nМОП технологии.

На рис. 2 показана возможная схема включения микросхем КМ1821РЕ55, К573РФ10 совместно с однокристальными микроЭВМ семейств МК48 и МК51.

Сигнал $I0/\overline{M}$ имеет то же функциональное назначение, что и для КМ1821РУ55. В показанной на рис. 2 схеме в режиме работы с внешней памятью программ МК48 и МК51 по сигналу \overline{PME} обращаются по чтению к постоянной памяти микросхем КМ1821РЕ55, К573РФ10, а в режиме работы с внешней памятью данных по сигналам \overline{WR} и \overline{RD} работают с портами ввода/вывода.

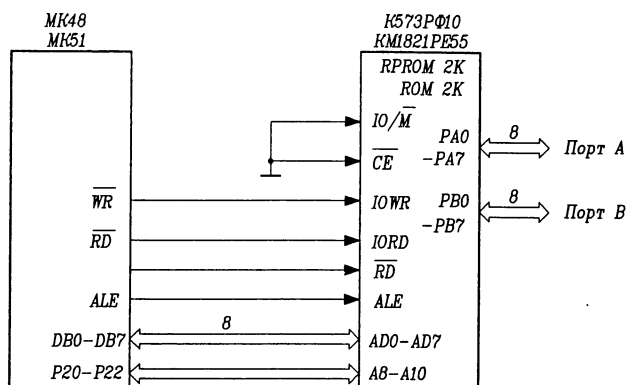


Рис. 2. Схема включения МК48, МК51 совместно с БИС КМ1821РУ55 и К573РФ10

КР1810ВК56 — многофункциональный универсальный контроллер поддержки микропроцессора, предназначенный для увеличения эффективности микропроцессорных систем путем объединения функций четырех контроллеров периферии (КР580ВВ51А, КР580ВВ55А, КР1810ВВ54, КР1810ВВ59А) в одном корпусе.

БИС КР1810ВК56 изготавливаются по nМОП технологии и выпускаются в 40-выводном пластмассовом корпусе 2123.40-2 с двусторонним расположением выводов.

БИС КР1810ВК56 обеспечивает выполнение следующих функций:

- программируемого последовательного асинхронного интерфейса с разрядностью информационного кода 5, 6, 7 или 8 бит; контроль по четности/нечетности; формирование стоп-бита, равного 1; 1,5 или 2 битам информации. Скорость обмена последовательного приемопередатчика до 19,2 Кбит/с без использования внешней синхронизации и до 1 Мбит/с при использовании внешней синхронизации;

- пяти программируемых восьмиразрядных таймеров/счетчиков, четыре из которых могут быть скаскадированы в два шестнадцатиразрядных таймера-счетчика;

- двух программируемых восьмибитных параллельных портов ввода/вывода, причем порт 1 может быть запрограммирован для квитирования порта 2 и выполнения функций счетчика внешних событий. Каждый из восьми бит порта 1 может быть запрограммирован как вход или как выход, а восемь бит порта 2 могут

объединяться по два полубайта, индивидуально программируемых как входы или как выходы;

— программируемого восьмиуровневого контроллера прерываний, семь уровней которого обслуживают внутренние ресурсы микросхемы, а один обслуживает внешние запросы на прерывание.

Аналогом БИС КР1810ВК56 является микросхема 8256АН фирмы Intel.

На рис. 3 представлена возможная схема включения КР1810ВК56 совместно с однокристальными микроЭВМ семейств МК48 и МК51.

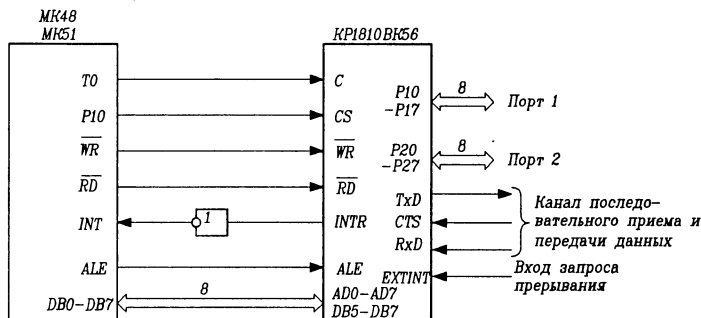


Рис. 3. Схема включения МК48, МК51 совместно с БИС КР1810ВК56

Приложение 2.
Производители и поставщики микросхем однокристалльных микроЭВМ семейств
МК48 и МК51

Информация приводится по состоянию на 1 августа 1992 года.

Серия 1816

ПО "Квазар"

252136, г. Киев, ул. Сырецкая, 1

Номер телетайпа	131077
Позывной телетайпа	"Антей"
Приемная, телефон	442-70-42, 434-83-44
Отдел сбыта	434-88-48, 442-61-19

Завод "Квантор"

283830, г. Збараж Тернопольской обл., ул. Ленина, 6

Номер телетайпа	202185
Позывной телетайпа	"Корунд"
Приемная, телефон	2-15-32, 2-18-61
Отдел сбыта	2-19-54

ПО "Родон"

284021, г. Ивано-Франковск, ул. Волчинецкая, 225

Номер телетайпа	292130
Позывной телетайпа	"Лазер"
Приемная, телефон	3-22-23, 3-01-80
Отдел сбыта	2-22-50, 9-32-56

Завод "Квадр"

251080, г. Борзна Черниговской обл., ул. К. Маркса, 113

Номер телетайпа	192825
Позывной телетайпа	"Темп"
Приемная, телефон	2-11-47, 2-11-97
Отдел сбыта	2-10-92, 2-12-75

НПО "Электроника"

394007, г. Воронеж, Ленинский проспект, 119а

Номер телетайпа	153154
Позывной телетайпа	"Тайм"
Приемная, телефон	22-04-81, 22-57-09
Отдел сбыта	22-06-64, 22-05-80

Серия 1830

ПО "Квазар"

ПО "Родон"

Серия 1835, 1847

НПО "Интеграл"

220064, г. Минск, ул. Казинца

Номер телетайпа	252168
Позывной телетайпа	"Скиф"
Приемная, телефон	77-32-22, 77-32-21
Отдел сбыта	77-24-32

Разработчики: Прохорчик С. В., Сергеев А. А., НПО "Интеграл",

тел. 78-31-98

Серия 1850

НПП "Восток"

630075, г. Новосибирск, ул. Дуси Ковальчук, 276

Номер телетайпа	1095
Позывной телетайпа	"Тантал"
Телефон	26-47-17, Букреев Е. В.

Приложение 3.
ОМЭВМ семейства MCS®-48 и MCS®-51 выпускаемые фирмой Intel

Микроконтроллеры семейства MCS®-48

Устройство	Память программ (байт)	ОЗУ (байт)	Таймер/счетчик	Каналов аналогового ввода	Линий вв./выв.	Тактовая частота (МГц)
8048AH	1К ПЗУ	64	1	0	27	11
8035AHL	внешняя	64	1	0	27	11
8049AH	2К ПЗУ	128	1	0	27	11
8039AHL	внешняя	128	1	0	27	11
8050AH	4К ПЗУ	256	1	0	27	11
8040AHL	внешняя	256	1	0	27	11
8748H	1К УФПЗУ	64	1	0	27	11
8749H	2К УФПЗУ	128	1	0	27	11

Микроконтроллеры семейства MCS®-51

8031AH	внешняя	128	2	0	32	12
8051AH	4К ПЗУ	128	2	0	32	12
8751H	4К УФПЗУ	128	2	0	32	12
8751BH	4К УФПЗУ	128	2	0	32	12
8032AH	внешняя	256	3	0	32	12
8052AH	8К ПЗУ	256	3	0	32	12
8752BH	8К УФПЗУ	256	3	0	32	12
80C31BH	внешняя	128	2	0	32	12, 16
80C51BH	4К ПЗУ	128	2	0	32	12, 16
87C51	4К УФПЗУ	128	2	0	32	12, 16, 20, 24i
80C32	внешняя	256	3	0	32	12, 16, 20, 24i
80C52	8К ПЗУ	256	3	0	32	12, 16, 20, 24i
87C52	8К УФПЗУ	256	3	0	32	12, 16, 20, 24i
80C54	внешняя	256	3	0	32	12, 16, 20, 24i
87C54	16К УФПЗУ	256	3	0	32	12, 16, 20, 24i
80C58	внешняя	256	3	0	32	12, 16, 20, 24i
87C58	32К УФПЗУ	256	3	0	32	12, 16, 20, 24i
80C51FA	внешняя	256	3	0	32	12, 16
83C51FA	8К ПЗУ	256	3	0	32	12, 16
87C51FA	8К УФПЗУ	256	3	0	32	12, 16, 20, 24i
83C51FB	16К ПЗУ	256	3	0	32	12, 16, 20, 24i
87C51FB	16К УФПЗУ	256	3	0	32	12, 16, 20, 24i
83C51FC	32К ПЗУ	256	3	0	32	12, 16, 20, 24i
87C51FC	32К УФПЗУ	256	3	0	32	12, 16, 20, 24i
80C51GB	внешняя	256	3	8	48	12, 16
83C51GB	8К ПЗУ	256	3	8	48	12, 16
87C51GB	8К УФПЗУ	256	3	8	48	12, 16
80C152JA	внешняя	256	2	0	40	16, 5
80C152JB	внешняя	256	2	0	56	16, 5
83C152JA	8К ПЗУ	256	2	0	40	16, 5
80C51SL-BG	внешняя	256	2	4	87	16
81C51SL-BG	8К *ПЗУ	256	2	4	87	16
83C51SL-BG	8К ПЗУ	256	2	4	87	16
80C51SLAH	внешняя	256	2	4	87	16
81C51SLAH	8К *ПЗУ	256	2	4	87	16
83C51SLAH	8К ПЗУ	256	2	4	87	16
87C51SLAH	8К УФПЗУ	256	2	4	87	16
80C51SLAL	внешняя	256	2	4	87	16
81C51SLAL	8К *ПЗУ	256	2	4	87	16
83C51SLAL	8К ПЗУ	256	2	4	87	16
87C51SLAL	8К УФПЗУ	256	2	4	87	16

Память программ (байт): *ПЗУ = SystemSoft Standard BIOS

Тактовая частота (МГц): 24i = 24 МГц для внутренних операций

Защита: L1 = 1 Lock Bits; L2 = 2 Lock Bits
L3 = 3 Lock Bits; P = Protection

производимые фирмой Intel

Устройство	Технология производства	Тип корпуса	Защита	Основные особенности
8048AH	HMOS	P	нет	
8035AHL	HMOS	P	—	
8049AH	HMOS	N, P	нет	
8039AHL	HMOS	N, P	—	
8050AH	HMOS	D, N, P	нет	
8040AHL	HMOS	D, P	—	
8748H	HMOS	D, P	нет	
8749H	HMOS	D, N, P	нет	

производимые фирмой Intel

8031AH	HMOS	D, N	—	
8051AH	HMOS	D, N, P	P	
8751H	HMOS	D	L1	
8751BH	HMOS	N, P	L2	
8032AH	HMOS	D, N, P	—	
8052AH	HMOS	D, N, P	нет	
8752BH	HMOS	D, N, P	L2	
80C31BH	CHMOS	D, N, P, S	—	
80C51BH	CHMOS	D, N, P, S	P	
87C51	CHMOS	D, N, P, S	L3	
80C32	CHMOS	N, P, S	—	
80C52	CHMOS	N, P, S	L1	
87C52	CHMOS	D, N, P, S	L3	
80C54	CHMOS	N, P, S	L1	
87C54	CHMOS	D, N, P	L3	
80C58	CHMOS	N, P, S	L1	
87C58	CHMOS	D, N, P, S	L3	
80C51FA	CHMOS	N, P, S	—	Программируемая матрица счетчика (ПМС)
83C51FA	CHMOS	N, P, S	L1	Программируемая матрица счетчика (ПМС)
87C51FA	CHMOS	D, N, P, S	L3	Программируемая матрица счетчика (ПМС)
83C51FB	CHMOS	N, P, S	L1	Программируемая матрица счетчика (ПМС)
87C51FB	CHMOS	D, N, P, S	L3	Программируемая матрица счетчика (ПМС)
83C51FC	CHMOS	N, P, S	L1	Программируемая матрица счетчика (ПМС)
87C51FC	CHMOS	D, N, P, S	L3	Программируемая матрица счетчика (ПМС)
80C51GB	CHMOS	N1	—	8-канальный 8-бит. АЦП, 2 ПМС, 6 портов вв./выв.
83C51GB	CHMOS	N1	L1	8-канальный 8-бит. АЦП, 2 ПМС, 6 портов вв./выв.
87C51GB	CHMOS	N1	L3	8-канальный 8-бит. АЦП, 2 ПМС, 6 портов вв./выв.
80C152JA	CHMOS	P1, N1	—	Многорежимный последовательный порт
80C152JB	CHMOS	N1	—	Многорежимный последовательный порт
83C152JA	CHMOS	P1, N1	нет	Многорежимный последовательный порт
80C51SL-BG	CHMOS	Ku	—	Контроллер клавиатуры
81C51SL-BG	CHMOS	Ku	нет	Контроллер клавиатуры
83C51SL-BG	CHMOS	Ku	нет	Контроллер клавиатуры
80C51SLAH	CHMOS	Ku	—	4-канальный 8-битовый АЦП
81C51SLAH	CHMOS	Ku	нет	4-канальный 8-битовый АЦП
83C51SLAH	CHMOS	Ku	нет	4-канальный 8-битовый АЦП
87C51SLAH	CHMOS	Ku	нет	Контроллер клавиатуры
80C51SLAL	CHMOS	Sb	—	Пониженное напряжение питания
81C51SLAL	CHMOS	Sb	нет	Пониженное напряжение питания
83C51SLAL	CHMOS	Sb	нет	Пониженное напряжение питания
87C51SLAL	CHMOS	Sb	нет	Пониженное напряжение питания

Тип корпуса: D = 40LD CerDIP

P = 40LD PDIP

Ku = 100LD QPF (Quad Flat Pack)

P1 = 48LD PDIP

N = 44L PLCC

S = 44LD QPF (Quad Flat Pack)

N1 = 68LD PLCC

Sb = 100LD SQFP

Приложение 4.
Система команд однокристальных микроЭВМ семейства UPI-42

7-4\3-0	0000(0)	0001(1)	0010(2)	0011(3)	0100(4)	0101(5)	0110(6)	0111(7)	3-0/7-4
0000(0)	NOP	IDL**	OUT DBB,A	ADD A,#data	JMP (page 0)	EN I		DEC A	0000(0)
0001(1)	INC @Rr		JB0 addr	ADDC A,#data	CALL (page 0)	DIS I	JTF addr	INC A	0001(1)
0010(2)	XCH A,@Rr		IN A,DBB	MOV A,#data	JMP (page 1)	EN TCNTI	JNT0 addr	CLR A	0010(2)
0011(3)	XCHD A,@Rr		JB1 addr		CALL (page 1)	DIS TCNTI	JT0 addr	CPL A	0011(3)
0100(4)	ORL A,@Rr		MOV A,T	ORL A,#data	JMP (page 2)	STRT CNT	JNT1 addr	SWAP A	0100(4)
0101(5)	ANL A,@Rr		JB2 addr	ANL A,#data	CALL (page 2)	STRT T	JT1 addr	DA A	0101(5)
0110(6)	ADD A,@Rr		MOV T,A		JMP (page 3)	STOP TCNT		RRC A	0110(6)
0111(7)	ADDC A,@Rr		JB3 addr		CALL (page 3)		JF1 addr	RR A	0111(7)
1000(8)				RET	JMP (page 4)	CLR F0	JOBf addr		1000(8)
1001(9)	MOV STS,A		JB4 addr	RETR	CALL (page 4)	CPL F0	JNZ addr	CLR C	1001(9)
1010(A)	MOV @Rr,A			MOVP A,@A	JMP (page 5)	CLR F1		CPL C	1010(A)
1011(B)	MOV @Rr,#data		JB5 addr	JMPP @A	CALL (page 5)	CPL F1	JF0 addr		1011(B)
1100(C)					JMP (page 6)	SEL RB0	JZ addr	MOV A,PSW	1100(C)
1101(D)	XRL A,@Rr		JB6 addr	XRL A,#data	CALL (page 6)	SEL RB1	JNIBF addr	MOV PSW,A	1101(D)
1110(E)				MOVP3 A,@A	JMP (page 7)	EN DMA*	JNC addr	RL A	1110(E)
1111(F)	MOV A,@Rr		JB7 addr		CALL (page 7)	EN FLAGS	JC addr	RLC A	1111(F)
7-4/3-0	0000(0)		0010(2)	0011(3)	0100(4)	0101(5)	0110(6)	0111(7)	3-0\7-4

* Только для микросхем серии UPI-42

** Только для микросхем ЭКР1847ВГ6

7-4\3-0	1000(8)	1001(9)	1010(A)	1011(B)	1100(C)	1101(D)	1110(E)	1111(F)	3-0/7-4
0000		IN A,PR			MOVD A,PD				0000
0001	INC Rr								0001
0010	XCH A,Rr								0010
0011		OUTL PR,A			MOVD PD,A				0011
0100	ORL A,Rr								0100
0101	ANL A,Rr								0101
0110	ADD A,Rr								0110
0111	ADDC A,Rr								0111
1000		ORL PR,#data			ORLD PD,A				1000
1001		ANL PR,#data			ANLD PD,A				1001
1010	MOV Rr,A								1010
1011	MOV Rr,#data								1011
1100	DEC Rr								1100
1101	XRL A,Rr								1101
1110	DJNZ Rr,addr								1110
1111	MOV A,Rr								1111
7-4/3-0	1000(8)	1001(9)	1010(A)	1011(B)	1100(C)	1101(D)	1110(E)	1111(F)	3-0\7-4

Приложение 5.
Система команд однокристальных микроЭВМ семейства МК48

7-4\3-0	0000(0)	0001(1)	0010(2)	0011(3)	0100(4)	0101(5)	0110(6)	0111(7)	3-0/7-4
0000(0)	NOP	IDL	OUTL BUS,A	ADD A,#DATA	JMP (page 0)	EN I		DEC A	0000(0)
0001(1)	INC @R	JB0 ADDR	ADDC A,#DATA	CALL (page 0)	DIS I	JTF ADDR	INC A		0001(1)
0010(2)	XCH A,@R		MOV A,#DATA	JMP (page 1)	EN TCNTI	JNT0 ADDR	CLR A		0010(2)
0011(3)	XCHD A,@R	JB1 ADDR		CALL (page 1)	DIS TCNTI	JT0 ADDR	CPL A		0011(3)
0100(4)	ORL A,@R	MOV A,T	ORL A,#DATA	JMP (page 2)	STRT CNT	JNT1 ADDR	SWAP A		0100(4)
0101(5)	ANL A,@R	JB2 ADDR	ANL A,#DATA	CALL (page 2)	STRT T	JT1 ADDR	DA A		0101(5)
0110(6)	ADD A,@R	MOV T,A		JMP (page 3)	STOP TCNT		RRC A		0110(6)
0111(7)	ADDC A,@R	JB3 ADDR		CALL (page 3)	ENT0 CLK	JF1 ADDR	RR A		0111(7)
1000(8)	MOVX A,@R		RET	JMP (page 4)	CLR F0	JNI ADDR			1000(8)
1001(9)	MOVX @R,A	JB4 ADDR	RETR	CALL (page 4)	CPL F0	JNZ ADDR	CLR C		1001(9)
1010(A)	MOV @R,A		MOVP A,@A	JMP (page 5)	CLR F1		CPL C		1010(A)
1011(B)	MOV @R,#DATA	JB5 ADDR	JMPP @A	CALL (page 5)	CPL F1	JF0 ADDR			1011(B)
1100(C)				JMP (page 6)	SEL RB0	JZ ADDR	MOV A,PSW		1100(C)
1101(D)	XRL A,@R	JB6 ADDR	XRL A,#DATA	CALL (page 6)	SEL RB1		MOV PSW,A		1101(D)
1110(E)			MOVP3 A,@A	JMP (page 7)	SEL MB0	JNC ADDR	RL A		1110(E)
1111(F)	MOV A,@R	JB7 ADDR		CALL (page 7)	SEL MB1	JC ADDR	RLC A		1111(F)
7-4/3-0	0000(0)		0010(2)	0011(3)	0100(4)	0101(5)	0110(6)	0111(7)	3-0\7-4

7-4\3-0	1000(8)	1001(9)	1010(A)	1011(B)	1100(C)	1101(D)	1110(E)	1111(F)	3-0/7-4
0000(0)	INS A,BUS	IN A,PR			MOVD A,PD				0000(0)
0001(1)	INC REG								0001(1)
0010(2)	XCH A,REG								0010(2)
0011(3)		OUTL PR,A			MOVD PD,A				0011(3)
0100(4)	ORL A,REG								0100(4)
0101(5)	ANL A,REG								0101(5)
0110(6)	ADD A,REG								0110(6)
0111(7)	ADDC A,REG								0111(7)
1000(8)	ORL BUS, #DATA	ORL PR,#DATA			ORLD PD,A				1000(8)
1001(9)	ANL BUS, #DATA	ANL PR,#DATA			ANLD PD,A				1001(9)
1010(A)	MOV REG,A								1010(A)
1011(B)	MOV REG,#DATA								1011(B)
1100(C)	DEC REG								1100(C)
1101(D)	XRL A,REG								1101(D)
1110(E)	DJNZ REG,ADDR								1110(E)
1111(F)	MOV A,REG								1111(F)
7-4/3-0	1000(8)	1001(9)	1010(A)	1011(B)	1100(C)	1101(D)	1110(E)	1111(F)	3-0\7-4

Приложение 6.
Система команд однокристальных микроЭВМ семейства МК51

7-4\3-0	0000(0)	0001(1)	0010(2)	0011(3)	0100(4)	0101(5)	0110(6)	0111(7)	3-0/7-4
0000(0)	NOP	AJMP addr	LJMP addr	RR A	INC A	INC direct	INC @R0	INC @R1	0000(0)
0001(1)	JBC bit,addr	ACALL addr	LCALL addr	RRC A	DEC A	DEC direct	DEC @R0	DEC @R1	0001(1)
0010(2)	JB bit,addr	AJMP addr	RET	RL A	ADD A,#data	ADD A,direct	ADD A,@R0	ADD A,@R1	0010(2)
0011(3)	JNB bit,addr	ACALL addr	RETI	RLC A	ADDC A,#data	ADDC A,direct	ADDC A,@R0	ADDC A,@R1	0011(3)
0100(4)	JC addr	AJMP addr	ORL direct,A	ORL direct, #data	ORL A,#data	ORL A, #direct	ORL A, @R0	ORL A,@R1	0100(4)
0101(5)	JNC addr	ACALL addr	ANL direct,A	ANL direct, #data	ANL A,#data	ANL A, #direct	ANL A,@R0	ANL A,@R1	0101(5)
0110(6)	JZ addr	AJMP addr	XRL direct,A	XRL direct, #data	XRL A,#data	XRL A, #direct	XRL A,@R0	XRL A,@R1	0110(6)
0111(7)	JNZ addr	ACALL addr	ORL C,bit	JMP @A+ DPTR	MOV A,#data	MOV direct, #data	MOV @R0, #data	MOV @R1, #data	0111(7)
1000(8)	SJMP addr	AJMP addr	ANL C,bit	MOVC A, @A+PC	DIV AB	MOV direct, direct	MOV direct, @R0	MOV direct, @R1	1000(8)
1001(9)	MOV DPTR, #data	ACALL addr	MOV bit,C	MOVC A,@A+ DPTR	SUBB A,#data	SUBB A,direct	SUBB A,@R0	SUBB A,@R1	1001(9)
1010(A)	ORL C,/bit	AJMP addr	MOV C,bit	INC DPTR	MUL AB		MOV @R0, direct	MOV @R1, direct	1010(A)
1011(B)	ANL C,/bit	ACALL addr	CPL bit	CPL C	CJNE A, #data, addr	CJNE A, direct, addr	CJNE @R0, #data, addr	CJNE @R1, #data, addr	1011(B)
1100(C)	PUSH direct	AJMP addr	CLR bit	CLR C	SWAP A	XCH A, #direct	XCH A,@R0	XCH A,@R1	1100(C)
1101(D)	POP direct	ACALL addr	SETB bit	SETB C	DA A	DJNZ direct, addr	XCHD A,@R0	XCHD A,@R1	1101(D)
1110(E)	MOVX A, @DPTR	AJMP addr	MOVX A,@R0	MOVX A,@R1	CLR A	MOV A,direct	MOV A,@R0	MOV A,@R1	1110(E)
1111(F)	MOVX @DPTR, A	ACALL addr	MOVX @R0,A	MOVX @R1,A	CPL A	MOV direct,A	MOV @R0,A	MOV @R1,A	1111(F)
	0000(0)	0001(1)	0010(2)	0011(3)	0100(4)	0101(5)	0110(6)	0111(7)	3-0\7-4

7-4\3-0	1000(8)	1001(9)	1010(A)	1011(B)	1100(C)	1101(D)	1110(E)	1111(F)	3-0/7-4
0000(0)	INC R0	INC R1	INC R2	INC R3	INC R4	INC R5	INC R6	INC R7	0000
0001(1)	DEC R0	DEC R1	DEC R2	DEC R3	DEC R4	DEC R5	DEC R6	DEC R7	0001
0010(2)	ADD A,R0	ADD A,R1	ADD A,R2	ADD A,R3	ADD A,R4	ADD A,R5	ADD A,R6	ADD A,R7	0010
0011(3)	ADDC A,R0	ADDC A,R1	ADDC A,R2	ADDC A,R3	ADDC A,R4	ADDC A,R5	ADDC A,R6	ADDC A,R7	0011
0100(4)	ORL A,R0	ORL A,R1	ORL A,R2	ORL A,R3	ORL A,R4	ORL A,R5	ORL A,R6	ORL A,R7	0100
0101(5)	ANL A,R0	ANL A,R0	ANL A,R2	ANL A,R3	ANL A,R4	ANL A,R5	ANL A,R6	ANL A,R7	0101
0110(6)	XRL A,R0	XRL A,R1	XRL A,R2	XRL A,R3	XRL A,R4	XRL A,R5	XRL A,R6	XRL A,R7	0110
0111(7)	MOV R0,#data	MOV R1,#data	MOV R2,#data	MOV R3,#data	MOV R4,#data	MOV R5,#data	MOV R6,#data	MOV R7,#data	0111
1000(8)	MOV direct,R0	MOV direct,R1	MOV direct,R2	MOV direct,R3	MOV direct,R4	MOV direct,R5	MOV direct,R6	MOV direct,R7	1000
1001(9)	SUBB A,R0	SUBB A,R1	SUBB A,R2	SUBB A,R3	SUBB A,R4	SUBB A,R5	SUBB A,R6	SUBB A,R7	1001
1010(A)	MOV R0,direct	MOV R1,direct	MOV R2,direct	MOV R3,direct	MOV R4,direct	MOV R5,direct	MOV R6,direct	MOV R7,direct	1010
1011(B)	CJNE R0, #data, addr	CJNE R1, #data, addr	CJNE R2, #data, addr	CJNE R3, #data, addr	CJNE R4, #data, addr	CJNE R5, #data, addr	CJNE R6, #data, addr	CJNE R7, #data, addr	1011
1100(C)	XCH A,R0	XCH A,R1	XCH A,R2	XCH A,R3	XCH A,R4	XCH A,R5	XCH A,R6	XCH A,R7	1100
1101(D)	DJNZ R0,addr	DJNZ R1,addr	DJNZ R2,addr	DJNZ R3,addr	DJNZ R4,addr	DJNZ R5,addr	DJNZ R6,addr	DJNZ R7,addr	1101
1110(E)	MOV A,R0	MOV A,R1	MOV A,R2	MOV A,R3	MOV A,R4	MOV A,R5	MOV A,R6	MOV A,R7	1110
1111(F)	MOV R0,A	MOV R1,A	MOV R2,A	MOV R3,A	MOV R4,A	MOV R5,A	MOV R6,A	MOV R7,A	1111
7-4/3-0	1000(8)	1001(9)	1010(A)	1011(B)	1100(C)	1101(D)	1110(E)	1111(F)	3-0\7-4

"МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ"

ВНИМАНИЮ
РАЗРАБОТЧИКОВ И ПРОИЗВОДИТЕЛЕЙ
ЦИФРОВЫХ СИСТЕМ!

Предлагаем аппаратуру и программное обеспечение:

- * Схемные эмуляторы:
 - ОЭВМ K1816BE48, 49, 35;
 - K1816BE31, 51;
 - МП K1821BM85;
- * Программатор БИС ПЗУ, ППЗУ, ПЛМ, ОЭВМ
(4 сменных модуля, 3 из них поставляются в виде опций).
- * Система проектирования для ОЭВМ 8096/196.
- * Система проектирования для МП 8086/8088/80186.
- * Интегрированные системы:
 - "Паскаль-51" для ОЭВМ K1816BE51, 31;
 - "Паскаль-85" для МП K1821BM85.
- * Программно-логические модели (отладчики) ОЭВМ
и МП, системы программирования на языке ассемблер.
- * Аппаратно-программная система разработки
контроллеров на базе ОЭВМ K1816BE31 (плата, монитор,
интегрированный ассемблер).
- * Одноплатный контроллер на базе ОЭВМ K1816BE31.
- * Программируемый логический анализатор (32 МГц).

Все приборы работают с IBM PC.
Осуществляется сопровождение новыми версиями ПО,
гарантийное обслуживание.

**Мы разрабатываем и производим микропроцессорные контроллеры
и системы по заказам организаций.
Воспользуйтесь нашим многолетним опытом -
Вы сэкономите время и деньги.**

115409, Москва, Каширское ш., 31, МИФИ,
телефон (095) 323-93-57 (возможно 324-91-55)



Фирма БИНОМ

В 1994 г. предлагает следующие издания:

Однокристалльные микроЭВМ

Справочник

Современные логические КМОП ИС

Справочник

Микропроцессоры и периферийные ИС

Справочник

БИС для телевидения

Справочник

Микропроцессоры ф. Intel

Электронные схемы на ОУ

FLOPPY и HARD диски

Библия пользователя персонального компьютера

Си в примерах

Экономика и бизнес

Таможня-94

Популярная литература

900 способов жить дольше

Как мужчине оставаться молодым

Здоровье для собак и кошек



103473, Москва-473, а/я 133, БИНОМ



Тел. (095) 973-25-65

Факс (095) 973-21-88

Полиграфическая фирма «**Красный пролетарий**»
быстро и качественно выполнит все виды
полиграфических работ.

Заказы принимаются по адресу:
ул. Краснопролетарская, д. 16
телефоны: 258-89-28, 258-89-42

*Боборыкин Алексей Вячеславович
Липовецкий Геннадий Петрович
Литвинский Геннадий Викторович
Оксинь Олег Николаевич
Прохорчик Сергей Владимирович
Проценко Людмила Валентиновна
Петренко Николай Васильевич
Сергеев Александр Анатольевич
Сивобород Павел Владимирович*

Однокристалльные микроЭВМ

Художественное оформление *Р. Бушуева*

Подписано в печать 01.04.94. Формат 70×108 1/16. Печ. л. 25.
Бумага типографская. Гарнитура «Таймс». Печать офсетная.
Тираж 30000 экз. Заказ 4588

Издательство «МИКАП», 1994 г.
Москва, ул. Садово-Каретная, 4/6, стр. 1
Лицензия на издательскую деятельность № 061794 от 12.11.1992 г.

Отпечатано с готовых диапозитивов
в полиграфической фирме «Красный пролетарий»
103473, Москва, Краснопролетарская, 16.

В книге приведены подробные описания восьмиразрядных однокристалльных микроЭВМ семейств МК48, МК51 и МК42 (зарубежные аналоги — однокристалльные микроконтроллеры семейств MCS-48, MCS-51 и UPI-42 фирмы Intel, США).

Основное внимание уделено детальному описанию организации, режимов работы и системы команд. Приведены электрические параметры, временные диаграммы, режимы и условия эксплуатации, а также примеры использования микросхем указанных семейств ОМЭВМ.

Книга является незаменимым пособием для разработки и программирования микропроцессорных систем на базе описанных ОМЭВМ и их зарубежных аналогов.



БИНОМ