Федеральное агентство по образованию
ГОУ ВПО “Уральский государственный технический университет–УПИ”

И.Н. Огородников

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА

Учебник

Издание 2-е, переработанное и дополненное

Научный редактор – доц., канд.-физ.-мат. наук Г.Д. Ведьманов

Екатеринбург
2007

УДК 004.382.7+075.8
ББК 32.973.26-04я73
О39

Рецензенты:
кафедра физики, прикладной математики и информатики Уральского технического института связи и информатики (зав. кафедрой – проф., д-р физ.-мат. наук В.Е. Сидоров);

зав. лаб. Института промышленной экологии УрО РАН, проф., д-р физ.-мат. наук. А.Н. Вараксин

Огородников И.Н.
О39 Микропроцессорная техника: учебник /И.Н. Огородников. 2-е изд., перераб. и доп. Екатеринбург: УГТУ-УПИ, 2007. 380 с.

ISBN 978-5-321-00975-8
ISBN 5-321-00975-9

Учебник по курсу «Микропроцессорная техника» предназначен для студентов физико-технического факультета, обучающихся по специальностям «Электроника и автоматика физических установок», «Радиационная безопасность человека и окружающей среды», «Биомедицинская инженерия» и «Инженерное дело в медико-биологической практике».

Рассмотрены общие вопросы организации однокристальных микропроцессоров и микроконтроллеров, проведен сравнительный анализ микроконтроллеров различных типов, детально рассмотрены организация, функционирование, система команд микроконтроллеров MCS51, а также инструментальные средства подготовки программного обеспечения.

Библиогр.: 28 назв. Табл. 44. Рис. 126. Прил. 3.

ISBN 978-5-321-00975-8
ISBN 5-321-00975-9

УДК 004.382.7+075.8
ББК 32.973.26-04я73

ПРЕДИСЛОВИЕ

В основу предлагаемого учебника положен опыт чтения автором курса лекций по основам микропроцессорной техники студентам физико-технического факультета ГОУ ВПО “Уральский государственный технический университет–УПИ”, специализирующимся в областях автоматики и электроники, инструментальных средств и приборов радиационной безопасности человека и окружающей среды, в области приборов, аппаратов, систем и комплексов для медицинских, биологических и биофизических исследований. Материал учебника предназначен как для начинающих изучать основы микропроцессорной техники, так и для более подготовленных читателей, которым была бы полезна систематизация полученных ранее знаний и практических навыков. Исходя из такой задачи, автор рассмотрел не только основные принципы организации микропроцессорной системы и устройство относительно простых микропроцессоров и микроконтроллеров, но и привел детальное описание конкретных микроконтроллеров в объеме, достаточном для проектирования и практического использования.

В первой части лекционного курса рассматриваются общие вопросы организации микропроцессорных систем.

Во второй части обсуждаются вопросы организации интерфейса и процессов ввода-вывода данных.

В третьей части рассмотрена организация 8- и 16-разрядных однокристальных микропроцессоров (i8080, i8085, i8086/88 и i80186/188).

Четвертая часть посвящена общим вопросам организации однокристальных микроконтроллеров, описанию типовых функциональных узлов, интегрируемых в микроконтроллеры. Обсуждаются критерии и алгоритмы действий при выборе типа микроконтроллера под заданные требования проекта.

В пятой части проведен сравнительный анализ микроконтроллеров различных типов, включая PIC (Microchip), SX (Scenix), AT89 и AVR AT90 (Atmel), Z86E (Zilog), An15 (Ангстрем).

В шестой части детально рассмотрена структурная организация, функционирование и система команд микроконтроллеров MCS51. Данные по MCS51 приведены в объеме, достаточном для последующего использования в лабораторной практике и проектировании.

Микроконтроллерная платформа MCS51 динамично развивается вширь и вглубь. Развитие происходит сразу по нескольким различным направлениям. Седьмая часть посвящена обсуждению наиболее важных направлений этого развития: модификации стандарт-

ного ядра i8051/i8052; микроконтроллерам семейства MCS-251/151; 16-разрядным микроконтроллерам семейства MCS-96, которые *де-факто* являются индустриальным стандартом; микроконтроллерам фирм Philips и Analog Devices. Особое внимание при этом уделено микроконверторам ADuC фирмы Analog Devices, т.к. практическая часть курса (лабораторный практикум и курсовое проектирование) базируется именно на этой элементной базе.

В заключительной восьмой части рассмотрены инструментальные средства разработки программного обеспечения (внутрисхемные эмуляторы, программные симуляторы, оптимизирующий компилятор C51, макроассемблер A51, интегрированные инструментальные среды). На примере макроассемблера фирмы Keil Software дано детальное описание возможностей языка ассемблера ASM-51.

Следует отметить, что микропроцессорная техника представляет собой весьма обширную динамично развивающуюся область технических знаний, что требует постоянного обновления лекционного материала. В книгу включен лишь весьма ограниченный круг вопросов, выбор и глубина освещения которых продиктованы, в первую очередь, требованиями Государственных образовательных стандартов высшего профессионального образования по направлениям 140300 – Ядерные физика и технологии, 200300 – Биомедицинская инженерия и 200400 – Биомедицинская техника. Учебник может быть полезен и для студентов других родственных специальностей.

Для освоения курса микропроцессорной техники необходимо знание основ теории цепей и сигналов, информатики, физики полупроводниковых приборов и цифровой электроники.

При составлении программы и подготовке лекционного курса были широко использованы материалы и данные, представленные в литературных источниках [1–28], которые автор рекомендует студентам для более углубленного изучения материала, для подготовки к зачетам и экзаменам. При работе над учебником были использованы также обширные фактические данные, представленные в Интернете на многочисленных форумах, тематических порталах и сайтах фирм - изготовителей.

1. ОРГАНИЗАЦИЯ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ

1.1. Цифровые устройства с фиксированными и программно-управляемыми функциями

Цифровые устройства как таковые являются предметом изучения отдельной дисциплины “Цифровая электроника”. С точки зрения микропроцессорной техники все схемы цифровой электроники можно разделить на два принципиально различных класса: устройства с фиксированными функциями и устройства с программно-управляемыми функциями.

1.1.1. Цифровые устройства с фиксированными функциями

Цифровые устройства с фиксированными функциями реализуют заданный набор функций, который не изменяется в процессе работы устройства. Все многообразие схем с фиксированными функциями можно формально свести, в свою очередь, к двум видам: *комбинационные* и *последовательностные* схемы. В дальнейшем комбинационные схемы будем обозначать КЛ (комбинационная логика), а последовательностные схемы – ПС. Различия между КЛ и ПС имеют фундаментальный характер.

Комбинационные схемы. Выходные величины схем КЛ зависят только от текущего значения входных величин (аргументов). Предыстория значения не имеет.

В задачу синтеза комбинационных схем входит построение схемы устройства по заданным условиям его работы и при заданном базисе элементов. Задание комбинационного устройства сводится к заданию тех функций, которые оно должно реализовать. Число функций определяется числом выходов комбинационного устройства. Проектирование комбинационных схем состоит из этапов абстрактного и схемного синтеза.

Абстрактный синтез включает в себя: формирование задачи (словесное описание функций устройства, определение типа устройства), описание устройства на формализованных языках (таблица истинности, карта Карно, аналитическое выражение и т.п.), минимизацию булевых функций; построение логической схемы устройства.

Схемный синтез включает в себя: переход в требуемый базис, построение принципиальной схемы, разработку монтажной схемы, изготовление устройства и его испытания.

На рис. 1.1 показаны различные обозначения, применяемые для изображения КЛ на функциональных схемах. Здесь $x_0, x_1, \ldots, x_{n-1}$ обозначают электрические линии, по каждой из которых передается зна-

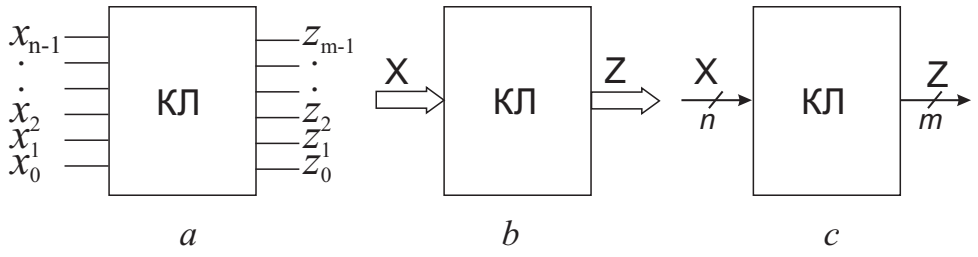


Рис. 1.1. Изображение схем КЛ: *a* – подробное; *b* – упрощенное; *c* – стилизованное

чение одного двоичного разряда (бита) входных данных. Для упрощения изображения *n*-разрядные входные данные обозначаются как X. Выходные *m*-разрядные данные $z_0, z_1, \ldots, z_{m-1}$ имеют аналогичные обозначения: $Z=\{z_0, z_2, \ldots, z_{n-1}\}$. Между входными и выходными данными существует взаимно однозначная связь $Z = F(X)$. Для указания размерности двоичных данных используют общепринятые обозначения:

> *бит* (bit = **b**inary dig**it**) – один разряд двоичного числа. Может принимать одно из двух возможных логических значений: 0 или 1 (“ложь” или “истина”);
>
> *тетрада* (nibble) – 4 двоичных разряда (4 бита). Может принимать одно из 16 возможных значений;
>
> *байт* (byte) – 8 двоичных разрядов (8 битов). Может принимать одно из 256 возможных значений;
>
> *слово* (word) – *n* двоичных разрядов (*n* битов). Может принимать одно из 2^n возможных значений.

Последовательностные схемы. В отличие от схем КЛ, выходные величины схем ПС зависят не только от того, какие сигналы присутствуют на его входах в данный момент времени, но и от того, какие последовательности сигналов поступали на входы устройства в предшествующие моменты времени, т.е. ПС помнит свою предысторию и хранит ее в своей памяти. Наличие памяти принципиально отличает ПС от схем КЛ. Для описания ПС, помимо состояний входов X и выходов Z, необходимо также знать состояние памяти (внутреннее состояние) Y.

В общем виде ПС может быть представлена в виде трех частей: схемы возбуждения **Ф**, памяти **Y/Y′** и схемы выходов **S** (рис. 1.2). При этом схемы возбуждения и выходов являются комбинационными ло-

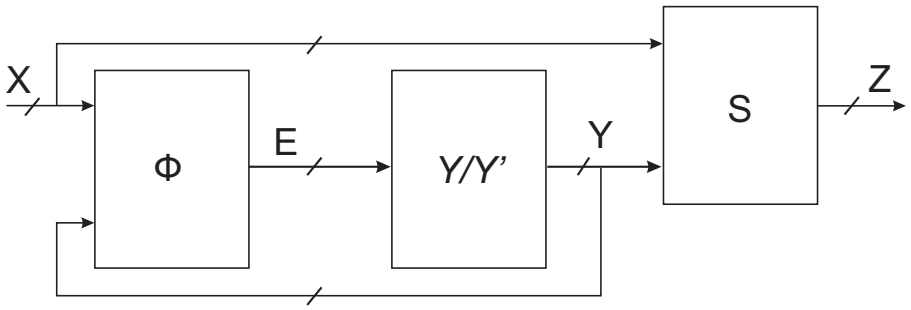


Рис. 1.2. Последовательностная схема

гическими схемами с функциями $E = \Phi(X,Y)$ и $Z = S(X,Y)$, которые можно задать обычным образом: таблицами истинности, картами Карно, аналитическими выражениями.

Для описания функционирования памяти необходимы правила, регламентирующие последовательность переключения внутренних состояний Y и последовательность выходных сигналов в зависимости от последовательности поступления входных сигналов, т.е. от предыдущих состояний Y'. Закон функционирования памяти может задаваться в виде графов, таблиц переключений или уравнений.

На рис. 1.3 приведен пример некоторой последовательностной схемы с четырьмя внутренними состояниями Y, обозначенными цифрами от 1 до 4. Переходы между этими состояниями происходят под действием входных сигналов A, B, C, D. Графы переходов дают полное описание закона функционирования данной последовательностной схемы.

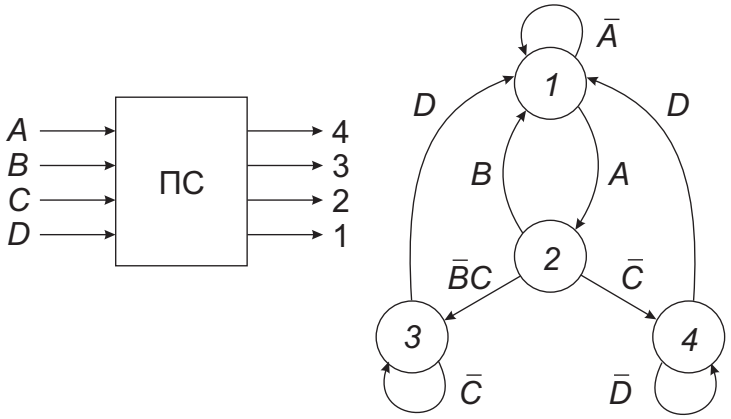


Рис. 1.3. Графы переходов последовательностной схемы

Таблица переключений (табл. 1.1) иллюстрирует альтернативный способ описания той же самой последовательностной схемы. В левой

Таблица 1.1. Таблица переключений последовательностной схемы

Состояние	Сигналы				
	A	B	$\overline{B}C$	$\overline{C}$	D
1	2				
2		1	3	4	
3					1
4					1

колонке указаны исходные состояния до перехода, в заголовках столбцов перечислены поступившие сигналы, а в самой таблице указаны состояния системы после перехода под действием данных сигналов. Сопоставление рис. 1.3 и табл. 1.1 позволяет убедиться в идентичности двух способов описания одной и той же последовательностной схемы.

Недостатки схем с фиксированными функциями. Несмотря на фундаментальные различия, схемы КЛ и ПС принадлежат к одному и тому же классу устройств с фиксированными логическими функциями. При масштабировании (усложнении функций) таких устройств возникают общие проблемы обеспечения надежности устройства, снижения его габаритов, ограничения тока потребления. Наиболее кардинальное решение этих проблем – переход на интегральную технологию. При этом, однако, возникают другие проблемы, связанные со стоимостью проекта, временем разработки технологии, изготовления шаблонов и т.п., временем и полнотой тестирования интегральных схем (ИС). В итоге разработка заказных больших интегральных схем (БИС) стóит весьма дорого и окупается лишь для крупных серий.

Для малотиражных изделий в качестве частичных решений используют универсальные БИС (насколько это возможно для устройств с фиксированными функциями) либо сложные программируемые логические интегральные схемы (СПЛИС).

1.1.2. Цифровые устройства с программно-управляемыми функциями

Альтернативным классом цифровых устройств являются устройства с программно-управляемыми функциями. Такие устройства способны выполнять некоторое количество различных функций. В каждый момент времени устройство может быть настроено на выполне-

ние какой-то одной функции. В следующий момент устройство можно настроить на выполнение другой функции. Комбинируя по очереди соответствующие настройки, можно с помощью одного такого устройства выполнить весь заданный алгоритм обработки информации. Устройства с программно-управляемыми функциями выпускают в виде стандартных универсальных БИС. Системы, построенные на их основе, удовлетворяют двум принципам: принципу программного управления и принципу магистрально-модульной организации связей.

Принцип программного управления включает в себя следующие основные положения:

- любая функция, реализуемая устройством, является последовательностью элементарных действий – операций. Каждая операция задается специальной инструкцией или командой, служащей для настройки цифрового устройства на выполнение заданного элементарного действия;
- процесс реализации функции в устройстве описывается в форме алгоритма, называемого программой. Программа представляется в терминах команд и логических условий. Логические условия используются для управления порядком следования команд;
- программа используется как форма представления функции устройства, на основе которой определяются структура и порядок функционирования устройства во времени;
- программа предварительно размещается в памяти устройства, а не вводится команда за командой в процессе его работы.

Пример устройства с программно-управляемыми функциями представлен на рис. 1.4. Программа размещена в памяти команд (ПК). После размещения программы в памяти устройство управления (УУ)

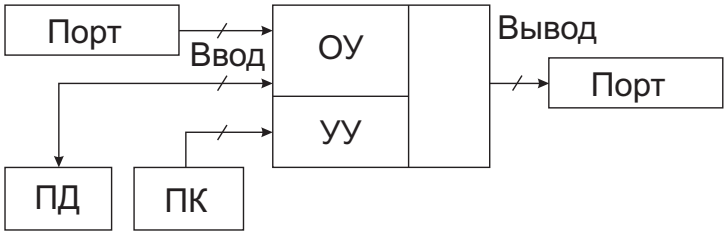


Рис. 1.4. Устройство с программно-управляемыми функциями

начинает циклически выполнять три действия: 1) выборку команды из памяти команд; 2) декодирование (интерпретацию) кода команды; 3) выполнение операции, соответствующей данной команде. Для выполнения операции УУ настраивает соответствующим образом устройство обработки (ОУ). В состав ОУ входят арифметико-логическое

устройство (АЛУ) и ряд вспомогательных функциональных узлов. Входные данные для обработки поступают от периферийного устройства через порт ввода, а результаты обработки выдаются внешнему устройству через порт вывода. Память данных (ПД) служит для временного хранения входных и промежуточных данных.

Команда или инструкция (Command, Instruction) – двоичный код, служащий для настройки программно-управляемого устройства на выполнение заданной операции.

Система команд (Command set) – совокупность всех команд, допустимых для данного программно-управляемого устройства.

Программа (Program) – последовательность инструкций (команд) и логических условий, реализующих заданный алгоритм.

Арифметико-логическое устройство (ALU) – цифровое устройство с программным управлением, выполняющее арифметические и логические операции.

Порт (Port) – средство для подключения периферийных устройств.

Магистраль или шина (Bus) – группа линий передачи информации, объединенных общей функцией.

Команды из памяти выбираются одна за другой по порядку. Наличие логических условий может изменять порядок выборки команд. Цикл “выборка–декодирование–выполнение” повторяется все время, пока устройство функционирует.

1.1.3. Магистрально-модульная организация связей

Магистрально-модульный принцип организации связей обеспечивает обмен информацией между функциональными и конструктивными модулями различного уровня с помощью магистралей, объединяющих входные и выходные шины.

Необходимость магистрально-модульной организации связей обусловлена требованиями их оптимизации, повышения компактности, формализации топологии для автоматизированного проектирования и тестирования. Немаловажным фактором является возможность расширения системы (модификации) за счет подключения новых дополнительных модулей.

Принцип магистрально-модульной организации связей состоит в регуляризации и упорядочении связей между функциональными узлами - модулями посредством перехода от межмодульных связей к обмену данными через общую шину - магистраль. Регуляризация предполагает закономерную повторяемость элементов структуры и связей между ними.

Процедура подключения модулей к магистрали является типовой для всех модулей и поддается формальному описанию и стандартизации. Это открывает совершенно новые возможности для разработчиков: любая новая аппаратура, разработанная с соблюдением соответствующего стандарта, может быть легко подключена к системе.

Усложнение организационной структуры связей неизбежно ведет к некоторому замедлению процессов обмена данными. Однако в большинстве случаев выигрыш от регуляризации связей существенно превышает проигрыш от потери быстродействия. В особых случаях, когда решающим фактором является именно скорость обмена данными, руководствуются другими принципами.

На рис. 1.5 приведен пример системы с магистрально-модульной организацией связей. Общая шина (ОШ), или магистраль, группирует линии связей по функциональному признаку. В системе может быть несколько общих шин, различающихся по функциональному назначе-

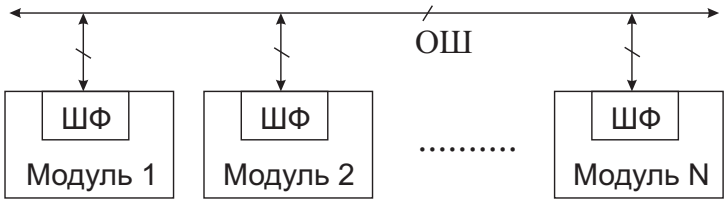


Рис. 1.5. Магистрально-модульная организация связей

нию. Шинные формирователи (ШФ) входят в состав каждого модуля и служат для управления обменом данных по магистрали. Функции шинного формирователя: 1) передача данных (от модуля к магистрали); 2) прием данных (от магистрали к модулю); 3) отключение модуля от магистрали при отсутствии обмена данными.

Схемотехника шинных формирователей. Схемотехника шинных формирователей может базироваться на различных принципах. Рассмотрим построение трех типов шинных формирователей.

1. Шинный формирователь на основе схем мультиплексор-демультиплексор (рис. 1.6). В этом примере любой из пяти модулей, подключенных к мультиплексору (MUX), может быть выбран по адресу и подключен к общей линии связи. Демультиплексор (DEMUX) выполняет обратную функцию – позволяет подключить общую шину к

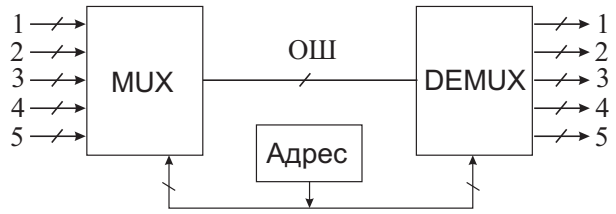


Рис. 1.6. ШФ на основе схем мультиплексор-демультиплексор

любому модулю, выбранному по адресу. Основные недостатки этой схемы: число выводов электронных переключателей конструктивно ограничено, поэтому а) подключение новых модулей может быть сопряжено с большими техническими трудностями; б) не удается реализовать отключение от общей шины при отсутствии обмена данными. На практике такая схема коммутации каналов применяется лишь для уменьшения количества контактов в разъемах.

2. Шинный формирователь на основе схем с открытым коллектором (рис. 1.7). Такие схемы допускают объединение на общей нагрузке ("монтажное ИЛИ"). В исходном состоянии все транзисторные ключи

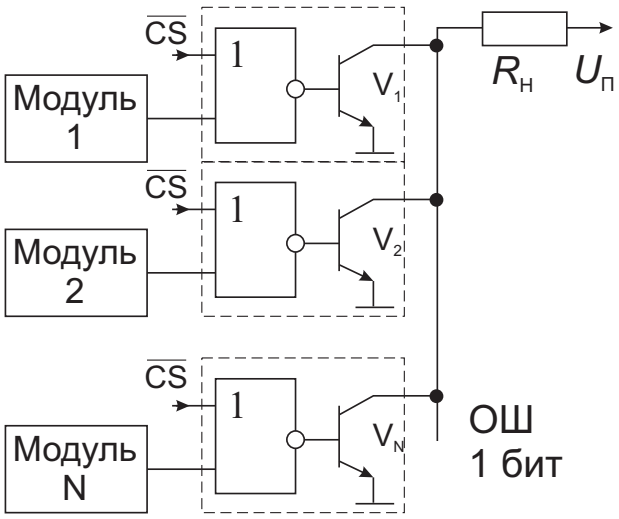


Рис. 1.7. ШФ на основе схем с открытым коллектором

V_i закрыты, линия ОШ нагружена лишь их внутренними сопротивлениями r_i. Это соответствует единичному логическому уровню. Любой открытый ключ устанавливает на линии ОШ состояние логического нуля. Таким образом, все модули могут по очереди выдавать данные на линию. Разрешение выхода на линию осуществляется сигналом $\overline{CS}$=0 (CS = **C**hip **S**elect). Основной недостаток данной схемы в том, что число подключаемых устройств сильно ограничено. Действительно, суммарное сопротивление всех ключей, соединенных по параллельной схеме, будет r, а емкость – C (1.1). Легко видеть, что при возрастании числа модулей N соотношение $r/R_н$ будет быстро

падать до неприемлемо малого уровня, а емкость C будет быстро возрастать. Предельно-допустимые значения этих параметров обусловливают ограничение по количеству устройств (N), подключаемых к общей шине.

$$r = \left(\sum_{i=1}^{N} r_i^{-1} \right)^{-1}; \qquad C = \sum_{i=1}^{N} C_i. \tag{1.1}$$

На практике данное схемотехническое решение применяется, главным образом, для организации шинных формирователей внутри кристалла микросхемы.

3. Шинный формирователь на основе логических элементов с тремя состояниями. Элемент с тремя состояниями (рис. 1.8) может

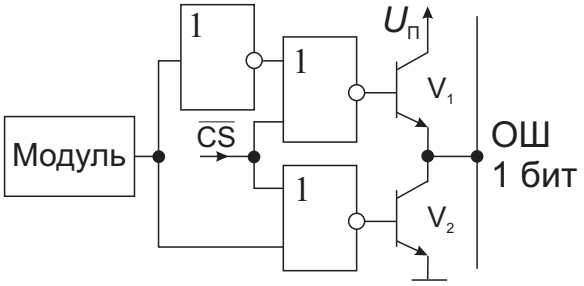


Рис. 1.8. Элемент с тремя состояниями на биполярных транзисторах

находиться в одном из трех состояний. При $\overline{CS}$=0 логические уровни нуля или единицы с выхода модуля передаются на линию ОШ. При $\overline{CS}$=1 оба ключа закрыты и линия ОШ отключена от модуля. Это третье состояние элемента, в котором он имеет высокое внутреннее сопротивление. В этом состоянии элемента потенциал на линии ОШ становится плавающим, т.е. он определяется не данным элементом, а какими-то другими устройствами, подключенными к линии.

Замена биполярных транзисторов на комплементарные МОП-транзисторы (КМОП-структуры) позволяет избавится от дополнительных логических вентилей и оптимизировать схемотехнику элемента с тремя состояниями. Логический элемент с тремя состояниями на КМОП-структурах (рис. 1.9) применяется в серийных шинных формирователях для межкристальных и межблочных соединений. Его конструкция отличается высокой нагрузочной способностью. Высокое входное сопротивление изолированных затворов входных ключей позволяет произвольно комбинировать такие элементы, подключая, например, вход одного к выходу другого для обеспечения двунаправленного обмена. На рис. 1.10 приведен фрагмент функциональной схемы серийного неинвертирующего шинного формирователя. Линии общей шины обозначены DB (**D**ata **B**inary, DBIN), линии, по которым информация вводится на шину, - DI (**D**ata **I**nput), линии вывода с общей шины

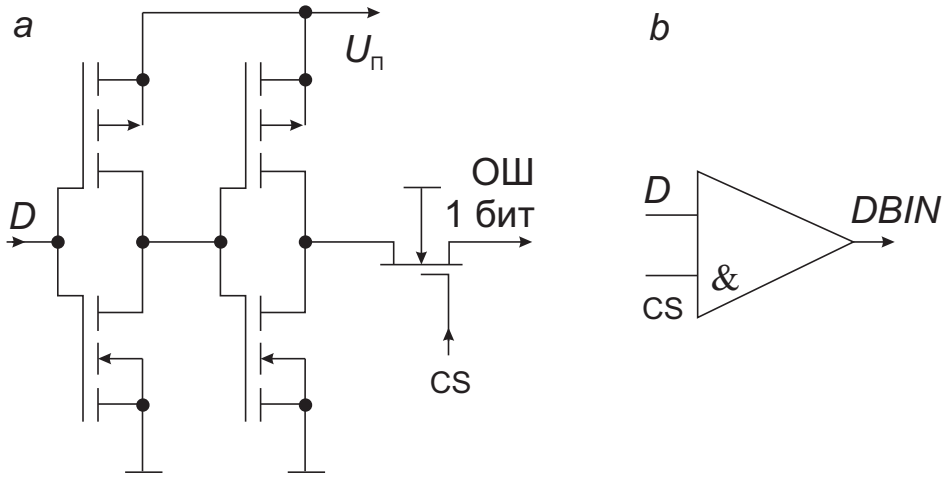


Рис. 1.9. Неинвертирующий элемент с тремя состояниями на КМОП-структурах (*a*) и его условное обозначение (*b*)

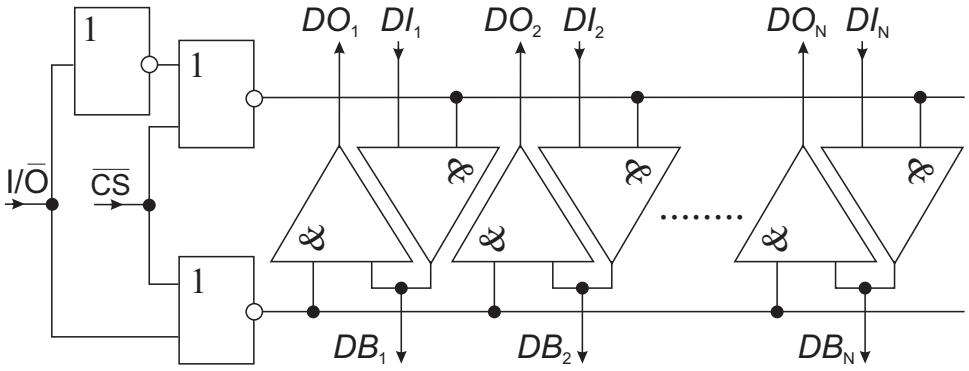


Рис. 1.10. Структура серийного шинного формирователя

в модуль - DO (**D**ata **O**utput). Направления потоков данных в системе отсчитывают от центрального модуля, обычно это модуль микропроцессора. Сигнал I/$\overline{O}$ служит для переключения направления потока данных: ввод или вывод.

Функциональная схема микропроцессорной системы. На рис. 1.11 показана функциональная схема микропроцессорной системы, удовлетворяющая сразу обоим принципам: программного управления и магистрально-модульной организации. Именно эта базовая система будет являться предметом нашего дальнейшего анализа. Система состоит из тех же самых функциональных узлов, что и ее прототип, приведенный на рис. 1.4. В отличие от прототипа (рис. 1.4), все потоки данных здесь сгруппированы в три общие шины (магистрали), различающиеся по функциональному назначению. Шина управления (ШУ) служит для передачи всех управляющих сигналов, таких как $\overline{CS}$, I/$\overline{O}$ и т.п. Шина адреса (ША) предназначена для передачи адресов каких-либо элементов системы (например, адресов ячеек памяти и т.п.) при

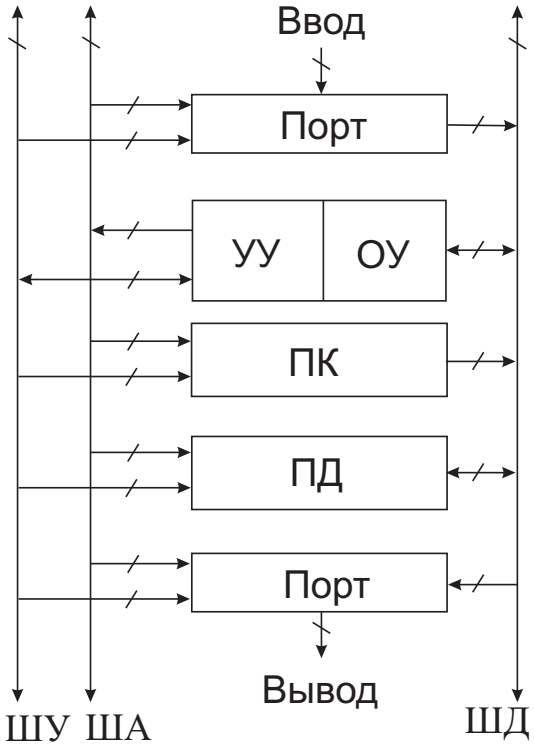


Рис. 1.11. Функциональная схема микропроцессорной системы

обращении к ним. Шина данных (ШД) служит для пересылки данных между устройствами системы. Шинные формирователи входят в состав каждого функционального узла, но на рис. 1.11 они не показаны.

Система также циклически выполняет три типовых действия: выборку команды из памяти команд, декодирование кода операции (КОП) и выполнение операции. Однако детали этих действий несколько отличаются от прототипа, поскольку потоки данных распределены по разным шинам.

Описание работы системы выглядит следующим образом:

1) устройство управления выставляет на ША адрес ячейки ПК, где содержится команда для выборки;

2) устройство управления выставляет на ШУ сигналы для доступа к памяти команд ($\overline{CS}$, $I/\overline{O}$ и т.п.);

3) память команд выдает на ШД код операции, содержащийся в адресуемой ячейке;

4) устройство управления принимает КОП с ШД и размещает его в своем внутреннем регистре команд;

5) устройство управления декодирует КОП, размещенный в регистре команд, и настраивает обрабатывающее устройство на выполнение заданной операции;

6) обрабатывающее устройство выполняет заданную операцию;

7) все действия, начиная с п.1, циклически повторяются.

Данные операции циклически повторяются, пока не будет выпол-

нена вся программа, хранящаяся в памяти программ. Приведенный на рис. 1.11 вариант организации микропроцессорной системы не является единственным. Существует множество разновидностей или способов построения системы. Для обозначения различных видов системы используют термин “архитектура” системы.

1.2. Архитектура системы

Архитектура – логическая организация микропроцессора или микроконтроллера, рассматриваемая с точки зрения пользователя; она определяет возможности микропроцессора по аппаратной и программной реализации функций, необходимых для построения микропроцессорной системы.

Понятие архитектуры отражает:

1) концептуальную структуру, т.е. совокупность компонентов, составляющих микропроцессор или микроконтроллер, и связей между ними; для пользователя достаточно ограничиться регистровой моделью микропроцессора;

2) способы представления и форматы данных;

3) способы обращения ко всем программно-доступным для пользователя элементам структуры (адресация к регистрам, ячейкам постоянной и оперативной памяти, внешним устройствам);

4) набор операций, выполняемых микропроцессором;

5) характеристики управляющих слов и сигналов, вырабатываемых микропроцессором и поступающих в него извне;

6) реакцию на внешние сигналы (система обработки прерываний и т.п.).

Понятие архитектуры не включает в себя такие проблемы, как передача потоков данных внутри процессора, конструктивные особенности логических схем и специфику технологии производства.

Архитектуру можно, в известном смысле, понимать как некоторую модель системы, которая на заданном уровне детализации описывает актуальные характеристики системы. В рамках архитектурной модели становится возможным сравнивать и оценивать требуемые характеристики систем, отбрасывая все остальные несущественные для данного сравнения особенности организации системы.

1.2.1. Некоторые типы архитектуры

Число различных типов архитектуры исчисляется сотнями. Мы рассмотрим лишь некоторые из них, имеющие отношение к обсуждаемым далее типам микропроцессоров и микроконтроллеров.

По способу организации пространства памяти микропроцессорной системы различают два основных типа архитектур: архитектуру фон Неймана и гарвардскую архитектуру (рис. 1.12).

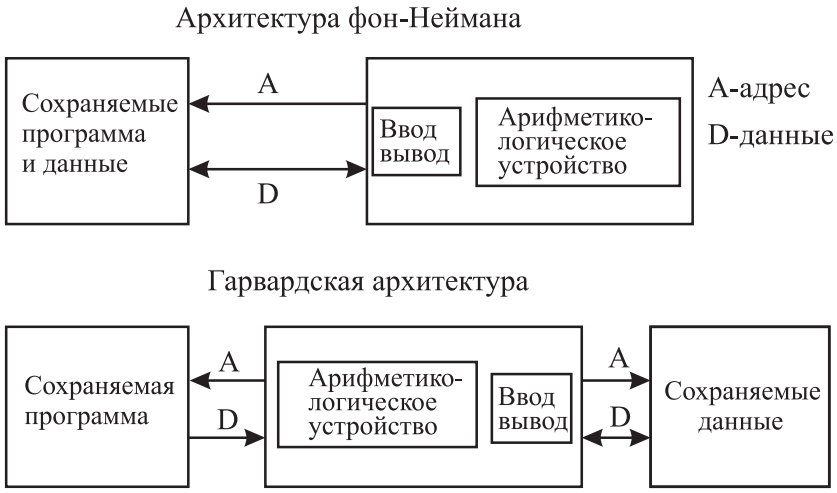


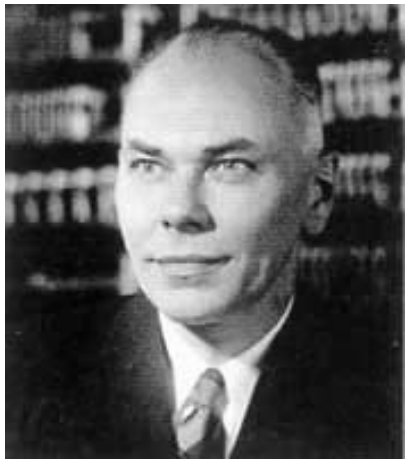
Рис. 1.12. Архитектура фон Неймана и гарвардская архитектура

> *Архитектура фон Неймана* (принстонская архитектура) – организация пространства памяти, при которой для хранения программ и данных используется одно пространство памяти.

Название архитектуры дано по имени математика Джона фон Неймана (John von Neumann), предложившего в 40-е годы XX века кодировать программы в формате, соответствующем формату данных. В основу архитектуры положены принципы фон Неймана:

- принцип программного управления, согласно которому программа состоит из набора команд, выполняемых процессором друг за другом в определенной последовательности;
- принцип однородности памяти, согласно которому программы и данные хранятся в одной и той же памяти;
- принцип адресности, согласно которому основная память состоит из нумерованных ячеек и процессору в любой момент времени доступна любая ячейка.

Отличительные черты архитектуры фон Неймана: единственная последовательно адресуемая память; программа и данные хранятся в одной памяти, адреса областей которой составляют последовательность; память является линейной и одномерной; отсутствует явное различие между командами и данными. Их идентифицируют неявным способом при выполнении операций; назначение данных не является их неотъемлемой составной частью. Оно определяется логикой программы.



Говард Айхен (1900-1973)

Джон фон Нейман (1903-1957)

Преимуществами архитектуры фон Неймана являются более простая внутренняя структура микропроцессора и меньшее количество управляющих сигналов: одно арифметико-логическое устройство, через которое проходит поток данных, и одно устройство управления, через которое проходит поток команд.

> *Гарвардская архитектура* – организация пространства памяти, при которой память программ и память данных физически и логически разделены и имеют свои собственные адресные пространства и способы доступа к ним.

Гарвардская архитектура была разработана Говардом Айхеном (Howard Aiken) в конце 1930-х годов в Гарвардском университете (отсюда название).

Основное отличие гарвардской архитектуры состоит в том, что память программ и память данных разделены, они используют физически разделенные линии передачи и требуют дополнительных управляющих сигналов. Это позволяет микропроцессору или микроконтроллеру пересылать команды и данные одновременно.

Гарвардская архитектура является более сложной, однако она позволяет более гибко манипулировать информацией, реализовывать компактно кодируемый набор машинных команд и, в ряде случаев, ускорять работу микропроцессора. Представителями такой архитектуры являются микроконтроллеры семейства MCS51 фирмы Intel.

В настоящее время выпускаются микропроцессоры со смешанной архитектурой, в которых память программ и память данных имеют единое адресное пространство, однако различные механизмы доступа.

На рис. 1.13 приведена схема классификации типов архитектуры по системе команд и особенностям выполнения операций. Поскольку англоязычные аббревиатуры (и некоторые названия) очень широко

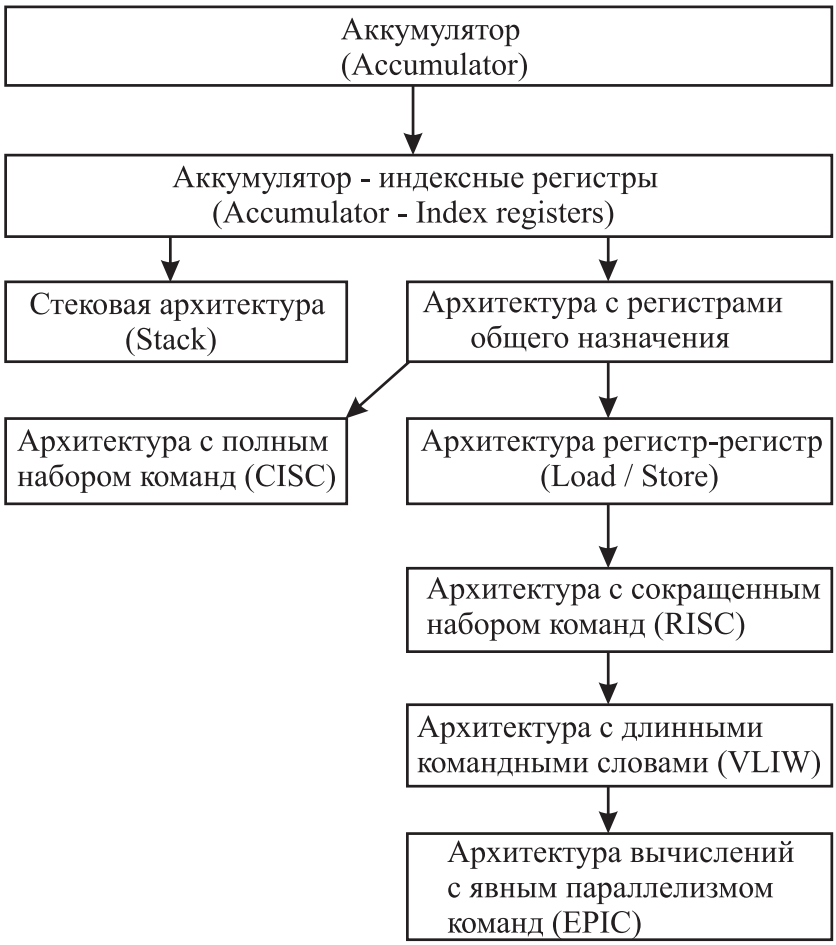


Рис. 1.13. Классификация типов архитектуры по системе команд

распространены в литературе, мы специально приведем их рядом с русскоязычными названиями. Архитектуры набора команд на этой схеме обозначают некоторые идеализированные концепции. Реальные микропроцессоры и микроконтроллеры иногда бывает достаточно сложно отнести к одной из них.

Архитектура аккумулятор (рис. 1.14) очень проста и наиболее близка к архитектуре простейшего калькулятора. Процессор имеет единственный регистр – аккумулятор, давший название архитектуре. Его содержимое комбинируется в арифметико-логическом устройстве процессора с единственным операндом. Напомним, что операндом называются данные для обработки - то, над чем выполняется операция. Результат операции также помещается в аккумулятор. Существует две команды для загрузки значения из памяти в аккумулятор и выгрузки его из аккумулятора в память. Данная архитектура использует адрес основной памяти в качестве единственного операнда команды. Ссылка на аккумулятор производится неявно с помощью кода операции. При этом нет необходимости в коде команды выделять специальную область для адресов операнда и результата.

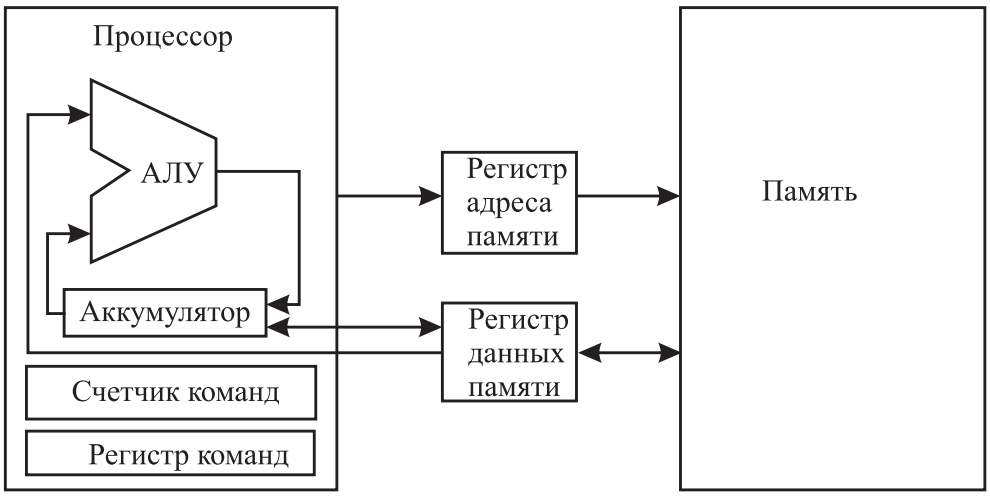


Рис. 1.14. Архитектура аккумулятор

К недостаткам архитектуры *аккумулятор* можно отнести относительно низкое быстродействие, объясняемое тем, что аккумулятор является *узким местом*, в которое каждый раз необходимо сначала занести операнд перед выполнением операции.

Стековая архитектура. Арифметические команды стековой архитектуры не имеют операндов (рис. 1.15). Стековая организация дает возможность использовать безадресные команды, код которых имеет наименьшую длину. Безадресные команды оперируют данными, находящимися на вершине стека и непосредственно под ней. При выполнении операции исходные операнды извлекаются из стека, а результат передается на вершину стека. Существует две команды для загрузки значения из памяти на вершину стека и выгрузки значения на вершине стека в память. Эти команды используют адрес основной памяти в качестве единственного операнда команды.

Стековая архитектура обладает высокой вычислительной эффективностью. Существует специальный язык высокого уровня FORTH, построенный на основе безадресных команд. Такая архитектура используется в специализированных процессорах высокой производительности и, в частности, в RISC-процессорах.

Архитектура регистр-регистр. Архитектура регистр-регистр характеризуется наличием высокоскоростной регистровой памяти, расположенной на одном кристалле с процессором (рис. 1.16). Эта архитектура близка к стековой, но операнды могут быть взяты из любого регистра высокоскоростной памяти. В отличие от аккумулятора рабочие регистры адресуются явно в коде команды. С помощью специальных команд можно производить обмен данными между регистровым

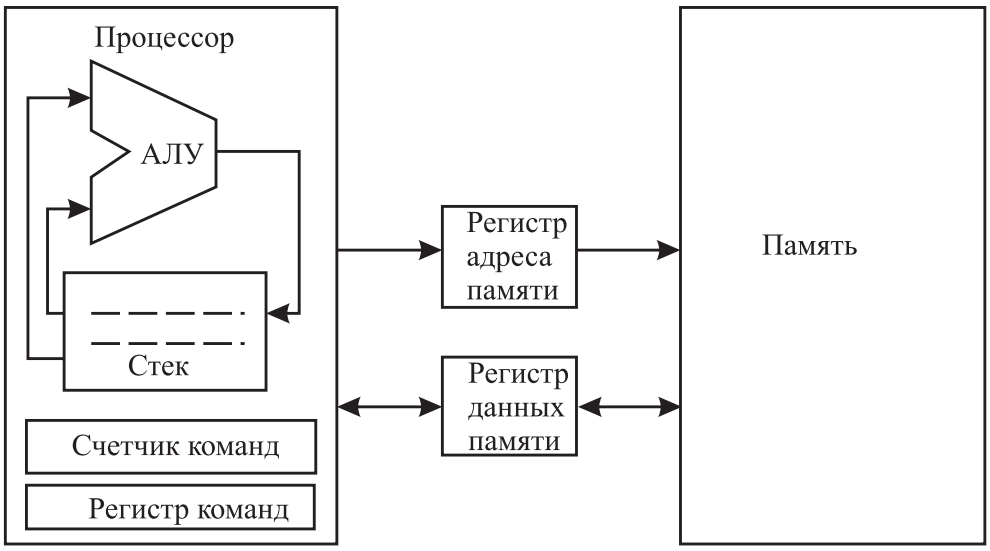


Рис. 1.15. Стековая архитектура

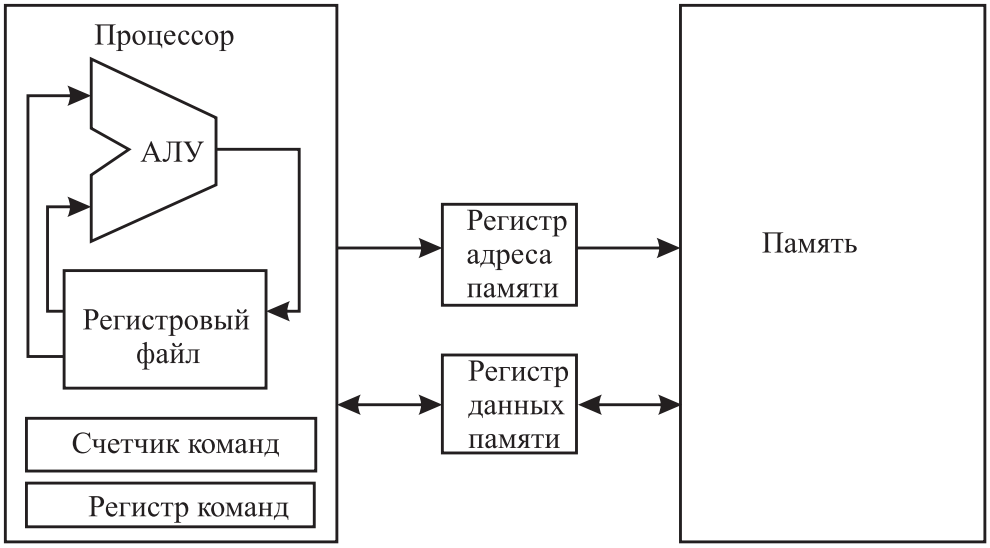


Рис. 1.16. Архитектура регистр-регистр

файлом (регистровой памятью) и основной памятью. Арифметические и логические команды в качестве операндов используют три регистра: два из них являются источниками, а в третий помещается результат.

Архитектуры типа CISC, RISC и VLIW обозначают некоторые идеализированные концепции. Большинство широко известных реальных процессоров занимают промежуточное положение на шкале, левый край которой может быть помечен концепцией CISC, а правый край шкалы – RISC.

Полный набор команд. Complete Instruction Set Computer (CISC). Для CISC-процессоров характерно сравнительно небольшое число регистров общего назначения, большое количество машинных команд,

некоторые из них нагружены семантически аналогично операторам высокоуровневых языков программирования и выполняются за много тактов, большое количество методов адресации и форматов команд различной разрядности, преобладание двухадресного формата команд, наличие команд обработки типа регистр-память.

Сокращенный набор команд. Reduced Instruction Set Computer (RISC). Данная архитектура основана на результатах статистического анализа частоты востребованности различных команд микропроцессора. Оказалось, что даже у процессоров со сложной организацией большая часть времени уходит на выполнение простых команд. Имеет место так называемая теорема 20/80, а именно: 20 % команд используется в 80 % случаев, а оставшиеся 80 % команд используются в 20 % случаев. Это наблюдение легло в основу работ по созданию IBM801 – первой RISC-машины, разработка которой была завершена к 1979 году.

Само понятие RISC было введено Дэвидом Паттерсоном (David Patterson), преподавателем университета Беркли, в 1980 году.

Основными чертами концепции RISC-архитектуры являются:

- одинаковая длина команд;
- единый формат команд или, по крайней мере, использование не более двух-трех форматов;
- регистры, являющиеся операндами всех арифметических и логических команд;
- команды, выполняющие только простые действия;
- выполнение любой команды не дольше чем за один такт;
- большой регистровый файл;
- только простая адресация.

В системах программирования для RISC-архитектуры практически всегда присутствуют компиляторы, имеющие большие возможности по оптимизации кода.

Архитектура с длинным командным словом. Архитектура с длинным командным словом – это статическая суперскалярная архитектура. Несколько простых команд упаковывается компилятором в длинное слово. Слово соответствует набору функциональных устройств. Распараллеливание кода производится на этапе компиляции, и в машинном коде уже присутствует явный параллелизм.

Архитектура вычислений с явным параллелизмом команд. Explicitly Parallel Instruction Computing (EPIC) является развитием архитектуры с длинным командным словом. Концепция EPIC разработана совместно компаниями Intel и Hewlett-Packard. Она обладает достоинствами

VLIW, но лишена ее недостатков (например, использует специальные механизмы для исключения неэффективности кодов традиционных VLIW-архитектур, требовавших применения пустых команд для заполнения пустых машинных тактов).

1.3. Полупроводниковые запоминающие устройства

Запоминающие устройства (ЗУ) служат для хранения информации и обмена ею с другими цифровыми устройствами. Микросхемы памяти в общем объеме выпуска интегральных схем занимают до 40 % и играют важнейшую роль во многих системах различного назначения. Микросхемы и системы памяти постоянно совершенствуются как в области схемотехники, так и в области развития новых архитектур. В настоящее время созданы и используются десятки различных типов ЗУ.

1.3.1. Система параметров

Важнейшие параметры ЗУ находятся в противоречии. Так, например, большая информационная емкость не сочетается с высоким быстродействием, а быстродействие в свою очередь не сочетается с низкой стоимостью. Поэтому системам памяти свойственна многоступенчатая иерархическая структура, и в зависимости от роли того или иного ЗУ его реализация может быть существенно различной. В наиболее развитой иерархии памяти можно выделить следующие уровни:

- *регистровые ЗУ*, находящиеся в составе процессора или других устройств (т.е. внутренние для этих блоков), благодаря которым уменьшается число обращений к другим уровням памяти, реализованным вне процессора и требующим большего времени для операций обмена информацией;
- *кэш-память*, служащая для хранения копий информации, используемой в текущих операциях обмена. Высокое быстродействие кэш-памяти повышает производительность микропроцессоров;
- *основная память* (оперативная, постоянная, полупостоянная), работающая в режиме непосредственного обмена с процессором и по возможности согласованная с ним по быстродействию. Исполняемый в текущий момент фрагмент программы обязательно находится в основной памяти;
- *специализированные* виды памяти, характерные для некоторых специфических архитектур (многопортовые, ассоциативные, видеопамять и др.);
- *внешняя память*, хранящая большие объемы информации. Эта память обычно реализуется на основе устройств с подвижным но-

сителем информации (магнитные и оптические диски, магнитные ленты и др.). В настоящем пособии устройства внешней памяти не рассматриваются.

Основные параметры ЗУ

Информационная емкость – максимально возможный объем хранимой информации. Выражается в битах или словах (в частности, в байтах). Бит хранится запоминающим элементом (ЗЭ), а слово – ячейкой памяти (ЯП), т.е. группой ЗЭ, к которым возможно лишь одновременное обращение. Добавление к единице измерения множителя "**К**" (кило) означает умножение на $2^{10} = 1024$, множителя "**М**" (мега) – умножение на $2^{20} = 1048576$, множителя "**Г**" (гига) – умножение на $2^{30} = 1073741824$, а множителя "**Т**" (тера) – умножение на 2^{40}.

Организация ЗУ – произведение числа хранимых слов на их разрядность. Это дает информационную емкость ЗУ, однако при одной и той же информационной емкости организация ЗУ может быть различной, так что организация является самостоятельным важным параметром. Пример организации: 1К×12 – это соответствует 1024 ячейкам памяти по 12 битов каждая, т.е. 1К 12-разрядных слов.

Быстродействие (производительность) ЗУ оценивают временами считывания, записи и длительностями циклов чтения/записи. *Время считывания* – интервал между моментами появления сигнала чтения и слова на выходе ЗУ. *Время записи* – интервал после появления сигнала записи, достаточный для установления ЯП в состояние, задаваемое входным словом. Минимально допустимый интервал между последовательными чтениями или записями образует соответствующий цикл. Длительности циклов могут превышать время чтения или записи, т.к. после этих операций может потребоваться время для восстановления необходимого начального состояния ЗУ.

Время чтения, записи и длительности циклов – традиционные параметры. Для некоторых современных ЗУ они должны быть дополнены новыми. Причиной является более сложный характер доступа к хранимым данным, когда обращение к первому слову некоторой группы слов (пакета) требует большего времени, чем обращение к последующим. Для таких режимов вводят параметр *времени доступа при первом обращении* (Latency) и *темпа передач* для последующих слов пакета (Bandwidth). Темп передач в свою очередь оценивается двумя значениями – *предельным* (внутри пакета) и *усредненным* (с учетом Latency). С уменьшением пакета усредненный темп снижается, все более отличаясь от предельного.

Помимо рассмотренных основных параметров для ЗУ указывают еще целый набор временны́х интервалов. Перечисленные выше динамические параметры являются *эксплуатационными* (измеряемыми).

Кроме них, существует ряд *режимных параметров*, обеспечение которых необходимо для нормального функционирования ЗУ, поскольку оно имеет несколько сигналов управления, для которых должно быть обеспечено определенное взаимное расположение во времени. Для этих сигналов задаются длительности и ограничения по взаимному положению во времени.

Один из возможных наборов сигналов ЗУ (рис. 1.17) включает в себя следующие сигналы:

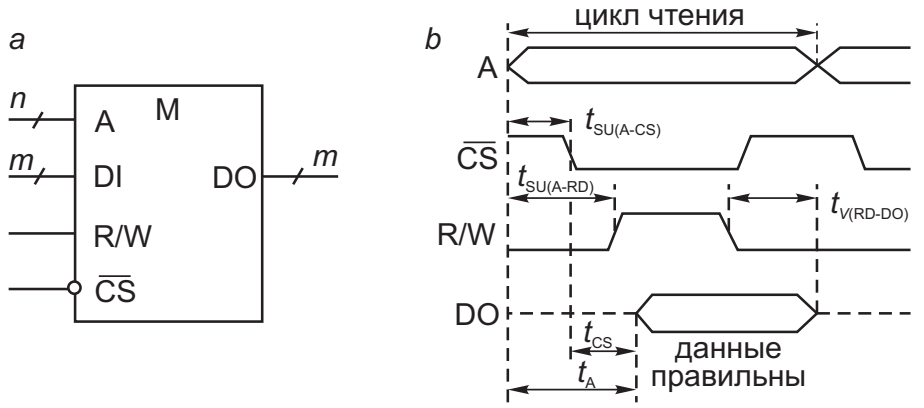


Рис. 1.17. Типичные сигналы ЗУ (*а*) и их временные диаграммы (*б*)

- `A` – адрес, разрядность которого n определяется числом ячеек ЗУ, т.е. максимально возможным числом хранимых в ЗУ слов. Для ЗУ типично число ячеек, выражаемое целой степенью двойки. Адрес является номером ячейки, к которой идет обращение. Очевидно, что разрядность адреса связана с числом хранимых слов N соотношением $n = \log_2 N$ (имеется в виду максимально возможное число хранимых слов). Например, ЗУ с информационной емкостью 64К слов имеет 16-разрядные адреса;
- `CS` – (Chip Select) или `CE` (Chip Enable), который разрешает или запрещает работу данной микросхемы;
- `R/W` – (Read/Write) задает выполняемую операцию (при единичном значении – чтение, при нулевом – запись);
- `DI` и `DO` (Data Input) и (Data Output) – шины входных и выходных данных, разрядность которых m определяется организацией ЗУ (разрядностью его ячеек). В некоторых ЗУ эти линии объединены.

Требования к взаимному временному положению двух сигналов (`A` и `B`) задаются временами предустановки, удержания и сохранения.

Время предустановки сигнала `A` относительно сигнала `B` $t_{\mathrm{SU(A-B)}}$ есть интервал между началами обоих сигналов.

Время удержания $t_{\mathrm{H(A-B)}}$ – это интервал между началом сигнала `A` и окончанием сигнала `B`.

Время сохранения $t_{V(A-B)}$ – интервал между окончанием сигнала A и окончанием сигнала B.

Длительности сигналов обозначаются как t_W (индекс от слова Width – ширина).

Для ЗУ характерна следующая последовательность сигналов. Прежде всего подается адрес, чтобы последующие операции не коснулись какой-либо другой ячейки, кроме выбранной. Затем разрешается работа микросхемы сигналом CS (CE) и подается строб чтения/записи R/W (взаимное положение сигналов CS и R/W для разных ЗУ может быть различным). Если задана, например, операция чтения, то после подачи перечисленных сигналов ЗУ готовит данные для чтения, что требует определенного времени. Задний фронт сигнала R/W, положение которого во времени должно обеспечивать установление правильных данных на выходе ЗУ, считывает данные.

Пример временной диаграммы для рассмотренного набора сигналов ЗУ и операции чтения приведен на рис. 1.17.

Индексом *A* (от слова Access) обозначаются согласно стандарту времена доступа – интервалы времени от появления того или иного управляющего сигнала до появления информационного сигнала на выходе. Время доступа относительно сигнала адреса обозначается, если следовать правилу, как $t_{A(A)}$, но часто просто как t_A. Аналогично этому время доступа относительно сигнала CS, т. е. $t_{A(CS)}$, часто обозначается просто как t_{CS}. Время t_A называют также временем выборки, а время t_{CS} – временем выбора.

Кроме отмеченных параметров для ЗУ используется и ряд других (уровни напряжений, токи, емкости выводов, температурный диапазон и т.д.), которые не требуют специального рассмотрения, т.к. они традиционны для цифровой схемотехники. Исключение составляет свойство энергонезависимости, т.е. способность ЗУ сохранять данные при отключении напряжения питания. Энергонезависимость может быть естественной, т.е. присущей самим ЗЭ, или искусственной, достигаемой введением резервных источников питания, автоматически подключаемых к накопителю ЗУ при снятии основного питания.

1.3.2. Классификация ЗУ

Для классификации ЗУ (рис. 1.18) важнейшим признаком является способ доступа к данным.

Память с произвольным доступом

При адресном доступе код на адресном входе указывает ячейку, с которой ведется обмен. Все ячейки адресной памяти в момент обра-

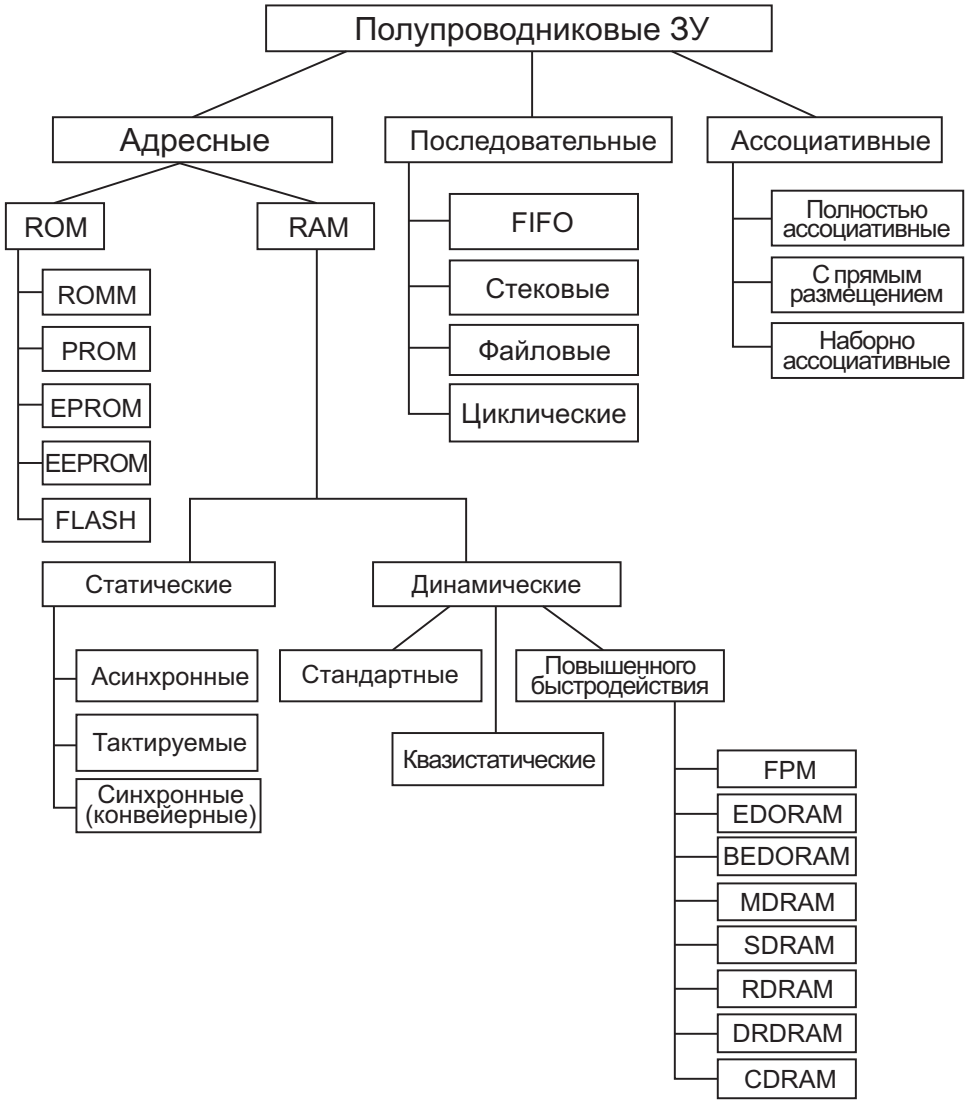


Рис. 1.18. Классификация полупроводниковых ЗУ

щения равнодоступны. Эти ЗУ наиболее разработаны, и другие виды памяти часто строят на основе адресной с соответствующими модификациями.

Адресные ЗУ делятся на RAM (Random Access Memory) и ROM (Read-Only Memory). Русские синонимы термина RAM: ОЗУ (оперативные ЗУ или ЗУ с произвольной выборкой). Оперативные ЗУ хранят данные, участвующие в обмене при исполнении текущей программы, которые могут быть изменены в произвольный момент времени. Запоминающие элементы ОЗУ, как правило, не обладают энергонезависимостью.

В ROM (русский эквивалент – ПЗУ, постоянные ЗУ) содержимое либо вообще не изменяется, либо изменяется, но редко и в специальном режиме. Для рабочего режима – это “память только для чтения”. RAM делятся на статические и динамические.

В статических ОЗУ – SRAM (Static RAM) запоминающими элементами являются триггеры, сохраняющие свое состояние, пока схема находится под питанием и нет новой записи данных. Статические ЗУ можно разделить на асинхронные, тактируемые и синхронные (конвейерные). В *асинхронных* сигналы управления могут задаваться как импульсами, так и уровнями. Например, сигнал разрешения работы CS может оставаться неизменным и разрешающим на протяжении многих циклов обращения к памяти. В *тактируемых ЗУ* некоторые сигналы обязательно должны быть импульсными; например сигнал разрешения работы $\overline{\text{CS}}$ в каждом цикле обращения к памяти должен переходить из пассивного состояния в активное (должен формироваться фронт этого сигнала в каждом цикле). Этот тип ЗУ часто называют синхронным. Здесь использован термин “тактируемые”, чтобы “освободить” термин “синхронные” для новых типов ЗУ, в которых организован *конвейерный* тракт передачи данных, синхронизируемый от тактовой системы процессора, что дает повышение темпа передач данных в несколько раз. Статические ЗУ в 4…5 раз дороже динамических и приблизительно во столько же раз меньше по информационной емкости. Их достоинством является высокое быстродействие, а типичной областью использования – схемы кэш-памяти.

Динамические ЗУ – DRAM (Dynamic RAM) характеризуются наибольшей информационной емкостью и невысокой стоимостью, поэтому именно они используются как основная память микропроцессорных систем. Данные в динамических ЗУ хранятся в виде зарядов конденсаторов, образуемых элементами МОП-структур. Саморазряд конденсаторов ведет к разрушению данных, поэтому они должны периодически (каждые несколько миллисекунд) регенерироваться. В то же время плотность упаковки динамических элементов памяти в несколько раз превышает плотность упаковки, достижимую в статических RAM.

Регенерация данных в динамических ЗУ осуществляется с помощью специальных контроллеров. Разработаны также ЗУ с динамическими запоминающими элементами, имеющие внутреннюю встроенную систему регенерации, у которых внешнее поведение относительно управляющих сигналов становится аналогичным поведению статических ЗУ. Такие ЗУ называют *квазистатическими*.

Поскольку от этой памяти требуется высокое быстродействие, разработаны многочисленные архитектуры повышенного быстродействия, перечисленные в классификации (см. рис. 1.18). В частности, конвейерная организация ЗУ играет важную роль в повышении быстродействия динамических ЗУ (вариант SDRAM).

Постоянные ЗУ. Постоянная память типа ROM(M) программируется при изготовлении методами интегральной технологии с помощью одной из используемых при этом масок. В русском языке ее можно назвать “масочные ПЗУ”. Для потребителя – это в полном смысле слова постоянная память, т.к. изменить ее содержимое он не может.

Программируемые ЗУ. В следующих трех разновидностях ROM в обозначениях присутствует буква P (от Programmable). Это программируемая пользователем память (в русской терминологии ППЗУ – программируемые ПЗУ). Ее содержимое записывается либо однократно (в PROM), либо может быть заменено путем стирания старой информации и записи новой (в EPROM и EEPROM). В EPROM стирание выполняется с помощью облучения кристалла ультрафиолетовыми лучами, ее русское название “репрограммируемое ПЗУ с УФ-стиранием”. В EEPROM стирание производится электрическими сигналами, ее русское название “репрограммируемое ПЗУ с электрическим стиранием”. Английские названия расшифровываются как Electrically Programmable ROM и Electrically Erasable Programmable ROM. Программирование PROM и репрограммирование EPROM и EEPROM производятся в обычных лабораторных условиях с помощью либо специальных программаторов, либо специальных режимов без специальных приборов (для EEPROM). Запись данных и для EPROM и для EEPROM производится электрическими сигналами.

Память типа Flash по запоминающему элементу подобна памяти типа EEPROM (или иначе E^2PROM), но имеет структурные и технологические особенности, позволяющие выделить ее в отдельный вид. В схемах Flash-памяти не предусмотрено стирание отдельных слов, стирание информации осуществляется либо для всей памяти одновременно, либо для достаточно больших блоков. Это позволяет упростить схемы ЗУ, т.е. получить высокую степень интеграции и быстродействия при снижении стоимости. Наряду с одновременным стиранием всего содержимого ЗУ имеются схемы с блочной структурой, в которых весь массив разбит на блоки, стираемые независимо друг от друга. Объем таких блоков сильно разнится: от 256 байт до 128 Кбайт.

Число циклов репрограммирования Flash-памяти хотя и велико, но ограничено, т.е. ячейки при перезаписи “изнашиваются”. Для увеличения срока службы ЗУ используют специальные алгоритмы для “разравнивания” числа перезаписей по всем блокам микросхемы.

Flash-память имеет архитектурные и схемотехнические разновидности, обусловливающие два основных направления ее эффективного использования: хранение не очень часто изменяемых данных (обновляемых программ в частности) и замена памяти на магнитных дисках (файловая Flash-память).

Память с последовательным доступом

В ЗУ с последовательным доступом записываемые данные образуют некоторую очередь. Считывание происходит из очереди слово за словом, либо в порядке записи, либо в обратном порядке. Моделью такого ЗУ является последовательная цепочка ЗЭ, в которой данные передаются между элементами. Память с последовательным доступом строится либо с использованием продвижения данных в цепочке элементов (по подобию с регистрами сдвига), либо с хранением данных в адресном ЗУ с необходимым управлением адресом доступа.

Основными представителями этого вида памяти являются видеопамять, буфер FIFO и стек (LIFO).

Прямой порядок считывания имеет место в *буферах FIFO* с дисциплиной "первый пришел – первый вышел" (First In – First Out), а также в файловых и циклических ЗУ. Интервалы между словами могут быть совершенно различными, т.к. моменты записи слова в буфер и считывания из него задаются внешними сигналами управления независимо друг от друга. Возможность иметь разный темп приема и выдачи слов необходима, например, если приемник способен принимать данные, поступающие регулярно с некоторой частотой, а источник информации выдает слова в более быстром темпе и, может быть, к тому же нерегулярно.

Разница между памятью FIFO и файловым ЗУ состоит в том, что в FIFO запись в пустой буфер сразу же становится доступной для чтения, т. е. поступает в конец цепочки (модели ЗУ). В файловых ЗУ данные поступают в начало цепочки и появляются на выходе после некоторого числа обращений, равного числу элементов в цепочке. При независимости операций считывания и записи фактическое расположение данных в ЗУ на момент считывания не связано с каким-либо внешним признаком. Поэтому записываемые данные объединяют в блоки, обрамляемые специальными символами конца и начала (файлы). Прием данных из файлового ЗУ начинается после обнаружения приемником символа начала блока.

В циклических ЗУ слова доступны одно за другим с постоянным периодом, определяемым емкостью памяти. К такому типу среди полупроводниковых ЗУ относится видеопамять (VRAM).

Видеопамять работает циклично, на ее выходе последовательно в порядке сканирования экрана монитора лучом появляются коды, задающие параметры светосистемы (яркость, цвет) элементарных точек экрана – пикселов. Текущее изображение на мониторе – кадр – представлено последовательностью слов, длина которой равна числу пикселов на экране. Слово, соответствующее одному пикселу, может

иметь разрядность от 8 (для черно-белых мониторов) до 24 (для полноцветного режима).

При реализации на основе адресной памяти циклический доступ к данным обеспечивается счетчиком адреса с модулем, равным числу запоминаемых слов. При считывании после каждого обращения адрес увеличивается на единицу, обеспечивая последовательное обращение ко всем ячейкам ЗУ. При переполнении счетчика формируется сигнал начала кадра для управления монитором (для запуска кадровой синхронизации). В первом случае сигнал переполнения счетчика и его переход на начальный адрес являются сигналом начала передачи блока данных из основной памяти или видеобуфера.

Во втором случае адрес изменяемой ячейки (номер пиксела) и данные сохраняются в буфере, а в момент совпадения этого адреса и содержимого счетчика выполняется один цикл записи нового слова. Все остальное время ЗУ работает обычным образом.

Считывание в обратном порядке свойственно стековым ЗУ, для которых реализуется дисциплина “последний пришел – первый вышел”. Такие ЗУ называют *буферами LIFO* (Last In – First Out). Время доступа к конкретной единице хранимой информации в последовательных ЗУ представляет собою случайную величину. В наихудшем случае для такого доступа может потребоваться просмотр всего объема хранимых данных.

Ассоциативный доступ реализует поиск информации по некоторому признаку, а не по ее расположению в памяти (адресу или месту в очереди). В наиболее полной версии все хранимые в памяти слова одновременно проверяются на соответствие признаку, например на совпадение определенных полей слов (тегов – от английского слова tag) с признаком, задаваемым входным словом (теговым адресом). На выход выдаются слова, удовлетворяющие признаку. Дисциплина выдачи слов, если тегу удовлетворяют несколько слов, а также дисциплина записи новых данных могут быть разными. Основная область применения ассоциативной памяти в микропроцессорных системах — кэширование данных.

Кэш-память (Cache) запоминает копии информации, передаваемые между устройствами (прежде всего между процессором и основной памятью). Она имеет небольшую емкость в сравнении с основной памятью и более высокое быстродействие (реализуется на триггерных элементах памяти). При чтении данных сначала выполняется обращение к кэш-памяти. Если в кэше имеется копия данных адресованной ячейки основной памяти, то кэш выдает данные на общую шину данных. Эффективность кэширования обусловлена тем, что большинство

прикладных программ имеют циклический характер и многократно используют одни и те же данные. Поэтому после первого использования данных из относительно медленной основной памяти повторные обращения требуют меньше времени.

В полностью ассоциативной кэш-памяти (Fully Associated Cache Memory, FACM) каждая ячейка хранит данные, а поле "тег" – полный физический адрес информации, копия которой записана. При любых обменах физический адрес запрашиваемой информации сравнивается с полями "тег" всех ячеек. При совпадении адреса с тегом: в режиме чтения данные из кэша выдаются на ШД; в режиме записи данные с ШД записываются в найденную ячейку кэш-памяти. При отсутствии совпадения адреса с тегом сначала ищется свободная или наиболее давно не используемая ячейка кэш-памяти, затем: в режиме чтения данные из основной памяти помещаются (вместе с адресом) в эту ячейку и выдаются на ШД; в режиме записи данные размещаются в этой ячейке кэш-памяти. Копирование данных в основную память выполняется аппаратно под управлением специального контроллера, когда нет обращений к памяти. Кэш-память типа FACM – наиболее совершенная, но и наиболее сложная и дорогая память. Существует несколько разновидностей FACM, отличающихся по экономичности затрат аппаратных средств на их реализацию.

В кэш-памяти с прямым размещением (с прямым отображением) несколько страниц основной памяти строго соответствуют одной строке кэша. Тег для кэш-памяти с прямым размещением сильно сокращается по разрядности. Достоинство кэш-памяти с прямым размещением – экономичность по аппаратным затратам. Недостаток – ограничения на расположение страниц в кэше, что может не позволить сформировать в нем оптимальный набор страниц.

Наборно-ассоциативная кэш-память (с ассоциацией по нескольким направлениям) – это промежуточный по сложности и эффективности вариант между двумя предыдущими типами кэш-памяти. Возможность свободного размещения страниц в наборе позволяет сформировать в кэше лучший состав страниц, т.к. имеется возможность выбрать ту или иную заменяемую страницу.

В современных микропроцессорных системах кэш первого уровня, обозначаемый L1 (Level) и расположенный внутри процессора, обычно имеет наборно-ассоциативную структуру, а кэш второго уровня L2, расположенный вне процессора, – структуру с прямым размещением.

Технико-экономические параметры ЗУ существенно зависят от их схемотехнической реализации. По этому признаку также возможна классификация ЗУ, однако удобнее рассматривать этот вопрос применительно к отдельным типам памяти.

1.3.3. Основные структуры запоминающих устройств

Адресные ЗУ представлены в классификации статическими и динамическими оперативными устройствами и памятью типа ROM. Многочисленные варианты этих ЗУ имеют много общего с точки зрения структурных схем, что делает более рациональным не конкретное рассмотрение каждого ЗУ в полном объеме, а изучение некоторых обобщенных структур с последующим описанием запоминающих элементов для различных ЗУ. Общность структур особенно проявляется для статических ОЗУ и памяти типа ROM. Структуры динамических ОЗУ имеют свою специфику. Для статических ОЗУ и памяти типа ROM наиболее характерны структуры 2D, 3D и 2DM.

Структура 2D

В структуре 2D (рис. 1.19) запоминающие элементы ЗЭ организованы в прямоугольную матрицу размерностью $M = k \times m$, где M – информационная емкость памяти в битах; k – число хранимых слов; m – их разрядность. Разрядность адреса A: $n = \log_2 k$. Дешифра-

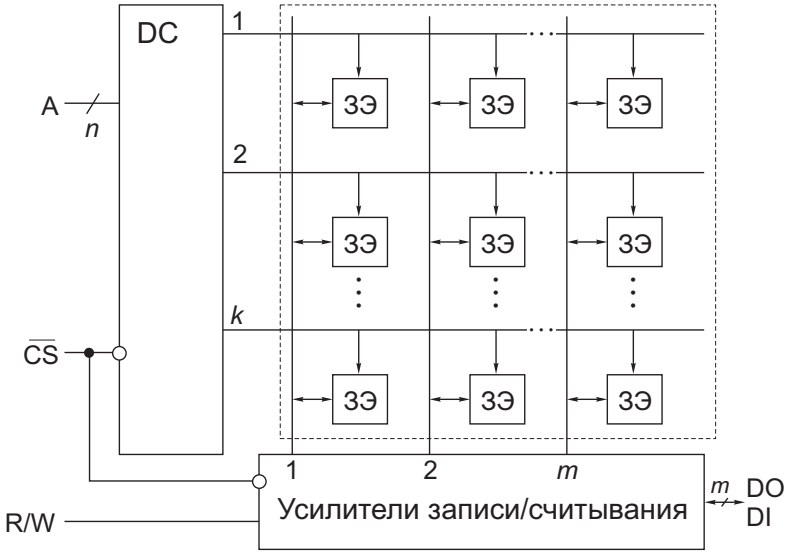


Рис. 1.19. Структура ЗУ типа 2D

тор адреса DC при наличии разрешающего сигнала $\overline{CS}$ активизирует одну из выходных линий, разрешая одновременный доступ ко всем элементам выбранной строки, хранящей слово, адрес которого соответствует номеру строки. Элементы одного столбца соединены вертикальной линией – внутренней линией данных (разрядной линией, линией записи/считывания). Элементы столбца хранят одноименные биты всех слов. Направление обмена устанавливается усилителями записи/считывания под воздействием сигнала R/W.

Структура 2D применяется лишь в ЗУ малой информационной емкости, т.к. при возрастании емкости происходит неоправданное усложнение электронных схем. В частности, дешифратор адреса должен иметь число выходов, совпадающее с числом хранимых слов.

Структура 3D

Структур 3D позволяет существенно упростить дешифраторы адреса с помощью двухкоординатной выборки ЗЭ. Принцип двухкоординатной выборки поясняется (рис. 1.20) на примере ПЗУ, реализующей только операцию чтения. Адрес разрядности n делится на две поло-

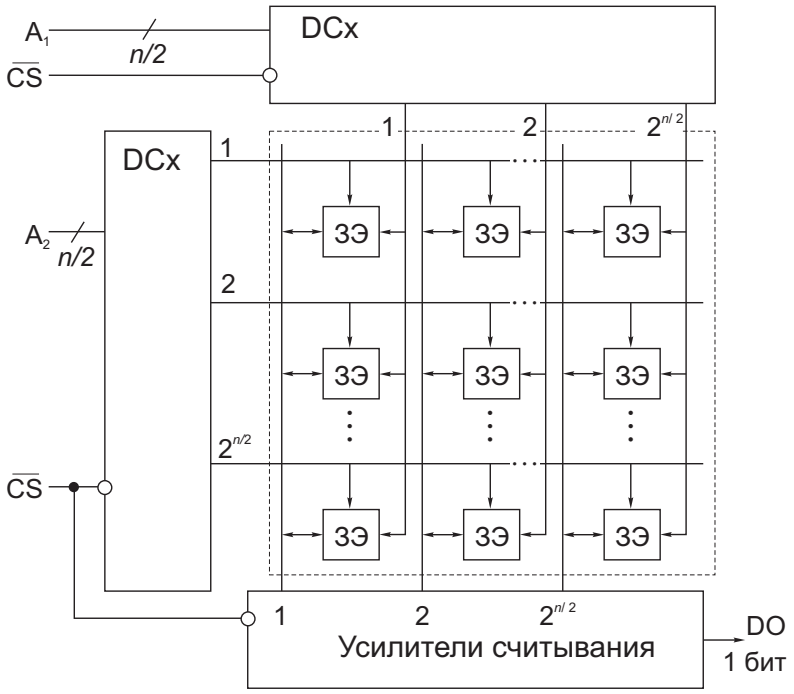


Рис. 1.20. Структура ЗУ типа 3D с однобитовой организацией

винки, каждая из которых декодируется отдельно. Выбирается ЗЭ, находящийся на пересечении активных линий выходов обоих дешифраторов. Таких пересечений будет как раз $2^{n/2} \times 2^{n/2} = 2^n$. Эта цифра гораздо меньше, чем 2^n при реальных значениях n. Уже для ЗУ с небольшой информационной емкостью видна существенная разница: для структуры 2D при хранении 1K слов требуется дешифратор с 1024 выходами, а для структуры 3D нужны два дешифратора по 32 выхода каждый. Недостатком структуры 3D является в первую очередь усложнение элементов памяти, имеющих двухкоординатную выборку.

Структура 3D (см. рис. 1.20) применяется и в ЗУ с многоразрядной организацией. В этом случае несколько матриц управляются двумя дешифраторами, относительно которых эти матрицы включены параллельно. Каждая матрица выдает один бит адресованного слова, а число матриц равно разрядности хранимых слов.

Структуры типа 3D имеют также довольно ограниченное применение, поскольку в структурах типа 2DM (2D модифицированная) сочетаются достоинства обеих рассмотренных структур – упрощается дешифрация адреса и не требуется запоминающих элементов с двухкоординатной выборкой.

Структура 2DM

Структура 2DM (рис. 1.21) для матрицы запоминающих элементов с адресацией от дешифратора DCx имеет до некоторой степени характер структуры 2D: возбужденный выход дешифратора выбирает

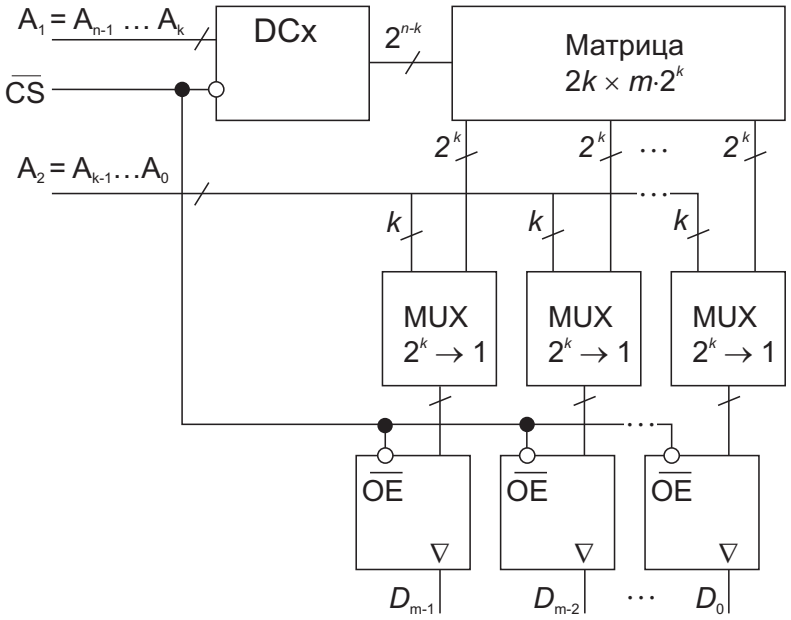


Рис. 1.21. Структура ЗУ типа 2DM для ПЗУ

целую строку. Однако в отличие от структуры 2D длина строки не равняется разрядности хранимых слов, а многократно ее превышает. При этом число строк матрицы уменьшается и соответственно уменьшается число выходов дешифратора. Для выбора одной из строк служат не все разряды адреса, а лишь их старшая часть $A_{n-1}\ldots A_k$. Оставшаяся младшая часть адреса $A_{k-1}\ldots A_0$ используется, чтобы выбрать необходимое слово из того множества слов, которое содержится в строке. Это выполняется с помощью мультиплексоров, на адресные входы которых подаются коды $A_{k-1}\ldots A_0$. Длина строки равна $m2^k$, где m – разрядность хранимых слов. Из каждого "отрезка" строки длиной 2^k мультиплексор выбирает один бит. На выходах мультиплексоров формируется выходное слово. По разрешающему сигналу $\overline{\text{CS}}$, поступающему на входы OE управляемых буферов с тремя состояниями, выходное слово выдается на внешнюю шину. Структура типа 2DM с успехом применяется и для построения ОЗУ.

1.3.4. Постоянные запоминающие устройства

Запоминающие устройства типа ROM имеют многоразрядную организацию и обычно выполняются по структуре 2DM. Простейшие ЗУ могут иметь структуру 2D. Технологии изготовления постоянных ЗУ разнообразны: ТТЛ(Ш), КМОП, *n*-МОП и др.

Масочные ЗУ

Элементом связи в масочных ЗУ могут быть диоды, биполярные транзисторы, МОП-транзисторы и т.д. В матрице ЗУ (рис. 1.22, *a*) горизонтальные линии являются линиями выборки слов, а вертикальные – линиями считывания. Считываемое слово определяется расположе-

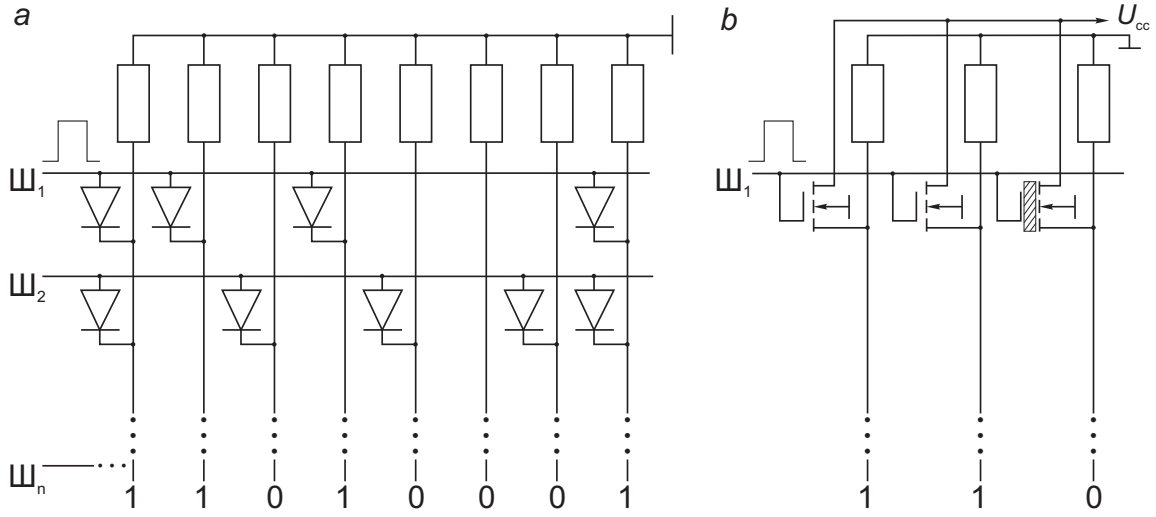


Рис. 1.22. Матрица масочного ЗУ на диодных (*a*) и МОП-транзисторных (*b*) запоминающих элементах

нием диодов в узлах координатной сетки. При наличии диода высокий потенциал выбранной горизонтальной линии передается на соответствующую вертикальную линию, и в данном разряде слова появляется сигнал логической единицы. При отсутствии диода потенциал близок к нулевому, т.к. вертикальная линия через резистор связана с землей. На рис. 1.22, *a* при возбужденной линии выборки Ш1 считывается слово 11010001 (это слово хранится в ячейке номер 1). При возбуждении Ш2 считывается слово 10101011 (оно хранится в ячейке номер 2). Линии выборки являются выходами дешифратора адреса, каждая адресная комбинация возбуждает свой выход дешифратора, что приводит к считыванию слова из адресуемой ячейки.

В матрице с диодными элементами в одних узлах матрицы диоды изготавливаются, а в других – нет. Для удешевления производства при изготовлении ЗУ стремятся варьировать только один шаблон так, чтобы одни элементы связи были законченными и работоспособными, а

другие – незавершенными и как бы отсутствующими. Для матриц с МОП-транзисторами, часто в МОП-транзисторах, соответствующих хранению нуля, просто увеличивают толщину подзатворного окисла, что ведет к увеличению порогового напряжения транзистора. В этом случае рабочие напряжения ЗУ не в состоянии открыть транзистор. Постоянное закрытое состояние транзистора аналогично его отсутствию. Матрица с МОП-транзисторами показана на рис. 1.22, *b*.

ЗУ с масочным программированием отличаются компактностью запоминающих элементов и, следовательно, высоким уровнем интеграции. При больших объемах производства масочное программирование предпочтительно, но при недостаточной тиражности ЗУ затраты на проектирование и изготовление шаблонов для технологического программирования ЗУ окажутся чрезмерно высокими.

ЗУ типа PROM

В ЗУ типа PROM микросхемы программируются устранением или созданием специальных перемычек. В исходной заготовке имеются (или отсутствуют) все перемычки. После программирования остаются (или возникают) только необходимые.

Устранение части перемычек характерно ЗУ с плавкими перемычками (типа *fuse* – предохранитель). При этом в исходном состоянии ЗУ имеет все перемычки, а при программировании часть их ликвидируется путем расплавления импульсами тока достаточно большой амплитуды и длительности.

В ЗУ с плавкими перемычками эти перемычки включаются в электроды диодов или транзисторов. Перемычки могут быть металлическими или поликристаллическими. В исходном состоянии запоминающий элемент хранит логическую единицу, логический нуль нужно записать путем расплавления перемычки.

Создание части перемычек соответствует схемам, которые в исходном состоянии имеют непроводящие перемычки в виде пары встречно включенных диодов или тонких диэлектрических слоев, пробиваемых при программировании с образованием низкоомных сопротивлений. Схемы с тонкими пробиваемыми диэлектрическими перемычками (типа *antifuse*) наиболее компактны и совершенны. Их применение характерно для программирования логических СБИС.

Второй тип запоминающего элемента PROM – два диода, включенных навстречу друг другу. В исходном состоянии сопротивление цепочки велико и практически соответствует разомкнутой цепи. Для записи логической единицы к цепочке прикладывают повышенное напряжение и пробивают обратно смещенный диод. Диод пробивается с образованием в нем короткого замыкания, которое играет роль появившейся проводящей перемычки.

Запоминающие элементы с плавкими перемычками и парами диодов показаны на рис. 1.23 в исходном состоянии и после программирования.

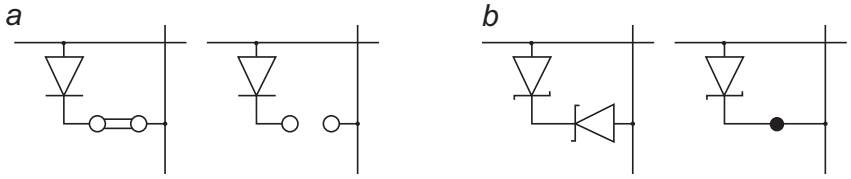


Рис. 1.23. Запоминающие элементы с плавкими перемычками (*a*) и диодными парами (*b*) в исходном состоянии и после программирования

Программирование ЗУ с плавкими перемычками реализуется простыми аппаратными средствами. Однако плавкие перемычки занимают на кристалле относительно много места. Уровень интеграции ЗУ с такими перемычками существенно ниже, чем в масочных ЗУ. В то же время простота программирования пользователем и невысокая стоимость в свое время обусловили широкое распространение ЗУ типа PROM.

ЗУ типа EPROM и EEPROM

В репрограммируемых ЗУ типов EPROM и EEPROM (E^2PROM) возможно стирание старой информации и замена ее новой в результате специального процесса, для проведения которого ЗУ выводится из рабочего режима. Рабочий режим (чтение данных) – процесс, выполняемый с относительно высокой скоростью. Замена же содержимого памяти требует выполнения гораздо более длительных операций.

По способу стирания старой информации различают ЗУ со стиранием ультрафиолетовыми лучами (EPROM или в русской терминологии РПЗУ-УФ, т.е. репрограммируемые ПЗУ с ультрафиолетовым стиранием) и электрическим стиранием (E^2PROM или РПЗУ-ЭС).

Запоминающими элементами современных РПЗУ являются транзисторы типов МНОП и ЛИЗМОП (добавление ЛИЗ к обозначению МОП происходит от слов **Л**авинная **И**нжекция **З**аряда).

МНОП-транзистор отличается от обычного МОП-транзистора двухслойным подзатворным диэлектриком. На поверхности кристалла расположен тонкий слой двуокиси кремния SiO_2, далее – более толстый слой нитрида кремния Si_3N_4 и затем уже затвор (рис. 1.24, *a*). На границе диэлектрических слоев возникают центры захвата заряда. Благодаря туннельному эффекту, носители заряда могут проходить через тонкую пленку окисла толщиной не более 5 нм и скапливаться на границе раздела слоев. Этот заряд и является носителем информации, хранимой МНОП-транзистором. Заряд записывают созданием под затвором напряженности электрического поля, достаточной для

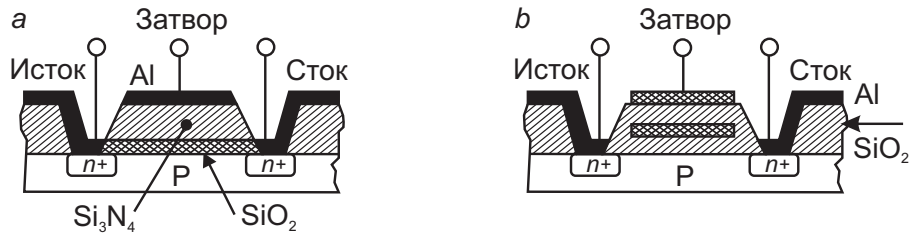


Рис. 1.24. Структуры транзисторов типов МНОП (*a*) и ЛИЗМОП с двойным затвором (*b*)

возникновения туннельного перехода носителей заряда через тонкий слой SiO_2. На границе раздела диэлектрических слоев можно создавать заряд любого знака в зависимости от направленности электрического поля в подзатворной области. Наличие заряда влияет на пороговое напряжение транзистора.

Для МНОП-транзистора с *n*-каналом отрицательный заряд на границе раздела слоев повышает пороговое напряжение (экранирует воздействие положительного напряжения на затворе, отпирающего транзистор). При этом пороговое напряжение возрастает настолько, что рабочие напряжения на затворе транзистора не в состоянии его открыть (создать в нем проводящий канал). Транзистор, в котором заряд отсутствует или имеет другой знак, легко открывается рабочим значением напряжения. Так осуществляется хранение бита в МНОП: одно из состояний трактуется как отображение логической единицы, другое – нуля.

При программировании ЗУ используются относительно высокие напряжения, около 20 В. После снятия высоких напряжений туннельное прохождение носителей заряда через диэлектрик прекращается и заданное транзистору пороговое напряжение остается неизменным.

После $10^4 \ldots 10^6$ перезаписей МНОП-транзистор перестает устойчиво хранить заряд. РПЗУ на МНОП-транзисторах энергонезависимы и могут хранить информацию месяцами, годами и десятками лет. Перед новой записью старая информация стирается записью нулей во все запоминающие элементы. Тип ЗУ – РПЗУ-ЭС.

Транзисторы типа ЛИЗМОП всегда имеют так называемый плавающий затвор, который может быть единственным или вторым, дополнительным к обычному (управляющему) затвору. Транзисторы с одним плавающим затвором используются в ЗУ типа РПЗУ-УФ, а транзисторы с двойным затвором пригодны для применения как в РПЗУ-УФ, так и в РПЗУ-ЭС. Рассмотрим более современный тип – ЛИЗМОП-транзистор с двойным затвором (см. рис. 1.24, *b*).

Принцип работы ЛИЗМОП с двойным затвором близок к принципу работы МНОП-транзистора – здесь также между управляющим затвором и областью канала помещается область, в которую при про-

граммировании можно вводить заряд, влияющий на величину порогового напряжения транзистора. Только область введения заряда представляет собой не границу раздела слоев диэлектрика, а окруженную со всех сторон диэлектриком проводящую область (обычно из поликристаллического кремния), в которую, как в ловушку, можно ввести заряд, способный сохраняться в ней в течение очень длительного времени. Эта область и называется плавающим затвором.

При подаче на управляющий затвор, исток и сток импульса положительного напряжения относительно большой амплитуды 20…25 В в обратно смещенных *p-n* переходах возникает лавинный пробой, область которого насыщается электронами. Часть электронов, имеющих энергию, достаточную для преодоления потенциального барьера диэлектрической области, проникает в плавающий затвор. Снятие высокого программирующего напряжения восстанавливает обычное состояние областей транзистора и запирает электроны в плавающем затворе, где они могут находиться длительное время (в высококачественных приборах – многие годы).

Заряженный электронами плавающий затвор увеличивает пороговое напряжение транзистора настолько, что в диапазоне рабочих напряжений проводящий канал в транзисторе не создается.

При отсутствии заряда в плавающем затворе транзистор работает в обычном ключевом режиме.

Стирание информации может производиться двумя способами – ультрафиолетовым облучением и электрическими сигналами.

В первом случае корпус ИС имеет специальное прозрачное окошко для облучения кристалла. Двуокись кремния и поликремний прозрачны для ультрафиолетовых лучей. Эти лучи вызывают в областях транзистора фототоки и тепловые токи, что делает области прибора проводящими и позволяет заряду покинуть плавающий затвор. Операция стирания информации этим способом занимает десятки минут, информация стирается сразу во всем кристалле. В схемах с УФ-стиранием число циклов перепрограммирования существенно ограничено, т.к. под действием ультрафиолетовых лучей свойства материалов постепенно изменяются. Число циклов перезаписи у отечественных ИС равно 10…100.

Электрическое стирание информации осуществляется подачей на управляющие затворы низкого (нулевого) напряжения, а на стоки – высокого напряжения программирования. Электрическое стирание имеет преимущества: можно стирать информацию не со всего кристалла, а выборочно (индивидуально для каждого адреса). Длительность процесса “стирание-запись” значительно меньше, сильно ослабляются ограничения на число циклов перепрограммирования (допускается $10^4 \ldots 10^6$ таких циклов). Кроме того, перепрограммировать ЗУ можно, не извлекая микросхему из устройства, в котором она работает.

В то же время схемы с электрическим стиранием занимают больше места на кристалле, в связи с чем уровень их интеграции меньше, а стоимость выше. В последнее время эти недостатки быстро преодолеваются и ЭС-стирание вытесняет УФ-стирание.

Предшественниками двухзатворных ЛИЗМОП-транзисторов были однозатворные, имевшие только плавающий затвор. Эти транзисторы изготовлялись обычно с *p*-каналом, поэтому введение электронов в плавающий затвор приводило к созданию в транзисторе проводящего канала, а удаление заряда – к исчезновению такого канала. При использовании этих транзисторов запоминающие элементы состоят из двух последовательно включенных транзисторов: ключевого МОП-транзистора обычного типа для выборки адресованного элемента и ЛИЗМОП-транзистора, состояние которого определяет хранимый бит. Стирание информации производится ультрафиолетовыми лучами.

Подключение двухзатворных ЛИЗМОП-транзисторов к линиям выборки строк и линиям чтения в матрицах ЗУ показано на рис. 1.25. Запись логического нуля осуществляется путем заряда плавающего за-

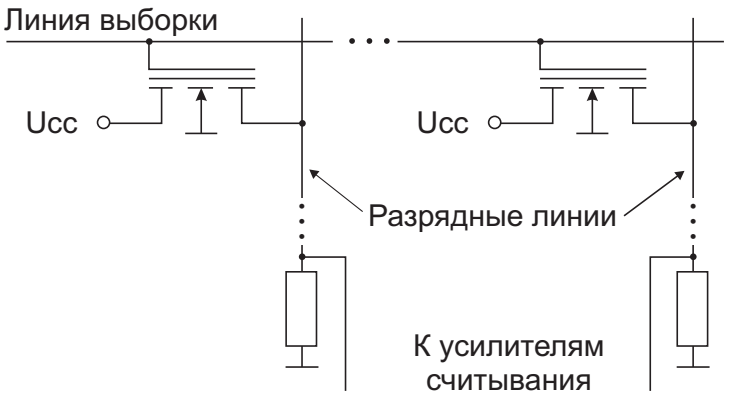


Рис. 1.25. Схема подключения ЛИЗМОП-транзисторов с двойным затвором к линиям выборки и считывания в РПЗУ

твора инжекцией “горячих” электронов в режиме программирования. Стирание информации, под которым понимается удаление заряда из плавающего затвора, приводит к записи во все запоминающие элементы логических единиц, т.к. в данном случае опрашиваемые транзисторы открываются и передают напряжение U_{cc} на линии считывания.

1.3.5. Статические запоминающие устройства

Область применения относительно дорогостоящих статических ОЗУ в системах обработки информации определяется их высоким быстродействием. В частности, они широко используются в кэш-памяти, которая при сравнительно малой емкости должна иметь максимальное быстродействие.

Статические ОЗУ (SRAM), как правило, имеют структуру 2DM, часть их при небольшой информационной емкости строится по структуре 2D. Запоминающими элементами статических ОЗУ служат триггеры с цепями установки и сброса. В связи с этим статические ОЗУ называют также триггерными. Триггеры можно реализовать по любой схемотехнологии (ТТЛ(Ш), $И^2Л$, ЭСЛ, *n*-МОП, КМОП, AsGa и др.), соответственно которой существуют разнообразные схемы ЗУ. Различие в параметрах этих ЗУ отражает специфику той или иной схемотехнологии. В последнее время наиболее интенсивно развиваются статические ЗУ, выполненные по схемотехнологии КМОП, которая по мере уменьшения топологических норм технологического процесса приобретает высокое быстродействие при сохранении своих традиционных преимуществ.

Запоминающие элементы статических ЗУ

Запоминающий элемент ЗУ на *n*-МОП транзисторах (рис. 1.26, *a*) представляет собой RS-триггер на транзисторах Т1 и Т2 с ключами выборки Т3 и Т4. При обращении к данному ЗЭ появляется высокий

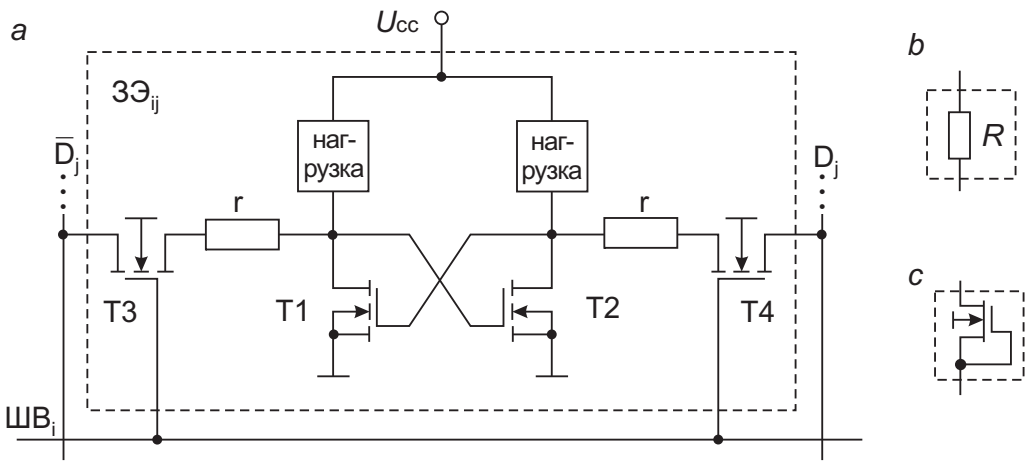


Рис. 1.26. Схема триггерного запоминающего элемента на *n*-МОП транзисторах (*a*) и варианты нагрузок для схемы триггера (*b*, *c*)

потенциал на шине выборки $ШВ_i$ (через *i*, *j* соответственно обозначены номера строки и столбца, на пересечении которых расположен $ЗЭ_{ij}$). Этот потенциал открывает ключи выборки (транзисторы Т3, Т4) по всей строке, и выходы триггеров строки соединяются со столбцовыми шинами считывания-записи. Одна из столбцовых шин связана с прямым выходом триггера (обозначена через D_j), другая – с инверсным ($\overline{D}_j$). Через столбцовые шины можно считывать состояние триггера (штриховыми линиями показан дифференциальный усилитель считывания). Через них же можно записывать данные в триггер, подавая низкий потенциал логического нуля на ту или иную шину.

При подаче нуля на выход $\overline{D}_j$ снижается стоковое напряжение транзистора Т1, что запирает транзистор Т2 и повышает напряжение на его стоке. Это открывает транзистор Т1 и фиксирует созданный на его стоке низкий уровень даже после снятия сигнала записи. Триггер установлен в состояние логической единицы. Аналогичным образом нулевым сигналом по шине D_j можно установить триггер в нулевое состояние. При выборке строки со своими столбцовыми шинами соединяются все триггеры строки, но только одна пара шин связывается с выходными цепями считывания или входной цепью записи в соответствии с адресом столбца.

Резисторы *r* служат для уменьшения емкостных токов в моменты открывания ключевых транзисторов и реализуются как части диффузионных областей этих транзисторов.

В качестве нагрузки могут быть использованы двухполюсники, показанные на рис. 1.26, *b*. В первом случае – это *n*-МОП транзистор со встроенным каналом и нулевым напряжением затвора, т.е. обычный элемент нагрузки в схемах с *n*-каналом.

Стремление к режиму микротоков привело к схеме с нагрузочным поликремниевым резистором (второй случай, нагрузка типа рис 1.26, *c*). Высокоомные нагрузочные резисторы изготовляются из поликристаллического кремния и пространственно расположены над областью транзисторов, что придает схеме также и высокую компактность. Режим микротоков нужен для кристаллов высокого уровня интеграции, но он создает и ряд трудностей, в первую очередь – низкую скорость переключения триггера (микротоки не в состоянии быстро перезаряжать паразитные емкости схемы) и маломощность выходных сигналов. Первый недостаток преодолевается тем, что триггер переключается под воздействием мощных сигналов записи информации через ключевые транзисторы, а не за счет только внутренних токов цепей обратных связей. Вторая особенность требует применения высокочувствительных усилителей считывания. Это объясняет использование так называемых усилителей-регенераторов в статических ЗУ (ранее они были характерны только для динамических).

Запоминающие элементы статических ОЗУ (КМОП технология), показаны на рис. 1.27,*a* в обозначениях США. Эти элементы построены так же, как и элементы на *n*-МОП транзисторах, и не требуют дополнительных пояснений.

Выходной каскад с третьим состоянием

На рис 1.27, *b* показан выходной каскад с третьим состоянием, используемый в КМОП ЗУ. Низкий уровень сигнала $\overline{CS}$ и высокий уровень сигнала R/W, означающие разрешение операции чтения, создают

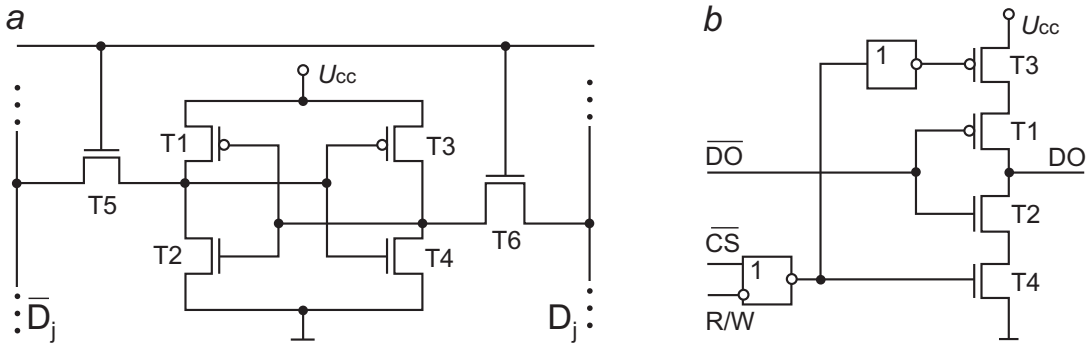


Рис. 1.27. Схемы триггерного запоминающего элемента (*a*) и выходного каскада (*b*) в схемотехнике КМОП

на выходе элемента ИЛИ-НЕ высокий уровень логической единицы, открывающий транзисторы Т3 и Т4 и тем самым позволяющий нормально работать инвертору на транзисторах Т1 и Т2, через который данные передаются на выход. При всех иных комбинациях сигналов $\overline{CS}$ и R/W выход элемента ИЛИ-НЕ имеет низкий уровень логического нуля, при котором транзисторы Т3 и Т4 заперты и выход DO находится в состоянии “отключено”.

1.3.6. Динамические запоминающие устройства

В динамических ЗУ (DRAM) данные хранятся в виде зарядов емкостей МОП-структур и основой ЗЭ является просто конденсатор небольшой емкости. Такой ЗЭ значительно проще триггерного, содержащего 6 транзисторов, что позволяет разместить на кристалле намного больше ЗЭ (в 4…5 раз) и обеспечивает динамическим ЗУ максимальную емкость. В то же время конденсатор неизбежно теряет со временем свой заряд, и хранение данных требует их периодической регенерации (через несколько миллисекунд).

Запоминающие элементы

Известны конденсаторные ЗЭ разной сложности. В последнее время практически всегда применяют однотранзисторные ЗЭ – лидеры компактности, размеры которых настолько малы, что на их работу стали влиять даже α-частицы, излучаемые элементами корпуса ИС.

Электрическая схема и конструкция однотранзисторного ЗЭ показаны на рис. 1.28. Ключевой транзистор отключает запоминающий конденсатор от линии записи-считывания или подключает его к ней. Сток транзистора не имеет внешнего вывода и образует одну из обкладок конденсатора. Другой обкладкой служит подложка. Между обкладками расположен тонкий слой диэлектрика – оксида кремния SiO_2.

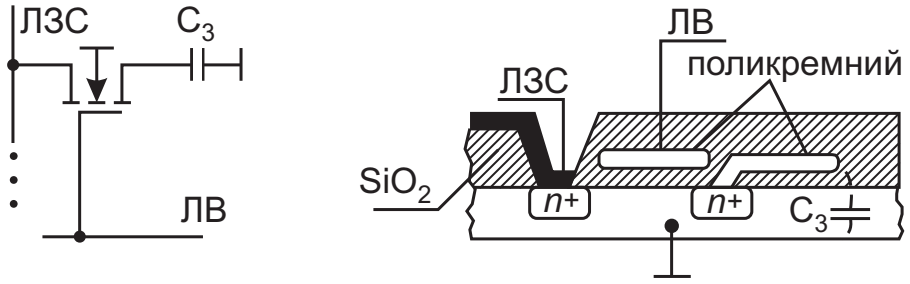


Рис. 1.28. Схема и структура запоминающего элемента динамического ЗУ

В режиме хранения ключевой транзистор заперт. При выборке данного ЗЭ на затвор подается напряжение, отпирающее транзистор. Запоминающая емкость через проводящий канал подключается к линии записи-считывания и в зависимости от заряженного или разряженного состояния емкости различно влияет на потенциал линии записи-считывания. При записи потенциал линии записи-считывания передается на конденсатор, определяя его состояние. Считывание является разрушающим – подключение запоминающей емкости к линии записи-считывания изменяет ее заряд. Мерами преодоления отмеченных недостатков служат способы увеличения емкости (без увеличения площади ЗЭ), уменьшения емкости линий записи-считывания и применение усилителей регенераторов для считывания данных.

ЗУ повышенного быстродействия

Современные микропроцессоры характеризуются высоким быстродействием. Это требует и увеличения скорости работы ОЗУ, обменивающихся информацией с процессорами. Особенно остро эта задача стоит перед разработчиками динамических ОЗУ, которые благодаря максимальной информационной емкости и низкой стоимости занимают ведущее место в составе основной памяти компьютеров.

В последнее время предложен ряд вариантов динамических ОЗУ повышенного быстродействия. *Методы, использованные в этих ОЗУ, основаны на предположении о кучности адресов при обращениях к ОЗУ.* Это отвечает тенденции, проявляющейся при выполнении самых разных программ и состоящей в том, что адреса последующих обращений к ОЗУ вероятнее всего расположены рядом с адресом текущего обращения.

Вариант FPM. Вариант FPM (**F**ast **P**age **M**ode, быстрый страничный режим доступа) эффективен, если после обращения к некоторому ЗЭ следующее обращение будет к ЗЭ в той же строке. Сравним такую ситуацию с более общей.

При чтении по произвольному адресу старший полуадрес выбирает строку, затем младший полуадрес выбирает столбец в матрице ЗЭ. При этом сначала требуется перезарядить шину выборки строки, а затем шину выборки столбца, что сопровождается соответствующими задержками.

При обращении к строке (странице) во всех ЗЭ строки проходят процессы, соответствующие двум первым фазам полного цикла обмена, и эти элементы готовы к выполнению очередных фаз. При обращении к данным в пределах одной страницы адрес строки остается неизменным, изменяются только адреса столбцов. Изменяет состояние фактически только определенная группа ключей. Пока не изменился номер страницы, в циклах обмена исключены некоторые этапы, что сокращает длительность циклов.

Режим FPM – начало линии развития методов повышения быстродействия динамических ЗУ. По быстродействию его возможности уже намного превышены более поздними разработками, тем не менее метод FPM находит свою область применения, и соответствующие ЗУ до сих пор занимают достаточно большой сектор рынка.

Дополнительные средства для организации режима FPM просты: требуется лишь проверять принадлежность очередного адреса текущей странице (строке), что позволяет выполнять цикл страничного режима. В противном случае требуется выполнение обычного (полного) цикла. Разработанные ОЗУ типа FPM обеспечивают времена обращения к ЗУ 30…40 нс, что допускает их работу с процессорными шинами на тактовой частоте до 33 МГц.

Структуры типа EDORAM. Структуры типа EDORAM (**E**xtended **D**ata **O**ut RAM, т.е. ОЗУ с расширенным выводом данных) близки к структурам FPM и отличаются от них модификацией процесса вывода данных. В EDORAM данные в усилителях-регенераторах не сбрасываются. При этом на кристалле как бы появляется статический регистр, хранящий строку. При обращениях в пределах строки (страницы) используется чтение данных из регистра, т.е. быстродействующей статической памяти. Это увеличивает быстродействие ЗУ. Разработанные EDORAM допускают работу на частотах до 50 МГц. Такие ЗУ получили широкое распространение, в частности, из-за тесной преемственности с разработанными ранее ЗУ типа FPM, замена которых на EDORAM требует лишь небольших изменений в схеме и синхросигналах ЗУ.

Структуры типа BEDORAM. В структуре типа BEDORAM (**B**urst EDORAM, т.е. с пакетным расширенным доступом) содержится дополнительно счетчик адресов столбцов. При обращении к группе слов

(пакету) адрес столбца формируется обычным способом только в начале пакетного цикла. Для последующих передач адреса образуются быстро с помощью инкрементирования счетчика. Память типа BEDORAM не получила широкого распространения из-за появления сильного конкурента – синхронных DRAM (SDRAM).

Структуры типа MDRAM. В структурах MDRAM (**M**ultibank DRAM, многобанковые ОЗУ) память делится на части (банки). Обращение к банкам поочередное, чем исключается ожидание перезаряда шин. Пока считываются данные из одного банка, другие имеют "передышку" на подготовку, после которой появляется возможность обращения к ним без дополнительного ожидания. При нарушении очередности и повторном обращении к тому же банку выполняется полный цикл обращения к памяти. Чем больше банков, тем меньше будет повторных последовательных обращений в один и тот же банк. Поскольку процессор чаще всего считывает данные по последовательным адресам, то эффект ускорения работы ЗУ достигается уже при делении памяти всего на два блока, а именно на один с нечетными адресами, другой – с четными. Банки ЗУ типа MDRAM могут строиться на обычных DRAM без каких-либо схемных изменений.

Структуры типа SDRAM. Хотя переход от базовой структуры DRAM к архитектурам FPM и EDORAM повысил быстродействие памяти, этого оказалось недостаточно для современных компьютеров и графических систем. Память типа SDRAM (**S**ynchronous DRAM) заняла сейчас важное место в качестве быстродействующей памяти с высокой пропускной способностью. В SDRAM синхросигналы памяти тесно увязаны с тактовой частотой системы, в них используется конвейеризация тракта продвижения информации, может применяться многобанковая структура памяти и др. Синхронные DRAM появились в 1994 г. в виде двухбанковых систем с трехступенчатым конвейером и имели пропускную способность 250 Мбайт/с. Эти ЗУ работали на частоте 125 МГц при U_{cc} = 3.3 В и топологической норме 0.5 мкм. Причем площадь кристалла (113.7 мм^2) практически не отличалась от площади кристаллов обычных DRAM той же емкости.

До более подробного ознакомления с памятью типа SDRAM рассмотрим общий вопрос о конвейеризации трактов обработки информации. Сущность конвейеризации заключается в разбиении трактов обработки информации на ступени. На рис. 1.29, *а* показан тракт обработки данных, содержащий входной и выходной регистры и логическую схему между ними. Исходя из тезиса о возможности подачи новых входных данных только после окончания обработки старых,

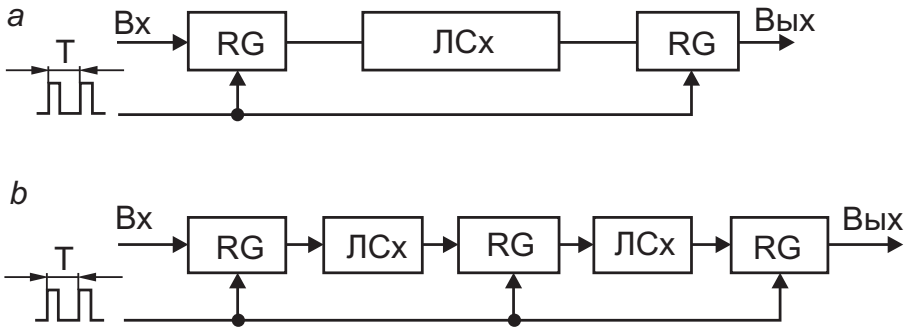


Рис. 1.29. Исходный (*a*) и конвейеризированный (*b*) тракты обработки цифровой информации

получим минимальный период тактовых импульсов для этой схемы:

$$T_{\min} = t_{\text{рг}} + t_{\text{кц}} + t_{\text{su}}, \qquad (1.2)$$

где $t_{\text{рг}}$ – задержка входного регистра на пути "такт-выход"; $t_{\text{кц}}$ – задержка сигнала в комбинационной цепи (логической схеме); t_{su} – время предустановки выходного регистра.

Уменьшения $T_{\min}$, т.е. повышения частоты тактовых импульсов, можно добиться снижением $t_{\text{кц}}$ путем расщепления логической схемы на ступени, разделенные регистрами (рис. 1.29,*b*). Если логическая схема расщепляется по глубине ровно пополам, то новое значение минимального периода тактовых импульсов определится тем же соотношением, что и для схемы, показанной на рис. 1.29,*a*, однако численное значение задержки логической схемы нужно будет уменьшить вдвое.

Применение конвейера увеличивает поток информации от входа к выходу за единицу времени, хотя, в то же время, единица информации проходит от входа к выходу за большее время, чем в схеме без конвейеризации.

В микросхемах SDRAM внешние управляющие сигналы фиксируются положительными фронтами тактовых импульсов и используются для генерации команд, управляющих процессами в ЗУ. Первое слово после формирования адреса появляется с запаздыванием на несколько тактов (Access Latency). Время доступа при этом "обычное", т.е. такое, каким бы оно было в стандартном ЗУ. Работу конвейера можно определить как параллельное функционирование последовательно активизируемых блоков. В микросхемах SDRAM предусматривают возможность регулировки запаздывания первого доступа в целях приспособления памяти к частотным требованиям системы и длины пакета, в котором слова читаются или записываются в каждом такте после единственной команды.

Структуры типа RDRAM. Микросхемы названы по имени фирмы разработчика – Rambus (RDRAM – **R**ambus DRAM). Они пред-

ставляют собою байт-последовательную память с очень высоким темпом передачи байтов. Основными новшествами архитектурного плана являются синхронизация обоими фронтами тактовых импульсов и специальный новый интерфейс Rambus Channel. Синхронизация принципиально сходна с применяемой в SDRAM.

В первой разработке при частоте тактовых импульсов 250 МГц получен темп 1 передачи байтов 500 МГц (2 нс/байт). В дальнейшем частота повысилась еще в 1.5…3 раза.

Интерфейс Rambus Channel имеет всего 13 сигнальных линий, что значительно меньше, чем у традиционных микросхем памяти. В интерфейсе нет специализированных адресных линий. Вместо обычной адресации по интерфейсу посылаются пакеты, включающие в себя команды и адреса. Вначале посылается пакет запросов, на который память отвечает пакетом подтверждения, после чего идет пакет данных. Из-за такого процесса первый доступ к данным оказывается сильно запаздывающим. В первой разработке запаздывание составляло 128 нс. Поэтому при чтении отдельных слов RDRAM совершенно неэффективна. Средняя частота передачи байтов зависит от длины пакета данных. При обмене пакетами по 256 байт средняя частота будет 400 МГц (к 2 нс добавляется 0.5 нс на байт), при пакетах по 64 байта – 250 МГц и т.д.

RDRAM идеально подходит для графических и мультимедийных приложений с типичным для них процессом – быстрой выдачей длинной последовательности слов для формирования изображения на экране или сходных с этим задач.

Структуры типа DRDRAM. Это близкий родственник RDRAM, называемый **D**irect RDRAM (DRDRAM). В этой разновидности архитектуры RDRAM преодолен такой фактор, как большое время запаздывания при первом доступе к данным. Естественно, это расширило область использования DRDRAM.

Сегодня в области быстродействующих DRAM доминируют синхронные (SDRAM). Для некомпьютерных применений, требующих больших емкостей памяти, эта ситуация может сохраниться на многие годы. В компьютерных схемах DRDRAM представляется сильной альтернативой. Имея времена первого доступа, такие же как у SDRAM, DRDRAM не деградируют по скорости при произвольных обращениях больше, чем обычные синхронные DRAM. Пропускная же способность у них продолжает увеличиваться. Уже имеются микросхемы DRDRAM с 16-разрядным интерфейсом (первоначальные варианты RDRAM имели 8-разрядные). При работе на тактовой частоте 400 МГц и схемотехнике DDR (**D**ouble **D**ata **R**ate), предусматривающей тактирование процессов обоими фронтами импульсов, такие

DRDRAM дают пропускную способность (Bandwidth) внутри пакета 1.6 Гбайт/с. Можно сказать, что в извечной гонке с процессорами ЗУ впервые из догоняющих стали опережающими, поскольку цифру 1.6 Гбайт/с сейчас вряд ли можно использовать в системах.

Структуры типа CDRAM. В структурах CDRAM (**C**ached DRAM, кэшированная DRAM) на одном кристалле с DRAM размещена статическая кэш-память (кэш первого уровня). При этом кэш обеспечивает быстрый обмен с процессором, если информация находится в кэше, а также быстрое обновление своего содержимого. Последняя возможность связана с тем, что размещение кэша на одном кристалле с DRAM делает связи между ними внутренними (реализуемыми внутри кристалла), а в этом случае разрядность шин может быть большой и обмен может производиться большими блоками данных. Например, в CDRAM фирмы Ramtron применена 2048-разрядная шина для обновления содержимого кэша. Как синоним обозначения CDRAM иногда используется обозначение EDRAM (**E**nhanced DRAM). Кэширование, как и всегда, эффективно при выполнении программ, для которых промахи относительно кэша достаточно редки.

1.4. Устройства обработки данных и управления

1.4.1. Устройство обработки данных

Устройство обработки данных выполняет арифметические и логические операции. К арифметическим операциям относят сложение, вычитание, умножение и деление. В предельном случае список арифметических операций может ограничиваться лишь операцией сложения. Логические операции выполняются поразрядно без учета переносов. Типичный список логических операций: конъюнкция (логическое ИЛИ), дизъюнкция (логическое И), инверсия (логическое отрицание), сложение по модулю два (исключающее ИЛИ) и операции сдвига.

Функциональная схема устройства обработки (рис. 1.30) состоит из АЛУ, аккумулятора, регистра временного хранения (Рег.вр.хр.), регистров общего назначения (РОН), регистра признаков и регистра сдвига. Рассмотрим их подробнее в порядке возрастания сложности.

Регистры аккумулятор и временного хранения

Регистр аккумулятор (накопитель) служит для размещения одного из операндов АЛУ. При выполнении операции в аккумулятор записывается результат операции. Это возможно в силу специальной схемотехники аккумулятора.

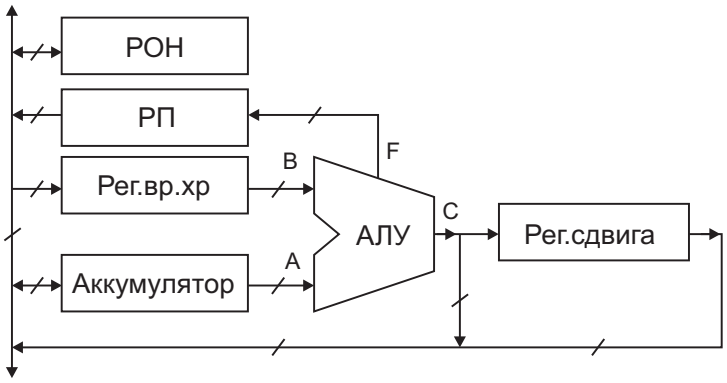


Рис. 1.30. Функциональная схема устройства обработки

Регистр временного хранения служит для размещения второго операнда АЛУ при выполнении бинарных (двухоперандовых) операций. По своей конструкции это обычный регистр-защелка.

Регистр признаков

Регистр признаков (флаговый регистр) предназначен для хранения битовых признаков (флагов), необходимых для работы АЛУ. Типичный состав хранимых признаков: `C` – перенос (заем), `Z` – нуль, `S` – знак числа, `P` – четность. В конкретном процессоре могут поддерживаться и другие признаки. Разрядность регистра признаков обычно совпадает с разрядностью обрабатываемых данных.

Регистры общего назначения

Регистры общего назначения представляют собой набор из нескольких регистров и служат для временного хранения промежуточных результатов и данных. РОН размещаются обычно на одном кристалле с процессором и доступны по короткому внутреннему адресу, минуя внешние шины адреса и данных. В этой связи время доступа к РОН гораздо короче такового для ячеек основной памяти. Отсюда другое название регистров общего назначения – сверхоперативное запоминающее устройство (СОЗУ).

Регистр сдвига

Регистр сдвига служит для аппаратного выполнения функций сдвига данных на один разряд в сторону старшего бита (сдвиг влево) или в сторону младшего бита (сдвиг влево).

Содержимое одного из крайних разрядов выдавливается из регистра сдвига наружу и во всех случаях перемещается в бит переноса `C` регистра признаков (рис. 1.31).

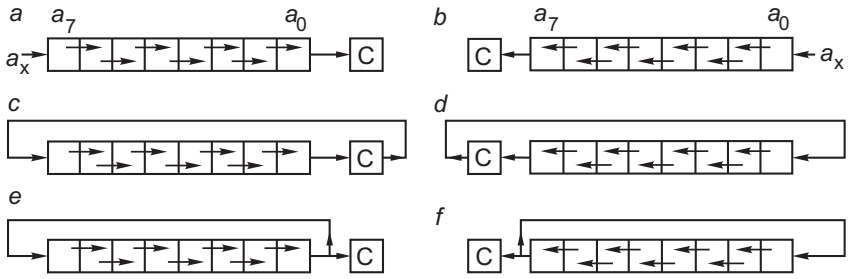


Рис. 1.31. Выполнение операций линейного (*a*, *b*) и циклического (*c* – *f*) сдвигов в регистре

Другой крайний разряд регистра сдвига заполняется битом данных a_x, поступающих извне. Операции сдвига классифицируют по источнику этого бита.

Линейные сдвиги (рис. 1.31, *a*, *b*) разделяют на арифметические и логические сдвиги.

При *линейном арифметическом сдвиге* бит $a_x = 0$. С точки зрения двоичной арифметики эти сдвиги соответствуют умножению (сдвиг влево) и делению (сдвиг вправо) на два.

При *линейном логическом сдвиге* не происходит поступления бита данных извне. Крайний разряд сохраняет свое значение и копируется в соседний разряд регистра сдвига. Говорят, что происходит размножение содержимого крайнего разряда. Через определенное количество сдвигов весь регистр будет заполнен содержимым крайнего бита.

При *циклических сдвигах* (рис. 1.31, *c* – *f*) крайние биты логически соединены между собой и данные при сдвиге перемещаются по кольцу. Бит переноса `C` при этом может входить в состав кольца (циклический сдвиг через бит переноса), либо оставаться вне кольца.

Арифметико-логическое устройство

Пример организации простейшего АЛУ приведен на рис. 1.32. На входах `A` и `B` АЛУ расположены электронные переключатели - мультиплексоры (MUX). Адресные линии MUX позволяют выбрать один из четырех источников данных: входные данные АЛУ, инверсное значение входных данных, арифметическая единица, нуль. Выходы мультиплексоров подключены одновременно к арифметическому сумматору и комбинационным логическим схемам. Управляющие линии (a_1a_0) позволяют настроить комбинационные схемы на выполнение одной из логических операций. Выходы сумматора и комбинационных логических схем коммутируются выходным мультиплексором, который позволяет передать на выход АЛУ результат арифметической либо логической операции. Признаки переноса или заема, возникающие в результате выполнения арифметических операций, передаются в регистр признаков.

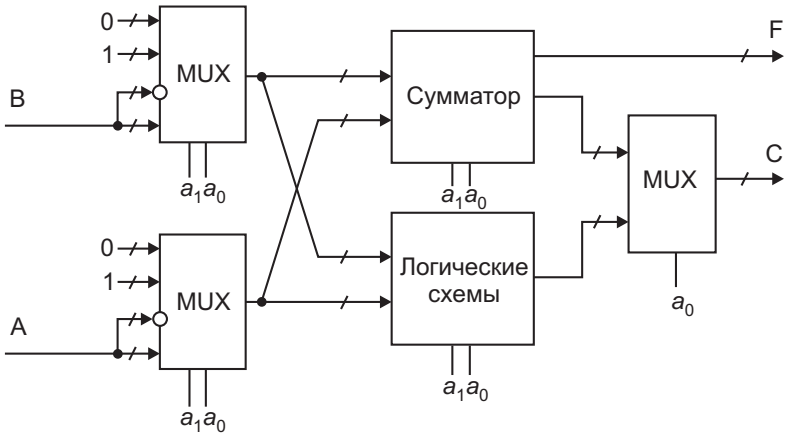


Рис. 1.32. Функциональная схема арифметико-логического устройства

Рассмотрим пример работы АЛУ при выполнении арифметической операции “5 минус 7”. Первый операнд (5) находится в регистре временного хранения (вход B), а второй (7) находится в аккумуляторе (вход A). Для рассматриваемой схемы АЛУ данная операция реализуется с помощью двух микроопераций:

1) *Получение дополнительного кода* числа (−7). Для этого MUX(A) подключен к инверсному входу канала A, MUX(B) – к единице, выход АЛУ подключен через выходной мультиплексор к сумматору. В результате выполнения микрооперации в аккумуляторе размещается дополнительный код числа минус 7: $(\overline{7} + 1)$;

2) *Сложение двух операндов.* Мультиплексоры MUX(A) и MUX(B) подключаются к прямым входам данных, выход АЛУ подключен к сумматору. В результате выполнения микрооперации в аккумуляторе получаем сумму числа 5 и дополнительного кода числа минус 7: $(5 + (\overline{7} + 1))$. Это и есть искомый результат.

Все настройки при выполнении микроопераций проводились с помощью управляющих линий типа (a_1a_0). Для относительно простых устройств количество управляющих линий может быть 20–30, а более сложные устройства могут иметь 70–100 и более линий управления. Все подобные линии управления обычно упорядочивают путем подключения к специальному регистру. Это позволяет сопоставить каждой допустимой микрооперации определенный управляющий код, загружаемый в данный регистр. Подобный управляющий код называют микрокомандой, а сам регистр – регистром микрокоманд. Управление электронными схемами при помощи микрокоманд называют микропрограммным управлением.

Принцип микропрограммного управления

- любая операция, реализуемая устройством, является последовательностью элементарных действий – микроопераций;

- для управления порядком следования микроопераций используются логические условия;
- процесс выполнения операций в устройстве описывается в форме алгоритма, представляемого в терминах микроопераций и логических условий, называемого микропрограммой;
- микропрограмма используется как форма представления функции устройства, на основе которой определяются структура и порядок функционирования устройства во времени.

1.4.2. Устройство управления

Устройство управления формирует распределенную во времени и пространстве последовательность внутренних и внешних управляющих сигналов, обеспечивающих выборку, декодирование и выполнение команд.

Два подхода к организации устройства управления: первичный управляющий автомат (ПУА) и микропрограммное устройство управления.

Устройство управления на базе ПУА

Микрокоманды хранятся в виде аппаратных связей первичного управляющего автомата, который тактируется тактовым генератором ТГ (рис. 1.33). Управляющий автомат вырабатывает адрес для выборки и помещает его в регистр адреса (счетчик команд) РС. Код

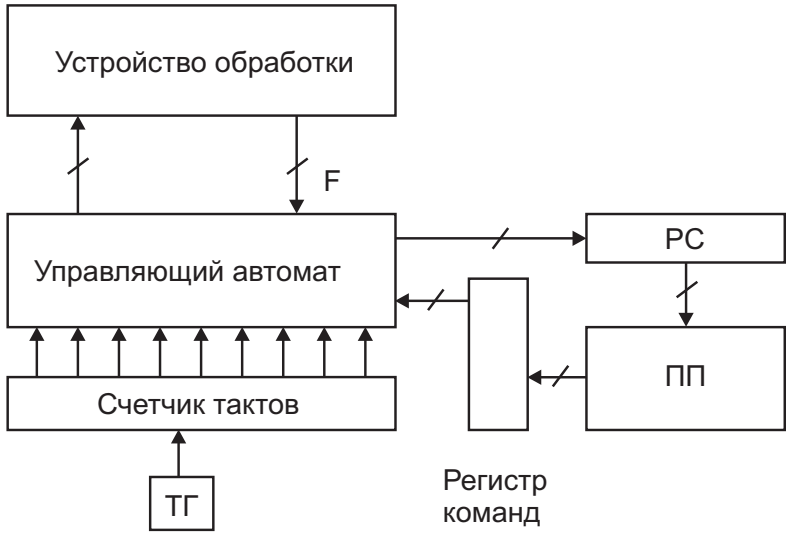


Рис. 1.33. Устройство управления на основе первичного управляющего автомата

операции, хранящийся в адресованной ячейке, выдается из памяти программ (ПП) в регистр команд. Далее происходит декодирование

кода операции, и управляющий автомат вырабатывает распределенную во времени и пространстве последовательность управляющих сигналов, являющихся аналогом микрокоманды. Управляющий код поступает в регистр микрокоманд устройства обработки, настраивая его на выполнение заданной микрооперации. Процесс “выборки - декодирования - выполнения” повторяется периодически, обеспечивая выполнение алгоритма программы.

Процесс выборки кодов операций может испытывать ветвление под действием того или иного признака. На схеме (см. рис. 1.33) показана обратная связь от устройства обработки через признаки `F`.

Микропрограммное устройство управления

В микропрограммном устройстве управления (рис. 1.34) последовательности микрокоманд, реализующие выполнение заданной опе-

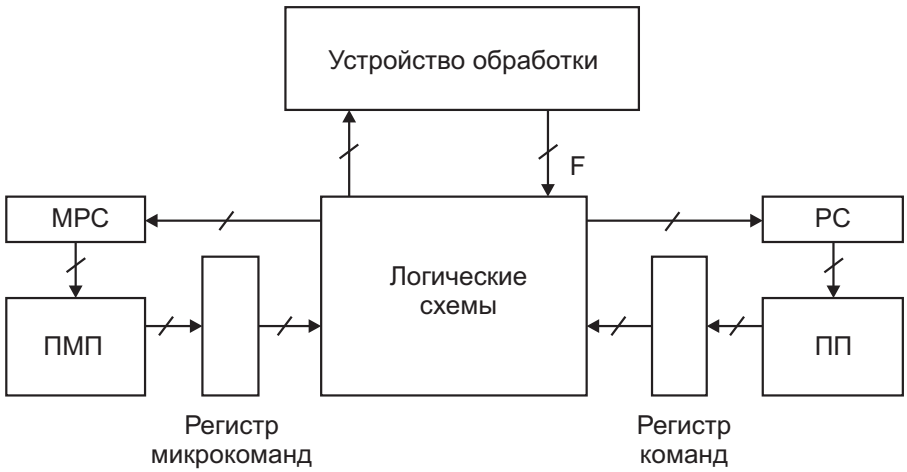


Рис. 1.34. Микропрограммное устройство управления

рации, хранятся в виде микропрограмм в специальном запоминающем устройстве – памяти микропрограмм (ПМП). Схема выборки из памяти программ похожа на предыдущий случай: логические схемы управления вырабатывают адрес для выборки кода операции, помещают этот адрес в регистр PC, код операции из ПП поступает в регистр команд.

В процессе декодирования логические схемы управления определяют стартовый адрес микропрограммы в памяти микропрограмм и помещают этот адрес в регистр адреса микрокоманд (MPC). Производится выборка микрокоманды из памяти микропрограмм в регистр микрокоманд и ее выдача в устройство обработки. Затем по порядку выбираются следующие микрокоманды, пока не будет выполнена вся микропрограмма, соответствующая заданной операции. Процесс выборки микрокоманд может испытывать ветвление в зависимости от признаков `F`, поступающих из устройства обработки.

1.4.3. Организация микрокоманд

Микрокоманда – это многоразрядное управляющее слово, обеспечивающее выполнение заданной микрооперации. Микрокоманды состоят из *полей* – групп разрядов, объединенных по функциональному назначению. На рис. 1.35 приведен пример структуры микрокоман-

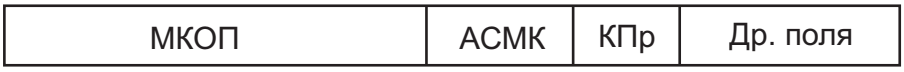


Рис. 1.35. Структура микрокоманды

ды. Для обозначения полей микрокоманды использованы типовые обозначения: `МКОП` – код микрооперации; `АСМК` – адрес следующей микрокоманды; `КПр` – код признаков; `Др. поля` – прочие поля микрокоманды.

Оптимизация микрокоманд позволяет ускорить обработку микрокоманд и снизить требования к системным ресурсам для их хранения. Рассмотрим основные пути оптимизации микрокоманд: определение совместимости во времени этапов выборки и выполнения микрокоманд; анализ способа формирования адреса следующей микрокоманды; выбор способа кодирования микрокоманд; выбор типа синхронизации при формировании микроопераций.

Оптимизация этапов выборки и выполнения

В последовательной схеме этапы выборки и выполнения (реализации) микрокоманд следуют друг за другом, занимая смежные временные промежутки (рис. 1.36, *а*). Выборка осуществляется устройством

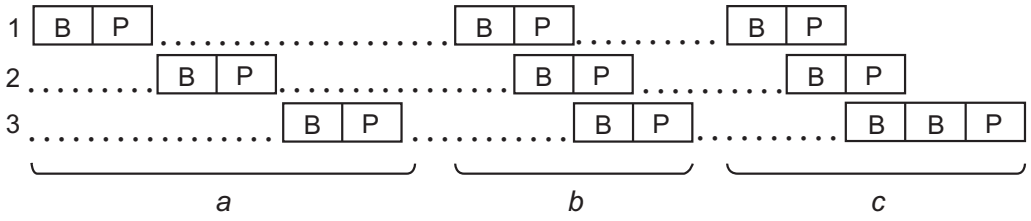


Рис. 1.36. Этапы выборки (В) и выполнения (реализации – Р) микрокоманды

управления, а выполнение – устройством обработки. На этапах выборки и выполнения работают разные функциональные узлы, поэтому иногда эти этапы возможно осуществлять параллельно: выполнение текущей и выборку следующей микрокоманд.

Параллельная схема (рис. 1.36, *b*) позволяет ускорить выполнение микропрограммы и, следовательно, команды за счет оптимального расположения во времени этапов выполнения текущей и выборки следующей микрокоманд. Недостаток такой схемы проявляется в точках

ветвления микропрограммы, когда заранее неизвестно, какую следующую микрокоманду необходимо выбирать из памяти микропрограмм. Это становится известным только после завершения выполнения текущей микрокоманды.

Параллельно-последовательная схема приведена на рис. 1.36, *c*. На линейных участках микропрограмма работает по параллельной схеме. В точках ветвления также производится упреждающая выборка следующей микрокоманды, но после выполнения текущей микрокоманды сначала производится проверка правильности сделанной выборки. Если выборка сделана правильно, то выбранная микрокоманда выполняется, а если выборка была ошибочной, то производится новая выборка. Далее, на линейных участках снова реализуется параллельная схема. Задержка во времени в точках ветвления носит статистический характер. Количество точек ветвления в микропрограмме заметно уступает количеству микрокоманд на линейных участках. В целом параллельно-последовательная схема по скорости выполнения микропрограмм может в пределе приближаться к параллельной схеме.

Оптимизация формирования АСМК

Поле АСМК содержит адрес следующей микрокоманды, т.е. микрокоманды, которую следует выбирать из памяти после выполнения текущей микрокоманды. Связь микрокоманд через поле АСМК определяет последовательность их выборки при выполнении микропрограммы. Такая система формирования адреса может работать и на линейных участках микропрограммы и в точках ветвления (рис. 1.37).

Рис. 1.37. Структура линейной (*a*) и управляющей (*b*) микрокоманд

Легко однако видеть, что на линейных участках микропрограммы такая система является избыточной, т.к. здесь вполне можно обойтись без поля АСМК. Действительно, если микрокоманды расположить в правильной последовательности друг за другом по возрастанию адресов ячеек памяти, то адрес следующей микрокоманды легко сформировать аппаратно. Для этого нужно инкрементировать (увеличить на

единицу) адрес текущей микрокоманды. Данный способ называют автоинкрементным способом формирования адреса. Исключение поля АСМК из микрокоманды при этом позволяет уменьшить длину микрокоманд на 8–10 разрядов.

Для микрокоманд условных переходов автоинкрементный способ формирования АСМК неприменим, т.к. возможны варианты АСМК в зависимости от результата предыдущей операции и содержимого поля КПр. Микрокоманды условных переходов должны иметь принудительную передачу управления по заданному в поле АСМК адресу. Это требует иной структуры микрокоманды для передачи управления, а именно – наличия поля АСМК.

На практике реализуют одновременно оба типа микрокоманд, отличающиеся структурой полей: без поля АСМК – для линейных микрокоманд, и с полем АСМК – для микрокоманд передачи управления. Для облегчения их идентификации при декодировании в структуру микрокоманды обычно вводят однобитовое поле (см. рис. 1.37), несущее информацию о типе микрокоманды: 0 – линейная, 1 – управляющая.

Кодирование микрокоманд

Микрокоманды характеризуются высокой разрядностью (30–40 для простых устройств и 70–100 для устройств среднего класса сложности). В то же время система микрокоманд содержит, как правило, только несколько сотен неповторяющихся кодов. Иными словами, микрокоманды имеют избыточную разрядность. Легко подсчитать, что при этом используется лишь одна миллиардная часть допустимых состояний микрокоманд. Это является основанием для их кодирования, т.е. уменьшения разрядности микрокоманд без потери содержащейся в них информации. Микрокоманды хранятся и пересылаются в закодированном виде, а перед выполнением подвергаются процессу декодирования. Для декодирования используется аппаратный дешифратор, преобразующий микрокоманду в исходное состояние.

Выбор способа кодирования микрокоманд определяется системой команд, структурой процессора, требованиями к быстродействию и стоимости и представляет собой очень сложную задачу. Ниже рассматриваются основные способы кодирования микрокоманд.

Горизонтальное кодирование. Простейшим вариантом кодирования микрокоманд является горизонтальное кодирование, при котором каждый разряд поля кода микрооперации однозначно определяет управляющий сигнал для выполнения микрооперации (см. рис. 1.35). Из-за большого набора микроопераций (от нескольких десятков до нескольких сотен) горизонтальное кодирование может потребовать

большой разрядности памяти микрокоманд, что и является основным его недостатком. Достоинством горизонтального кодирования является возможность параллельной работы нескольких устройств обработки, что позволяет существенно повысить быстродействие и приводит к высокой степени загрузки оборудования. Цель декодирования – преобразовать микрокоманду в горизонтальное представление.

Вертикальное кодирование. Другим, противоположным, подходом к кодированию микрокоманд в целях максимального сокращения разрядности полей микрокоманды является вертикальное кодирование (рис. 1.38). При этом способе микрокоманда кодируется как единое

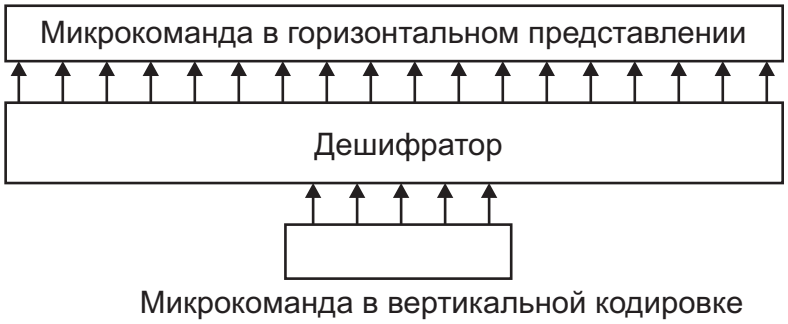


Рис. 1.38. Вертикальное кодирование микрокоманд

целое, без учета внутренней структуры. Для декодирования микрокоманд в вертикальном представлении требуется дополнительный дешифратор микроопераций, а кроме того, увеличивается время выполнения микрокоманды за счет временных задержек в дешифраторе и отсутствует возможность одновременного формирования нескольких микроопераций.

Смешанное кодирование. Развитием способов кодирования микрокоманд в целях устранения основных недостатков, присущих горизонтальному и вертикальному способам, является горизонтально-вертикальное, или смешанное, кодирование микрокоманд (рис. 1.39). При данном варианте кодирования в отдельных полях кода микроопераций объединяют взаимоисключающие наборы микроопераций для обеспечения их параллельного выполнения, как это имеет место при чисто горизонтальном кодировании. Микрокоманда кодируется с учетом ее внутренней структуры: любые поля или группы полей кодируются отдельно. Степень сжатия при этом способе кодирования меньше, чем для вертикального кодирования, но потери времени на этапе декодирования уменьшены, т.к. каждая группа имеет свой дешифратор – декодирование групп идет параллельно. Дешифраторы, декодирующие код микрооперации отдельных полей, образуют уровень

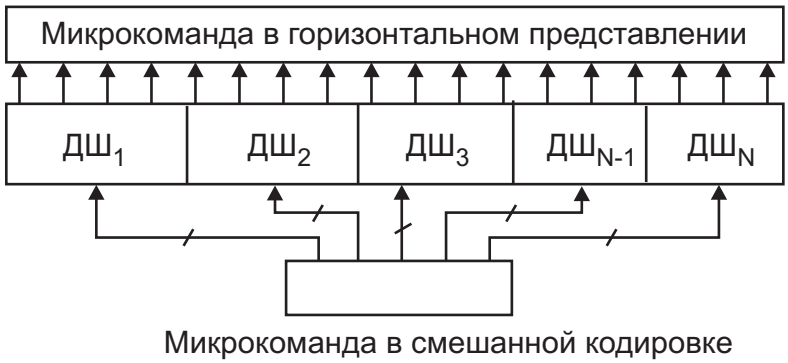


Рис. 1.39. Смешанное кодирование микрокоманд

схем дешифрации микроопераций, реализуемых в каждом поле в течение одного микрокомандного цикла. Данный способ кодирования находит широкое применение в микропрограммных устройствах.

Косвенное кодирование. В рассмотренных выше вариантах кодирования микрокоманд каждое поле МКОП формирует определенные фиксированные управляющие сигналы и интерпретируется всегда одинаковым образом. Однако в некоторых процессорах в целях дальнейшего сокращения разрядности микрокоманды одно и то же поле может быть использовано для формирования управляющих сигналов для различных блоков, и его функции в этом случае определяются другим полем. Переменные форматы сокращают разрядность микрокоманды, но требуют введения дополнительных дешифраторов, логических схем и приводят к временным задержкам из-за коммутации полей микрокоманды (рис. 1.40). Косвенное кодирование похоже на смешанное кодирование, в котором из микрокоманды изъяты группы разрядов, не несущих информации по текущей микрооперации.

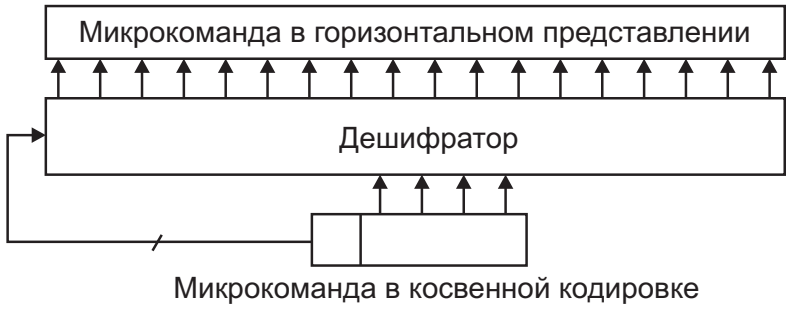


Рис. 1.40. Косвенное кодирование микрокоманд

Двухуровневое кодирование. В современных процессорах широко используется двухуровневая структура микрокоманд для интерпретации команды. Например, в процессоре фирмы Motorola первый уро-

вень микрокоманд использует вертикальное кодирование микрокоманд, выбирающих широкие горизонтальные микрокоманды второго уровня, называемые нанокомандами. Основой для использования данного принципа кодирования является то, что в случае, например, горизонтального кодирования часто для реализации микропрограмм используется небольшая часть комбинаций параллельно выполняемых микроопераций из максимально возможного их числа. Поэтому за счет хранения данных комбинаций в отдельной памяти нанокоманд небольшой емкости и разрядности, требуемой для горизонтального кодирования микрокоманды, можно достигнуть значительного сокращения общей емкости памяти с максимальным параллелизмом при выполнении микроопераций. На рис. 1.41 представлена структур-

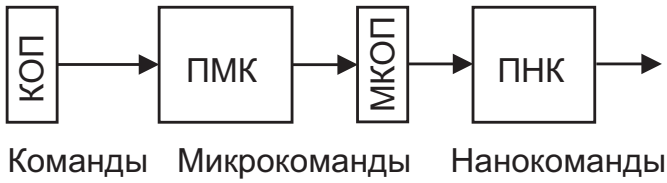


Рис. 1.41. Двухуровневое кодирование микрокоманд

ная схема устройства с двухуровневым кодированием микрокоманд, в котором на первом уровне используется вертикальное кодирование, а на втором – горизонтальное. Из памяти программ выбирается код операции (КОП), при декодировании определяется стартовый адрес микропрограммы в памяти микропрограмм (ПМК). Выполнение микрокоманд осуществляется следующим образом. Первая микрокоманда с вертикальным кодированием извлекается из памяти микрокоманд (ПМК), поле кода микрооперации МКОП является адресом нанокоманды из памяти нанокоманд (ПНК). Выбранная нанокоманда и определяет множество микроопераций, реализуемых в текущем микрокомандном цикле. Например, память для хранения микропрограмм объемом 4К микрокоманд по 32 бита при числе нанокоманд 256 может быть реализована как совокупность памяти микрокоманд емкостью 4К слов по 1 байту и памяти нанокоманд емкостью 256 слов по 32 бита. Это позволяет сократить емкость памяти примерно в 3 раза. Использование ПНК целесообразно в случае многократного повторения в микропрограммах микрокоманд.

Синхронизация микрокоманд

В основе синхронизации микрокоманды лежит количество тактирующих сигналов, необходимых для ее реализации. С этой точки зрения выделяют однотактные микрокоманды (рис. 1.42) и многотактные, для реализации которых требуется последовательность тактиру-

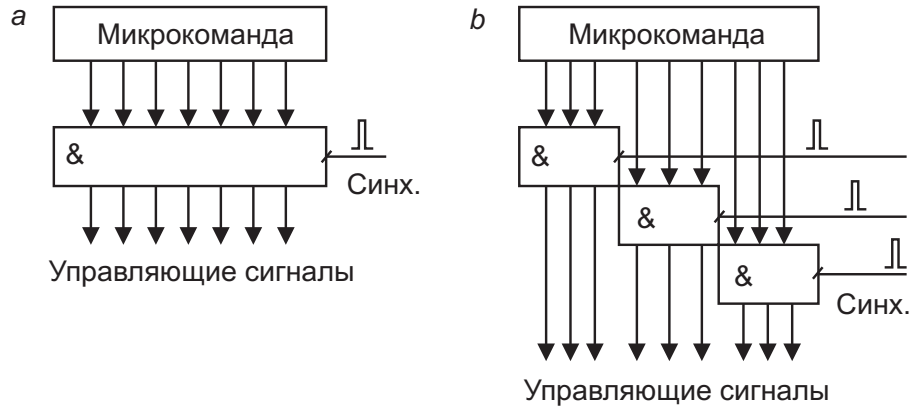


Рис. 1.42. Однотактная (*a*) и многотактная (*b*) синхронизация микрокоманд

ющих сигналов. При этом в микрокоманду может быть включен дополнительный разряд, который определяет тип синхронизации. Многотактная синхронизация позволяет минимизировать число микрокоманд в памяти, но требует большего объема оборудования для управления временными интервалами отдельных фаз сложной микрокоманды. Многотактная синхронизация упрощает параллельную выборку и выполнение микрокоманд, а также связи между источниками и приемниками информации при реализации микрокоманд. Достоинством же однотактной синхронизации является простота ее технической реализации.

1.5. Общие сведения о микропроцессорах

1.5.1. Общая характеристика

Первый процессор как программно функционирующее устройство, способное выполнять арифметические и логические операции, а также осуществлять ветвление алгоритма своего функционирования в зависимости от результата предыдущих вычислений, был создан в 40-е годы прошлого столетия в США специалистами фирмы IBM. Он представлял собой устройство на электромеханических реле, занимал несколько этажей здания, имел крайне низкое быстродействие и надежность и был пригоден лишь для очень узкого класса специфических вычислений. По мере прогресса электронной техники совершенствовалась и элементная база для построения процессоров. Появлялись процессоры на электронных лампах, транзисторах и дискретных логических микросхемах малой степени интеграции. По мере совершенствования процессоры имели все меньшие габаритные размеры, потребляли все меньше энергии, обладали все большей производительностью и надежностью. Однако они все еще были мало пригодны для выполнения операций управления в реальном масштабе време-

ни, а потому использовались в основном только для определенного класса вычислительных задач.

Настоящая революция в технике произошла после появления первого микропроцессора. Первый однокристальный микропроцессор был разработан фирмой Intel. Руководил группой разработчиков конструктор Federico Faggin. Микропроцессор `i4004` (Computer-on-chip) появился в 1971 году, он содержал около 2.3 тысяч транзисторов, был выполнен по технологии 10 мкм, имел производительность 60 тысяч операций в секунду и стоил $200. По своей производительности микропроцессор был сопоставим с электронно-вычислительной машиной `ENIAC`, содержащей около 85 тысяч электронных ламп и занимавшей объем 85 $м^3$.

Уменьшение стоимости, потребляемой мощности и габаритных размеров, повышение надежности и производительности микропроцессоров способствовали значительному расширению сферы их использования. Наряду с традиционными вычислительными системами они все чаще стали использоваться в задачах управления. При этом перед микропроцессором ставились задачи программного управления различными объектами в реальном масштабе времени.

Основными направлениями эволюции микропроцессоров являются увеличение разрядности одновременно производимых вычислений и уменьшение времени выполнения вычислений. В табл. 1.2 приведена выборочная хронология появления некоторых микропроцессоров фирмы Intel.

Возрастание числа транзисторов в кристалле микропроцессора подчиняется эмпирическому правилу (закону) Мура. В 1965 г. Гордон Мур (Gordon Moore) установил, что в среднем каждые 18 месяцев появляется новая микросхема памяти, которая содержит в 2 раза больше транзисторов. Масштабы этого роста видны из следующего сравнения. В 1978 году авиабилет по маршруту Нью-Йорк – Париж стоил $900, а перелет длился 7 часов. Если бы авиаиндустрия развивалась в соответствии с законом Мура, то сегодня авиабилет на тот же маршрут стоил бы менее цента, а перелет занял бы менее одной секунды.

1.5.2. Организация микропроцессора

Микропроцессор – это программно-управляемое устройство, предназначенное для обработки цифровой информации и управления процессом этой обработки, выполненное в виде одной или нескольких интегральных схем с высокой степенью интеграции электронных элементов.

Функции микропроцессора (МП):

- извлечение кода операции из памяти программ;

Таблица 1.2. Хронология появления некоторых микропроцессоров фирмы Intel

Наименование микропроцессоров	Год выпуска	Кол-во транзисторов, млн.шт	Разрядность	Тактовая частота, МГц	Технология, нм
i4004	1971	0.0023	4	0.74	10000
i4040	1972		4	0.74	10000
i8008	1972	0.0035	8	0.8	10000
i8080	1974	0.0050	8	2	6000
i8085	1976	0.0065	8	6	3000
i8086	1978	0.0290	16	10	
i8088	1979	0.0290	8/16	8	
i80186	1982		16	20	
i80286	1982	0.120	16	12	1500
i386	1985	0.275	32	33	1000
i486	1989	1.2	32	100	600
Pentium	1993	3.1	32	200	350
Pentium II	1997	7.5	32	350	250
Pentium III	1999	24	32	750	
Pentium IV	2000	42	32	2000	
Itanium	2001	220	64	800	
Itanium II	2003	410	64	1500	180
Itanium II*	2006	1700	64	2000	90

* С ядром `Montecito`.

- декодирование кода операции;
- выполнение декодированной команды;
- передача данных между устройствами системы;
- реагирование на внешние воздействия;
- установление общей синхронизации и управление системой.

Разрядность микропроцессора – количество бит информации, которое микропроцессор может обработать с помощью одной команды.

Разрядность микропроцессора определяется разрядностью его АЛУ, внутренних регистров данных и внешней шины данных. На сегодняшний день существуют 4-, 8-, 16-, 32- и 64-разрядные микропроцессоры. Для того чтобы обрабатывать информацию с разрядностью большей, чем разрядность микропроцессора, необходимо реализовывать специальные алгоритмы вычислений с повышенной разрядностью. Эти алгоритмы требуют дополнительного времени для своего выполнения. Поэтому повышение разрядности микропроцессора при заданной разрядности вычислений напрямую связано с увеличением быстродействия системы.

На рис. 1.43 приведена функциональная схема простейшего микропроцессора. Это минимальный набор узлов, реализующих все функ-

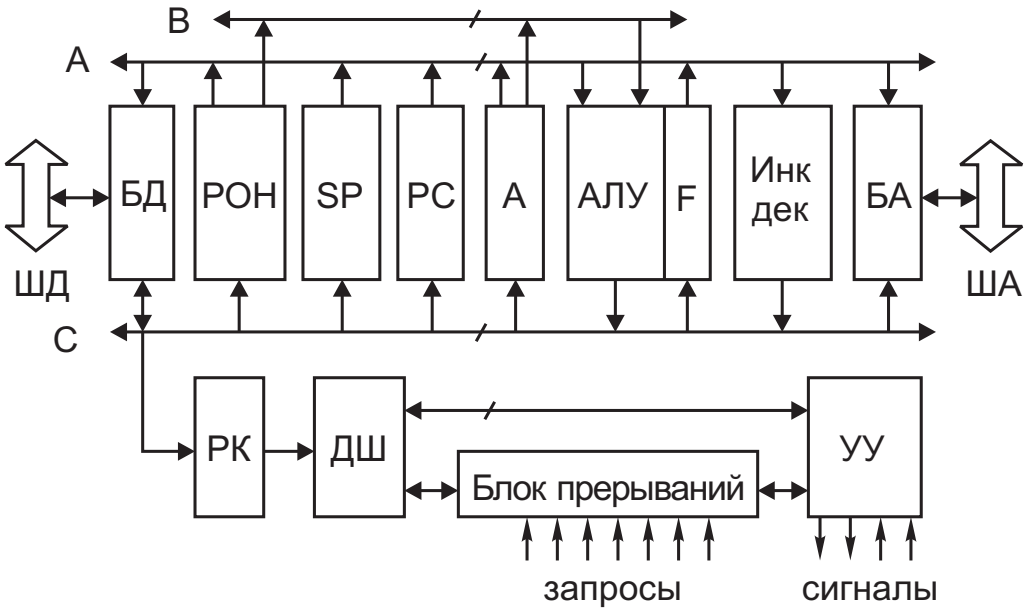


Рис. 1.43. Функциональная схема микропроцессора

ции микропроцессора. Объемными стрелками ША и ШД обозначены внешние шины адреса и данных. Внутренние шины данных обозначены буквами A, B, C. Взаимодействие с внешними шинами осуществляется через буферы адреса (БА) и данных (БД), которые содержат шинные формирователи. Регистровая структура представлена регистрами общего назначения (РОН), указателем стека (Stack Pointer – SP), счетчиком команд (Program Counter – PC), аккумулятором, регистром признаков (F), регистром команд (РК). Обозначения других узлов: АЛУ – арифметико-логическое устройство, Инк/дек – схема для реализации аппаратной операции инкремент/декремент, ДШ – дешифратор команд, Блок прерываний и УУ – устройство управления.

Блок прерываний служит для выполнения функции реагирования на внешние воздействия, его устройство и функционирование будут детально обсуждаться в п.2.3.

1.5.3. Классификация микропроцессоров

По числу больших интегральных схем (БИС) в микропроцессорном комплекте различают микропроцессоры однокристальные, многокристальные и многокристальные секционные.

Процессоры имеют сложную функциональную структуру, содержат большое количество электронных элементов и множество разветвленных связей. Для обоснования классификации микропроцессоров по числу БИС надо распределить все аппаратные блоки процессора

между основными тремя функциональными частями: обрабатывающей, управляющей и интерфейсной.

Однокристальные микропроцессоры – функционально законченные микропроцессоры с фиксированной архитектурой, разрядностью и системой команд. Микрокомандный уровень для пользователя недоступен, система команд является неизменяемой, она ориентирована на широкий круг задач. Однако универсальный характер системы команд подразумевает, что для каждой конкретной задачи система команд не является оптимальной. Однокристальные микропроцессоры имеют, как правило, универсальное назначение, они получаются при реализации всех аппаратных средств процессора в виде одной БИС или сверхбольшой БИС. По мере увеличения степени интеграции элементов в кристалле и числа выводов корпуса параметры однокристальных микропроцессоров улучшаются. Однако возможности однокристальных микропроцессоров ограничены аппаратными ресурсами кристалла и корпуса.

Многокристальный микропроцессор. Это сборка микропроцессора из нескольких БИС. Все характеристики – как для однокристальных МП. Функционально-законченные сам микропроцессор и его модули. Функциональная законченность БИС многокристального микропроцессора означает, что его части выполняют заранее определенные функции и могут работать автономно. Для получения многокристального микропроцессора необходимо провести разбиение его логической структуры на функционально законченные части и реализовать их в виде БИС. Граница разбиения на кристаллы – шина С. Закрытый микропрограммный уровень, фиксированы система команд и архитектура.

На рис. 1.44,*а* показано функциональное разбиение структуры процессора при создании многокристального микропроцессора (пунктирные линии), содержащего БИС обрабатывающего (ОУ), управляющего (УУ) и интерфейсного (ИУ) устройств.

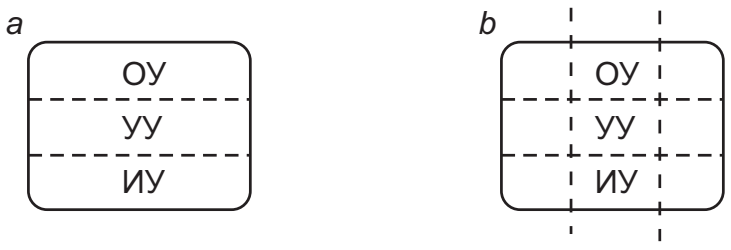


Рис. 1.44. Функциональная структура процессора (*а*) и ее разбиение для реализации процессора в виде комплекта секционных БИС (*b*)

Выбираемые из памяти команды распознаются и выполняются каждой частью микропроцессора автономно, и поэтому может быть обеспечен режим одновременной работы всех БИС МП, т.е. конвейерный поточный режим исполнения последовательности команд программы (выполнение последовательности с небольшим временным сдвигом). Такой режим работы значительно повышает производительность микропроцессора.

Многокристальные секционные микропроцессоры (микропроцессоры с разрядно-модульной организацией) получаются в том случае, когда в виде БИС реализуются части (секции) логической структуры процессора при функциональном разбиении ее вертикальными плоскостями (рис. 1.41,*b*). Это – многокристальный микропроцессор с открытым микропрограммным уровнем. Для построения многоразрядных микропроцессоров при параллельном включении секций БИС в них добавляются средства “стыковки”.

Для создания высокопроизводительных многоразрядных микропроцессоров требуется столь много аппаратных средств, не реализуемых в доступных БИС, что может возникнуть необходимость еще и в функциональном разбиении структуры микропроцессора горизонтальными плоскостями. В результате рассмотренного функционального разделения структуры микропроцессора на функционально и конструктивно законченные части создаются условия реализации каждой из них в виде БИС. Все они образуют комплект секционных БИС микропроцессора.

Таким образом, микропроцессорная секция – это БИС, предназначенная для обработки нескольких разрядов данных или выполнения определенных управляющих операций. Секционность БИС МП определяет возможность “наращивания” разрядности обрабатываемых данных или усложнения устройств управления микропроцессора при “параллельном” включении большего числа БИС.

Области применения секционных микропроцессоров:

- замена аппаратных решений программными: вместо схем комбинационной логики, работающих в реальном масштабе времени;
- эмуляция на микропрограммном уровне других микропроцессоров, имеющих богатое программное обеспечение;
- микропроцессоры без промежуточной трансляции с языков высокого уровня;
- параллельная обработка информации;
- микропроцессоры с повышенной разрядностью.

По назначению различают *универсальные* и *специализированные* микропроцессоры.

Универсальные микропроцессоры могут быть применены для решения широкого круга разнообразных задач. При этом их эффективная производительность слабо зависит от проблемной специфики решаемых задач. В системе команд МП заложена алгоритмическая универсальность, означающая, что выполняемый машиной состав команд позволяет получить преобразование информации в соответствии с любым заданным алгоритмом.

К универсальным МП относятся и секционные микропроцессоры, поскольку для них система команд может быть оптимизирована в каждом частном проекте создания секционного микропроцессора.

Специализированные микропроцессоры предназначены для решения определенного класса задач, а иногда только для решения одной конкретной задачи. Их существенными особенностями являются простота управления, компактность аппаратурных средств, низкая стоимость и малая мощность потребления. Специализированные МП имеют ориентацию на ускоренное выполнение определенных функций, что позволяет резко увеличить эффективную производительность при решении только определенных задач. Специализация МП, т.е. его проблемная ориентация на ускоренное выполнение определенных функций, позволяет резко увеличить эффективную производительность при решении только определенных задач.

Специализированные МП можно разделить на микроконтроллеры, ориентированные на выполнение сложных последовательностей логических операций, математические МП, предназначенные для повышения производительности при выполнении арифметических операций за счет, например, матричных методов их выполнения, МП для обработки данных в различных областях применений и т.д.

С помощью специализированных МП можно эффективно решать новые сложные задачи параллельной обработки данных. Например, интеграл свертки (конволюция) позволяет осуществить более сложную математическую обработку сигналов, чем широко используемые методы корреляции. Последние в основном сводятся к сравнению и определению подобия всего лишь для двух серий данных: входных, передаваемых формой сигнала, и фиксированных опорных. Конволюция дает возможность в реальном масштабе времени находить соответствие для сигналов изменяющейся формы путем сравнения их с различными эталонными сигналами, что, например, позволяет эффективно выделить полезный сигнал на фоне шума.

Разработанные однокристальные конвольверы используются в устройствах опознавания образов в тех случаях, когда возможности сбора данных превосходят способности системы обрабатывать эти данные.

По виду обрабатываемых входных сигналов различают цифровые и аналоговые микропроцессоры. Сами микропроцессоры – цифровые устройства, однако они могут иметь встроенные аналого-цифровые и цифро-аналоговые преобразователи. Поэтому входные аналоговые сигналы передаются в МП через преобразователь в цифровой форме, обрабатываются и после обратного преобразования в аналоговую форму поступают на выход. С архитектурной точки зрения такие микропроцессоры представляют собой аналоговые функциональные преобразователи сигналов и называются аналоговыми микропроцессорами. Они выполняют функции любой аналоговой схемы (например, производят генерацию колебаний, модуляцию, смещение, фильтрацию, кодирование и декодирование сигналов в реальном масштабе времени и т.д., заменяя сложные схемы, состоящие из операционных усилителей, катушек индуктивности, конденсаторов и т.д.). При этом применение аналогового микропроцессора значительно повышает точность обработки аналоговых сигналов и их воспроизводимость, а также расширяет функциональные возможности за счет программной "настройки" цифровой части микропроцессора на различные алгоритмы обработки сигналов.

Обычно в составе однокристальных аналоговых МП имеется несколько каналов аналого-цифрового и цифро-аналогового преобразования. В аналоговом микропроцессоре разрядность обрабатываемых данных достигает 24 бит и более, большое значение уделяется увеличению скорости выполнения арифметических операций.

Отличительная черта аналоговых микропроцессоров – способность к переработке большого объема числовых данных, т.е. к выполнению операций сложения и умножения с большой скоростью, при необходимости – даже за счет отказа от операций прерываний и переходов. Аналоговый сигнал, преобразованный в цифровую форму, обрабатывается в реальном масштабе времени и передается на выход обычно в аналоговой форме через цифро-аналоговый преобразователь. При этом согласно теореме Котельникова частота квантования аналогового сигнала должна вдвое превышать верхнюю частоту сигнала.

Сравнение цифровых микропроцессоров производится сопоставлением времени выполнения ими списков операций. Сравнение же аналоговых микропроцессоров производится по количеству эквивалентных звеньев аналого-цифровых фильтров рекурсивных фильтров второго порядка. Производительность аналогового микропроцессора определяется его способностью быстро выполнять операции умножения: чем быстрее осуществляется умножение, тем больше эквивалентное количество звеньев фильтра в аналоговом преобразователе и тем более сложный алгоритм преобразования цифровых сигналов можно задавать в микропроцессоре.

По характеру временной организации работы микропроцессоры делят на *синхронные* и *асинхронные*.

Синхронные микропроцессоры – микропроцессоры, в которых начало и конец выполнения операций задаются устройством управления (время выполнения операций в этом случае не зависит от вида выполняемых команд и величин операндов).

Асинхронные микропроцессоры позволяют начало выполнения каждой следующей операции определить по сигналу фактического окончания выполнения предыдущей операции. Для более эффективного использования каждого устройства микропроцессорной системы в состав асинхронно работающих устройств вводят электронные цепи, обеспечивающие автономное функционирование устройств. Закончив работу над какой-либо операцией, устройство вырабатывает сигнал запроса, означающий его готовность к выполнению следующей операции. При этом роль естественного распределителя работ принимает на себя память, которая в соответствии с заранее установленным приоритетом выполняет запросы остальных устройств по обеспечению их командной информацией и данными.

По организации структуры микропроцессорных систем различают *одно-* и *многомагистральные* МП.

В одномагистральных МП все устройства имеют одинаковый интерфейс и подключены к единой информационной магистрали, по которой передаются коды данных, адресов и управляющих сигналов.

В многомагистральных МП устройства группами подключаются к своей информационной магистрали. Это позволяет осуществить одновременную передачу информационных сигналов по нескольким (или всем) магистралям. Такая организация систем усложняет их конструкцию, однако увеличивает производительность.

По количеству выполняемых программ различают *одно-* и *мультипрограммные* микропроцессоры.

В однопрограммных микропроцессорах выполняется только одна программа. Переход к выполнению другой программы происходит после завершения текущей программы.

В мультипрограммных микропроцессорах одновременно выполняется несколько (обычно несколько десятков) программ. Организация

мультипрограммной работы микропроцессорных управляющих систем позволяет осуществить контроль за состоянием и управлением большим числом источников или приемников информации.

1.6. Программное обеспечение микропроцессоров

1.6.1. Языки и уровни программирования

Цифровые устройства функционируют под воздействием внутренних и внешних управляющих сигналов. Все возможные алгоритмы работы определяются совокупностью аппаратных связей. Изменение алгоритма работы аппаратной схемы требует ее модификации на аппаратном уровне.

В программно-управляемых устройствах взаимодействие с аппаратным ядром опосредовано программными надстройками различного уровня (рис. 1.45). Микропрограммный уровень является наиболее

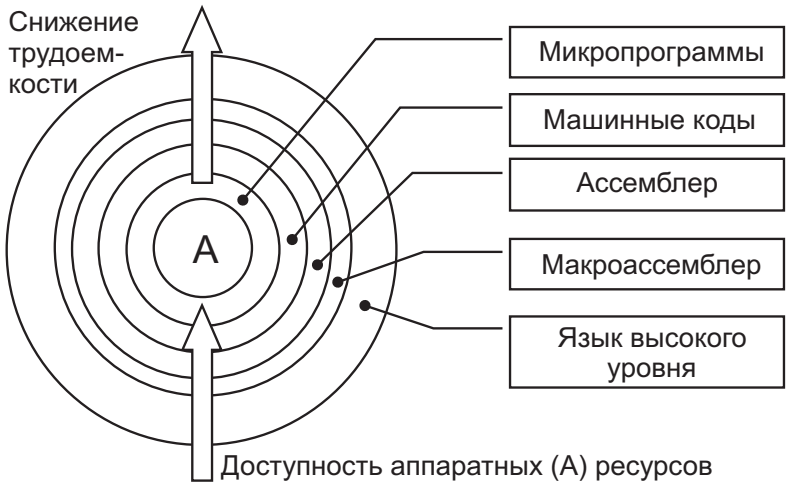


Рис. 1.45. Языки и уровни программирования

близким к аппаратному ядру. Каждый разряд поля кода микрокоманды однозначно определяет управляющий сигнал для выполнения соответствующей микрооперации. В большинстве микропроцессоров микрокомандный уровень не доступен для пользователя.

Далее идет уровень машинных команд (машинных кодов). Именно этот уровень является в большинстве случаев самым низким уровнем, доступным для пользователя. Каждая машинная команда поддержана соответствующей микропрограммой. Система команд микропроцессора определяет совокупность функций аппаратного ядра, доступных с уровня машинных команд.

С точки зрения пользователя машинные команды – это просто двоичные коды. Разрядность машинных команд обычно существенно ниже разрядности микрокоманд, но трудоемкость программирования

в машинных кодах остается чрезвычайно высокой. В некоторых случаях для снижения трудоемкости составления программы в машинных кодах применяют системы счисления с основанием 8 или 16. Перевод написанной программы в двоичные коды осуществляется в дальнейшем аппаратными средствами.

Дальнейшее снижение трудоемкости написания программ достигается на уровне языка ассемблера. В этом случае каждой машинной команде сопоставляют команду на языке ассемблера. Эти команды представляют собой буквенные сочетания, составленные обычно из аббревиатур английских глаголов, соответствующих функции команды. Например, типичная команда ассемблера `MOV` получена от английского глагола “to move” – перемещать. Трансляцию программы, написанной на языке ассемблера, в машинные коды осуществляют при помощи специальной программы под названием ассемблер. Иногда такой процесс называют ассемблированием программы.

Язык ассемблера – это входной язык программы ассемблера. Система команд языка ассемблера полностью идентична системе машинных команд. Для микропроцессоров с различной системой машинных команд языки ассемблера будут также различаться, поэтому язык ассемблера считается машинно-зависимым языком программирования.

На уровне макроассемблера составление программы осуществляется в терминах макрокоманд (макросов), реализующих укрупненные действия – макрооперации. В подготовленной программе макроассемблер сначала заменяет макрокоманды на заранее подготовленные фрагменты текста на языке ассемблера, а потом производит ассемблирование программы.

Наиболее удаленный от аппаратного ядра уровень программирования представлен языками высокого уровня. Некоторые из них считаются машинно-независимыми языками. Однако многие языки высокого уровня (например, язык С) предоставляют пользователю возможность делать в программе вставки фрагментов на языке ассемблера.

Отметим две противоположные тенденции, показанные на рис. 1.45 объемными стрелками: снижение трудоемкости программирования при удалении от аппаратного ядра и увеличение доступности аппаратных ресурсов при переходе к более низким уровням программирования. Выбор уровня программирования для конкретной задачи определяется балансом между этими противоположными тенденциями.

1.6.2. Подготовка программного обеспечения

Входным языком микропроцессора в большинстве случаев является уровень машинных команд. Любую программу, созданную на другом, более высоком уровне, перед выполнением необходимо транс-

лировать на уровень машинных команд. Это подразумевает использование служебных программ и определенной технологии подготовки программного обеспечения.

На рис. 1.46 показана подготовка программы на языке ассемблера и этапы ее трансляции до уровня машинных команд.

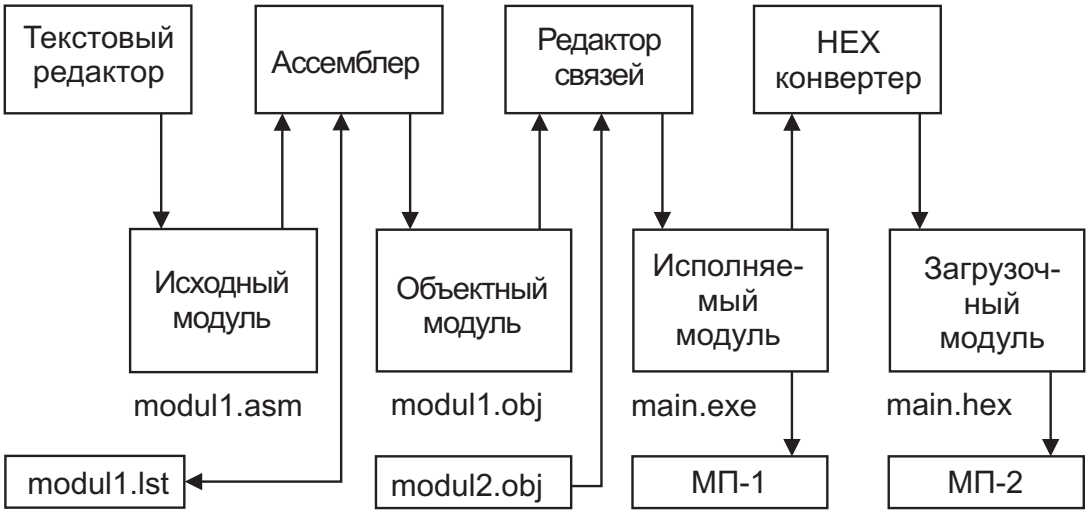


Рис. 1.46. Этапы подготовки программного обеспечения

Текстовый редактор. Исходный текст программы на языке ассемблера готовится в любом текстовом редакторе, работающем в ASCII кодах. Подготовленный текст программы сохраняется в виде файла на машинном носителе. Такой файл называется исходным модулем. Исходный модуль можно при необходимости редактировать текстовым редактором.

Ассемблер. Программа ассемблер позволяет использовать метки, символическую адресацию, форматные преобразования, распределение памяти, генерацию данных и выполнение арифметических и логических операций с константами на этапе ассемблирования. Большинство ассемблеров являются двухпроходными, т.к. для получения правильного объектного кода они осуществляют два просмотра исходного модуля. Во время первого прохода ассемблер создает таблицу символов и собирает все имена, определяемые в программе. Таблица записывается в файл `modul1.lst` (см. рис. 1.46). Во время второго прохода он транслирует программу, используя информацию, собранную при первом проходе. В общем случае символические имена могут быть определены в любом месте программы, поскольку ассемблер просматривает всю программу.

Обычно ассемблер выдает на выходе все или некоторые из перечисленных документов:

- объектный файл `modul1.obj`;
- листинг программы вместе с машинными кодами;
- список ошибок ассемблирования;
- таблицу символических имен, используемых в программе, с указанием их значений;
- таблицу перекрестных ссылок, содержащую перечень имен, и перечень всех команд, в которых они используются;
- список внешних ссылок (перечень имен подпрограмм или переменных, определенных за пределами данного исходного модуля).

Ассемблеры классифицируются как абсолютные и перемещающие в зависимости от того, формируют они информацию, позволяющую загружать программу в любую область памяти, или нет. Первый адрес, который занимает программа в памяти, называется адрес загрузки. В абсолютном ассемблере адрес загрузки известен во время ассемблирования. Поэтому при построении адресных констант и команд передачи управления ассемблер использует абсолютные адреса памяти. Далее обсуждается процесс подготовки программного обеспечения с использованием абсолютного ассемблера.

Результатом работы программы ассемблера является объектный модуль, записанный в файл `modul1.obj` (см. рис. 1.46). Термин *объектный модуль* подчеркивает тот факт, что программа может состоять из нескольких частей (объектных модулей), допускающих автономную трансляцию. На рис. 1.46 показаны два объектных модуля: `modul1.obj` и `modul2.obj`. В объектных модулях могут содержаться внешние ссылки к именам, определенным в других модулях. Все внешние обращения оформляются таблицей в самом объектном модуле.

Редактор связей. Функция редактора связей (Linker) заключается в компоновке одного загрузочного модуля из нескольких объектных модулей. В этой связи редактор связей часто называют *компоновщиком*. Необходимость выполнения компоновки объясняется внешними ссылками, которые не могли быть обработаны при ассемблировании отдельных модулей. Редактор связей обрабатывает все объектные модули программы и строит собственные таблицы внешних и входных имен. При этом могут быть обнаружены ошибки, связанные с неопределенными и многократно определенными внешними именами. Все адреса, зависящие от значений внешних имен, соответствующим образом корректируются.

Редактор связей формирует на выходе исполняемый модуль, который может быть размещен в памяти микропроцессорной системы `МП-1` (см. рис. 1.46) и запущен на выполнение.

HEX-конвертер. Во многих случаях разработка программы и ее выполнение происходят в разных микропроцессорных системах. Для транспортировки программы по различным каналам связи в другую микропроцессорную систему (например, МП-2, рис. 1.46) формируют загрузочный модуль в общепринятом гексадецимальном (hex) формате. Этот формат получают при помощи HEX-конвертера.

Рассмотрим формат hex-файла, разработанный фирмой Intel. Это строчно-ориентированный формат, в котором используются только печатные символы ASCII. Исключение составляют лишь символы возврата каретки и перевода строки. Каждая строка в файле имеет вид

```
:NNAAAARRHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHCCTT
```

Символы на этой схеме обозначают:

: – символ двоеточия помечает начало записи в строке;
NN – число в шестнадцатеричном формате, соответствующее количеству байтов в записи (строке). Длина строки не может превышать 255 байтов;
AAAA – четыре шестнадцатеричных цифры, соответствующие адресу местоположения первого байта в памяти программ;
RR – тип записи (равно 00 для данных, 01 – конец файла, 03 – адрес загрузки);
H – все поля, помеченные этим знаком, состоят из шестнадцатеричных цифр (0–9,ABCDEF), соответствующих байтам данных;
CC – контрольная сумма в шестнадцатеричном формате;
TT – признак конца строки (возврат каретки, перевод строки).

Последней строкой файла будет запись, согласующаяся с приведенным выше форматом, в которой значение счетчика команд будет нулевым :00000001FF. Контрольная сумма определяется следующим образом:

```
сумма =  NN + (ст. байт адреса) + мл. байт адреса +
       + RR + (сумма всех байтов данных);

контрольная сумма =  ((-сумма) & 0FFH).
```

Здесь (–сумма) имеет смысл отрицательного числа в дополнительном коде. Операция логического умножения усекает контрольную сумму до одного байта. В качестве примера проанализируем следующую запись:

```
:04004800929501305C
```

Здесь число байтов данных NN=04H, начальный адрес записи в памяти программ AAAA=0048H, тип записи RR=00H (данные). Далее идут четыре байта кода программы (байты данных записи): 92H, 95H, 01H, 30H. Последний байт записи 5CH представляет собой контрольную сумму. Проверим это. По определению

сумма = 04H + 00H + 48H +00H + 92H + 95H + 01H + 30H = 01A4H.

В дополнительном коде (−сумма) = ($\overline{01A4H}$ + 1) = 0FE5CH. Контрольная сумма будет 0FE5CH & 0FFH = 5CH.

Полученный файл main.hex передают по каналам связи в другую микропроцессорную систему МП−2. В этой системе специальная программа *резидентный загрузчик* преобразует загрузочный модуль в исполняемый модуль и размещает его в памяти в соответствии с заданным загрузочным адресом.

1.6.3. Виды программного обеспечения

Выделяют *резидентное* и *внешнее* программное обеспечение для микропроцессоров.

В резидентное программное обеспечение входят программы, размещенные в ресурсах той микропроцессорной системы, в которой они выполняются. Под термином *ресурс* здесь подразумевают любые виды запоминающих устройств, входящих в состав системы. Управляющие микропроцессорные системы обычно имеют весьма скромные ресурсы и соответственно небогатое резидентное программное обеспечение:

- программа МОНИТОР служит для взаимодействия микропроцессорного устройства с внешней средой. Эта программа позволяет выполнять простейшие действия по загрузке программного обеспечения, запуску программ на выполнение, проверке, тестированию и отладке системы;
- прикладные программы пользователя обеспечивают функционирование микропроцессорного устройства в соответствии с заданным алгоритмом;
- программы, поддерживающие процесс разработки на базовом языке программирования.

Специализированные микропроцессорные устройства могут иметь ресурсы, достаточные для хранения лишь прикладных программ. Другие виды резидентного программного обеспечения в них могут отсутствовать.

Внешнее программное обеспечение размещается вне ресурсов данной микропроцессорной системы, как правило, на дополнительных

носителях или в ресурсах другой, более мощной системы. Типичный состав внешнего программного обеспечения:

- операционная система, которая может рассматриваться как дальнейшее развитие программы Монитор;
- поддержка программных средств разработки на языках высокого уровня;
- поддержка средств разработки на базовом языке;
- прикладные программы пользователя.

Для разработки программного обеспечения используют *системы развития*. Полный комплект программного обеспечения в таких системах является резидентным. Другое название этих систем – *инструментальные системы*. Системы развития функционируют на базе мощных микропроцессорных устройств, включают в себя все имеющееся программное обеспечение и некоторые программно-аппаратные средства для отладки.

Программное обеспечение развития может включать в себя текстовый редактор, трансляторы (компиляторы) с языков высокого уровня, ассемблер, редактор связей, программный симулятор, отладчик, профилировщик, трассировщик и др. Более подробно это обсуждается в гл. 8.

Если программное обеспечение развития функционирует в микропроцессорной системе, отличающейся по платформе от той системы, для которой разрабатываются прикладные программы, то к характеристике системы развития добавляют приставку *кросс*: кросс-системы, кросс-ассемблер, кросс-программное обеспечение и т.д.

1.6.4. Представление данных в микропроцессоре

Система счисления – это код, в котором использованы специальные символы для обозначения количества каких-либо объектов. Количество символов в системе счисления называется основанием системы счисления (P). Разряды целой части нумеруются в порядке возрастания справа налево начиная с нуля. Весовой коэффициент разряда в общем случае определяется как P^n, где n – порядковый номер разряда.

Двоичные целые без знака. Числа представляются в прямом двоичном коде. Ниже приведены примеры одно- и многобайтных чисел, а также указаны диапазоны их изменения. Значение числа D определяется как сумма значений всех разрядов, умноженных на соответствующие весовые коэффициенты.

$$D = \sum_{i=0}^{N-1} a_i \cdot 2^i. \qquad (1.3)$$

Многобайтные числа обычно размещают в памяти подряд по возрастанию адресов: “младший байт – по младшему адресу, старший байт – по старшему адресу”.

Однобайтные целые без знака (0.....255)

a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

Многобайтные целые без знака (0.....2^N-1)

a_{N-1}	a_{N-2}	a_{N-3}	a_{N-4}	a_{N-5}	a_{N-6}	a_{N-7}	a_{N-8}	.....	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

Старший байт — Младший байт

Двоичные целые со знаком в прямом коде. Старший бит (S) содержит информацию о знаке числа: 0 – положительное и 1 – отрицательное числа. Остальные $N-1$ битов служат для размещения модуля числа в прямом двоичном коде. Отметим, что нуль может быть представлен двумя способами: -0 и $+0$.

Однобайтные целые со знаком (-127.....0.....+127)

S	a_6	a_5	a_4	a_3	a_2	a_1	a_0

Многобайтные целые без знака (-2^{N-1}-1.....0.....+2^{N-1}-1)

S	a_{N-2}	a_{N-3}	a_{N-4}	a_{N-5}	a_{N-6}	a_{N-7}	a_{N-8}	.....	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

Старший байт — Младший байт

Двоичные целые со знаком в дополнительном коде. Дополнительный код имеет фиксированную разрядность. Старший разряд содержит информацию о знаке числа: 0 – положительное и 1 – отрицательное число. Для положительных чисел остальные биты дополнительного кода совпадают с прямым двоичным кодом этих чисел. Для отрицательных чисел остальные биты дополнительного кода получают путем прибавления единицы к модулю числа, представленного в обратном двоичном коде.

Информация о знаке в старшем разряде при этом получается автоматически. Все переносы из старшего разряда игнорируют. Применение дополнительного кода в цифровых устройствах позволяет заменить операцию вычитания суммированием чисел, представленных в дополнительном коде.

Диапазоны чисел составляют $-128\ldots0\ldots+127$ для однобайтных чисел и $-2^{N-1}\ldots0\ldots+(2^{N-1}-1)$ для многобайтных N-битных чисел.

Числа в двоично-десятичном коде. Для представления десятичных чисел в двоичном коде используют двоично-десятичный код (ДДК). Каждую десятичную цифру 0…9 кодируют четырьмя битами (тетрадой) прямого двоичного кода. При байтовой организации системы различают два вида такой кодировки: распакованный ДДК и упакованный ДДК. Поскольку в тетраде шесть оставшихся старших состо-

Распакованный ДДК (0....9)

0	0	0	0	a_3	a_2	a_1	a_0

Упакованный ДДК (0.....99)

b_3	b_2	b_1	b_0	a_3	a_2	a_1	a_0

яний (A, B, C, D, E, F) не используются, то при выполнении арифметических операций необходима коррекция результата. Алгоритм двоично-десятичной коррекции зависит от вида арифметической операции.

Многобайтные числа в ДДК размещают в памяти по возрастанию адресов: “по младшим адресам – младшие байты, а по старшим адресам – старшие байты”. Числа со знаком в двоично-десятичном коде представляют следующим образом: знак числа размещают в старшей тетраде, а остальные тетрады содержат модуль числа в ДДК.

Дробные числа с фиксированной точкой. Точка разделяет целую и дробную части. Для целых чисел точка не указывается, но подразумевается. Разряды дробной части нумеруются слева направо от двоичной точки. К номеру добавляется знак минус. Первый разряд дробной части будет -1, потом -2 и т.д. При такой нумерации сохраняется единое определение весового коэффициента разряда: P^n. В примере указано подразумеваемое положение двоичной точки, разделяющей два разряда целой части и шесть разрядов дробной части числа D.

$$D = \sum_{i=-6}^{1} a_i \cdot 2^i. \qquad (1.4)$$

Диапазон чисел при этом составляет от 0 до 3 (целая часть) и от 0 до $(1-2^{-6})$ – дробная часть. В общем случае максимальное значение модуля дробной части всегда остается меньшим единицы на величину веса самого младшего (правого) разряда дробной части.

Число с фиксированной точкой

a_1	a_0	a_{-1}	a_{-2}	a_{-3}	a_{-4}	a_{-5}	a_{-6}

^

Основной вопрос при использовании этого формата – выбор оптимального положения фиксированной двоичной точки. Для каждого

числового диапазона есть свой наиболее оптимальный выбор. При работе с числами, принадлежащими различным диапазонам, неизбежен компромисс при определении местоположения фиксированной точки. Следствием этого может быть потеря точности, для компенсации которой потребуется увеличить разрядность чисел. Это недостаток формата чисел с фиксированной точкой.

Числа с плавающей точкой. Во многих микропроцессорных системах в качестве стандарта принят формат чисел с плавающей точкой IEEE-754. Принципиальным различием между типами чисел с плавающей точкой является точность представления, диапазон представления и требуемая для их хранения память. Числа этого формата представляют в нормализованной форме в виде двух компонентов: *мантиссы* (M) и *порядка* (P). Вместо слова *порядок* часто используют термин *экспонента*. Оба компонента могут быть одно- или многобайтными. Порядок (экспоненту) представляют в виде двоичного целого со знаком в прямом коде, а мантиссу – в виде модуля нормализованного дробного числа в прямом коде с фиксированной точкой: $1/2 \le M < 1$. Однобитовый знак мантиссы S кодируют обычным образом: 0 – положительное, 1 – отрицательное число.

$$D = (-1)^S \cdot M \cdot 2^P. \tag{1.5}$$

Ограничение диапазона мантиссы продиктовано требованием однозначности представления любого числа. Действительно, если диапазон мантиссы не ограничен, то, например, единицу можно представить в виде различных наборов мантиссы и порядка ($\mathrm{M} = 2^{-n}$, $\mathrm{P} = n$): (1, 0), (1/2, 1), (1/4, 2), (1/8, 3) и т.д.

Для однобайтных мантиссы и порядка диапазон изменения модуля представляемых чисел будет от 2^{-128} до $(1 - 2^{-8}) \cdot 2^{+127}$. Отметим, что ограничение диапазона мантиссы порождает проблему т.н. *машинного нуля*. В приведенном примере наименьшим по модулю числом (т.е. машинным нулем) будет 2^{-128}.

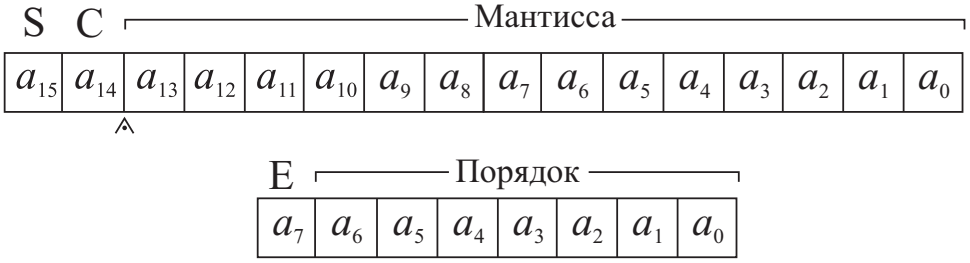
Форматы чисел микропроцессорной системы MC2702. Реализация любого формата представления чисел требует соответствующей программной поддержки, т.е. набора программ, выполняющих математические и иные операции над числами в данном формате. Обычно такой набор оформляют в виде библиотеки подпрограмм. В качестве примера приведем краткое описание форматов чисел из библиотеки подпрограмм промышленной микропроцессорной системы `MC2702`.

В библиотеке размещено 20 подпрограмм для выполнения четырех арифметических действий со знаком и для вычисления элементарных функций (тригонометрические функции, квадратный корень,

факториал, экспоненциальная функция, показательная функция, логарифмическая функция). Библиотека поддерживает два формата целых чисел и один формат чисел с плавающей точкой.

Двоичные целые числа со знаком в прямом коде: однобайтные (−127…+127) и двухбайтные (−16383…+16383). В двухбайтных числах модуль занимает младших 14 битов (a_0–a_{13}), бит a_{14} отведен под флаг переноса (C), а бит a_{15} содержит информацию о знаке числа (S).

Каждое число с плавающей точкой занимает в памяти три байта: двухбайтная мантисса и однобайтный порядок. Форматы мантиссы и порядка соответствуют структуре двухбайтных и однобайтных целых чисел данной библиотеки. Двоичная точка в мантиссе подразумевается между битами a_{13} и a_{14}. На схеме обозначены: S – знак мантиссы,



C – флаг переноса, E – знак порядка. Значение числа D вычисляют по формуле

$$D = (-1)^S \cdot M \cdot 2^{(-1)^E \cdot P}, \tag{1.6}$$

где M – модуль мантиссы; P – модуль порядка. Диапазон изменения чисел с плавающей точкой составляет $0.5867 \cdot 10^{-38} \ldots 0.1704 \cdot 10^{+39}$ по модулю.

В табл. 1.3 приведены примеры представления чисел в формате с плавающей точкой `MC2702`. Значения байтов приведены в шестнадцатеричной системе счисления. Первый байт – порядок (P), далее идут старший (MH) и младший (ML) байты мантиссы. В памяти байты числа располагаются по возрастанию адресов: ML, MH, P.

1.6.5. Адресация данных

Команды, входящие в систему команд микропроцессора, принято называть *операторами*. Данные, над которыми операторы выполняют операцию, называются *операндами*. *Адресация* – это способ определения адреса операнда.

Большинство способов адресации связано с необходимостью вычисления адреса основной памяти, по которому производится фактическое обращение к ней. Этот адрес называют *исполнительным*. В зависимости от того, какие методы адресации реализованы в конкретном процессоре, в нем имеются те или иные адресные регистры.

Таблица 1.3. Примеры представления чисел в формате с плавающей точкой

Число	Машинное представление		
	Порядок	Мантисса	
	P	MH	ML
−1	01	A0	00
0	FF	20	00
1	01	20	00
2	02	20	00
2.5	02	28	00
2.75	02	2C	00
$1.6022 \cdot 10^{-19}$	BE	2F	49

Более сложные методы адресации требуют большего времени для вычисления адреса операнда. Наиболее распространенными методами адресации, используемыми в современных моделях микропроцессоров, являются следующие.

Неявная, или подразумеваемая. Адрес операнда включен в соответствующую микропрограмму и в коде команды не присутствует. Данный тип адресации часто используется для команд, работающих с аккумулятором.

Регистровая адресация. Операнд находится в регистре. Адрес регистра включен в код операции. Поле адреса в команде отсутствует.

Прямая адресация. Физический адрес операнда расположен в соответствующем поле адреса.

Непосредственная адресация. Непосредственное значение операнда расположено в соответствующем поле адреса.

Косвенная регистровая адресация. Физический адрес операнда расположен в регистре косвенного адреса DP (Data Pointer). Адрес регистра включен в код операции. Поле адреса в команде отсутствует. В качестве DP может выступать регистр общего назначения (РОН) или специальный адресный регистр.

Косвенная автоинкрементная/автодекрементная адресация. Физический адрес операнда расположен в регистре косвенного адреса DP. Адрес регистра включен в код операции. Поле адреса в команде отсутствует. После (либо до) выполнения операции содержимое DP автоматически инкрементируется/декрементируется, чтобы указывать на следующий элемент таблицы.

Адресация базовая со смещением. Базовый адрес операнда расположен в регистре базы BP (Base Pointer). Адрес регистра включен в код операции. Смещение адреса операнда относительно базового адреса расположено в соответствующем поле адреса. В качестве BP может выступать РОН или специальный адресный регистр.

Индексная адресация. Базовый адрес операнда расположен в соответствующем поле адреса. Смещение адреса операнда относительно базового адреса расположено в индексном регистре X (Index). В качестве X может выступать РОН или специальный адресный регистр.

Адресация базовая с индексированием. Базовый адрес операнда расположен в регистре базы BP, смещение адреса операнда относительно базового адреса расположено в индексном регистре X. Адреса регистров включены в код операции. Поле адреса в команде отсутствует. В качестве X и BP могут выступать РОН или специальные адресные регистры.

Относительная адресация. Адресацию, при реализации которой исполнительный адрес вычисляется как сумма фиксированного смещения в команде и текущего значения счетчика команд, называют относительной адресацией. Когда выполняется сложение с содержимым счетчика команд, последний уже указывает адрес очередной команды. Смещение может иметь знак.

Косвенная относительная адресация. Этот способ адресации отличается от относительной тем, что вычисленный рассмотренным только что способом адрес является не исполнительным адресом, а косвенным адресом операнда, т.е. адресом, по которому из памяти извлекается исполнительный адрес.

Относительная адресация с индексированием. В этом способе адресации базовым адресом является содержимое счетчика команд, а в

команде указываются смещение и номер регистра, который содержит дополнительное смещение (индекс). Исполнительный адрес формируется суммированием содержимого счетчика команд, смещения и индекса.

Страничная адресация. При использовании страничного способа адресации адресное пространство разбивается на ряд страниц одинаковой длины. Размер страницы равен 2^k байтов. Это позволяет представлять n-разрядный исполнительный адрес в виде двух частей: номера страницы (старшие n разрядов) и адреса байта на странице (младшие k разрядов). Страничный способ адресации не имеет никакого отношения к физической организации памяти. Этот способ просто используется для указания адреса с помощью небольшого количества разрядов. Команда содержит только адрес байта на странице. Номер страницы формируется одним из следующих путей: а) номеру страницы присваивается нулевое значение – адресация по нулевой странице; б) номер страницы устанавливается равным значению старших разрядов счетчика команд; при этом исполнительный адрес располагается на той странице, где находится выполняемая команда – адресация с использованием текущей страницы; в) номер страницы определяется по содержимому регистра страниц, в который предварительно программным путем загружается требуемый номер страницы – адресация с использованием регистра страниц.

Сегментная адресация. Вся память разбита на сегменты определенного объема. Адрес сегмента хранится в сегментном регистре, смещение адреса относительно начала сегмента расположено в соответствующем поле адреса либо в индексном регистре X. В качестве X может выступать РОН или специальный адресный регистр.

Обычно в микропроцессоре одновременно используется несколько способов адресации. Способ адресации указывается либо неявно кодом операции, либо в явной форме в специальном поле адресной части команды.

2. ИНТЕРФЕЙС И ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА

2.1. Интерфейс микропроцессорной системы

Интерфейс (Interface) – это совокупность линий, шин, сигналов, электронных схем, алгоритмов, протоколов, процедур, обеспечивающих обмен информацией между устройствами системы.

2.1.1. Интерфейс микропроцессорной системы и контроллеры периферийных устройств

Совокупность шин адреса, данных и управления образует системный интерфейс (рис. 2.1). Периферийные устройства (ПУ) подключаются к микропроцессорной системе (МПС) через системный интер-

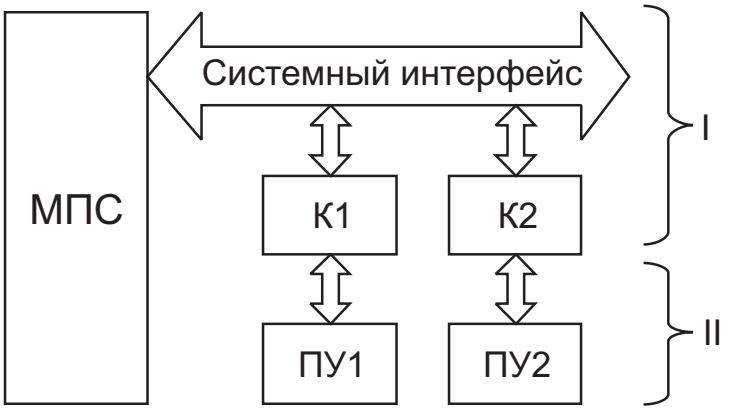


Рис. 2.1. Интерфейс микропроцессорной системы и контроллеры периферийных устройств

фейс. Большое разнообразие различных типов системных интерфейсов, с одной стороны, и периферийных устройств – с другой, создают проблемы при организации микропроцессорной системы.

Для повышения гибкости и модифицируемости системы используют двухуровневую схему подключения периферийных устройств: первый (верхний) уровень образован системным интерфейсом и контро́ллерами (К) периферийных устройств (рис. 2.1).

Контроллер выполняет две основные функции:

1) согласование конкретного периферийного устройства с системным интерфейсом;

2) управление работой периферийного устройства.

Простые контроллеры, выполняющие главным образом только первую функцию, иногда называют *адаптерами*.

На втором (нижнем) уровне расположены сами периферийные устройства. Все виды взаимодействия с ними в микропроцессорной системе заменяются взаимодействием с соответствующими контроллерами.

Обращение к контроллерам осуществляется по адресам, как к обычным устройствам ввода-вывода (УВВ). На программном уровне каждый контроллер представлен несколькими портами ввода-вывода. Адреса всех портов образуют адресное пространство устройств ввода-вывода. В случае совмещенных адресных пространств УВВ и основной памяти адресное пространство УВВ входит в состав общего адресного пространства микропроцессорной системы (рис. 2.2). Численные

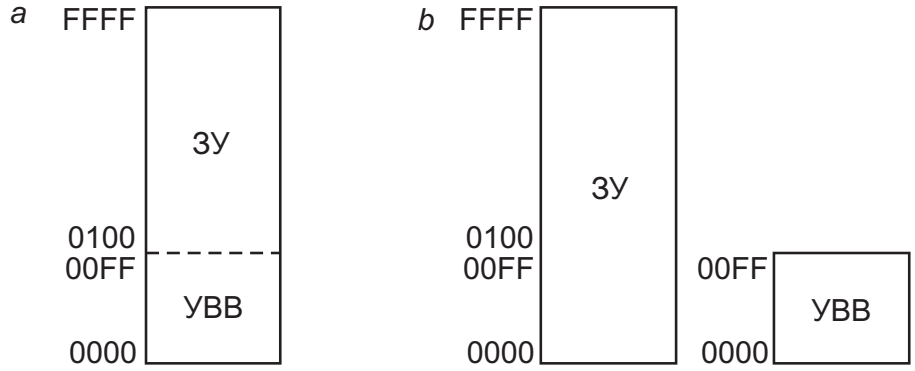


Рис. 2.2. Совмещенные (*a*) и разделенные (*b*) адресные пространства УВВ и основной памяти

значения адресов УВВ и основной памяти при этом всегда различны. Обращение к портам ввода-вывода и ячейкам памяти ЗУ осуществляется одними и теми же командами чтения и записи. Для доступа используют один общий набор сигналов шины управления: R (Read) и W (Write).

В случае разделенных адресных пространств УВВ и основной памяти численные значения адресов УВВ и ЗУ могут совпадать (рис. 2.2). Для обращения к портам УВВ используют отдельные команды ввода и вывода. Для доступа к УВВ применяют специальные сигналы INP (Input) и OUTP (Output).

Организация контроллера определяется как форматами данных и режимами работы периферийного устройства, так и характеристиками конкретного системного интерфейса. Отметим в качестве примера два стандартных системных интерфейса: Microbus и Multibus (И41).

Интерфейс Microbus предназначен для организации работы 8-разрядных микропроцессорных устройств, он содержит всего 36 линий, включая 16 линий адреса, 8 линий данных и 12 линий управления.

Интерфейс Multibus (И41) предназначен для организации работы 16-разрядных микропроцессорных устройств, он содержит 72 линии, включая 20 линий адреса, 16 линий данных и 36 линий управления.

Блок сопряжения

Выбор типа системного интерфейса однозначно определяет схемотехнику той части контроллера, которая осуществляет сопряжение с системным интерфейсом. Эта часть контроллера называется блоком сопряжения. На рис. 2.3 показана упрощенная схема блока сопряжения для работы в системе с разделенными адресными пространствами

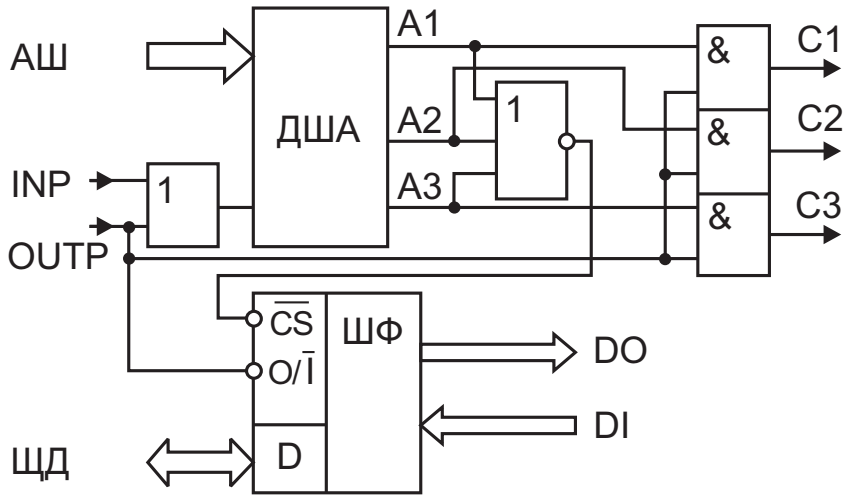


Рис. 2.3. Блок сопряжения с системным интерфейсом

УВВ и ЗУ. Блок сопряжения, приведенный на рис. 2.3 в качестве примера, рассчитан на работу в составе контроллеров, имеющих не более трех портов ввода-вывода. При необходимости блок сопряжения может быть спроектирован на большее количество портов ввода-вывода.

Рассмотрим пример работы блока сопряжения при выводе данных в порт с адресом `A2`.

1. Микропроцессор выставляет адрес `A2` на шине адреса, сигнал `OUTP` на шине управления и данные для вывода на шине данных.

2. На вход дешифратора адреса (ДША) поступают адрес с ША и сигнал `OUTP`. Логический элемент ИЛИ на входе дешифратора осуществляет функцию разграничения адресных пространств УВВ и ЗУ: в исходном состоянии дешифратор блокирован и не реагирует на сигналы АШ, даже если численные значения поступающих адресов совпадают с заданными кодами. Сигнал `INP` или `OUTP` снимает блокировку. При совпадении поступившего адреса с настройкой дешифратора происходит возбуждение соответствующей выходной линии. В рассматриваемом примере возбуждается выходная линия `A2`.

3. Сигнал `A2` формирует строб `C2` для управления регистром-защелкой порта вывода и сигнал $\overline{\text{CS}}$ для активизации шинного формирователя. Направление передачи данных шинного формирователя задается сигналом `OUTP`.

4. Данные с ШД поступают на выход `DO` шинного формирователя.

Работа блока сопряжения при вводе данных осуществляется сходным образом: данные с входа DI поступают на ШД.

2.1.2. Классификация способов обмена данными

Организация контроллера зависит также от способа (вида) обмена данными. На рис. 2.4 показана классификация различных видов обмена данными. Во всех случаях обмен данными происходит словами, передаваемыми один за другим последовательно во времени. При байтовом формате слов говорят, что имеет место “байт-последовательный” обмен данными. Различают виды обмена данными по параллельному и последовательному каналам.

Рис. 2.4. Классификация видов обмена данными

Параллельный канал. Все разряды слова передаются одновременно, каждый по своей линии. При байтовой организации говорят о “байт-последовательном, бит-параллельном” обмене данными.

Последовательный канал. Разряды слова передаются последовательно во времени, друг за другом по одной линии связи. При байтовой организации говорят о “байт-последовательном, бит-последовательном” обмене данными.

2.2. Программно-управляемый обмен данными по параллельному каналу

Виды обмена данными различают по способу управления: программное или аппаратное управление. При программно-управляемом обмене данными каждое слово (байт) данных обязательно проходит через регистры микропроцессора. При этом достигается полный программный контроль потока данных. Однако временные характеристики потока данных могут лимитироваться временными параметрами работы микропроцессора.

Программно-управляемые способы обмена по параллельному каналу классифицируются по способам проверки готовности периферийного устройства к обмену данными.

2.2.1. Синхронный обмен

В случае программно-управляемого синхронного обмена данными по параллельному каналу не производится проверки готовности периферийного устройства к обмену (рис. 2.5). При отсутствии такой проверки приемник и передатчик данных должны иметь одинаковый

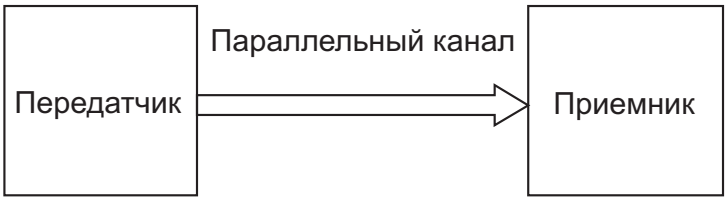


Рис. 2.5. Синхронный обмен по параллельному каналу

внутренний темп обработки данных. Отсюда название – *синхронный обмен* данными по параллельному каналу. Это самый быстрый способ обмена данными, т.к. приемник и передатчик работают в ничем не регулируемом, максимально возможном темпе. В некоторых системах в таком режиме обмена функционируют микропроцессор и основная память. Однако если партнеры имеют разный темп обработки данных, то синхронный способ не может быть применен.

Пример контроллера для синхронного программно-управляемого вывода по параллельному каналу показан на рис. 2.6. Основу кон-

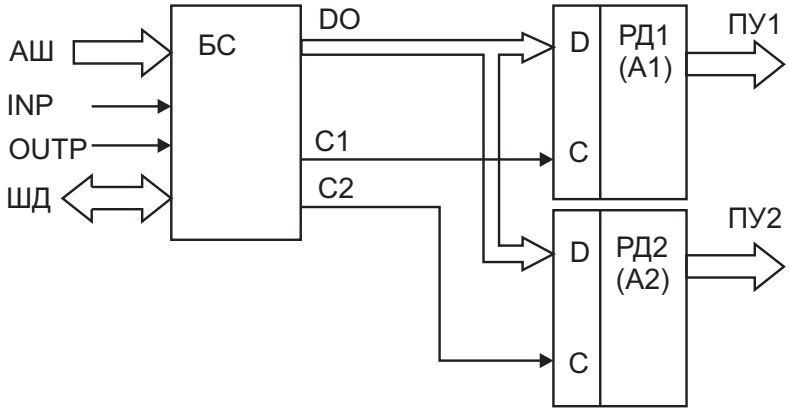


Рис. 2.6. Контроллер синхронного вывода данных по параллельному каналу

троллера составляет блок сопряжения (БС), к которому добавлены два регистра-защелки данных РД1 и РД2. В адресном пространстве УВВ данный контроллер занимает адреса `A1` и `A2`. Стробы `C1` и `C2` здесь используются для выбора одного из регистров данных.

Алгоритм работы (рис. 2.7, *а*) контроллера весьма прост. Микропроцессор через блок сопряжения выводит байт данных на линию `DO`, блок сопряжения формирует строб `C1`, по которому этот байт записывается в регистр данных РД1. Этот регистр является портом вывода, и периферийное устройство ПУ1 может считывать с него данные. Далее, микропроцессор через блок сопряжения выводит в регистр данных РД1 следующий байт и т.д. Темп выдачи байтов определяется лишь внутренними характеристиками микропроцессорной системы и выполняемой программы. В процессе работы не производится никакой проверки готовности ПУ1 к приему следующего байта.

2.2.2. Асинхронный обмен

В случае программно-управляемого асинхронного обмена данными по параллельному каналу производится проверка готовности периферийного устройства к обмену (рис. 2.8). Для этого предусмотрены две дополнительные однобитовые линии, по которым передаются служебные сигналы *квитирования*: “Данные готовы” и “Данные приняты”. Вывод каждого байта в параллельный канал передатчик сопровождает сигналом “Данные готовы”. Приемник по этому сигналу считывает данные и после их обработки выдает ответный сигнал “Данные приняты”. После поступления данного сигнала передатчик выводит следующий байт данных. Партнеры обмена при этом могут иметь различные внутренние темпы обработки данных, но общий

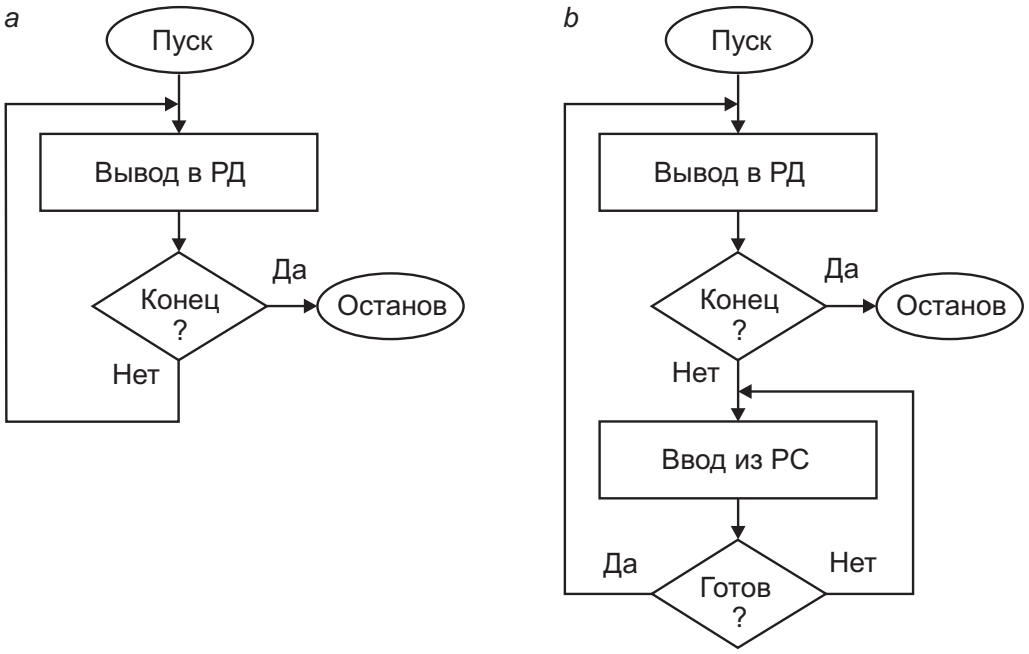


Рис. 2.7. Блок-схема алгоритма вывода данных по параллельному каналу в синхронном (*a*) и асинхронном (*b*) режимах

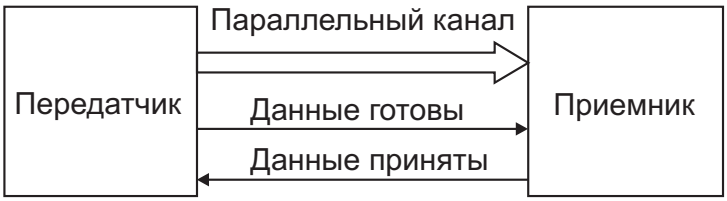


Рис. 2.8. Асинхронный обмен по параллельному каналу

темп обмена данными выравнивается сигналами квитирования по самому медленному партнеру. Отсюда название – асинхронный обмен.

Пример контроллера для асинхронного программно-управляемого вывода данных по параллельному каналу показан на рис. 2.9. Признак занятости периферийного устройства выставляется на флаговом RS-триггере при выводе байта в регистр данных РД1. Выход Q этого триггера используется для формирования сигнала единичного уровня “Данные готовы”. Сброс флагового триггера осуществляется сигналом “Данные приняты” от периферийного устройства. Для программной проверки готовности к обмену в контроллере (см. рис. 2.9) поддерживают специальный регистр признаков состояния (РПС). На программном уровне регистр РПС – обычный порт ввода с адресом `A2`. При вводе данных из этого порта информация флагового триггера переписывается в регистр РПС. В соответствии с алгоритмом, микропроцессор циклически программными средствами опрашивает регистр РПС, дожидаясь готовности периферийного устройства (рис. 2.7, *b*). Асинхронный способ обмена иногда называют обменом данными по флагам. Этот способ характеризуется простой програм-

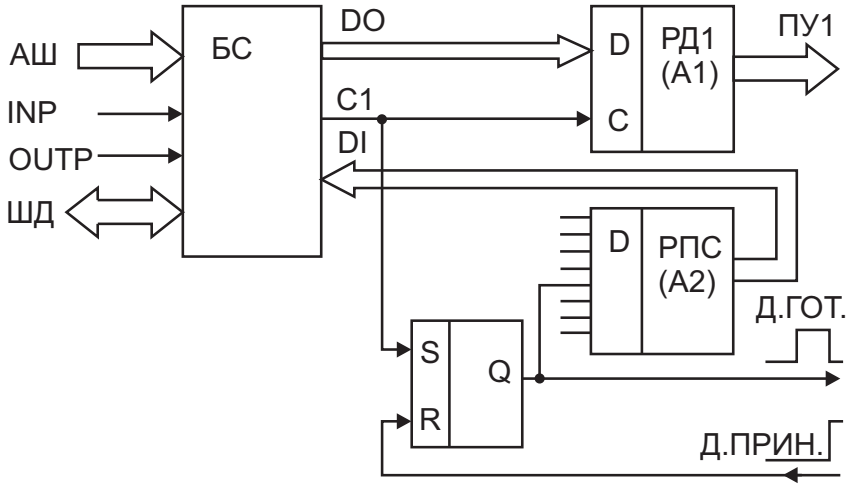


Рис. 2.9. Контроллер асинхронного вывода данных по параллельному каналу

мной и аппаратной реализацией. Основной недостаток проявляется при больших различиях во внутренних темпах работы партнеров и заключается в больших временных издержках на процесс опроса флага занятости.

В схемах, обслуживающих обмен данными по параллельному каналу, как правило, используется базовая структура БИС Intel 8255A (К580ВВ55А). Эта программируемая БИС представляет собой однокристальное устройство параллельного ввода-вывода и обеспечивает двунаправленный обмен с квитированием или без него при программном обмене или обмене по прерываниям. Устройство содержит три 8-разрядных порта ввода-вывода, причем один из них может быть разделен на два 4-разрядных канала.

2.3. Обмен данными по прерываниям

Программно-управляемый обмен данными по прерываниям характеризуется аппаратной проверкой готовности периферийного устройства. Это позволяет устранить основной недостаток асинхронного способа обмена – непроизводительные потери времени процессора в циклах ожидания. Режим прерываний может быть применим как к параллельному, так и последовательному каналам.

2.3.1. Организация системы прерываний

На рис. 2.10 приведен пример организации системы прерываний. Если периферийное устройство (ПУ) готово к операции ввода-вывода, то оно через контроллер периферийного устройства (КПУ) посылает на вход контроллера прерываний (КПр) сигнал запроса на обслуживание `IRQ`. По существу, этот сигнал представляет собой инвертирован-

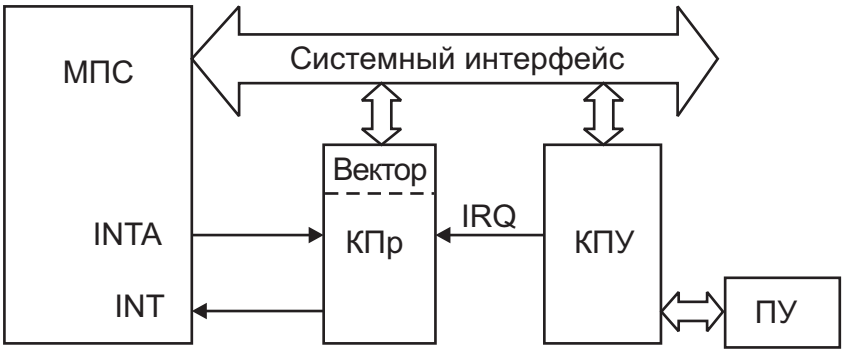


Рис. 2.10. Организация системы прерываний

ный сигнал флагового триггера (рис. 2.11). Контроллер прерываний формирует и передает процессору сигнал запроса прерывания INT.

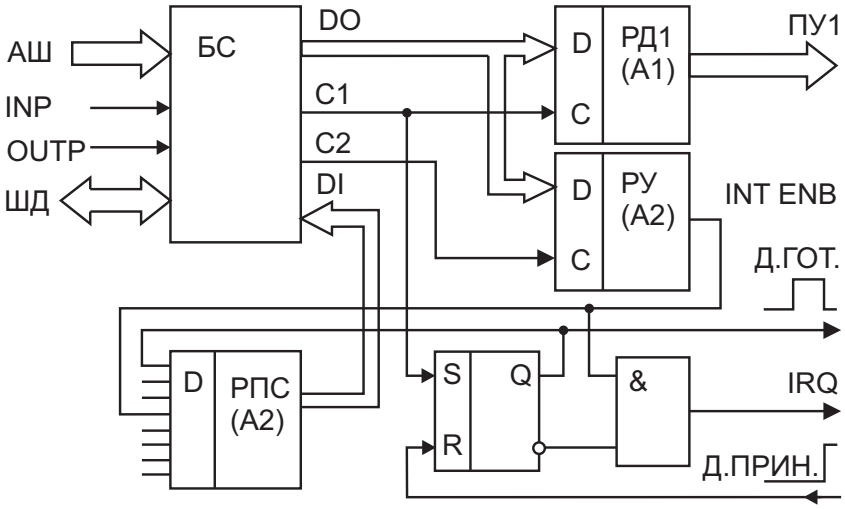


Рис. 2.11. Контроллер для вывода данных по прерываниям

Сигнал INT может появиться в произвольный момент времени, асинхронно по отношению к действиям процессора, поэтому, реагируя на сигнал INT, процессор вначале завершает выполнение текущей команды. Если прерывания процессору разрешены (не замаскированы), то он формирует сигнал INTA подтверждения прерывания. До получения этого сигнала контроллер прерываний сохраняет активный уровень сигнала INT.

Процессор прерывает (временно приостанавливает) текущую программу, запоминает содержимое счетчика команд PC и некоторых других внутренних регистров в стеке, причем содержимое PC обычно запоминается автоматически. Далее процессор идентифицирует прерывающее устройство, переходит к соответствующей подпрограмме обслуживания прерывания и выполняет ее. После этого содержимое регистров извлекается из стека, восстанавливается состояние прерванной программы и возобновляется ее выполнение.

После окончания подпрограммы обслуживания прерывания счет-

чик команд содержит адрес команды, которая выполнялась бы при отсутствии прерывания, поэтому после обслуживания прерывания программа продолжается обычным образом.

Действия микропроцессора при реагировании на сигнал прерывания похожи на действия при вызове подпрограммы. Однако вызов подпрограммы запрограммирован и полностью предсказуем, а переход к обслуживанию прерывания инициируется внешним сигналом, момент появления которого предсказать невозможно. Тем не менее внешняя аналогия реакции на прерывание и вызов подпрограммы позволяет считать прерывание аппаратным вызовом подпрограммы.

Адрес подпрограммы обслуживания прерываний задается вектором прерывания. В различных микропроцессорных системах формат вектора прерываний может быть различным, но в любом случае вектор прерывания содержит всю необходимую информацию для перехода к подпрограмме прерывания.

Вектор прерывания может представлять собой:

- адрес подпрограммы обслуживания прерывания;
- код команды вызова подпрограммы обслуживания прерывания;
- индекс (номер вектора прерывания), который программно или аппаратно сопоставлен с подпрограммой обслуживания прерываний.

Микропроцессор после завершения каждой команды проверяет наличие сигнала прерывания до перехода к следующей команде. Переход к подпрограмме обслуживания прерывания происходит, если только INT=1 и прерывания разрешены.

Процессор реагирует на запросы маскируемых прерываний по линии INT, если установлен внутренний триггер разрешения прерываний INTE, называемый также маской (рис. 2.12). Если триггер INTE=0, прерывания запрещены (замаскированы) и процессор не реагирует на сигнал INT=1. С помощью команд разрешения EI и запрещения DI прерываний можно программно управлять состоянием триггера INTE. При обработке прерывания устанавливается триггер прерываний INTA, что приводит к запрещению инкремента счетчика команд и формированию сигнала подтверждения прерывания INTA. Отметим, что при этом сбрасывается триггер INTE и в дальнейшем разрешить прерывания можно только командой EI. После выполнения команды EI процессор обязательно выполняет еще одну команду, даже если на входе INT действует сигнал прерывания INT=1. Для программного управления прерываниями каждого периферийного устройства в их регистрах управления (РУ) предусмотрен специальный бит INT ENB разрешения прерываний (маска) (см. рис. 2.11). Иногда биты масок всех устройств объединяются в специальной регистр.

Запросы немаскируемых прерываний по входу NMI процессор вос-

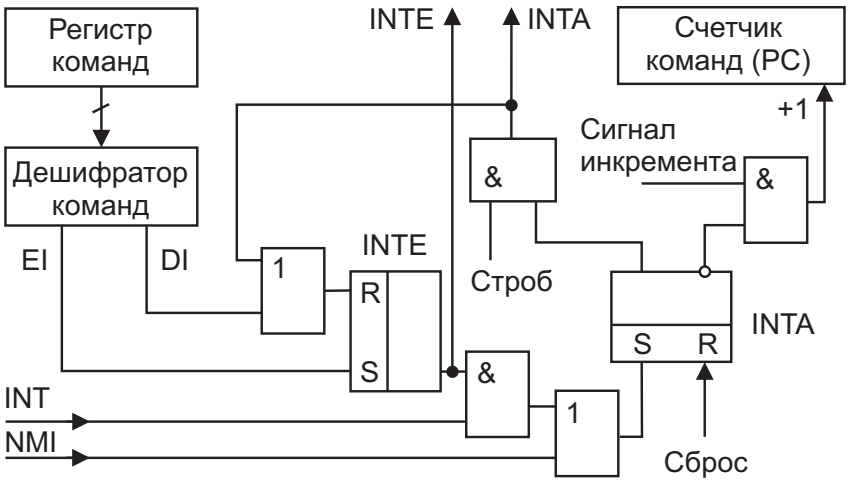


Рис. 2.12. Внутренняя логическая схема прерываний

принимает всегда, независимо от того, разрешены прерывания или нет. Однако до реакции на сигнал NMI процессор завершает выполнение текущей команды. Обычно на вход NMI подается сигнал от схемы, фиксирующей уменьшение напряжения сети до некоторого критического уровня. Быстродействие процессора достаточно велико, чтобы за время от момента восприятия сигнала NMI до уменьшения напряжения питания ниже минимального допустимого выполнить несколько команд. Этими командами содержимое всех внутренних регистров и другая важная информация записываются в энергонезависимую память.

Если число периферийных устройств в системе прерываний больше одного, то сигналы их запросов на обслуживание объединяются по схеме ИЛИ и подаются на вход INT процессора. При поступлении активного сигнала INT=1 без дополнительных действий и аппаратных средств невозможно определить, какое устройство должен обслуживать процессор. Возникает проблема идентификации прерывающего устройства, т.е. однозначного перехода к определенной подпрограмме обслуживания. Кроме того, запросы на обслуживание могут формироваться одновременно несколькими устройствами. При наличии нескольких источников запросов прерывания должен быть установлен определенный порядок (дисциплина) в обслуживании поступающих запросов. Другими словами, между запросами (и соответствующими подпрограммами прерываний) должны быть установлены приоритетные соотношения, определяющие, какой из нескольких поступивших запросов подлежит обработке в первую очередь, и устанавливающие, имеет право или не имеет данный запрос (прерывающая программа) прерывать ту или иную программу. Процедура организации перехода к подпрограмме прерывания включает в себя выделение из выставленных запросов такого, который имеет наибольший приоритет.

Различают *абсолютный* и *относительный* приоритеты. Запрос,

имеющий абсолютный приоритет, прерывает выполняемую программу и инициирует выполнение соответствующей прерывающей программы. Запрос с относительным приоритетом является первым кандидатом на обслуживание после завершения выполнения текущей программы.

Если наиболее приоритетный из выставленных запросов прерывания не превосходит по уровню приоритета выполняемую процессором программу, то запрос прерывания игнорируется или его обслуживание откладывается до завершения выполнения текущей программы.

Существует несколько способов, различающихся скоростью реакции процессора и объемом дополнительных аппаратных средств. При реализации любого способа необходимо назначить устройствам определенные приоритеты и учитывать их таким образом, чтобы процессор, реагируя на сигнал прерывания, выбирал для обслуживания запрашивающее устройство с максимальным приоритетом. Системы прерываний классифицируют по объему перечисленных действий, которые реализуются аппаратно или программно. В зависимости от способа решения этих проблем различают *невекторные* и *векторные* прерывания.

2.3.2. Невекторные прерывания

Общая схема организации невекторных прерываний (рис. 2.13) состоит из контроллеров периферийных устройств (КПУ) и одного общего контроллера прерываний (КПр). Запросы на обслуживание IRQ

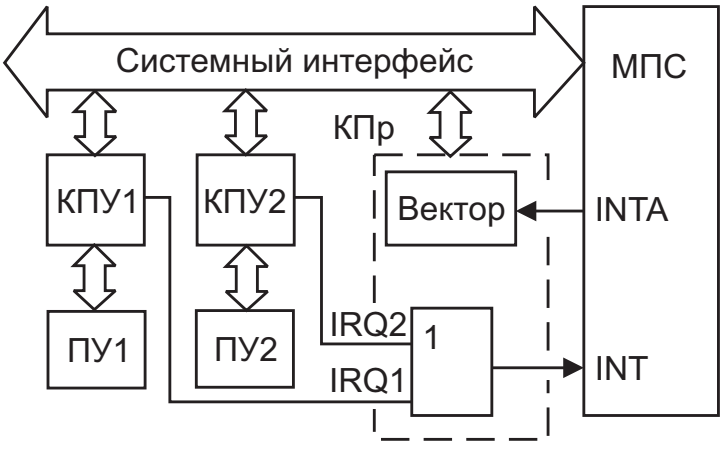


Рис. 2.13. Организация невекторных прерываний

от всех КПУ объединяются по схеме ИЛИ в контроллере прерываний и подаются на вход `INT` процессора. Внутренняя логика процессора показана на рис. 2.12: при установленном триггере INTE и сигнале INT=1 процессор выполняет типовые действия по подготовке к передаче управления подпрограмме обслуживания прерывания:

1) завершение текущей команды;
2) запрет автоинкремента счетчика команд PC;
3) блокировка доступа к основной памяти;
4) выдача сигнала подтверждения прерываний `INTA`;
5) переход в режим выборки кода операции.

Контроллер прерываний (рис. 2.14) содержит регистр вектора прерываний (РВП). В регистр РВП до начала обмена данными записывают значение вектора прерывания, соответствующее выбранной подпрограмме обслуживания этого прерывания. По сигналу `INTA` вектор прерывания из регистра РВП выставляется на шину данных. Процессор, работающий в режиме выборки кода операции, считывает с ШД вектор прерываний, декодирует его по заданным правилам и начинает выполнять операцию передачи управления подпрограмме, соответствующей вектору прерывания. Переход к подпрограмме также состоит из ряда типовых действий:

1) сохранение в стеке точки возврата (содержимое регистра PC);
2) загрузка в счетчик команд адреса подпрограммы;
3) снятие запрета автоинкремента;
4) снятие блокировки основной памяти;
5) выборка кода первой команды подпрограммы.

Система невекторных прерываний подразумевает вызов одной и той же подпрограммы при обслуживании различных периферийных устройств. Проблема идентификации периферийного устройства, запросившего прерывание, решается при этом путем опроса (поллинга) источников прерывания. Поллинг может осуществляться различными способами как программно, так и аппаратно.

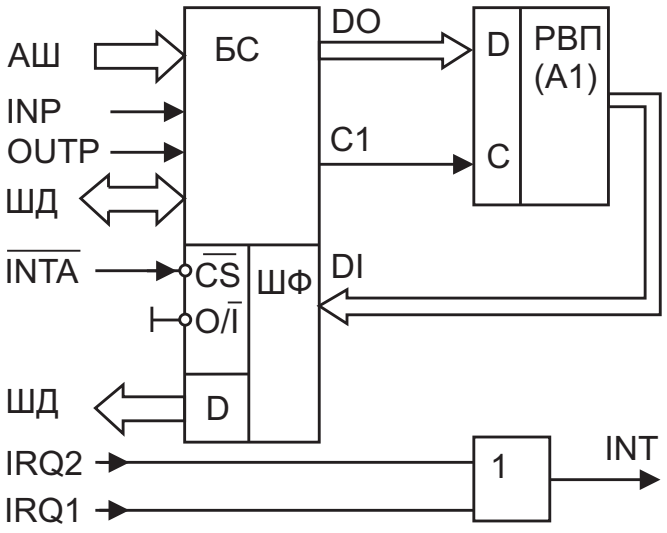


Рис. 2.14. Контроллер невекторных прерываний

Программный поллинг

Идентификация путем программного опроса (поллинга) контроллеров периферийных устройств заключается в следующем. Каждый КПУ (см. рис. 2.11) содержит регистр признаков состояний (РПС), который фиксирует состояния флага готовности периферийного устройства и бита разрешения прерываний `INT ENB`. Реагируя на прерывание, процессор переходит к подпрограмме поллинга (рис. 2.15). Первым проверяется флаг готовности устройства У1 с наибольшим приоритетом. Если это устройство не посылало запрос на обслужи-

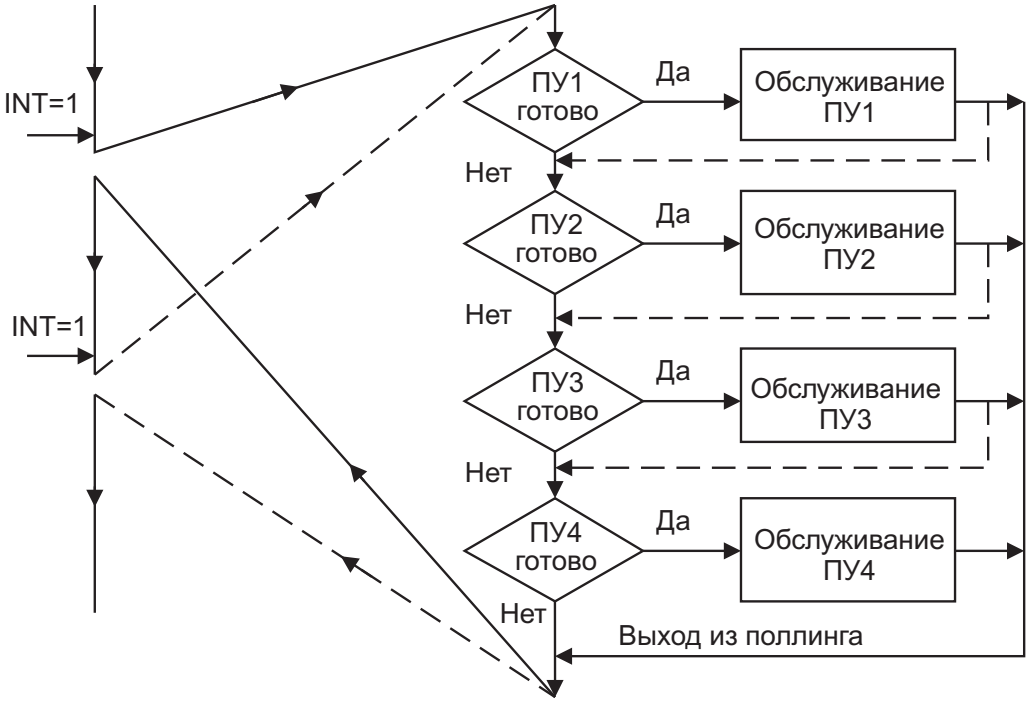


Рис. 2.15. Программный поллинг

вание, то идет опрос следующего устройства и т.д. Когда встречается первое устройство, готовое к операциям ввода-вывода, управление передается блоку подпрограммы обслуживания этого устройства. По завершении обслуживания в поллинге может быть запрограммировано одно из следующих действий:

1) управление возвращается в основную программу без проверки готовности остальных устройств. Здесь гарантируется обязательная проверка в каждом цикле поллинга устройств с высоким приоритетом, так как обслуживание их блокирует обслуживание устройств с меньшими приоритетами;

2) управление возвращается к программе поллинга, т.е. в точку проверки прерывания следующего устройства (на рис. 2.15 показано штриховыми линиями). Этот способ гарантирует проверку в каждом цикле поллинга всех устройств.

Конкретная реализация поллинга зависит от особенностей системы команд процессора и конфигурации аппаратных средств. Например, для ускорения поллинга сигналы прерываний всех устройств подключаются к специальному регистру. Поллинг реализуется посредством ввода в процессор содержимого этого регистра и анализа состояний отдельных бит с помощью команд сдвига или маскирования. Последовательность анализа определяется приоритетами устройств, и при обнаружении первого установленного бита осуществляется переход к соответствующей подпрограмме обслуживания.

Недостаток программного поллинга заключается в необходимости проверки всех устройств, даже тех из них, которые не требуют обслуживания. Каждая проверка представляет собой последовательность команд, которые должен выполнять процессор. При увеличении числа устройств быстро увеличивается и число команд, что приводит к непроизводительным потерям времени процессора и замедлению его реакции на запросы устройств. В результате поллинг вносит значительную задержку между моментом, когда устройство сигнализирует о своей готовности к операциям ввода-вывода, и моментом собственно передачи данных. Основное достоинство программного поллинга заключается в простоте его реализации, почти не требующей дополнительных аппаратных средств.

Схема циклического опроса запросов прерываний

Схема циклического опроса запросов прерываний представляет собой разновидность аппаратного поллинга. В этой схеме идентификации биты запросов на прерывание всех устройств объединяются в специальной регистр запросов. Опрос k линий запросов прерывания (или разрядов регистра запросов прерывания) производится последовательно (циклически) с помощью n-разрядного счетчика ($2^n = k$), на который с некоторой частотой поступают импульсы от генератора. Поиск приоритетного запроса прерывания начинается со сброса счетчика в нулевое состояние, при этом импульсы генератора начинают поступать на вход счетчика. При помощи дешифратора и элементов “И” в каждом такте поиска проверяется наличие запроса прерывания, номер которого совпадает с кодом счетчика. Если на данном входе нет запроса прерывания, то после инкремента счетчика проверяется следующий по порядку вход. Если имеется запрос, то в процессор посылается сигнал прерывания `INT` и прекращается поступление импульсов на вход счетчика, т.е. завершается цикл просмотра входов системы прерывания. Содержимое счетчика — код номера старшего по приоритету выставленного запроса — используется для формирования адреса подпрограммы обслуживания прерывания. После передачи управления прерывающей подпрограмме счетчик сбрасывается.

Циклический (последовательный) опрос входов системы прерывания в аппаратурном отношении сравнительно прост, однако время реакции и при этом методе все-таки велико, особенно при большом числе источников запросов. Поэтому во многих случаях, например в микропроцессорах, предназначенных для работы в режиме реального времени, применяют схемы, позволяющие определять номер выставленного запроса или уровня прерывания старшего приоритета за один такт.

Цепочечная однотактная схема приоритетных запросов

Данную схему невекторных прерываний называют также дейзи-цепочкой, она представляет собой однотактную разновидность аппаратного поллинга (рис. 2.16). Процедура определения приоритетного запроса инициируется сигналом ACK, поступающим на цепочку после-

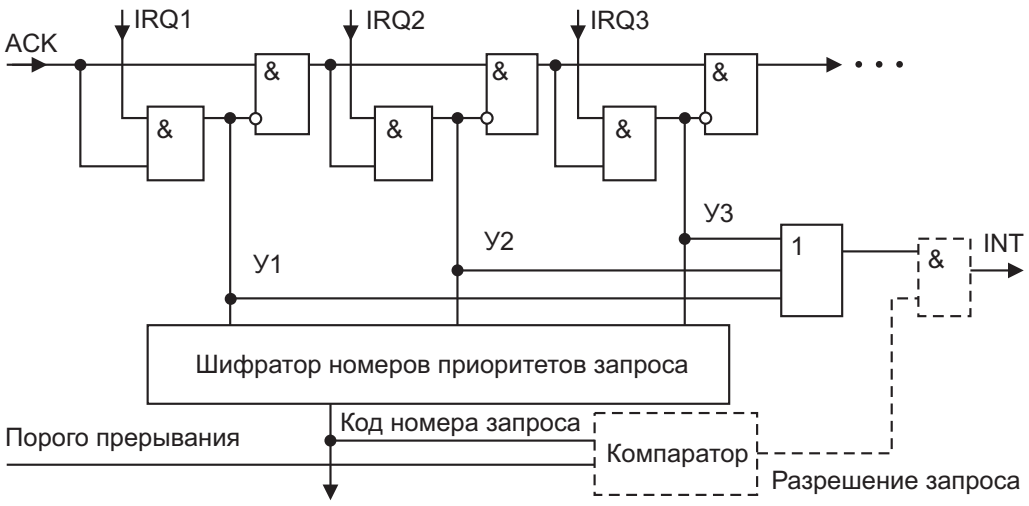


Рис. 2.16. Цепочечная однотактная схема определения приоритетного запроса

довательно включенных логических схем. При отсутствии запросов этот сигнал пройдет через цепочку и сигнал общего запроса прерывания не сформируется. Если среди выставленных запросов прерывания наибольший приоритет имеет *i*-й запрос, то распространение сигнала Приоритет правее логической схемы с номером *i* блокируется. На *i*-1 выходе цепочечной схемы будет сигнал $y_i = 1$, на всех других – 0. В процессор поступит общий сигнал прерывания, при этом шифратор по сигналу $y_i = 1$ сформирует код номера *i*-го запроса, принятого к обслуживанию. Этот код считывается прерывающей подпрограммой и используется для формирования начального адреса подпрограммы обслуживания прерывания. Порог прерывания позволяет в ходе вычислительного процесса задавать минимальный уровень приоритета запросов, которым разрешается прерывать основную программу.

В системе невекторных прерываний с идентификацией источников прерываний путем поллинга переход при прерывании происходит по одному и тому же адресу и инициирует одну и ту же прерывающую подпрограмму. Прерывающая подпрограмма для идентификации источника прерываний проводит программный поллинг либо считывает результат аппаратного поллинга. После идентификации источника запроса прерывающая подпрограмма формирует адрес начала соответствующей запросу подпрограммы обслуживания и передает ей управление.

2.3.3. Векторные прерывания

Принципиальное отличие векторных прерываний состоит в назначении каждому периферийному устройству своего собственного вектора прерывания и, следовательно, отдельной подпрограммы обслуживания этого прерывания. При получении вектора прерываний микропроцессор сразу передает управление подпрограмме обслуживания прерывания. Однако эта подпрограмма уже не содержит поллинга, а сразу выполняет действия по обслуживанию периферийного устройства. Это позволяет значительно ускорить реагирование микропроцессора на запрос прерывания.

Организация ввода-вывода с прерыванием, значительно сокращая непроизводительные потери времени процессора, связана с введением дополнительных аппаратных средств. В частности, это требуется для решения проблем установления приоритета в обслуживании запросов. Существуют различные аппаратные решения проблемы приоритета. В этом отношении различают векторные системы с *интерфейсным* и *внеинтерфейсным* заданием вектора прерываний.

Интерфейсное задание векторов прерываний

В этом случае каждый вектор прерывания формируется соответствующим контроллером периферийного устройства, запросившего обслуживание. Интерфейсное задание векторов прерываний не требует значительных аппаратных ресурсов. По такому принципу построена система векторных прерываний типа приоритетной цепочки (другие названия этой системы – аппаратный поллинг, гирляндное или каскадное включение устройств).

В *приоритетной цепочке* процессор и все периферийные устройства соединяются таким образом, что процессор может осуществить автоматический аппаратный опрос (аппаратный поллинг) в целях идентификации прерывающего устройства (рис. 2.17). Когда процессор реагирует на запрос прерывания, он формирует сигнал подтверждения

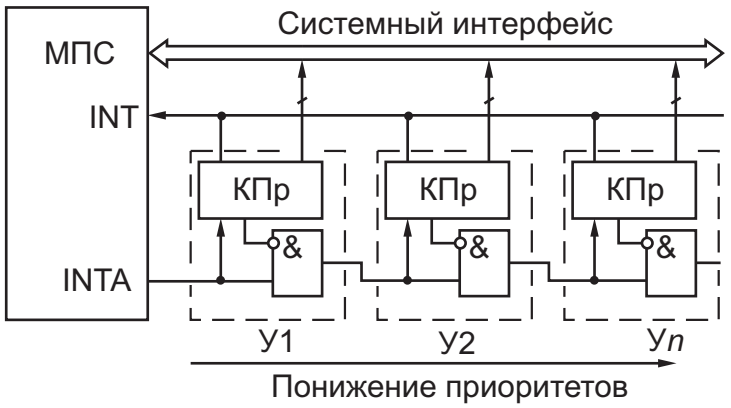


Рис. 2.17. Организация приоритетной цепочки

прерывания INTA на линии, которая последовательно проходит через все устройства. При прохождении сигнала по цепочке проверяется состояние флажков готовности устройств. Если устройство не формирует прерывания, сигнал INTA проходит в следующее устройство, пока не встретится прерывающее (активное) устройство. Оно блокирует дальнейшее распространение сигнала INTA по цепочке. Приоритеты устройств определяются их позиционной близостью к процессору по линии INTA. Активное устройство, получив сигнал INTA, выдает на шину данных свой вектор прерываний. По этому вектору процессор сразу передает управление подпрограмме обслуживания прерывания.

Вектор прерывания в подобных системах встраивается в интерфейсную плату. Приоритет устройства определяется размещением его контроллера в разъеме, занимающем фиксированное положение в схеме. На рис. 2.18 приведена схема контроллера периферийных устройств для вывода данных по векторным прерываниям в системе с приоритетной цепочкой. При поступлении сигнала INTA контроллер, пославший сигнал INT, выдает из регистра РВП на шину данных вектор прерывания.

При наличии нескольких одновременных запросов прерываний приоритетная цепочка обеспечивает быструю идентификацию устройства с наибольшим приоритетом. По линии прерывания к процессору можно подключить значительное количество периферийных устройств. Основной недостаток этого способа связан с задержкой распространения сигнала INTA в цепочке. Однако общее время идентификации прерывающего устройства оказывается намного меньше, чем при программном поллинге. Другой недостаток проявляется, если обслуживание запросили одновременно два или более периферийных устройства: обслуживание менее приоритетных устройств будет отложено на время обслуживания более приоритетных, как и в системе прерывания с программным опросом.

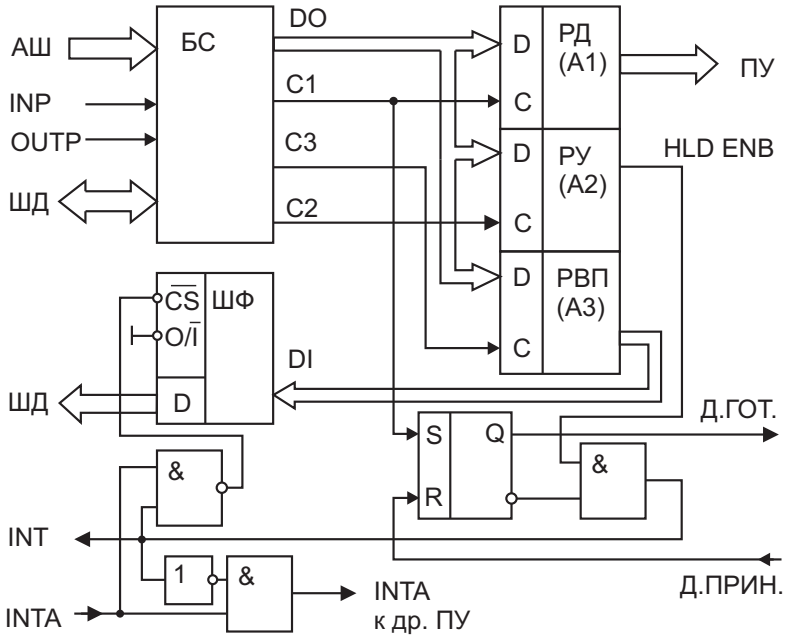


Рис. 2.18. Контроллер вывода данных по векторным прерываниям в системе с приоритетной цепочкой

Вложенные прерывания

Запрещение прерываний на время обслуживания любого периферийного устройства может привести к потере запросов прерываний быстродействующих, высокоприоритетных устройств, появляющихся при обслуживании устройств с меньшими приоритетами. Для исключения такой ситуации и обеспечения быстрой реакции процессора на прерывание необходимо решить две проблемы: каким-либо образом очень быстро (в пределе с быстродействием комбинационных схем) идентифицировать запрос прерывания устройства с максимальным приоритетом из имеющихся запросов прерываний; фиксировать текущий приоритет (порог) любой выполняемой процессором программы (в том числе и всех подпрограмм обслуживания прерываний) и разрешать ее прерывание только при возникновении запроса прерывания с большим приоритетом.

Прерывание подпрограмм обслуживания прерываний называется *вложением прерываний* (рис. 2.19). До момента t_1 выполняется основная программа, которой назначается наименьший приоритет, чтобы процессор реагировал на любые прерывания. В момент t_1 запрашивает обслуживание устройство `У4` и процессор переходит на его подпрограмму обслуживания. В свою очередь эта подпрограмма прерывается в момент t_2 запросом от устройства `У3` с более высоким приоритетом. Подпрограмма обслуживания `У3` в момент t_3 прерывается запросом устройства `У2` с еще более высоким приоритетом, и по завершении обслуживания `У2` в момент t_4 управление возвращается к продолжению обслуживания устройства `У3`. В интервале t_5 - t_6 аналогичным

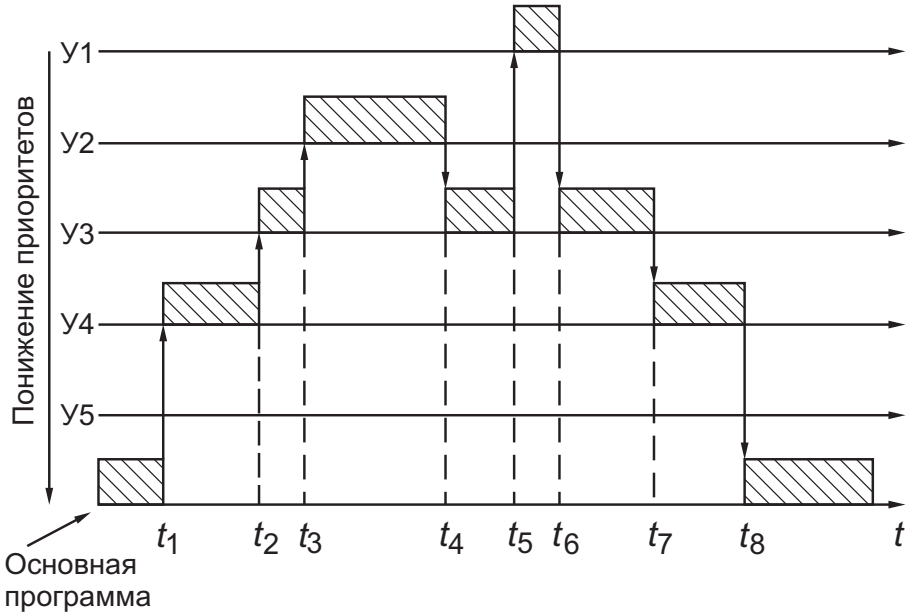


Рис. 2.19. Вложенные прерывания

образом обслуживается запрос устройства `У1` с максимальным приоритетом, после чего управление последовательно возвращается к прерванным подпрограммам обслуживания устройства `У3` и `У4`. Наконец, в момент t_8 возобновляется выполнение основной программы. Разумеется, чтобы процессор реагировал на запросы прерываний, в начале каждой подпрограммы обслуживания их необходимо разрешать командой `EI`. Эти проблемы обычно решаются с помощью специальных БИС приоритетных прерываний.

Многоуровневые векторные прерывания

Существует более сложная, но и более гибкая система прерываний. Она рассчитана на обработку не только внешних аппаратных прерываний от периферийных устройств, но и внутренних аппаратных прерываний, инициируемых при выполнении программы в случае возникновения особых ситуаций (переполнение, нереализованная операция и т.п.). Система прерываний называется многоуровневой, если процессор имеет несколько линий (уровней) запросов прерываний и если подпрограммы обслуживания одного уровня могут прерываться запросами на другом уровне. При наличии n уровней допустимая глубина вложений прерываний равна n. Многоуровневые системы прерываний удобны для группировки периферийных устройств с похожими характеристиками (и одинаковыми приоритетами) на одном и том же уровне. Каждому периферийному устройству назначается аппаратный приоритетный уровень. Приоритеты служат характеристиками устройств, и программно их изменять нельзя. Разрешение и запрещение

прерываний осуществляется в каждом периферийном устройстве, для чего в их регистрах управления и состояния имеется специальный бит разрешения прерываний (маски) `INT ENB` (см. рис. 2.11). Любое устройство с разрешенными прерываниями, приоритет которого выше приоритета выполняемой программы, может прервать процессор.

Внеинтерфейсное задание векторов прерываний

При внеинтерфейсном задании векторов контроллеры внешних устройств не имеют регистров для хранения векторов прерывания, а для идентификации устройств, запросивших обслуживание, используется общий для всех периферийных устройств контроллер многоуровневых векторных прерываний. В конструкции контроллера векторных прерываний предусмотрены несколько регистров векторов прерываний и аппаратная реализация различных схем установления приоритетов.

Общая схема организации векторных прерываний (рис. 2.20) во многом похожа на предыдущую схему невекторных прерываний, показанную на рис. 2.13. В примере (рис. 2.21) двухуровневый контрол-

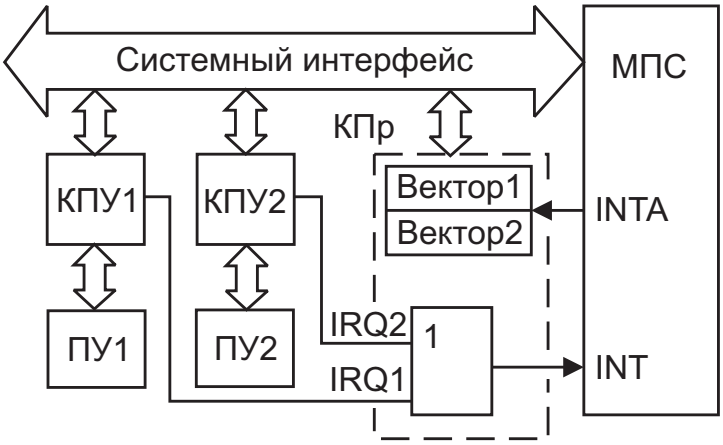


Рис. 2.20. Организация векторных прерываний

лер векторных прерываний содержит два регистра РВП, которые могут быть программно загружены до начала обмена. Сигналы `IRQ1` и `IRQ2` используются не только для формирования запроса на прерывание `INT`, но и для выработки адреса (`Адр`), управляющего мультиплексором векторов прерываний. В зависимости от того, по какой линии пришел запрос на обслуживание, логическая схема выдает на шину DI соответствующий вектор прерываний. Это позволяет аппаратным образом решить проблему идентификации устройства, запросившего прерывание. Каждому запросу на обслуживание соответствует свой вектор прерывания и, следовательно, своя подпрограмма обслуживания прерывания. В этой связи отпадает необходимость в программном

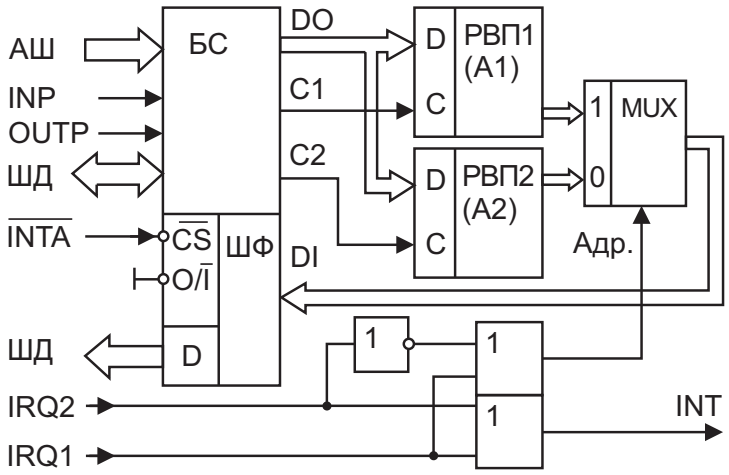


Рис. 2.21. Контроллер векторных прерываний

опросе для идентификации устройства, запросившего прерывание. После выдачи контроллером вектора прерывания сразу происходит переход к подпрограмме обслуживания именно этого периферийного устройства. Приоритет уровней прерываний в этом контроллере задан аппаратно схемой формирования адреса (Адр). Легко проследить по схеме (рис. 2.21), что уровень IRQ1 имеет высший приоритет.

В системе векторных прерываний с внеинтерфейсным заданием векторов прерываний используются обычно специализированные аппаратные средства. Значительная часть аппаратных средств для организации прерываний сосредоточена в БИС программируемых контроллеров прерываний.

В качестве примера рассмотрим возможности БИС программируемого контроллера прерываний Intel 8259A (К580ВН59). Этот контроллер обеспечивает управление 8-уровневыми векторными приоритетными прерываниями, число которых может быть расширено до 64. Допускается программное маскирование (запрещение) прерываний. Поддержаны следующие режимы работы:

1. *Вложенные прерывания*. Каждому из 8 входов запросов прерываний IRQ7-0 назначается фиксированный приоритет в порядке возрастания, и запрос с большим приоритетом прерывает обслуживание прерываний с меньшими приоритетами.

2. *Круговой (циклический) приоритет*. Каждому входу IRQ7-0 назначается приоритет, но теперь после запроса прерывания и выполнения соответствующей подпрограммы обслуживания приоритеты изменяются в круговом порядке таким образом, что последний обслуженный вход будет иметь низший приоритет. Этот режим характерен для таких применений, в которых периферийные устройства имеют одинаковый приоритет и ни одному из них нельзя отдать предпочтение.

3. *Адресуемые приоритеты*. Режим аналогичен второму режиму,

но допускает программное определение входа IRQ, которому назначается низший приоритет.

4. *Режим опроса*. В этом режиме прерывания процессора запрещаются, а требующее обслуживания периферийное устройство идентифицируется с помощью считывания состояния контроллера.

2.4. Прямой доступ к памяти

Обмен данными с аппаратным управлением носит название "прямой доступ к памяти" (ПДП) или *Direct Memory Access* (DMA). Аппаратное управление обменом данных осуществляет специальный контроллер ПДП. Микропроцессор в обмене не участвует и отключается от системного интерфейса. Данные передаются от одного устройства к другому через системный интерфейс, минуя микропроцессор. Одним из партнеров обмена чаще всего выступает основная память микропроцессорной системы. Отсюда название – ПДП.

Возможны два вида ПДП – с блочными или одиночными передачами. В первом работа процессора останавливается на все время передачи блока данных, во втором передачи слов в режиме ПДП перемежаются с выполнением программы, и для передач ПДП выделяются отдельные такты машинных циклов, в которых процессор не использует системный интерфейс. Одиночные передачи называют ПДП с захватом машинного цикла. Они бывают двух видов: без блокировки и с блокировкой микропроцессора. Каждый командный цикл начинается с машинного цикла выборки команды. В этом машинном цикле есть такт декодирования кода операции, в котором системные шины не используются. На это время системный интерфейс можно отдать для ПДП и передать одно слово. Производительность системы может возрасти из-за параллелизма процессов обмена и обработки данных, благодаря тому, что ПДП будет для процессора "невидимым". Сам же обмен с ПДП будет небыстрым, темп обмена нерегулярен, может и замедлить выполнение программы, если цикл ПДП не уложится в интервал, соответствующий такту процессора.

При непрерывной передаче блока данных скорость обмена ограничивается лишь длительностью циклов ЗУ, быстродействием самого контроллера и скоростью выдачи/приема данных внешним устройством.

В отличие от процессов прерывания, при ПДП обмен выполняется без участия программы, поэтому содержимое рабочих регистров процессора не изменяется и на вхождение в режим ПДП не требуется затрат времени. ПДП предоставляется по завершении текущего машинного цикла.

Для единиц байтов применяется первый вид ПДП, для больших

блоков байтов – второй. На рис. 2.22 показана схема организации обмена данными по методу ПДП между двумя периферийными уст-

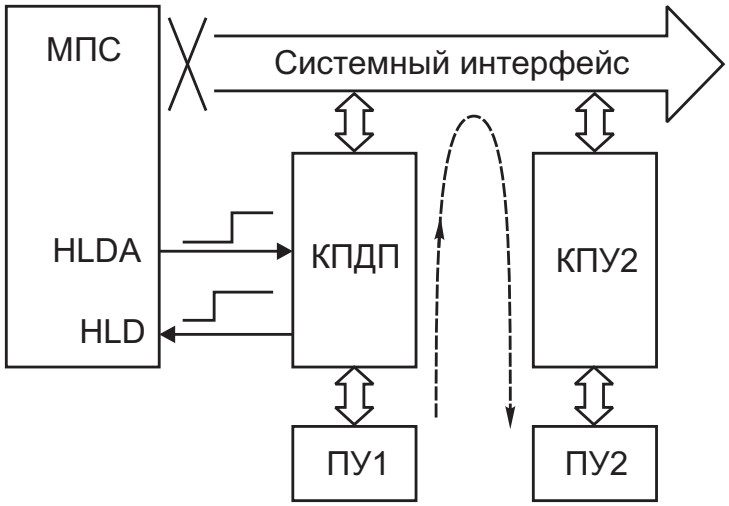


Рис. 2.22. Организация прямого доступа к памяти

ройствами ПУ1 и ПУ2. В качестве ПУ2, подключенного через контроллер КПУ2 к системному интерфейсу, может выступать, например, основная память микропроцессорной системы (МПС). Периферийное устройство ПУ1 оснащено контроллером ПДП (КПДП). Последовательность действий при обмене данными методом ПДП:

1) микропроцессор инициирует (программирует) контроллер ПДП, размещая в регистрах КПДП следующую информацию: количество байтов передаваемых данных; область памяти (адреса) в ЗУ, которые будут использоваться; начальный адрес блока данных. В регистре управления КПДП устанавливается бит `HLD ENB`, разрешающий контроллеру режим прямого доступа к памяти;

2) в какой-то момент времени периферийное устройство ПУ1 формирует признак (флаг) готовности. По этому признаку контроллер ПДП посылает микропроцессору сигнал `HLD`, который называется *требование ПДП*. Название сигнала происходит от слова HOLD – захват. Микропроцессор не имеет программных средств, чтобы отклонить требование ПДП. Запретить ПДП можно, только сбросив бит `HLD ENB` в контроллере ПДП;

3) завершив текущий машинный цикл, микропроцессор отключается от системного интерфейса (шинные формирователи переводятся в третье состояние) и выдает сигнал `HLDA` – *подтверждение ПДП*;

4) контроллер ПДП захватывает управление системным интерфейсом и осуществляет передачу блока данных в соответствии с заданными условиями. Функциональная организация контроллера ПДП должна предусматривать формирование всей совокупности сигналов управления системным интерфейсом;

5) после окончания передачи блока данных контроллер ПДП снимает сигнал HLD. В ответ на это процессор снимает сигнал HLDA и вновь подключается к системному интерфейсу.

Контроллер прямого доступа к памяти

На рис. 2.23 приведена упрощенная схема контроллера ПДП, которая, однако, позволяет проследить все основные особенности его организации. Работу начинают с программирования контроллера путем записи параметров обмена в его регистры:

1) в регистр управления РУ (А1) записывают `HLD ENB`=0. Это соответствует запрету режима ПДП на время программирования контроллера;

2) в автодекрементный регистр счетчик байтов СБ (А2) записывают количество байтов в блоке данных для пересылки;

3) в автоинкрементный регистр адреса РА (А3) записывают адрес первого байта в блоке данных;

4) в регистр управления РУ (А1) записывают `HLD ENB`=1. Это соответствует разрешению режима ПДП.

На этом программная часть работы с контроллером заканчивается. Микропроцессор может далее выполнять какие-либо другие программы. Контроллер ПДП ожидает прихода данных от периферийного устройства.

Периферийное устройство, подготовив байт данных (D), выдает его во входной регистр данных (РД) контроллера (рис. 2.23). Байт данных сопровождается стробом данных (СД). По этому стробу происходит запись байта данных в регистр данных и установка триггера

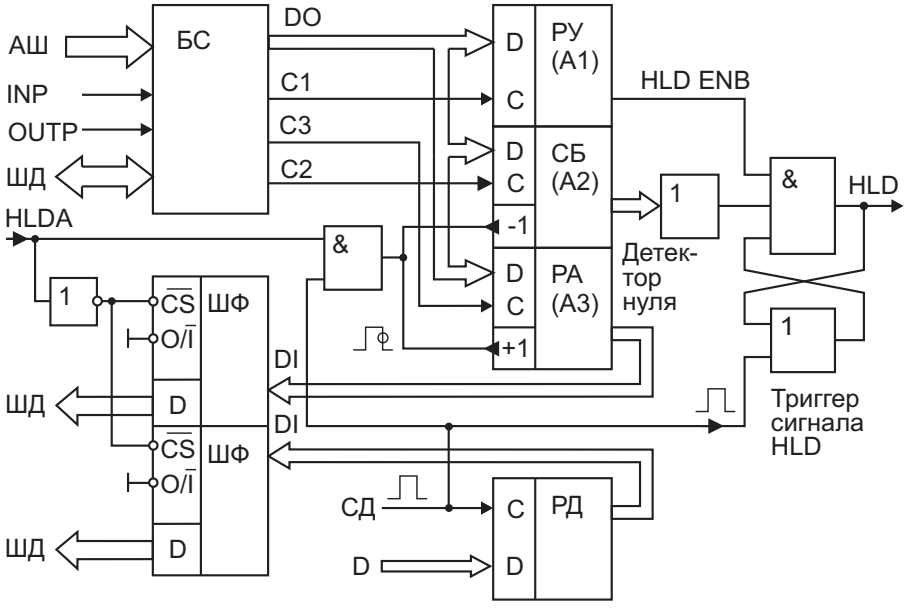


Рис. 2.23. Контроллер прямого доступа к памяти

сигнала HLD. Сигнал требования ПДП поступает на вход микропроцессора, который завершает текущий машинный цикл и выдает ответный сигнал HLDA.

Сигнал подтверждения ПДП используется для формирования сигналов $\overline{CS}$ активизации обоих шинных формирователей. Содержимое регистров адреса и данных через линии DI поступает на шины адреса и данных соответственно. Формирование сигналов записи в память на схеме не показано.

По спаду строба данных (СД) происходит декремент счетчика байтов (СБ) и инкремент адресного регистра (РА). При ненулевом содержимом счетчика байтов триггер сигнала HLD остается в установленном состоянии, требование ПДП продолжает действовать и сигнал HLDA=1. Контроллер ожидает прихода нового байта данных от периферийного устройства.

Если содержимое счетчика байтов стало нулевым, то на выходе детектора нуля (см. рис. 2.23) появится логический нуль, который сбросит триггер сигнала HLD=0. Процессор снимет сигнал HLDA и подключится к системному интерфейсу.

Для реализации режимов ПДП разработаны и серийно выпускаются БИС контроллеров ПДП, которые благодаря программированию могут обслуживать ПДП с учетом конкретных требований различных систем. Примером КПДП может служить БИС Intel 8237A (К580ВТ57), имеющая 4 независимых канала ПДП и возможность каскадирования схем до любого числа каналов. Контроллеры ПДП позволяют организовать обмен типа “память-память”, т.е. решать задачу перемещения блоков данных в адресном пространстве системы. Частным случаем этой задачи является, например, начальная загрузка памяти программ после включения микропроцессорной системы.

2.5. Программно-управляемый обмен данными по последовательному каналу

С увеличением расстояния обмен данными по параллельному каналу становится неприемлемо сложным и экономически невыгодным. В этом случае применяют преобразование параллельного кода в последовательный для его передачи по одной сигнальной линии.

Кроме того, последовательный канал характеризуется сравнительно небольшим числом сигнальных линий. Это дает ему определенные преимущества, например, при интегрировании устройств в систему, даже если устройства расположены на малых расстояниях друг от друга.

Тракт передачи последовательных данных в общем случае включает в себя передатчик и приемник данных. Система передачи может

быть *симплексной*, *полудуплексной* или *дуплексной*. В первом случае данные передаются только в одну сторону, во втором – в обе, но с разделением во времени, в третьем – в обоих направлениях одновременно. Скорость передачи данных характеризуют количеством изменений состояний канала в секунду. Единицей измерения является *бод* (baud).

Реализация последовательного канала на физическом уровне может быть самой разнообразной. Основные усилия в совершенствовании физической реализации последовательного канала направлены на повышение скорости обмена данными, увеличение расстояния и помехозащищенности канала.

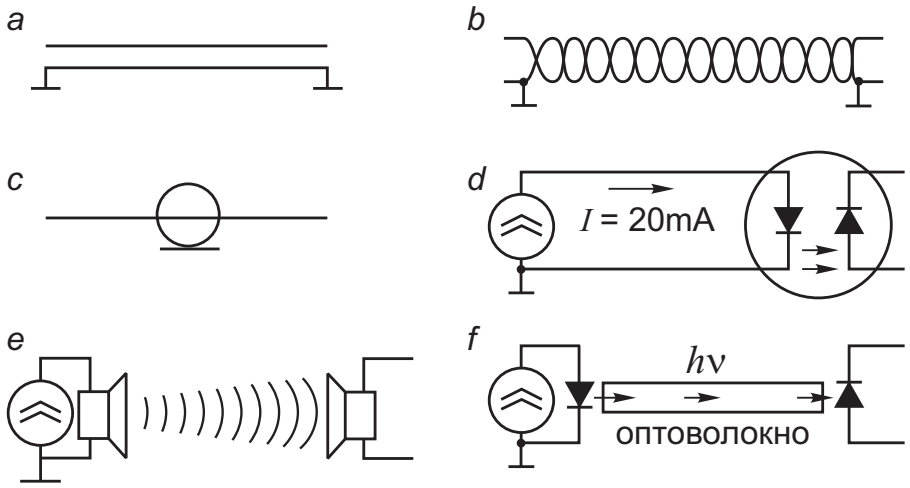


Рис. 2.24. Реализация последовательного канала на физическом уровне: *a* – двухпроводная линия; *b* – витая пара; *c* – коаксиальная линия; *d* – токовая петля; *e* – акустика или радио; *d* – оптоволоконная линия

Для правильного приема необходимо точное определение приемником моментов времени, в которые следует считывать с линии очередной бит данных. Это подразумевает установление синхронизации процессов в передатчике и приемнике. Можно связать передатчик и приемник специальной линией для синхронизации. Можно обойтись и без такой линии, применив синхронизацию приемника самим передаваемым сигналом, для чего сигналу придается определенная структура. Протоколы последовательного обмена задают два его вида: *синхронный* и *асинхронный*.

2.5.1. Синхронный обмен

При синхронной передаче байты (слова) следуют один за другим слитно, поэтому можно говорить о передаче массива байтов. Если очередной символ не готов, передача не останавливается, передатчик продолжает посылать в линию синхронизирующие импульсы. Синхронный обмен повышает скорость передачи данных. На рис. 2.25

показана схема организации синхронного обмена данными по последовательному каналу. Общепринятые обозначения сигналов и линий

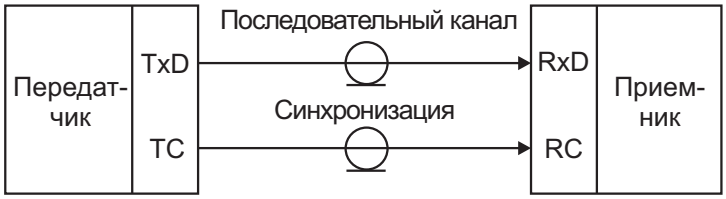


Рис. 2.25. Синхронный обмен данными по последовательному каналу

последовательного интерфейса: TxD – выход передатчика, RxD – вход приемника, TC – выход синхронизации, RC – вход синхронизации.

Различают две разновидности синхронных передач – с внутренней и внешней синхронизацией.

При *внутренней синхронизации* перед массивом данных передают слова - байты (слова) синхронизации (одно или два). При отсутствии передачи передатчик не перестает работать, а посылает в линию байты (слова) синхронизации, пока не возобновится передача данных. Приемник при этом находится в режиме активного ожидания (Hunt – охота). Он сравнивает каждое принятое слово со словом синхронизации. Если результат сравнения отрицательный, то обращения к данному приемнику нет. Слова данных не разделяются служебными битами.

При *внешней синхронизации* в канал связи добавляется дополнительная линия синхронизации (см. рис. 2.25). Временные диаграммы пересылки байта данных при внешней синхронизации показаны на рис. 2.26. Характерной особенностью синхронного режима является равенство частот модуляции (т.е. выхода битов в линию TxD) и синхронизации.

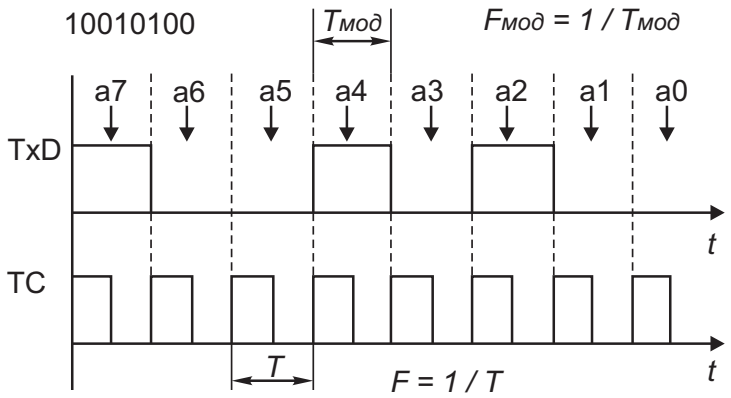


Рис. 2.26. Передача байта данных в синхронном режиме

Контроллер синхронной передачи данных

Основу контроллера для вывода данных по последовательному каналу составляет преобразователь параллельного кода в последовательный. В контроллере синхронной передачи данных (рис. 2.27) такой преобразователь организован на основе последовательно-параллельного регистра сдвига с параллельным входом и последовательным выходом (Parallel Input, Serial Output - PISO).

Большая часть схемы контроллера для синхронной передачи идентична рассмотренной ранее схеме контроллера периферийного устройства для работы по прерываниям (см. рис. 2.11). Новые узлы, до-

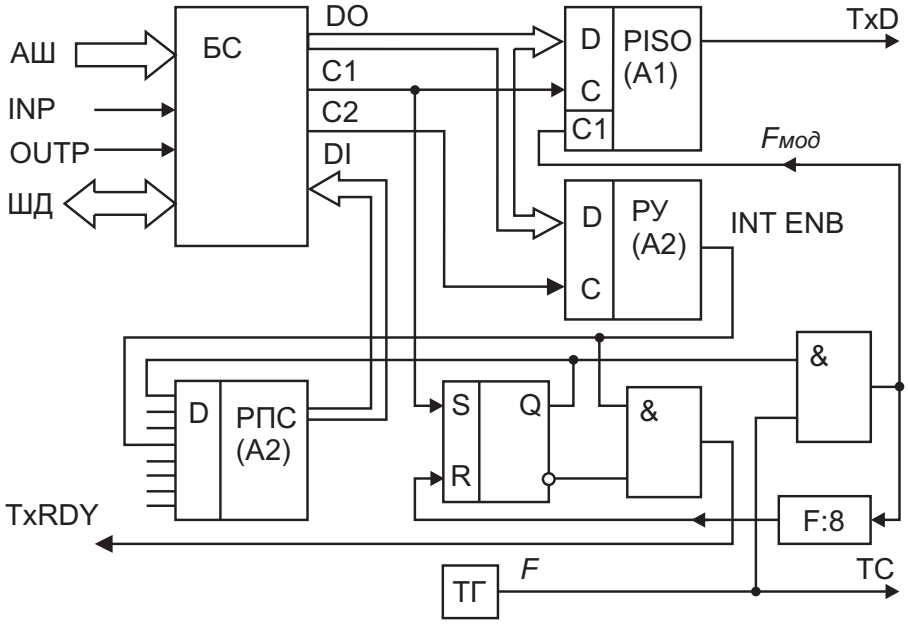


Рис. 2.27. Контроллер синхронного вывода данных

бавленные в схему контроллера: регистр сдвига PISO, параллельный вход которого подключен в качестве регистра данных (A1); тактовый генератор (ТГ) с частотой F; счетчик-делитель по модулю 8 для подсчета выданных в линию битов; логический элемент для подачи импульсов сдвига с частотой $F_{\text{мод}}$ на вход C1 регистра сдвига.

Последовательность действий контроллера при передача байта выглядит следующим образом:

1) микропроцессор выводит байт данных в порт вывода с адресом A1. Этот порт соответствует параллельному входу регистра сдвига PISO;

2) строб выхода C1 устанавливает флаговый триггер. Это соответствует загрузке регистра сдвига байтом данных;

3) единичное состояние флагового триггера поступает на логический элемент 2И, который начинает пропускать импульсы сдвига $F_{\text{мод}}$ на вход C1 регистра сдвига;

4) под действием этих импульсов регистр сдвига начинает выдавать на выход TxD последовательный код, который соответствует битам загруженного в PISO байта данных. Вывод происходит от младших битов к старшим. Количество выведенных битов фиксируется счетчиком-делителем (F:8). Этот счетчик автоматически сбрасывается каждый раз при записи в PISO нового байта данных;

5) при переполнении счетчика-делителя формируется сигнал переноса, который подается на вход R флагового триггера и сбрасывает его. Это останавливает подачу импульсов сдвига на вход регистра PISO;

6) окончание передачи байта и соответственно готовность контроллера принять следующий байт данных можно контролировать программно или аппаратно. В первом случае проводят программный опрос регистра РПС и контроль флага занятости регистра сдвига. Это соответствует выводу данных в регистр PISO по флагам. Во втором случае, при установленном в регистре РУ бите разрешения прерывания `INT ENB`, формируется сигнал `TxRDY`. Этот сигнал может быть использован как запрос на обслуживание контроллеру прерываний. Тогда вывод каждого последующего байта производится подпрограммой обслуживания прерывания.

2.5.2. Асинхронный обмен

При асинхронном обмене данные передаются по мере их готовности. Интервал между байтами (словами) может быть различным, хотя интервалы между битами в одном байте (слове) фиксированы. При отсутствии готовых данных линия простаивает.

При асинхронной передаче посылка (кадр), т.е. группа битов, отображающих передаваемое слово данных, имеет следующий формат: начало посылки отмечается нулевым старт-битом, за ним следуют биты данных младшими разрядами вперед, затем необязательный бит контроля по модулю два (бит четности/нечетности) и заканчивается посылка 1; 1.5 или 2 единичными стоп-битами (рис. 2.28). В отсутствие передачи линия находится под высоким потенциалом (активная пауза), соответствующим логической единице. Появление низкого уровня означает поступление старт-бита, свидетельствующего о последующей передаче известного заранее числа информационных битов. Далее может идти контрольный бит четности (нечетности). Стоп-бит также используется для проверки правильности передачи, но уже по другому критерию. Контролируется правильность формата посылки. Отсутствие на позиции стоп-бита высокого уровня напряжения свидетельствует об ошибке формата (кадра, обрамления). Длительность стоп-бита определяет минимальный промежуток между окончанием данного слова и началом следующего. Этот промежуток составляет

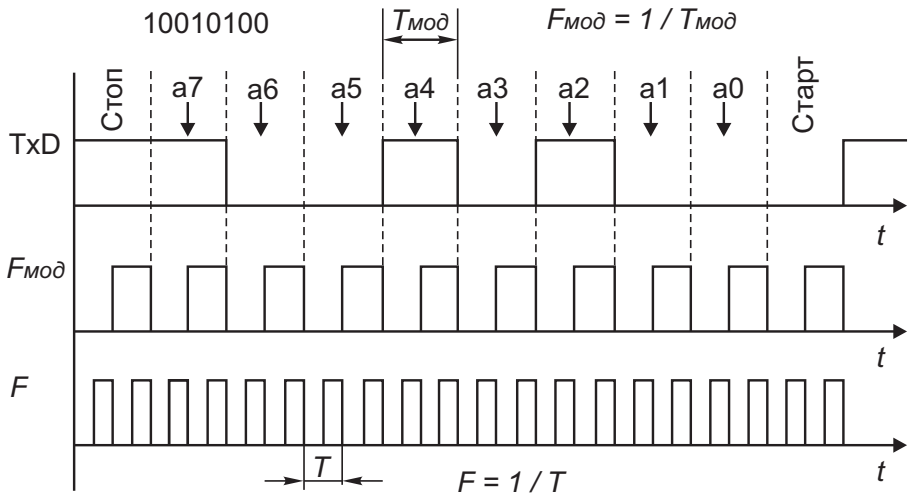


Рис. 2.28. Структура кадра для асинхронной передачи

1…2 интервала, соответствующих биту.

Приемник синхронизируется самим сигналом и должен считывать значения битов в серединах их интервалов, где искажения импульсов наименее влияют на величину считываемого уровня. Это требование достигается следующим образом. Передатчик и приемник имеют свои генераторы тактовых импульсов, работающие на одинаковой частоте. При отсутствии передачи передатчик устанавливает в линии высокий уровень напряжения (Mark). Появление нуля (старт-бита) отмечает начало передачи, которое фиксируется спадом напряжения ($1 \rightarrow 0$). От этого фронта начинает работать генератор приемника. Приемник выдерживает интервал в половину длительности бита, проверяет, есть ли еще нуль на входе (контролирует истинность старт-бита с целью исключить реакцию на кратковременную помеху), и затем начинает воспринимать данные с интервалом в длительность бита (если старт-бит не подтвердился, то приемник возвращается в исходное состояние).

Частоты генераторов передатчика и приемника реально отличаются, поэтому отсчеты постепенно “сползают” с середины битов и смещаются к тому или иному краю импульсов. Однако за время короткой посылки (не более 10…11 битов) смещение отсчетов с середины битов легко сделать приемлемо малым.

Выборка отсчетов с середины битов производится благодаря тому, что частота тактового генератора в N-раз больше частоты следования битов (обычно выбирают N = 1; 16; 64). На рис. 2.28 приведены временные диаграммы для N = 2.

Контроллер асинхронной передачи данных

Схема контроллера асинхронной передачи данных по последовательному каналу (рис. 2.29) почти идентична рассмотренной ранее

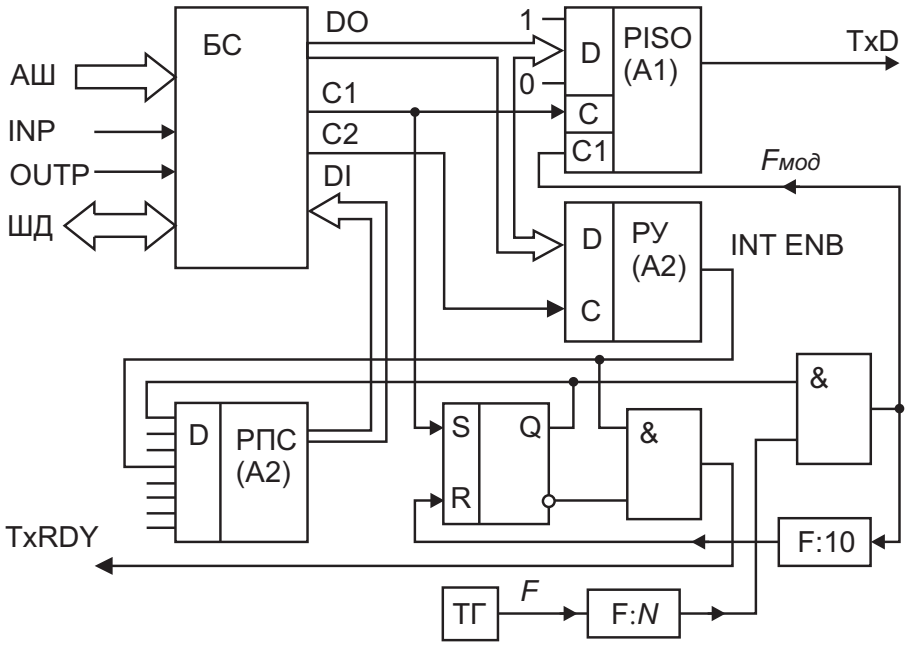


Рис. 2.29. Контроллер асинхронного вывода данных

схеме синхронного контроллера (см. рис. 2.27). Отличия состоят в следующем: регистр сдвига PISO содержит 10 разрядов, т.е. помимо 8 битов данных аппаратно формируются старт-бит (0) и стоп-бит (1); счетчик-делитель по модулю 10; дополнительный делитель частоты тактового генератора с коэффициентом деления *N*. Выход синхронизации TC отсутствует.

Для организации последовательного канала на практике используют серийно выпускаемые БИС контроллеров под названием универсальный программируемый синхронно/асинхронный приемопередатчик – УСАПП (UniversalSynchronous/Asynchronous Receiver/Transmitter – USART). Примером таких контроллеров могут служить БИС Intel 8251A (К580ВВ51А). Контроллеры, в которых реализуются только асинхронные протоколы, называются УАПП (UART).

USART/USART – универсальный синхронный/асинхронный полнодуплексный интерфейс с двумя однонаправленными несбалансированными линиями данных. Контроллер представляет собой электронное устройство, состоящее из одной интегральной микросхемы, которая может посылать и принимать данные по асинхронным последовательным каналам связи, таким как RS232. UART является программируемым устройством, поскольку в нем предусмотрены специальные управляющие сигналы, позволяющие разработчику задать параметры работы устройства, такие как скорость передачи данных, четность, количество стоп-битов и сигналы управления модемом.

2.5.3. Интерфейсы последовательного канала

Интерфейс RS232C

Интерфейс RS-232C/CCITT V24 является наиболее распространенным стандартом для построения последовательных каналов связи. Он содержит две однонаправленные несбалансированные линии для обмена данными между двумя устройствами на расстоянии до 12 м (C<2500 пФ). Скоростной режим до 115 кбит/с.

Стандарт на данный интерфейс был разработан Ассоциацией электронной промышленности (EIA). Стандартизованы физический уровень, 25-контактный разъем типа DB25 с 20 сигналами. Интерфейс предназначен для работы с оборудованием двух видов:

- терминальное оборудование (**D**ata **T**erminal **E**quipment – DTE);
- связное оборудование (**D**ata **C**ommunication **E**quipment – DCE).

Терминальное оборудование может передавать или принимать данные по последовательному каналу. Оно как бы оканчивает (to terminate) последовательную линию. Связное оборудование не является оконечным оборудованием последовательного канала, а служит для соединения (связи) терминальных устройств между собой. На рис. 2.30 приведен пример последовательной линии связи двух компьютеров (ПК) через модемы. В этом примере компьютеры являются терминальным

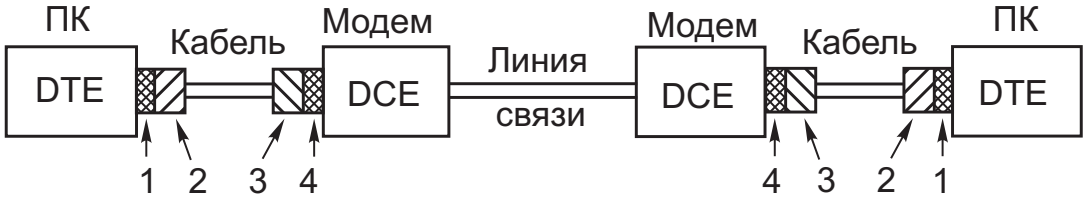


Рис. 2.30. Последовательная линия связи

оборудованием, а модемы – связным оборудованием. На терминальном оборудовании закреплены 25-контактные разъемы (штырьки) типа DB25M – (1). На связном оборудовании закреплены 25-контактные разъемы (гнезда) типа DB25F – (4). Компьютер и модем соединены *модемным кабелем*, оборудованным кабельными разъемами DB25F – (2) и DB25M – (3). Каждый провод модемного кабеля соединяет контакты с одинаковыми номерами.

Сигналы интерфейса RS-232C по функциональному признаку подразделяются на следующие классы:

- *последовательные данные* (например, TxD, RxD). Интерфейс обеспечивает два независимых последовательных канала данных: первичный (главный) и вторичный (вспомогательный). Оба канала могут работать в дуплексном режиме, т.е. одновременно осуществляют прием и передачу данных;

• *управляющие сигналы квитирования* (например, RTS, CTS). С помощью сигналов квитирования DTE-устройство начинает диалог с DCE-оборудованием до начала фактического обмена данными;

• *сигналы синхронизации* (например, TC, RC) используются при работе в синхронном режиме. На практике вспомогательный канал RS-232C применяется редко. В асинхронном режиме из 25 линий обычно используют только девять линий и соответственно применяют 9-контактные разъемы *D*-типа (табл. 2.1).

Таблица 2.1. Линии интерфейса RS-232C в 9-контактном разъеме

Контакт	Сигнал	Назначение	Направление
1	DCD	Обнаружение несущей данных	DTE ← DCE
2	RxD	Прием данных	DTE ← DCE
3	TxD	Передача данных	DTE → DCE
4	DTR	Готовность терминала	DTE → DCE
5	SG	Земля сигнальная	
6	DSR	Готовность модема	DTE ← DCE
7	RTS	Запрос передачи	DTE → DCE
8	CTS	Сброс передачи	DTE ← DCE
9	RI	Индикатор звонка	DTE ← DCE

Спецификация сигналов квитирования:

$\overline{\text{DSR}}$ (Data Set Redy) – запрос готовности передатчика терминала, сигнал связан с одноразрядным портом и может быть проверен процессором чтением слова состояния. Низкий уровень этого сигнала говорит о том, что модем имеет информацию для передачи процессору;

$\overline{\text{DTR}}$ (Data Terminal Redy) – этот сигнал является реакцией на запрос DSR. Активизируется соответствующим битом командного слова, если процессором разрешен обмен с модемом. Связан с разрешением модему посылки данных на вход терминала;

$\overline{\text{RTS}}$ (Request to send) – сигнал связан с одноразрядным выходным портом. Он является запросом готовности приемника принять данные. Задается программированием соответствующего бита в командном слове, когда процессору разрешен обмен с модемом;

$\overline{\text{CTS}}$ (Clear to Send) – сигнал готовности приемника терминала принять данные. Низкий уровень этого сигнала разрешает передачу последовательных данных. При снятии этого сигнала во время работы передатчика он будет передавать все данные, записанные до запрещения передачи, прежде чем остановится.

Для обеспечения высокой помехозащищенности в интерфейсе используют сравнительно высоковольтные уровни сигналов, отличаются от логических уровней ТТЛ:

- *логический нуль* (SPACE, зеленый): от +3 до +25 В;
- *логическая единица* (MARK, красный): от −3 до −25 В.

Преобразование ТТЛ-уровней в уровни интерфейса RS-232C и наоборот производится специальными микросхемами драйвера линии (1488) и приемника линии (1489).

Для соединения двух DTE-устройств по последовательному каналу применяют специальный “нуль-модемный кабель”. Его функция заключается в изменении сигнальных линий таким образом, чтобы превратить DTE-оборудование в DCE-устройство. На рис. 2.31 показан один из возможных вариантов организации нуль-модемного соединения с использованием 9-контактных разъемов. Для обеспечения работы интерфейса требуется соответствующий набор сигналов квитирования.

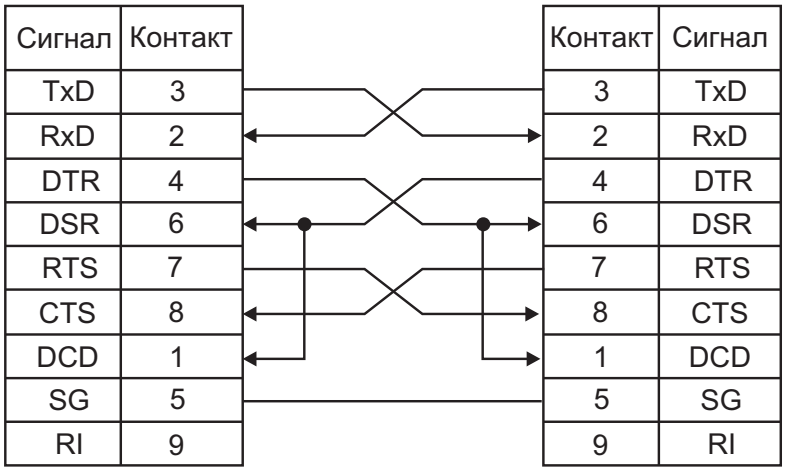


Рис. 2.31. Нуль-модемное соединение в 9-контактном варианте

В простейшем асинхронном варианте для организации последовательного канала связи без сигналов квитирования достаточно всего трех линий: `TxD`, `RxD`, `Gnd`.

Разработаны новые стандарты, направленные на устранение недостатков первоначальных спецификаций интерфейса `RS-232C`. Эти стандарты позволяют улучшить согласование линии, увеличить расстояние и повысить скорость передачи данных. Отметим среди них интерфейсы: `RS-422` (балансная система, допускающая импеданс линии до 50 Ом), `RS-423` (небалансная система с минимальным импедансом линии 450 Ом), `RS-449` (стандарт с очень высокой скоростью передачи данных, в котором несколько изменены функции схем и применяется 37-контактный *D*-разъем) и `RS-485` (двунаправленная сбалансированная линия передачи с максимальной скоростью передачи

10 Мбит/с, поддерживает многоточечные соединения до 32 узлов и передачу на расстояние до 1200 м, поддерживает полудуплексную связь – для передачи и приема данных достаточно одной витой пары).

Интерфейс SPI

SPI (**S**erial **P**eripheral **I**nterface) – полнодуплексный скоростной синхронный 4-проводной интерфейс был разработан фирмой Motorola. Этот интерфейс быстро становится промышленным стандартом, он поддержан фирмами Microchip, Motorola, Atmel и многими другими. Интерфейс SPI совместим с известным интерфейсом Microwire, разработанным в National Semiconductor. Организация системы для обмена данными по SPI интерфейсу показана на рис. 2.32. В основу интерфей-

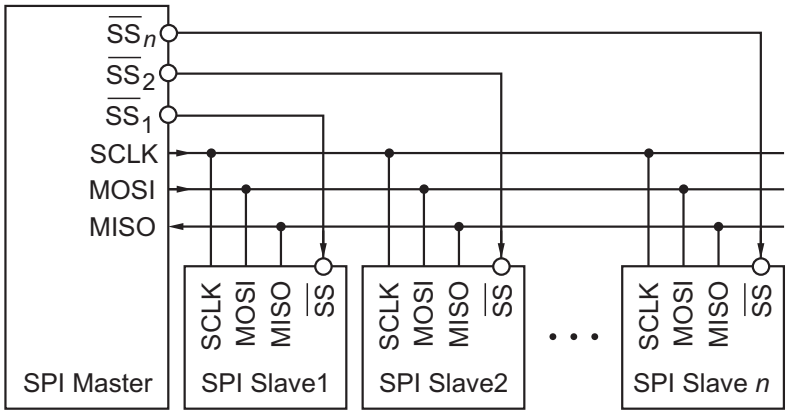


Рис. 2.32. Организация интерфейса SPI

са положена концепция ведущий-ведомый (Master-Slave). Протокол обмена не стандартизован. Назначение выводов интерфейса по спецификации фирмы Motorola:

`MISO` – (Master In, Slave Out) ведущий - вывод, ведомый - ввод;

`MOSI` – (Master Out, Slave In) ведущий - ввод, ведомый - вывод;

`SCK` – (SPI ClocK) линия тактирования (синхронизации);

$\overline{\text{SS}}$ – (Slave Select) линия аппаратного выбора ведомого устройства.

Система удобна тем, что одноименные выводы ведущего и ведомого устройств просто соединяются между собой, т.е. ведущий (MISO) – ведомый (MISO), ведущий (MOSI) – ведомый (MOSI). Международная спецификация имеет другую систему обозначений:

`SDI` — вводимые данные (MISO или MOSI);

`SDO` — выводимые данные (MISO или MOSI);

$\overline{\text{CS}}$ — выбор кристалла.

В обозначениях данной спецификации следует соединять между собой разноименные выводы, т.е. ведущий (SDO) – ведомый (SDI), ведущий (SDI) – ведомый (SDO). Интерфейс SPI поддерживает скоростные режимы в широком диапазоне до 10 Мбит/с.

Интерфейс I2C

I2C = I^2C = IIC (**I**nter-**I**ntegrated **C**ircuit) – полудуплексный 2-проводный синхронный последовательный интерфейс был разработан фирмой Philips Corporation. Организация системы для обмена данными по I^2C интерфейсу показана на рис. 2.33. Описание приведено по версии 2.1, 2000 г. В основу интерфейса также положена концепция ведущий-ведомый (Master-Slave). Спецификация выводов интерфейса I2C следующая:

`SCL` – (**S**erial **C**lock **L**ine) линия тактирования(синхронизации);

`SDA` – (**S**erial **DA**ta Line) линия передачи-приема данных.

Доступ к ведомым устройствам осуществляется программно по 7-битному адресу, можно адресовать 127 устройств, а один адрес зарезервирован. Оставшийся бит адреса выполняет роль флага “Чтение-Запись”.

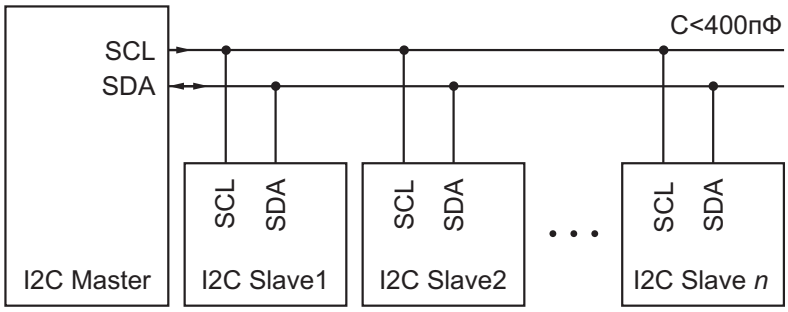


Рис. 2.33. Организация интерфейса I2C

Области применения: связь кристаллов БИС между собой в микропроцессорной системе. Скорости не экстремально большие, но вполне достаточны для аудио/видео потоков данных. Линия протяженностью до 2 м. Интерфейс поддерживает скоростные режимы: стандартный – 100 Кбит/с; быстрый – 400 Кбит/с; высокоскоростной – 3.4 Мбит/с.

Интерфейс CAN

CAN (**C**ontrol **A**rea **N**etwork) является промышленным стандартом (ISO 11898/11519, CAN версия 2.0). Он предназначен для организации высоконадежных недорогих каналов связи в распределенных системах управления. Интерфейс широко применяется в промышленности, энергетике и на транспорте. Позволяет строить как дешевые мультиплексные каналы, так и высокоскоростные сети. Представляет из себя дифференциальный двухпроводный асинхронный старт-стопный интерфейс. Организация интерфейса CAN показана на рис. 2.34. Интерфейс с применением протокола CAN легко адаптируется к физической среде передачи информации. Это может быть дифференциальный сигнал, оптоволокно, просто открытый коллектор и т.п. Несложно дела-

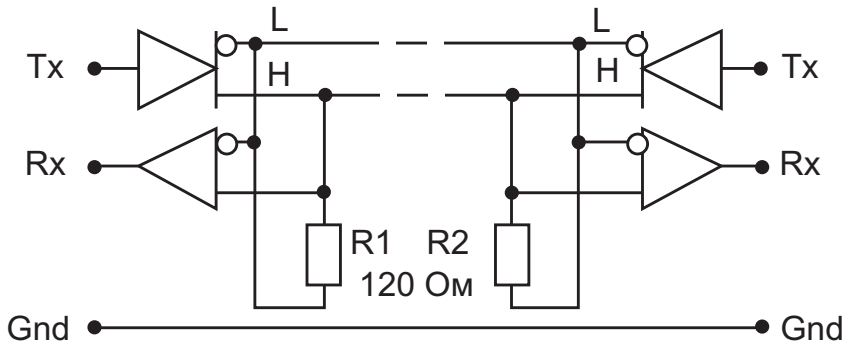


Рис. 2.34. Организация интерфейса CAN

ется гальваническая развязка. В качестве последовательного канала допускается использовать экранированную витую пару (STP), неэкранированную витую пару (UTP) или простую двухпроводную линию (Ribbon cable). Элементная база, поддерживающая CAN, широко выпускается в индустриальном исполнении. Информация одновременно поступает на все узлы сети. Имеется стандартизованный протокол обмена и распознавания устройств. Максимальное расстояние передачи данных зависит от скорости обмена. Интерфейс поддерживает скоростные режимы: минимум – 10 Кбит/с (до 1 км); стандарт – 20 Кбит/с; скоростной – 1 Мбит/с (до 40 м).

Уровни напряжения: высокий (H): 2.75 – 4.5 В, низкий (L): 0.5 – 2.25 В, разность: 1.5 – 3.0 В. Ток выхода – 100 мА. Кодирование логических уровней осуществляется по схеме Non-Return-to-Zero (NRZ): логический нуль, если было изменение уровня; логическая единица, если значение уровня сохраняется.

Каждый узел содержит закрепленный на корпусе 9-контактный разъем `9DBM` (штырьки). В лабораторных условиях допустимы также другие виды разъемов. Например, часто используют сравнительно дешевый 8-контактный разъем `RJ-45`.

Максимальное число абонентов, подключенных к данному интерфейсу, определяется нагрузочной способностью примененных приемопередатчиков. Например, при использовании *трансивера* фирмы PHILIPS PCA82C250 она равна 110.

Протокол CAN использует оригинальную систему адресации сообщений. Каждое сообщение снабжается идентификатором, который определяет назначение передаваемых данных, но не адрес приемника. Любой приемник может реагировать как на один идентификатор, так и на несколько. На один идентификатор могут реагировать несколько приемников.

Протокол CAN обладает развитой системой обнаружения и сигнализации ошибок. Для этих целей используется поразрядный контроль, прямое заполнение битового потока, проверка пакета сообще-

ния CRC-полиномом, контроль формы пакета сообщений, подтверждение правильного приема пакета данных. Хемминговый интервал d=6. Общая вероятность необнаруженной ошибки 4.7×10^{-11}. Система арбитража протокола CAN исключает потерю информации и времени при “столкновениях” на шине.

Интерфейс USB

USB (**U**niversal **S**erial **B**us) разработан совместно фирмами Compaq, Digital Equipment, IBM, Intel, Microsoft и Northern Telecom. Последовательный канал USB представляет собой дифференциальную двунаправленную последовательная шину, состоящую из 4 линий: витая пара (сигналы), питание (от +4.7 до +5.2 В, ток до 500 мА) и земля. Максимальная длина до 5 м. Интерфейс USB предназначен для интегрирования периферийных устройств в систему. Организация интерфейса USB показана на рис. 2.35. В систему интерфейса USB входят

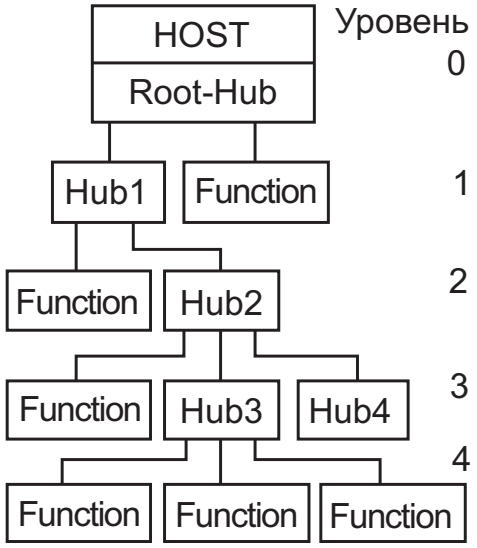


Рис. 2.35. Организация интерфейса USB

следующие виды оборудования: головное устройство (Host) – одно на всю систему, концентраторы (Hub) – можно каскадировать до третьего уровня, конечные периферийные устройства (Function) – до 127 устройств (адрес – 7 битов), подключаются к концентраторам на уровнях 0…3. Контроллер USB занимает в системе только одно прерывание независимо от количества подключенных к шине устройств. Логические уровни: 2.8 В (H), 0.3 В (L) – 1-й и 2-й режимы; 400 мВ (H) - 0 В (L) – скоростной режим. Импеданс кабеля – 90 Ом. Используется NRZI-кодирование логических уровней (нуль – изменение уровня, единица – сохранение уровня). Применяют специальные USB-разъемы: тип A – для подключения со стороны концентратора или головного оборудования и тип B – для подключения со стороны периферийного уст-

ройства. Скорости передачи (версия 2.0): 10 кбит/с – 100 кбит/с (Low-speed); 500 кбит/с – 10 Мбит/с (Full-speed); 25 Мбит/с – 400 Мбит/с (High-speed).

Все передачи данных по интерфейсу инициируются головным устройством (Host). Данные передаются в виде пакетов. В интерфейсе USB используется несколько разновидностей пакетов:

Token Paket – пакет-признак описывает тип и направление передачи данных, адрес устройства и порядковый номер адресуемой части USB-устройства (конечная точка). Пакеты-признаки бывают нескольких типов: IN, OUT, SOF, SETUP;

Data Packet – пакет с данными содержит передаваемые данные;

Handshake Packet – пакет согласования предназначен для сообщения о результатах пересылки данных; пакеты согласования бывают нескольких типов: ACK, NAK, STALL.

Таким образом, каждая транзакция состоит из трех фаз: фаза передачи пакета-признака, фаза передачи данных и фаза согласования.

В USB используются несколько типов пересылок информации:

• *управляющая пересылка* (Control Transfer) – используется для конфигурации устройства, а также для других специфических для конкретного устройства целей;

• *потоковая пересылка* (Bulk Transfer) – используется для передачи относительно большого объема информации;

• *пересылка с прерыванием* (Iterrupt Transfer) – используется для передачи относительно небольшого объема информации, для которого важна своевременная его пересылка. Имеет ограниченную длительность и повышенный приоритет относительно других типов пересылок;

• *изохронная пересылка* (Isochronous Transfer) – также называется потоковой пересылкой реального времени. Информация, передаваемая в такой пересылке, требует реального масштаба времени при ее создании, пересылке и приеме.

В связи с тем, что в интерфейсе USB реализован сложный протокол обмена информацией, в устройстве сопряжения с интерфейсом USB обязательно должен присутствовать микропроцессорный блок, обеспечивающий поддержку протокола.

3. ОДНОКРИСТАЛЬНЫЕ 8- И 16-РАЗРЯДНЫЕ МИКРОПРОЦЕССОРЫ

Микропроцессоры фирмы Intel и различные их аналоги и модификации широко применяются во всем мире. Эта фирма разработала первый микропроцессор, затем целый ряд микропроцессорных семейств. При изучении организации микропроцессоров целесообразно ориентироваться на конкретные образцы. В настоящем разделе обсуждаются однокристальные 8- и 16-разрядные микропроцессоры i8080, i8085a и i8086. Это достаточно простые микропроцессоры, на примере которых легко проследить основные принципы их работы.

3.1. Микропроцессор i8080

3.1.1. Общая характеристика

Однокристальный 8-разрядный микропроцессор с фиксированной системой команд i8080 был разработан фирмой Intel и появился в 1974 г. Впоследствии многие другие фирмы (NEC, AMD, Philips и др.) также освоили технологию производства этого микропроцессора и наладили выпуск многочисленных его клонов. На территории бывшего СССР этот микропроцессор выпускался предприятиями "Электроприбор" (с 1983 г.), "Квазар" (с 1988 г.), "Днепр" (с 1992 г.) и др. Первоначальное название микропроцессора было КР580ИК80А, т.к. ГОСТ не содержал буквенного идентификатора для обозначения микропроцессора. После введения нового ГОСТа кристалл стали маркировать как КР580ВМ80А.

Микропроцессор i8080 выполнен по *n*-МОП (6 мкм) технологии на Si–кристалле площадью 30 мм^2 (4.16×4.85 мм^2) и содержит в своем составе около 5 тыс. транзисторов. Тактовая частота составляет 2 МГц номинально (максимальная – до 3 МГц). Кристалл размещен в пластмассовом или керамическом корпусе типа DIP (Dual In-line Package) с 40 выводами. Для работы микропроцессора необходимы три уровня электропитания с общей мощностью 1.25 Вт: U_{cc}^1 = +5.24 В, U_{cc}^2 = +12 В и U_{cc}^3 = −5 В.

Микропроцессор характеризуется 16-разрядной адресной шиной (АШ) и байтовой структурой памяти. Объем адресного пространства составляет 64 Кб. Выделенная шина данных имеет 8 битов. При помощи внешних БИС микропроцессорного комплекта возможно организовать векторные прерывания, прямой доступ к памяти и программно-

управляемый обмен данными через 256 портов ввода-вывода. Микропроцессор имеет средства для организации разделенных адресных пространств ввода-вывода и основной памяти.

Микропроцессоры обычно выпускаются в составе микропроцессорного комплекта, куда помимо самого микропроцессора входят различные вспомогательные и интерфейсные БИС. Так, в состав микропроцессорного комплекта КР580 входят следующие БИС:

КР580ВМ80А – центральный микропроцессор;
КР580ВК28 – системный контроллер;
КР580ВВ55 – программируемый параллельный интерфейс;
КР580ВВ51 – программируемый последовательный интерфейс;
КР580ВН59 – программируемый контроллер прерываний;
КР580ВИ53 – программируемый 3-канальный таймер;
КР580ВТ57 – программируемый 4-канальный контроллер ПДП;
КР580ВГ75 – программируемый контроллер видеотерминала;
КР580ВГ79 – программируемый контроллер клавиатуры и дисплея;
КР580ГФ24 – генератор тактовых импульсов;
КР580ВА86 – 8-битный шинный формирователь;
КР580ВА87 – инвертирующий 8-битный шинный формирователь;
КР580ИР82 – 8-битный буферный регистр;
КР580ИР82 – 8-битный инвертирующий буферный регистр.

3.1.2. Структурная схема микропроцессора

Структурная схема микропроцессора i8080 показана на рис. 3.1, обозначение выводов – на рис. 3.2. Микропроцессор имеет восьмиразрядную (внутреннюю) шину данных, через которую функциональные узлы обмениваются информацией. На схеме приняты следующие обозначения:

А (Accumulator) – регистр-аккумулятор, выполненный на двухступенчатых триггерах и способный хранить одновременно два слова (один из операндов и результат операции);
Т (Temporary) – регистр вре́менного хранения одного из операндов;
АЛУ – арифметико-логическое устройство, выполняющее действия над двумя операндами, подаваемыми на его входы из регистров `A` и `T`. АЛУ непосредственно выполняет лишь операции сложения, вычитания, сдвига, сравнение операндов, поразрядные логические операции (конъюнкция, дизъюнкция, сложение по модулю 2);
F – регистр признаков (флагов), т.е. битов, указывающих признаки результатов арифметических или логических операций, выполненных в АЛУ. Указываются пять признаков: **Z** (Zero) – нулевой

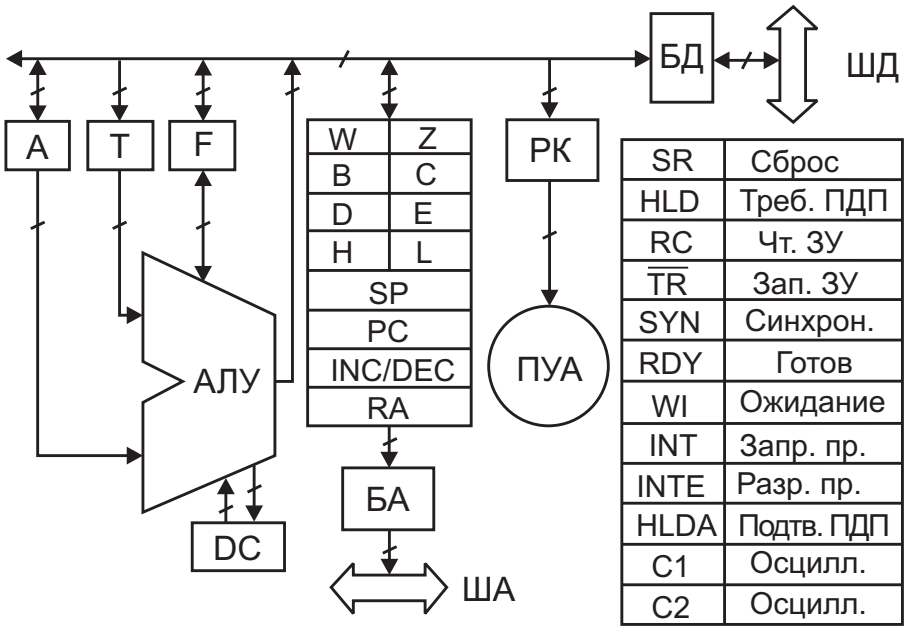


Рис. 3.1. Структурная схема микропроцессора i8080

результат, **C** (Carry)– перенос (или заем), **AC** (Auxiliary Carry) – вспомогательный перенос, **S** (Sign) – знак, **P** (Parity) – четность веса слова. Признак вспомогательного переноса (переноса между младшей и старшей тетрадами восьмиразрядного слова) нужен при выполнении операций над числами в двоично-десятичном коде. Смысл остальных признаков ясен из их наименования. Признаки служат для управления ходом процесса обработки информации. Спецификация битов регистра признаков:

a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
S	Z	0	Ac	0	P	1	Cy

DC – схема десятичной коррекции результата сложения. Используется для выполнения операции сложения чисел в двоично-десятичном коде;
РК – регистр команд (кода операции);
ПУА – первичный управляющий автомат;
БА и БД – буферы адреса и данных соответственно.

Блок регистров

С внутренней шиной данных через мультиплексор связан блок регистров общего назначения и специальных функций. Регистры обозначены буквами `W`, `Z`, `B`, `C`, `D`, `E`, `H`, `L`, `SP` и `PC`. Регистры `W` и `Z` предназначены только для временного хранения данных при выборке команды из памяти. Пользователю они программно недоступны. Регистры `B`, `C`, `D`, `E`, `H` и `L` являются регистрами общего назначения. Эти восьмиразряд-

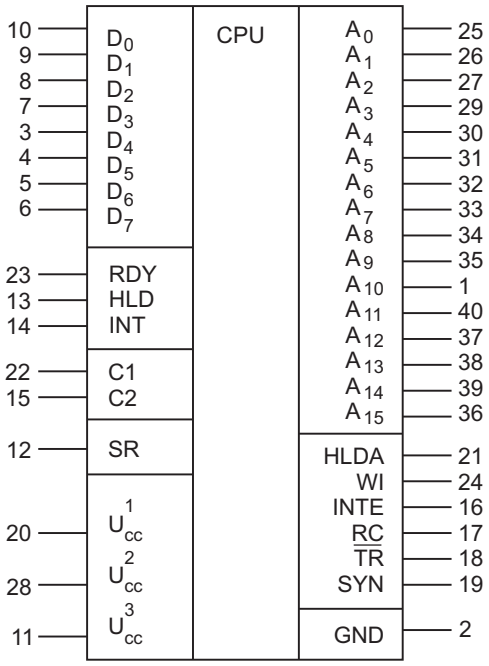


Рис. 3.2. Обозначение микропроцессора i8080 на принципиальной схеме

ные регистры могут применяться либо по отдельности, либо в составе регистровых пар BC, DE, HL. Пары регистров именуются по первым (старшим) регистрам пары: B, D и H. Пара регистров HL, как правило, используется для размещения в ней адреса при косвенной адресации. В блоке регистров имеются также 16-разрядные регистры и PC.

Регистр SP (Stack Pointer) – указатель стека. Аппаратно стек реализуется в ОЗУ, где для него выделяется определенная область. Указатель стека хранит адрес последней занятой ячейки памяти. В микропроцессоре i8080 стек является перевернутым: при записи данных в стек содержимое указателя стека уменьшается. Иными словами, стек растет в направлении от больших адресов к меньшим. Основное назначение стека – обслуживание процесса передачи управления подпрограмме.

Регистр PC (Program Counter) – счетчик команд содержит адрес следующего кода операции для выборки. При сбросе микропроцессора счетчик команд обнуляется. Содержимое счетчика команд после выборки очередного байта кода операции из памяти автоматически инкрементируется. При выборке многобайтной команды второй и третий байты команды поступают в регистры W и Z, которые не адресуются программно, а используются только блоком внутреннего управления.

Схема INC/DEC изменяет передаваемые через нее байты на +1 и −1 соответственно. Такие операции называются инкрементом и декрементом.

Регистр команд РК принимает из памяти первый байт кода операции, который поступает на вход первичного управляющего автома-

та ПУА. После декодирования ПУА формирует последовательность микрокоманд, необходимых для выполнения операции. Первичный управляющий автомат реализован на базе схем с фиксированными логическими функциями и выполняет роль устройства управления микропроцессора.

Сигналы синхронизации и управления

Блок управления и синхронизации обменивается с внешними устройствами специальными сигналами (см. рис. 3.1).

SR – вход сигнала сброса микропроцессора в начальное состояние. Сигнал может поступить в любое время по команде оператора. Автоматически формируется при включении питания. Под его воздействием сбрасываются регистры `PC` и `PK`, триггеры разрешения прерываний `INTE`, подтверждения ПДП `HLDA` и т.п.;

C1, C2 – на эти выводы подаются сигналы внешнего тактового генератора, т.к. микропроцессор i8080 не имеет встроенного тактового генератора. В качестве внешнего тактового генератора обычно используют микросхему КР580ГФ24, входящую в микропроцессорный комплект;

SYN – выход сигнала для синхронизации микропроцессорной системы;

$\overline{\text{RC}}$ – выход сигнала чтения из основной памяти системы;

TR – выход сигнала записи в основную память системы;

RDY – входной сигнал, показывающий, что память или УВВ готовы к обмену с микропроцессором. Если готовности нет, микропроцессор входит в состояние ожидания, которое может длиться любое число тактов вплоть до появления единичного сигнала `RDY`;

WI – выходной сигнал, показывающий, что микропроцессор находится в состоянии ожидания сигнала `RDY`. Сигнал автоматически снимается после прихода единичного уровня на вход `RDY`;

INT – вход запроса векторного прерывания, вызывающий генерацию сигнала `INTA`, если прерывание разрешено микропроцессору. После аппаратного сброса прием сигнала запрещается (прерывания запрещены);

INTE – выходной сигнал разрешения прерываний, показывающий состояние триггера разрешения прерывания. Сбрасывается в 0 после команды запрета прерываний DI, а также после приема сигнала прерывания INT или сигнала сброса SR;

HLD – вход сигнала требования ПДП;

HLDA – выход сигнала подтверждения ПДП. Формируется в конце текущего машинного цикла. Свидетельствует об отключении микропроцессора от системного интерфейса.

3.1.3. Тактирование и синхронизация в системе

Командный цикл (КЦ) начинается с выборки кода операции. Первый машинный цикл М1 – это выборка кода операции. В М1 микропроцессор получает первый байт команды. После этого могут быть еще один или два машинных цикла типа *чтение памяти*, поскольку команда может быть однобайтной, двухбайтной или трехбайтной.

Машинный цикл (МЦ) состоит из машинных тактов (рис. 3.3), в которых выполняются типовые действия – микрооперации. В командном цикле может быть 1…5 машинных циклов. Число тактов в различных машинных циклах – 3…5. Большинство машинных циклов содержат три машинных такта. Сигналы, реализующие тот или иной МЦ, генерируются первичным управляющим автоматом на основании информации, содержащейся в первом байте команды.

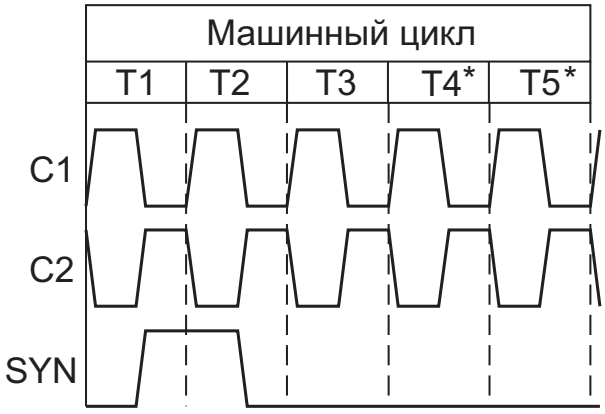


Рис. 3.3. Тактирование и синхронизация в системе

Машинный такт (МТ) равен по длительности временному интервалу в один период колебаний тактового генератора. Каждый машинный такт соответствует нахождению управляющего автомата в одном состоянии. Исключение составляют машинные такты ожидания прихода внешнего импульса, например – сигнала `RDY`.

В машинном цикле каждый машинный такт специфицирован по выполняемой управляющим автоматом функции:

T1 – микропроцессор выставляет адрес на АШ, слово состояния процессора (ССП) – на ШД. Сигнал `SYN` идентифицирует информацию на ШД как ССП, служит стробом записи ССП во внешний регистр и идентифицирует начало каждого машинного цикла;
T2 – проверка готовности внешних устройств к обмену (ожидание сигнала RDY);
T3 – чтение или запись данных (операции обмена данными);
T4, T5 – холостые машинные такты для выполнения команды, если это необходимо.

Типы машинных циклов и слово состояния

Длительность одного и того же машинного цикла в различных командах может быть разной. Микропроцессор i8080 имеет 10 типов машинных циклов:

1) выборка кода операции – машинный цикл M1;
2) чтение из памяти;
3) запись в память;
4) Чтение из стека;
5) запись в стек;
6) ввод из УВВ;
7) вывод в УВВ;
8) подтверждение прерывания;
9) останов;
10) прерывания в режиме останова.

Цикл любой команды формируется комбинацией одного или нескольких машинных циклов. Комбинация перечисленных выше 10 машинных циклов обеспечивает выполнение более 70 различных команд микропроцессора. В начале каждого машинного цикла на ШД выдается однобайтное слово состояния процессора и формируется сигнал SYN. Каждому машинному циклу соответствует свое значение ССП (табл. 3.1).

Таблица 3.1. Спецификация ССП по машинным циклам

d_i	Сигнал состояния	Машинные циклы									
		1	2	3	4	5	6	7	8	9	10
0	INTA	0	0	0	0	0	0	0	1	0	1
1	$\overline{WO}$	1	1	0	1	0	1	0	1	1	1
2	STACK	0	0	0	1	1	0	0	0	0	0
3	HLTA	0	0	0	0	0	0	0	0	1	1
4	OUT	0	0	0	0	0	0	1	0	0	0
5	M1	1	0	0	0	0	0	0	1	0	1
6	INP	0	0	0	0	0	1	0	0	0	0
7	MEMR	1	1	0	1	0	0	0	0	1	0

Каждый бит ССП специфицирован по выполняемой функции:

D0 – INTA, сигнал подтверждения прерывания. Используется для разрешения выдачи на ШД первой команды после прерывания при активном сигнале INT;

D1 – $\overline{WO}$, указывает, что операция в текущем цикле цикле является операцией записи (при D1 = 0) или чтения (при D1 = 1);
D2 – STACK, означает наличие на шине адреса содержимого указателя стека;
D3 – HLTA, сигнал подтверждения команды останова HLT;
D4 – OUT, указывает, что в текущем цикле выполняется операция вывода;
D5 – M1, указывает, что текущий цикл служит для выборки первого байта команды;
D6 – INP, указывает, что в текущем цикле выполняется операция ввода;
D7 – MEMR, указывает, что в текущем цикле производится чтение памяти.

Формирование системных управляющих сигналов

Набор управляющих сигналов: чтение из памяти ($\overline{MEMR}$), запись в память ($\overline{MEMW}$), ввод данных ($\overline{INP}$), вывод данных ($\overline{OUTP}$), подтверждение прерывания ($\overline{INTA}$) – обеспечивает прием и передачу данных между микропроцессором, памятью и периферийными устройствами в определенные интервалы времени в соответствии с диаграммой переходов машинного цикла. Данные сигналы непосредственно не формируются микропроцессором i8080, для их формирования используются сигналы RC, $\overline{TR}$ и необходимые признаки из слова состояния (рис. 3.4). Слово состояния загружается в 8-битный регистр посредством сигнала синхронизации SYN и тактирующего сигнала C2, формируемого генератором тактовых импульсов. В качестве регистра ССП используют БИС многорежимного буферного регистра, входя-

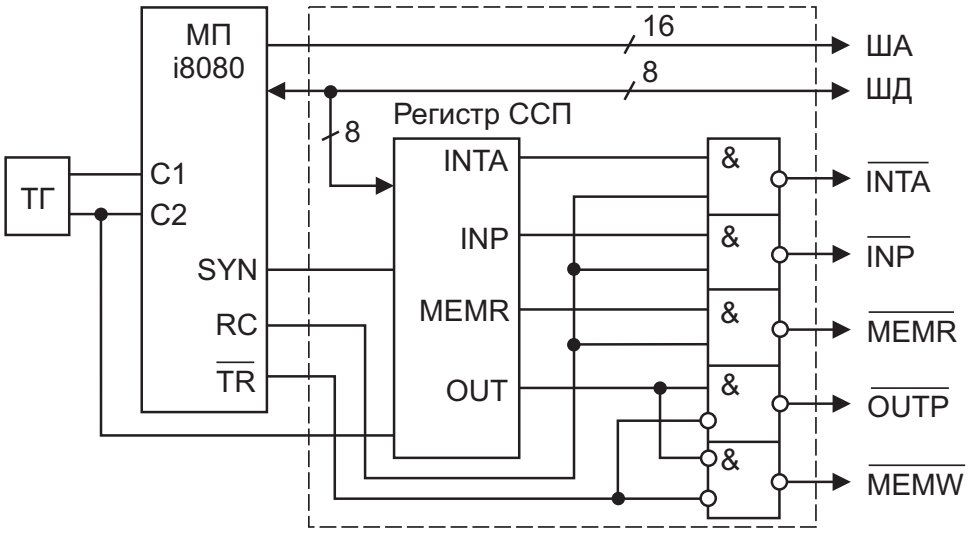


Рис. 3.4. Формирование системных управляющих сигналов

щую в состав микропроцессорного комплекта. Для повышения нагрузочной способности линий ША и ШД могут быть использованы также БИС шинных формирователей. Часть схемы (см. рис. 3.4), окруженная штриховой линией, выполняет функции системного контроллера. При практической реализации микропроцессорной системы системный контроллер выполняют обычно на базе специализированной БИС системного контроллера КР580ВК28. Остальные сигналы шины управления (INT, HLD, HLDA, SR и т.п.) снимаются непосредственно с выводов микропроцессора.

Алгоритм работы устройства управления в МЦ типа М1

Алгоритм работы первичного управляющего автомата в машинных циклах типа М1 показан на рис. 3.5. К машинным циклам типа М1 относят циклы М1…М7. На диаграмме использованы следующие условные обозначения: Tw – состояние ожидания микропроцессора, Twh – состояние ожидания при выполнении команды останова HLT, T1…T5 – состояния микропроцессора, соответствующие машинным тактам. Внутренние триггеры `HLTA`, `INTA` и `HLDA` – триггеры признаков подтверждения останова, прерывания и прямого доступа к памяти соответственно.

Работа управляющего устройства начинается с подачи сигнала сброса SR. Каждый машинный цикл содержит 3, 4 или 5 состояний в зависимости от типа выполняемой команды. Первое состояние T1 идентифицируется сигналом SYN, который используется для загрузки ССП в регистр. Кроме того, в момент времени T1 на ША выдается адрес устройства, к которому будет обращение.

В состоянии T2 микропроцессор анализирует признак подтверждения останова HLTA, сигнал готовности RDY и соответственно переходит в состояния Tw или Twh.Из состояния останова Twh микропроцессор может быть выведен сигналами SR, INT или HLD. Если имел место сигнал требования ПДП, то внутренний триггер HLDA устанавливается в состояние 1.

В состоянии T3 действия определяются типом машинного цикла. В цикле выборки микропроцессор интерпретирует код на ШД как байт кода операции, а в циклах чтения из памяти, записи в память, ввода и вывода информации осуществляется обмен данными.

Состояния T4 и T5 являются необязательными, т.к. используются для внутренних преобразований в микропроцессоре.

В конце каждого машинного цикла производятся три проверки. Если был установлен триггер `HLDA`, то предоставляется режим ПДП. Если данный машинный цикл является последним в цикле команды, то проверяется наличие сигналов INT=1 и разрешения прерываний INTE=1. При наличии обоих сигналов устанавливается триггер `INTA`.

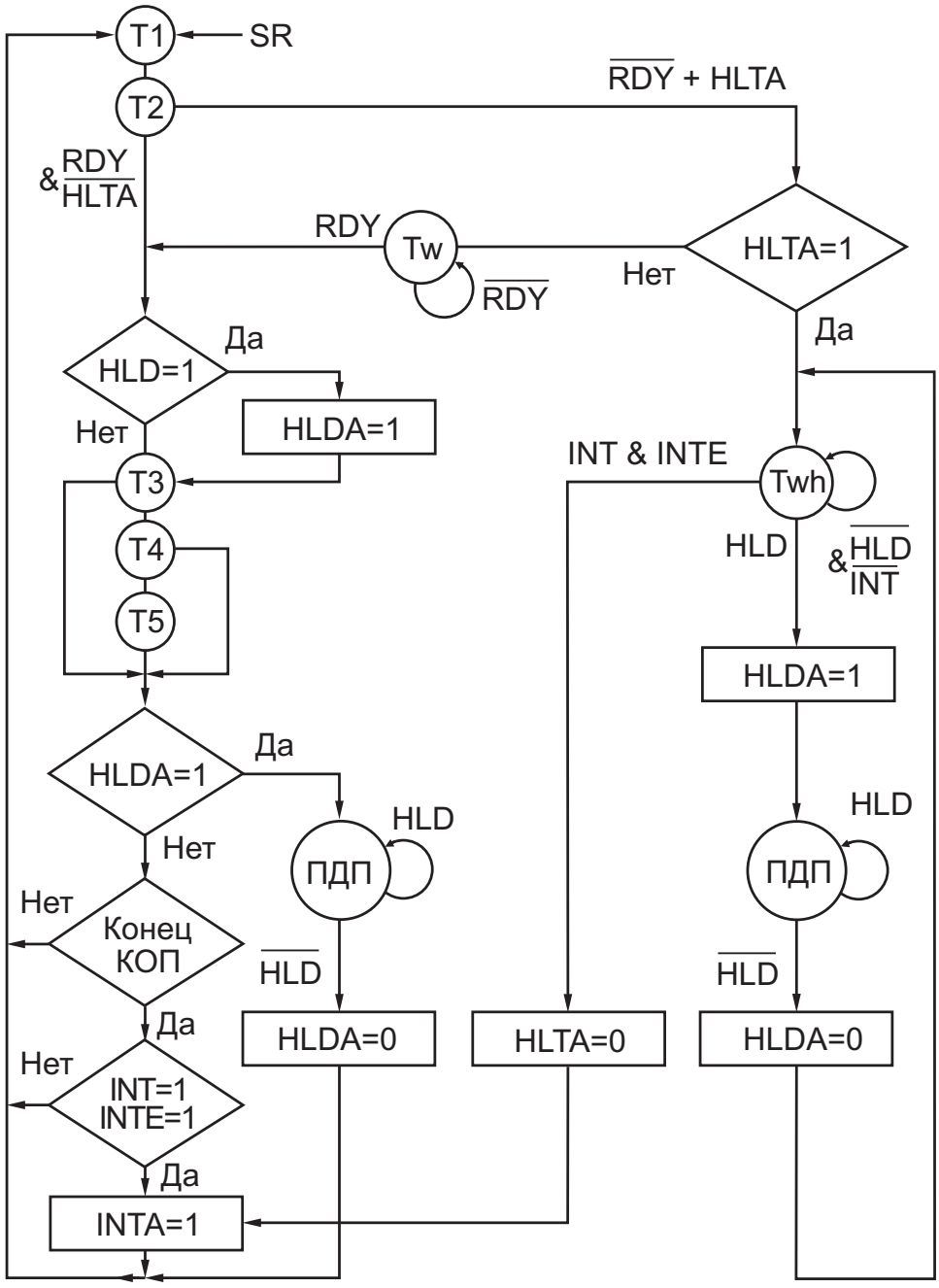


Рис. 3.5. Алгоритм работы первичного управляющего автомата

При установленном триггере `INTA` следующий машинный цикл будет M8 – машинный цикл прерывания.

Из диаграммы (см. рис. 3.5) видно, что сигнал RDY является наиболее приоритетным внешним сигналом. Следующим по уменьшению приоритета является сигнал HLD, который проверяется в каждом машинном цикле. Режим ПДП может быть предоставлен по завершении любого машинного цикла. Наиболее низкий приоритет у сигнала INT. Проверка этого сигнала производится только в конце командного цикла, и при INTE=1 прерывание может быть подтверждено только по завершении текущей команды.

В качестве примера рассмотрим командный цикл выполнения операции IN PORT ввода данных из порта с адресом PORT. Данная команда содержит два байта: первый байт – код операции, второй байт – адрес порта. Командный цикл (рис. 3.6) состоит из трех машинных

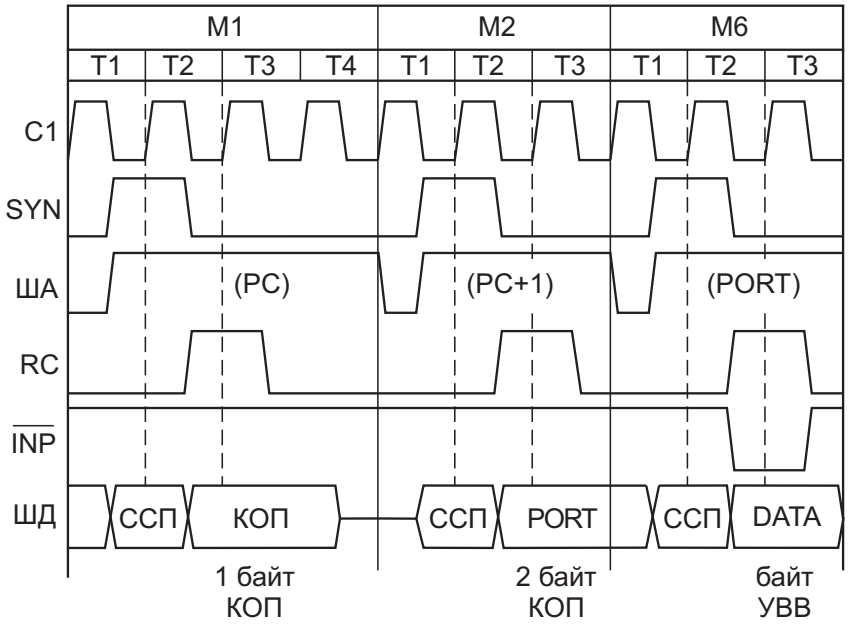


Рис. 3.6. Выполнение команды IN PORT

циклов и занимает 10 машинных тактов. Результатом выполнения команды является пересылка байта данных (DATA) из порта в аккумулятор. В первом машинном цикле (M1) производится выборка кода операции из основной памяти. Машинный такт T4 предназначен для декодирования КОП. В результате декодирования определяется, в частности, что команда содержит второй байт. Для его чтения предназначен второй машинный цикл (M2). Второй байт команды представляет собой адрес порта ввода (PORT). В третьем машинном цикле (M6) на адресную шину выставляется адрес порта (PORT) и выполняется ввод данных из адресуемого порта.

Работа управляющего автомата в режиме прерывания

Периферийное устройство инициирует прерывания путем формирования единичного уровня на входе INT микропроцессора. Сигнал запроса прерывания может возникнуть в любой момент времени, в любом машинном цикле команды. Из диаграммы переходов (см. рис. 3.5) видно, что текущая команда сначала завершится и только после этого микропроцессор переходит к машинному циклу подтверждения прерывания, который имеет ряд отличий от цикла выборки команды. Во-первых, в слове состояния дополнительно к признаку машинного цикла выборки кода операции содержится признак INTA, подтверждающий, что сигнал прерываний воспринят микропроцессором. Во-

вторых, счетчик команд не инкрементируется в данном цикле, т.к. увеличенный на единицу после выборки предыдущей команды он должен быть запомнен в стеке без изменения. В-третьих, вместо команды из памяти программ на шину данных передается вектор прерываний.

На рис. 3.7 показана работа системы прерываний с использованием вектора прерываний в виде однобайтной команды `RST N`. Сигнал запроса прерываний фиксируется в триггере INTA (диаграмма 10) лишь при одновременном выполнении трех условий: единичный уровень сигнала окончания командного цикла и возврата к машинному ци-

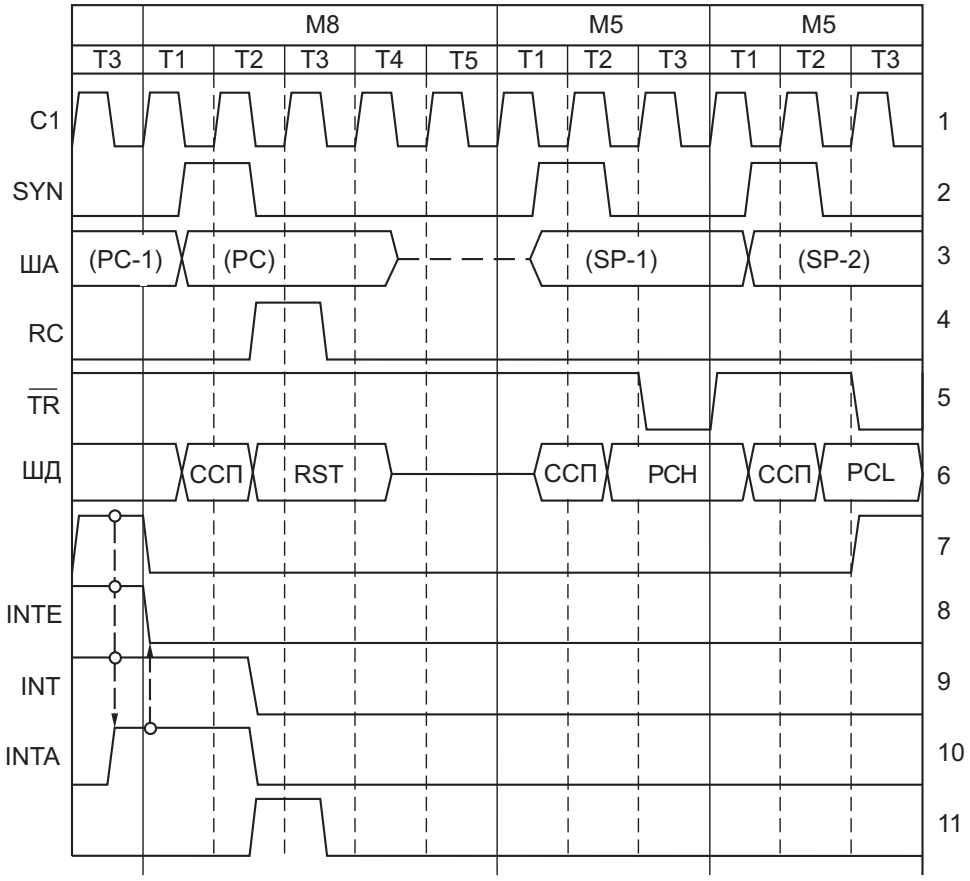


Рис. 3.7. Работа управляющего автомата в режиме прерываний

клу M1 (диаграмма 7), единичный уровень триггера разрешения прерываний INTE (диаграмма 8) и единичный уровень сигнала запроса прерывания INT (диаграмма 9). При INTA=1 следующий машинный цикл будет не M1, а M8 – машинный цикл обработки прерывания.

В микропроцессорной системе типовой конфигурации машинный цикл M8 имеет следующие особенности:

• из-за наличия внутреннего сигнала запрета (диаграмма 11) счетчик команд PC не инкрементируется;

• сигнал чтения памяти $\overline{\text{MEMR}}$ при RC=1 не формируется, т.к. в слове состояния машинного цикла M8 бит MEMR=0 (см. рис. 3.4);

• код операции на ШД выставляется контроллером периферийного устройства, а не памятью программ;

• триггер разрешения прерываний сбрасывается (INTE=0), т.е. запрещается обслуживание других прерываний;

• триггер INTA сбрасывается после формирования слова состояния с битом d_0=1 (INTA).

Следующие два машинных цикла имеют тип M5 – запись в стек. Они служат для сохранения точки возврата (содержимое счетчика команд записывается в стек) и загрузки в PC адреса подпрограммы обслуживания прерываний, сформированного из вектора прерывания. Завершение процесса входа в режим прерывания фиксируется внутренним сигналом возврата к машинному циклу M1 (диаграмма 7).

В следующем машинном цикле M1 будет происходить выборка кода первой команды подпрограммы обслуживания прерывания. Отметим, что при самых благоприятных условиях процесс входа в режим прерываний занял 12 машинных тактов. При тактовой частоте 2 МГц это составит 6 мкс.

3.1.4. Система команд i8080

Программистская модель включает в себя совокупность программно-доступных узлов микропроцессора. В i8080 программно-доступными являются следующие узлы:

1) регистры однобайтовые (B, C, D, E, H, L, A, F);

2) регистры двухбайтовые (SP, PC) и пары регистров (BC, DE, HL);

3) общее адресное пространство памяти программ и данных. Формат адреса основной памяти – 2 байта;

4) адресное пространство устройств ввода-вывода. Формат адреса устройства ввода - вывода – 1 байт.

Для доступа к этим узлам поддерживаются пять видов адресации операндов:

1) *неявная* (подразумеваемая) адресация. В этом случае адрес фиксирован на микрокомандном уровне. Программист не имеет средств для его модификации.

2) *регистровая* адресация. Операнд или оба операнда хранятся в регистрах микропроцессора. Для доступа к регистрам используют короткие двух- и трехбитовые адреса для внутреннего доступа (табл. 3.2).

3) *непосредственная* адресация. Данные (непосредственный операнд) размещены в памяти программ следом за кодом соответствующей операции. Формат непосредственного операнда – 1 или 2 байта.

4) *прямая* адресация. Адрес операнда (1 или 2 байта) размещен в памяти программ следом за кодом соответствующей операции.

5) *косвенная* адресация. Адрес операнда размещен в специальном двухбайтовом регистре (регистр косвенного адреса). В качестве регистров косвенного адреса в i8080 обычно выступает пара регистров

HL. Некоторые команды (LDAX, STAX) позволяют для косвенного адреса использовать регистры BC и DE.

Таблица 3.2. Адреса регистров i8080

Регистры								Пары регистров			
B 000	C 001	D 010	E 011	H 100	L 101	M 110	A 111	BC 00	DE 01	HL 10	SP 11

Все множество команд микропроцессора i8080 можно подразделить на 5 групп (прил. 1):

1) команды пересылки данных, обеспечивающие пересылку данных между регистрами или памятью и регистрами. Команды данной группы не формируют признаков результатов операций;

2) арифметические команды, обеспечивающие выполнение операций сложения и вычитания, инкремента и декремента. Один операнд для бинарных операций хранится в аккумуляторе, другой – в регистре или ячейке памяти. Результат помещается в аккумулятор. Система команд не содержит операций умножения и деления – их необходимо выполнять программным путем с помощью подпрограмм. Отрицательные числа при выполнении арифметических операций необходимо преобразовывать в дополнительный код;

3) логические команды реализуют операции логического сложения и умножения, исключающего ИЛИ, инвертирования, левого и правого сдвигов и некоторые другие. Исходные операнды хранятся в регистрах или ячейках памяти, а результат размещается в аккумуляторе;

4) команды передачи управления, в число которых входят команды безусловной и условной передачи управления, обращения и выхода из подпрограмм. Данные команды не формируют признаков результатов операций;

5) команды прочие, к которым отнесены команды ввода и вывода, обращения к стековой памяти. Кроме того, в эту группу входит ряд команд для управления работой микропроцессора.

По формату команды бывают одно-, двух- и трехбайтовые. Первый байт всегда содержит код операции, далее идут один или два байта, содержащие непосредственный операнд или адрес.

Возможные типы структуры первого байта команды представлены на рис. 3.8. В структурах использованы следующие условные обозначения: SSS – трехбитовый адрес регистра-источника; DDD – трехбитовый адрес регистра-приемника; rp – двухбитовый адрес пары регистров; NNN – трехбитовый номер вектора прерывания; CCC — трехбитовый код условия передачи управления. Заштрихованные биты

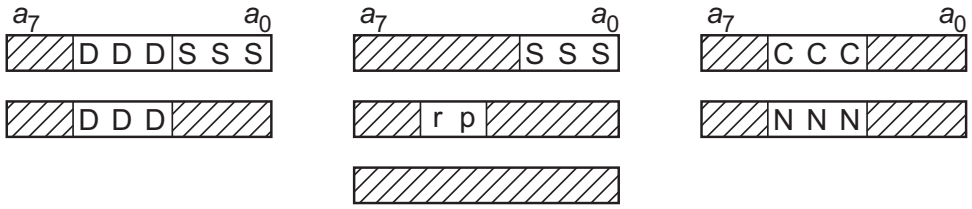


Рис. 3.8. Типы организации структуры первого байта команды

(рис. 3.8) соответствуют неструктурированной части кода операции. В табл. 3.3 приведены трехбитовые коды условий передачи управления (CCC). В системе команд i8080 используется 8 условий передачи управления по состоянию 4 флагов регистра признаков: нуля (Z), переноса (C), четности (P) и знака (S). Коды соответствуют установленным или сброшенным значениям этих флагов. В мнемоническом обозначе-

Таблица 3.3. Коды условий передачи управления

Категория	Поле cond							
	NZ	Z	NC	C	PO	PE	P	M
Код ccc	000	001	010	011	100	101	110	111
Признак	Z=0	Z=1	C=0	C=1	P=0	P=1	S=0	S=1

нии кодов условий используют символические обозначения флагов, а также буквы: **N** (Non), **O** (Odd, нечетный), **E** (Even, четный), **P** (Plus) и **M** (Minus). Буквенные коды используют для подстановки в поле `cond` команд передачи управления по условию. Так, вместо `Jcond` получают восемь команд: JNZ, JZ, JNC,…, JM.

3.2. Микропроцессор i8085

Микропроцессор i8085 – 8-битный микропроцессор, выпущенный компанией Intel в марте 1976 года. Представляет собой усовершенствованную версию процессора Intel 8080. Данный микропроцессор производится по 3-мкм технологии, это позволило уместить на кристалл, по площади равный кристаллу i8080, 6500 транзисторов. Кроме этого, данный процессор работает от единственного источника питания с напряжением +5 В, в результате чего он и получил в конце названия цифру “5” — 8085. Тактовая частота оригинального процессора i8085/i8085A/i8085AH составляла 2 МГц, были также выпущены модели с частотами 6 МГц (модель i8085A(H)-1) и 5 МГц (модель i8085A(H)-2). На кристалле нового микропроцессора располагались

также генератор синхронизации и контроллер приоритетных прерываний, позволяющий обслуживать прерывания с 4 дополнительных входов запросов прерываний. В микропроцессор i8085 были добавлены две новые команды для управления прерываниями. Система команд содержит 79 инструкций. Корпус микропроцессора i8085 такой же, как и у i8080: 40-контактный керамический или пластмассовый DIP-40.

Несмотря на многолетний возраст, этот микропроцессор продолжает выпускаться промышленностью и присутствует в каталогах ведущих фирм. Областью его применения являются системы автоматического управления аппаратурой и системы электронного взвешивания и вычисления цены товара. Отечественным аналогом i8085 является БИС К1821ВМ85А.

3.2.1. Структурная схема микропроцессора

Структурная схема микропроцессора i8085 показана на рис. 3.9. Микропроцессор имеет восьмиразрядную (внутреннюю) шину данных, через которую функциональные узлы обмениваются информацией. На схеме приняты следующие обозначения: Буфер адреса (БА) с тремя состояниями выхода выдает на линии адресной шины $A_{15}\ldots A_8$ старший байт адреса.

Буфер шины AD (БАД) с тремя состояниями выдает на шину AD с разделением во времени младший байт адреса или выдает/принимает байт данных.

Микропроцессор i8085 имеет 8 машинных циклов:

1) выборка команды (OF, Open Fetch);
2) Чтение памяти (MR, Memory Read);
3) запись в память (MW, Memory Write);
4) ввод данных (IOR, Input-Output Read);
5) вывод данных (IOW, Input-Output Write);
6) подтверждение прерывания (INA, Interrupt Acknowledge);
7) освобождение шин (BI, Bus Idle);
8) останов (HALT).

Вместо слова состояния в начале каждого машинного цикла формируются сигналы состояния (S_1, S_0), идентифицирующие тип цикла и действующие в течение всего цикла. Машинный цикл микропроцессора i8085 содержит от 3 до 6 машинных тактов. В командном цикле может содержаться от 4 до 18 тактов.

Функции выводов и сигналов:

$A_{15}\ldots A_8$ – выходные линии с тремя состояниями для выдачи старшего байта адреса памяти или полного байта адреса УВВ. Переходят в третье состояние в режимах HOLD, HALT и RESET;

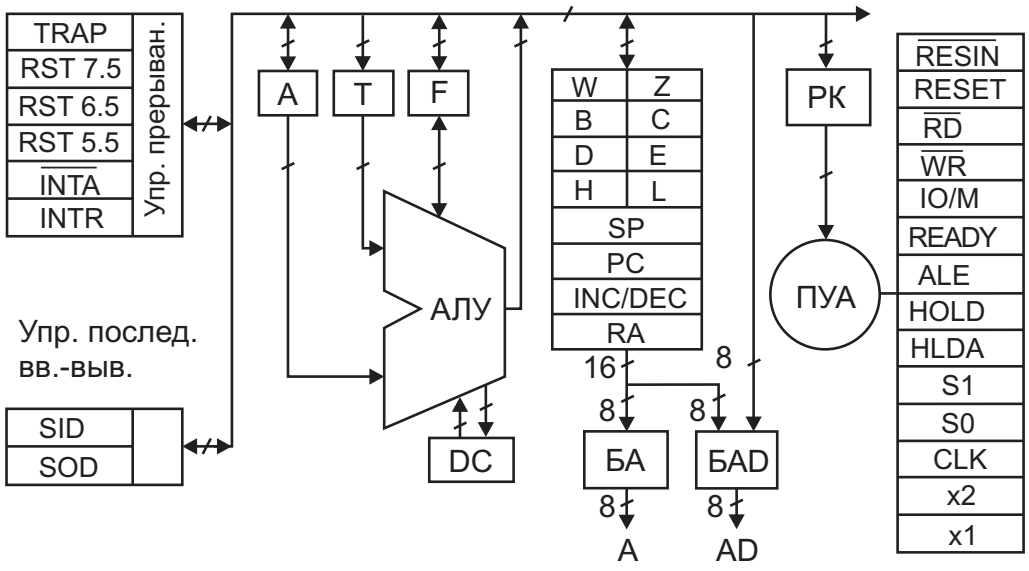


Рис. 3.9. Структурная схема микропроцессора i8085

$A_7 \ldots A_0$ – двунаправленная мультиплексированная шина с тремя состояниями для выдачи младшего байта адреса памяти или полного адреса УВВ в первом машинном такте машинного цикла, после чего используются как ШД. При адресации УВВ адресная информация обеих полушин дублируется;

ALE – строб разрешения загрузки младшего байта адреса памяти во внешний регистр для его хранения в течение машинного цикла. Появляется в первом такте машинного цикла. Регистр загружается задним фронтом сигнала ALE;

$\overline{RD}$, $\overline{WR}$ – стробы чтения и записи. Низкий уровень соответствующего сигнала свидетельствует о том, что адресованная ячейка памяти или УВВ должны выполнить операцию чтения или записи (ввода или вывода). Переходят в третье состояние в режимах HOLD, HALT и RESET;

READY – вход сигнала готовности внешнего устройства;

S_1, S_0 – сигналы состояния микропроцессора. Формируются в начале и сохраняются во время всего машинного цикла;

IO/M – сигнал выбора памяти или УВВ. При высоком уровне происходит обращение к УВВ, при низком – к памяти. Совместно с сигналами S_1 и S_0 сигнал IO/M идентифицирует тип машинного цикла (табл. 3.4);

x_1 и x_2 – эти выводы присоединяются к частотно-задающим (хронирующим) цепям для обеспечения работы внутреннего тактового генератора микропроцессора. Частота на выводах x_1 и x_2 в два раза выше рабочей частоты;

$\overline{RESIN}$ – вход сигнала сброса микропроцессора;

CLK – выход синхроимпульсов для синхронизации микропроцессор-

ной системы. Частота CLK в два раза ниже частоты на ножках x_1 и x_2;

RESET – выходной сигнал сброса для внешних модулей системы, привязан к тактовым импульсам CLK и отличается от $\overline{\text{RESIN}}$ по фазе;

INTR – (Interrupt Request) – вход запроса векторного прерывания, вызывающий генерацию строба $\overline{\text{INTA}}$, если прерывание разрешено программой. Адрес подпрограммы (вектор прерывания) выдается контроллером внешнего устройства. При сбросе прием сигнала запрещен;

$\overline{\text{INTA}}$ – (Interrupt Acknowledge) – выход строба подтверждения векторного прерывания после завершения текущего командного цикла. Используется для чтения вектора прерывания;

RST 5.5; RST 6.5; RST 7.5 – входы запросов радиального прерывания типа RST n (n = 5.5; 6.5; 7.5). Начальные адреса подпрограмм обслуживания прерываний равны $8 \times n$. Приоритеты фиксированы, высший приоритет у входа 7.5. Приоритеты всей группы запросов выше приоритета запроса INTR. Запросы маскируемые, причем независимо друг от друга;

TRAP – вход запроса немаскируемого прерывания, имеющий максимальный приоритет;

SID, SOD – (Serial Input Data, Serial Output Data) – вход и выход последовательной передачи данных. По команде RIM входной бит загружается в старший бит аккумулятора, по команде SIM – выводится из этого разряда;

HOLD – вход сигнала требования ПДП;

HLDA – выход сигнала подтверждения ПДП.

Таблица 3.4. Сигналы состояния и управления

Тип МЦ	Сигналы состояния			Сигналы управления		
	IO/M	S_1	S_0	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{INTA}}$
OF	0	1	1	0	1	1
MR	0	1	0	0	1	1
MW	0	0	1	1	0	1
IOR	1	1	0	0	1	1
IOW	1	0	1	1	0	1
INA	1	1	1	1	1	0
BI	TC	*	*	1	1	1
HALT	TC	0	0	TC	TC	1

Примечание. TC – третье состояние; * – любое состояние.

3.2.2. Система прерываний

В системе прерываний i8085 имеется пять входов для запросов прерывания (INTR, RST 5.5; RST 6.5; RST 7.5; TRAP). В табл. 3.5 приведены основные характеристики этих прерываний. Четыре линии пре-

Таблица 3.5. Параметры системы прерываний i8085

Прерывание	Вход	Вектор	Приоритет	Примечание
TRAP	Стат.	24H	0	Немаскируемое
RST 7.5	Динам.	3CH	1	Маскируемое
RST 6.5	Стат.	34H	2	Маскируемое
RST 5.5	Стат.	2CH	3	Маскируемое
INTR	Стат.	Внешний	4	Маскируемое

рываний имеют фиксированные векторы прерываний. Прерывание INTR должно ввести в действие команду CALL, код которой (вектор прерывания) формирует внешнее устройство. Вектор прерывания в этом случае загружается из контроллера внешнего устройства по стробу подтверждения прерывания $\overline{\text{INTA}}$.

При организации прерываний решаются задачи маскирования запросов и установления уровней приоритетов. *Маскирование* состоит в запрещении действия соответствующего входа. Входы запросов могут быть маскируемыми или немаскируемыми, т.е. принимаемыми всегда.

Вход `TRAP` является немаскируемым и имеет наивысший приоритет. Он не может быть запрещен командами программы. К этому входу подключают сигналы, оповещающие о наиболее важных событиях в микропроцессорной системе, появление которых требует безусловной реакции.

Прерывания по входам INTR RST 5.5, RST 6.5 и RST 7.5 являются маскируемыми, т.е. могут быть разрешены командой EI или запрещены командой DI. Эти команды действуют на все четыре входа одновременно. Начальный сброс запрещает обслуживание этих запросов. Для разрешения маскируемых прерываний необходимо выполнить команду EI.

Для раздельного маскирования запросов RST n в системе команд i8085 имеются две специальные команды SIM (Set Interrupt Mask) и RIM (Reset Interrupt Mask). Вход RST 7.5 является динамическим, реагирует на положительный фронт сигнала, а остальные входы – статические, реагируют на уровень сигнала. Запрос RST 7.5 фиксируется триггером с динамическим входом, после снятия сигнала запрос не сбрасывается и сохраняется, пока не будет обработано прерывание или до команды SIM, или RESET.

Во время обработки прерываний, пока не выполнится команда EI, запрещаются другие прерывания кроме TRAP. Немаскируемое прерывание TRAP блокирует другие прерывания, но сохраняет состояние разрешения уже поступившего сигнала прерывания.

3.2.3. Последовательный ввод-вывод

Микропроцессор имеет два вывода для передачи и приема последовательных данных: SOD (Serial Output Data) и SID (Serial Input Data).

Вывод SOD управляется командой SIM, а сигнал с вывода SID считывается командой RIM. Эти команды упоминались ранее как команды установки и сброса масок для входов прерываний RST n, они же используются и для управления последовательным вводом-выводом.

До выполнения команды SIM в аккумуляторе необходимо сформировать слово, биты которого интерпретируются согласно табл. 3.6: SOD – последовательный вывод данных; SOE (Serial Input Enable) – сигнал, единичное значение которого передает последовательные

Таблица 3.6. Спецификация битов при выполнении команд SIM и RIM

Биты аккумулятора							
7	6	5	4	3	2	1	0
SOD SID	SJE I7.5	X I6.5	R7.5 I5.5	MSE IE	M7.5 M7.5	M6.5 M6.5	M5.5 M5.5

данные SOD на одноименный вывод микропроцессора; бит 5 не используется; R7.5 сбрасывает вход RST7.5 (напомним, что сигнал по этому входу принимается триггером с динамическим управлением); MSE (Mask Set Enable) – сигнал, активное состояние которого разрешает действие битов 2…0, биты M7.5…M5.5 маскируют запросы RST7.5…RST5.5, если соответствующий бит имеет единичное значение.

Например, установка SOD=1, разрешение RST6.5, сброс триггера RST7.5 и маскирование RST7.3 и RST 5.5 будут выполнены двумя командами по программе

```
MVI A,11011101b ; установка битов аккумулятора
SIM             ; изменение масок и бита SOD
```

Для ввода последовательных данных через контакт SID используется команда RIM, обеспечивающая ввод последовательных данных и чтение масок прерывания. После выполнения команды RIM в аккумуляторе фиксируется слово согласно табл. 3.6: SID – последовательные

данные ввода через контакт SID; I7.5, I6.5 и I5.5 – логические уровни на выводах RST7.5, RST6.5 и RST5.5 соответственно; IE – сигнал разрешения прерывания; M7.5, M6.5 и M5.5 – логические уровни масок.

Биты I7.5, I6.5 и I5.5 индицируют уровни во время выполнения команды RIM. Бит IE показывает, какая из команд (EI или DI) выполнялась последней, на него также влияет наличие в данное время режима прерывания, поскольку режим прерывания сопровождается сбросом триггера IE, запрещая другие прерывания. Биты M7.5, M6.5 и M5.5 индицируют текущее состояние масок прерывания.

3.3. Микропроцессор i8086

Микропроцессор i8086 относится к классу 16-разрядных микропроцессоров первого поколения. Среди других микропроцессоров этого класса отметим i8088, i80186 и i80188.

Однокристальный шестнадцатиразрядный микропроцессор Intel 8086 появился в 1978 г. Он размещен на кристалле с геометрическими размерами 5.5×5.5 мм^2, который помещен в корпус с 40 выводами типа DIP-40. БИС i8086 содержит около 29 тыс. транзисторов и потребляет 1.7 Вт от источника питания +5 В. Тактовая частота 4.77…10 МГц. На территории бывшего СССР этот микропроцессор под названием КМ1810ВМ86 выпускался с 1988 г. различными предприятиями: “Квантор” (с 1990 г.), “Родон” (с 1991 г.), “Квазар” (с 1992 г.) и др.

Микропроцессор выполняет операции над 8- и 16-разрядными данными, представленными в двоичном или двоично-десятичном коде, может обрабатывать отдельные биты, а также строки (цепочки) или массивы данных. Он имеет встроенные аппаратные средства умножения и деления.

Микропроцессор имеет внутреннее сверхоперативное запоминающее устройство емкостью 14 шестнадцатиразрядных регистров. Шина адреса является 20-разрядной, что позволяет непосредственно адресовать 1 М ячеек памяти.

Пространство адресов УВВ составляет 64 К. В БИС i8086 реализована многоуровневая векторная система прерываний с количеством векторов до 256. Предусмотрена организация режима ПДП.

Особенностью микропроцессора i8086 является возможность частичной реконфигурации аппаратной части для обеспечения работы в двух режимах – минимальном и максимальном. Режимы работы задаются аппаратно. В минимальном режиме, используемом для построения однопроцессорных систем, микропроцессор самостоятельно формирует все сигналы управления внутренним системным интерфейсом. В максимальном режиме, используемом для построения многопроцессорных систем, микропроцессор формирует на линиях состоя-

ния двоичный код, который зависит от типа цикла шины. В соответствии с этим кодом системный контроллер (например, i8288) формирует сигналы управления шиной. Контакты, которые высвободились в результате кодирования информации, используются для управления мультипроцессорным режимом. При использовании математического сопроцессора i8087 необходимо выбирать максимальный режим.

Среднее время выполнения команды занимает 12 тактов. Микропроцессор i8086 унаследовал большую часть системы команд i8080. Все современные процессоры в обязательном порядке поддерживают набор команд процессора i8086, обеспечивая совместимость “снизу-вверх”.

3.3.1. Структурная схема микропроцессора

В микропроцессоре i8086 применена конвейерная архитектура, которая позволяет совмещать во времени циклы выполнения и выборки из памяти кода следующих команд. Это достигается параллельной работой двух сравнительно независимых устройств – операционного устройства и устройства управления каналом (УУК). Структурная схема i8086 приведена на рис. 3.10. Операционное устройство выполняет команду, а устройство сопряжения с каналом осуществляет взаимодействие с внешней шиной – выставляет адреса, считывает коды команд и операнды, записывает данные в память или УВВ.

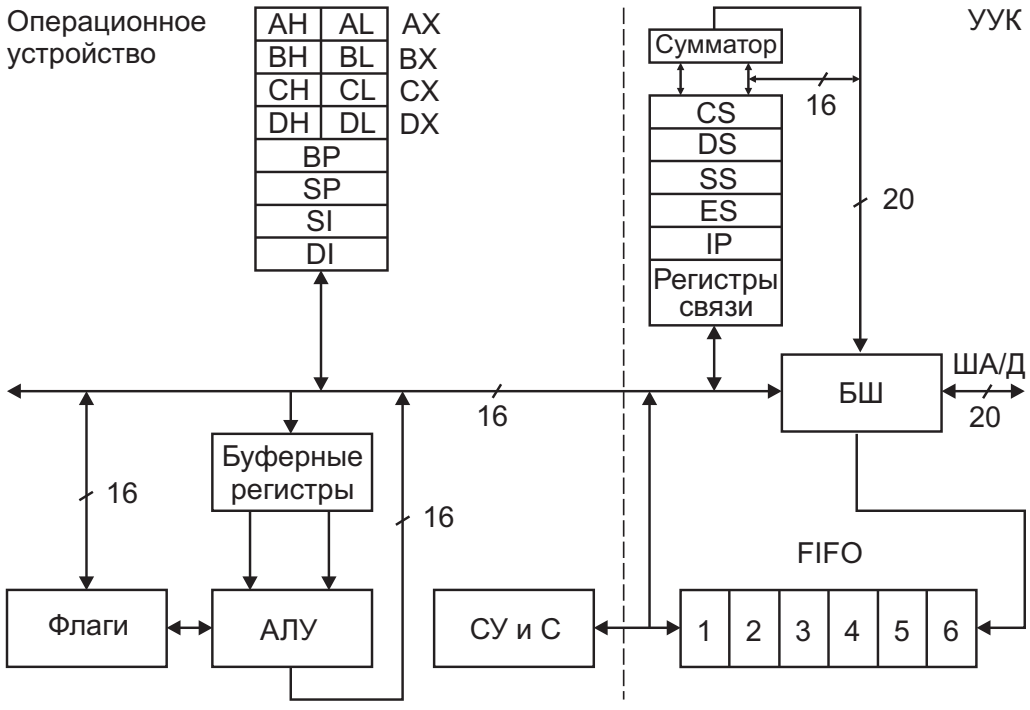


Рис. 3.10. Структурная схема микропроцессора i8086

Операционное устройство состоит из регистров общего назначения, предназначенных для хранения промежуточных результатов вычислений – данных и адресов; АЛУ с буферными регистрами; регистра флагов; схемы управления и синхронизации, которая декодирует коды команд и генерирует управляющие сигналы. Устройство управления каналом состоит из шестибайтной очереди (FIFO) команд, четырех сегментных регистров CS, DS, ES и SS, указателя команд IP, сумматора, а также вспомогательных регистров связи и буферных схем шин адреса/данных. Если операционное устройство занято выполнением команды, то устройство управления каналом самостоятельно инициирует опережающую выборку кодов команд из памяти в очередь команд. Выборка из памяти очередного командного слова осуществляется тогда, когда в очереди обнаруживаются два свободных байта. Очередь увеличивает быстродействие процессора в случае последовательного выполнения команд. При выборке команд переходов, запросов и возвращений из подпрограмм, обработки запросов прерываний очередь команд сбрасывается, и выборка начинается с нового места программной памяти. Еще одной задачей устройства управления каналом является формирование физического 20-разрядного адреса из двух 16-разрядных слов. Первым словом является содержимое одного из сегментных регистров CS, SS, DS или ES, а второе слово зависит от типа адресации операнда или кода команды. Суммирование 16-разрядных слов происходит со смещением на 4 разряда и осуществляется с помощью специального сумматора, который входит в состав устройства управления каналом.

Назначение выводов

Условное графическое изображение микропроцессора приведено на рис. 3.11. Назначение контактов БИС зависит от режима работы. Восемь контактов имеют двойное обозначение, причем обозначения в скобках соответствуют максимальному режиму. В табл. 3.7 приведены назначения контактов, одинаковые для максимального и минимального режимов, в табл. 3.8 – назначение контактов, которые используются только в минимальном режиме, в табл. 3.9 – назначение контактов, которые используются только в максимальном режиме. Буквой *z* обозначены трехстабильные выходы, которые переводятся в высокоимпедансное состояние при переходе в режим ПДП.

Линии состояния. Линии ST2…ST0 – выходы сигналов состояния - идентифицируют тип цикла шины, который выполняется в соответствии с табл. 3.10.

Циклом шины называют обращение к ячейке памяти или внешнему устройству. Однако в 16-разрядных процессорах цикл шины может инициировать не только МП, но и арифметический сопроцессор или

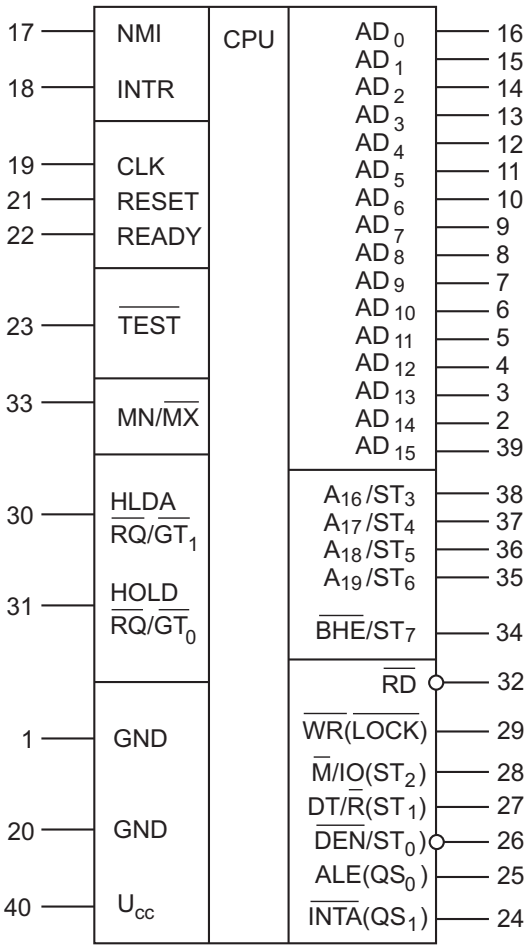


Рис. 3.11. Обозначение микропроцессора i8086 на принципиальной схеме

a_{15}	a_{14}	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
—	—	—	—	OF	DF	IF	TF	SF	ZF	—	AF	—	PF	—	CF

Рис. 3.12. Спецификация битов регистра признаков i8086

специализированный процессор ввода/вывода. Начало цикла определяется переходом линий состояния ST2–ST0 из пассивного состояния (111) в активное, а конец – обратным переходом в пассивное состояние. Сигналы ST2–ST0 подаются на входы контроллера шины i8288, который декодирует их и формирует сигналы управления системной шиной. Сигнал ST2 является логическим эквивалентом сигнала M/$\overline{IO}$, а сигнал ST1 – эквивалентом сигнала DT/$\overline{R}$. Сигналы ST4, ST3 определяют, какой сегментный регистр используется в данном цикле (табл. 3.11). Сигналы ST4 и ST3 также могут использоваться для расширения адресного пространства системы. В этом случае отдельный банк памяти в 1 Мб выделяется любому из четырех сегментов.

Таблица 3.7: Назначение выводов i8086

Вывод	Назначение	Тип
1	2	3
AD15–A0	Address/data – мультиплексная двунаправленная шина адреса/данных (ADB – Address Data Bus), по которой с разделением во времени передаются адреса и данные. Адреса передаются в первом такте цикла шины и сопровождаются сигналом ALE, а данные – во второй половине цикла шины и сопровождаются сигналом $\overline{\text{DEN}}$	Вход/ выход (z)
$\overline{\text{BHE}}$/ST7	Byte High Enable/Status 7 – выходной сигнал разрешения старшего байта/сигнал состояния. В первом такте цикла одновременно с адресной информацией передается сигнал $\overline{\text{BHE}}$. Активный (L) уровень $\overline{\text{BHE}}$ означает, что по старшей половине AD15-AD8 шины адреса/данных передаются 8-разрядные данные. Сигнал $\overline{\text{BHE}}$ используется для разрешения доступа к старшему банку памяти или к внешнему устройству с байтовой организацией, подключенному к старшей половине шины данных. В других тактах формируется сигнал состояния ST7	Выход (z)
$\overline{\text{RD}}$	Read – выходной сигнал чтения. Указывает на то, что микропроцессор выполняет цикл чтения	Выход (z)
READY	Ready – входной сигнал готовности, подтверждающий, что ячейка памяти или устройство ввода/вывода, адресуемое в команде, готово к взаимодействию с МП при передаче данных	Вход
INTR	Interrupt Request – входной сигнал запроса (при H-уровне) маскированного прерывания. Если прерывания разрешены, МП переходит к подпрограмме обработки прерывания, в противном случае – игнорирует этот сигнал	Вход

Таблица 3.7: (Окончание)

1	2	3
RESET	Reset (Clear) – сигнал аппаратного сброса (при H-уровне). Переводит МП в начальное состояние, при котором сброшены сегментные регистры (кроме CS, все разряды которого устанавливаются в единичное состояние), указатель команд IP, все флаги, регистры очереди команд и все внутренние триггеры устройства управления. Сигнал RESET не влияет на состояние РОН. Во время действия сигнала RESET все выходы, имеющие три состояния, переводятся в высокоимпедансное состояние; выходы, которые имеют два состояния, становятся пассивными. Минимальная продолжительность сигнала RESET при первом включении МП составляет 50 мкс, а при повторном запуске – четыре такта синхронизации, т.е. 0.8 мкс при тактовой частоте 5 МГц. После окончания сигнала RESET начинается цикл выборки из памяти команды с адресом 0FFFFH:0000	Вход
$\overline{\text{TEST}}$	Test – входной сигнал проверки. Сигнал используется вместе с командой ожидания WAIT, выполняя которую МП проверяет уровень сигнала $\overline{\text{TEST}}$. Если $\overline{\text{TEST}}$=0, МП переходит к выполнению следующей после WAIT команды. Если $\overline{\text{TEST}}$=1, МП находится в состоянии ожидания, выполняет холостые такты и периодически, с интервалом 5TCLK, проверяет значение сигнала $\overline{\text{TEST}}$	Вход
CLK	Clock – входные тактовые импульсы, обеспечивающие синхронизацию работы МП	Вход
MN/$\overline{\text{MX}}$	Minimum/maximum – вход сигнала выбора минимального или максимального режима. Определяет режим работы МП: при 1 – минимальный, при 0 – максимальный	Вход

Таблица 3.8: Минимальный режим i8086

Вывод	Назначение	Тип
1	2	3
INTA	Interrupt Acknowledge – выходной сигнал подтверждения прерывания, определяющий чтение вектора прерывания	Выход
ALE	Address Latch Enable – выходной сигнал разрешения фиксации адреса; выдается в начале каждого цикла шины и используется для записи адреса в регистр-фиксатор	Выход
$\overline{\text{DEN}}$	Data Enable – выходной сигнал разрешения данных, который определяет появление данных на шине адреса/данных	Выход (*z*)
DT/$\overline{\text{R}}$	Data Transmit/Receive (Output-Input) – выходной сигнал передачи/приема данных; определяет направление передачи данных по ADB. Предназначен для управления шинными формирователями и действует на протяжении всего цикла шины	Выход (*z*)
M/$\overline{\text{IO}}$	Memory/Input-Output – выходной сигнал признака обращения к памяти (M/$\overline{\text{IO}}$=1) или внешнему устройству (M/$\overline{\text{IO}}$=0). Используется для распределения адресного пространства памяти и устройств ввода/вывода	Выход (*z*)
$\overline{\text{WR}}$	Write – выходной сигнал записи. Указывает на то, что МП выполняет цикл записи в память или внешнее устройство, и сопровождает данные, которые выдаются МП на шину данных	Выход (*z*)
HOLD	Hold – входной сигнал запроса захвата шин от внешнего устройства или контроллера прямого доступа к памяти	Вход
HLDA	Hold Acknowledge – выходной сигнал подтверждения захвата. Сигнал указывает на то, что МП перевел свои шины адреса/данных, адреса/состояния и управления в z-состояние	Выход

Таблица 3.9: Максимальный режим i8086

Вывод	Назначение	Тип
1	2	3
ST2–ST0	Выходные сигналы линий состояния. Характеризуют тип выполняемого цикла шины; используются для формирования управляющих сигналов	Выход (*z*)
$\overline{RQ}/\overline{GT0}$ $\overline{RQ}/\overline{GT1}$	Request/Grant (Request/Enable) – два входных/выходных сигнала запроса/предоставления локальной шины; используются для связи с другими процессорами, в частности с арифметическим сопроцессором. Линия RQ/$\overline{GT1}$ имеет меньший приоритет	Вход/ выход
LOCK	Lock – выходный сигнал блокировки (занятости) шины – сигнал монополизации управления шиной; формируется во время выполнения команды с префиксом LOCK и информирует другие процессоры и устройства о том, что они не должны запрашивать системную шину	Выход
QS1, QS0	Queue Status – два выходных сигнала состояния очереди; идентифицируют состояние внутренней шестибайтной очереди команд и действуют на протяжении такта синхронизации после выполнения операции над очередью. Сигналы QS1, QS0 предназначены для сопроцессора, который контролирует шину адреса/данных, фиксирует момент выборки из памяти программ предназначенной для него команды с префиксом ESC, а после этого следит за очередью команд и определяет момент, когда эта команда должна выполняться	Выход

К выводам S4 и S3 подключают дешифратор, который выбирает соответствующий банк памяти. Такой прием обеспечивает расширение адресной памяти до 4 Мбайт и защиту от ошибочной записи в сегмент, который перекрывается с другими сегментами. Сигнал ST5 соответствует состоянию флага разрешения прерываний IF: ‘0’ – прерывание

Таблица 3.10. Идентификация типа цикла шины

Линии состояния			Тип цикла шины
ST2	ST1	ST0	
0	0	0	Подтверждение прерывания ($\overline{\text{INTA}}$)
0	0	1	Ввод (Чтение УВВ)
0	1	0	Вывод (Запись УВВ)
0	1	1	Останов
1	0	0	Выборка кода операции
1	0	1	Чтение ЗУ
1	1	0	Запись в ЗУ
1	1	1	Нет цикла

Таблица 3.11. Идентификация сегментного регистра

Линии состояния		Сегментный регистр
ST4	ST3	
0	0	ES – дополнительный сегмент
0	1	SS – сегмент стека
1	0	CS – сегмент кода
1	1	DS – сегмент данных

запрещено, ‘1’ – прерывание разрешено. Сигналы ST6, ST7 не используются и зарезервированы для последующих моделей МП.

Идентификация состояния очереди команд осуществляется с помощью сигналов QS1, QS0 (табл. 3.9). Значение этих линий определяет операцию над очередью команд в соответствии с табл. 3.12.

Линии запроса/предоставления локальной шины. Двунаправленные линии $\overline{\text{RQ}}/\overline{\text{GT0}}$, $\overline{\text{RQ}}/\overline{\text{GT1}}$ используются для передачи импульсных сигналов запроса/разрешения доступа к локальной шине (каналу). Процесс доступа к шине осуществляется в таком порядке: сначала устройство, которое подключено к локальной шине и требует доступа к общим ресурсам, формирует импульс продолжительностью один такт; после этого в конце текущего цикла МП выдает соответствующий импульс, подтверждающий возможность доступа к локальной шине. В следующем такте МП переводит шины адреса/данных и управления в высокоимпедансное состояние и отключается от канала. По окончании работы с каналом устройство выдает в ту же линию третий импульс, который указывает на окончание захвата канала. В следую-

Таблица 3.12. Идентификация состояния очереди команд

QS1	QS2	Операция над очередью
0	0	Операции нет, в последнем такте не было выборки из очереди
0	1	Из очереди выбран первый байт команды
1	0	Очередь пуста; была опустошена командой передачи управления
1	1	Из очереди выбран следующий байт команды

щем такте МП восстанавливает управление шиной и продолжает вычисления. Все три импульса имеют одинаковую продолжительность и низкий активный уровень. Сигналы на линиях независимы, однако при одновременном поступлении запросов линия $\overline{RQ}/\overline{GT0}$ имеет более высокий приоритет, чем линия $\overline{RQ}/\overline{GT1}$. Любая из двух рассмотренных линий используется для установления режима захвата шин и эквивалентна паре HOLD и HLDA МП i8086 в минимальном режиме.

3.3.2. Организация памяти

Память представляет собой массив информационной емкостью 1 Мбайт, т.е. 2^{20} восьмиразрядных ячеек. В памяти хранятся как байты, так и 16-разрядные слова. Слова располагаются в двух соседних ячейках памяти: старший байт хранится в ячейке со старшим адресом, младший – в ячейке с младшим адресом. Адресом слова считается адрес его младшего байта. Начальные (00000H–003FFH) и конечные адреса (FFFF0H–FFFFFH) зарезервированы для системы прерываний и начальной установки соответственно.

Организация памяти, при которой каждому адресу соответствует содержимое одной ячейки памяти, называется линейной. В МП i8086 используется сегментная организация памяти, которая характеризуется тем, что программно доступной является не вся память, а лишь некоторые сегменты, т.е. области памяти. Внутри сегмента используется линейная адресация.

Введение сегментной организации можно объяснить следующим образом. Микропроцессор i8086 представляет собой 16-разрядный процессор, т.е. имеет 16-разрядную внутреннюю шину, 16-разрядные регистры и сумматоры. Стремление разработчиков БИС адресовать по возможности больший массив памяти обусловило использование 20-разрядной шины данных. Для сравнения: 16-разрядная шина адреса разрешает адресовать 2^{16}=64 Кбайт; 20-разрядная – 2^{20}=1 Мбайт.

Для формирования 20-разрядного адреса в 16-разрядном процессоре используют информацию двух 16-разрядных регистров. В МП i8086 20-разрядный адрес формируется из двух 16-разрядных адресов, которые называют логическими. Первый логический адрес, дополненный справа четырьмя нулями, представляет собой начальный адрес сегмента емкостью 64 Кбайт. Второй логический адрес определяет смещение в сегменте, т.е. определяет расстояние от начала сегмента до адресованной ячейки. Если это расстояние равно 0000, то адресуется первая ячейка сегмента, если FFFFH – последняя. Таким образом, логическое адресное пространство разделено на блоки соседних адресов емкостью 64 Кбайт, т.е. сегменты.

Физический 20-разрядный адрес ячейки памяти формируется из двух 16-разрядных адресов – адреса сегмента Seg и исполнительного адреса EA (Executive Address), которые суммируются со смещением на четыре разряда (рис. 3.13).

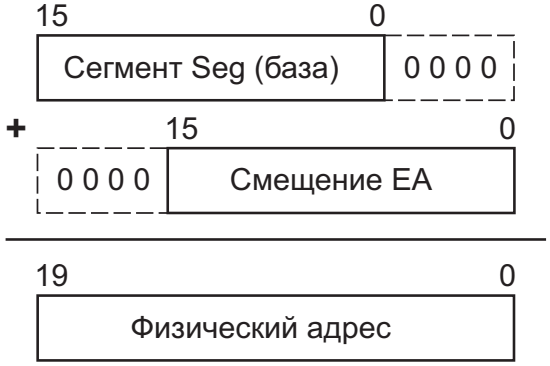


Рис. 3.13. Формирование физического адреса

Смещение адреса сегмента на четыре разряда влево эквивалентно его умножению на 16. Следовательно, физический адрес равняется 16×Seg + EA. В качестве первого логического адреса (базы) Seg используется содержимое одного из четырех сегментных регистров: CS (Code Segment – сегмент кодов), DS (Data Segment – сегмент данных), ES (Extended Segment – дополнительный сегмент), SS (Stack Segment – сегмент стека). Второй логический адрес EA, или смещение, зависит от сегмента. Так, в сегменте кодов EA используется содержимое указателя команд IP, в сегментах данных значение EA зависит от средства адресации операнда, в сегменте стека для указания второго логического адреса используются регистры SP или BP.

Преобразование логических адресов в физические всегда однозначно, т.е. паре Seg и EA отвечает единственный физический адрес. Обратное преобразование не является однозначным. В дальнейшем будем обозначать физический адрес в виде Seg:EA, где вместо Seg и EA могут использоваться как обозначения регистров, так и 16-разрядные данные.

Емкость памяти 1 Мбайт, начиная с нулевого адреса, разбивается на параграфы по 16 байтов. Сегмент может начинаться только на границе параграфа, т.е. в адресе сегмента младшие четыре бита – нулевые. Размещение сегментов в памяти произвольное: они могут частично или полностью перекрываться либо не иметь общих областей.

Изменяя значения первого и второго логических адресов, можно адресовать любую ячейку из общей памяти емкостью 1 Мбайт. На рис. 3.14,*a* показано расположение в пространстве 1 Мбайт четырех

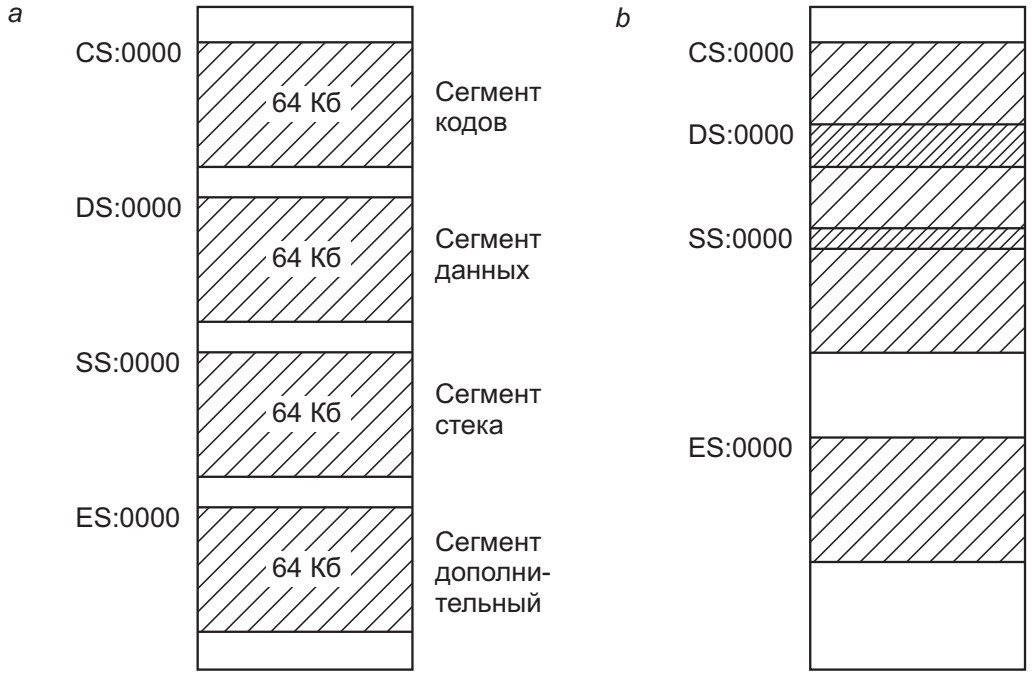


Рис. 3.14. Размещение сегментов в пространстве памяти 1 Мбайт: *a* – без перекрытия; *b* – с частичным перекрытием

сегментов по 64 Кбайт без перекрытия. Начальные адреса сегментов определяются содержимым 16-разрядных сегментных регистров, которые дополнены справа четырьмя нулевыми битами. Смещение в сегменте кодов определяется содержимым регистра IP, смещение в сегменте данных и дополнительном сегменте данных – эффективным адресом EA, который указывается в команде, смещение в сегменте стека – содержимым регистра SP.

В сегментах кодов расположены коды команд, т.е. программа в машинных кодах; в остальных сегментах – данные. Программа может обращаться только к данным в сегментах, которые обозначены на рис. 3.14 заштрихованными прямоугольниками.

Изменяя содержимое сегментных регистров, можно передвигать сегменты в границах всей памяти 1 Мбайт. На рис. 3.14,*b* показано расположение сегментов кодов, данных и стека с частичным перекрытием. Это происходит при различии содержимого сегментных регистров менее чем на 64 Кбайт/16 = 4 Кбайт.

Регистр флагов (см. рис. 3.12) хранит признаки результатов выполнения арифметических и логических операций и управляющие признаки. Последние можно установить или сбросить программно. Типы флагов представлены в табл. 3.13.

Циклы шины процессора. На протяжении цикла шины МП выставляет адрес ячейки памяти или УВВ в шину адреса, формирует управляющие сигналы чтения/записи, а после этого считывает или записывает данные. Кроме циклов "Чтение" и "Запись" в память или УВВ, существуют циклы "Подтверждение прерывания" и "Захват шин". Цикл шины может инициировать не только независимый процессор i8086, но и арифметический сопроцессор или сопроцессор ввода/вывода. Различают циклы шины в минимальном и максимальном режимах.

Циклы обращения к портам отличаются от циклов обращения к памяти тем, что старшие разряды шины адреса имеют нулевое значение (при косвенной адресации с помощью DX сигналы на линиях A19-A16 имеют L-уровень; при прямой адресации L-уровень приобретают сигналы на линиях A19-A8). На линиях в первом случае и A15-A0 и A7-A0 во втором – устанавливается адрес порта.

Цикл "Подтверждение прерывания" формируется аналогично циклу "Чтение" порта, но вместо активного сигнала $\overline{\text{IOR}}$ активным является сигнал $\overline{\text{INTA}}$, а шина адреса процессором не управляется.

3.3.3. Организация прерываний

Процессор i8086 может обрабатывать до 256 типов прерываний. Каждому прерыванию соответствует свой вектор – двойное слово, которое содержит адрес CS:IP вызываемой подпрограммы. Под векторы прерываний в общем пространстве адресов памяти отводится 1 Кбайт, начиная с нулевого адреса.

При переходе к подпрограмме обработки прерываний INT N (где N – тип прерывания) процессор перемещает в стек содержимое регистров IP, CS, регистра флагов F, сбрасывает флаг разрешения прерывания IF; вычисляет адрес $4\times N$, первое слово по этому адресу перемещает в IP, а второе – в CS.

Сброс флага прерывания IF не разрешает прерывать выполнение подпрограммы обработки прерывания до ее завершения или выполнения команды разрешения STI. Последней командой подпрограммы обработки прерывания является команда IRET. По этой команде процессор выбирает из стека адрес возврата (адрес команды, следующей за командой INT) и содержимое регистра флагов.

Прерывания (табл. 3.14) делятся на внешние аппаратные и внутренние. Запросы прерываний IRQ (Interrupt Request) внешних аппаратных прерываний поступают в систему прерываний или на линию

Таблица 3.13. Спецификация битов регистра признаков

Признак	Назначение	Операнд	
		8	16
AF	Auxiliary Flag – флаг вспомогательного переноса/заема из младшей тетрады в старшую (из разряда D3 в разряд D4). Используется в командах обработки двоично-десятичных чисел	+	–
CF	Carry Flag – флаг переноса/заема. Устанавливается при выходе результата суммирования (вычитания) беззнаковых операндов за границу диапазона. В командах сдвига флаг CF фиксирует значение старшего бита	+	+
OF	Overflow Flag – флаг переполнения, устанавливается при выходе знакового результата за границы диапазона	+	+
SF	Sign Flag – флаг знака. Дублирует значение старшего бита результата. SF = 0 для положительных чисел и SF = 1 – для отрицательных	+	+
PF	Parity Flag – флаг паритета (четности). Устанавливается при четном количестве единиц в результате	+	-
ZF	Zero Flag – флаг нулевого результата. Устанавливается при нулевом результате операции	+	+
DF	Direction Flag – флаг управления направлением обработки строк. При DF = 1 индексные регистры SI, DI автоматически декрементируются на количество байтов операнда, при DF = 0 – инкрементируются	-	-
IF	Interrupt-enable Flag – флаг разрешения прерывания. При IF = 1 разрешается выполнение маскированных аппаратных прерываний	-	-
TF	Trap Flag – флаг трассировки (пошагового режима). При TF = 1 после выполнения каждой команды вызывается внутреннее прерывание INT 1	-	-

Таблица 3.14. Спецификация векторов прерываний

Назначение	*N*	Приоритет	Время вызова, МТ
Прерывание по ошибке	0	1	50
Пошаговое прерывание	1	4	50
Немаскируемое прерывание	2	2	50
Прерывание по точкам разрыва	3	1	52
Прерывание по переполнению INTO	4	1	53
Прерывание, опред. пользователем	5 – 31	1	51
Маскируемые аппаратные прерывания	32 – 255	3	61

немаскированного прерывания NMI МП. Система прерывания формирует сигнал INTR маскированного прерывания МП. Заметим, что маскированное прерывание отличается от немаскированного тем, что первое может быть запрещено программно – командой сброса флага разрешения прерываний IF. В этом случае при поступлении запросов прерывания они будут игнорироваться.

Внутренние прерывания процессора разделяют на программные и аппаратные (табл. 3.14). Источниками внутренних программных прерываний являются: ошибка деления (тип 0), пошаговый режим (тип 1), команда INTO (тип 4).

Внутренние программные прерывания INT N и INT 3 выполняются по команде прерывания и разрешают вызывать подпрограммы обработки прерываний (например, сервисные подпрограммы BIOS и DOS) без применения дальних вызовов подпрограмм. В отличие от INT N, прерывание INT 3 является однобайтной командой и обычно используется для передачи управления подпрограмме-отладчику. Выполнение программных прерываний не зависит от флага разрешения прерывания IF.

Внутренние аппаратные прерывания процессора возникают в следующих случаях:

- при делении на ноль (тип 0);
- при установленном флаге трассировки (тип 1). В этом случае прерывание происходит после выполнения каждой команды;
- после команды INTO (тип 4), если в регистре признаков установлен флаг переполнения OF.

Аппаратные прерывания возникают при активном уровне сигналов на выводах NMI (немаскированное прерывание – тип 2) и INTR (маскированные, типы 5–255). Маскированные прерывания выполняются

при установленном флаге IF. При переходе к подпрограмме обработки аппаратного прерывания процессор последовательно формирует два цикла подтверждения прерывания, в которых генерируется сигнал $\overline{\text{INTA}}$. После второго импульса $\overline{\text{INTA}}$ контроллер прерываний передает по шине данных номер вектора прерывания *N*. Далее действия процессора аналогичны выполнению программного прерывания. Обработка текущего прерывания может быть прервана немаскированным прерыванием или другим маскированным прерыванием высшего приоритета в том случае, если подпрограмма-обработчик установит флаг разрешения прерывания IF. Немаскированное прерывание выполняется независимо от состояния флага IF.

3.3.4. Организация системы команд i8086

Программная модель

Программная модель МП i8086 состоит из РОН, сегментных регистров, указателя команд и регистра флагов. Регистры общего назначения делятся на регистры данных и регистры-указатели. К регистрам данных относятся четыре 16-разрядных регистра: AX, BX, CX, DX. Любой из этих регистров состоит из двух 8-разрядных регистров, которые можно независимо адресовать символическими именами AH, BH, CH, DH (старшие байты – High) и AL, BL, CL, DL (младшие байты – Low). Регистры-указатели SP (Stack Pointer, указатель стека), BP (Base Pointer, базовый регистр, SI (Source Index, индекс источника), DI (Destination Index, индекс назначения) являются 16-разрядными и предназначены, как правило, для хранения адресов операндов при косвенной адресации.

Все РОН можно использовать для хранения данных, но в некоторых командах допускается использование определенного регистра по умолчанию: AX – при умножении, делении, вводе и выводе слов; AL – при умножении, делении, вводе и выводе байтов, десятичной коррекции, преобразовании байтов (команда XLAT); AH – при умножении и делении байтов; BX – при трансляции; CX – как счетчик циклов и указатель длины строк в строчных командах; CL – для хранения числа сдвигов в командах; DX – при умножении и делении слов, вводе и выводе с косвенной адресацией; SP – при операциях со стеком; SI, DI – при строковых операциях. В отличие от 8-разрядных МП, в регистре SP 16-разрядных процессоров хранится смещение последней занятой ячейки стека относительно начала сегмента стека, а полный адрес стека определяется как SS:SP. Сегментные регистры CS, DS, ES, SS определяют начальные адреса четырех сегментов памяти. Использование сегментных регистров определяется типом обращения к памяти (табл. 3.15). Для некоторых типов обращений допускается

Таблица 3.15. Использование регистров при адресации памяти

Тип обращения к памяти	Сегментный регистр		Смещение
	по-умолчанию	явно	
Выборка команд	CS	–	IP
Стековые операции	SS	–	SP
Адресация переменной	DS	CS, ES, SS	EA
Строка-источник*	DS	CS, ES, SS	SI
Строка-приемник*	ES	–	DI
Использование BP при обращении к стеку при чтении/записи	SS	CS, ES, SS	EA

*Строки данных (массивы, цепочки), которые принимают участие в строковых командах.

замена сегментного регистра по умолчанию на альтернативный, что осуществляется с помощью префиксов CS:, DS:, SS:, ES:.

Адресация портов ввода/вывода. Пространство адресов портов ввода/вывода не сегментировано, занимает 64 Кбайт и адресуется 16 младшими разрядами 20-разрядной шины адреса. Порты могут быть как 8-, так и 16-разрядными. Любые два смежных 8-разрядных порта можно рассматривать как 16-разрядный порт аналогично слову в памяти. При этом для обмена с 8-разрядными портами используется регистр AL, а с 16-разрядными – регистр AX. Первые 256 портов (с номерами 0–0FFH) можно адресовать с помощью прямой адресации. Все 64 Кбайт портов адресуются косвенно – с помощью регистра DX.

Типы адресации операндов. В МП i8086 используются те же основные типы адресации (прямая, регистровая, непосредственная и косвенная), что и для 8- разрядных процессоров, однако косвенная адресация имеет следующие разновидности: базовая, индексная, базово-индексная.

Базовая адресация. Эффективный адрес операнда EA вычисляется суммированием содержимого базовых регистров BX или BP и смещения (8- или 16-разрядного знакового числа). В частном случае смещения может не быть.

Индексная адресация. При индексной адресации в качестве адреса смещения используется сумма содержимого индексных регистров SI или DI и смещения в виде числа.

Базово-индексная адресация. Эффективный адрес операнда EA равен сумме содержимого базовых регистров JBX или BP, индексных

регистров SI или DI и смещения – некоторого числа, задаваемого в команде. Заметим, что числовое смещение может отсутствовать.

Базовая и индексная адресации применяются для обращения к элементам одномерного массива, базово-индексная – к элементам двумерного массива.

Структура и форматы команд

Система команд микропроцессора i8086 содержит 135 базовых команд (91 мнемокод). Форматы команд составляют от 1 до 6 байтов. Максимальная длина команды соответствует глубине очереди команд FIFO микропроцессора. Общая структура команд представлена на рис. 3.15. Звездочкой отмечены байты, которые могут отсутствовать в составе команды. Байт кода операции КОП является обязательным и присутствует в каждой команде. В байте КОП выделены два одно-

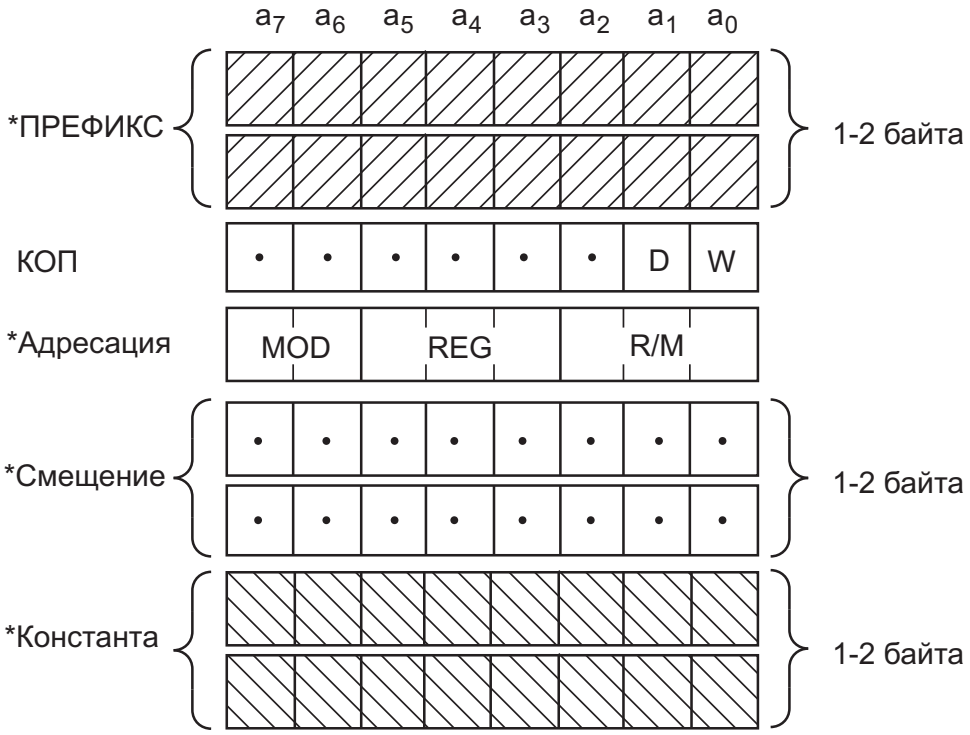


Рис. 3.15. Структура и форматы команд i8086

битовых поля: ‘D’ и ‘W’. Бит D задает направление пересылки данных в случае, когда в команде присутствует байт способа адресации. При D = 1 поле R/M в байте адресации задает источник, а поле REG – приемник. При D = 0 источник адресуется полем REG, а приемник – полем R/M. Бит W задает размерность регистра, указанного в поле REG. При W = 1 подразумевается 16-разрядный регистр, а при W = 0 – 8-разрядный регистр.

Поля MOD, REG и R/M байта адресации определяют способы задания местоположения операндов. В табл. 3.16 приведены способы

вычисления эффективного адреса операндов в зависимости от кодов полей R/M и MOD. Один (DISP8) или два (DISP16) байта смещения идут следом за байтом адресации (рис. 3.15). Их роль в формировании эффективного адреса полностью определяется табл. 3.16. При значении поля MOD = 11 поле R/M адресует регистры микропроцессора. Спецификация поля R/M в этом случае будет совпадать со спецификацией поля REG, задающего адреса 8- и 16-разрядных регистров микропроцессора в соответствии с табл. 3.17.

Таблица 3.16. Вычисление эффективного адреса

R/M	MOD		
	00	01	10
000	BX + SI	BX + SI + DISP8	BX + SI + DISP16
001	BX + DI	BX + DI + DISP8	BX + DI + DISP16
010	BP + SI	BP + SI + DISP8	BP + SI + DISP16
011	BP + DI	BP + DI + DISP8	BP + DI + DISP16
100	SI	SI + DISP8	SI + DISP16
101	DI	DI + DISP8	DI + DISP16
110	DISP16	BP + DISP8	BP + DISP16
111	BX	BX + DISP8	BX + DISP16

Таблица 3.17. Адресация регистров микропроцессора

W	REG или (R/M при MOD=11)							
	000	001	010	011	100	101	110	111
0	AL	CL	DL	BL	AH	CH	DH	BH
1	AX	CX	DX	BX	SP	BP	SI	DI

Далее могут идти один или два байта данных (константа), представляющие собой непосредственный операнд.

Перед кодом операции КОП могут быть один или два байта префикса. Префиксы не создают нового машинного кода, но изменяют алгоритм выполнения операции. Примерами могут служить уже упоминавшиеся префиксы сегментных регистров (CS:, DS:, SS: и ES:), которые изменяют способ вычисления физического адреса (табл. 3.15).

3.3.5. Список команд i8086

Все команды микропроцессора i8086 (прил. 2) можно разделить на шесть групп:

1) команды пересылки данных;
2) арифметические команды;
3) логические команды, включая команды сдвига;
4) команды передачи управления, включая команды прерываний;
5) строковые команды;
6) команды управления состоянием микропроцессора.

Обсудим кратко особенности команд каждой группы.

Команды пересылки данных

Команды пересылки данных не модифицируют состояния регистра признаков. Исключение составляют лишь две команды, которые непосредственно изменяют содержимое регистра признаков: SAHF и POPF.

В данную группу входят:

- традиционные команды пересылки (MOV) с операндами в виде сегментного регистра, 8/16-разрядного РОНа, 8/16-разрядного операнда в памяти или непосредственного операнда-источника;
- команды обмена данными между операндами (XCHG);
- команды табличных преобразований данных (XLAT);
- команды для вычисления эффективного адреса и его загрузки в регистр (LEA);
- команды для быстрого извлечения из памяти и загрузки четырехбайтных адресов (LDS, LES);
- пересылки данных между AX и младшим байтом регистра признаков (LAHF, SAHF);
- команды записи (PUSH) в стек и извлечения из стека (POP) двухбайтовых операндов. Порядок операций при записи в стек: декремент SP, запись старшего байта в [SS:SP], декремент SP, запись младшего байта в [SS:SP]. При извлечении из стека применяется обратный порядок операций. Команды PUSH/POP работают с сегментными регистрами, 16-разрядными РОНами и 16-разрядными словами в памяти. Для работы с регистром признаков используют специальные стековые команды PUSHF и POPF;
- команды ввода/вывода. Адресация портов может быть прямая однобайтная (0…255) или косвенная через двухбайтный регистр DX (0…64K). Одним операндом является 8/16-разрядный порт, а другим – регистры AL или AX соответственно.

Арифметические команды

В отличие от предыдущих микропроцессоров, где результат операции размещался только в регистре-аккумуляторе, в i8086 результат арифметической операции помещается в левый операнд, в качестве которого может выступать любой 8/16-разрядный регистр общего назначения или ячейка памяти.

В группу арифметических команд входят:

- команды сложения (ADD) и вычитания (SUB);
- команды сложения (ADC) и вычитания с (SBB) с учетом бита CF;
- команды инкремента (INC) и декремента (DEC);
- команда смены знака операнда (NEG);
- команда сравнения (CMP), которая работает как команда вычитания, но результат вычитания не записывается, а устанавливаются лишь флаги регистра признаков;
- команды коррекции результатов сложения (AAA, DAA) и вычитания (AAS, DAS) чисел в двоично-десятичном коде (ДДК). Поддержаны распакованный (AAA, AAS) и упакованный (DAA, DAS) двоично-десятичные коды;
- команды умножения (MUL, IMUL) и деления (DIV, IDIV) порядковых (MUL, DIV) и целых со знаком (IMUL, IDIV). Деление на нуль вызывает прерывание INT 0. Коррекция умножения (AAM) и деления (AAD) распакованных ДДК;
- команды преобразования байта в слово (CBW) и преобразования слова в двойное слово (CWD). Преобразование достигается повторением содержимого старшего (знакового) бита AL.7 или AX.15 во всех добавляемых разрядах.

Логические команды

Логические команды имеют ту же самую структуру операндов, что и арифметические команды. Помимо традиционного набора логических команд инверсии (NOT), дизъюнкции (AND), конъюнкции (OR) и сложения по модулю два (XOR) в группу логических операций входит команда TEST. Эта команда выполняет операцию логического ‘И’ без записи результата операции в левый операнд. Весь смысл операции – в установке флагов, соответствующих результату операции.

Операции сдвигов представлены: командами циклических сдвигов с включением CF в цепь сдвигов (RCL/RCR) и без включения CF в цепь сдвига (ROL/ROR); командами арифметических (SAL/SAR) и логических (SHL/SHR) линейных сдвигов. Особенностью является возможность задания количества позиций сдвига в регистре CL.

Строковые команды

Строкой (массивом, цепочкой) называется последовательность байтов, подряд идущих в памяти. Строка представляет собой дальнейшее развитие понятия “слово” данных. Длина строки ограничена значением 64 K. Элементом строки может быть байт (B) или двухбайтовое слово (W). Для адресации элементов строк используют регистры DS:[SI] (элемент строки-источника) и ES:[DI] (элемент строки-приемника). После выполнения операции над элементом строки содержимое регистра SI или DI автоматически изменяется, чтобы адресовать следующий элемент строки. Направление изменения адреса задается флагом DF регистра признаков: инкремент при DF = 0 и декремент при DF = 1.

Для автоматического повторения строковых операций предусмотрены префиксы аппаратных циклов. Регистр CX является счетчиком цикла. Условиями окончания аппаратных циклов являются обнуление CX (REP), ZF = 0 ИЛИ CX = 0 (REPNE, REPZ), ZF = 1 ИЛИ CX = 0 (REPE, REPNZ).

Команды поддерживают пять операций над элементами строк: копирование байта из элемента строки-источника в элемент строки-приемника (MOVSB, MOVSW); загрузка элемента строки источника в регистр AL (LODSB) или AX (LODSW); размещение в элементе строки-приемника содержимого AL (STOSB) или AX (STOSW); сравнение элементов строки-источника и строки-приемника с записью результата сравнения в регистр флагов (CPMSB, CPMSW); сравнение элемента строки-источника с содержимым регистров AL (SCASB) или AX (SCASW) с записью результата сравнения в регистр флагов.

Команды передачи управления

В системе команд i8086 определены три типа переходов, отличающихся форматом адреса и предельной длиной перехода.

SHORT – короткие переходы в пределах страницы памяти размером 256 байтов. Адрес перехода представляет собой однобайтное число со знаком в дополнительном коде, которой суммируется с содержимым регистра IP. Таким образом, реализуются переходы от текущего значения IP на (–128 … +127) байтов памяти команд. Знак минус соответствует переходам назад. Все условные переходы являются короткими.

NEAR – внутрисегментные (близкие) переходы в пределах одного сегмента памяти. Адрес перехода представляет собой 16-разрядное смещение, которое загружается в регистр IP. Содержимое сегментных регистров не изменяется.

FAR – межсегментные (далекие) переходы в пределах всего адресного пространства микропроцессора. Адрес перехода содержит четыре байта: 16-разрядное смещение и 16-разрядную базу.

Слова SHORT, NEAR и FAR являются зарезервированными словами языка ассемблера i8086. Они предназначены для конкретизации формата адреса перехода при ассемблировании.

Команды безусловного перехода по прямому и косвенному адресу (JMP), вызова подпрограммы (CALL) и возврата из подпрограммы (RET) допускают все три формата адреса перехода.

Команды условного перехода (J(cond)) по короткому прямому адресу осуществляют передачу управления по заданному относительному адресу, если выполняется указанное условие (табл. 3.18). Команда состоит из первой буквы “J” и мнемонического обозначения одного из условий. Например, для условия “NZ” запишем команду в виде “JNZ” и т.д. Выделяют три типа условий. К первому типу относят десять условий, связанных с наличием или отсутствием одного из флагов (CF, ZF, PF, SF или OF) регистра признаков. Второй тип условий предназначен для работы с командами сравнения двух операндов. Формулировка условий удобна для описания результатов сравнения (больше, меньше, равно и т.д.), а сами условия представляют собой логические комбинации соответствующих флагов (табл. 3.18). Третий тип условий служит для контроля опустошения (обнуления) регистров. В системе команд i8086 имеется лишь одно такое условие: CX = 0.

Команды перехода к подпрограмме (CALL) работают по традиционной схеме: запись точки возврата в стек и передача управления по указанному адресу. Команды возврата из подпрограммы (RET) извлекают из стека адрес точки возврата. Для различных типов переходов предусмотрены различные пары CALL/RET. Команда RET может иметь параметр *n* для коррекции указателя стека.

Команды прерываний (INT n) отличаются от команд CALL: во-первых, помимо точки возврата автоматически записывают в стек регистр флагов; во-вторых, оперируют только с 32-разрядными адресами, т.е. осуществляют межсегментные переходы. Для возврата из подпрограммы прерываний служит команда IRET, которая извлекает из стека 32-разрядный адрес точки возврата и сохраненное содержимое регистра признаков.

Команды управления состоянием МП

К этой группе команд относят команды для манипулирования управляющими битами регистра признаков (CF, DF, IF), для работы в мультипроцессорной системе (ESC, LOCK), а также команды пропуска операции и останова.

Таблица 3.18. Условия передачи управления

Мнемоника (cond)		Условие	Описание условия
NC		CF = 0	Отсутствие переноса
C		CF = 1	Перенос
NZ		ZF = 0	Неравенство нулю
Z		ZF = 1	Равенство нулю
NO		OF = 0	Отсутствие переполнения
O		OF = 1	Переполнение
NS		SF = 0	Положительный результат
S		SF = 1	Отрицательный результат
NP	PO	PF = 0	Нечетность
P	PE	PF = 1	Четность
NE		ZF = 0	оп1 $\neq$ оп2
E		ZF = 1	оп1 = оп2
NAE	B	CF = 1	(б/з) оп1 $<$ оп2
AE	NB	CF = 0	(б/з) оп1 $\geq$ оп2
NA	BE	(CF = 1) + (ZF = 1)	(б/з) оп1 $\leq$ оп2
A	NBE	(CF = 0) & (ZF = 0)	(б/з) оп1 $>$ оп2
NGE	L	SF $\neq$ OF	(зн) оп1 $<$ оп2
G	NLE	SF = (ZF & OF)	оп1 $>$ (зн) оп2
GE	NL	SF = OF	(зн) оп1 $\geq$ оп2
NG	LE	(SF $\neq$ OF) + (ZF = 0)	(зн) оп1 $\leq$ оп2
CXZ		(CX) = 0	Нуль в регистре CX

Примечание. оп1, оп2 – операнды первый и второй соответственно; (б/з) – отсутствие знака у операнда; (зн) – наличие знака у операнда.

3.4. Микропроцессоры z80, i8088, i80186, i80188

Микропроцессор z80

Zilog Z80 — 8-разрядный микропроцессор, разработанный и производимый фирмой Zilog с 1976 года. Он широко использовался в домашних и персональных компьютерах, а также во встраиваемых и военных системах. Z80, вместе с его наследниками и клонами, составляют одно из наиболее широко использовавшихся семейств микропроцессоров, которое вместе с семейством MOS Technology 6502 было доминирующим семейством на рынке 8-разрядных компьютеров с 1970-х до середины 1980-х годов.

Zilog Z80 был разработан спустя некоторое время после появления на рынке i8080. Новый процессор создавался бинарно-совместимым с 8080, так что большая часть кода для i8080 могла функционировать на новом процессоре, в частности — операционная система CP/M.

Z80 имел ряд улучшений по сравнению с i8080: расширенный набор команд, включая побитовые операции, поблочное копирование, поблочный ввод/вывод, инструкции поиска, новые регистры IX и IY и инструкции для них, новые режимы прерываний, два отдельных блока регистров, между которыми можно быстро переключиться, единственный 5-вольтовый источник питания, значительно меньшая цена. Первые модели Z80 работали на тактовой частоте 2.5 МГц, более поздние модели достигали частоты до 20 МГц.

В бывшем Советском Союзе был создан полностью совместимый клон Z80 — микропроцессор Т34ВМ1.

Будучи промышленным лидером с начала выпуска, 8-разрядный микропроцессор Z80 до сих пор популярен среди производителей микроконтроллеров. Архитектура Z80 идеально подходит для использования во встраиваемых системах управления. Она основана на сдвоенных блоках регистров, что позволяет быстро выполнять действия над регистрами и обрабатывать прерывания.

Микропроцессор i8088

Данный МП отличается от i8086 тем, что имеет внешнюю 8- разрядную шину данных при внутренней 16-разрядной шине. Уменьшение разрядности шины данных упрощает построение блоков памяти интерфейса с внешними устройствами, но производительность процессора снижается на 20-30%. Структурная схема i8088 аналогична схеме МП i8086, однако длина очереди команд сокращена до 4 байт, а опережающая выборка кодов команд выполняется при наличии одного свободного байта. Эти свойства оптимизируют конвейер с учетом разрядности шины. С программной точки зрения процессоры иден-

тичны, их система команд и набор регистров одинаковы. Так же как и МП i8086, МП i8088 выполняет 8- и 16-разрядные логические и арифметические операции, включая умножение и деление в двоичном и двоично-десятичном кодах, операции со строками, поддерживает режимы прерывания, прямого доступа к памяти, операции с портами. Расположение и назначение выводов МП i8086 и i8088 совпадают, за исключением того, что в i8088 линии AD15–AD8 используются только для адреса, а линия $\overline{\text{BHE}}$ заменена линией состояния ST0. Сигналы ST0, DT/$\overline{\text{R}}$ и IO/$\overline{\text{M}}$ могут использоваться для идентификации типа цикла шины в соответствии с табл. 3.10.

Микропроцессоры i80186 (i80188)

Встраиваемый микропроцессор i80186 (i80188) появился в 1982 г. Он, помимо улучшенного ядра i8086, содержит на своем кристалле множество компонентов поддержки, ранее представленных отдельными БИС. Это позволило сократить количество микросхем в системе и уменьшить ее стоимость. Система команд нового семейства была также расширена по сравнению с i8086 (i8088).

Новые компоненты: два контроллера прямого доступа к памяти (DMA) со схемами прерываний; дешифраторы адреса (программируемые схемы выбора кристалла); трёхканальный программируемый таймер/счётчик; генератор синхронизации; программируемый контроллер прерываний.

В 1987 году был выпущен процессор i80C186. Он производился по улучшенной технологии (CHMOS III), что позволило увеличить тактовую частоту процессоров вдвое, а потребляемую мощность снизить в 4 раза. При этом была сохранена совместимость по выводам микросхемы с предыдущими процессорами.

В 1990 году был выпущен процессор 80C186EB с модульным ядром 80C186 (Modular Core). БИС поддержки были спроектированы как модули со стандартными интерфейсами. В связи с переходом на новую технологию (CHMOS IV) и модульную структуру удалось снизить потребляемую мощность. Процессор i80C186EB нашел применение в переносной аппаратуре (например, сотовые телефоны).

Процессоры 80C186XL, 80C186EA и 80C186EC были выпущены в 1991 году. Они также базируются на модульном ядре 80C186. Процессор i80C186XL обладает высокой производительностью и низким энергопотреблением. Процессор i80C186EA объединяет в себе процессор i80C186 с новыми возможностями управления энергопотреблением. Процессор i80C186EC включает в себя дополнительные элементы, которых не имели другие процессоры семейства i80C186.

4. ОДНОКРИСТАЛЬНЫЕ МИКРОКОНТРОЛЛЕРЫ И МИКРОЭВМ

4.1. Термины и определения

Широкое использование микропроцессорной техники для задач управления привело к появлению на рынке специализированных микропроцессорных устройств, ориентированных на подобного рода применения. Особенностью этих микросхем является то, что, помимо собственно процессора, на этом же кристалле расположена и система ввода-вывода, позволяющая снизить функциональную сложность и габаритные размеры микропроцессорной системы управления. Подобные устройства получили название *микроконтроллеры*.

Одними из первых микроконтроллеров были микросхемы семейства MCS-48 фирмы Intel, выпущенные в 1976 году. В 1981 году фирма Intel выпустила новое семейство 8-разрядных микроконтроллеров MCS51, которые получили огромное распространение во всем мире и дали толчок бурному развитию микроконтроллеров. Вслед за фирмой Intel микроконтроллеры начинают выпускать и другие ведущие производители микропроцессорной техники. Фирма Motorola выпустила самый популярный в мире 8-разрядный микроконтроллер M68HC05. Эволюция микроконтроллеров соответствовала общему прогрессу микропроцессорной техники. Увеличивалась разрядность микроконтроллеров, их быстродействие, совершенствовалась встроенная система ввода-вывода. Появились 16-разрядные (MCS-96 фирмы Intel, M68HC12, M68HC16 фирмы Motorola и многие другие), а затем и 32-разрядные (например, M68HC32 фирмы Motorola) микроконтроллеры. С 1984 года начало развиваться направление так называемых RISC-микроконтроллеров, обладающих повышенным быстродействием. Представителями подобных устройств являются семейства SAB80C166 фирмы Siemens, MPC500 (на базе PowerPC ядра) фирмы Motorola и др.

Уклон в сторону управления накладывает отпечаток на особенность архитектуры микроконтроллеров. Дальнейшая интеграция позволила объединить на одном кристалле с процессором не только систему ввода-вывода, но и блок памяти. В последнем случае принято говорить о микроконтроллере как об однокристальной микроЭВМ.

Рассмотрим основные понятия и определения, необходимые для дальнейшего изложения. На рис. 4.1 представлена упрощенная структурная схема микропроцессорной системы управления. Процессор вы-

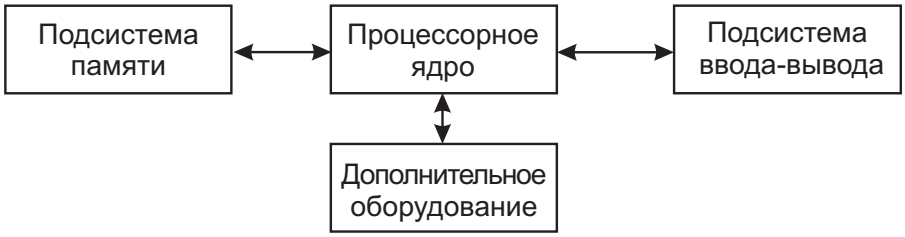


Рис. 4.1. Упрощенная структурная схема микропроцессорной системы

полняет все программные действия, необходимые в соответствии с алгоритмом работы устройства. В блоке памяти хранятся команды программы функционирования процессора, а также значения констант и переменных величин, участвующих в вычислениях. Блок ввода-вывода выполняет функцию сопряжения микропроцессорной системы с объектом управления.

Микроконтроллер – вычислительно-управляющее устройство, предназначенное для выполнения функций логического контроля и управления периферийным оборудованием, выполненное в виде одной БИС и сочетающее в себе микропроцессорное ядро и набор встроенных устройств ввода-вывода.

Однокристальная микроЭВМ – вычислительно-управляющее устройство, объединяющее на одном кристалле с процессором, систему ввода-вывода, блок памяти и некоторые другие дополнительные блоки.

При изложении материала мы не будем делать различий между однокристальными микроконтроллерами и микроЭВМ. Во всех случаях будет использоваться общий термин – микроконтроллер.

Микроконтроллер состоит из нескольких подсистем, включая процессорное ядро, подсистему ввода-вывода, подсистему памяти и ряд дополнительных блоков. К дополнительному оборудованию относят встроенные периферийные устройства: многоканальные генераторы сигналов с широтно-импульсной модуляцией (ШИМ), аналого-цифровые преобразователи, блоки векторных преобразований координат, таймеры-счетчики, сторожевые таймеры и т.п. Примерами таких устройств могут служить микроконтроллеры ADMC330 фирмы Analog Devices, TMS320C240 фирмы Texas Instruments, 56800 фирмы Motorola, векторный сопроцессор ADMC200 фирмы Analog Devices.

4.2. Основное оборудование микроконтроллера

4.2.1. Процессорное ядро

Современные микроконтроллеры могут быть построены как по гарвардской (MCS51 Intel), так и по фон-неймановской архитектурам (MCS96 Intel, 80C166 Siemens). Все они имеют внешнюю системную магистраль для обмена данными с внешней памятью и дополнительными периферийными устройствами. Классические семейства микроконтроллеров (MCS51) имеют, как правило, мультиплексные шины адреса/данных, что обусловлено необходимостью минимизировать размер микросхемы. Однако современные быстродействующие микроконтроллеры используют разделенные шины без мультиплексирования, что ускоряет работу системы. Некоторые модели микроконтроллеров имеют возможность работать либо с мультиплексной, либо с демультиплексной шиной в зависимости от требуемой конфигурации системы. В случае демультиплексной шины контроллер быстрее обменивается данными по магистрали. При работе с мультиплексной шиной освободившиеся выводы используются в качестве портов ввода-вывода (MCS251 Intel, 80C166 Siemens).

Почти все микроконтроллеры выполняют только операции с фиксированной точкой. Существуют 8-разрядные (MCS51 Intel, MC6805 Motorola), 16-разрядные (MCS-96 Intel, 80C166 Siemens, MC6816 Motorola) и 32-разрядные (MC683 Motorola, MPC500 PowerPc) микроконтроллеры. Системы команд микроконтроллеров поддерживают, как правило, широкий набор методов адресации.

4.2.2. Подсистема памяти

Различают микроконтроллеры с аккумуляторной (MCS51) и регистровой (MCS-96) организацией. Количество регистров и их разрядность зависит от конкретной модели. Зачастую микроконтроллеры имеют несколько банков регистров (MCS-48, MCS51, 80C166).

Резидентная (внутренняя) память данных (РПД) определенного объема присутствует в простом микроконтроллере практически всегда. Она обменивается данными с процессорным ядром по внутренней магистрали микроконтроллера, которая может быть организована иначе, чем внешняя. Поэтому обмен данными с внутренней памятью данных, как правило, осуществляется быстрее, чем с внешней. Разновидности памяти данных: регистровый файл, различные виды EEPROM с ресурсами от 100 тыс. до 1 млн. операций записи и системой страховки от сбоя при записи.

Варианты реализации резидентной (внутренней) памяти программ (РПП) могут быть различными.

1. РПП отсутствует. В этом случае микроконтроллер выполняет программу, считывая команды из внешней памяти программ через системную магистраль.

2. РПП выполнена в виде масочного ПЗУ. В этом случае микроконтроллер не нуждается во внешней памяти программ. Однако программа во внутреннюю память записывается однократно на этапе изготовления кристалла и не может быть изменена в дальнейшем. Как правило, программа, записанная во внутреннюю память, выполняется быстрее, чем из внешней памяти.

3. РПП выполнена в виде однократно программируемого ППЗУ. В этом случае пользователь сам может записать программу во внутреннюю память, но лишь однажды. Для записи программы, как правило, необходим специальный программатор. Однако существуют микроконтроллеры, способные программировать сами себя. Например, в MCS-96 программатор реализован внутри кристалла микроконтроллера.

4. РПП выполнена в виде ППЗУ с УФ-стиранием (EPROM). В этом случае память программ может быть многократно перепрограммирована с помощью программатора. Перед очередным программированием она должна быть очищена с помощью УФ-излучения.

5. РПП выполнена в виде Flash–памяти. В этом случае память программ может быть многократно перепрограммирована в процессе работы системы.

6. РПП реализована в виде оперативного запоминающего устройства (ОЗУ). В этом случае для загрузки программы после включения питания используется так называемый Boot Strep загрузчик. Это механизм, позволяющий после включения питания загрузить начальную программу функционирования по последовательному каналу связи либо по системной магистрали.

4.2.3. Подсистема ввода-вывода

Подсистема ввода-вывода состоит из набора разнообразных устройств, выполняющих специфические функции управления и контроля.

Параллельные порты ввода-вывода

Порты могут быть либо однонаправленными для выполнения функции ввода или ввода, либо квазидвунаправленными. Такие порты могут выполнять функции как входа, так и выхода (в каждый конкретный момент времени либо вход, либо выход). Микроконтроллеры, поддерживающие битовую адресацию, способны управлять состоянием каждого вывода порта отдельно, поддерживают альтернатив-

ные функции (АФ) входа и выхода. Упрощенная структура квазидвунаправленного порта с одним управляющим битом приведена на рис. 4.2, *a*. Подобная схемотехника применяется в MCS51. Для вво-

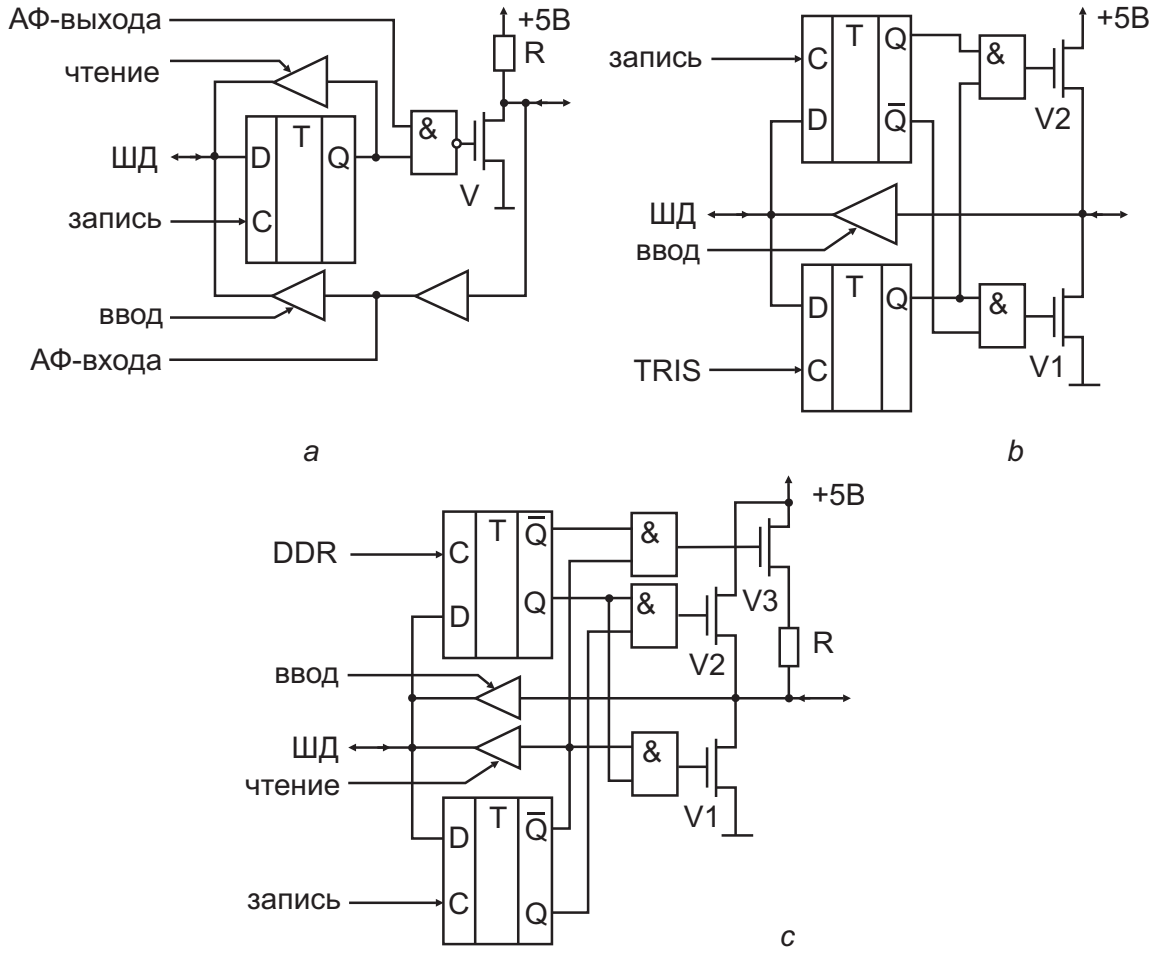


Рис. 4.2. Схемотехника портов ввода-вывода с управлением одним (*a*), двумя (*b*) и тремя (*c*) битами

да необходимо закрыть транзистор V. В PIC-микроконтроллерах используют схемотехнику с двумя управляющими битами (рис. 4.2, *b*) и триггером переключения направления, управляемым стробом TRIS. Если порт работает как выход, то один из выходных транзисторов открыт, а другой закрыт в соответствии с уровнем выходного сигнала. Для того чтобы порт работал как вход, необходимо записать в триггер направления 0, тогда выходные ключи V1 и V2 будут закрыты, т.е. будет иметь место высокий импеданс линии входа. В микроконтроллерах типа AVR используют три управляющих бита (рис. 4.2, *c*), регистр переключения направления, управляемый стробом DDR, и ключ V3 с резистором R, обеспечивающие тонкую настройку в режиме ввода. Имеется возможность считывать информацию не только с входной линии, но и непосредственно из буфера выходного регистра данных.

Последовательные порты ввода-вывода

Модуль последовательного ввода-вывода используется для обмена данными между микроконтроллером и удаленным периферийным узлом. Данные передаются в последовательном коде, т.е. биты передаются во временной последовательности друг за другом по одному каналу связи. Это даёт возможность использовать один канал связи вместо нескольких в случае параллельного кода. Однако при этом снижается быстродействие канала связи.

Микроконтроллер может содержать либо отдельные синхронные и асинхронные порты ввода-вывода, либо универсальный порт, способный работать в нескольких режимах. Перечислим некоторые виды последовательных портов, характерных для однокристальных микроконтроллеров:

- последовательный синхронный или синхронно-асинхронный приемопередатчик (УАПП или УСАПП), представляющий собой последовательный порт ввода-вывода в стандарте, который совпадает с RS232 во всем, кроме уровней сигнала. УСАПП и УАПП обычно поддерживают стандартный для RS232 набор скоростей передачи, вплоть до 115200 бит/с, хотя во многих случаях могут работать и на более высоких скоростях, особенно в синхронном режиме;
- последовательный синхронный периферийный интерфейс SPI, представляющий собой последовательный синхронный порт ввода-вывода по трехпроводному последовательному каналу;
- последовательный синхронный периферийный интерфейс I^2C, представляющий собой последовательный синхронный порт ввода-вывода по двухпроводному последовательному каналу;
- последовательный сетевой интерфейс CAN, представляющий собой последовательный асинхронный порт ввода-вывода;
- универсальная последовательная шина USB.

Имеет место тенденция постоянного повышения предельных скоростей передачи по мере развития элементной базы. Данная тенденция находит свое отражение в появлении новых редакций стандартов на последовательные интерфейсы, закрепляющих эти новые, более высокие значения предельных параметров. Например, новые редакции стандарта специфицируют последовательную шину USB для скоростей обмена 12 Мбит/с (USB-1.1) и 480 Мбит/с (USB-2.0).

4.3. Таймеры и процессоры событий

Микроконтроллеры работают в режиме реального времени, управляя внешними объектами или считывая данные с внешних источни-

ков сигналов. В этой связи неотъемлемой частью современного микроконтроллера является подсистема реального времени. Важнейшей частью этой подсистемы служат цифровые таймеры и различные устройства на их основе. Таймеры используются для отсчета временных интервалов (интервальные таймеры) либо для подсчета внешних событий (счетчики внешних событий). В более сложных устройствах подсистемы реального времени таймеры работают непрерывно, задавая масштаб времени – временную базу. В этом случае все отсчеты интервалов времени и другие действия в режиме реального времени выполняются аппаратно.

4.3.1. Таймеры/счетчики

Включают в себя таймеры/счетчики общего назначения и специализированные счетчики с коммутируемыми источниками синхронизации.

Интервальный таймер/счетчик общего назначения

Интервальный таймер/счетчик общего назначения представляет собой интегрированное в микроконтроллер аппаратное устройство, состоящее из счетчика импульсов (обычно по модулю 8 или 16), схемы управления и схемы переноса. Конкретные режимы работы и структура интервальных таймеров/счетчиков зависят от модели микроконтроллера. Схема управления обычно содержит один или несколько управляющих программно-доступных регистров для задания режима работы и управления работой таймера/счетчика. Для запуска и остановки счетчика/таймера используют либо специальные команды, либо общие команды для установки (сброса) определенных битов регистра управления. Счетчик представляет собой программно-доступный регистр, содержимое которого может считываться и модифицироваться обычными командами пересылки данных. Некоторые модели микроконтроллеров допускают считывание содержимого таймера/счетчика “на лету”, т.е. без остановки его работы.

Источником счетных импульсов может быть либо один из входов микроконтроллера, либо системный синхросигнал с фиксированной частотой (возможно, предварительно деленной на заданное число).

В первом случае устройство выполняет функции счетчика внешних событий (отрицательных или положительных перепадов) на входе микросхемы. Счет может производиться либо в одном, либо в обоих направлениях. В последнем случае направление счета определяется уровнем сигнала на соответствующем входе микросхемы.

Во втором случае устройство выполняет функции интервального таймера, так как фактически считает интервалы времени постоянной

длительности, задаваемые высокостабильным тактовым генератором.

При переходе содержимого счетчика из наибольшего состояния в наименьшее (или наоборот) схема переноса устанавливает специальный флаг переполнения (переноса). Этот флаг может опрашиваться программно или аппаратно. В последнем случае при установке флага переполнения формируется внутренний запрос на прерывание.

Основными недостатками классического таймера/счетчика являются:

1) потери времени на выполнение команд пуска и останова таймера, приводящие к ошибкам при измерении временных интервалов;

2) сложности при формировании временных интервалов (меток времени), отличных от периода полного коэффициента счета;

3) невозможность одновременного обслуживания (измерения или формирования импульсного сигнала) сразу для нескольких каналов.

Первые два недостатка устранены в усовершенствованном модуле интервального таймера/счетчика, применяемом в ряде микроконтроллеров (например, Intel MCS-51). Дополнительная логика счетного входа позволяет тактовым импульсам поступать на вход счетчика, если уровень сигнала на одной из линий ввода соответствует логической единице. Такое решение повышает точность измерения временных интервалов, так как пуск и остановка таймера производится аппаратно. Кроме того, в усовершенствованном модуле таймера/счетчика реализован режим аппаратной перезагрузки счетчика произвольным кодом в момент переполнения. Это позволяет формировать временные последовательности с периодом, отличным от периода полного коэффициента счета.

Широтно-импульсный модулятор

Встроенный широтно-импульсный модулятор (ШИМ) предназначен для генерации широтно-модулированного сигнала на выходе микросхемы без участия процессора. На таймер блока подается системный синхросигнал (возможно, через предделитель). В регистр периода процессор записывает число, соответствующее периоду ШИМ. При совпадении содержимого этого регистра и таймера последний сбрасывается в нулевое состояние и на соответствующем выходе микросхемы формируется положительный перепад. В регистре длительности процессор записывает число, соответствующее длительности импульса. При совпадении его содержимого с содержимым таймера на выходе микросхемы формируется отрицательный перепад.

При такой организации возможная разрядность ШИМ зависит от частоты сигнала. В некоторых моделях микроконтроллеров используются специализированные многоканальные модули ШИМ, в частности для управления приводом (Intel 80C196MC).

Цифровой компаратор

Цифровой компаратор осуществляет аппаратную операцию сравнения двух операндов. Он представляет собой интегрированную в микроконтроллер электронную схему, состоящую из программно-доступного регистра для записи одного из операндов и логических элементов, реализующих функцию равнозначности. Другой операнд чаще всего представлен содержимым таймера/счетчика. При совпадении значений обоих операндов вырабатывает логический сигнал – признак. Этот признак может быть опрошен программно или аппаратно.

4.3.2. Модули захвата и сравнения

Все усовершенствования интервального таймера/счетчика не устраняют его главного недостатка – одноканального режима работы. Имеют место два направления совершенствования подсистемы реального времени микроконтроллера. В первом случае увеличивают количество модулей таймеров/счетчиков (микроконтроллеры со структурой Intel MCS-51/52, Mitsubishi и Hitachi). Во втором случае принципиально изменяют структуру модуля таймера/счетчика, при которой увеличение числа каналов достигается не за счет увеличения числа счетчиков, а за счет введения дополнительных аппаратных средств: модулей входного захвата (Input capture — IC) и выходного сравнения (Output compare — OC). По такому пути идут в микроконтроллерах Intel (80xc51FA, MCS-96, MCS-251) и Motorola). Эти модули, называемые ещё модулями быстрого ввода-вывода, предназначены для быстрой генерации события на выходе микросхемы либо быстрой реакции на событие на её входе без участия процессора. Они имеют два основных режима работы – входной и выходной.

Модуль входного захвата

Таймер модуля считает импульсы системного синхросигнала, проходящие, возможно, через предделитель частоты, создавая временную базу. Схема детектора события “наблюдает” за уровнем напряжения на одном из входов микроконтроллера. В большинстве случаев – это одна из линий порта ввода/вывода. Изменение уровня логического сигнала (фронт или спад) воспринимается как заданное событие – событие захвата. В момент, когда на соответствующем входе микросхемы происходит событие захвата, вырабатывается строб записи и содержимое таймера фиксируется в регистре времени события. Таким образом, запоминается время события, которое затем может быть считано процессором по соответствующей команде. Этот режим исполь-

зуется, в частности, для измерения частоты и длительности входного импульсного сигнала.

Обычно предусмотрена возможность выбора типа сигнала на входе, воспринимаемого как событие: положительный (передний) фронт сигнала, отрицательный (задний) спад сигнала, любое изменение логического уровня сигнала. Выбор типа события захвата устанавливается в процессе инициализации таймера и может неоднократно изменяться в ходе выполнения программы. Каждое событие захвата приводит к установке триггера входного захвата и появлению на его выходе флага (признака) входного захвата. Признак может быть считан программно, а если прерывания по событию захвата разрешены – формируется запрос на прерывание.

Модуль выходного сравнения

Таймер модуля считает импульсы, поступающие с постоянной частотой, соответствующей сигналу синхронизации (возможно, деленной на предварительном делителе). Таймер работает непрерывно, создавая временную базу. Цифровой компаратор также непрерывно сравнивает текущее состояние таймера с кодом, который записан в регистре выходного сравнения. В момент равенства кодов на одном из выходов микроконтроллера устанавливается заданный уровень логического сигнала. Обычно предусмотрено три типа изменения сигнала на выходе в момент события выходного сравнения: установка высокого логического уровня; установка низкого логического уровня; инвертирование сигнала на выходе. При наступлении события сравнения устанавливаются триггер выходного сравнения и соответствующий ему признак выходного сравнения. Аналогично режиму входного захвата состояние триггера выходного сравнения может быть считано программно, а если прерывания по событию сравнения разрешены – формируется запрос на прерывание. Таким образом, можно запрограммировать определённое выходное событие в заданное время и осуществить его без участия процессора. Режим выходного сравнения предназначен прежде всего для формирования временных интервалов заданной длительности.

Аппаратные средства усовершенствованного таймера позволяют решить многие задачи управления в реальном времени. Однако по мере роста сложности алгоритмов управления отчетливо проявляются ограничения модулей усовершенствованного таймера, а именно недостаточное число каналов захвата и сравнения, принадлежащих одному счетчику временной базы. Это не позволяет сформировать синхронизированные между собой многоканальные импульсные последовательности; однозначно определенная конфигурация канала (или захват, или сравнение) часто не удовлетворяет потребностям решае-

мой задачи; формирование сигналов по методу широтно-импульсной модуляции требует программной поддержки, что снижает максимально достижимую частоту выходного сигнала.

Контроллер периферийных событий

Этот контроллер предназначен для обеспечения некоторого блока событий в заданное время без участия процессора. Заданные события выполняются на микропрограммном уровне. Такими блоками событий могут быть передача блока информации из одного места памяти (или устройства ввода-вывода) в другое, последовательный опрос нескольких каналов АЦП, передача информации по последовательному каналу связи. Использование такого механизма обработки событий позволяет снизить загрузку процессора и распараллелить процесс обработки информации.

4.3.3. Процессор событий

Следующим этапом развития модулей подсистемы реального времени микроконтроллеров являются процессоры событий. Впервые модули процессоров событий были использованы компанией Intel в МК семейства 8xC51Fx. Этот модуль получил название программируемого счетного массива (Programmable Counter Array – PCA). Другое название – процессор событий.

PCA обеспечивает более широкие возможности работы в реальном масштабе времени и в меньшей степени расходует ресурсы центрального процессора, чем стандартный и усовершенствованный таймеры/счетчики. К преимуществам PCA также можно отнести более простое программирование и более высокую точность.

В микроконтроллере Intel 8xC51Fx процессор событий состоит из 16-разрядного таймера-счетчика и пяти 16-разрядных модулей сравнения-захвата (compare-capture) (рис. 4.3). Таймер PCA является базой

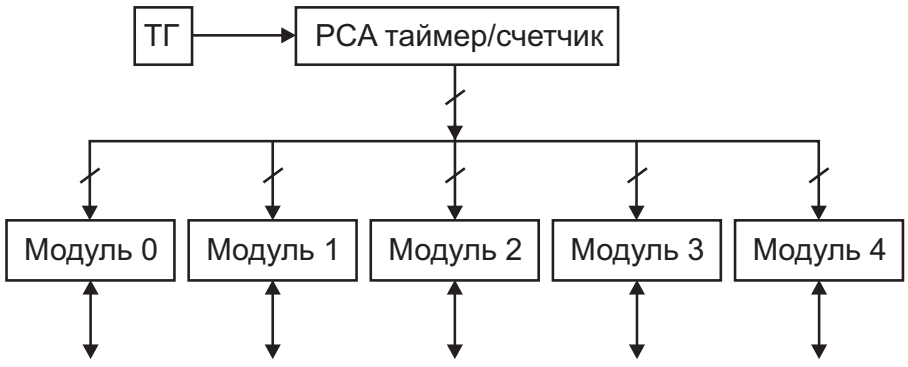


Рис. 4.3. Организация процессора событий фирмы Intel

времени для пяти модулей и единственным таймером, обслуживающим PCA. На его счетный вход могут подаваться следующие сигналы:

- выход делителя на 12 тактового генератора МК;
- выход делителя на 4 тактового генератора МК;
- сигнал переполнения таймера 0;
- внешний входной сигнал на выводе ECI (P1.2).

Любой из модулей сравнения-защелки может быть запрограммирован для работы в следующих режимах: защелкивания по фронту и/или спаду импульса на входе CEXi; программируемого таймера; высокоскоростного выхода; широтно-импульсного модулятора.

Режим защелкивания по импульсу на входе МК эквивалентен режиму входного захвата (IC) усовершенствованного таймера.

Режимы программируемого таймера и высокоскоростного выхода близки по своим возможностям к режиму выходного сравнения (OC).

В режиме ШИМ на соответствующем выводе МК формируется последовательность импульсов с периодом, равным периоду базового таймера/счетчика PCA. Значение 8-разрядного кода, записанное в младший байт регистра-защелки соответствующего модуля, задает скважность формируемого сигнала. При изменении кода от 0 до 255 скважность меняется от 100 до 0.4%.

Режим ШИМ очень прост с точки зрения программного обслуживания. Если изменения скважности не предполагается, то достаточно один раз занести соответствующий код в регистр данных модуля, инициализировать режим ШИМ – импульсная последовательность будет воспроизводиться с заданными параметрами без вмешательства программы.

Модуль 4 может быть также запрограммирован как сторожевой таймер.

4.4. Дополнительное встроенное оборудование

4.4.1. Модули преобразования данных

Аналоговый компаратор

Аналоговый компаратор осуществляет аппаратную операцию сравнения уровней двух внешних аналоговых сигналов. Он представляет собой электронную схему компаратора, интегрированную в микроконтроллер. Выходной логический уровень аналогового компаратора может быть опрошен программно или аппаратно.

Модули АЦП и ЦАП

Модуль аналого-цифрового преобразователя (АЦП) предназначен для преобразования входной аналоговой информации в цифровую и передачи ее в процессор для дальнейшей обработки. Модуль цифро-аналогового преобразователя (ЦАП) выполняет обратную операцию. Микроконтроллеры могут иметь несколько каналов ЦАП и АЦП. Преобразование информации на том или ином канале инициализируется соответствующей командой.

Преобразования начинаются выбором требуемого канала, который осуществляется записью номера канала в специальный служебный регистр. Аналоговый мультиплексор пропускает на выход сигнал выбранного канала. После этого подается сигнал фиксации на устройство выборки – хранения. Оно фиксирует уровень выбранного аналогового сигнала на весь период преобразования. Затем подается сигнал начала преобразования. АЦП представляет собой, как правило, АЦП последовательных приближений. Время преобразования составляет обычно единицы миллисекунд, однако в некоторых новейших микроконтроллерах (ADuC812 фирмы Analog Devices) может доходить до 5 мкс. После окончания преобразования АЦП выдает сигнал готовности, по которому его выходная информация записывается в буферный регистр, откуда она затем может быть считана процессором по соответствующей команде. При этом вырабатывается запрос на прерывание. Некоторые микроконтроллеры имеют блоки АЦП, способные работать в автоматическом режиме, опрашивая несколько каналов последовательно без участи процессора (80C166, Intel).

4.4.2. Модули мониторирования состояния

Для минимизации повреждения кода программы или данных вследствие возникновения катастрофических программных или внешних сбоев системы микроконтроллеры комплектуются мониторными функциями. Наиболее часто встречаются две мониторные функции, конфигурируемые программно через специальные регистры.

Сторожевой таймер

Сторожевой таймер (Watch Dog Timer – WDT) служит для предотвращения зависания программы в необслуживаемых приложениях, т.е. в системах, работающих в автоматическом режиме без участия оператора. Источником счетных импульсов служит либо системный синхросигнал с фиксированной частотой (возможно, предварительно деленной на заданное число), либо автономный генератор импульсов фиксированной частоты. Высокая стабильность частоты генератора

импульсов не требуется, поэтому для его построения часто используют простой RC-генератор прямоугольных импульсов.

Действие сторожевого таймера разрешается установкой специального бита в регистре управления WDT. Сброс данного бита запрещает действие WDT. При наличии разрешения сторожевой таймер будет формировать системный сброс, если программа пользователя не обновляет его содержимое в интервале предустановленного времени. Интервал можно программно изменять от миллисекунд до секунд путем установки соответствующего значения в специальном регистре.

Технология защиты критического участка программы следующая. В начале защищаемого участка устанавливаются бит разрешения WDT и величина интервала ожидания. В конце участка производится запрет WDT или обновление его содержимого. В правильно работающей программе счетчик сторожевого таймера никогда не достигнет переполнения. Если же на защищаемом участке выполняемой программы произойдет сбой, то сторожевой таймер сформирует сигнал аппаратного сброса микроконтроллера, который возобновит свою работу с самого начала.

Супервизор питания

Супервизор (монитор) питания – BROD или BROWN OUT – формирует запрос прерывания, когда напряжение питания падает ниже определённого порогового значения. Бит прерывания не будет очищаться в течение установленного интервала времени (например, не менее 256 мс) и до тех пор, пока напряжение источника не станет выше порогового значения.

Эта функция гарантирует, что пользователь успеет спасти рабочие регистры во избежание возможной потери данных из-за низкого питания. При наличии энергонезависимой памяти данных (EEPROM) супервизор питания является весьма эффективным средством защиты от сбоев по питанию. Выполнение программного кода не продолжится до тех пор, пока не установится номинальный уровень питания. Монитор прерывания обычно защищен от импульсных помех в цепи прерывания.

4.5. Критерии выбора микроконтроллера для проекта

Выбор конкретного типа микроконтроллера является одним из самых важных решений, от которого зависит успех или провал задуманного проекта. При выборе микроконтроллера необходимо учесть и оценить большое количество факторов. За основу последовательности продуманных действий, приводящих к окончательному решению,

может быть принят следующий алгоритм, составленный зарубежными специалистами[1]. Для того чтобы принять правильное решение, результаты действий, предусмотренные данным алгоритмом, необходимо анализировать в свете своих собственных навыков, знаний и требований.

Цель анализа. Основная цель анализа – выбрать наименее дорогой микроконтроллер (чтобы снизить общую стоимость системы), но в то же время удовлетворяющий спецификации системы, т.е. требованиям по производительности, надежности, условиям применения и т.д. Общая стоимость системы включает все: инженерные исследования и разработку, производство (комплектующие и труд), гарантийный ремонт, дальнейшее усовершенствование, обслуживание, совместимость, простоту в обращении и т.д.

Процесс выбора. Приступая к выбору, разработчик должен вначале задаться вопросом: *что должен делать микроконтроллер в моей системе?* Ответ на этот простой вопрос определяет требуемые для разрабатываемой системы характеристики микроконтроллера, что является определяющим фактором в процессе выбора.

Второй шаг – проведение поиска микроконтроллеров, которые удовлетворяют всем системным требованиям. Он обычно включает подбор литературы, технических описаний и технических журналов, а также консультации. В настоящее время стала вполне доступной информация о предлагаемых как традиционных, являющихся промышленным стандартом микроконтроллерах, так и новейших микроконтроллерах. Хорошо, если системным требованиям будет удовлетворять хорошо знакомый микроконтроллер. В противном случае должен быть проведен вторичный поиск, чтобы найти микроконтроллер, который наиболее полно удовлетворяет предъявляемым требованиям, имеет минимум внешних навесных компонентов и подходит по стоимости и габаритам. Однокристальный микроконтроллер предпочтительней из-за надежности и низкой цены.

Последняя стадия выбора состоит из нескольких этапов, цель которых – сузить список приемлемых микроконтроллеров до одного. Эти этапы включают в себя анализ цены, доступности, средств разработки, поддержки производителя, стабильности производства конкретных микроконтроллеров и наличия других производителей или поставщиков. Для того чтобы прийти к оптимальному решению, возможно, весь процесс придется повторить несколько раз.

[1]Перевод ТОО "Торнадо Модульные Системы", Россия, Новосибирск.

Критерии выбора

Основные критерии выбора микроконтроллера представлены ниже в порядке значимости. Каждый критерий детально объясняется в дальнейшем.

Пригодность для прикладной системы:

- Может ли система быть сделана на однокристальном микроконтроллере или ее можно реализовать на основе какой-либо специализированной микросхемы?
- Имеет ли микроконтроллер требуемое число контактов/портов ввода-вывода (в случае их недостатка он не сможет выполнить работу, а в случае избытка цена будет слишком высокой)?
- Имеет ли он все требуемые периферийные устройства: последовательные порты ввода-вывода, РПД, РПП, АЦП, ЦАП и т.д.?
- Имеет ли он другие периферийные устройства, которые не потребуются в системе?
- Обеспечивает ли ядро процессора необходимую производительность, т.е. вычислительную мощность, позволяющую обрабатывать системные запросы в течение всей жизни системы на выбранном прикладном языке? Слишком много – расточительно, слишком мало – не будет работать.
- Выделено ли в бюджете проекта достаточно средств, чтобы позволить себе использовать данный микроконтроллер. Для ответа на этот вопрос обычно требуются расценки поставщика. Если данный микроконтроллер неприемлем для проекта, все остальные вопросы становятся несущественными, и вы должны начать поиски другого микроконтроллера.

Доступность:

- Существует ли устройство в достаточных количествах?
- Производится ли оно сейчас?
- Что ожидается в будущем?

Поддержка разработчика:

- ассемблеры;
- компиляторы;
- средства отладки:
  - оценочный модуль (плата развития);
  - внутрисхемные эмуляторы;
  - насадки для логических анализаторов;
  - отладочные мониторы;
  - отладчики программ в исходных текстах.

Информационная поддержка:

- примеры применения;
- сообщения об ошибках;
- утилиты, в том числе *бесплатные* ассемблеры;
- примеры исходных текстов.

Поддержка приложений у поставщика:

- Есть ли специальная группа, которая занимается только поддержкой приложений?
- Есть ли инженеры, техники или продавцы?
- Насколько квалифицирован поддерживающий персонал, действительно ли он заинтересован в помощи вам при решении вашей проблемы?
- Существует ли телефонная и/ или факсимильная связь?

Надежность фирмы производителя:

- компетентность, подтвержденная разработками;
- надежность производства, т.е. качество продукции;
- время работы в этой области.

Системные требования

Проведение системного анализа вашего проекта позволит определить и требования к микроконтроллеру. Какие требуются периферийные устройства? Применяются ли битовые операции или только числовые? Сколько требуется манипуляций для обработки данных? Должна ли система управляться по прерываниям, по готовности или по командам человека? Каким количеством устройств (битов ввода-вывода) необходимо управлять? Какие устройства из числа многих возможных типов I/O устройств должны контролироваться и управляться: терминалы, выключатели, реле, клавиши, сенсоры (температура, свет, напряжение и т.д.), звуковые устройства, визуальные индикаторы (LCD-дисплеи, LED), аналого-цифровые, цифро-аналоговые преобразователи? Одно или несколько напряжений питания требуется для системы? Насколько отказоустойчив источник питания? Будет ли работать устройство при напряжении вашей сети питания? Должны ли напряжения удерживаться в узком фиксированном диапазоне изменений или же система может работать при большой нестабильности? Какой рабочий ток? Изделие должно работать от сети или от батарей? Если от батарей – должны ли использоваться перезаряжаемые батареи? Если это так, то каково время работы без перезарядки и какое для нее требуется время?

Существуют ли ограничения по размеру, весу, эстетическим параметрам, таким как форма и/или цвет? Существуют ли такие специфические требования к условиям окружающей среды, как военные условия, температура, влажность, атмосфера (взрывоопасная, коррозийная и т.д.), давление/ высота? Пользовательское программное обеспечение должно базироваться на дисках или ROM? Изделие работает в реальном времени? Если да – собираетесь ли вы создать или приобрести ядро программ реального времени или, возможно, будет достаточно обычной широко используемой версии? Достаточно ли персонала и времени для развития вашего собственного ядра программ? Как будут оплачиваться авторские права и программное обеспечение? Для решения задач реального времени требуется большая исследовательская работа, чтобы удовлетворить их особым требованиям.

Основные особенности микроконтроллера

Микроконтроллеры в целом можно разделить на группы 8-, 16- и 32-разрядных по размеру их арифметических и индексных регистров, хотя некоторые разработчики считают, что 8/16/32-разрядную архитектуру определяет разрядность шины. Способен ли дешевый микроконтроллер удовлетворить требованиям системы или требуется дорогой 16- или 32-разрядный? Может ли 8-разрядная программная эмуляция особенностей 16/32-разрядного микроконтроллера разрешить использование дешевого 8-разрядного, жертвуя размером исполняемого кода и скоростью? Например, может ли 8-разрядный микроконтроллер быть использован с программным макросом, чтобы эмулировать 16-разрядный аккумулятор и операции индексирования? Выбор прикладного языка (высокого уровня вместо ассемблера) может сильно повлиять на производительность системы, которая затем будет диктовать выбор 8/16/32-разрядной архитектуры, но ограничение цены может отвергнуть этот выбор.

Тактовая частота или скорость шины определяет, сколько вычислений может быть выполнено за единицу времени. Некоторые микроконтроллеры, в основном ранних разработок, имеют узкий диапазон допустимой тактовой частоты, в то время как другие могут работать вплоть до нулевой частоты. Иногда выбирается специфическая тактовая частота, чтобы сгенерировать другую тактовую частоту, требуемую в системе, например, для задания скоростей последовательной передачи. В основном вычислительная мощность, потребляемая мощность и стоимость системы увеличиваются с повышением тактовой частоты. Цена системы при повышении частоты увеличивается из-за стоимости не только микроконтроллера, но также и всех требующихся дополнительных микросхем, таких как ОЗУ, ПЗУ, программируемые логические интегральные схемы и контроллеры шины.

Рассмотрим технологию, с использованием которой изготовлен микропроцессор *n*-МОП (NMOS) и которая использовалась в микроконтроллерах ранних разработок, сравним с современной CMOS технологией с высоким уровнем интеграции (HCMOS). В отличие от ранних NMOS-процессоров, в HCMOS уровни сигналов изменяются в диапазоне от 0 до уровня напряжения питания. В связи с этим обстоятельством предпочтение отдается HCMOS процессорам. Кроме того, HCMOS потребляют меньшую мощность и, следовательно, меньше нагреваются. Геометрические размеры элементов в HCMOS меньше, что позволяет иметь более плотные схемы и, таким образом, работать при более высоких скоростях. Более плотный дизайн также уменьшает стоимость отдельного микроконтроллера, так как на кремниевой пластине того же размера можно получить большее количество чипов. По этим причинам сегодня подавляющее большинство микроконтроллеров производятся с использованием HCMOS-технологии.

Возможности микроконтроллера

За счет достижения более высокого уровня интеграции и надежности при сохранении низкой цены все микроконтроллеры оснащены встроенными дополнительными устройствами. Эти устройства под управлением микропроцессорного ядра микроконтроллера выполняют определенные функции. Встроенные устройства обладают повышенной надежностью, поскольку они не требуют никаких внешних электрических цепей. К наиболее известным встроенным устройствам относятся устройства памяти и порты ввода-вывода, таймеры, системные часы/генератор. Отметим, что таймеры включают в себя как часы реального времени, так и таймеры прерываний. Следует принимать во внимание диапазон и разрешение таймера так же, как и другие функции (функции сравнения и/или захвата входных линий при измерении длительности сигнала). Средства ввода-вывода включают последовательные порты связи, параллельные порты (линии ввода-вывода), аналого-цифровые преобразователи, цифро-аналоговые преобразователи, драйверы жидкокристаллического дисплея или драйверы вакуумного флуоресцентного дисплея.

Другими, реже используемыми встроенными ресурсами являются внутренняя/ внешняя шина, таймер слежения за нормальным функционированием системы, сторожевая схема, система обнаружения отказов тактового генератора, возможность выбора конфигурации памяти и системный интеграционный модуль (SIM). SIM обычно заменяет внешнюю *склеивающую* логику, необходимую для организации взаимодействия микроконтроллера с внешними устройствами через заданные контакты микросхемы.

В большинство микроконтроллеров с внутрисхемными ресурсами включается блок конфигурационных регистров для управления этими ресурсами. Иногда сам этот блок может быть отражен в различные места карты памяти. Иногда имеется пользовательский и/или фабричный тестовый регистр, указывающий на то, какое значение производитель придает качеству. Наличие конфигурационных регистров приводит к проблеме случайного изменения желаемой конфигурации *блуждающим* кодом. Для предотвращения такой случайной возможности используется механизм *блокировки*. Иными словами, до того, как регистр конфигурации может быть изменен, биты в другом регистре должны быть изменены в определенной последовательности. Хотя регистры конфигурации могут сначала испугать своей сложностью, но они крайне ценны, поскольку обеспечивают высокую гибкость конфигурации при низкой стоимости, так что одному микроконтроллеру можно найти самые различные применения.

Набор команд микроконтроллера

Необходимо внимательно изучить набор команд и регистров каждого микроконтроллера, так как они играют важнейшую роль в определении возможностей системы в целом. Изучили ли ваши программисты индексные режимы адресации в связи с предполагаемыми нуждами вашей системы? Есть ли такие специальные команды, которые будут использоваться в вашей системе, как умножение, деление и табличное интерполирование? Есть ли такие режимы энергосбережения для экономии батарейного питания, как стоповый, стоповый с низким потреблением мощности и/или с ожиданием? Есть ли какие-либо команды битовых манипуляций (установка бита, очистка бита, тест бита, изменение бита, команды перехода по установленному/очищенному биту), облегчающие применение микроконтроллера, или команды манипуляции с битовыми полями?

Будьте осторожны с замечательными командами, которые совершают много действий в одной команде. Реальным критерием производительности является количество тактовых циклов, требуемое для выполнения задачи, а не количество исполненных команд. Для справедливого сравнения лучше закодировать одинаковую программу и сравнить полное число выполненных тактовых циклов и использованных байтов. Есть ли в карте операционных кодов нереализованные инструкции, и что получится, если они случайно выполнятся? Обработает ли система подобную ситуацию корректно обработчиком *исключительных* событий или это приведет к выходу системы из строя?

Прерывания микроконтроллера

Проверка структуры прерываний необходима всегда, когда создается система реального времени. Сколько линий или уровней прерывания имеется и сколько их требуется для вашей системы? Имеется ли маска уровней прерывания? Когда уровень прерывания подтвержден, есть ли индивидуальные векторы для программы обработчика прерывания или должны опрашиваться все возможные источники прерывания, чтобы определить источник? В критических по скорости применениях, таких как управление принтером, критерием выбора подходящего микроконтроллера может быть время реакции на прерывание, т.е. время от начала прерывания (в худшем случае – фазированного относительно тактового генератора микроконтроллера) до выполнения первой команды соответствующего обработчика прерывания.

Характеристика вашей лаборатории

Критически проанализируйте имущественное состояние вашей лаборатории. Располагает ли ваша лаборатории достаточными средствами для обучения персонала тонкостям производства систем на основе микроконтроллеров и использования средств их разработки? Обладает ли ваша лаборатория достаточными средствами разработки или же вы будете покупать либо арендовать их? Если рассматривается новый микроконтроллер, существуют ли доступные средства разработки, такие как компиляторы языка высокого уровня, ассемблеры/компоновщики, прототипные модули и отладчики/эмуляторы? Достаточно ли легко расширяются имеющиеся у вас средства разработки для новых микроконтроллеров? Нужно ли нанимать и обучать дополнительный персонал для этого проекта? Можете ли вы привлечь эксперта для обучения остальных членов вашей команды? Позволяет ли вам бюджет наем дополнительного постоянного штата и/или работников по контракту? Удовлетворена ли ваша лаборатория микроконтроллерами, имеющимися в настоящее время на рынке, а также обслуживанием?

Характеристика поставщика

Третий шаг в сокращении списка технически приемлемых микроконтроллеров – проверка их производителей и поставщиков, т.е. компаний, с которыми вы планируете вступить в длительные отношения на взаимовыгодной основе. Поставщик может быть производителем микроконтроллеров или дилером, который является полномочным представителем нескольких производителей. Наилучшим образом удовлетворит ваши запросы поставщик с более широким ассор-

тиментом продуктов и репутацией высокого качества, надежности, обслуживания и своевременной поставки при справедливой цене. Кроме того, чем больше продуктов вы покупаете у одного поставщика, тем большие преимущества вы получаете в отношении цены, услуг и поддержки. Всегда имейте в виду, что, хотя долларовый объем вашей покупки может казаться вам высоким, это всегда относительная величина к общему объему продаж поставщика. Поставщики, которые снабжают не только микроконтроллерами, но и памятью (RAM, ROM), дискретными устройствами (транзисторами, диодами и т.д.), стандартными цифровыми логическими устройствами (7400, 74HC00 и т.д.), специальными микросхемами, заказными приборами, специализированными микросхемами и программируемыми логическими устройствами, смогут лучше удовлетворить ваши растущие запросы. Имеет ли производитель и/или поставщик какие-либо награды за качество, надежность, сервис и/ или поставку? Не следует слишком доверять самоприсуждаемым наградам.

Характеристика производителя

Другими критериями в выборе производителя/поставщика микроконтроллера являются стабильность, его монопольное положение, сведения из литературы и поддержка. Стабильность может быть надежно проверена путем установления стажа работы производителя в этой области и его достижений. Отдел снабжения и кредитный отдел вашей компании могут помочь вам в этих вопросах. Монопольное положение поставщика, к сожалению, обычно норма, так как большинство производителей микроконтроллеров редко пересекаются в производстве с другими производителями. Если производитель имеет хорошие показатели в снабжении, доставке и цене, то его монопольное положение не должно являться препятствием.

Поддержка производителя

Прямая поддержка производителя включает маркетинг/продажи и прикладную инженерную поддержку. Когда вы звоните, обращаясь за помощью, можете ли вы прямо связаться с тем, кто вам нужен, или вам приходится играть в *глухой телефон*? Передаются ли звонки немедленно? Есть ли номер факса? Сколько телефонных линий доступно? Телефонные линии всегда заняты? Есть ли у них система коммутации или секретарь передает ваши сообщения ответственному за поддержку? В какие часы работает персонал поддержки? Имеют ли они другие обязанности, кроме поддержки? Каков количественный состав обслуживающего персонала? С готовностью ли ему поможет заводской персонал, а именно специалисты по готовой продукции, по

производству, по качеству, электронщики, программисты? Дружат ли заводские инженеры с персоналом поддержки? Знающий ли персонал поддержки, имеет ли нужные навыки, и выполняют ли они своевременно то, что обещали, например решить вашу проблему или выслать вам что-нибудь? Приходит ли это обычной почтой, платите ли вы за быструю доставку? Есть ли у производителя электронная доска объявлений или страничка в Internet, на которых можно получить такую информацию, как прикладные программы, новости о продуктах, свежие программы, исходные тексты, сообщения об ошибках, электронную почту, конференции? Какие поддерживаются скорости передачи? Сколько телефонных линий доступно? Какие часы работы? Нужна ли вам особая марка компьютера и/или модема для доступа? Есть ли системный оператор (sysop)?

Информационная поддержка

Литература охватывает широкий набор печатных материалов, которые могут помочь вам сделать правильный выбор. Она включает информационные выпуски производителя, такие как технические описания и рекомендации по применению, а также издания, доступные в местном книжном магазине и/или библиотеке. Издания из местного магазина и/или библиотеки не только указывают на популярность производителя/микроконтроллера, но и предлагают беспристрастные мнения, если они высказаны независимыми от производителя авторами.

Заключительный этап выбора

Для окончательного шага в процессе выбора постройте таблицу, в которой расположите рассматриваемые микроконтроллеры в одной графе, а их важные характеристики – в другой. Затем приложите бланки технических описаний производителей, чтобы получить справедливое наглядное сравнение. Некоторые производители имеют предварительно сделанные сравнительные описания их микроконтроллеров, которые упростят вашу задачу; но проверьте по техническим описаниям, все ли новейшие продукты представлены. Среди возможных характеристик – цена (на ожидаемый объем продукции, включая предсказание будущей цены, т.е. уменьшится ли цена, если вы вольетесь в производство?), RAM, ROM, EPROM, EEPROM, таймер(ы), АЦП, ЦАП, последовательные порты, параллельные порты, скорость шины (минимальная/максимальная), специальные команды (умножение, деление и т.д.), число доступных прерываний, время отклика прерывания, размер корпуса и его тип, требования по питанию и другие детали, важные для устройства вашей системы.

Если после всего этого у вас в списке все еще больше одного микроконтроллера, рассмотрите возможности расширения системы, а также стоимость. Какое расширение, по вашему мнению, может понадобиться в будущих версиях этого продукта? И наконец, рассмотрите цену: если два микроконтроллера стоят одинаково, но один предлагает немного больше возможностей, которые не требуются сегодня, но сделали бы будущие расширения доступными без добавочных затрат, выбирайте этот микроконтроллер.

Окончательный выбор подходящего микроконтроллера для вашего проекта – нелегкое решение. Микроконтроллеры стали более сложными устройствами с тех пор, как были добавлены внутрисхемные ресурсы. И с тех пор, как процесс движется в сторону все большей внутрисхемной интеграции внешних ресурсов для понижения стоимости системы, решение становится все более сложным. Данный алгоритм не навязывает разработчику какой-либо выбор, его цель – указать все возможные критерии выбора, которые должны быть приняты во внимание в процессе принятия решения.

5. СРАВНИТЕЛЬНЫЙ АНАЛИЗ НЕКОТОРЫХ МИКРОКОНТРОЛЛЕРОВ

Конкретный набор встроенных устройств, их характеристики и возможные режимы работы зависят от выбранной модели микроконтроллера. В табл. 5.1 сопоставлены параметры микроконтроллеров нескольких различных типов. Рассмотрим более подробно характеристики микроконтроллеров разных семейств.

Таблица 5.1. Сопоставление параметров некоторых микроконтроллеров

Тип МК	РПП	РПД		f_T/n	I/O	Таймер	Дополнительное оборудование
		1	2				
PIC12C508	512×12	25	–	4/4	6	1	
PIC15C55	1024×12	24	–	20/4	20	1	
PIC1400	4096×14	192	–	20/4	22	2	2×I2C/SPI, 2ЦАП 8-кан.16-бит-АЦП
PIC16F676	1024×14 Flash	62	128	20/4	12	2	8-кан.10-бит-АЦП, ИОН, ан.комп., WDT
PIC17C44	9196×16	454	–	33/4	33	4	USART
PIC18F248	8192×16 Flash	768	256	40/4	23	4	5кан.10-бит-АЦП CAN 2ан.комп. MI2C/MSPI WDT ШИМ USART
AT90S1200	1K	–	64	12/1	15	1	Ан.комп/SPI
AT90S4414	4K	256	256	8/1	32	2	Ан.комп. SPI UART
ATmega103	128K	4K	4K	6/1	48	3	Ан.комп. SPI UART
SX28AC100	2048×12	136	–	100/1	20	1	Ан.комп
SX52BD	4096×12	256	–	50/1	40	3	Ан.комп
AT89C1051	1K	64	–	24/6	15	2	Ан.комп
AT89C51	4K	128	–	24/8	15	2	UART
КР1816ВЕ51	4K	128	–	12/6	32	2	UART
КР1878ВЕ1 (An15E03)	1024×16 Flash	128	64	4/2	12	1	WDT
ADuC812BS	8K Flash	256	640	12/6	32	3	WDT, UART 2ЦАП, I2C/SPI 8-кан.12-бит-АЦП
Z86E31	2K	125	–	16/6	24	2	Ан.комп
Z86E33	2K	237	–	12/6	24	2	Ан.комп

Примечание. РПП – резидентная память программ, указана емкость ПЗУ и разрядность, если разрядность отличается от 8 бит; РПД – резидентная память данных: 1 – ОЗУ и 2 – EEPROM; f_T – тактовая частота, МГц; *n* – число машинных тактов на команду; I/O – число линий ввода-вывода; в колонке *Дополнительное оборудование* указано дополнительное встроенное оборудование.

5.1. Микроконтроллеры фирмы Microchip

PIC-микроконтроллеры разрабатываются и выпускаются американской фирмой Arizona Microchip Technology Inc. (Microchip). История развития этих микроконтроллеров насчитывает уже около 30 лет.

5.1.1. Историческая справка

В 1965 году компания General Instruments (GI) основала Отделение микроэлектроники (Microelectronics Division) и начала разрабатывать первые конкурентоспособные образцы перепрограммируемых запоминающих устройств с ультрафиолетовым (EPROM) и электрическим (EEPROM) стиранием. Впоследствии отделение GI Microelectronics Division выполняло разработку широкой гаммы цифровых и аналоговых устройств. Этим отделением разработаны, в частности, семейства звуковых процессоров AY3-××××, AY5-××××. В 70-80-е годы процессоры GI AY3-8910 и AY3-8912 применялись практически во всех микрокомпьютерах "Синклер", "Ямаха" и MSX.

В начале 70-х годов фирма GI разработала и выпустила 16-разрядный микропроцессор CP1600. Для ускорения операций ввода-вывода в микропроцессорных системах с данным процессором фирма GI в середине 1975 года разработала специальный однокристальный контроллер ввода-вывода – **P**eripheral **I**nterface **C**ontroller (сокращенно **PIC**). Основное назначение этого контроллера – ускоренное выполнение операций ввода-вывода при сравнительно небольшом объеме вычислительных операций. Это обусловило ограничения на его систему команд. Архитектура нижнего ряда сегодняшних PIC-микроконтроллеров PIC16C5x соответствует архитектуре, разработанной в 1975 году. Представленный в 1975 году PIC-контроллер производился по NMOS-технологии и был доступен только в масочном варианте.

В начале 80-х годов фирма GI произвела свою реструктуризацию, сконцентрировавшись на разработке силовых полупроводников. Отметим, что в настоящее время компания General Instruments под именем General Semiconductors продолжает весьма успешно работать в этом секторе рынка. Отделение GI Microelectronics Division было переименовано в GI Microelectronics Inc и существовало одно время как отдельный филиал, а затем было продано вместе с заводом в г. Чандлер штата Аризона (Chandler, Arizona). Это была длительная работа представителей рискового капитала: они избавились от производства большинства микросхем линеек AY3, AY5 и других и сконцентрировались на разработках и производстве PIC-микроконтроллеров и различных видов запоминающих устройств EPROM и EEPROM с параллельным и последовательным доступом. После переименования предприятие получило название Arizona Microchip Technology Inc.

(Microchip). Компания Microchip четко позиционирована на рынок встраиваемых систем управления (Embedded Control) и сейчас является одной из лидирующих компаний в своей области. Так, если в 1990 году по объему продаж она была на 20-м месте, в 1993-м – на 8-м, то в 1997 году, несмотря на сильнейшую конкуренцию со стороны фирмы Atmel с ее новыми микроконтроллерами типа AVR, компания Microchip вышла на 2-е место, пропустив впереди себя только фирму Motorola.

5.1.2. Основные особенности микроконтроллеров

Основные преимущества микроконтроллеров PIC заключены в их названии: аббревиатура PIC означает Peripheral Interface Controller, т.е. быстрый процессор ввода-вывода. Очень часто возникает необходимость в небольшом дешевом микропроцессоре, обладающем ограниченными вычислительными возможностями, низким энергопотреблением, развитой периферией и способном осуществлять ввод-вывод сигналов с большой скоростью. Это, например, питающееся от телефонной линии оборудование (АОНы, микроАТС), портативные приборы с батарейным питанием, средства автомобильной сигнализации, АЦП, подключаемые к последовательному порту компьютера и от него же и питающиеся, медицинские приборы и т.п. PIC-контроллеры выпускаются как в больших, так и малых корпусах и имеют низкую стоимость. Например, микроконтроллер PIC12C×××× собран в корпусе DIP (8-выводов) со встроенным тактовым генератором, цепочкой сброса и 6-ножками ввода-вывода. Контроллер PIC16C505 имеет корпус с 14-выводами, встроенным генератором и стоит 49 центов.

Большинство полезных качеств PIC-контроллеров сосредоточено в трех словах: CMOS, RISC, Harvard. Современная CMOS (КМОП) технология позволяет выпускать микроконтроллеры, работающие на частотах от 0 Гц до 40 МГц. Потребляемый ток, который зависит от частоты, составляет единицы миллиампер для частот 1…10 МГц и десяток микроампер для более низких частот. Перевод в спящий режим (SLEEP) доводит потребление до единиц микроампер. Благодаря RISC-системе команд и гарвардской архитектуре быстродействие даже на низких частотах остается достаточно высоким и требуется всего 4 такта на одну команду (8 тактов – для команд переходов). Для сравнения отметим, что в MCS51 машинный цикл составляет 12 тактов, а команды могут иметь несколько машинных циклов.

При традиционной архитектуре команды имеют переменную длину, а в RISC-архитектуре длина команд постоянна. Это обусловливает высокое быстродействие и экономию программной памяти. Три семейства PIC контроллеров имеют 12-, 14- и 16- разрядные команды и соответственно 33, 35 и 58 инструкций.

5.1.3. Встроенные аппаратные средства

Кроме экономного использования программной памяти, высокого быстродействия, низкого потребления и широкого диапазона питания (от 2 до 6 В для некоторых устройств), еще одним бесспорным достоинством PIC-контроллеров является развитость встроенных аппаратных средств. В большинстве контроллеров применен универсальный тактовый генератор, режим которого программируется на работу с одним из трех типов внешних кварцевых резонаторов (высокочастотный, среднечастотный и низкочастотный) или на режим с внешней RC-цепочкой (если стабильность тактовой частоты некритична). Некоторые типы контроллеров имеют режим встроенного генератора. При этом не требуется внешних цепей и не используются выводы корпуса.

Кроме тактового генератора, во всех контроллерах имеется внутренний RC-генератор, который используется для работы сторожевого таймера Watch dog. Этот генератор, если он включен при программировании, тактирует специальный счетчик, который должен сбрасываться программой через определенные промежутки времени, не превышающие заданный временной интервал. Если в результате зависания или некорректности работы программы счетчик не будет вовремя сброшен, произойдет перезапуск контроллера. Особым случаем применения сторожевого таймера является дежурный режим. Контроллер может находиться в состоянии “спячки” (SLEEP) с крайне низким энергопотреблением. При этом через временной интервал, равный периоду срабатывания сторожевого таймера, он может выходить из “спячки”, просматривать входные сигналы и, если ничего не произошло, возвращаться в состояние SLEEP. Если же произошли определенные изменения состояния входов, то микроконтроллер переходит к программе обработки, выполнив которую, он возвращается в состояние SLEEP. Если для обработки не требуется высокое быстродействие, то можно просто использовать низкую тактовую частоту. Однако когда нужно микропотребление энергии в среднем и большое быстродействие на некоторое время, то лучший выход – это именно использование спящего режима с периодическим пробуждением без или совместно с прерыванием по изменению. Сигнал СБРОС (MCLR) может формироваться внешними или внутренними цепями, что упрощает и удешевляет схемотехнику. Практически все контроллеры имеют хотя бы один счетчик-таймер, который может использоваться для счета внешних импульсов, генерации периодических прерываний или отсчета временных интервалов.

Связь с внешним миром осуществляется при помощи портов ввода-вывода, которые в PIC-контроллерах имеют свои особенности. В PIC-контроллерах каждому порту соответствует два регистра – данных и

направления (TRIS). При этом в зависимости от содержания регистра направления каждый вывод может быть индивидуально запрограммирован или на ввод, или на вывод. На всякий случай, во многих контроллерах имеется один вывод с открытым стоком, который требует подключения нагрузки при работе на выход. Все входы контроллеров защищены от перенапряжения диодами на 0 и Vcc и имеют высокую нагрузочную способность (20-25 мА). Некоторые порты имеют индивидуально подключаемые к входам источники тока, играющие роль подтягивающих (pull-up) резисторов. Отдельные входы могут быть запрограммированы так, что вызывают прерывание при изменении своего состояния (доступно также в спящем режиме).

Для работы с аналоговыми сигналами определенные типы контроллеров имеют встроенные АЦП или компараторы. Существуют контроллеры со встроенным управлением жидко-кристаллическими индикаторами, выходами ШИМ, разнообразными последовательными интерфейсами (синхронный, асинхронные I^2C, SPI), имеющими встроенную энергонезависимую память данных, для запоминания разных констант, настроек и т.п. Для работы вместе с более мощными микропроцессорными системами некоторые контроллеры имеют PSP (Parallel Slave Port).

5.1.4. Микроконтроллеры с Flash-памятью программ

Номенклатура микроконтроллеров с электрически перезаписываемой памятью программ непрерывно увеличивается. В настоящее время выпускается уже достаточно широкая номенклатура PIC-микроконтроллеров с Flash-памятью программ. В маркировке таких контроллеров обычно присутствует буква (F). К этой группе относятся, например, микроконтроллеры PIC12F629, PIC12F675, PIC16F87x, PIC16F87xA и др. Благодаря применению современных технологий большинство PIC- микроконтроллеров фирмы Microchip обеспечивают 100 тыс. циклов перезаписи Flash-памяти программ и 1 млн. циклов перезаписи EEPROM-памяти данных.

С середины 2002 года Microchip Technology Inc. выпускает 14-выводные Flash-микроконтроллеры, имеющие в своем составе 8-канальный 10-разрядный АЦП с программным выбором источника опорного напряжения (ИОН), компаратор с внутренним программируемым 32-уровневым источником опорного напряжения, прецизионный ($\pm 2\%$) внутренний тактовый генератор и функцию быстрого старта. Память микроконтроллеров PIC16F630 и PIC16F676 выполнена по технологии PMOS Electrically Erasable Cell (PEEC).

Микроконтроллеры PIC16F630 и PIC16F676 программно совместимы с 8-выводными контроллерами PIC12F629 и PIC12F675. Это позволяет проектировщикам без особых проблем перейти на микро-

контроллер с большим числом портов ввода-вывода. При этом снижаются затраты на проектирование и разработку устройства.

PIC16F630 и PIC16F676 имеют малый ток потребления в SLEEP-режиме (100 нА) во всем диапазоне напряжений питания (от 2.0 до 5.5 В). Быстрый запуск (2 мкс) и возможность работы таймера от внешнего кварцевого генератора в SLEEP-режиме делают микроконтроллеры PIC16F630 и PIC16F676 идеальными для устройств с пониженным энергопотреблением. Дополнительные периферийные модули могут быть программно включены или выключены с целью уменьшить энергопотребление.

Небольшие габаритные размеры, низкая стоимость и наличие модуля АЦП позволяют использовать микроконтроллеры PIC16F630 и PIC16F676 в портативных приборах, игрушках, автомобильных устройствах и т.д. Ориентировочная стоимость кристаллов при крупном заказе: PIC16F630 – $0.79, PIC16F676 – $0.91.

С середины 2003 года доступны 8-разрядные высокопроизводительные Flash-микроконтроллеры 18-й серии, например PIC18FXX8, имеющие от 28 до 80 выводов в зависимости от модификации. Все эти микроконтроллеры имеют CAN2.0B интерфейс. Приведем краткую характеристику PIC18F248. Микроконтроллер имеет сброс при включении питания (POR), таймер включения питания (PWRT), таймер запуска генератора (OST), программируемый сброс по снижению напряжения питания (BOR), сторожевой таймер (WDT) с собственным встроенным RC-генератором для повышения надежности работы, режим экономии энергии (SLEEP) и возможность выбора источника тактового сигнала. Контроллер позволяет программировать на плате через последовательный порт с использованием двух выводов (ICSPT), проводить отладку на плате через последовательный порт с использованием двух выводов (ICD), а также имеет программируемую защиту кода и возможность самопрограммирования. Встроенные запоминающие устройства контроллера рассчитаны на не менее чем 1 тыс. циклов записи/стирания Flash-памяти программ и 100 тыс. циклов записи/стирания памяти данных EEPROM с периодом хранения данных до 40 лет.

5.1.5. Программирование микроконтроллеров

Большинство PIC-контроллеров, не имеющих Flash-память, выполнено в исполнении OTP, т.е. однократно программируемый кристалл. Для их отладки используется специальный отладочный вариант JW с кварцевым окошком для стирания ультрафиолетовым излучением. Без крайней необходимости при отладке не следует устанавливать биты защиты (секретности).

Запись программы в кристалл может осуществляться двумя методами – параллельным и последовательным. Старое семейство (16C5××) и первые микросхемы серии 17C××× имели только параллельный режим записи, все остальные, кроме параллельного, имеют последовательный режим, который позволяет использовать более простые программаторы и осуществлять запись программы в установленную в плате микросхему (ISP – внутрисхемное программирование). Внутрисхемное программирование обеспечивает большую гибкость при производстве радиоэлектронных устройств, позволяя уменьшить стоимость обновления программного обеспечения, выполнить калибровку устройства, записать идентификационную и калибровочную информацию.

Для Flash-микроконтроллеров имеется мощный и недорогой внутрисхемный отладчик MPLAB-ICD2, работающий под управлением интегрированной среды проектирования MPLAB-IDE. MPLAB-ICD2 позволяет разработчику выполнять отладку текста программы по шагам и при этом наблюдать за состоянием регистров. Выполнение программы может быть запущено в реальном масштабе времени или в режиме анимации с указанием точки остановки.

По конструктивному исполнению PIC-контроллеры выпускаются в обычном пластмассовом корпусе (DIP), керамическом корпусе с окном (JW, отладочные варианты), различных корпусах типа SIOC, PLCC, PQFP и других, а также в виде кристаллов для гибридных интегральных схем. Микроконтроллеры PIC16F630 и PIC16F676 выпускаются в корпусах 14TSSOP, 14SOIC и 14DIP, а PIC18F248 – в корпусах 28DIP и 28SOIC 300mil.

5.2. Микроконтроллеры фирмы Scenix

Американская компания Scenix Semiconductor Inc. (Scenix) не стала изобретать принципиально новый микроконтроллер со своей системой команд, а, имея патент на быструю флэш-технологию, кардинально переработала и улучшила микроконтроллер семейства PIC16Cx фирмы Microchip. Это позволило создать микроконтроллеры с производительностью более 50 миллионов операций в секунду (mips), анонсированные в декабре 1997 года. Ещё через год появилась версия с производительностью 100 mips. С 2000 года выпускаются усовершенствованные микроконтроллеры SX48BD/SX52BD с расширенной памятью программ и данных.

SX-микроконтроллеры фирмы Scenix представляют собой 8-разрядные высокоскоростные микроконтроллеры с внутренней Flash-памятью программ объёмом 2К слов. Они выполнены с использованием современных методов проектирования и технологий и имеют

полностью статическую КМОП-реализацию, обеспечивающую работу при тактовых частотах от 0 до 50 МГц. RISC-подобная архитектура использует только 43 команды, большинство из которых выполняются за один такт. Команды перехода требуют два или три такта, а для команды IREAD требуется один или четыре такта при работе в режиме совместимости или в турборежиме.

Благодаря многим новым решениям в проектировании и производстве микроконтроллеры SX обеспечивают высокую экономическую эффективность. Тестирование микроконтроллеров SX показывает, что большинство кристаллов при гарантированной частоте 50 МГц могут работать на частотах до 150 МГц. Высокое быстродействие микроконтроллеров позволяет использовать их в качестве периферийных устройств, заменяя аппаратное решение программным. Такое решение обеспечивает уменьшение числа компонентов, время разработки, увеличивает гибкость проектов и в конечном счете – стоимость системы. Благодаря наличию специального интерфейса микроконтроллеры имеют возможность программирования непосредственно в устройстве.

5.2.1. Основные особенности микроконтроллеров

Основные характеристики SX-микроконтроллеров следующие:

- скорость выполнения команд – 50 миллионов операций в секунду при тактовой частоте 50 МГц;
- время выполнения каждой команды – один машинный такт;
- Flash-память программ емкостью 2048×12 бит с ресурсом 10 тыс. циклов записи;
- совместимость по программному коду и выводам с контроллерами PIC16C5x;
- программирование на плате через выводы OSC;
- пошаговый режим и останов в контрольных точках с использованием вывода OSC2;
- внутренний RC-генератор частотой 4 МГц±8 % с возможностью деления частоты (коэффициент деления от 1 до 128);
- выбираемый пользователем источник тактовой частоты (внутренний RC-генератор, внешний генератор, кварцевый генератор, внешний RC-генератор);
- встроенный аналоговый компаратор;
- схема перезапуска при уменьшении напряжения питания (4.2 В, отключаемая);
- многовходовая схема *пробуждения* контроллера из экономичного режима (8 выводов);
- все выходы обеспечивают втекающий/вытекающий ток до 30 мА;

- полный комплект средств разработки фирмы Parallax, Inc.

Особенности центрального процессора:

- цикл выполнения команды – 20 нс при частоте 50 МГц;
- полностью статическое построение обеспечивает работу на частотах от 0 до 50 МГц;
- 33 команды, совместимые с командами контроллера PIC16C5x, и 10 дополнительных команд для повышения эффективности кода;
- глубокий аппаратный стек подпрограмм (8 уровней);
- одноуровневый стек прерываний;
- фиксированное время реакции на прерывание: 60 нс на внутреннее и 100 нс на внешнее при тактовой частоте 50 МГц;
- аппаратное сохранение значений регистров PC, W, STATUS и FSR при прерывании;
- регистр W в пространстве RTCC (таймера/счетчика) обеспечивает дополнительную гибкость.

Особенности периферийных устройств и ввода-вывода:

- каждый вывод можно запрограммировать как вход или выход;
- каждый вход может быть запрограммирован на ТТЛ или КМОП уровни;
- к каждому выводу можно подключить внутреннюю нагрузку (около 20 кОм на вывод питания VDD);
- для портов B и C можно выбрать входные триггеры Шмитта;
- все выходы обеспечивают втекающий/вытекающий ток до 30 мА.

Архитектура

Микроконтроллеры SX имеет гарвардскую архитектуру, т.е. программный код и память данных размещены в разных областях памяти и обращение к ним обеспечивается независимыми шинами.

Расширенный конвейер *выборка – декодирование – выполнение – запись* обеспечивает выполнение команды за один такт. В конвейере микроконтроллеров команды обрабатываются за четыре стадии. На первой стадии команда выбирается из памяти. На второй стадии команда декодируется, в то время как следующая команда выбирается из памяти. На третьей стадии команда выполняется, а на четвёртой её результат записывается в память. Если конвейер загружен полностью, то все команды выполняются за один такт. Это обеспечивает цикл выполнения команды 20 нс при тактовой частоте 50 МГц.

Flash-память программ микроконтроллеров SX имеет информационную емкость 2048×12 бит. Память данных включает 136 байт статической памяти и регистры специальных функций. Статическая память

допускает прямую и косвенную адресацию. Все регистры специальных функций (включая регистр W) отображены в память данных.

Восьмибитное АЛУ выполняет арифметические и логические операции. Регистр W является рабочим регистром АЛУ. Обычно в нём хранится один из операндов в команде с двумя операндами. В зависимости от выполняемой команды АЛУ может изменять флажки переноса (C), нуля (O) и переноса цифры (DC).

Микроконтроллеры SX имеют специальные возможности, которые обеспечивают снижение стоимости системы и пониженное энергопотребление. Таймер сброса по подаче питания и сброса устройства устраняют необходимость применения внешней схемы сброса. Имеется возможность выбора одной из пяти конфигураций тактового генератора, в том числе программируемого внутреннего генератора 4 МГц. Энергосберегающий режим SLEEP, сторожевой таймер и возможность защиты кода уменьшают стоимость системы и улучшают её целостность.

Программирование и отладка

Микроконтроллеры SX удобны для разработки программ. Поскольку для программ используется Flash-память, нет необходимости в применении ультрафиолетовых устройств стирания. Проблемы, которые могут возникать при стандартной эмуляции с использованием отдельного внешнего контроллера, в данном случае отсутствуют.

Микроконтроллеры SX имеют все встроенные средства, необходимые для эмуляции непосредственно на плате, что обеспечивает надёжные и дешёвые решения для эмуляции встроенных приложений. Микроконтроллеры обеспечиваются интегрированной средой обработки SX-Key, включающей редактор, макроассемблер, отладчик и программатор. Система SX-Key поставляется фирмой Parallax Inc., которая является ведущим разработчиком инструментальных средств программирования для семейства 8-разрядных микроконтроллеров PIC фирмы Microchip.

Приложения

Архитектура микроконтроллеров SX подходит для многих известных приложений, таких как контроллеры процессов, модули удалённой телеметрии, модули охраны/мониторирования, однако их быстродействие открывает целый новый мир возможностей. При производительности до 50 миллионов операций в секунду аппаратные периферийные устройства могут быть заменены на программное обеспечение. Эти модули программного обеспечения называются *виртуальными периферийными устройствами*. Они обеспечивают уменьшение чи-

сла компонентов, время разработки, увеличивают гибкость проектов и, в конечном счете, стоимость системы.

Виртуальные периферийные модули:

- интерфейсы: последовательный, параллельный, I^2C, Microwire, (μ-Wire), Dallas μ-Wire, SPI, DMX-512, X-10, IR;
- генерация и измерение частоты;
- многозадачный режим, прерывания и работа с сетями;
- контроллеры динамической памяти;
- синтез звука;
- фазо-импульсная и широтно-импульсная модуляция;
- дельта-сигма АЦП;
- спектральный анализ;
- передача/распознавание тонового набора (DTMF) в телефонии;
- 300/1200 bps модем;
- квадратурный кодер/декодер;
- видеоконтроллер.

Экономичный режим

Микроконтроллеры SX имеют экономичный режим, который инициализируется с помощью команды SLEEP. Вывести микроконтроллер из этого состояния могут следующие события: сигнал переполнения сторожевого таймера (WDT), соответствующий переход внешнего сигнала на одном из выводов с функцией многовыводного пробуждения или внешний сигнал сброса на выводе MCLR. Функция многовыводного пробуждения доступна на выводах порта B. В экономичном режиме может работать только сторожевой таймер. Если при переходе в экономичный режим сторожевой таймер включен, то после выполнения команды SLEEP он сбрасывается в ноль, бит T0 регистра состояния устанавливается в 1, а бит PD сбрасывается в 0. Для достижения максимально низкой мощности потребления в экономичном режиме сторожевой таймер должен быть выключен, пробуждение микроконтроллера при этом должно осуществляться одним из двух альтернативных методов.

Поддержка прерываний

Микроконтроллеры SX поддерживают как внутренние, так и внешние прерывания. До восьми независимых внешних прерываний доступно через функцию многовыводного пробуждения без перевода микроконтроллера в экономичный режим. Внутреннее прерывание может быть сгенерировано при переполнении таймера/счетчика. Все

прерывания имеют одинаковый приоритет. Поступившие прерывания обрабатываются последовательно. После начала обработки одного прерывания другие прерывания блокируются до возврата его из обработки. Счётчик команд (PC) помещается в стек прерывания; состояния регистров FSR, состояния (STATUS) и W также сохраняются. Вектор подпрограммы обработки прерываний имеет адрес 0. При выполнении подпрограммы обслуживания прерываний опрашиваются регистры WKPND_B, RTCC (таймер/счетчик) и STATUS, и в результате анализа изменения их состояния выполняются соответствующие действия. При возвращении из процедуры обработки прерывания содержимое регистра счетчика команд и других сохранённых регистров восстанавливается. Микроконтроллеры имеют две команды выполнения возврата из прерывания: RETI и RETIW. Команда RETI восстанавливает значения вышеуказанных регистров из стека и специальных теневых регистров. Команда RETIW кроме действий, выполняемых командой RETI, модифицирует регистр RTCC, суммируя с ним двоичное дополнительное значение регистра W. Обе эти команды разблокируют прерывания от всех источников.

Организация памяти

Память программ. 2К слов памяти программы разделены на четыре страницы по 512 слов. Последовательность команд управляется через счетчик команд (PC), который автоматически увеличивается при выполнении линейных команд. Команды перехода могут требовать предустановки битов PA0 и PA1 регистра состояния. Поэтому для лёгкости программирования и увеличения производительности в систему команд была добавлена команда Page, выполняющая эту операцию; 8-уровневый стек из 11-битных слов обеспечивает работу до 8 вложенных подпрограмм.

Память данных. Сто тридцать шесть регистров памяти данных составлены из восьми банков по 16 регистров плюс восемь глобальных регистров. Все регистры могут быть адресованы как непосредственно, так и косвенно (с использованием регистра FSR). Регистры специальных функций отображаются в память данных, так что они могут быть считаны и записаны, как все остальные регистры.

Программирование памяти программ

Микроконтроллер SX был разработан с обеспечением возможности программирования на плате. Иными словами, после того как микроконтроллер установлен на печатной плате устройства, его мож-

но программировать непосредственно в устройстве, используя интерфейс с 4 выводами и средства разработки SX-Key. В большинстве случаев нет необходимости в иных добавлениях к устройству, кроме 4-контактного разъёма.

Помимо напряжения питания и общего провода только два вывода (OSC1 и OSC2) используются при программировании. Вывод OSC1 используется для подачи напряжения программирования V_{PP} = 12.5 В, вывод OSC2 – для последовательной передачи данных программирования памяти программ и чтения/проверки записанных данных. Для предотвращения доступа к содержимому памяти программ после завершения разработки служит бит защиты CP в регистре FUSE.

Компаратор

Выводы порта B (RB0…RB2) используются для подключения встроенного компаратора. Выход компаратора может быть доступен через вывод порта RB0 и через бит CMP_RES регистра CMP_B управления портом B. Биты выбора/управления этого регистра очищаются при сбросе, отключая компаратор. В приложениях, требующих пониженного потребления по питанию, необходимо отключить компаратор до перевода микроконтроллера в экономичный режим. Для правильного функционирования порты RB0…RB2 должны быть правильно запрограммированы: RB0 – как выход, а RB1 и RB2 – как входы.

Процедура аппаратного сброса

Сброс микроконтроллеров может быть инициирован подачей напряжения питания, сигналом на выводе MCLR и схемой контроля напряжения питания. Вектор сброса находится вверху памяти программ и имеет адрес 7FFH для памяти объёмом 2К слов, 3FFH – для памяти объёмом 1К слов и IFFH – для памяти 512 слов. Обычно при подаче напряжения питания достаточно встроенной схемы сброса. Однако если скорость напряжения питания недостаточна, может потребоваться внешняя RC-схема, подключаемая к выводу MCLR.

Схема сброса при понижении напряжения питания

Схема сброса при понижении напряжения питания (BROWN OUT) инициирует сброс SX-микроконтроллеров, когда напряжение питания падает ниже определённого значения, но не уменьшается до нуля, а затем восстанавливается. Порог срабатывания схемы составляет около 4.2 В. Схема может быть отключена установкой битов BOR0, BOR1 регистра FUSEX в 1.

5.3. Микроконтроллеры AT89 фирмы Atmel

Микроконтроллеры серии AT89 с системой команд и архитектурой MCS51 выпускаются фирмой Atmel Corporation (Atmel) с 1984 г. Отличительной особенностью этих микроконтроллеров является применение Flash-памяти программ. Эта особенность позволяет практически мгновенно изменять программный код микроконтроллера, что существенно сокращает цикл разработки. Микроконтроллеры в корпусе с 40/44 выводами полностью совместимы по выводам с контроллерами 80C51 и обеспечивают возможность использования наработанных программ и прямой замены микросхем. Flash-память программ делает также возможным дистанционное изменение программного кода встроенных микроконтроллеров непосредственно у заказчика.

5.3.1. Основные особенности микроконтроллеров

Основные особенности микроконтроллеров AT89:

- 8-разрядный процессор, оптимизированный для приложений управления;
- обширные возможности побитовой обработки;
- встроенная Flash-память программ;
- встроенная оперативная память;
- двунаправленные и индивидуально адресуемые линии ввода-вывода;
- один или несколько 16-разрядных таймеров/счетчиков;
- полнодуплексный UART;
- разветвленная структура прерываний;
- встроенный тактовый генератор;
- экономичные режимы: IDLE и POWER DOWN;
- встроенная память EEPROM (AT89S);
- последовательный интерфейс SPI (AT89S);
- сторожевой таймер (AT89S).

Рассмотрим особенности архитектуры микроконтроллеров.

Организация памяти

Микроконтроллеры имеют отдельные адресные пространства для памяти программы и данных. Это позволяет обращаться к памяти данных 8-битными адресами, чем обеспечивается быстрота операций, выполняемых с памятью 8-разрядным процессором. Вместе с тем с помощью регистра DPTR может быть сгенерирован 16-битный адрес данных. Таким образом, может быть адресовано до 64 Кбайт внешней памяти, для которой контроллер генерирует сигналы чтения и записи $\overline{RD}$ и $\overline{WR}$. Для памяти программы обеспечивается только чтение.

Непосредственно адресуется до 64 Кбайт памяти программ. Для чтения внешней памяти программ контроллер генерирует сигнал $\overline{\text{PSEN}}$. Внешние память программ и память данных могут быть объединены посредством логического И сигналов контроллера $\overline{\text{RD}}$ и $\overline{\text{PSEN}}$.

После выполнения процедуры сброса выполнение программы начинается с адреса 0000H. С адреса 0003H располагаются блоки обработки прерываний, занимающие по 8 байтов. Если процедура обработки прерывания занимает не более 8 байтов, она может располагаться в этом блоке. Процедуры обработки прерываний большего размера размещаются в других областях памяти программы, а управление передаётся им из блоков обработки прерываний командами безусловного перехода. Если прерывания в программе не используются, адреса, зарезервированные под блоки обработки прерываний, могут быть заняты кодом программы. Нижние адреса памяти программы могут адресовать как встроенную Flash-память программы, так и внешнюю память, в зависимости от того, соединён ли вывод управления внешним доступом EA с цепью питания или с общим проводом соответственно. Внешняя память программы может быть адресована 8-разрядным адресом с использованием порта ввода-вывода P0 для организации мультиплексированной шины адреса/данных или 16-разрядным адресом с использованием портов ввода-вывода P0 и P2, причём последний передаёт старший байт адреса.

Внешняя память данных может иметь объём до 64 Кбайт. Адресуется она так же, как и память программ 8- или 16-разрядными адресами. Пространство памяти данных разделено на 3 блока: нижние 128 ячеек, верхние 128 ячеек и область регистров специальных функций (Special Function Registers - SFR). Внутренняя память данных всегда адресуется одним байтом, что соответствует максимальному объёму памяти 256 байт. Однако применение различных способов адресации позволяет использовать до 384 байт внутренней памяти: при прямой и косвенной адресации пространства свыше 7FH адресуются различные области памяти (верхние 128 ячеек и область SFR).

В нижних 128 ячейках внутренней памяти первые 32 ячейки заняты четырьмя банками по 8 регистров, адресуемых командами программы как R0…R7. Выбор банка, в котором адресуются регистры, обеспечивается соответствующей установкой двух битов в слове состояния программы (PSW). Такая архитектура позволяет более эффективно использовать кодовое пространство, так как команды обращения к регистрам короче команд прямой адресации памяти. Следующие 16 ячеек после банков регистров образуют блок битовой адресации. Система команд микроконтроллера включает широкий выбор команд битовой адресации, которые могут непосредственно адресовать 128 бит в этой области. Адресуемые биты имеют адреса 00H…7FH.

Все байты в нижних 128 ячейках могут адресоваться прямым и косвенным методом адресации. К верхним 128 ячейкам, доступным лишь в контроллерах с объёмом памяти 256 байт, можно обращаться только с помощью косвенной адресации. Пространство SFR (регистров специальных функций) включает порты ввода-вывода, регистры таймеров, регистры управления периферийными устройствами и т.д. Эта область может адресоваться только прямой адресацией. Структура пространства идентична структуре аналогичного пространства контроллеров семейства MCS51, однако имеются дополнительные регистры. Шестнадцать адресов в пространстве SFR может адресоваться и побайтно и поразрядно. Адреса поразрядно адресуемых регистров заканчиваются тремя нулями. Адреса битов этих регистров имеют значения 80H…FFH.

Система команд

Система команд микроконтроллеров оптимизирована для 8-разрядных приложений управления и обеспечивает ряд быстрых способов адресации для доступа к внутренней оперативной памяти. Система команд обеспечивает обширную поддержку для однобитовых переменных как отдельного типа данных, позволяя выполнять прямое разрядное манипулирование в управлении и логических системах, которые требуют булевой обработки.

Слово состояния программы (PSW) содержит биты состояния, которые отражают текущее состояние процессора и размещается в пространстве SFR. Слово состояния программы содержит бит переноса CY, бит дополнительного переноса (для операций с ДДК – двоично-десятичным кодированием) AC, биты выбора банка RS0 и RS1, флаг переполнения OV, бит контроля по четности P и два определяемых пользователем флажка состояния.

Микроконтроллеры имеют команды прямой и косвенной адресации, регистровые команды и специальные команды для некоторых регистров. В последнем случае код команды непосредственно указывает на регистр, с которым будет производиться операция. При прямой адресации операнд определён 8-разрядным полем адреса в команде. Этим методом могут быть адресованы только внутренняя оперативная память и пространство SFR. При косвенной адресации в команде указан регистр, который содержит адрес операнда. Таким методом может адресоваться как внутренняя, так и внешняя оперативная память. В качестве регистра адреса для 8-разрядных адресов может быть или указатель вершины стека, или регистры R0 или R1 выбранного банка. Регистром адреса для 16-разрядных адресов может быть только 16-разрядный регистр указателя данных DPTR. К банкам регистров, которые содержат регистры R0…R7, можно обращаться командами,

чьи коды операции включают 3-разрядную спецификацию регистра. Команды, которые обращаются к регистрам этим способом, обеспечивают эффективное использование кода программы, так как при этом в команде отсутствует байт адреса. Банк, в котором текущей командой адресуется регистр, выбирается соответствующей установкой двух битов в слове состояния программы PSW.

Значение константы может следовать за кодом операции в памяти программ. К ПП можно обращаться только через индексную адресацию, 16-разрядный базовый регистр (DPTR или счетчик команд PC) указывает на начало текущей команды.

Машинный цикл микроконтроллеров содержит 6 состояний S1-S6, каждое из которых занимает два такта тактового генератора. Таким образом, длительность машинного цикла составляет 12 тактов тактового генератора, или при тактовой частоте 12 МГц – 1 мкс. Команда программы может быть выполнена в течение одного или нескольких машинных циклов, например команда MOVX занимает два машинных цикла.

Система прерываний

Микроконтроллеры имеют 5 источников прерываний: 2 внешних прерывания, 2 прерывания по таймеру и прерывание от последовательного порта. В некоторых контроллерах имеются дополнительные источники прерывания в соответствии с особенностями их архитектуры. Прерывание по каждому из источников может быть индивидуально разрешено либо запрещено путём установки или сброса соответствующих битов в регистре разрешения прерываний IE, расположенном в пространстве SFR. В процессе выполнения программы состояние флагов прерываний считываются в пятом состоянии машинного цикла и опрашиваются в следующем цикле. Для каждого из источников прерываний может быть запрограммирован один из двух уровней приоритета путём установки или сброса соответствующего бита в регистре приоритетов прерываний IP, расположенном в пространстве SFR. Низкоприоритетное прерывание может быть прервано высокоприоритетным прерыванием, но не другим низкоприоритетным прерыванием. Выполнение процедуры высокоприоритетного прерывания не может быть прервано никаким прерыванием. Если одновременно поступило два запроса на прерывание с разными уровнями приоритета, то сначала выполняется процедура высокоприоритетного прерывания. При поступлении запросов на прерывание с одинаковым уровнем приоритета порядок выполнения процедур обработки прерывания определяется внутренней последовательностью опроса. В процессе обработки прерывания аппаратно сгенерированная процедура LCALL помещает содержимое счетчика команд PC в стек и загружает

начальным адресом соответствующего блока обработки прерывания. Кроме счетчика команд автоматически в стеке не сохраняются никакие другие регистры. За сохранение других необходимых регистров отвечает программист. В ряде случаев это позволяет сократить время обработки прерывания. В результате множество функций обработки прерываний, которые являются типичными в приложениях управления: переключение вывода порта, перезагрузка таймера или чтение буфера последовательного порта, – могут быть завершены скорее, чем это было бы возможно при другой архитектуре. Многие приложения требуют более двух уровней приоритетности прерываний, обеспечиваемых аппаратными средствами микроконтроллеров. В таком случае возможно применение простого программного кода, с помощью которого эмулируется третий уровень приоритетности прерывания.

Последовательный порт

Последовательный порт микроконтроллеров – полнодуплексный, с буфером приёмника-передатчика. Доступ к регистрам приёма и передачи осуществляется через регистр SBUF в пространстве SFR. При выполнении записи в этот регистр загружается регистр передачи, чтение обеспечивает доступ регистру приёма. При тактовой частоте микроконтроллера 12 МГц в зависимости от установленного режима последовательный порт обеспечивает скорость обмена до 1 Мбит/с. Однако для обеспечения стандартных скоростей обмена 1200…19200 бод необходимо тактировать микроконтроллер с частотой 11.059 МГц, при этом для получения необходимой скорости обмена используется один из таймеров.

Экономичные режимы работы

Для обеспечения экономии потребления энергии микроконтроллеры имеют два программно управляемых режима работы с пониженной мощностью. В режиме IDLE (холостой ход) процессор выключен, в то время как оперативная память и встроенные периферийные устройства продолжают функционировать. В режиме POWER DOWN все устройства микроконтроллера выключены, однако данные в оперативной памяти продолжают сохраняться.

Микроконтроллеры имеют два режима работы с пониженным потреблением по питанию: Idle и Power Down, переход в которые обеспечивается установкой соответствующих битов в регистре PCON пространства SFR.

В режиме Idle (IDL = 1) тактовый генератор продолжает работать, и обеспечивается работа периферийных устройств: таймеров, блока

обработки прерываний и последовательного порта, при этом процессор микроконтроллера останавливается в ожидании поступления прерывания. В этом режиме потребление тока уменьшается приблизительно на 15 % от потребления полностью активного устройства.

В режиме Power Down (PD = 1) останавливается тактовый генератор микроконтроллера, однако содержимое встроенной памяти и регистров пространства SFR сохраняется. В этом режиме типовое потребление микроконтроллера составляет менее 15 нА, и в любом другом случае – не более 600 нА. Выход из состояния Power Down возможен только при выполнении аппаратного сброса. При сбросе переинициализируются все регистры пространства SFR, однако содержимое внутренней памяти данных не изменяется.

Микроконтроллеры разработаны с применением статической логики, которая не требует непрерывной синхронизации. Поэтому частота тактового генератора может быть уменьшена или же он может быть остановлен в ожидании события, требующего обработки. Это также способствует снижению потребления по питанию.

Аппаратный сброс запускает также тактовый генератор микроконтроллера. На время пребывания микроконтроллера в режиме Power Down напряжение питания может быть снижено до 2 В. Однако напряжение питания не должно быть понижено до того, как микроконтроллер перешел в режим Power Down, и должно быть восстановлено перед выполнением процедуры сброса. Сигнал сброса должен быть достаточно длительным для того, чтобы стабилизировалась работа тактового генератора (обычно не более 10 мс).

5.4. Микроконтроллеры AVR фирмы Atmel

5.4.1. Общая характеристика AVR микроконтроллеров

Микроконтроллеры AVR серии AT90 с RISC-архитектурой выпускаются фирмой Atmel Corporation (Atmel) с 1997 г. Архитектура была разработана с использованием достижений полупроводниковой микроэлектроники и возможностей программного обеспечения 1990-х годов. Созданные в результате микроконтроллеры имеют самое высокое соотношение производительность/потребление энергии, доступное на рынке 8-разрядных микроконтроллеров.

Языки программирования высокого уровня быстро становятся стандартным методом программирования для встроенных микроконтроллеров из-за уменьшения времени разработки и упрощенной поддержки сопровождения. В разработке AVR-архитектуры принимали участие эксперты языка Си, чтобы гарантировать, что аппаратные средства и программное обеспечение позволят получать высокоэффективный код.

Для того чтобы оптимизировать размер кода, эффективность и потребляемую мощность, AVR-архитектура включает большой блок регистров с быстрым доступом и быстрые команды с однотактным циклом.

На рис. 5.1 показана логическая организация микроконтроллера Atmel AVR AT90S2333/4433. Блок регистров с быстрым доступом состоит из 32 универсальных рабочих регистров. Традиционные АЛУ базировались на архитектуре, которая требовала большого количества команд для передачи данных между АЛУ и памятью. AVR-архитектура с большим количеством рабочих регистров не нуждается в таких командах.

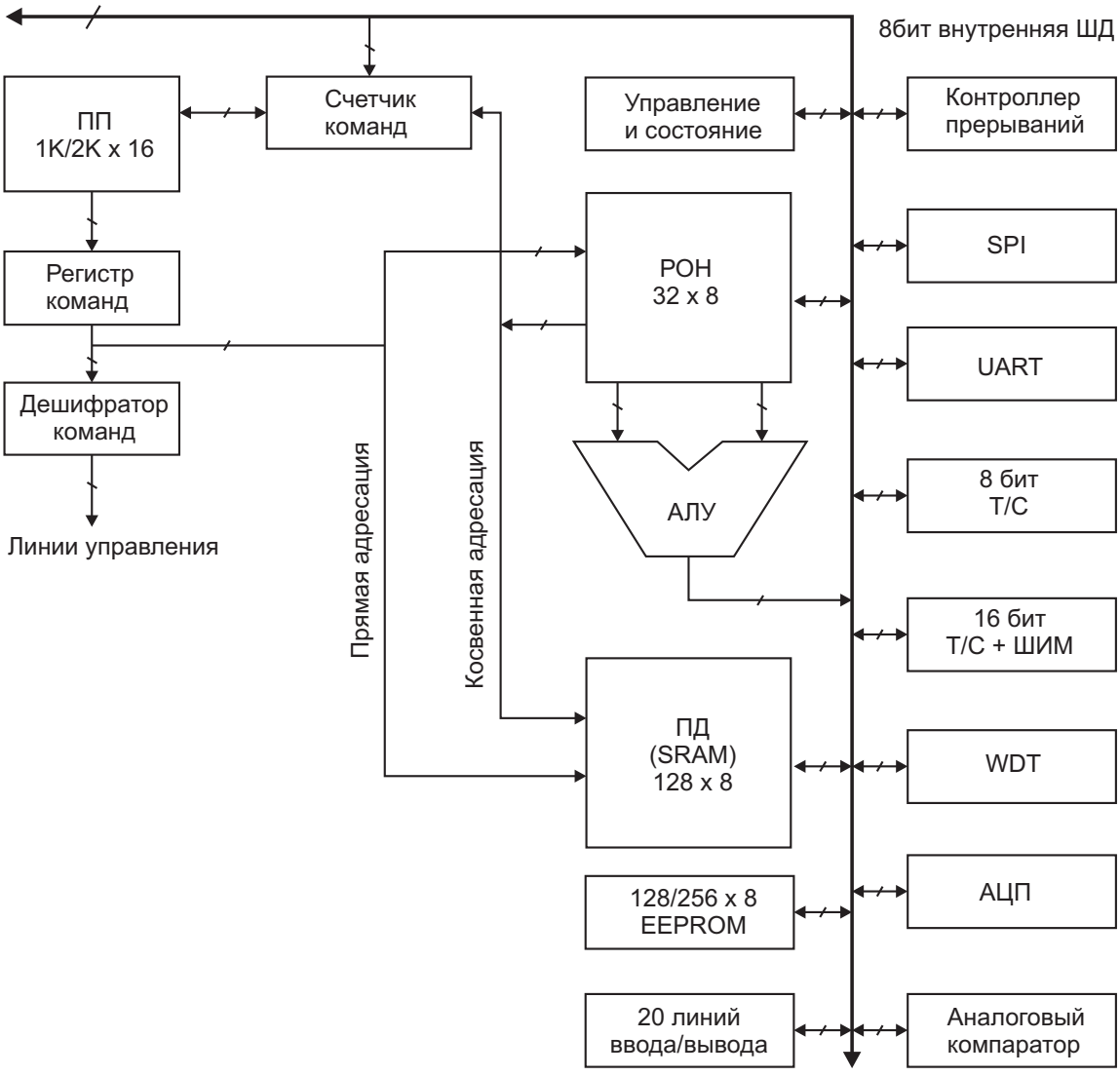


Рис. 5.1. Логическая организация микроконтроллера AVR AT90S2333/4433

AVR-контроллеры имеют двухступенчатый конвейер команд, благодаря чему за один такт обеспечивается выполнение одной команды и выборка из памяти следующей команды. В других CISC- и RISC-

подобных архитектурах цикл выполнения команды разделён на фазы (до 12 фаз). AVR-микроконтроллеры выполняют команды за один такт тактового генератора, и это первые действительно RISC-процессоры на 8-разрядном рынке.

AVR-архитектура поддерживает полный спектр контроллеров: от простых контроллеров с малым числом выводов до устройств более высокого класса с большим объемом встроенной памяти. Архитектура в стиле гарвардской непосредственно адресует до 8 Мбайт памяти программ и 8 Мбайт памяти данных. Универсальные регистры контроллеров отображаются в адресном пространстве памяти данных, что обеспечивает возможность быстрого переключения.

Семейство AVR-микроконтроллеров производится по КМОП-технологии (с низким энергопотреблением) фирмы Atmel. Перепрограммируемая энергонезависимая Flash-память обеспечивает перепрограммирование контроллеров как с помощью обычного программатора, так и на плате через последовательный интерфейс SPI. Объединяя расширенную RISC-архитектуру с загружаемой Flash-памятью на одном кристалле, AVR-семейство предлагает мощное решение для встроенных систем управления.

Старшие модели микроконтроллеров обеспечивают широтно-импульсную модуляцию (ШИМ, PWM) с использованием 16-разрядного таймера/счетчика (деление тактовой частоты на 510, 1022 и 2046). Универсальный асинхронный приемопередатчик (UART) обеспечивает полнодуплексный обмен данными с частотой до 115200 бит/с. Последовательный трехпроводный интерфейс SPI обеспечивает синхронный обмен данными с периферийными устройствами или другими микроконтроллерами со скоростью до 5 Мбит/с.

5.4.2. Микроконтроллер AT90S1200

Рассмотрим более подробно младшую модель семейства – контроллер AT90S1200. Вот основные его характеристики:

- 89 команд, выполняемых за один такт;
- 32 восьмибитных универсальных регистра;
- рабочая частота 0 Гц…16 МГц (минимальный цикл выполнения команды 62.5 нс);
- 16-битные команды;
- 8-битные данные;
- 1 Кбайт встроенной перепрограммируемой Flash-памяти программ с ресурсом 1000 циклов записи/стирания;
- возможность перепрограммирования на плате через последовательный интерфейс SPI;

- 64 байта электрически перепрограммируемой памяти EEPROM с ресурсом 100 тыс. циклов записи/стирания;
- 15 индивидуально программируемых линий ввода-вывода;
- максимальный втекающий ток 20 мА, максимальный вытекающий ток 3 мА;
- встроенный аналоговый компаратор;
- 8-разрядный таймер/счетчик с 10-разрядным программируемым предварительным делителем;
- автоматический сброс при подаче напряжения питания;
- таймер включения при подаче напряжения питания;
- сторожевой (Watch dog) таймер с собственным встроенным генератором частотой 1 МГц;
- наличие битов секретности для защиты кода;
- биты идентификации;
- экономичные режимы: IDLE и POWER DOWN;
- энергонезависимая, экономичная, быстрая КМОП-технология;
- полностью статическая архитектура;
- широкий диапазон рабочего напряжения питания и температуры: 2.7…6.0 В; −55…+125 °С;
- низкое энергопотребление: 2 мА (типично для 3 В, 4 МГц); 500 мкА (типично для 3 В, 4 МГц в режим Idle); 15 мкА (максимально для 3 В в режиме Power Down при включенном сторожевом таймере); 1 мкА (максимально для 3 В в режиме Power Down при отключенном сторожевом таймере).

AVR-ядро микроконтроллера объединяет систему команд с 32 универсальными рабочими регистрами. Все 32 регистра непосредственно соединены с арифметико-логическим устройством (АЛУ), благодаря чему в одной команде обеспечивается доступ к двум регистрам. В результате архитектура обеспечивает производительность, до десяти раз превышающую производительность стандартных CISC-микроконтроллеров. Архитектура поддерживает языки высоких уровней так же эффективно, как и оптимизированные программы на ассемблере.

Режим Idle останавливает АЛУ, обеспечивая при этом работу регистров, таймера/счетчика, сторожевого таймера и системы прерываний, для продолжения функционирования. Режим Power Down сохраняет содержимое регистров, останавливая тактовый генератор и отключая все другие функции контроллера до следующего внешнего прерывания или аппаратного сброса.

АЛУ выполняет арифметические и логические операции между регистрами или между константой и регистром. Операции с одиночными регистрами также выполняются в АЛУ. AVR использует гарвардскую архитектуру с разделенными шинами данных и программ. Обращение к памяти программ конвейеризовано. В то время как одна команда вы-

полняется, следующая команда извлекается из памяти программ. Эта концепция обеспечивает возможность выполнения команд в каждом такте.

Команды с относительной адресацией определены в адресном пространстве из 512 команд. Все AVR-команды имеют формат одиночного 16-разрядного слова, т.е. каждый адрес памяти программы содержит одиночную 16-разрядную команду. На время обработки прерываний и вызовов подпрограмм значение счетчика адреса (PC) сохраняется в специализированном трехуровневом аппаратном стеке.

Пространство памяти ввода-вывода содержит 64 адреса для функций периферийных устройств, таких как регистры управления, таймер/счетчик, и других функций ввода-вывода.

Гибкий модуль прерывания имеет управляющие регистры в пространстве ввода-вывода и дополнительный общий бит разрешения прерываний в регистре состояния. Все прерывания имеют отдельные векторы в таблице векторов прерываний, размещаемой в начале памяти программы. Прерывания имеют приоритет в соответствии с позицией их вектора прерывания. Прерывание с меньшим адресом вектора имеет более высокий приоритет.

Все команды обращения к регистрам имеют прямой доступ ко всем регистрам за один такт. Исключение составляют пять арифметических и логических команд SBCI, SUBI, CPI, ANDI, ORI, одним из операндов которых является константа, и команды LDI загрузки константы. Эти команды обращаются ко второй половине регистров – R16…R31. Регистр R30 также служит 8-разрядным указателем для косвенной адресации регистров.

Для того чтобы включить энергосберегающие режимы, необходимо установить в 1 бит SE регистра MCUCR в пространстве ввода-вывода и выполнить команду SLEEP. Если в то время, когда микроконтроллер находится в энергосберегающем режиме, происходит разрешенное прерывание, микроконтроллер выполняет подпрограмму обработки прерывания, а затем переходит к выполнению команды, следующей за командой SLEEP. Содержимое универсальных регистров и регистров ввода-вывода при этом остается неизменным. Если во время нахождения микроконтроллера в энергосберегающем режиме происходит внешний сброс, то микроконтроллер выполняет процедуру сброса.

Если бит SM регистра MCUCR сброшен в 0, то команда SLEEP приводит к переходу в режим IDLE, в котором останавливается процессор микроконтроллера. При этом таймер/счетчики, сторожевой таймер и система прерывания продолжают функционировать. Это позволяет продолжить работу микроконтроллера при наступлении внешних и внутренних прерываний, например прерывания от переполнения таймера или сброса сторожевого таймера.

Если бит SM регистра MCUCR установлен в 1, то команда SLEEP приводит к переходу в режим POWER DOWN, при котором останавливается внешний тактовый генератор. Если сторожевой таймер был включен, то он по окончании заданного периода запустит контроллер. Если сторожевой таймер заблокирован, то только внешний сброс или внешнее прерывание может запустить контроллер.

8-разрядный счетчик/таймер микроконтроллера снабжен 10-разрядным предварительным делителем частоты. Таймер может работать как от внутреннего генератора или предварительного делителя, так и от внешнего источника тактовой частоты. С предварительного делителя на таймер может подаваться тактовая частота микроконтроллера, деленная на 8, 64, 256 или 1024.

Сторожевой таймер тактируется от отдельного встроенного генератора с тактовой частотой около 1 МГц при напряжении питания 5 В и имеет предварительный делитель частоты. С помощью предварительного делителя интервал сброса сторожевого таймера может быть установлен от 16 до 2048 тактов генератора. Специальная команда WDR сбрасывает сторожевой таймер. Если сторожевой таймер сбрасывается не командой WDR, а по истечении временного интервала, то микропроцессором выполняется процедура сброса. Адреса электрически перепрограммируемой памяти (EEPROM) располагаются в пространстве адресов ввода-вывода. Время записи в EEPROM составляет 2.5…4 мс в зависимости от напряжения питания. Функция автосинхронизации при этом позволяет программе обнаружить, когда может быть записан следующий байт. Схема слежения предотвращает запись в EEPROM при пониженном напряжении питания. После записи или чтения EEPROM АЛУ задерживает на два такта выполнение следующей команды.

5.5. Микроконтроллеры фирмы "Ангстрем"

Открытое акционерное общество (ОАО) "Ангстрем" (Зеленоград, г. Москва) разработало и с 1998 г. выпускает микроконтроллерное ядро ТЕСЕЙ, содержащее широкий набор функциональных модулей. Ядро ТЕСЕЙ предназначено для создания и массового производства большого семейства 8-разрядных микроконтроллеров.

Характерной особенностью ядра ТЕСЕЙ являются:

- гарвардская RISC-архитектура, позволяющая выполнять любую из 52 команд 16-разрядного формата за два такта частоты процессора;
- единая система команд для всего семейства с возможностью адресации до двух операндов, находящихся в памяти;
- 4-ступенчатый конвейер выполнения команд;

- малое время отклика на прерывание и сохранение контекста;
- широкий диапазон конфигураций внутренних памяти команд, памяти данных и периферийных устройств.

Микроконтроллеры семейства ТЕСЕЙ предназначены для использования в системах управления, работающих в режиме реального времени. Они отличаются высокой производительностью, наличием энергонезависимой памяти данных, возможностью многократного перепрограммирования памяти программ, небольшим количеством внешних выводов и низким энергопотреблением.

Микроконтроллерное ядро ТЕСЕЙ включает в себя комплекс программных и аппаратных средств для автоматизации проектирования, отладки и аттестации программ микроконтроллеров. Комплекс состоит из компилятора ассемблера микроконтроллеров ТЕСЕЙ – TESSA 0.1, пакета программ отладочной среды микроконтроллеров ТЕСЕЙ и аппаратного эмулятора микроконтроллеров ТЕСЕЙ.

На базе ядра ТЕСЕЙ уже создано и выпускается несколько серийных микроконтроллеров универсального и специализированного применения. В качестве примера назовем серийные изделия: КР1878ВЕ1 — универсальный управляющий микроконтроллер, КР1878ВЕ2 – специализированная БИС поведенческих развивающих игр и КБ5004ВЕ1 – специализированная БИС банковской интеллектуальной платёжной карты с многоуровневой системой защиты.

5.5.1. Микроконтроллер КР1878ВЕ1

Микроконтроллер КР1878ВЕ1 разработан на основе отечественного микроконтроллерного ядра ТЕСЕЙ, предназначенного для построения 8-разрядных RISC-микроконтроллеров реального времени. Данный микроконтроллер ориентирован на использование в системах управления реального времени. Он отличается малым количеством внешних выводов, низким током потребления, высокой производительностью, наличием энергонезависимой памяти данных и возможностью многократного перепрограммирования памяти команд. При необходимости и при достаточном объёме партии микроконтроллер может поставляться в варианте с масочным ПЗУ программ.

Основные характеристики микроконтроллера КР1878ВЕ1:

- электрически стираемое ППЗУ команд — 1К×16 бит;
- статическое ОЗУ данных — 128×8 бит;
- EEPROM данных 64×8 бит;
- система команд из 52 команд;
- тактовая частота от 32 кГц до 8 МГц;
- время выполнения любой команды – 2 такта (250 нс при частоте 8 МГц);

- семь уровней прерываний (начальный пуск, системная ошибка, сторожевой таймер, порт A, порт B, таймер, конец записи в EEPROM);
- время реакции на прерывание – 3 такта;
- 12 (13) линий ввода/вывода с индивидуальным управлением направлением и прерыванием от любой линии;
- максимальный ток – 25 мА;
- 16-разрядный таймер с 8-разрядным делителем частоты;
- сторожевой таймер с автономным генератором;
- напряжение питания – Vcc = (4.0…6.0) В;
- ток потребления менее 2 мА при Vcc = 5 В и f = 5 МГц; 50 мкА при Vcc = 5 В и f = 32 кГц; менее 1 мкА в режиме STOP.

Микроконтроллер КР1878ВЕ1 содержит функционально законченные устройства, необходимые для локального управления широким кругом разнообразных бытовых и промышленных объектов, в том числе центральный процессор, Flash-память программ, ОЗУ данных, EEPROM данных, сторожевой таймер, 2 порта ввода/вывода и таймер общего назначения. Производительность микроконтроллера до 4 mips на тактовой частоте 8 МГц. Обмен данными между центральным процессором, ОЗУ данных и периферийными устройствами производится по единой шине (рис. 5.2).

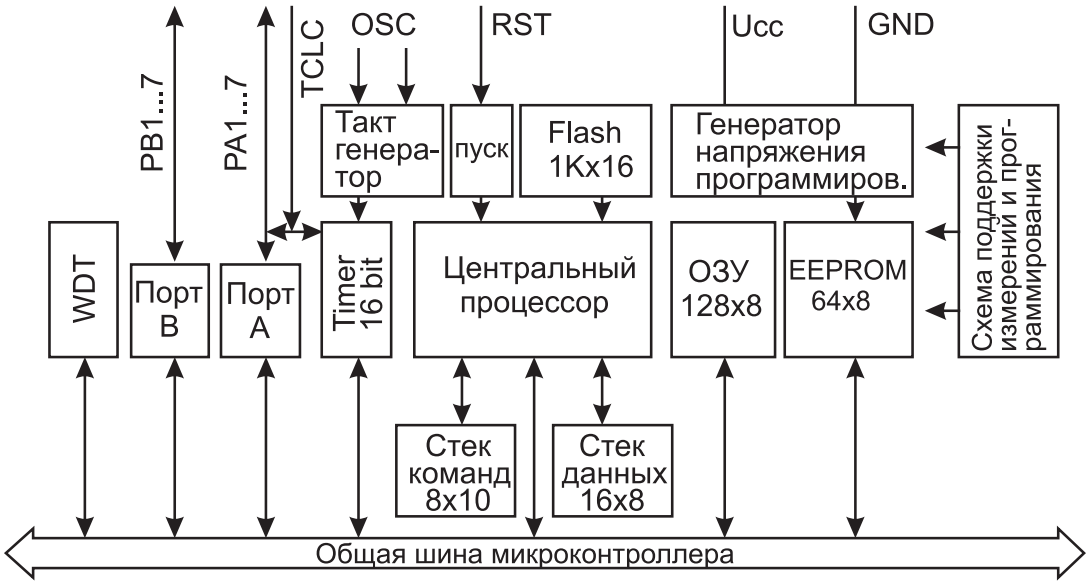


Рис. 5.2. Функциональная блок-схема микроконтроллера КР1878ВЕ1

Микроконтроллер изготовлен по КМОП-технологии и выпускается в 18-выводном пластмассовом корпусе ДИП типа 2104.18-8. По цоколевке КР1878ВЕ1 совместим с подобными микроконтроллерами фирм Microchip, Zilog и др. Это даёт возможность отечественным потребителям, перепрограммировав реализацию своих алгоритмов, за-

менить ранее применённые импортные микроконтроллеры на отечественные с лучшими, в ряде случаев, техническими характеристиками.

Применение микроконтроллера КР1878ВЕ1 облегчается его обеспеченностью программными и аппаратными средствами автоматизации программирования и отладки, включающими как кросс-систему на основе персонального компьютера, так и резидентную систему на основе аппаратного эмулятора.

5.6. Программируемые микроконтроллеры фирмы Zilog

Программируемые микроконтроллеры широкого применения Programmable Consumer Controller Processors (PCCP) типа Z8 выпускаются фирмой Zilog. В составе обширной номенклатуры фирмы Zilog на сегодняшний день всего 16 разновидностей приборов типа PCCP Z8. Все они начинаются с аббревиатуры Z86E, в которой последняя буква показывает наличие встроенной однократно программируемой памяти программ ППЗУ. Эти 16 контроллеров PCCP Z86E делятся на две группы: контроллеры общего назначения и контроллеры с расширенным набором функций.

5.6.1. Контроллеры общего назначения

Семейство универсальных микроконтроллеров фирмы Zilog характеризуется минимальными отступлениями от принципов, заложенных в основу архитектуры и системы команд микропроцессора Z80. К ядру микропроцессора Z80 была добавлена минимальная периферия (встроенные ППЗУ, ОЗУ (регистровый файл), генератор, порты ввода-вывода, таймеры, компараторы, сторожевой таймер (WDT), узел прерываний). Любой из приборов, входящих в состав семейства, представляет собой усеченную версию базового кристалла Z86E40, который содержит ресурсы, максимально возможные для микроконтроллеров этого класса. Усечение младших приборов семейства идет по разным направлениям, связанным с уменьшением их встроенных ресурсов и, прежде всего, объемов встроенных ППЗУ и регистрового файла, количества выводов корпуса, максимально возможной частоты внутреннего тактирования и стоимости.

Архитектура микроконтроллеров PCCP Z8 включает генератор тактовых импульсов, систему сброса контроллера, центральный процессор с АЛУ и регистром флагов состояния (РФС), регистровый файл, встроенную память программ, переключаемую специальным счетчиком команд, специальный механизм адресации внешних устройств памяти, который обеспечивает синхронизацию работы АЛУ с микросхемами внешней памяти (только для Z86E40), а также набор

периферийных устройств, состоящий из портов ввода-вывода, компараторов, функциональных таймеров, системы прерываний, сторожевого таймера WDT и средств обеспечения режимов пониженного энергопотребления, узла сброса прибора в начальное состояние.

Регистры. Микроконтроллеры РССР Z8 могут иметь три типа независимых друг от друга и никак не пересекающихся друг с другом областей памяти: регистровый файл, память программ, память данных. Регистровый файл состоит из регистров двух типов – регистров общего назначения (РОН) и регистров специального назначения (РСН). Объем регистрового файла составляет от 60 до 256 байт в зависимости от типа прибора. Регистры регистрового файла РССР Z8 могут быть доступны и как 8-битные байты, а для ряда операций – и как 16-разрядные слова. РОН являются универсальными регистрами, которые исполняют роль встроенного ОЗУ для контроллеров РССР Z8, не имеющих аппарата для подключения внешней памяти. Любой из РОН может быть использован в качестве аккумулятора, указателя адреса, регистра данных или индексного регистра.

РСН служат для обеспечения управления периферийными устройствами контроллеров РССР Z8: переключения портов ввода-вывода, выбора режимов их работы, установок системы прерывания, обеспечения режимов работы контроллера и встроенных таймеров, в том числе и WDT. Кроме того, к РСН РССР Z8 относятся регистры процессора, которые включают такие необходимые для работы центрального процессора прибора элементы памяти, как регистровый указатель, указатели стека, РФС и регистр счетчика команд.

Память программ. Память программ емкостью до 64 Кбайт может быть использована в составе микроконтроллерной системы, построенной на основе РССР Z8. Однако подобное достижимо лишь для контроллера типа Z86E40, имеющего аппаратный механизм для подключения внешней памяти. Остальные контроллеры семейства в качестве памяти программ могут использовать только внутреннюю встроенную память программ, объем которой лимитирован конкретным типом устройства. Таким образом, программная память может быть внутренней (встроенной) и внешней, но лишь для Z86E40.

Память программ предназначена для хранения кодов управляющей программы, подготовленной пользователем на этапе разработки программируемого устройства на базе контроллеров семейства РССР Z8. Помимо кода программы, в первых 12 ячейках программной памяти располагаются адреса каждого из 6 векторов прерывания любого из контроллеров. Поэтому при сбросе контроллера программный счетчик устанавливается на адрес 000CH, откуда и начинает свое действие собственно программа управления любым из контроллеров Z8.

Для внутреннего сегмента памяти программ любого из контроллеров семейства РССР Z8 возможна защита памяти программ, блокировка доступа к которой может осуществляться на этапе программирования контроллеров.

Память данных. Память данных объемом до 64 Кбайт имеет смысл только в отношении микроконтроллера типа Z86E40, который содержит механизм адресации внешней памяти. При этом разделение обращения к внешней памяти программ и к внешней памяти данных осуществляется сигналом DM. Для всех остальных контроллеров семейства РССР Z8, не имеющих механизма для подключения внешних ресурсов памяти, в качестве памяти программ выступает набор РОН. В частности, именно он используется для организации стека при разработке программного обеспечения для контроллеров, не содержащих аппарата подключения внешней памяти, что возможно благодаря программной установке вершины стека.

Система команд. Контроллеры семейства РССР Z8 имеют 50 типов инструкций, которые реализуют пересылки операндов и блоков данных, их арифметические и логические преобразования (в том числе сдвиги), а также двоично-десятичные преобразования. Для адресации операторов может быть использовано шесть способов адресации. Возможна работа с байтами, а для некоторых инструкций – возможна работа с 16-разрядными словами. Кроме того, имеется специальный набор управляющих команд, обеспечивающих работу с подпрограммами, ветвление программ, взаимодействие со стеком и РСН, перевод контроллера в режимы пониженного энергопотребления. Система команд Z8 не поддерживает команды умножения и деления операндов; реализация этих операций достигается только за счет использования специальных программ арифметики.

Гибкая классическая система команд в сочетании с классической Intel-архитектурой процессоров семейства РССР Z8 дает поразительные результаты даже по сравнению с современными RISC-контроллерами. Там, где супербыстрый RISC-процессор использует 4–5 инструкций, любому из Z8 достаточно одной, особенно это заметно при реализации процедур обработки данных.

Порты ввода-вывода. Основным средством обмена информационными сигналами и управляющими воздействиями между внешней средой и микроконтроллером являются линии или порты ввода-вывода. У РССР Z8 их количество определяется типом корпуса, в который упакована та или иная модификация семейства. Максимальное число линий ввода-вывода содержит кристалл Z86E40, размещаемый в корпусах с наибольшим числом выводов, – четыре порта P0, P1, P2, P3, каждый из которых содержит по 8 линий ввода-вывода. Порты P0 и P1 могут быть сконфигурированы как в качестве функциональных выво-

дов, так и для использования в качестве мультиплексированной шины адреса/данных при подключении к контроллеру внешней памяти. При этом порт P1 служит для передачи данных и генерации младшего байта адреса, а порт P0 – только для генерации старшего байта адреса.

Синхронизация процесса взаимодействия с внешним адресным пространством осуществляется четырьмя специальными управляющими сигналами, генерируемыми на выводах AS, DS, R/W, DM, которые имеет только кристалл Z86E40. Остальные порты P2 и P3 могут использоваться только в качестве функциональных. При этом номер каждого из портов, а также номер каждой из линий внутри порта жестко связаны с перечнем выполняемых им функций. Так, каждая из линий порта P2 может быть побитно сконфигурирована либо для ввода, либо для вывода информации, а в целом – для всех линий этого порта, работающих на вывод, выбирается один из двух возможных типов выходных цепей: обычный цифровой вывод (ТТЛ уровень) или открытый коллектор.

Младшие четыре линии порта P3 предназначены только для ввода информации, однако они могут быть определены либо для работы в аналоговой моде (использование встроенных компараторов), либо для работы в цифровой моде. Старшие четыре линии порта P3 предназначены только для вывода информации (причем только уровнем ТТЛ). Линии порта P0 так же, как и линии порта P3, разделены на младшую и старшую тетрады, каждая из которых может быть отдельно запрограммирована либо для работы по вводу (только уровнем ТТЛ), либо по выводу информации (уровнем ТТЛ или открытый коллектор для Z86E30/31/40). Все линии порта P1 настраиваются вместе или на прием, или на выдачу байта (только уровнем ТТЛ). Если любой из выводов любого из портов процессора сконфигурирован на ввод, то фиксация уровня внешнего сигнала на соответствующей ножке производится аппаратно сразу двумя независимыми схемами — регистром-защелкой и триггером Шмидта. Конфигурация каждого из портов задается программно посредством специальных регистров P01M для портов P0 и P1 и P1M, P2M, P3M для остальных портов соответственно, а ввод-вывод информации по каждой линии осуществляется через организацию записи/чтения регистров P0, P1, P2, P3 соответственно.

Если прибор Z86E40, размещенный в самом многовыводном корпусе, содержит линии каждого из портов, то кристаллы Z86E30 и Z86E31, упакованные в 28-выводные корпуса, включают лишь порты P1, P2 и P3, а контроллеры, размещаемые в 18-выводных корпусах, содержат порт P2 целиком, первые 3 линии порта P0 и линии P3.1, P3.2, P3.3 порта P3.

Каждый из микроконтроллеров РССР Z8 имеет в своем составе два встроенных компаратора, входы которых могут быть выбраны в качестве альтернативных функций для трех линий порта P3. При этом

исследуемый аналоговый сигнал принимается на выводы P3.1 и P3.2, а опорный сигнал, общий для каждого из компараторов, подводится к выводу P3.3. Каждый из выходов компараторов связан с индивидуальным прерыванием контроллера. Подобная конфигурация аналоговых входов, а также достаточно высокая чувствительность встроенных компараторов позволяет легко организовывать на их базе различные типы аналого-цифровых преобразователей. Однако повышенная чувствительность аналоговых входов порта P3 требует их дополнительной защиты от возможных помех. Такую защиту легко организовать с помощью двух диодов Шоттки, включенных в качестве ограничителей уровня импульсных сигналов относительно уровней земли и питания контроллера.

Для тактирования работы любого из Z8 служит узел встроенного тактового генератора. Параметры максимальной тактовой частоты для различных версий Z8 различны (от 8 до 16 МГц). Минимальное значение тактовой частоты также лимитировано на уровне 1 МГц. Для запуска встроенного генератора необходимо организовать относительно специальных выводов XTAL1 (вход) и XTAL2 одну из четырех возможных внешних цепей: либо это уже готовый внешний генератор заданной частоты, подключенный через КМОП-выход, либо высококачественный кварцевый резонатор, либо LC-цепь, либо RC-цепь.

Значение рабочей частоты контроллера также напрямую связано с его энергопотреблением. Достаточно сказать, что при минимальных значениях тактовой частоты потребление контроллера в 3–4 раза меньше потребления на максимальных значениях частоты тактирования.

Кроме того, потребление контроллеров РССР Z8, как впрочем для большинства статических микросхем, значительно зависит и от температурного режима работы прибора.

Режимы работы. Общее энергопотребление контроллера также определяется рабочим режимом, в котором он находится. Для РССР Z8 различают три основных режима работы. Это штатный рабочий режим, при котором все узлы контроллера функционируют в полном объеме (штатное потребление), и два режима пониженного потребления: режим HALT, при котором остановлено тактирование процессора, но все периферийные узлы поддерживают активное состояние ($\sim$50 % от штатного потребления), и режим STOP, при котором также останавливается тактирование, а все периферийные устройства переводятся в пассивное состояние ($\sim$7 % от штатного потребления). Столь малое потребление объясняется тем, что в режиме STOP прекращается работа тактового генератора микроконтроллера. Если же в режиме STOP отключить еще и WDT, то можно достигнуть величины $\sim$0.05 %

от штатного потребления. Выход из режима HALT осуществляется при срабатывании любого из прерываний, а из режима STOP – либо по сигналу сброса, который может быть активизирован, в том числе и WDT, либо при выполнении одного из условий, задаваемых заранее в регистре специального назначения SMR.

Счетчики-таймеры. В состав периферийных устройств любого из контроллеров семейства РССР Z8 входят 8-разрядные таймеры-счетчики, которые обеспечивают возможность работы процессора в реальном масштабе времени. Каждый из членов семейства содержит по два таких независимых узла T0 и T1, и лишь самый младший кристалл Z86E02 имеет один таймер-счетчик. На входе каждого из таймеров имеется программируемый 6-разрядный делитель. Тактирование таймеров-счетчиков может производиться как от тактового генератора микроконтроллера, так и от внешнего источника, подключенного через вывод P3.1. Оба счетчика могут работать только в режиме декремента. Содержимое каждого таймера-счетчика может быть не только заполнено предварительной уставкой, но и считано без нарушения или изменения режима его работы. Через специальные конфигурационные регистры устанавливается режим работы каждого из таймеров. Возможна установка одного из четырех следующих режимов: режим счета внешних событий, режим измерения интервала времени, режим временной задержки, режим тайм-аута. Переполнение каждого из счетчиков может быть зафиксировано системой прерываний контроллера индивидуально.

Система прерываний. Одним из наиболее важных узлов микроконтроллеров семейства РССР Z8 является система прерываний, которая включает шесть уровней прерываний. Источники прерываний при этом длятся на внешние и внутренние. К внешним относятся прерывания от выводов P3.0…P3.3 порта P3 (передний фронт импульса или срабатывание компаратора для выводов P3.2 и P3.1), а к внутренним – прерывания при переполнении таймеров-счетчиков. Управление системой прерываний осуществляется через специальные регистры или специальными командами. Каждый из уровней прерывания имеет соответствующий ему вектор прерывания, который является адресом программы обработки прерывания, т.е. указателем на подпрограмму, которая должна исполняться в случае срабатывания разрешенного прерывания. Каждый из векторов, соответствующий тому или иному уровню прерывания, располагается в 12 ячейках в самом начале адресного пространства памяти программ. После возникновения прерывания в счетчик команд автоматически загружается соответствующий вектор прерывания, и управление автоматически передается программе обработки прерываний. Это происходит после предварительного сохранения текущего содержимого счетчика команд и регистра фла-

гов состояния в стеке. Таким образом, процессор может в реальном масштабе времени моментально реагировать на внешние и внутренние события, требующие немедленного обслуживания. По окончании работы подпрограммы обработки прерывания управление возвращается прерванному фрагменту благодаря восстановлению, предварительно сохраненному в стеке значению текущего, на момент возникновения прерывания, состояния программного счетчика. Возможна отработка вложенных прерываний. Для каждого из прерываний программно может быть установлен свой приоритет по отношению к другим уровням.

Сброс. После сброса процессора система прерываний находится в пассивном состоянии и для ее активизации необходима соответствующая установка специализированных служебных регистров системы прерывания – регистра маски прерываний, регистра приоритета прерываний и регистра запроса прерываний.

Чрезвычайно важным для работы надежной работы любого микроконтроллера является организация цепи сброса. Микроконтроллеры PCCP Z8 имеют достаточно сложную структуру цепи сброса, которая учитывает максимальное количество причин, требующих приведения схемы процессора в исходное состояние. Старшая модель Z86E40 имеет специальный вывод RST, на котором возникновение низкого уровня сигнала может являться внешней причиной сброса процессора. Это удобно, если в качестве внешнего источника сброса используется специальная кнопка или схема интегрального супервизора, осуществляющего, например, мониторинг питания, или анализирующего состояние шин обмена процессора с внешней памятью. Однако даже если не использовать механизм внешнего сброса для кристалла Z86E40 (а у младших членов семейства это и невозможно, так как они не имеют отдельного вывода RST), то при включении питания любой из процессоров PCCP Z8 все равно запустится. Это обеспечивает специальная внутренняя цепь генерации сигнала “Сброс”, которая срабатывает при переходе напряжения питания контроллера через определенную границу ($\sim$2/3Uпитания). Уровень границы определяется тактовой частотой процессора и значением температуры окружающей среды.

В качестве еще одной причины, которая может привести к переводу контроллера в исходное состояние, является работа специального узла сторожевого таймера. Этот узел контроллера имеет отдельный генератор, никак не связанный со схемой тактового генератора. С помощью специального регистра можно установить период срабатывания WDT. Если на протяжении этого периода WDT не сброшен программно, он переполняется, результатом чего является сброс контроллера. Таким образом, значительно снижается вероятность сбоя процессора, связанная, как правило, с зависанием программного обеспечения.

Кроме того, микроконтроллеры Z8 имеют полезный механизм автоматического запуска WDT при включении питания. Поэтому программно WDT запускать необязательно: прибор выполнит это автоматически, как только напряжение питания схемы превысит ~3.5 В, что, в конечном счете, приводит к значительной устойчивости микропроцессорной системы при краевых условиях и флуктуациях питания.

5.6.2. Контроллеры с расширенным набором функций

Контроллеры PCCP Z8 с расширенным набором функций предназначены для решения задач, требующих расширения стандартной архитектуры семейства за счет как усовершенствования традиционных, так и введения новых периферийных узлов. Основным достоинством микроконтроллеров семейства PCCP Z8 является их высокая надежность и высокое отношение производительности к цене прибора с учетом общей стоимости разработки. Доказательство этого можно увидеть, если посмотреть, на какой элементной базе построены схемы управления большинства устройств, которые окружают нас в повседневной жизни, -– компьютерные клавиатуры и манипуляторы, кофемолки и микроволновые печи, пульты дистанционного управления телевизоров и видеомагнитофонов, приборы автомобильной электроники и преобразователи электроэнергии. Во многих типах этих и множества других устройств используют контроллеры PCCP Z8. В приборах, выпускаемых массово, используются не однократно “прошиваемые” члены семейства, а контроллеры с масочным ПЗУ (Z8Cxx и Z8Lxx). Однако сам факт их выбора большим количеством зарубежных производителей в качестве основного элемента системы управления для изделий широкого потребления приборов семейства PCCP Z8 говорит о надежности и рациональности использования этих микросхем при создании продукции массового спроса.

Фирма Zilog выпускает контроллеры семейства PCCP Z8 в двух исполнениях, различие между которыми определяется диапазоном рабочих температур, в которых может функционировать прибор. Доступны приборы с индексом S – стандартный температурный диапазон 0…+70°C и E – расширенный температурный диапазон -40…+105°C. Несмотря на некоторое увеличение цены на приборы расширенного диапазона, никаких специальных мер или особых доработок при их изготовлении не выполняется. Просто наличие индекса E в аббревиатуре обозначения контроллера говорит о том, что он прошел специальную проверку, гарантирующую соблюдение всех характеристик и режимов работы в расширенном температурном диапазоне, в отличие от приборов с индексом S, которые такой проверке не подвергались или были в ходе ее отбракованы.

5.6.3. Разработка программного обеспечения

Фирмой Zilog был разработан полномасштабный дешевый эмулятор-программатор Z8CCP00ZEM, специально предназначенный для работы только с процессорами PCCP Z8. Он продается фирмой, в соответствии с собственной протекционистской по отношению к пользователям политикой, по заниженной демпинговой цене, так как даже если попытаться оценить суммарную стоимость всех входящих в состав этого прибора компонентов, то эта сумма превысит цену готового Z8CCP00ZEM. Такой шаг Zilog не может не привлечь разработчиков микропроцессорных устройств к семейству Z8. Эмулятор-программатор позволяет производить полномасштабный процесс отладки управляющей программы, эмулируя в реальном масштабе времени работу контроллера выбранного типа, а затем "прошивать" подготовленное и отлаженное программное обеспечение в дешевые кристаллы. Таким образом, надобность в различных искусственных симуляторах, платах-конструкторах, программных отладочных мониторах и т.д., которые либо не дают возможности работы в реальном масштабе времени, либо сжирают значительное количество ресурсов самого процессора, отпадает.

Эмулятор Z8CCP00ZEM соединяется с персональным компьютером разработчика через COM-порт и питается от автономного источника питания +9 В. Для связи с макетом пользователя схема эмулятора содержит специальные шлейфы для различных типов корпусов эмулируемых кристаллов (DIP18, DIP28, DIP40). Напряжение питания и тактирование отлаживаемого приложения пользователя может производиться как с платы самого эмулятора, так и непосредственно с макета пользователя. Со стороны пользователя эмулятор поддерживается современной интегрированной оболочкой, которая специально разработана для использования в среде WindowsNT/Win95. В качестве эмуляционного кристалла в составе эмулятора используется специализированный DSP-процессор, позволяющий в реальном масштабе времени работать с любым мыслимым количеством точек останова, оперативно наблюдать и изменять содержимое всех ресурсов отлаживаемого прибора; не покидая интегрированной оболочки, корректировать отлаживаемую программу, включая отладку в исходном коде ассемблера, а также реализовывать пошаговый режим работы.

После окончательной доводки с помощью эмулятора программного обеспечения пользователь может воспользоваться этим же прибором для программирования внутренней памяти любого из микроконтроллеров семейства PCCP Z8. Для того чтобы "прошить" внутреннюю однократно программируемую память программ кодом подготовленного и отлаженного с помощью эмулятора программного обеспечения, сам кристалл размещается в одной из колодок с нулевым уси-

лием, которая соответствует его корпусу. Затем пользователь посредством специального режима интегрированной оболочки выбирает тип программируемого устройства, тип внешней цепи тактового генератора, режим работы WDT, а также моду защиты памяти данных и памяти программ и осуществляет процесс программирования. После его окончания шлейф эмулятора отсоединяется от макета пользователя, а вместо него подставляется дешевый, прошитый кодом отлаженной программы микроконтроллер.

Для программирования кристаллов семейства PCCP Z8 используется бесплатно распространяемые фирмой Zilog ассемблер ZASM и макроассемблер ZMASM, совместимые с оболочками эмуляторов Z8CCP00ZEM. Доступны две интегрированные оболочки для эмуляторов: Z8ICE (хорошо работает под W3.11) и Zilog Development Studio (полнофункциональная среда отладки для W95/98/NT). Существуют также компилятор С для комбинированного программирования, интегрированная среда Z8 COMPASS фирмы Programm Language Corp., а также продукция Avocet, IAR и др. Для самостоятельной разработки начинающим пользователем программного обеспечения Z8 в Интернете можно найти симуляторы этого микроконтроллера, в частности NOICE.

6. СТРУКТУРНАЯ ОРГАНИЗАЦИЯ И СИСТЕМА КОМАНД MCS51

6.1. Общее описание

Семейство 8-разрядных микроконтроллеров MCS51 было выпущено фирмой Intel в начале 80-х годов прошлого века. Первые модификации кристаллов (около 7) были выполнены по высокоуровневой *n*-МОП (HMOS) технологии и являлись функционально завершенными однокристальными микроЭВМ гарвардской архитектуры, один из основных принципов которой состоит в логическом разделении адресных пространств памяти программ и данных. Микроконтроллеры MCS51 являются функционально завершенными однокристальными микроЭВМ, содержащими все необходимые узлы для работы в автономном режиме, и предназначены для реализации различных цифровых алгоритмов управления. Удачный набор периферийных устройств, возможность гибкого выбора внешней или внутренней программной памяти и приемлемая цена обеспечили этому микроконтроллеру успех на рынке. С точки зрения технологии микроконтроллер i8051 являлся для своего времени очень сложным изделием – в кристалле было использовано 128 тысяч транзисторов, что в 4 раза превышало количество транзисторов в 16-разрядном микропроцессоре i8086.

Микроконтроллер MCS51 выполнен в корпусе БИС, имеющем 40 внешних выводов. Цоколевка корпуса MCS51 и наименования выводов показаны на рис. 6.1. Для работы MCS51 требуется один источник электропитания +5 В. Через четыре программируемых порта ввода-вывода MCS51 взаимодействует со средой в стандарте ТТЛ-схем с тремя состояниями выхода.

Корпус MCS51 имеет два вывода для подключения кварцевого резонатора, четыре вывода для сигналов, управляющих режимом работы МК, и восемь линий порта 3, которые могут быть запрограммированы пользователем на выполнение специализированных (альтернативных) функций обмена информацией со средой.

Важную роль в достижении такой высокой популярности семейства i8051 сыграла открытая политика фирмы Intel, родоначальницы архитектуры, направленная на широкое распространение лицензий на ядро 8051 среди большого количества ведущих полупроводниковых компаний мира. С развитием полупроводниковой технологии последующие версии микросхем MCS51 стали изготовлять по более

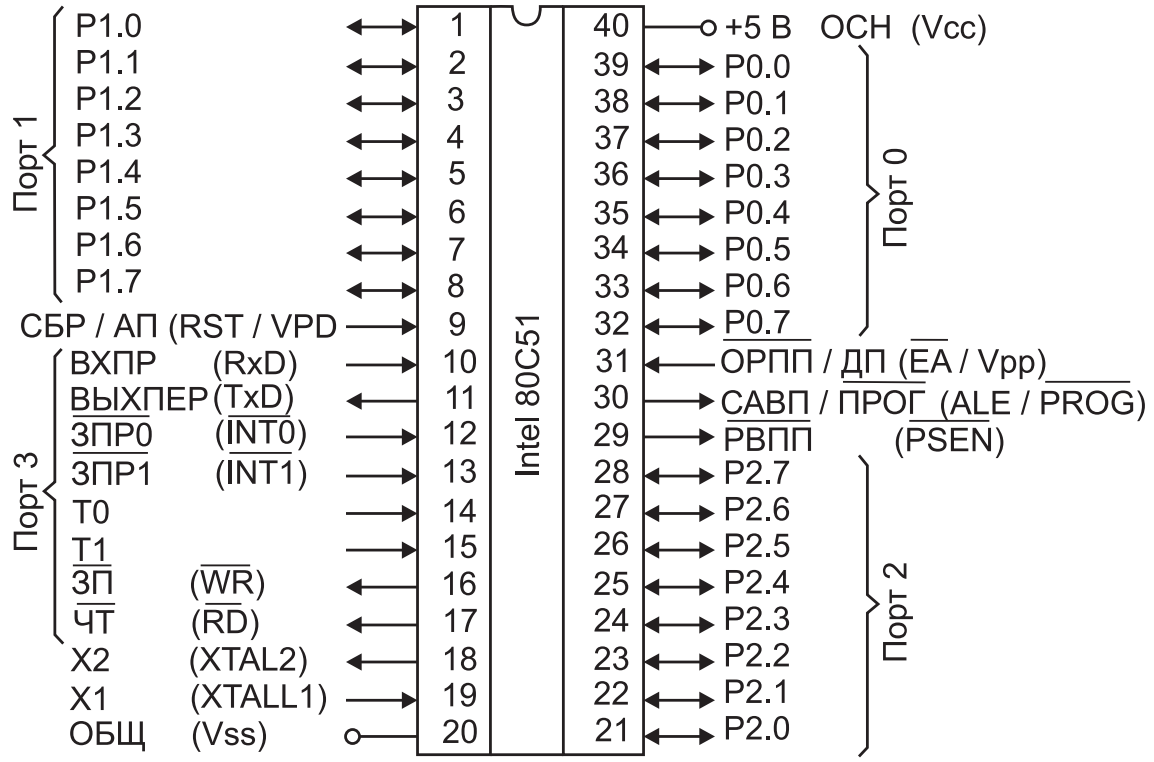


Рис. 6.1. Цоколевка корпуса MCS51 и наименование выводов

совершенной и низкопотребляющей КМОП (CHMOS) технологии (в активном режиме потребление кристаллов было доведено до 10 мА).

В результате на сегодняшний день существует более 200 модификаций микроконтроллеров семейства i8051, выпускаемых двумя десятками различных компаний. Эти модификации включают в себя кристаллы с широчайшим спектром периферии: от простых 20-выводных устройств с одним таймером и 1К программной памяти до сложнейших 100-выводных кристаллов с 12-разрядными АЦП, массивами таймеров-счетчиков, аппаратными 16-разрядными умножителями и 64 Кб программной памяти на кристалле. Максимальная тактовая частота достигает 24 МГц; для отдельных групп кристаллов – 33 МГц. Каждый год появляются все новые варианты представителей этого семейства. Основными направлениями развития являются увеличение быстродействия (повышение тактовой частоты и переработка архитектуры), снижение напряжения питания и потребления, увеличение объема ОЗУ и Flash-памяти на кристалле с возможностью внутрисхемного программирования, введение в состав периферии микроконтроллера сложных устройств типа системы управления приводами, CAN и USB интерфейсов и т.п.

Все микроконтроллеры из семейства MCS51 имеют общую систему команд. Наличие дополнительного оборудования влияет только на количество регистров специального назначения. Основными про-

изводителями клонов семейства x51 в мире являются фирмы Philips, Siemens, Intel, Atmel, Dallas, Oki, AMD, MHS, Gold Star, Winbond, Temic, Silicon Systems и ряд других. В рамках СССР производство микроконтроллера 8051 осуществлялось в Киеве, Воронеже (1816ВЕ31/51, 1830ВЕ31/51), Минске (1834ВЕ31) и Новосибирске (1850ВЕ31). Микроконтроллеры данного семейства выпускаются в PLCC, DIP и QFP корпусах и могут работать в следующих температурных диапазонах: коммерческий (0…+70°C); расширенный (-40…+85°C); для военного использования (-55…+125°C). Примерами микроконтроллеров семейства MCS51 с расширенными возможностями могут служить такие изделия как 8XC51FA, 8XC51GB и 80C152.

Для подготовки математического обеспечения микроконтроллеров MCS51 используются в основном языки ASM-51, C51, для которых существует ряд достаточно хорошо зарекомендовавших себя компиляторов, библиотек стандартных подпрограмм, программных симуляторов и аппаратных эмуляторов, производимых различными зарубежными и отечественными фирмами.

6.2. Структурная схема MCS51

Основу структурной схемы MCS51 (рис. 6.2) образует внутренняя двунаправленная 8-битная шина, которая связывает между собой все основные узлы и устройства: резидентную память, АЛУ, блок регистров специальных функций, устройство управления и порты ввода-вывода. Рассмотрим основные элементы структуры и особенности организации вычислительного процесса в MCS51.

6.2.1. Арифметико-логическое устройство

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

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

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

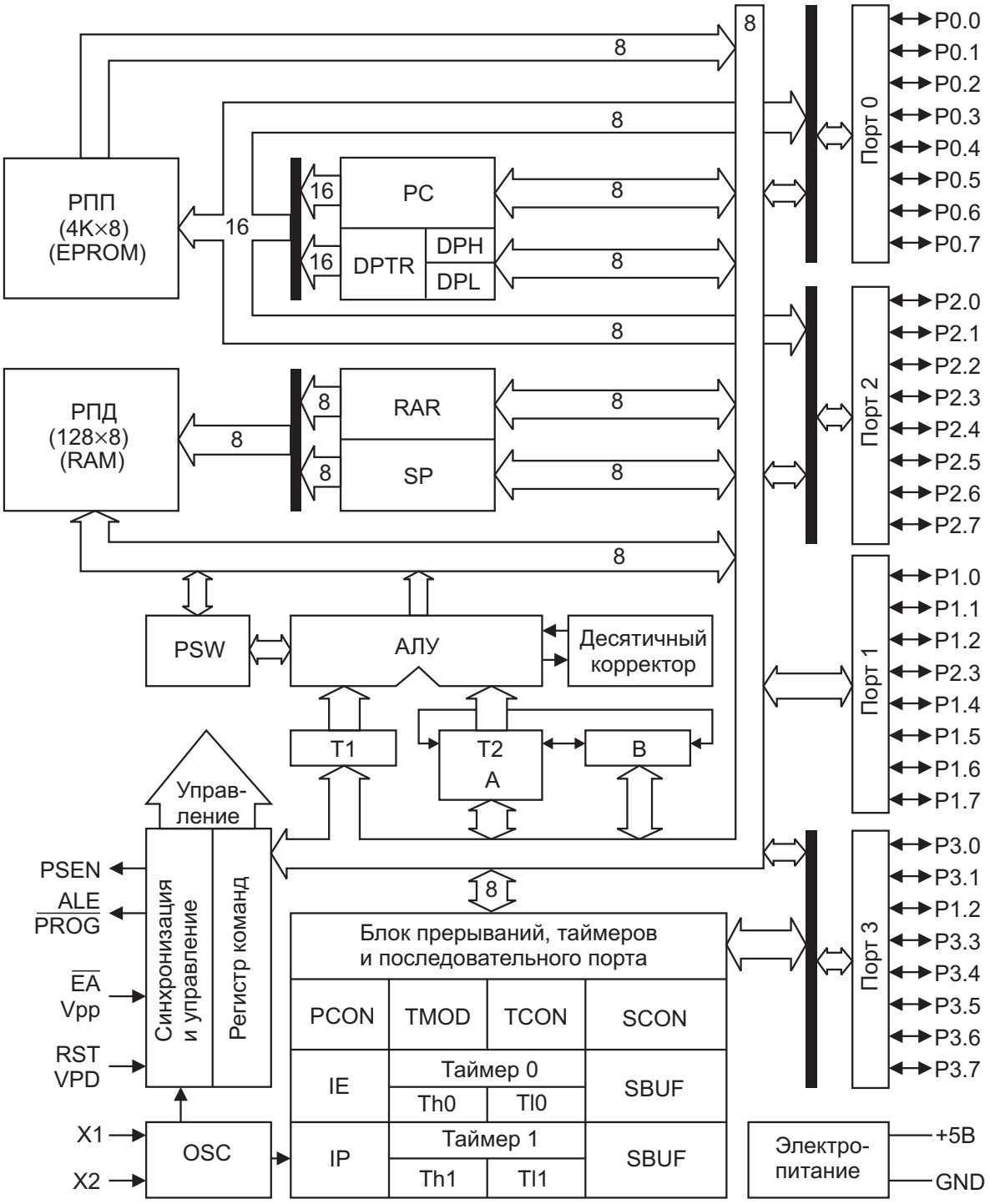


Рис. 6.2. Структурная схема MCS51

16-битных регистровых пар. В АЛУ реализуется механизм каскадного выполнения простейших операций для реализации сложных команд. Так, например, при выполнении одной из команд условной передачи управления по результату сравнения в АЛУ трижды инкрементируется счетчик команд, дважды производится чтение из РПД, выполняется арифметическое сравнение двух переменных, формируется 16-битный

адрес перехода и принимается решение о том, делать или не делать переход по программе. Все перечисленные операции выполняются в АЛУ всего лишь за 2 мкс.

Важной особенностью АЛУ является его способность оперировать не только байтами, но и битами. Отдельные программно доступные биты могут быть установлены, сброшены, инвертированы, переданы, проверены и использованы в логических операциях. Эта способность АЛУ оперировать битами столь важна, что во многих описаниях MCS51 говорится о наличии в нем булевского процессора. Для управления объектами часто применяются алгоритмы, содержащие операции над входными и выходными булевскими переменными (истина/ложь), реализация которых средствами обычных микропроцессоров сопряжена с определенными трудностями.

Таким образом, АЛУ может оперировать четырьмя типами информационных объектов: булевскими (1 бит), цифровыми (4 бита), байтными (8 бит) и адресными (16 бит). В АЛУ выполняется 51 различная операция пересылки или преобразования этих данных. Поскольку используется 11 режимов адресации (7 для данных и 4 для адресов), то путем комбинирования *операция /режим адресации* базовое число команд 111 расширяется до 255 из 256 возможных при однобайтном коде операции.

6.2.2. Резидентная память

Память программ и память данных, размещенные на кристалле MCS51, физически и логически разделены, имеют различные механизмы адресации, работают под управлением различных сигналов и выполняют разные функции.

Память программ

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

Память данных

Память данных (ОЗУ) предназначена для хранения переменных в процессе выполнения прикладной программы, адресуется одним бай-

том и имеет емкость 128 байт. Кроме того, к адресному пространству РПД (табл. 6.1) примыкают адреса регистров специальных функций (РСФ). Память программ, так же как и память данных, может быть расширена до 64 Кбайт путем подключения внешних БИС.

Аккумулятор и регистр PSW

Аккумулятор является источником операнда и местом фиксации результата при выполнении арифметических, логических операций и ряда операций передачи данных. Только с использованием аккумулятора могут быть выполнены операции сдвигов, проверка на нуль, формирование флага паритета и т.п.

При выполнении многих команд в АЛУ формируется ряд признаков операции (флагов), которые фиксируются в регистре слова состояния программы (**P**rogram **S**tate **W**ord – PSW). В табл. 6.2 приводится перечень флагов PSW, даются их символические имена и описываются условия их формирования. Наиболее *активным* флагом PSW является флаг переноса, который принимает участие и модифицируется в процессе выполнения множества операций, включая сложение, вычитание и сдвиги. Флаг переноса (C) выполняет также функции *булевого аккумулятора* в командах, манипулирующих битами. Флаг переполнения (OV) фиксирует арифметическое переполнение при операциях над целыми числами со знаком и делает возможным использование арифметики в дополнительных кодах. АЛУ не управляет флагами селекции банка регистров (RS0, RS1), и их значение полностью определяется прикладной программой и используется для выбора одного из четырех регистровых банков. Широкое распространение получило представление о том, что в микропроцессорах, архитектура которых опирается на аккумулятор, большинство команд работают с ним, используя адресацию *по умолчанию* (неявную). В MCS51 дело обстоит иначе. Хотя процессор в MCS51 имеет в своей основе аккумулятор, однако он может выполнять множество команд и без участия аккумулятора. Например, данные могут быть переданы из любой ячейки РПД в любой регистр, любой регистр может быть загружен непосредственным операндом и т.д. Многие логические операции могут быть выполнены без участия аккумулятора. При этом переменные могут быть инкрементированы, декрементированы и проверены (test) без использования аккумулятора. Флаги и управляющие биты могут быть проверены и изменены аналогично.

Регистры-указатели

Восьмибитный указатель стека (SP) может адресовать любую заданную область РПД. Его содержимое инкрементируется прежде, чем

Таблица 6.1. Блок регистров специальных функций

Символ	Наименование	Адрес	Исходное состояние
* ACC	Аккумулятор	0E0H	00H
* B	Расширитель аккумулятора	0F0H	00H
* PSW	Слово состояния программы	0D0H	00H
SP	Указатель стека	81H	07H
DPTR	Указатель данных (DPH)	83H	00H
	(DPL)	82H	00H
* P0	Порт 0	80H	0FFH
* P1	Порт 1	90H	0FFH
* P2	Порт 2	0A0H	0FFH
* P3	Порт 3	0B0H	0FFH
* IP	Приоритеты прерываний	0B8H	xxx00000B
* IE	Маски прерываний	0A8H	0xx00000B
TMOD	Режим таймера/счетчика	89H	00H
* TCON	Управление/статус таймера	88H	00H
TH0	Таймер 0 (старший байт)	8CH	00H
TL0	Таймер 0 (младший байт)	8AH	00H
TH1	Таймер 1 (старший байт)	8DH	00H
TL1	Таймер 1 (младший байт)	8BH	00H
* SCON	Управление приемопередатчиком	98H	00H
SBUF	Буфер приемопередатчика	99H	xxxxxxxxB
PCON	Управление мощностью	87H	0xxxxxxxB

Примечание. Регистры, имена которых отмечены знаком (*), допускают адресацию отдельных бит в соответствии с картой адресуемых битов (рис. 6.19); Исходное состояние – значение регистров после начального сброса.

данные будут запомнены в стеке в ходе выполнения команд PUSH и CALL. Содержимое регистра SP декрементируется после выполнения команд POP и RET. Подобный способ адресации элементов стека называют предынкрементным/постдекрементным. В процессе инициализации MCS51 после сигнала RST в регистр SP автоматически загружается код 07H. Это значит, что если прикладная программа не переопределяет стек, то первый элемент данных в стеке будет располагаться в ячейке РПД с адресом 08H. Двухбайтный регистр-указатель данных (DPTR) обычно используется для фиксации 16-битного адреса в операциях с обращением к внешней памяти. Командами MCS51 регистр-указатель данных может быть использован или как 16-битный регистр, или как два независимых 8-битных регистра (DPH и DPL).

Таблица 6.2. Формат слова состояния программы (PSW)

Символ	Позиция	Имя и назначение
C	PSW.7	Флаг переноса. Устанавливается и сбрасывается аппаратурными средствами или программой при выполнении арифметических и логических операций
AC	PSW.6	Флаг вспомогательного переноса. Устанавливается и сбрасывается только аппаратурными средствами при выполнении команд сложения и вычитания. Он сигнализирует о переносе или займе в бите 3
F0	PSW.5	Флаг F0. Может быть установлен, сброшен или проверен программой как флаг, специфицируемый пользователем
RS1	PSW.4	Выбор банка регистров. Устанавливается и сбрасывается программой для выбора рабочего банка регистров (см. примечание)
RS0	PSW.3	Выбор банка регистров. Устанавливается и сбрасывается программой для выбора рабочего банка регистров (см. примечание)
OV	PSW.2	Флаг переполнения. Устанавливается и сбрасывается аппаратурно при выполнении арифметических операций
–	PSW.1	Не используется
P	PSW.0	Флаг паритета. Устанавливается и сбрасывается аппаратурно в каждом цикле команды и фиксирует нечетное/четное число единичных битов в аккумуляторе, т.е. выполняет контроль по четности

Примечание. Выбор рабочего банка регистров

RS1	RS0	Банк	Границы адресов
0	0	0	00H–07H
0	1	1	08H–0FH
1	0	2	10H–17H
1	1	3	18H–1FH

Таймер/счетчик

В составе средств MCS51 имеются регистровые пары с символическими именами TH0, TL0 и TH1, TL1, на основе которых функционируют два независимых программно-управляемых 16-битных таймера/счетчика событий.

Буфер последовательного порта

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

Регистры специальных функций

Регистры с символическими именами IP, TMOD, IE, TCON, SCON и PCON используются для фиксации и программного изменения управляющих битов и битов состояния таймера/счетчика, схемы прерывания, приемопередатчика последовательного порта и для управления мощностью электропитания MCS51. Их организация будет описана ниже при рассмотрении особенностей работы MCS51 в различных режимах.

6.2.3. Устройство управления и синхронизации

Кварцевый резонатор, подключаемый к внешним выводам X1 и X2 корпуса MCS51, управляет работой внутреннего генератора, который в свою очередь формирует сигналы синхронизации.

Устройство управления MCS51 на основе сигналов синхронизации формирует машинный цикл фиксированной длительности, равной 12 периодам резонатора или шести состояниям первичного управляющего автомата (S1–S6). Каждое состояние управляющего автомата содержит две фазы (P1, P2) сигналов резонатора. В фазе P1, как правило, выполняется операция в АЛУ, а в фазе P2 осуществляется межрегистровая передача. Весь машинный цикл состоит из 12 фаз, начиная с фазы S1P1 и кончая фазой S6P2, как показано на рис. 6.3. Эта временная диаграмма иллюстрирует работу устройства управления MCS51 при выборке и исполнении команд различной степени сложности. Все заштрихованные сигналы являются внутренними и недоступны пользователю MCS51 для контроля. Внешними, наблюдаемыми сигналами являются только сигналы резонатора и выходной сигнал разрешения

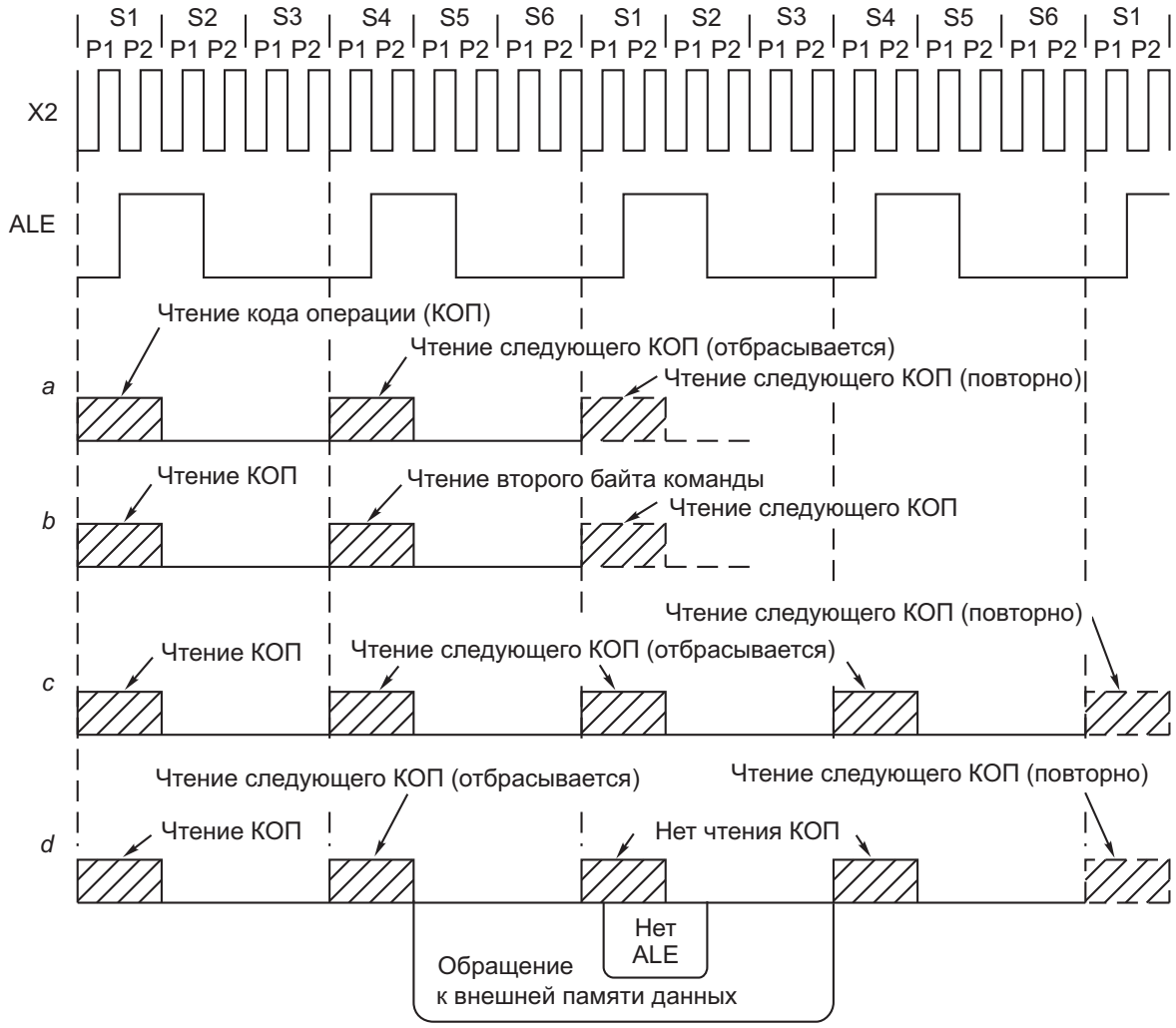


Рис. 6.3. Последовательности выборки и выполнения команд в MCS51: *a* – команда 1 байт/1 цикл, например INC A; *b* – команда 2 байта/1 цикл, например ADD A,#d; *c* – команда 1 байт/2 цикла, например INC DPTR; *d* – команда 1 байт/2 цикла, например MOVX

фиксации адреса внешней памяти (ALE). Как видно из временной диаграммы, строб адреса ALE формируется дважды за один машинный цикл (S1P2–S2P1 и S4P2–S5P1) и используется для управления процессом обращения к внешней памяти.

На рис. 6.3 приведены временные диаграммы выполнения некоторых команд MCS51, различающихся длиной и временем выполнения. Большинство команд MCS51 выполняется за один машинный цикл. Это могут быть как однобайтные команды (рис. 6.3, *a*), так и двухбайтные с непосредственным операндом (рис. 6.3, *b*). Некоторые команды, оперирующие с 2-байтными словами (рис. 6.3, *c*) или связанные с обращением к внешней памяти (рис. 6.3, *d*), выполняются за два машинных цикла. Только команды деления и умножения требуют четырех машинных циклов. На основе этих особенностей работы устройства управления MCS51 производится расчет времени исполнения прикладных программ.

6.2.4. Порты ввода-вывода информации

Все четыре порта MCS51 предназначены для ввода или вывода информации побайтно. Схемотехника портов ввода-вывода MCS51 для одного бита показана на рис. 6.4 (порты 1 и 2 имеют примерно такую же структуру, как и порт 3). Каждый порт содержит управляемые регистр-защелку, входной буфер и выходной драйвер.

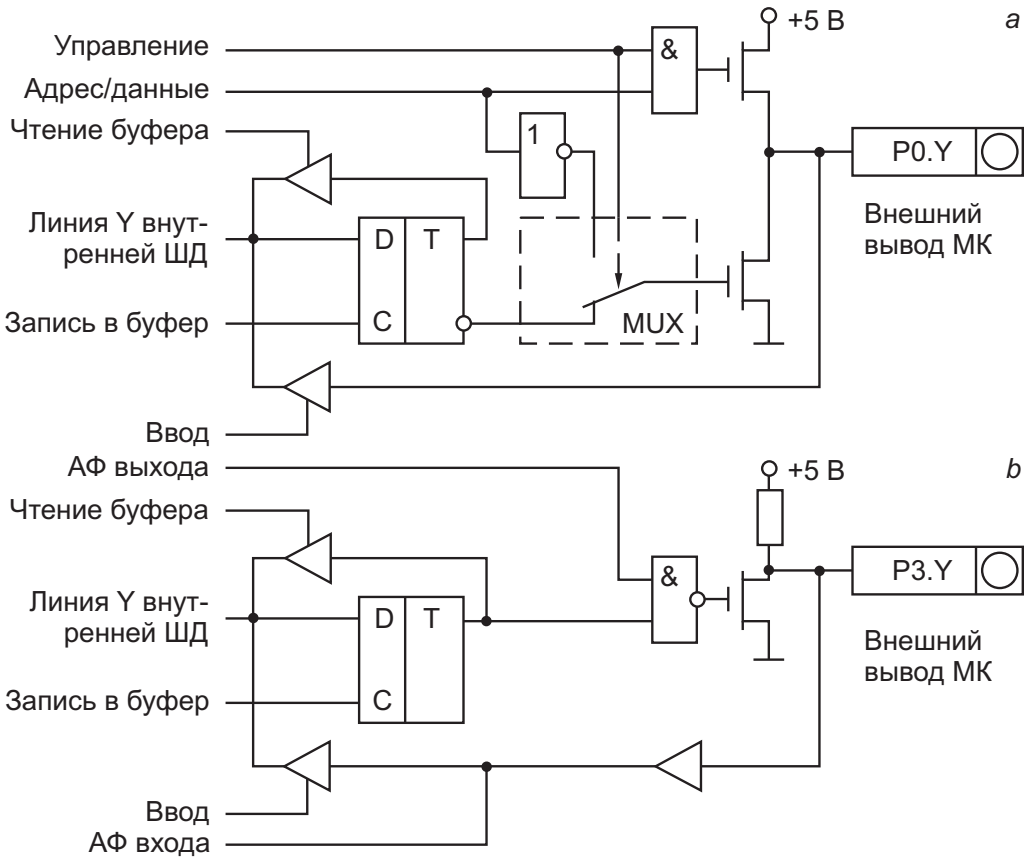


Рис. 6.4. Схемотехника портов ввода-вывода MCS51: *a* – порт 0; *b* – порт 3

Выходные драйверы портов 0 и 2, а также входной буфер порта 0 используются при обращении к внешней памяти (ВП). При этом через порт 0 в режиме временного мультиплексирования сначала выводится младший байт адреса ВП, а затем выдается или принимается байт данных. Через порт 2 выводится старший байт адреса в тех случаях, когда разрядность адреса равна 16 бит.

Все выводы порта 3 могут быть использованы для реализации альтернативных функций (АФ), перечисленных в табл. 6.3. Альтернативные функции могут быть задействованы путем записи 1 в соответствующие биты регистра-защелки (P3.0…P3.7) порта 3. Порт 0 является двунаправленным, а порты 1, 2 и 3 – квазидвунаправленными. Каждая линия портов может быть использована независимо для ввода или вывода информации. Для того чтобы некоторая линия порта ис-

Таблица 6.3. Альтернативные функции порта 3

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

пользовалась для ввода, в *D*-триггер регистра-защелки порта должна быть записана 1, которая закрывает МОП-транзистор выходной цепи. По сигналу RST в регистры-защелки всех портов автоматически записываются единицы, настраивающие их тем самым на режим ввода. Все порты могут быть использованы для организации ввода-вывода информации по двунаправленным линиям передачи. Однако порты 0 и 2 не могут быть использованы для этой цели в случае, если микроконтроллер имеет внешнюю память, связь с которой организуется через общую разделяемую шину адреса/данных, работающую в режиме временного мультиплексирования.

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

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

Входные сигналы для MCS51 могут формироваться ТТЛ-схемами или *n*-МОП-схемами. Допустимо использование в качестве источников сигналов для MCS51 схем с открытым коллектором или открытым стоком. Однако при этом время изменения входного сигнала при переходе из 0 в 1 окажется сильно затянутым.

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

ANL – логическое И, например ANL P1,A;
ORL – логическое ИЛИ, например ORL P2,A;
XRL – исключающее ИЛИ, например XRL P3,A;
JBC – переход, если в адресуемом бите единица, и последующий сброс бита, например JBC P1.1,LABEL;
CPL – инверсия бита, например CPL P3.3;
INC – инкремент порта, например INC P2;
DEC – декремент порта, например DEC P2;
DJNZ – декремент порта и переход, если его содержимое не равно нулю, например DJNZ P3,LABEL;
MOV PX.Y,C – передача бита переноса в бит Y порта X;
SET PX.Y – установка бита Y порта X;
CLR PX.Y – сброс бита Y порта X.

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

Причиной, по которой команды *чтение – модификация – запись* обеспечивают раздельный доступ к регистру-защелке порта и к внеш-

ним выводам порта, является необходимость исключения возможности неправильного прочтения уровней сигналов на внешних выводах. Для примера предположим, что линия Y порта X соединяется с базой мощного транзистора и выходной сигнал на ней предназначен для его управления. Когда в данный бит записана 1, то транзистор включается. Если для проверки состояния исполнительного механизма (в нашем случае – мощного транзистора) прикладной программе требуется прочитать состояние выходного сигнала в том же бите порта, то считывание сигнала с внешнего вывода порта, а не из D-триггера регистра-защелки порта приведет к неправильному результату: единичный сигнал на базе транзистора имеет относительно низкий уровень и будет интерпретирован в микроконтроллере как сигнал 0. Команды *чтение–модификация–запись* реализуют считывание из регистра-защелки, а не с внешнего вывода порта, что обеспечивает получение правильного значения 1.

На рис. 6.5 приведены временные диаграммы, иллюстрирующие процесс выполнения операций ввода-вывода информации через порты MCS51.

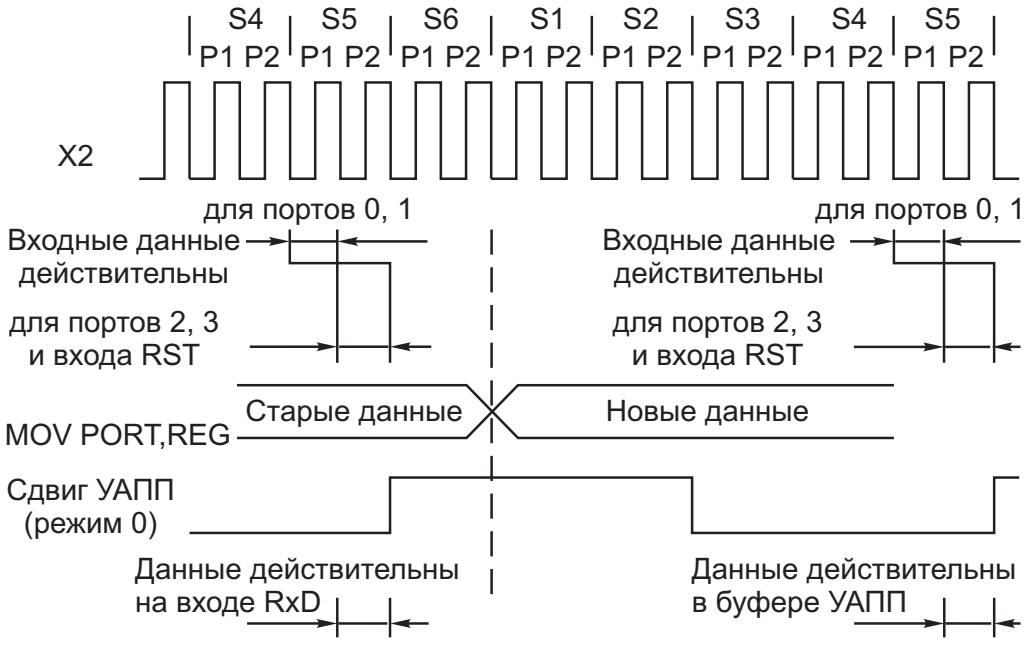


Рис. 6.5. Временные диаграммы операций ввода-вывода

6.2.5. Доступ к внешней памяти

В микроконтроллерных системах, построенных на основе MCS51, возможно использование двух типов внешней памяти: постоянной памяти программ (ВПП) и оперативной памяти данных (ВПД). Доступ

к ВПП осуществляется при помощи управляющего сигнала $\overline{\text{PSEN}}$, который выполняет функцию строб-сигнала чтения. Доступ к ВПД обеспечивается управляющими сигналами $\overline{\text{RD}}$ и $\overline{\text{WR}}$, которые формируются в линиях P3.7 и P3.6 при выполнении портом P3 альтернативных функций (см. табл. 6.3).

При обращении к ВПП всегда используется 16-битный адрес. Доступ к ВПД возможен с использованием 16-битного адреса (MOVX A,@DPTR) или 8-битного адреса (MOVX A,@Ri). В любых случаях использования 16-битного адреса старший байт адреса фиксируется (и сохраняется неизменным в течение одного цикла записи или чтения) в регистре-защелке порта P2.

Если очередной цикл внешней памяти (MOVX A,@DPTR) следует не сразу же за предыдущим циклом внешней памяти, то неизменяемое содержимое регистра-защелки порта P2 восстанавливается в следующем цикле. Если используется 8-битный адрес (MOVX A,@Ri), то содержимое регистра-защелки порта P2 остается неизменным на его внешних выводах в течение всего цикла внешней памяти.

Через порт P0 в режиме временного мультиплексирования осуществляется выдача младшего байта адреса и передача байта данных. Сигнал ALE должен быть использован для записи байта адреса во внешний регистр. Затем в цикле записи выводимый байт данных появляется на внешних выводах порта P0 только перед появлением сигнала $\overline{\text{WR}}$. В цикле чтения вводимый байт данных принимается в порт P0 по фронту стробирующего сигнала $\overline{\text{RD}}$. При любом обращении к внешней памяти устройство управления MCS51 загружает в регистр-защелку порта P0 код 0FFH, стирая тем самым информацию, которая могла в нем храниться.

Доступ к ВПП возможен при выполнении двух условий: либо на вход отключения резидентной памяти программ ($\overline{\text{EA}}$) подается активный сигнал, либо содержимое счетчика команд превышает значение 0FFFH. Наличие сигнала $\overline{\text{EA}}$ необходимо для обеспечения доступа к младшим 4К адресам адресного пространства ВПП при использовании МК31 (микроконтроллера без резидентной памяти программ).

Временны́е диаграммы на рис. 6.6 иллюстрируют процесс генерации управляющих сигналов ALE и $\overline{\text{PSEN}}$ при обращении к внешней памяти.

Основная функция сигнала ALE – обеспечить временное согласование передачи из порта P0 на внешний регистр младшего байта адреса в цикле чтения из ВПП. Сигнал ALE приобретает значение 1 дважды в каждом машинном цикле. Это происходит даже тогда, когда в цикле выборки нет обращения к ВПП. Доступ к ВПД возможен только в том случае, если сигнал ALE отсутствует. Первый сигнал

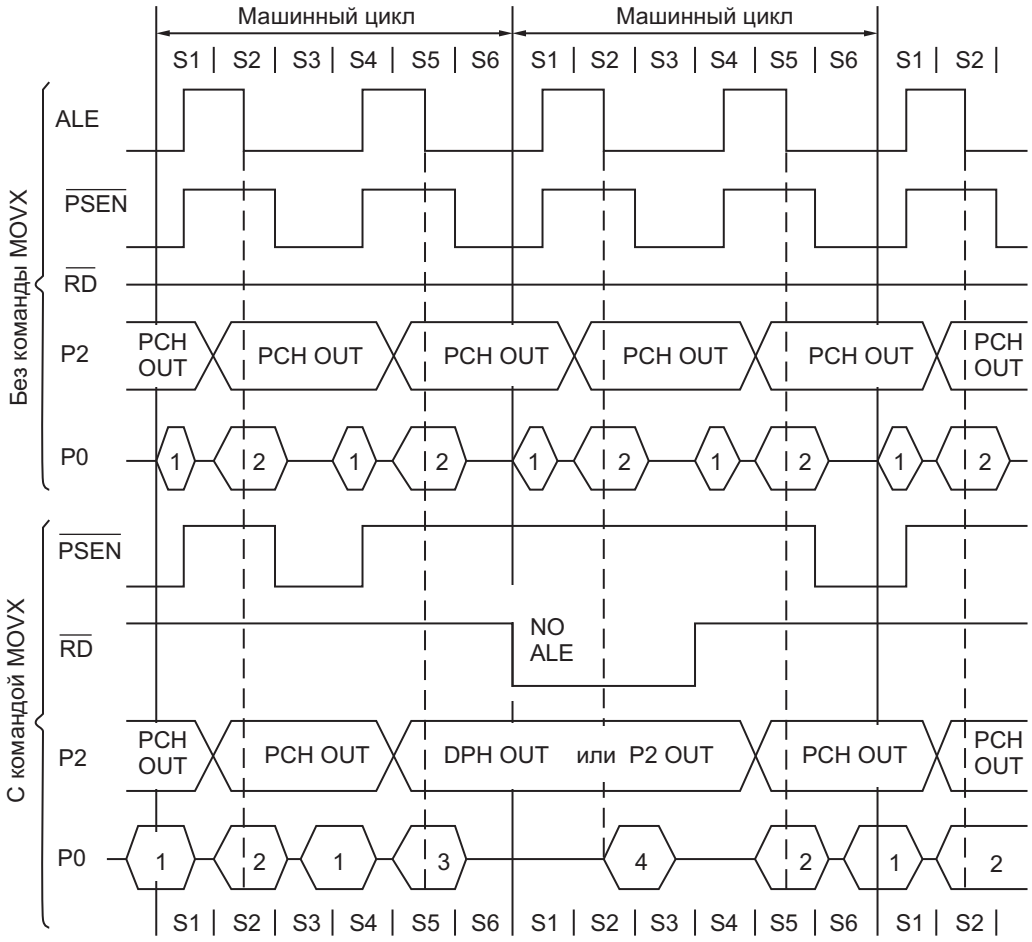


Рис. 6.6. Временные диаграммы операций с обращением к внешней памяти

ALE во втором машинном цикле команды MOVX блокируется. Следовательно, в любой МК-системе, не использующей ВПД, сигнал ALE генерируется с постоянной частотой, равной 1/16 частоты резонатора, и может быть использован для синхронизации внешних устройств или для реализации различных временных функций.

При обращении к РПП сигнал $\overline{\text{PSEN}}$ не генерируется, а при обращении к ВПП он выполняет функцию строб-сигнала чтения. Полный цикл чтения ВПД, включая установку и снятие сигнала $\overline{\text{RD}}$, занимает 12 машинных тактов (периодов тактового генератора).

Временны́е диаграммы на рис. 6.7 и 6.8 иллюстрируют процесс выборки команды из ВПП и работу ВПД в режимах чтения и записи соответственно.

Особый режим работы MCS51. Содержимое памяти программ MCS51 заполняется единожды на этапе разработки МК-системы и не может быть модифицировано в завершенном (конечном) изделии. По этой причине микроконтроллеры не являются машинами класси-

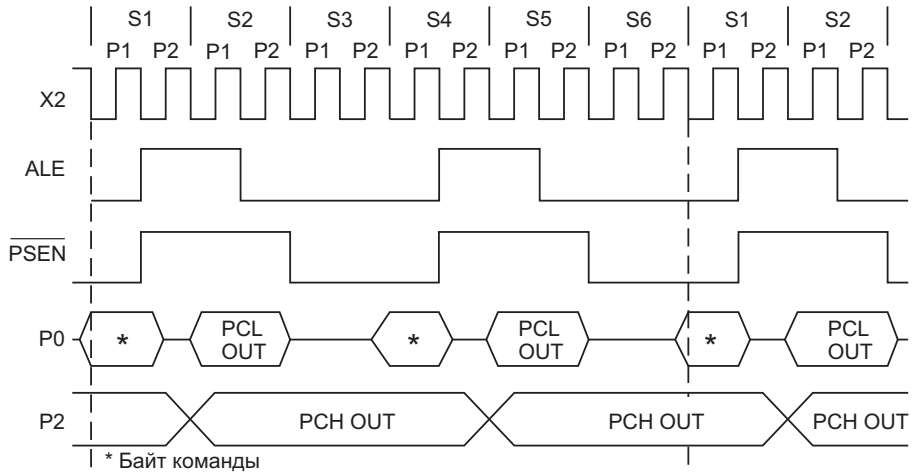


Рис. 6.7. Временная диаграмма выборки команды из ВПП

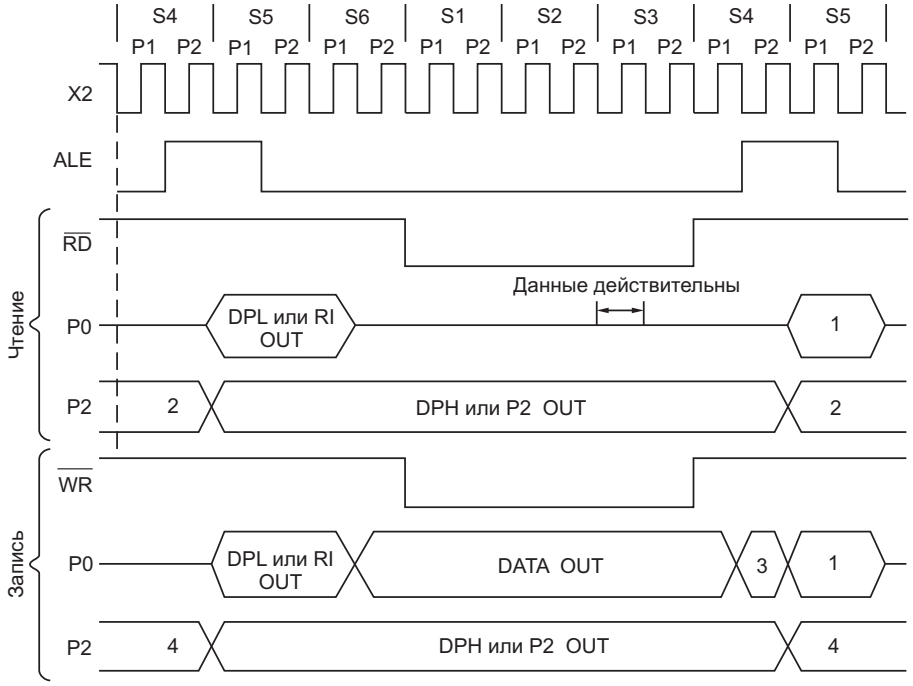


Рис. 6.8. Временная диаграмма работы с ВПД: 1 – PCL OUT, если используется ВПП; 2 – DPH или P2; 3 – PCL OUT; 4 – PCH или P2

ческой *фон неймановской* архитектуры. Оперативная память данных (резидентная или внешняя) не может быть использована для хранения кодов программы, так как в МК выборка команд производится только из области адресов памяти программ. Эта особенность архитектуры МК объясняется тем, что в большинстве применений МК требуется наличие одной неизменяемой прикладной программы, хранимой в ПЗУ, наличие ОЗУ небольшой емкости для временного хранения переменных и эффективных, а следовательно, разных методов адресации памяти программ и памяти данных. Однако на этапе раз-

работки и отладки прикладных программ машина *фон неймановского* типа оказывается очень удобной, так как позволяет разработчику оперативно изменять коды прикладной программы, размещаемой в ОЗУ. С этой целью МК-система может быть модифицирована для совмещения адресного пространства ВПП и ВПД путем подключения внешней логики (рис. 6.9). Здесь на выходе схемы И формируется строб-сигнал

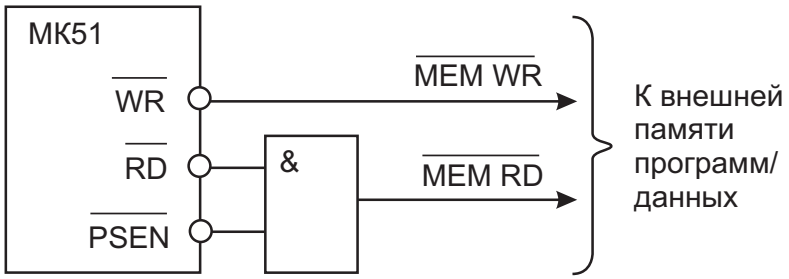


Рис. 6.9. Схема совмещения адресного пространства ВПП и ВПД

чтения, который может быть использован для объединения памяти программ и памяти данных во внешнем ОЗУ. При этом необходимо учитывать, что в MCS51 на схемном уровне реализуются пять различных и независимых механизмов адресации для доступа к РПП, РПД, ВПП, ВПД и блоку регистров специальных функций. Вследствие этого перемещаемая версия прикладной программы, которая отлаживается в среде внешней памяти программ/данных, будет отличаться от загружаемой в РПП (окончательной) версии программы.

Подобный способ организации управления внешней памятью может быть использован в тех применениях MCS51, где требуется оперативная перезагрузка или модификация прикладных программ (с помощью УВВ), как в ЭВМ классической архитектуры.

6.2.6. Таймер/счетчик

Два программируемых 16-битных таймера/счетчика (Т/С0 и Т/С1) могут быть использованы в качестве таймеров или счетчиков внешних событий. При работе в качестве таймера содержимое Т/С инкрементируется в каждом машинном цикле, т.е. через каждые 12 периодов резонатора. При работе в качестве счетчика содержимое Т/С инкрементируется под воздействием перехода из 1 в 0 внешнего входного сигнала, подаваемого на соответствующий (Т0, Т1) вывод MCS51. Опрос значения внешнего входного сигнала выполняется в момент времени S5P2 каждого машинного цикла. Содержимое счетчика будет увеличено на 1 в том случае, если в предыдущем цикле был считан входной сигнал высокого уровня (1), а в следующем – сигнал низкого уровня (0). Новое (инкрементированное) значение счетчика будет

сформировано в момент S3P1 в цикле, следующем за тем, в котором был обнаружен переход сигнала из 1 в 0. Поскольку на распознавание перехода требуется два машинных цикла, то максимальная частота подсчета входных сигналов равна 1/24 частоты резонатора. На длительность периода входных сигналов ограничений сверху нет. Для гарантированного прочтения входного считываемого сигнала он должен удерживать значение 1, как минимум, в течение одного машинного цикла MCS51.

Для управления режимами работы Т/С и для организации взаимодействия таймеров с системой прерывания используются два регистра специальных функций (TMOD и TCON – табл. 6.4 и 6.5 соответственно). Как следует из описания управляющих бит TMOD, для обоих Т/С режимы работы 0, 1 и 2 одинаковы. Режимы 3 для Т/С0 и Т/С1 различны. Рассмотрим кратко работу Т/С во всех четырех режимах.

Режим 0. Перевод любого Т/С в режим 0 делает его похожим на таймер МК48 (8-битный счетчик), на вход которого подключен 5-битный предделитель частоты на 32. Работу Т/С в режиме 0 на примере Т/С1 иллюстрирует рис. 6.10,*а*. В этом режиме таймерный регистр имеет разрядность 13 бит. При переходе из состояния *все единицы* в состояние *все нули* устанавливается флаг прерывания от таймера TF1. Входной синхросигнал таймера 1 разрешен (поступает на вход Т/С), когда управляющий бит TR1 установлен в 1 и либо управляющий бит GATE (блокировка) равен 0, либо на внешний вывод запроса прерывания $\overline{\text{INT1}}$ поступает уровень 1. Отметим попутно, что установка бита GATE в 1 позволяет использовать таймер для измерения длительности импульсного сигнала, подаваемого на вход запроса прерывания.

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

Режим 2. В режиме 2 работа организована таким образом, что пересечение (переход из состояния *все единицы* в состояние *все нули*) 8-битного счетчика TL1 приводит не только к установке флага TF1 Флаги!МК51!TF1 (рис. 6.10,*b*, но и автоматически перезагружает в TL1 содержимое старшего байта (TH1) таймерного регистра, которое предварительно было задано программным путем. Перезагрузка оставляет содержимое TH1 неизменным. В режиме 2 Т/С0 и Т/С1 работают совершенно одинаково.

Таблица 6.4. Регистр режима работы таймера/счетчика **TMOD**

Символ	Позиция	Имя и назначение
GATE	TMOD.7	Управление блокировкой T/C1. Если бит установлен, то таймер/счетчик 1 разрешен до тех пор, пока на входе INT1 высокий уровень и бит управления TR1 установлен. Если бит сброшен, то T/C разрешается, как только бит управления TR1 устанавливается
C/$\overline{T}$	TMOD.6	Бит выбора режима таймера или счетчика событий T/C1. Если бит сброшен, то работает таймер от внутреннего источника сигналов синхронизации. Если бит установлен, то работает счетчик от внешних сигналов на входе T1
M1	TMOD.5	Режим работы T/C1 (см. примечание)
M0	TMOD.4	
GATE	TMOD.3	Управление блокировкой T/C0. Если бит установлен, то таймер/счетчик 0 разрешен до тех пор, пока на входе INT0 высокий уровень и бит управления TR0 установлен. Если бит сброшен, то T/C разрешается, как только бит управления TR0 устанавливается
C/$\overline{T}$	TMOD.2	Бит выбора режима таймера или счетчика событий T/C0. Если бит сброшен, то работает таймер от внутреннего источника сигналов синхронизации. Если бит установлен, то работает счетчик от внешних сигналов на входе T0
M1	TMOD.1	Режим работы T/C0 (см. примечание)
M0	TMOD.0	

Примечание

M1	M0	Режим работы
0	0	Таймер МК48. TLx работает как 5-битный предделитель
0	1	16-битный таймер/счетчик. THx и TLx включены последовательно
1	0	8-битный автоперезагружаемый таймер/счетчик. THx хранит значение, которое должно быть перезагружено в TLx каждый раз по переполнению
1	1	Таймер/счетчик 1 останавливается. Таймер/счетчик 0: TL0 работает как 8-битный таймер/счетчик, и его режим определяется управляющими битами таймера 0. TH0 работает как 8-битный таймер, и его режим определяется управляющими битами таймера 1

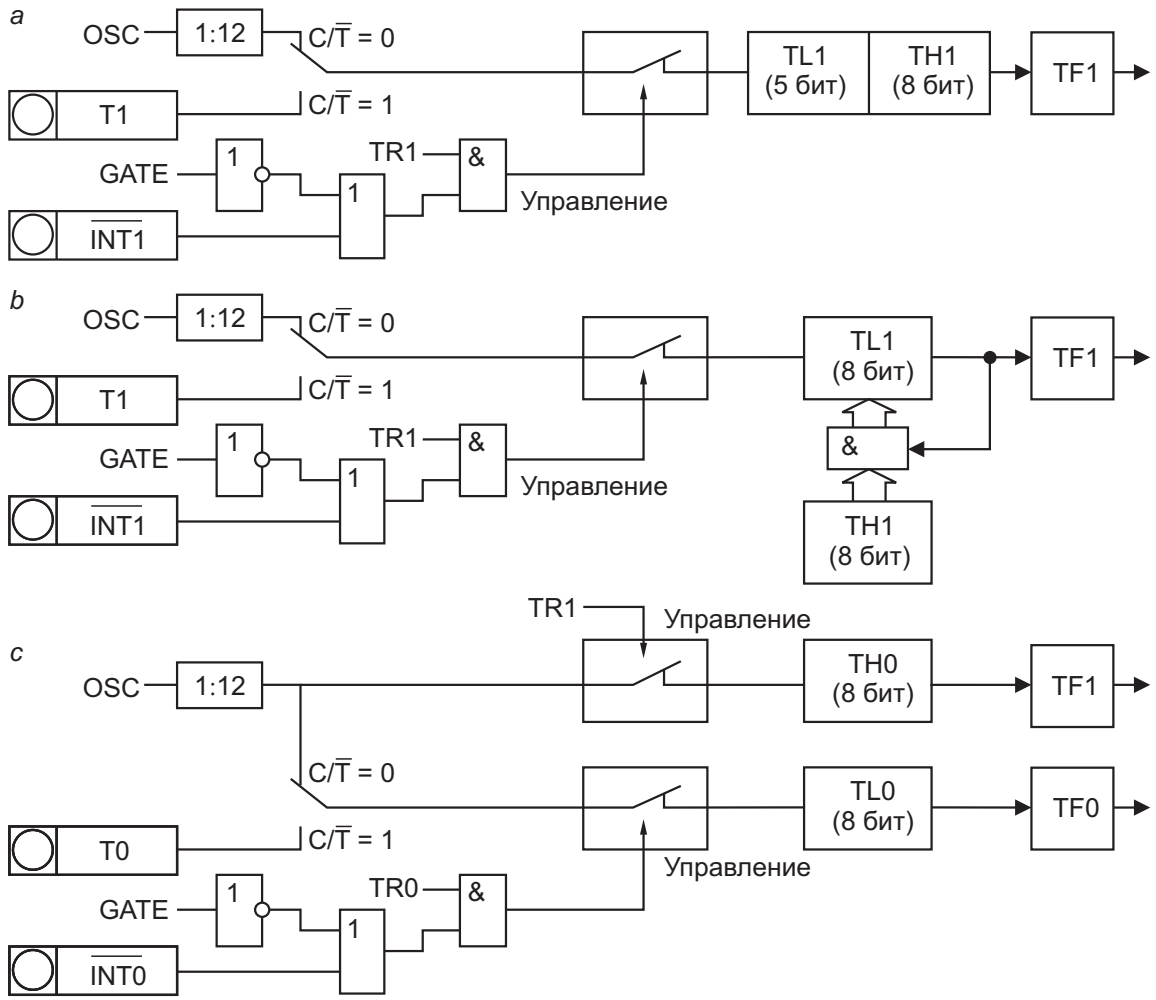


Рис. 6.10. Таймер/счетчик событий: *a* – T/C1 в режиме 0: 13-битный счетчик; *b* – T/C1 в режиме 2: 8-битный автоперегружаемый счетчик; *c* – T/C0 в режиме 3: два 8-битных счетчика

Режим 3. В режиме 3 T/C0 и T/C1 работают по-разному. T/C1 сохраняет неизменным свое текущее содержимое (как при сбросе управляющего бита TR1 в нуль). В режиме 3 TL0 и TH0 функционируют, как два независимых 8-битных счетчика (рис. 6.10,*c*). Работу TL0 определяют управляющие биты T/C0 (C/$\overline{T}$, GATE, TR0), входной сигнал $\overline{INT0}$ и флаг переполнения TF0. Работу TH0, который может выполнять только функции таймера (подсчет машинных циклов МК), определяет управляющий бит TR1. При этом TH0 использует флаг переполнения TF1. Режим 3 используется в тех случаях, когда требуется наличие дополнительного 8-битного таймера или счетчика событий. В режиме 3 MCS51 имеет в своем составе три таймера/счетчика. Если T/C0 используется в режиме 3, то T/C1 может быть включен, выключен, переведен в свой собственный режим 3, использован последовательным портом в качестве генератора частоты передачи или он может быть использован в любом применении, не требующем прерывания.

Таблица 6.5. Регистр управления/статуса таймера **TCON**

Символ	Позиция	Имя и назначение
TF1	TCON.7	Флаг переполнения таймера 1. Устанавливается аппаратурно при переполнении таймера/счетчика. Сбрасывается при обслуживании прерывания аппаратурно
TR1	TCON.6	Бит управления таймера 1. Устанавливается/сбрасывается программой для пуска/останова
TF0	TCON.5	Флаг переполнения таймера 0. Устанавливается аппаратурно. Сбрасывается при обслуживании прерывания
TR0	TCON.4	Бит управления таймера 0. Устанавливается/сбрасывается программой для пуска/останова таймера/счетчика
IE1	TCON.3	Флаг фронта прерывания 1. Устанавливается аппаратурно, когда детектируется срез внешнего сигнала $\overline{\text{ЗПР1}}$ ($\overline{\text{INT1}}$). Сбрасывается при обслуживании прерывания
IT1	TCON.2	Бит управления типом прерывания 1. Устанавливается/сбрасывается программно для спецификации запроса $\overline{\text{ЗПР1}}$ (срез/низкий уровень)
IE0	TCON.1	Флаг фронта прерывания 0. Устанавливается по срезу сигнала $\overline{\text{ЗПР0}}$. Сбрасывается при обслуживании прерывания
IT0	TCON.0	Бит управления типом прерывания 0. Устанавливается/сбрасывается программно для спецификации запроса $\overline{\text{ЗПР0}}$ (срез/низкий уровень)

6.3. Последовательный интерфейс

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

байта. Если к моменту окончания приема байта предыдущий байт не был считан из SBUF, то он будет потерян.

Последовательный порт MCS51 может работать в четырех различных режимах.

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

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

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

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

6.3.1. Регистр управления/статуса УАПП

Управление режимом работы УАПП осуществляется через специальный регистр с символическим именем SCON. Этот регистр содержит не только управляющие биты, определяющие режим работы последовательного порта, но и девятый бит принимаемых или передаваемых данных (RB8 и TB8) и биты прерывания приемопередатчика (RI и TI). Функциональное назначение битов регистра управления/статуса УАПП приводится в табл. 6.6.

Прикладная программа путем загрузки в старшие биты специального регистра SCON 2-битного кода определяет режим работы

Таблица 6.6. Регистр управления/статуса УАПП **SCON**

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

Примечание

SM0	SM1	Режим работы УАПП
0	0	Сдвигающий регистр расширения ввода-вывода
0	1	УАПП-8 бит. Изменяемая скорость передачи
1	0	УАПП-9 бит. Фиксированная скорость передачи
1	1	УАПП-9 бит. Изменяемая скорость передачи

УАПП. Во всех четырех режимах работы передача из УАПП инициируется любой командой, в которой буферный регистр SBUF указан как получатель байта. Прием в УАПП в режиме 0 осуществляется при условии, что RI = 0 и REN = 1. В режимах 1, 2, 3 прием начинается с приходом старт-бита, если REN = 1.

В бите TB8 программно устанавливается значение девятого бита данных, который будет передан в режиме 2 или 3. В бите RB8 фиксируется в режимах 2 и 3 девятый принимаемый бит данных. В режиме

1, если SM2 = 0, в бит RB8 заносится стоп-бит. В режиме 0 бит RB8 не используется.

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

6.3.2. Работа УАПП в мультимикроконтроллерных системах

В системах децентрализованного управления, которые используются для управления и регулирования в топологически распределенных объектах (например, прокатных станах, электроподвижном составе железных дорог и метрополитена, сборочных конвейерах и линиях гибких автоматизированных производств), возникает задача обмена информацией между множеством микроконтроллеров, объединенных в локальную вычислительно-управляющую сеть. Как правило, локальные сети на основе MCS51 имеют магистральную архитектуру с разделяемым моноканалом (коаксиальный кабель, витая пара, оптическое волокно), по которому осуществляется обмен информацией между микроконтроллерами.

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

Механизм межконтроллерного обмена информацией через последовательный порт MCS51 построен на том, что в режимах 2 и 3 программируемый девятый бит данных при приеме фиксируется в бите RB8. УАПП может быть запрограммирован таким образом, что при получении стоп-бита прерывание от приемника будет возможно только при условии RB8 = 1. Это выполняется установкой управляющего бита SM2 в регистре SCON.

Поясним процесс межконтроллерного обмена информацией на примере. Пусть ведущему МК требуется передать блок данных некоторому (или нескольким) ведомому МК. С этой целью ведущий МК в протокольном режиме *широковещательной* передачи (всем ведомым МК) выдает в моноканал байт-идентификатор абонента (код адреса МК-получателя), который отличается от байтов данных только тем, что в его девятом бите содержится 1. Программа реализации протокола сетевого обмена информацией должна быть построена таким образом, чтобы при получении байта-идентификатора (RB8 = 1) во всех

ведомых МК произошли прерывания прикладных программ и вызов подпрограмм сравнения байта-идентификатора с кодом собственного сетевого адреса. Адресуемый МК сбрасывает свой управляющий бит SM2 и готовится к приему блока данных. Остальные ведомые МК, адрес которых не совпал с кодом байта-идентификатора, оставляют неизменным состояние SM2 = 1 и передают управление основной программе. При SM2 = 1 информационные байты, передаваемые по моноканалу и поступающие в УАПП ведомых МК, прерывания не вызывают, т.е. игнорируются.

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

6.3.3. Скорость приема/передачи

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

В режиме 0 частота передачи зависит только от резонансной частоты f_q кварцевого резонатора $f_0 = f_q/12$. За один машинный цикл последовательный порт передает один бит информации.

В режимах 1, 2 и 3 скорость приема/передачи зависит от значения управляющего бита SMOD в регистре специальных функций PCON (табл. 6.7).

В режиме 2 частота передачи $f_2 = 2^{SMOD} f_q/64$. Легко видеть, что при SMOD = 0 частота передачи равна $(f_q/64)$, а при SMOD = 1: $(f_q/32)$.

В режимах 1 и 3 в формировании частоты передачи кроме управляющего бита SMOD принимает участие таймер 1. При этом частота передачи зависит от частоты переполнения (OVT1) и определяется следующим образом: $f_{1,3} = 2^{SMOD} f_{OVT1}/32$. Прерывание от таймера 1 в этом случае должно быть заблокировано. Сам T/C1 может работать и как таймер, и как счетчик событий в любом из трех режимов. Однако наиболее удобно использовать режим таймера с автоперезагрузкой (старшая тетрада TMOD = 0010B). При этом частота передачи определяется выражением $f_{1,3} = (2^{SMOD}/32)(f_q/12)(256 - (TH1))$. В табл. 6.8 приводится описание способов настройки T/C1 для получения типовых частот передачи данных через УАПП.

6.3.4. Особенности работы УАПП в различных режимах

Режим 0. На рис. 6.11 показаны упрощенная структурная схема УАПП и временная диаграмма его работы в режиме 0. Данные пере-

Таблица 6.7. Регистр управления мощностью **PCON**

Символ	Позиция	Наименование и функция
SMOD	PCON.7	Удвоенная скорость передачи. Если бит установлен в 1, то скорость передачи вдвое больше, чем при SMOD = 0
–	PCON.6	Не используются
–	PCON.5	Не используются
–	PCON.4	Не используются
GF1	PCON.3	Флаг, специфицируемый пользователем (флаг общего назначения)
GF0	PCON.2	Флаг, специфицируемый пользователем (флаг общего назначения)
PD	PCON.1	Бит пониженной мощности. При установке бита в 1 МК переходит в режим пониженной потребляемой мощности
IDL	PCON.0	Бит холостого хода. Если бит установлен в 1, то МК переходит в режим холостого хода

Примечание. При одновременной записи 1 в PD и IDL бит PD имеет преимущество. Сброс содержимого PCON выполняется путем загрузки в него кода 0XXX0000.

даются и принимаются через вывод RxD. Через вывод TxD выдаются синхросигналы сдвига.

Передача начинается любой командой, по которой в SBUF поступает байт данных. В момент времени S6P2 устройство управления MCS51 по сигналу *Запись в буфер* записывает байт в сдвигающий регистр передатчика, устанавливает триггер девятого бита и запускает блок управления передачей, который через один машинный цикл вырабатывает разрешающий сигнал *Посылка*. При этом в момент S6P2 каждого машинного цикла содержимое сдвигающего регистра сдвигается вправо (младшими битами вперед) и поступает на вывод RxD. В освобождающиеся старшие биты сдвигающего регистра передатчика записываются нули. При получении от детектора нуля сигнала *Передатчик пуст* блок управления передатчиком снимает сигнал *Посылка* и устанавливает флаг TI (момент S1P1 десятого машинного цикла после поступления сигнала *Запись в буфер*).

Прием начинается при условии REN = 1 и RI = 0. В момент S6P2 следующего машинного цикла блок управления приемником формирует разрешающий сигнал *Прием*, по которому на выход TxD передаются синхросигналы сдвига и в сдвигающем регистре приемника начи-

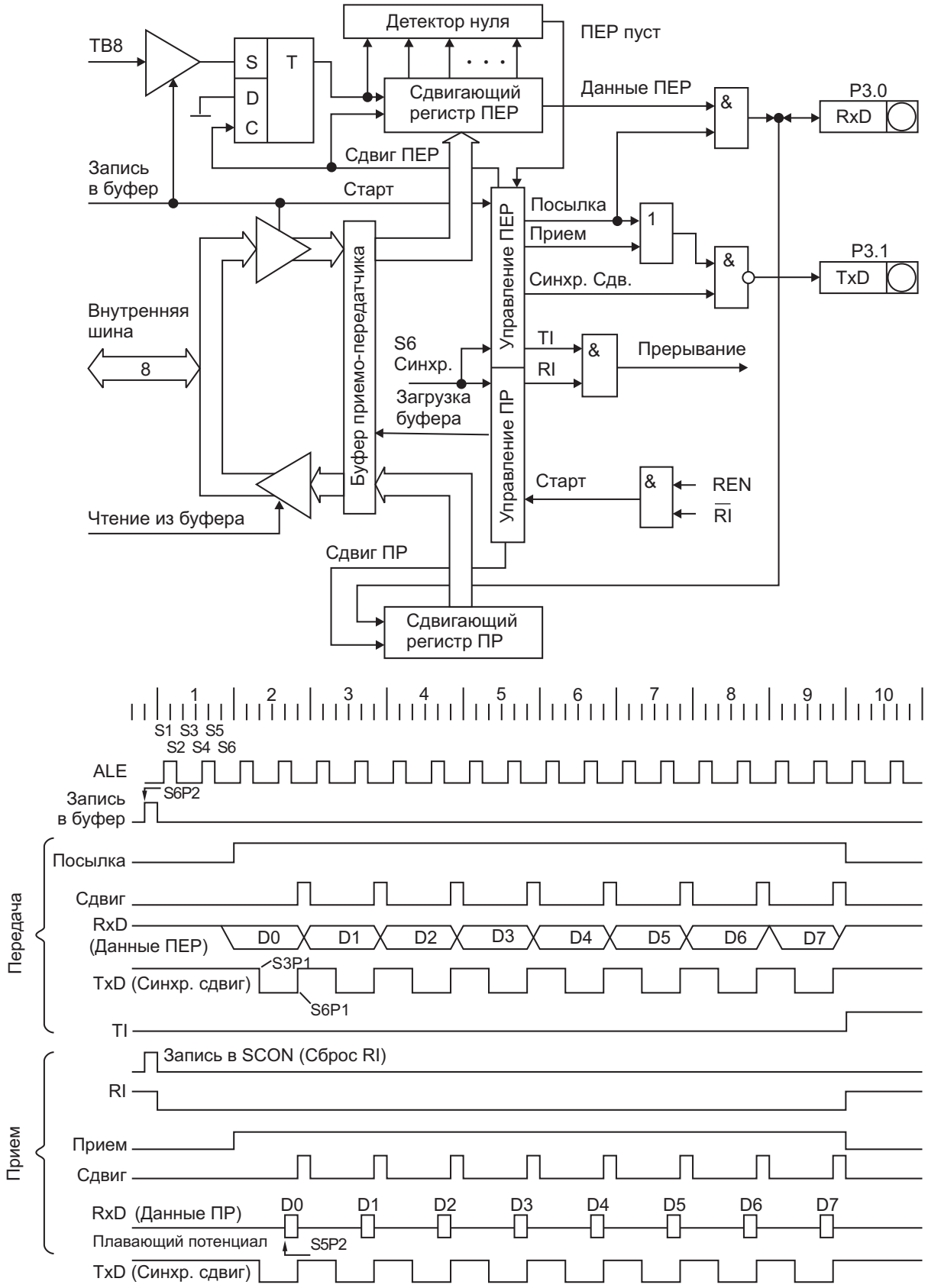


Рис. 6.11. УАПП в режиме 0: вверху – структурная схема; внизу – временные диаграммы работы

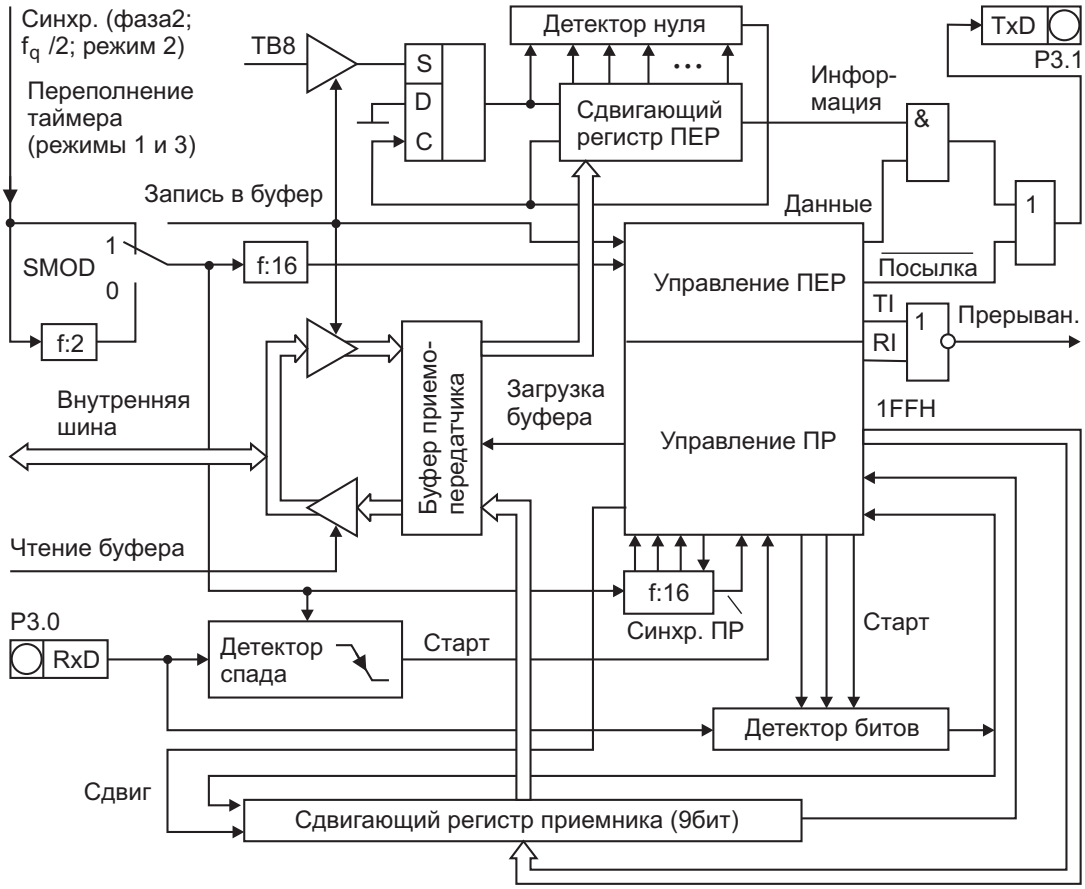


Рис. 6.12. Структурная схема УАПП в режимах 1, 2 и 3

Таблица 6.8. Настройка таймера 1 для управления частотой работы УАПП

Частота приема/передачи (Baud Rate) в режимах	Частота резона-тора, МГц	SMOD	Таймер/счетчик 1		
			C/T	Режим (MODE)	Перезаг-ружаемое число
0 (макс): 1 МГц	12	×	×	×	×
2 (макс): 375 кГц	12	1	×	×	×
1, 3: 62,5 кГц	12	1	0	2	0FFH
19.2 кГц	11.059	1	0	2	0FDH
9.6 кГц	11.059	0	0	2	0FDH
4.8 кГц	11.059	0	0	2	0FAH
2.4 кГц	11.059	0	0	2	0F4H
1.2 кГц	11.059	0	0	2	0E8H
137.5 Гц	11.059	0	0	2	1DH
110 Гц	6	0	0	2	72H
110 Гц	12	0	0	1	0FEEBH

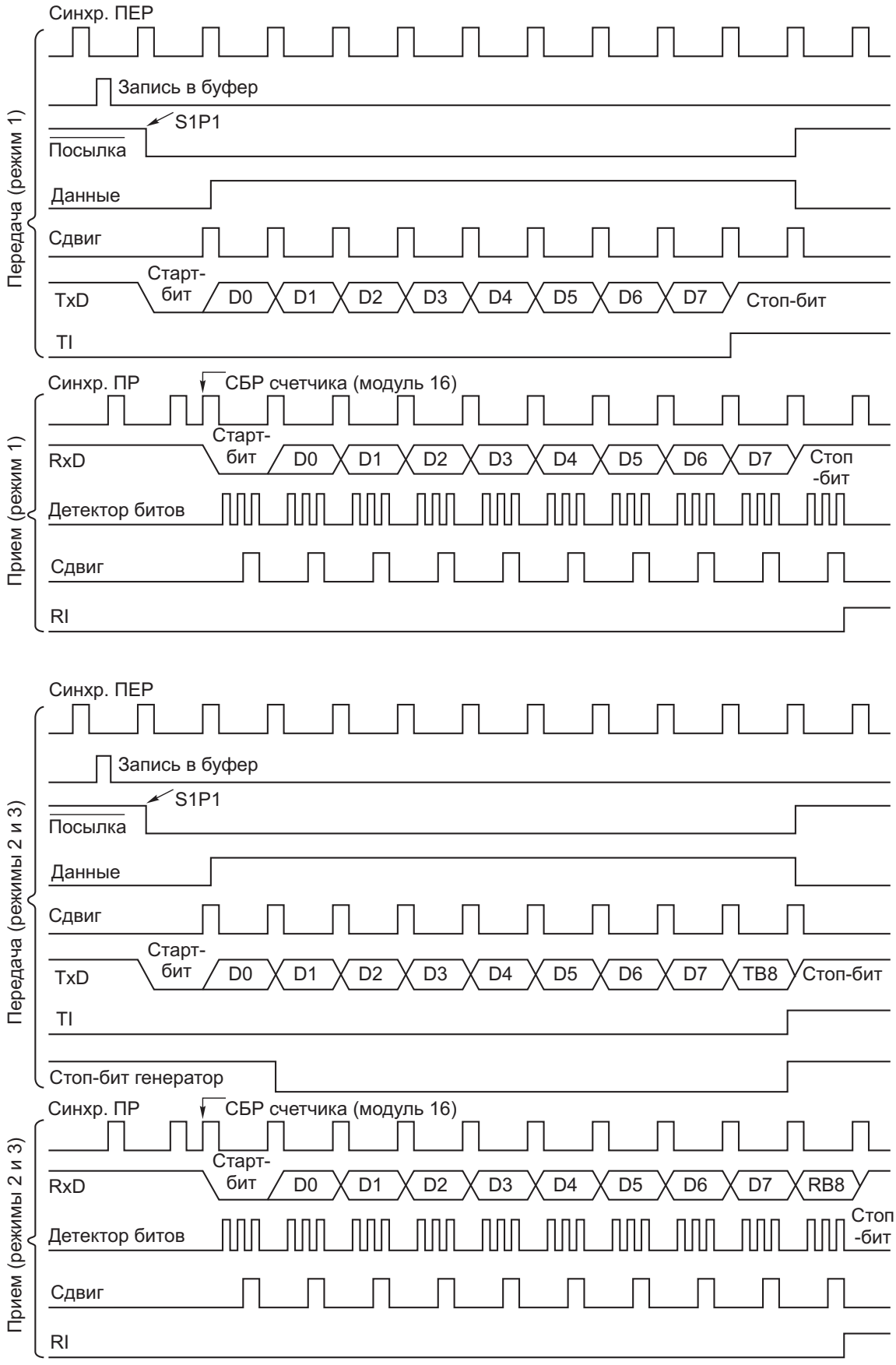


Рис. 6.13. Временные диаграммы работы УАПП для режимов 1, 2 и 3

нают формироваться значения битов данных, которые считываются с входа RxD в моменты S5P2 каждого машинного цикла. В момент S1P1 десятого машинного цикла после сигнала *Запись* в SCON блок управления приемником переписывает содержимое сдвигающего регистра в буфер, снимает разрешающий сигнал *Прием* и устанавливает флаг RI.

Режим 1. На рис. 6.12, 6.13 приведены структурная схема и временные диаграммы работы УАПП при приеме и передаче данных. Через вывод TxD УАПП передает, а с вывода RxD принимает 10 бит: старт-бит (0), 8 бит данных и стоп-бит (1). При приеме стоп-бит поступает в бит RB8 регистра SCON.

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

Прием начинается при обнаружении на входе RxD перехода сигнала из состояния 1 в состояние 0. Для этого под управлением внутреннего счетчика вход RxD опрашивается 16 раз за период представления бита. Как только переход из 1 в 0 на входе RxD обнаружен, в сдвигающий регистр приемника загружается код 1FFH, внутренний счетчик по модулю 16 немедленно сбрасывается и перезапускается для выравнивания его переходов с границами периодов представления принимаемых битов. Таким образом, каждый период представления бита делится на 16 периодов внутреннего счетчика. В состояниях 7, 8 и 9 счетчика в каждом периоде представления бита производится опрос сигнала на входе RxD. Считанное значение принимаемого бита – это то, которое было получено, по меньшей мере, дважды из трех замеров (мажоритарное голосование по принципу *два из трех*). Если значение, принятое в первом такте, не равно 0, то блок управления приемом вновь возвращается к поиску перехода из 1 в 0. Этот механизм обеспечивает подавление ложных (сбойных) старт-битов. Истинный старт-бит сдвигается в регистре приемника, и продолжается прием остальных бит посылки. Блок управления приемом сформирует сигнал *Загрузка буфера*, установит RB8 и флаг RI только в том случае, если в последнем такте сдвига выполняются два условия: бит RI = 0, и либо SM2 = 0, либо принятый стоп-бит *равен 1*. Если одно из этих двух условий не выполняется, то принятая последовательность бит теряется. В это время вне зависимости от того, выполняются указанные

условия или нет, блок управления приемом вновь начинает отыскивать переход из 1 в 0 на входе RxD.

Режимы 2, 3. Через вывод TxD УАПП передает или с вывода RxD принимает 11 бит: старт-бит (0), 8 бит данных, программируемый девятый бит и стоп-бит (1). На временной диаграмме (рис. 6.13) показана работа УАПП при передаче и приеме данных в режимах 2 и 3. Как видно, режимы 2 и 3 отличаются от режима 1 только наличием девятого программируемого бита. Вследствие этого несколько изменяются условия окончания цикла приема: блок управления приемником сформирует управляющий сигнал *Загрузка буфера*, загрузит RB8 и установит флаг RI только в том случае, если в последнем такте сдвига выполняются два условия: бит RI = 0 и либо SM2 = 0, либо значение принятого девятого бита данных равно 1.

6.4. Система прерываний

Упрощенная схема прерываний MCS51 показана на рис. 6.14. Внешние прерывания $\overline{\text{INT0}}$ и $\overline{\text{INT1}}$ могут быть вызваны либо уровнем, либо переходом сигнала из 1 в 0 на входах MCS51 в зависимости от значений управляющих бит IT0 и IT1 в регистре TCON. От внешних прерываний устанавливаются флаги IE0 и IE1 в регистре TCON, которые инициируют вызов соответствующей подпрограммы обслуживания прерывания. Сброс этих флагов выполняется аппаратурно только в том случае, если прерывание было вызвано по переходу (срезу) сигнала. Если же прерывание вызвано уровнем входного сигнала, то сбросом флага IE управляет соответствующая подпрограмма обслуживания прерывания путем воздействия на источник прерывания в целях снятия им запроса.

Флаги запросов прерывания от таймеров TF0 и TF1 сбрасываются автоматически при передаче управления подпрограмме обслуживания. Флаги запросов прерывания RI и TI устанавливаются блоком управления УАПП аппаратурно, но сбрасываться должны программой.

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

В блоке регистров специальных функций есть два регистра, предназначенных для управления режимом прерываний и уровнями приоритета. Форматы этих регистров, имеющих символические имена IE и IP, описаны в табл. 6.9 и 6.10 соответственно. Возможность программной установки/сброса любого управляющего бита в этих двух

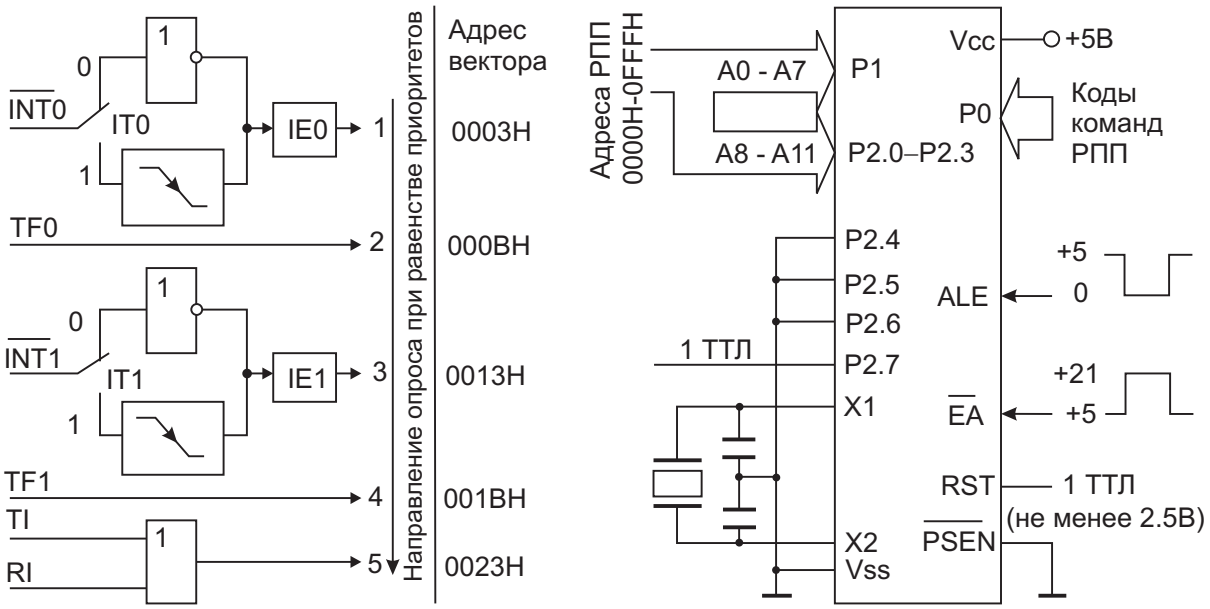


Рис. 6.14. Микроконтроллер MCS51: схема прерываний (слева) и схема программирования (справа)

регистрах делает систему прерываний MCS51 исключительно гибкой.

Флаги прерываний опрашиваются в момент S5P2 каждого машинного цикла. Ранжирование прерываний по уровню приоритета выполняется в течение следующего машинного цикла. Система прерываний сформирует аппаратурно вызов (LCALL) соответствующей подпрограммы обслуживания, если она не заблокирована одним из следующих условий:

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

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

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

Отметим, что если флаг прерывания был установлен, но по одному из перечисленных выше условий не получил обслуживания и к моменту окончания блокировки уже был сброшен, то запрос прерывания теряется и нигде не запоминается. По аппаратурно-сформированному коду LCALL система прерывания помещает в стек только содержимое счетчика команд (PC) и загружает в счетчик команд адрес вектора соответствующей подпрограммы обслуживания. По адресу вектора должна быть расположена команда безусловной передачи управления (JMP) к начальному адресу подпрограммы обслуживания прерывания. Подпрограмма обслуживания в случае необходимости должна начинаться командами записи в стек (PUSH) слова состояния программы (PSW), аккумулятора, расширителя, указателя данных и т.д.

Таблица 6.9. Регистр масок прерывания **IE**

Символ	Позиция	Имя и назначение
EA	IE.7	Снятие блокировки прерываний. Сбрасывается программно для запрета всех прерываний независимо от состояний IE4 – IE0
–	IE.6	Не используются
–	IE.5	Не используются
ES	IE.4	Бит разрешения прерывания от УАПП. Установка/сброс программой для разрешения/запрета прерываний от флагов T1 или RI
ET1	IE.3	Бит разрешения прерывания от таймера 1. Установка/сброс программой для разрешения/запрета прерываний от таймера 1
EX1	IE.2	Бит разрешения внешнего прерывания 1. Установка/сброс программой для разрешения/запрета прерываний
ET0	IE.1	Бит разрешения прерывания от таймера 0. Работает аналогично IE.3
EX0	IE.0	Бит разрешения внешнего прерывания 0, Работает аналогично IE.2

и заканчиваться командами восстановления из стека (POP). Подпрограммы обслуживания прерывания обязательно завершаются командой RETI, по которой в счетчик команд перезагружается из стека сохраненный адрес возврата в основную программу. Команда RET также возвращает управление прерванной основной программе, но при этом не снимает блокировку прерываний, что приводит к необходимости иметь программный механизм анализа окончания процедуры обслуживания данного прерывания.

6.5. Особые режимы работы MCS51

6.5.1. Режим загрузки и верификации прикладных программ

Под воздействием внешних электрических сигналов MCS51 может быть электрически запрограммирован или, иными словами, в РПП микроконтроллера могут быть загружены объектные коды прикладной программы. Содержимое РПП микроконтроллера может быть уничтожено выдержкой под ультрафиолетовым источником света (стирание) для последующего перепрограммирования. Микроконтроллер имеет средство защиты от программного *разбоя*, обеспечивающее не-

Таблица 6.10. Регистр приоритетов прерываний **IP**

Символ	Позиция	Имя и назначение
–	IP.7–IP.5	Не используются
PS	IP.4	Бит приоритета УАПП. Установка/сброс программой для присваивания прерыванию от УАПП высшего/низшего приоритета
PT1	IP.3	Бит приоритета таймера 1. Установка/сброс программой для присваивания прерыванию от таймера 1 высшего/низшего приоритета
PX1	IP.2	Бит приоритета внешнего прерывания 1. Установка/сброс программой для присваивания высшего/низшего приоритета внешнему прерыванию $\overline{\text{INT1}}$
PT0	IP.1	Бит приоритета таймера 0. Работает аналогично IP.3
PX0	IP.0	Бит приоритета внешнего прерывания 0. Работает аналогично IP.2

возможность прочтения содержимого РПП в конечном изделии и, следовательно, сохранение профессиональных секретов разработчика прикладного программного обеспечения.

Загрузка программ в РПП. В режиме программирования MCS51 должен работать на пониженной частоте (с резонатором 4…6 МГц) из-за необходимости мультиплексирования на внутренней шине адресной и кодовой информации. На рис. 6.14 приведена схема подключения MCS51 к программатору. Адрес ячейки РПП, в которую должен быть загружен байт прикладной программы, подается на выводы порта 1 и выводы P2.0…P2.3 порта 2. При этом загружаемый байт поступает в МК через порт 0. Выводы P2.4…P2.6 и $\overline{\text{PSEN}}$ должны быть заземлены, а на выводы P2.7 и RST необходимо подать уровень логической 1 (для входа RST уровень логической 1 – не менее 2.5 В, для остальных входов – стандартный уровень ТТЛ). На входе $\overline{\text{EA}}$/Vpp поддерживается уровень +5 В, но в момент загрузки байта он должен быть подключен к источнику напряжения с уровнем +21 В. В это время уровень на входе ALE/$\overline{\text{PROG}}$ должен быть не менее чем на 50 мс сброшен в 0. После этого напряжение на входе $\overline{\text{EA}}$/Vpp возвращается к уровню +5 В. Источник напряжения +21 В (Vpp) должен быть очень хорошо стабилизирован, так как превышение предельного значения +21.5 В на входе $\overline{\text{EA}}$/Vpp приводит к необратимым повреждениям РПП.

Запись бита защиты. Бит защиты РПП, будучи установлен, запрещает доступ к РПП любыми внешними средствами. Схема записи бита защиты показана на рис. 6.15. Процедура записи бита защиты такая же, как и при загрузке программ в РПП, но на вывод P2.6 должен подаваться уровень 1. Сигналы на выводах портов P0, P1 и P2.0...P2.3 могут быть в любом состоянии. Однажды установленный бит защиты можно сбросить только путем полного стирания РПП под источником УФ-излучения.

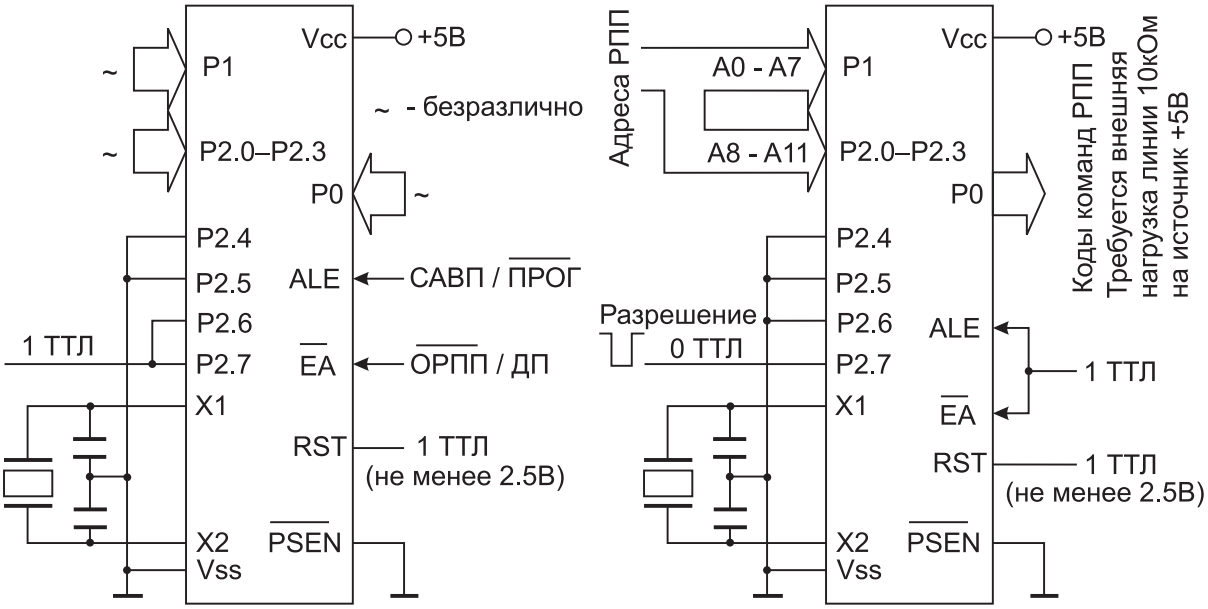


Рис. 6.15. Микроконтроллер MCS51: схема записи бита защиты (слева) и схема верификации программ (справа)

Верификация программ. Если бит защиты не запрограммирован, то содержимое РПП может быть прочитано в целях проверки правильности загрузки прикладной программы либо по ходу программирования, либо после окончания программирования MCS51. Доступ к ячейкам РПП (рис. 6.15) осуществляется так же, как и при программировании РПП, за исключением того, что на вывод P2.7 подается сигнал 0, используемый в качестве строб-сигнала чтения. Считанные коды команд и соответствующие им адреса РПП могут отображаться для визуального контроля на любом внешнем индикаторе и, кроме того, могут быть переданы в ОЗУ инструментальной ЭВМ для дизассемблирования.

Стирание РПП. Для стирания содержимого РПП микроконтроллер следует поместить под источник УФ-излучения с длиной волны менее 400 нм. Если кварцевая лампа имеет световую мощность

12 мВт/см^2, а расстояние между источником света и МК равно 2…3 см, то выдержка 20…30 мин обеспечивает световую дозу, достаточную для надежного стирания РПП. После стирания в матрице РПП содержатся все единицы.

Поскольку в спектре солнечного света и люминесцентного освещения содержится излучение с длиной волны менее 400 нм, то пребывание МК под воздействием этих источников света дольше установленного предела (около одной недели на солнечном свете или около трех лет при люминесцентном освещении) может привести к искажению содержимого РПП. Для защиты содержимого РПП от разрушения под воздействием света на окне корпуса МК укрепляют светонепроницаемый экран.

6.5.2. Работа MCS51 в пошаговом режиме

На этапе отладки прикладной программы в прототипе МК-системы очень удобным для разработчика оказывается режим пошагового (покомандного) исполнения программы. Система прерываний MCS51 позволяет реализовать этот режим работы путем использования нескольких дополнительных команд. Как было описано ранее (см. п. 6.4), внешний запрос прерывания не будет обслужен до тех пор, пока обслуживается прерывание с равным приоритетом. Этот запрос будет воспринят лишь после того, как одна команда после команды RETI будет выполнена. Иными словами, однажды вызвав подпрограмму обслуживания прерывания, вызвать ее вновь невозможно до тех пор, пока хотя бы одна команда основной программы не будет исполнена. Использовать это свойство системы прерываний MCS51 для реализации пошагового режима можно следующим образом: одно из внешних прерываний, например $\overline{\text{INT0}}$ (вывод P3.2 микроконтроллера), запрограммировать для представления запроса уровнем сигнала (T0 = 0), а подпрограмма обслуживания этого прерывания должна заканчиваться последовательностью команд

```
JNB   P3.2,$    ; Ожидание уровня 1 на входе INT0
JB    P3.2,$    ; Ожидание уровня 0 на входе INT0
RETI            ; Возврат и выполнение 1 команды
```

Здесь символом $ обозначено текущее содержимое счетчика команд. Теперь если на вывод $\overline{\text{INT0}}$ подавать сигнал от клавиши ШАГ отладочного модуля системы, то MCS51 по сигналу $\overline{\text{INT0}}$ = 0 вызовет подпрограмму обслуживания внешнего прерывания 0 и будет в ней находиться до тех пор, пока не обнаружит на входе $\overline{\text{INT0}}$ новый импульс (переход из 0 в 1 и снова в 0). По команде RETI произойдет

возврат в основную программу, выполнение в ней одной команды и немедленный вызов подпрограммы обслуживания внешнего прерывания 0.

6.5.3. Сброс, режим холостого хода и режим пониженного энергопотребления

Сброс в MCS51 осуществляется путем подачи на вход RST сигнала 1. Для уверенного сброса MCS51 этот сигнал 1 должен быть удержан на входе RST, по меньшей мере, в течение двух машинных циклов (24 периода резонатора). Квазидвунаправленные буферные схемы внешних выводов ALE и $\overline{\text{PSEN}}$ находятся при этом в режиме ввода. Под воздействием сигнала RST сбрасывается содержимое регистров: PC, ACC, B, PSW, DPTR, TMOD, TCON, T/C0, T/C1, IE, IP и SCON, в регистре PCON сбрасывается только старший бит, в регистр-указатель стека загружается код 07H, а в порты P0…P3 – коды 0FFH. Состояние регистра SBUF неопределенное. Сигнал RST не воздействует на содержимое ячеек РПД. Когда включается электропитание (Vcc), содержимое РПД не определено, за исключением операции возврата из режима пониженного энергопотребления.

На рис. 6.16 показана схема автоматического формирования сигнала RST при включении электропитания. Время, необходимое для полного заряда емкости, обеспечивает уверенный запуск резонатора и его работу в течение более чем двух машинных циклов.

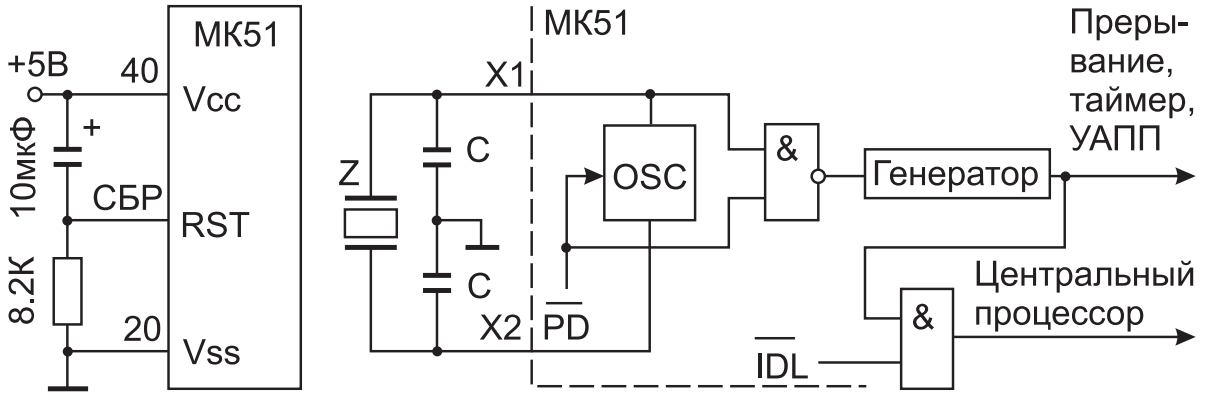


Рис. 6.16. Микроконтроллер MCS51: схема сброса при включении электропитания (слева) и схема управления от сигналов IDL и PD (справа)

Режим холостого хода. Любая команда, устанавливающая управляющий бит IDL (PCON.0) в регистре управления мощностью, переведет MCS51 в режим холостого хода. При этом внутренний генератор синхросигналов (см. рис. 6.16) продолжает работать. Все регистры

сохраняют свое значение. На выводах всех портов удерживается то логическое состояние, которое на них было в момент перехода в режим холостого хода. На выводах ALE и $\overline{\text{PSEN}}$ формируется уровень логической единицы.

Выйти из режима холостого хода можно или по сигналу СБР, или по прерыванию. Любой из разрешенных сигналов прерывания приведет к аппаратурному сбросу бита PCON.0 и прекратит тем самым режим холостого хода. После исполнения команды RETI (выход из подпрограммы обслуживания прерывания) будет исполнена команда, которая следует в программе за командой, переведшей MCS51 в режим холостого хода. В некоторых модификациях МК51 этот режим может отсутствовать.

Режим пониженного энергопотребления. В тех применениях МК-систем, где потребление электроэнергии, а следовательно, габаритные размеры и масса источника электропитания являются одними из основных показателей качества изделия, возможно использование MCS51 в режиме пониженного энергопотребления. Перевод MCS51 в этот режим возможен по команде, которая установит бит PCON.1 в регистре управления мощностью (см. табл. 6.7). В этом режиме останавливается генератор синхросигналов, содержимое РПД и регистров специальных функций сохраняется, а на выходных контактах портов удерживаются значения, соответствующие содержимому их буферных регистров. Выходы сигналов ALE и $\overline{\text{PSEN}}$ сбрасываются. При этом электропитание осуществляется через вывод RST/VPD.

В режиме пониженного энергопотребления напряжение электропитания Vcc может быть отключено. Перед выходом из режима оно должно быть восстановлено до номинального значения. Выход из режима пониженного энергопотребления возможен только по сигналу RST. При этом переопределяются все регистры специальных функций, но содержимое РПД не изменяется. В некоторых модификациях MCS51 этот режим может отсутствовать.

Защита от падения напряжения. При немгновенных отказах блока электропитания МК-системы можно обеспечить сохранность содержимого РПД с помощью маломощного (батарейного) аварийного источника электропитания, подсоединяемого к выводу RST/VPD. Для этого МК при получении сообщения о грозящем падении напряжения Vcc (прерывание от системы контроля электропитания) должен перегрузить в РПД основные параметры прерванного процесса и выдать сигнал, разрешающий подключение к выводу RST/VPD аварийного источника питания. Все эти процедуры МК должен успеть выполнить до того, как напряжение основного источника электропитания

упадет ниже рабочего предела. После восстановления номинального значения напряжения в основной цепи электропитания выполняется системный сброс, и источник аварийного электропитания может быть отключен.

6.5.4. Локальная управляющая сеть на основе MCS51

При управлении сложными протяженными объектами используются системы с распределенным управлением, состоящие из множества контроллеров, управляющих отдельными агрегатами объекта (например, силовой установкой корабля или локомотива, прокатным станом и т.п.). Между отдельными подсистемами должна обеспечиваться информационная связь. Поэтому обязательным элементом распределенной системы управления является локальная сеть, объединяющая отдельные контроллеры в систему.

На такую сеть возлагаются в основном простые функции передачи сообщений за гарантированное время. Протяженность линий связи обычно не превышает десятков метров, размер сообщения – нескольких десятков байтов, а время доставки сообщения – в пределах от 0.01 до 1.0 с. Типичными являются два режима информационного обмена в сети: широковещательный, когда передаваемое сообщение предназначается для всех остальных подсистем (микроконтроллеров сети), и абонентский, когда сообщение предназначается только для одного МК. Обычно первым способом передаются различные информационные параметры, используемые многими подсистемами. Это позволяет уменьшить загрузку сети за счет исключения множественных передач одного и того же сообщения различным адресатам. Вторым способом обычно передаются команды управления от центрального устройства к исполнительным или сообщения экстренного характера.

Наиболее широко распространены локальные сети двух структур: кольцевые и моноканальные (типа BITBUS). Последние, на наш взгляд, являются более удобными для рассматриваемого класса управляющих сетей, так как допускают простую наращиваемость и модифицируемость системы. Кроме того, в моноканальной сети время доставки сообщения не зависит от общего числа МК, и они обладают большей живучестью и надежностью.

В сетях с единым моноканалом все МК связаны между собой одной общей (разделяемой) линией связи (рис. 6.17). В качестве линии передачи данных обычно используется коаксиальный кабель или витая пара с согласующими резисторами на концах.

Существует несколько известных методов доступа к разделяемой линии (протоколов), позволяющих осуществлять обмен данными между многими МК сети, т.е. обеспечивающих разделение канала связи между многими подсистемами.

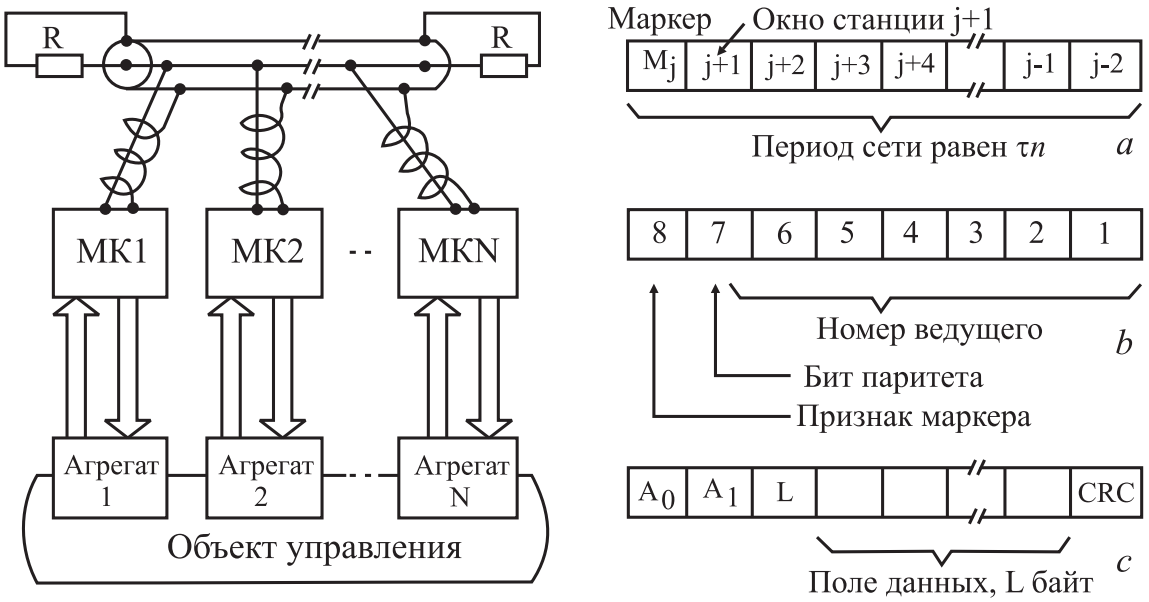


Рис. 6.17. Структура локальной управляющей сети на основе моноканала (слева) и интегральный маркерный доступ к моноканалу (справа): *a* – диаграмма одного периода сети; *b* – маркер; *c* – формат сообщения

Наиболее перспективным методом доступа к разделяемому моноканалу, на наш взгляд, является интервально-маркерный метод, позволяющий устранить конфликты в канале и наиболее полно использовать пропускную способность канала. Вкратце этот метод сводится к следующему:

1. При нулевой загрузке в канале периодически появляется маркер, генерируемый одним из МК сети. Маркер содержит номер МК, являющегося ведущим. Главная обязанность ведущего – поддерживать синхронизм в сети путем периодической выдачи маркера в канал.

2. Период генерации маркера состоит из определенного числа *окон*, равного числу МК в сети. Каждое окно имеет свой номер и принадлежит одной из подсистем.

3. В процессе захвата канала МК, желающий выдать свой пакет (сообщение), должен дождаться появления маркера и отсчитать от него свое окно. Если при этом его не опередят другие МК, то, дождавшись своего окна, подсистема может, не опасаясь конфликтов, начинать передачу данных (рис. 6.17, *a*).

4. После выдачи сообщения МК генерирует свой маркер и становится новым ведущим. Старый ведущий сети, распознав, что моноканал захвачен, освобождается от этой роли.

5. Отсчет момента времени от маркера до своего окна производится по следующему правилу. Длительность окна принимается равной времени передачи одного байта данных. Если ведущий МК имел номер l, то первое окно будет принадлежать МК с номером $l + 1$, затем

МК с номером $l+2$ и т.д. Время ожидания своего окна (T) можно определить как $T = \tau \cdot X$, где τ – время передачи одного байта, т.е. длительность окна. Число X определяется следующим образом:

$$X = \begin{cases} k-l-1, & \text{если } k > l; \\ k-l+1, & \text{если } k < l, \end{cases}$$

где k – номер МК, пытающегося захватить канал, $k = 0 \div (n-1)$;
l – номер ведущего, $l = 0 \div (n-1)$; n – число МК в сети.

6. Если самому ведущему необходимо выдать сообщение, то он может захватить канал во время своего окна, т.е. вместо генерации очередного маркера начать передавать свое сообщение.

7. Выдав маркер, ведущий МК запускает таймер на время $\tau \cdot (n-1)$; если за это время никто не захватит канал, то весь цикл повторяется еще раз и т.д.

8. Каждый МК принимает все байты, передаваемые по каналу. Для контроля пропажи маркера каждый МК после приема каждого байта запускает таймер на задержку $\tau \cdot (n+1)$. Таким образом, пропажа маркера (а следовательно, и синхронизма сети) фиксируется, если за время $\tau \cdot (n+1)$ не было передано ни одного байта.

9. При обнаружении пропажи маркера для восстановления синхронизма в сети каждый МК выполняет следующие простые действия: выдерживает паузу длительностью $\tau \cdot (i+1)$, где i – собственный номер МК; если во время паузы вновь не было принято никакой информации, то данный МК становится ведущим и генерирует новый маркер.

Этой процедурой обеспечивается автоматическое восстановление работы сети при отказе МК, являющегося в данный момент ведущим.

10. При интервально-маркерном методе удается избежать любых конфликтов в канале в силу следующих причин:

- контроль пропажи маркера ведется постоянно всеми МК и полностью синхронно, так как счетчики паузы корректируются примерно одновременно при приеме каждого байта и, следовательно, все МК обнаружат пропажу маркера одновременно;
- одновременно начинается отсчет паузы $\tau \cdot (i+1)$ всеми МК;
- микроконтроллер с меньшим номером первым генерирует маркер и восстановит синхронизм в сети.

Последовательный порт MCS51 допускает передачу 9-битных кодов. Используя это, можно легко ввести признак маркера таким образом, что байт маркера будет отличаться от любого информационного байта. На рис. 6.17,*b* представлена структура маркера; старший бит является признаком маркера (для маркера 1). Бит 7 используется для простейшего контроля по паритету. Семибитное поле адреса позволяет иметь в системе до 127 подсистем с номерами от 0 до 126. Адрес 127 зарезервирован для широковещательной передачи.

Рекомендуемый формат сообщения представлен на рис. 6.17,*с* и предусматривает следующие поля: A_0 – адрес получателя; A_1 – адрес отправителя; L – длина поля данных (0-255); CRC – байт контрольной суммы.

Для реализации подсистемы требуются следующие ресурсы: УАПП, таймер, два уровня прерываний. Необходимым набором ресурсов располагает MCS51, позволяющий вести передачу и прием данных со скоростью до 375 Кбит/с. Время передачи одного байта, обрамленного стартовым и стоповым битами (плюс 9-й разряд), составляет 58.7 мкс. Пропускная способность сети при этом равна примерно 17 Кбайт/с.

Микроконтроллер, работающий в составе распределенной системы управления на основе локальной сети, должен, кроме прикладной программы управления, иметь еще и программные средства доступа к моноканалу. Таким образом, МК должен работать в двухпрограммном режиме с разделением всех ресурсов между этими двумя сопрограммами. Разумеется, должен быть реализован механизм взаимодействия между сетевой и прикладной программами. Чаще всего этот механизм реализуется путем присвоения сетевой программе более высокого приоритета.

6.6. Система команд MCS51

6.6.1. Общие сведения о системе команд

Система команд MCS51 содержит 111 базовых команд, которые удобно разделить по функциональному признаку на пять групп: команды передачи данных, арифметических операций, логических операций, передачи управления и операций с битами. В ее состав входят команды умножения, деления, вычитания, операций над битами, операций со стеком и расширенный набор команд передачи управления.

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

На рис. 6.18 показаны 13 типов команд MCS51. Первый байт команды любых типа и формата всегда содержит код операции (КОП). Второй и третий байты содержат либо адреса операндов, либо непосредственные операнды.

Типы операндов. Состав операндов MCS51 включает в себя операнды четырех типов: биты, 4-битные цифры, байты и 16-битные слова.

MCS51 имеет 128 программно-управляемых флагов пользователя, а также допускает адресацию отдельных битов блока РСФ и портов.

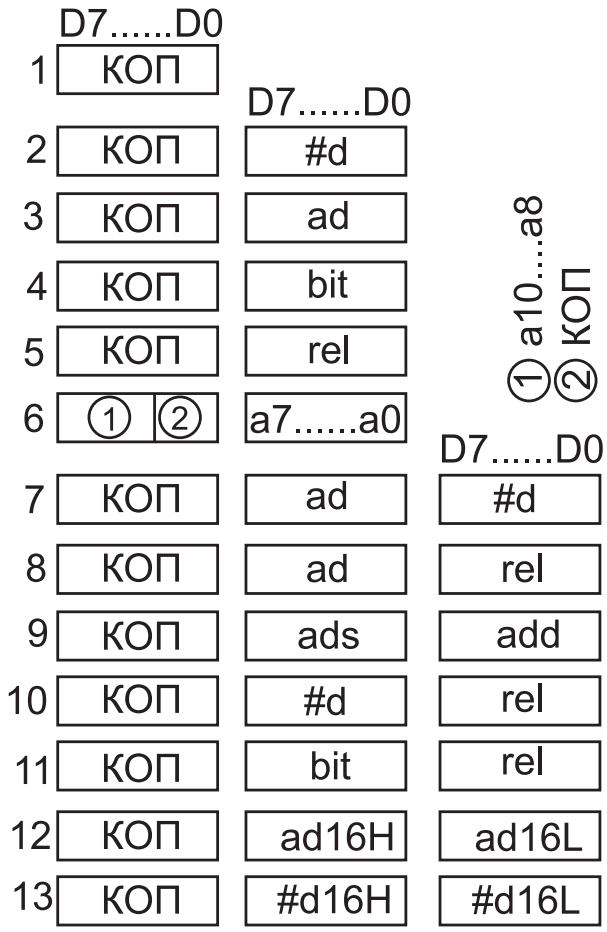


Рис. 6.18. Типы команд микроконтроллера MCS51

Для адресации бита используется прямой 8-битный адрес (bit). Косвенная адресация бит невозможна. Карты адресов отдельных битов представлены на рис. 6.19. Четырехбитные операнды используются только при операциях обмена (команды SWAP и XCHD). Восьмибитным операндом может быть ячейка памяти программ или данных (резидентной или внешней), константа (непосредственный операнд), РСФ, а также порты ввода-вывода. Порты и РСФ адресуются только прямым способом. Байты памяти могут адресоваться также и косвенным образом: через адресные регистры (R0, R1 DPTR и PC). Двухбайтные операнды – это константы и прямые адреса, для представления которых используются второй и третий байты команды.

Способы адресации данных. Система команд MCS51 допускает множество комбинаций способов адресации операндов в командах, что делает ее гибкой и универсальной. Набор команд MCS51 поддерживает следующие режимы адресации:

Прямая адресация. Операнд определяется 8-битовым адресом в инструкции. Прямая адресация используется только для младшей

РПД

	(D7)							(D0)
7FH								
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							
18H								
17H	Банк 2							
10H								
0FH	Банк 1							
08H								
07H	Банк 0							
00H								

РСФ

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

Рис. 6.19. Карты адресуемых битов в резидентной памяти данных – РПД и регистрах специальных функций – РСФ

половины внутренней памяти данных и регистров специальных функций (РСФ).

Косвенная адресация. Инструкция адресует регистр, содержащий адрес операнда во внешнем или внутреннем ОЗУ. Для указания 8-битовых адресов используют регистры R0 и R1 выбранного регистрового банка или указатель стека SP. Для 16-битовой адресации используется регистр указателя данных DPTR.

Регистровые инструкции. Регистры R0-R7 текущего регистрового банка могут быть адресованы через конкретные инструкции, содер-

жащие 3-битовое поле, указывающее номер регистра в самой инструкции. В этом случае сооответствующее поле адреса в команде отсутствует.

Операции с использованием специальных регистров. Некоторые инструкции используют индивидуальные регистры, например операции с аккумулятором, DPTR и т.д. В данном случае адрес операнда вообще не указывается в команде. Он предопределяется кодом операции.

Непосредственные константы. Константа может находиться прямо в команде за кодом операции.

Индексная адресация. Индексная адресация может использоваться только для доступа к программной памяти и только в режиме чтения. В этом режиме осуществляется просмотр таблиц в памяти программ. 16-битовый регистр (DPTR или программный счетчик) указывает базовый адрес требуемой таблицы, а аккумулятор указывает на точку входа в нее.

Символическая адресация. При использовании ассемблера ASM51 для получения объектных кодов программ допускается применение в программах символических имен регистров специальных функций, портов и их отдельных битов (см. рис. 6.19).

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

<имя РСФ или порта >.< номер бита >

Например, символическое имя пятого бита аккумулятора будет следующим: ACC.5. Символические имена РСФ, портов и их битов являются зарезервированными словами для ASM-51, и их не надо определять с помощью директив ассемблера.

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

Флаг паритета напрямую зависит от текущего значения аккумулятора. Если число единичных битов аккумулятора нечетное, то флаг P устанавливается, а если четное – сбрасывается. Все попытки изменить флаг P, присваивая ему новое значение, будут безуспешными, если содержимое аккумулятора при этом останется неизменным. Флаг AC устанавливается в случае, если при выполнении операции сложения/вычитания между тетрадами байта возник перенос/заем. Флаг C устанавливается, если в старшем бите результата возникает перенос или заем. При выполнении операций умножения и деления сбрасы-

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

В табл. 6.11 перечисляются команды, при выполнении которых модифицируются флаги результата. В таблице отсутствует флаг паритета, так как его значение изменяется всеми командами, которые изменяют содержимое аккумулятора. Кроме команд, приведенных в таблице, флаги модифицируются командами, в которых местом назначения результата определены PSW или его отдельные биты, а также командами операций над битами.

Таблица 6.11. Команды, модифицирующие флаги результата

Команды	Флаги	Команды	Флаги
ADD	C, OV, AC	CLR C	C=0
ADDC	C, OV, AC	CPL C	C=$\overline{C}$
SUBB	C, OV, AC	ANL C,b	C
MUL	C=0, OV	ANL C,/b	C
DIV	C=0, OV	ORL C,b	C
DA	C	ORL C,/b	C
RRC	C	MOV C,b	C
RLC	C	CJNE	C
SETB C	C=1		

Группа команд передачи данных

Большую часть команд данной группы (прил. 3) составляют команды передачи и обмена байтов. Команды пересылки бит представлены в группе команд битовых операций. Все команды данной группы не модифицируют флаги результата, за исключением команд загрузки PSW и аккумулятора (флаг паритета).

Структура информационных связей. В зависимости от способа адресации и места расположения операнда можно выделить девять типов операндов, между которыми возможен информационный обмен. Граф возможных операций передачи данных показан на рис. 6.20. Аккумулятор (A) представлен на этом графе отдельной вершиной, так как многие команды используют неявную (подразумеваемую) адресацию. Передачи данных в MCS51 могут выполняться без участия аккумулятора.

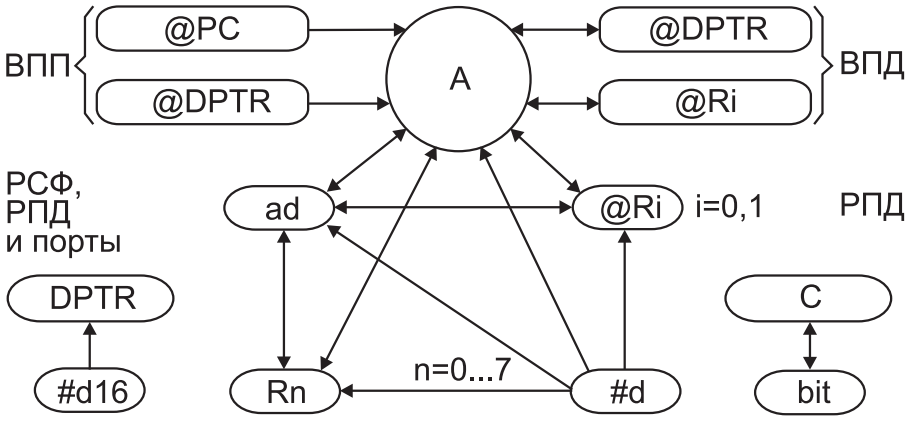


Рис. 6.20. Граф путей передачи данных в MCS51

Аккумулятор. Обращение к аккумулятору может быть выполнено с использованием неявной и прямой адресации. В зависимости от способа адресации аккумулятора применяется одно из символических имен: A или ACC (прямой адрес). При прямой адресации обращение к аккумулятору производится как к одному из РСФ, и его адрес указывается во втором байте команды.

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

Обращение к внешней памяти данных. Режим косвенной адресации ВПД также реализован в MCS51. При использовании команд MOVX @Ri обеспечивается доступ к 256 байтам внешней памяти данных.

Существует также режим обращения к расширенной ВПД, когда для доступа используется 16-битный адрес, хранящийся в регистре-указателе данных (DPTR). Команды MOVX @DPTR обеспечивают доступ к 65536 байтам ВПД.

Группа команд арифметических операций

Данную группу образуют 24 команды (см. прил. 3), выполняющие операции сложения, десятичной коррекции, инкремента/декремента байтов. Введены команды вычитания, умножения и деления байтов.

Команды ADD и ADDC допускают сложение аккумулятора с большим числом операндов. Аналогично командам ADDC существуют четыре команды SUBB, что позволяет производить вычитание байтов и многобайтных двоичных чисел. В MCS51 реализуется расширенный список команд инкремента/декремента байтов, введена команда инкремента 16-битного регистра-указателя данных.

Группа команд логических операций

Данную группу образуют 25 команд (см. прил. 3), реализующих те же логические операции над байтами, что и в МК48. Однако в MCS51 значительно расширено число типов операндов, участвующих в операциях. Имеется возможность производить операцию *исключающее ИЛИ* с содержимым портов. Команда XRL может быть эффективно использована для инверсии отдельных битов портов.

Группа команд операций с битами

Отличительной особенностью данной группы команд (см. прил. 3) является то, что они оперируют с однобитными операндами. В качестве таких операндов могут выступать отдельные биты некоторых регистров специальных функций (РСФ) и портов, а также 128 программных флагов пользователя.

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

Группа команд передачи управления

К данной группе команд (см. прил. 3) относятся команды, обеспечивающие условное и безусловное ветвление, вызов подпрограмм и возврат из них, а также команда пустой операции NOP. В большинстве команд используется прямая адресация, т.е. адрес перехода целиком (или его часть) содержится в самой команде передачи управления. Можно выделить три разновидности команд ветвления по разрядности указываемого адреса перехода.

Длинный переход. Переход по всему адресному пространству ПП. В команде содержится полный 16-битный адрес перехода (ad16). Трехбайтные команды длинного перехода содержат в мнемокоде букву L (Long). Всего существует две такие команды: LJMP – длинный переход и LCALL – длинный вызов подпрограммы. На практике редко возникает необходимость перехода в пределах всего адресного пространства и чаще используются укороченные команды перехода, занимающие меньше места в памяти.

Абсолютный переход. Переход в пределах одной страницы памяти программ размером 2048 байт. Такие команды содержат только 11

младших битов адреса перехода (ad11). Команды абсолютного перехода имеют формат 2 байта. Начальная буква мнемокода – A (Absolute). При выполнении команды в вычисленном адресе следующей по порядку команды ((PC)= (PC) +2) 11 младших битов заменяются на ad11 из тела команды абсолютного перехода.

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

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

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

Для организации программных циклов удобно пользоваться командой DJNZ, которая в качестве счетчика циклов может использовать не только регистр, но и прямо адресуемый байт (например, ячейка РПД).

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

```
WAIT:       CJNE   A,P0,WAIT      ; Ожидание события
```

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

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

Подпрограммы

Для обращения к подпрограммам необходимо использовать команды вызова подпрограмм (LCALL, ACALL). Эти команды в отличие от команд перехода (LJMP, AJMP) сохраняют в стеке адрес возврата в основную программу. Для возврата из подпрограммы необходимо выполнить команду RET. Команда RETI отличается от команды RET тем, что разрешает прерывания обслуженного уровня. Достаточно часто возникает необходимость такой организации вычислительного процесса, при которой подпрограмма вызывает другую подпрограмму, та в свою очередь вызывает следующую и т.д. Этот процесс называется вложением подпрограмм. Число подпрограмм, которые могут быть вызваны таким образом (глубина вложенности подпрограмм), ограничивается только емкостью стека.

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

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

Получили распространение три способа передачи параметров: через память, через регистры общего назначения и через регистр признаков. При передаче входных параметров через память основная программа обязательно содержит команды загрузки выбранных ячеек памяти, а подпрограмма – команды считывания из этих ячеек. При передаче выходных параметров подпрограмма должна загрузить выбранные ячейки памяти, а основная программа – считать. Передача па-

раметров через регистры осуществляется аналогичным образом. Третий способ передачи параметров – через регистр признаков – удобно использовать при передаче выходных параметров (например, в подпрограммах сравнения чисел). В этом случае подпрограмма должна установить (или сбросить) соответствующие признаки, а основная программа – проанализировать их значение. В распоряжении пользователя имеется также 128 флагов, доступных для модификации и анализа. Возможна также передача параметров через стек. Этот способ, в частности, позволяет использовать в качестве параметра содержимое счетчика команд.

Использование процедур, оформленных в виде подпрограмм, при разработке программного обеспечения имеет ряд достоинств. Прежде всего относительно простые модули, выделенные из сложной программы, могут программироваться несколькими разработчиками в целях сокращения времени проектирования. Еще более важным является то, что любая подпрограмма допускает автономную отладку. Это, как правило, многократно сокращает время отладки всего прикладного программного обеспечения. И, наконец, механизм использования подпрограмм, реализующих требуемый набор процедур, уменьшает длину прикладной программы, что имеет своим следствием уменьшение требующейся емкости памяти программ.

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

7. РАЗВИТИЕ ПЛАТФОРМЫ МКС51

Удачный набор периферийных устройств, возможность гибкого выбора внешней или внутренней программной памяти, приемлемая цена и доступность инструментальных средств отладки разного уровня стали ключевыми факторами успеха микроконтроллеров серии 8051 на мировом рынке. В процессе развития появились усовершенствованные модели микроконтроллеров семейства MCS51, оснащенные дополнительными периферийными устройствами [28].

Наряду с созданием сложных и высокоинтегрированных схем совершенствуются микросхемы, выпуск которых был освоен давно, например 8-разрядные микроконтроллеры или однокристальные микроЭВМ семейства MCS51. Эти микросхемы хорошо зарекомендовали себя в экономичных и сравнительно несложных устройствах. Основными направлениями модернизации данных микроконтроллеров являются увеличение внутренней памяти программ до 32 Кб, причем она может быть выполнена в виде масочного ПЗУ, однократно программируемого ПЗУ или с СППЗУ ультрафиолетовым стиранием; снижение потребляемой мощности путем применения КМОП-технологии и специальных режимов пониженного энергопотребления; увеличение тактовой частоты до 20 МГц; модификация режимов работы счетчиков-таймеров и последовательного порта; размещение на кристалле дополнительного оборудования.

В настоящее время фирма Intel прекратила производство кристаллов MCS51 и сосредоточила свои усилия главным образом на рынке “больших” микропроцессоров (Pentium, Xeon, Itanium и др.). Образовавшуюся нишу тотчас заполнили фирмы с не менее известными именами – Philips, Siemens, AMD, Atmel, Oki, SST, MHS, ISSI, LG и т.д. Некоторые из них выпускают микроконтроллеры собственной разработки с системой команд “базовой архитектуры” и развитой периферией, отвечающей современным потребностям. В этой связи факт ухода Intel с рынка MCS51 для конкретного разработчика может остаться и не замеченным, если только он не использует специализированные микроконтроллеры моделей Intel 8xC51GB и 80C152Jx – эти кристаллы не имеют своих точных аналогов среди изделий других фирм. Что же касается всех остальных МК семейства MCS51, то они многократно воспроизведены другими компаниями, и уход фирмы Intel с рынка никак на нем не скажется.

7.1. Модификация ядра Intel 8051/8052

В состав этого семейства входят микроЭВМ 80С52, 80С54, 80С58 (масочно программируемое ПЗУ), версии 87С52, 87С54 и 87С58 (СППЗУ с УФ-стиранием), а также микроЭВМ 80С32, не имеющая резидентного ПЗУ. Между собой они различаются также корпусами, рабочими интервалами температур, предельно допустимой тактовой частотой и рядом других параметров, отражаемых в буквенно-цифровой информации после обозначения типа микроЭВМ. Эту информацию можно получить из фирменных руководств Intel, AMD и других производителей микроЭВМ рассматриваемого семейства.

В отличие от 8051 микроЭВМ семейства 8052 имеют:

– встроенное ПЗУ объемом 8К (80С52), 16К (80С54) и 32К байтов (80С58);

– встроенное ОЗУ объемом 256 байтов;

– дополнительные специальные функциональные регистры;

– таймер/счетчик 2 (далее для краткости – Т/С2), способный работать в режимах защелки, таймера/счетчика, допускающего счет как на увеличение, так и на уменьшение, и генератора скорости передачи в бодах;

– программируемый последовательный интерфейс с детектированием ошибок передачи и автоматическим распознаванием адреса;

– шесть источников прерываний;

– расширенный режим снижения потребляемой мощности;

– флаг отключения питания;

– режим ONCE.

МикроЭВМ 8052 используют стандартный набор команд семейства 8051, их выводы взаимно однозначно соответствуют выводам этих микроЭВМ. Отличие заключается лишь в том, что, помимо ввода/вывода информации, выводы Р1.0 и Р1.1 i8052 могут выполнять альтернативные функции: первый из них играет роль внешнего входа для Т/С2, а второй управляет перезагрузкой/защелкиванием информации в регистры Т/С2.

7.1.1. Intel 8XC51FA

В качестве одной из перспективных моделей MCS51 можно считать микросхему 8XC51FA. В ее состав входят:

– четыре 8-битных параллельных порта;

– модуль PCA:

– последовательный порт;

– три 16-битных счетчика- таймера.

Микроконтроллеры с резидентной памятью программ позволяют защищать свои программные коды от копирования. Для этого

используется схема блокировки внутренней памяти программ, которая состоит из специальных битов (Lock bits) и кодирующего массива (Encryption Array). Запрограммировав один или несколько таких битов, можно полностью или частично заблокировать эту память. При полной блокировке будут невозможны чтение с внешней шины внутренней памяти программ, дальнейшее программирование кристалла, выполнение команд из внешней памяти программ. При частичной блокировке возможно запретить или разрешить вышеперечисленные действия по отдельности. Кодирующий массив используется для поразрядного выполнения логической операции XNOR над байтами из внутренней памяти программ и байтами из этого массива при верификации, если она разрешена.

Основным отличием моделей 8XC51FA от отечественных аналогов является наличие модуля PCA (Programmable Counter Array).

Это устройство состоит из 16-разрядного счетчика-таймера и пяти модулей сравнения-захвата. В качестве входных импульсов для счетчика-таймера могут служить сигналы:

– частота резонатора /12;
– переполнение от Timer 0;
– частота резонатора /4;
– внешний сигнал на контакте P1.2.

Каждый из пяти модулей сравнения-захвата может работать в следующих режимах :

– захват положительного или отрицательного фронта;
– программный таймер;
– скоростной вывод;
– генератор прямоугольных импульсов с заданной скважностью.

Четвертый модуль имеет также режим сторожевого таймера.

PCA рекомендуется использовать для измерения таких параметров, как ширина импульса, разность фаз, скважность и частота, а также для формирования на внешних выводах микроконтроллера прямоугольных сигналов. В принципе для этих целей можно использовать счетчики-таймеры, которые имеются на кристалле. Однако при использовании PCA повышается точность за счет того, что счетчик-таймер, входящий в состав PCA, может изменять свое значение трижды за машинный цикл. Отметим, что обычные счетчики-таймеры могут изменять свое значение лишь один раз за машинный цикл. Кроме того, PCA требует значительно меньшего вмешательства процессора.

7.1.2. Intel 8XC51GB

Большой интерес для разработчиков электронной аппаратуры могут представлять микроконтроллеры 8XC51GB. На кристалле этого устройства имеется следующее оборудование:

– шесть 8-битных параллельных портов:
– два модуля PCA ;
– три 16-битных счетчика- таймера;
– детектор падения частоты;
– два последовательных порта;
– отдельный Watchdog Timer;
– 8-канальный, 8-битный АЦП поразрядного приближения.

7.1.3. Intel 80C152

Развитие коммуникационных возможностей MCS51 нашло отражение в микроконтроллере 80C152, где наряду с обычным последовательным портом появляется GSC (Global Serial Channel). Это устройство поддерживает стандартные протоколы SDLC и применяемый в сетях Ethernet CSMA/CD. Также возможно использование протоколов, определенных пользователем. GSC обеспечивает работу на физическом и канальном уровнях согласно базовой модели открытых систем ISO. Для передачи информации используются NRZI и манчестерский коды. Кроме GSC микроконтроллер 80C152 имеет пять 8-битных параллельных портов для 48-выводного DIP корпуса (семь для 68-выводного PLCC), два канала ПДП и два счетчика-таймера.

7.1.4. Маркировка микроконтроллеров фирмы Intel

Для маркировки микросхем фирмой INTEL применяется система обозначений из нескольких полей:

```
1    2           3              4
X    XX   XXXXXXXXXXXXXXX XXXXXX
```

Первое поле содержит однобуквенный префикс, отражающий температурный диапазон микросхемы:

A – (Automotive), автомобильное исполнение для расширенного температурного диапазона (-40…+125°C);

M – (Military), исполнение по военным стандартам (-55…+125°C);

Q или **C** – (Commercial), “коммерческий” температурный диапазон (0…+70°C) с (160+8)-часовой динамической термотренировкой;

L или **E** – (Extended), “расширенный” температурный диапазон (–40…+85°C) с (160+8)-часовой динамической термотренировкой;

T – (Extended), расширенный (–40…+85°C) температурный диапазон без термотренировки;

I – (Industrial), исполнение по промышленным стандартам.

Второе поле содержит одно- или двухбуквенный префикс, указывающий на вариант исполнения корпуса микросхемы (Package Type).

Различных типов корпусов микросхем на сегодняшний день несколько десятков, поэтому в качестве примера приведем лишь некоторые обозначения:

A – Ceramic Pin Grid Array, (PGA);
C – Ceramic Dual In-Line Package, (CDIP);
K – Ceramic Quad Flatpack Package, (QFP);
KD – Plastic Quad Flatpack Package, Fine Pitch, Die Down, (PQFP);
KU – Plastic Quad Flatpack Package, Fine Pitch, Die Up, (PQFP);
N – Plastic Leaded Chip Carrier, (PLCC);
P – Plastic Dual In-Line Package, (PDIP);
SM – Single In-Line Leadless Memory Module, (SIMM);
U – Plastic Dual In-Line Package, Shrink Dip, (PDIPS);
Z – Zigzag In-Line Package, (ZIP).

Третье поле может содержать до 15 цифровых и буквенных символов, указывающих на тип конкретного устройства, расположенного на кристалле.

Четвертое поле может включать до шести цифровых и буквенных символов, отражающих различные особенности и варианты исполнения микросхем.

Дополнительную информацию по типам корпусов и их конструктивному исполнению можно найти в книге: Packaging Order Number 240800.

Применительно к описываемым микроконтроллерам семейства MCS51, первый символ третьего поля традиционно (для фирмы Intel) равен (8). Второй символ третьего поля обычно указывает на тип встроенного ПЗУ:

0 – масочное ПЗУ программ; кристалл без ПЗУ (для поздних версий кристаллов);
1 – масочное ПЗУ программ (Standard ROM Code, Firmware);
3 – масочное ПЗУ (для поздних версий кристаллов), (Customizable ROM Code);
7 – УФРПЗУ или однократно-программируемое ПЗУ (EPROM or OTP ROM);
8 – ЭСППЗУ (Flash - память на кристалле).

Далее может следовать буква, указывающая на технологические особенности изготовления:

отсутствие буквы – технология HMOS, питание 5 В;
C – технология CHMOS, питание 5 В;
L – технология CHMOS, питание 3 В.

Следующими символами третьего поля для микроконтроллеров семейства MCS51 являются номера (например, 31, 32, 51, 54, 58, 152) и от одной до четырех букв, которые отражают функциональные особенности кристаллов (например, объем ПЗУ, специфику группы кристаллов, наличие системы защиты памяти программ от несанкциони-

рованного доступа, возможность использования более совершенного алгоритма программирования Quick Pulse и тому подобное).

В оригинальной технической документации фирмы Intel все микроконтроллеры семейства MCS51 скомпонованы по группам продукции (Product Line), каждая из которых объединяет наиболее близкие по своим функциональным возможностям и электрическим параметрам версии кристаллов. Поскольку наименования микросхем одной группы различаются незначительно, то для обозначения каждой отдельной группы применяется обобщенная символика, образованная из маркировки конкретных микросхем, путем замены различающихся символов на "X". Таким образом, можно выделить следующие группы микроконтроллеров.

Группа 8X5X (8051 Product Line и 8052 Product Line): 8031AH, 8051AH, 8751H, 8051AHP, 8751H-8, 8751BH, 8032AH, 8052AH, 8752BH.

Группа 8XC51 (80C51 Product Line): 80C31BH, 80C51BH, 87C51.
Группа 8XC5X(8XC52/54/58 Product Line): 80C32, 80C52, 87C52, 80C54, 87C54, 80C58, 87C58.

Группа 8XC51FX (8XC51FA/FB/FC Product Line): 80C51FA, 83C51FA, 87C51FA, 83C51FB, 87C51FB, 83C51FC, 87C51FC.

Группа 8XL5X (8XL52/54/58 Product Line): 80L52, 87L52, 80L54, 87L54, 80L58, 87L58.

Группа 8XL51FX (8XL51FA/FB/FC Product Line): 80L51FA, 83L51FA, 87L51FA, 83L51FB, 87L51FB, 83L51FC, 87L51FC.

Группа 8XC51RX (8XC51RA/RB/RC Product Line): 80C51RA, 83C51RA, 87C51RA, 83C51RB, 87C51RB, 83C51RC, 87C51RC.

Группа 8XC51GB (8XC51GX Product Line): 80C51GB, 83C51GB, 87C51GB.

Группа 8XC152JX (8XC152 Product Line): 80C152JA, 83C152JA, 80C152JB, 80C152JC, 83C152JC, 80C152JD.

Группа 8XC51SL (8XC51SL Product Line): 80C51SL-BG, 81C51SL-BG, 83C51SL-BG, 80C51-AH, 81C51SL-AH, 83C51SL-AH, 87C51SL-AH, 80C51SL-AL, 81C51SL-AL, 83C51SL-AL, 87C51SL-AL.

7.2. Микроконтроллеры семейства Intel MCS-251/151

Изначально наиболее "узкими" местами архитектуры MCS51 были восьмиразрядное АЛУ на базе аккумулятора и относительно медленное выполнение инструкций (для самых "быстрых" из них требуется 12 периодов тактовой частоты). Это ограничивало применение МК семейства в устройствах, требующих повышенного быстродействия и

сложных вычислений (16- и 32-битных). Насущным стал вопрос принципиальной модернизации старой архитектуры. Проблема осложнялась тем, что к началу 90-х годов уже была создана масса наработок в области программного и аппаратного обеспечения, и одной из основных задач разработки новой архитектуры стала реализация аппаратной и программной совместимости со старыми разработками на базе MCS51. Для решения этой задачи была создана совместная группа из специалистов компаний Intel и Philips, но позднее пути этих двух фирм разошлись. В результате в 1995 г. появилось два существенно различающихся семейства: Intel MCS-251/151 и Philips 51XA.

Основные характеристики архитектуры MSC-251:

– 24-разрядное линейное адресное пространство, обеспечивающее адресацию до 16 Мбайтов памяти;

– регистровая архитектура, допускающая обращение к регистрам как к байтам, словам и двойным словам;

– страничный режим адресации для ускорения выборки инструкций из внешней программной памяти;

– очередь инструкций;

– расширенный набор команд, включающий 16-битные арифметические и логические инструкции;

– расширенное до 64 Кбайтов адресное пространство стека;

– выполнение самой “быстрой” инструкции за два такта;

– совместимость на уровне двоичного кода с программами для MCS51.

Система команд MCS-251 построена на базе двух наборов инструкций: первый является копией системы команд MCS51, а второй состоит из расширенных инструкций, реализующих преимущества архитектуры MSC-251. Перед использованием МК его необходимо сконфигурировать, т.е. с помощью программатора установить конфигурационные байты, определяющие, какой из наборов инструкций станет активным после включения питания. Если установить набор инструкций MCS51, то MSC-251 будет совместим с MCS51 на уровне двоичного кода (режим Binary Mode). Расширенные инструкции в этом режиме также доступны, но через “окно” – зарезервированный код инструкции 0A5H. Естественно, длина каждой расширенной инструкции увеличивается в таком случае на 1 байт.

Если же изначально установить набор расширенных инструкций, то программы, написанные для MCS51, потребуют повторной компиляции на кросс-средствах для MCS51, так как теперь уже стандартные инструкции будут доступны через то же “окно” 0A5H и длина их также увеличится на 1 байт. Такой режим называется Source Mode. Он позволяет с максимальной эффективностью использовать расширенные инструкции и достигнуть наибольшего быстродействия, но требует переработки программного обеспечения.

Для пользователей, ориентированных на применение MCS-251 в качестве механической замены MCS51, фирма Intel выпускает MCS-251 с уже запрограммированными битами конфигурации в состоянии Binary Mode. Такие микроконтроллеры получили обозначение MCS-151.

Помимо Intel, МК MCS-251 по ее лицензии выпускает компания Temic Semiconductors. Подробную информацию о ее продукции можно получить на web-site фирмы (`http://www.temic-semi.com`). Основные технические характеристики МК семейства MCS-251 приведены в табл. 7.1.

Таблица 7.1. Микроконтроллеры семейства Intel MCS-251/151

Микро-контроллер	ROM или EPROM, Кбайт	RAM, байт	Таймеры-счетчики	Последовательные каналы	Корпус (тип, число выводов)
8xC251SA	8	1024	3, PCA, WDT	UART	D40, L44
8xC251SB	16	1024	3, PCA, WDT	UART	D40, L44
8xC251SP	8	512	3, PCA, WDT	UART	D40, L44
8xC251SQ	16	512	3, PCA, WDT	UART	D40, L44
TSC8xC251G1	16	1024	3, WDT	UART, I2C, SPI	L40, Q44
TSC8xC251A1	24	1024	2, WDT	UART	D40, L44, Q44
8Xc151SA	8	256	3, PCA, WDT	UART	D40, L44
8xC151SB	16	256	3, PCA, WDT	UART	D40, L44

Примечание. Максимальная тактовая частота всех модификаций – 16 МГц, число линий ввода/вывода – 32. МК TSC8xC251A1 имеет четырехканальный восьмиразрядный АЦП. Напряжение питания для всех микроконтроллеров от 4.5 до 5.5 В, рабочий интервал температур от −40 до +85°С.

Принятые сокращения: PCA – массив программируемых счетчиков; WDT – сторожевой таймер; UART – универсальный асинхронный последовательный приемопередатчик; I2C – двухпроводная двунаправленная шина. Корпус: D – DIP, L – PLCC, Q – QFP.

7.3. Микроконтроллеры семейства Intel MCS-96

7.3.1. Общая характеристика

В семейство MCS-96 фирмы Intel (80C196) входит более 30 разновидностей микроконтроллеров. Это 16-разрядные, быстродействующие интегральные схемы высокой степени интеграции, ориентированные на решение задач управления процессами в реальном масштабе времени. Типичные области применения этих микроконтроллеров: управление двигателями, модемы, безъюзовые тормозные системы, контроллеры жестких дисков, медицинское оборудование.

История MCS-96 насчитывает более 15 лет. За это время специалисты фирмы Intel увеличили адресное пространство с 64 КБайт до 6 Мбайт, повысили тактовую частоту с 10 до 50 МГц, улучшили быстродействие в 16 раз и добились понижения цены на базовый кристалл примерно в 4 раза.

По сравнению с восьмиразрядными однокристальными микроконтроллерами данное микроконтроллерное семейство позволяет существенно расширить область применения встраиваемых микроконтроллеров в первую очередь за счет более высокой скорости и точности вычислений, а также за счет использования расположенных на кристалле новых периферийных устройств, обеспечивающих более высокую скорость обработки сигналов в управляющей системе и более высокую надежность функционирования системы.

Микроконтроллеры 80C196 фактически стали индустриальным стандартом для 16-разрядных встроенных систем управления, обеспечивая сочетание высоких технических показателей и экономической эффективности. Например, именно благодаря этим микроконтроллерам, установленным в системе управления зажиганием, специалистам концерна Ford удалось существенно снизить потребление топлива, уменьшить выбросы вредных веществ и одновременно повысить скоростные характеристики своих машин.

7.3.2. Структура микроконтроллера

Микроконтроллеры семейства MCS-96 являются микропроцессорными устройствами синхронного типа. Выполнение всех действий в микроконтроллере привязано во времени к тактовым сигналам, вырабатываемым внутренним генератором тактовых импульсов. Частота следования тактовых импульсов стабилизируется с помощью внешнего кварцевого резонатора. Высшее значение тактовой частоты (F_{max}) у микроконтроллеров разных типов может иметь значение 10, 12, 16 и 20 МГц.

Основными функциональными частями микроконтроллера являются процессор, память и периферия (набор периферийных устройств).

В состав процессора входят арифметико-логическое устройство (АЛУ, ALU) и регистровое оперативное запоминающее устройство (РОЗУ, RRAM).

АЛУ. В отличие от микроконтроллеров других семейств, АЛУ микроконтроллера семейства MCS-96 не имеет регистра-аккумулятора. В качестве регистра-аккумулятора может использоваться любой регистр РОЗУ. На частоте 16 МГц АЛУ выполняет 2 млн. оп/с в процессе элементарных операций над знаковыми/беззнаковыми данными длиной 1 или 2 байт. Для этих чисел имеются также операции умножения и деления (быстродействие: 580 тыс. умножений/с, 330 тыс. делений/с).

РОЗУ у микроконтроллеров разных типов может содержать 232, 360, 488 или 1000 восьмиразрядных регистров. Регистры РОЗУ используются для хранения только данных.

Память представлена постоянным (ПЗУ, ROM) запоминающим устройством. У микроконтроллеров некоторых типов в состав памяти входит оперативное запоминающее устройство (ОЗУ, RAM). Ячейки памяти в ОЗУ и ПЗУ могут использоваться для хранения данных и команд программы.

ПЗУ у контроллеров разных типов может содержать 8К, 12К, 16К, 24К или 32К восьмиразрядных ячеек памяти. В ПЗУ имеется область, предназначенная для хранения специальных данных (векторы прерывания, ключ защиты ПЗУ и другие специальные коды).

ОЗУ у контроллеров разных типов может иметь 128, 256 или 512 восьмиразрядных ячеек памяти. При использовании ОЗУ для размещения команд программы открывается возможность производить модификацию кода команд в процессе выполнения программы.

Процессор обращается к памяти через контроллер памяти (КП, MC). Через контроллер памяти осуществляется также обращение к внешней памяти, реализованной с помощью микросхем ОЗУ и ПЗУ. Контроллер памяти позволяет при одном обращении к памяти считывать или записывать как восьмиразрядные,так и шестнадцатиразрядные коды.

Максимальный суммарный объем внешней и внутренний памяти (без РОЗУ) у микроконтроллеров большинства типов составляет 64К×8 бит. У микроконтроллеров подсемейства NT суммарный объем памяти может быть доведен до 1М×8 бит. Микроконтроллеры, в обозначении типа которых на втором месте стоит цифра 0 (X=0), не имеют внутреннего ПЗУ. Его функции реализуются с помощью микросхем ПЗУ, входящих в состав внешней памяти.

Периферийные устройства микроконтроллера

Периферийные устройства семейства MCS-96 по выполняемым функциям могут быть отнесены к одной из шести групп:

- устройства ввода и вывода данных, представленных многоразрядными двоичными кодами;
- устройства ввода и вывода отдельных дискретных сигналов (включено-выключено);
- устройства ввода и вывода аналоговых сигналов;
- устройства обмена данными с другими микроконтроллерами и центральным процессором системы;
- устройства приема и обслуживания запросов прерывания;
- устройства контроля правильности функционирования микроконтроллера.

Для управления работой периферийных устройств и определения их состояния используются регистры специальных функций.

Таймеры. Два 16-разрядных таймера TIMER1 и TIMER2 обеспечивают синхронизацию работы устройства ввода-вывода импульсных сигналов (HSIO, High Speed In/Out unit) с реальным временем и внешними событиями. TIMER1 синхронизируется изнутри, тогда как TIMER2 синхронизируется снаружи.

Code RAM. Это дополнительное ОЗУ, в котором можно размещать исполняемый код. Этот код будет выполняться очень быстро, так как Code RAM имеет 16-разрядный интерфейс с нулевым циклом ожидания. Code RAM может принести существенную пользу в задачах, где требуется максимально быстрое выполнение только небольших фрагментов кода, позволяя при этом использовать сравнительно медленное и дешевое 8-битное ПЗУ для хранения остальной части программы. Конечно, эту память можно использовать и для размещения данных или стека.

Энергопотребление. Общее потребление – не более 75 мА на частоте 16 МГц. Имеются режимы с пониженным энергопотреблением: IDLE (30 мА) и POWER DOWN (0.1 мА).

Температурный диапазон и типы корпусов. Существует четыре разновидности по температурному диапазону работы: 0…+70°C (коммерческий), -40…+85°C (расширенный), –40…+125°C (автомобильный и военный). Кроме того, микроконтроллеры могут быть подвергнуты динамической электротермотренировке. Используются корпусы PLCC-68, QFP-80, керамический LCC-68 и керамический PGA-68.

7.3.3. Описание периферийных устройств

Устройства ввода и вывода данных

Ввод и вывод данных, представленных многоразрядными двоичными кодами, осуществляется через параллельные порты. В микроконтроллерах семейства MCS-96 используются восьмиразрядные и четырехразрядные порты. При этом микроконтроллер может иметь от четырех до восьми портов (табл. 7.2).

Два восьмиразрядных порта (P3 и P4) предназначены для подключения внешней памяти. Использование этих портов для ввода и вывода данных возможно лишь при ее отсутствии. Отдельные выводы параллельных портов могут выполнять альтернативные функции (прием запросов прерывания, вывод сигналов управления и др.). Для перевода выводов портов в режим альтернативных функций необходимо загрузить определенное управляющее слово в соответствующий регистр специальных функций.

Таблица 7.2. Характеристики портов ввода-вывода

Тип линии порта	Количество линий порта					Всего
	P0	P1	P2	P3 и P4	HSIO	
Двунаправленная	–	8	2	16/0	2	28/12
Только вход	8	–	4	–	2	14
Только выход	–	–	2	–	4	6
Итого:	8	8	8	16/0	8	48/32

Примечание. Порты 3/4 заняты, если используется внешняя шина.

Устройство ввода и вывода дискретных сигналов

Дискретные сигналы (включено-выключено) широко используются в системах управления. Изменение значения дискретного сигнала называется событием.

В микроконтроллерах семейства MCS-96 для обработки входных и формирования выходных событий используются специальные периферийные устройства, осуществляющие быстрый ввод и быстрый вывод без непосредственного участия процессора.

Быстрый ввод заключается в обнаружении события определенного типа на определенном входе микроконтроллера и запоминании времени его наступления в заданной системе отсчета времени. Быстрый

вывод заключается в формировании события определенного типа на заданном выходе микроконтроллера в заданный момент времени.

Для выполнения операций быстрого ввода и вывода в микроконтроллерах разных типов используется или блок быстрого ввода-вывода (HSIO), или блок процессоров событий (EPA).

В обоих блоках для формирования текущего значения времени используются шестнадцатиразрядные таймеры-счетчики, на счетные входы которых подаются сигналы времени от внутреннего генератора или от внешнего источника.

В блоке быстрого ввода-вывода (HSIO) для обработки входных событий и формирования выходных событий используются специализированные модули для ввода и для вывода, а в блоке процессоров событий (EPA) содержится набор универсальных модулей, каждый из которых при программировании настраивается на работу или в режиме быстрого ввода (capture-захвата), или в режиме быстрого вывода (compare-сравнения).

По результатам обработки входных событий могут вычисляться параметры импульсных последовательностей на входах микроконтроллера – период следования импульсов, их длительность, сдвиг во времени между импульсами на разных входах и другие параметры.

Блоки HSIO и EPA кроме операций быстрого ввода и вывода могут использоваться для формирования временных задержек (режим программного таймера), формирования сигналов специальной формы (например, сигнала с широтно-импульсной модуляцией), запуска аналого-цифрового преобразователя и выполнения некоторых других функций.

Устройства ввода и вывода аналоговых сигналов

У микроконтроллеров большинства типов в число периферийных устройств входит многоканальный аналого-цифровой преобразователь (ADC). Число каналов может быть равно 4, 6, 8, 13 или 14. Входное напряжение в канале может изменяться в пределах от 0 до 5 (5.12) В. В результате преобразования формируется восьмиразрядный или десятиразрядный двоичный код.

Запуск преобразования в канале может производиться по команде в программе или по сигналу из блока HSIO или EPA в заранее заданное время. Некоторые преобразователи могут работать в режиме сканирования входов.

На частоте 16 МГц время преобразования – 19.5 мкс. Имеется схема выборки/хранения, а также отдельные входы опорного напряжения и аналоговой земли.

Преобразование цифровых данных в аналоговый сигнал выполняется с использованием широтно-импульсного модулятора (PWM).

Широтно-импульсный модулятор формирует последовательность прямоугольных импульсов, следующих с постоянным периодом. Длительность импульса пропорциональна числу, преобразуемому в значение аналогового сигнала. Получаемая импульсная последовательность с выхода микроконтроллера с переменной скважностью подается на внешнюю интегрирующую схему, с выхода которой снимается аналоговый сигнал.

Диапазон изменения скважности импульсов – 256 градаций. Период импульсов может быть равен 256 или 512 тактам (31.25 или 15.625 кГц соответственно, для тактовой частоты 16 МГц).

В микроконтроллерах подсемейства MC кроме двух широтно-импульсных модуляторов имеется специальный блок, содержащий три широтно-импульсных модулятора, работающих совместно. Этот блок, называемый генератором периодических колебаний (WG), имеет три пары выходов. Разность напряжений на выходах одной пары представляет собой синусоидоподобный ступенчатый сигнал. Сигналы, снимаемые с трех пар выходов, могут быть использованы для питания трехфазных индукционных двигателей переменного тока. Блок позволяет также формировать сигналы для управления вентильными двигателями постоянного тока, шаговыми двигателями и для некоторых других целей.

Устройства обмена данными с другими микроконтроллерами и центральным процессором

Обмен данными с другими микроконтроллерами в управляющей системе, содержащей несколько совместно работающих микроконтроллеров, может осуществляться по последовательному каналу или путем совместного использования внешней памяти.

Обмен данными по последовательному каналу выполняется с использованием последовательного порта (SP). Обмен производится путем посылки отдельных кадров, каждый из которых содержит стартовый бит, семь или восемь информационных битов и один стоповый бит. В состав кадра может быть включен дополнительный бит, который используется для контроля по четности правильности пересылки данных или для различения кадров, содержащих адреса абонентов, и кадров, содержащих данные, при включении контроллера в простейшую локальную сеть.

Последовательный порт может также осуществлять последовательный ввод или вывод байтов с использованием внешних сдвигающих регистров, которые в этом случае выполняют функции дополнительных параллельных портов ввода или вывода.

У микроконтроллеров некоторых типов в число периферийных устройств входит второй последовательный порт (SSIO), с помощью

которого осуществляется непосредственный обмен байтами между двумя микроконтроллерами путем последовательной передачи байта и сопровождающей серии импульсов сдвига. Порт SSIO содержит два последовательных канала, каждый из которых может работать в режиме передачи или в режиме приема. Максимальная скорость обмена (на частоте 16 МГц): в асинхронном режиме – 1 Мбод, в синхронном режиме – 4 Мбод.

Микроконтроллеры почти всех модификаций имеют аппаратные средства, обеспечивающие совместное использование внешней памяти несколькими микроконтроллерами. Согласование работы микроконтроллеров при обращении к внешней памяти реализуется с помощью сигналов HOLD, HLDA, BREQ и дополнительной внешней аппаратуры.

У микроконтроллеров некоторых типов имеется "подчиненный" порт (Slave Port), предназначенный для обмена данными с центральным процессором в иерархической управляющей системе. Через "подчиненный" порт микроконтроллер подключается непосредственно к системной магистрали микропроцессорной системы. Обмен данными происходит под управлением центрального процессора, который обращается к микроконтроллеру, как к собственному порту ввода и вывода. При появлении необходимости передать данные в центральный процессор микроконтроллер посылает запрос прерывания.

Устройства приема и обслуживания запросов прерывания

Запросы прерывания текущей программы могут поступать от внешних источников или формироваться внутри микроконтроллера в различных периферийных устройствах. Общее число источников запросов прерывания у микроконтроллеров разных типов может быть 21, 28 или 37.

Запросы прерывания могут маскироваться путем посылки кодов маски в соответствующие регистры специальных функций. В микроконтроллерах всех типов имеется программный контроллер прерываний (PIC). Обслуживание запроса прерывания с использованием PIC заключается в переходе от выполнения текущей программы к выполнению другой определенной программы, составленной разработчиком программного обеспечения.

Адрес первой команды каждой прерывающей программы (вектор прерывания) хранится в определенной паре ячеек ПЗУ в области памяти, отведенной для хранения специальных данных. После завершения выполнения прерывающей программы происходит возврат к прерванной программе.

Прерывающая программа в свою очередь может быть прервана при поступлении любого незамаскированного запроса прерывания

вне зависимости от соотношения приоритетов запроса, вызвавшего переход к данной программе, и нового запроса прерывания.

В микроконтроллерах некоторых типов кроме программного контроллера прерываний имеется микропрограммный контроллер прерываний (PTS). Любой запрос прерывания, кроме нескольких особых запросов, может быть направлен для обслуживания или в PIC, или в PTS.

Обслуживание запроса прерывания с использованием PTS заключается в выполнении типовой микропрограммы, при этом выполнение операций по микропрограмме совмещается во времени с выполнением команд текущей программы. Микропрограммы PTS охватывают в основном пересылки данных. Прерывания, обслуживаемые PTS, отрабатываются быстрее, чем те, которые обслуживаются обычным способом. Однако программировать PTS непросто, а отлаживать – еще сложнее.

Устройства контроля правильности функционирования микроконтроллера

Все микроконтроллеры семейства MCS-96 имеют сторожевой таймер (WDT). Сторожевой таймер по прошествии определенного интервала времени переводит микроконтроллер в состояние сброса. Правильно работающая программа должна предотвращать сброс микроконтроллера от WDT путем периодического сброса в нулевое состояние самого WDT. При сбое в ходе программы сторожевой таймер своевременно не сбрасывается, и при его переполнении микроконтроллер переводится в состояние сброса, что предотвращает появление и развитие опасных ситуаций в системе управления.

Микроконтроллеры некоторых типов имеют схему обнаружения падения частоты генератора тактовых импульсов (OFD). При снижении частоты ниже определенного уровня OFD вырабатывает сигнал сброса и переводит микроконтроллер в состояние сброса. Это предотвращает появление опасных комбинаций сигналов на выходах микроконтроллера, которые могут возникнуть при остановке генератора тактовых импульсов в произвольный момент времени в процессе выполнения программы.

Характеристики микроконтроллеров подсемейств

К числу основных функциональных характеристик микроконтроллера относятся:

– емкость расположенных на кристалле регистрового оперативного запоминающего устройства (RRAM), постоянного запоминающего устройства (ROM), оперативного запоминающего устройства (RAM);

– максимальная тактовая частота (F_{max});
– число команд в системе команд (N);
– состав периферийных устройств.

По значению тактовой частоты может быть определено быстродействие микроконтроллера. У микроконтроллеров подсемейства 8X9Y команды коротких операций выполняются за 12 периодов тактовой частоты. При тактовой частоте 12 МГц микроконтроллеры данного подсемейства имеют быстродействие 1 млн. коротких операций в секунду. У микроконтроллеров остальных подсемейств команды коротких операций выполняются за 8 периодов тактовой частоты, и при тактовой частоте 16 Мгц обеспечивается быстродействие 2 млн. коротких операций в секунду.

При этом следует иметь в виду, что короткие операции в микроконтроллере семейства MCS-96 по своему содержанию существенно отличаются от коротких операций в микроконтроллере с регистром-аккумулятором. Так, например, одной короткой операции "сложение" в микроконтроллере семейства MCS-96 при представлении данных в формате "байт" соответствует последовательность из трех коротких операций в микроконтроллере семейства MCS51, а при представлении данных в формате "слово" соответствует последовательность из шести коротких операций.

Кроме того, в систему команд микроконтроллеров семейства MCS-96 входят команды умножения и деления чисел в формате "слово". В микроконтроллерах других семейств такие операции выполняются по подпрограммам, что резко увеличивает время их выполнения.

Отмеченные особенности существенно сокращают время вычислений в микроконтроллерах семейства MCS-96 по сравнению с микроконтроллерами других семейств.

В систему команд микроконтроллеров, реализованных по КМОП технологии, входят различные дополнительные команды, в числе которых имеется команда перевода микроконтроллера в энергосберегающие режимы – режим холостого хода и режим пониженного энергопотребления. В режиме холостого хода программа не выполняется, но функционируют все периферийные устройства, при этом потребление энергии от источника питания уменьшается на 60 %. В режиме пониженного энергопотребления прекращаются все процессы в микроконтроллере, но сохраняются данные в РОЗУ и ОЗУ. При этом ток потребления составляет единицы микроампер.

С появлением на рынке цифровых интегральных схем микроконтроллеров семейства MCS-96 фирмы Intel перед разработчиками систем, содержащих встроенные микроконтроллеры, открываются новые большие возможности по созданию высокосовершенных, малогабаритных, экономичных и надежных систем, приборов и устройств различного назначения.

7.3.4. Преимущества микроконтроллеров MCS-96

Кристаллы 80C196 изготавливаются по более современной технологии (с меньшим размером элементов на кристалле), поэтому достигаются более высокие тактовые частоты. Так, кристалл 80C196NU имеет тактовую частоту 50 МГц, а наиболее быстродействующие из семейства 8051 – 24 МГц.

Все 232 внутренних регистра 80C196 имеют статус “аккумуляторов” – к ним можно непосредственно применять все необходимые арифметические и логические операции. У 8051 для достижения тех же результатов зачастую необходимо выполнять дополнительные пересылки в аккумулятор и из него.

У 80C196 можно использовать 16-разрядную внешнюю шину. Кроме того, цикл шины 80C196 в 3-4 раза короче, чем у 8051. В результате 80C196 в 6-8 раз быстрее работает с внешней памятью. Отметим, что для того, чтобы снизить стоимость изделий, можно вводить в шину циклы ожидания и сократить ее ширину до 8 битов, но даже в этом случае 80C196 будет иметь преимущество в 2–3 раза.

В задачах, требующих 16- и 32-разрядных вычислений, 80C196 примерно на порядок быстрее, поскольку имеет полноценный набор 16-разрядных арифметических инструкций.

Одним из самых эффективных способов сокращения времени разработки программ для микроконтроллеров является применение языка Си. Язык Си базируется на широком использовании стека и указателей. Однако для 8051 использование Си затруднено и ведет к большим накладным расходам, и вот почему. Поскольку 8051 имеет небольшой стек, Си-компиляторы для 8051 генерируют дополнительный код, эмулирующий большой стек во внешней памяти данных. Вдобавок 8051 имеет всего один 16-разрядный указатель – DPTR, и компиляторам также приходится генерировать дополнительный код, чтобы компенсировать этот недостаток. Все это приводит к замедлению программ и увеличению их размера. У 80C196 таких проблем нет – стек имеет размер до 64 КБайт, а в качестве указателя можно использовать любое из 116 слов встроенной регистровой памяти.

7.4. Микроконтроллеры фирмы Philips

Фирму Philips можно по праву назвать чемпионом по количеству выпускаемых ею модификаций семейства 8051 – их более 100. В состав семейства 8051 от Philips входят микроконтроллеры в корпусах от 24 до 80 выводов, с тактовыми частотами до 40 МГц и напряжением питания от 1.8 В. Во всех микроконтроллерах Philips используется стандартное ядро MCS51, поэтому все временные и функциональные

характеристики полностью соответствуют характеристикам микроконтроллеров фирмы Intel. Фирма Philips значительные усилия направила на интегрирование широкого спектра периферийных устройств на базе ядра 8051.

Основные элементы периферии Philips:

– АЦП с точностью преобразования 10 разрядов;

– широтно-импульсные модуляторы;

– массивы программируемых счетчиков-таймеров;

– интерфейсы I2C, CAN;

– интерфейсы с процессорными шинами;

– EEPROM и FLASH на кристалле;

– специализированная периферия для телевизионной, видео- и аудиотехники.

Подробная информация о всех модификациях микроконтроллеров фирмы Philips содержится на web-site фирмы Philips:

http://www.philipsmcu.com .

С 1997 года фирма Philips переводит стандартные микроконтроллеры групп 80C51, 80C52/54/58 и 80C51FX на новую технологию, которую она сама так и назвала “Новая и улучшенная”.

Какие же новые возможности появились у хорошо известных кристаллов после модернизации?

– Максимальная тактовая частота кристаллов увеличена до 33 МГц;

– расширен диапазон напряжения питания от 2.7 до 5.5 В;

– количество аппаратных уровней прерываний увеличено до 4-х;

– во все кристаллы введена функция программируемого clock-out;

– UART заменен на улучшенный (enhanced);

– функция снижения электромагнитных помех (Lower EMI);

– добавлен второй регистр DPTR;

– потребление энергии для питания микроконтроллера снижено на 50 %. В сочетании с 3-вольтовым питанием это может дать экономию до 75 % по сравнению с предыдущими образцами; снижена цена на 30 %.

Фактически такие новые возможности дают второе рождение старым кристаллам. Проблема для разработчика, однако, состоит в том, что маркировка микроконтроллеров после модернизации не изменилась, из-за чего возможна путаница между старыми и новыми модификациями.

Кроме того, фирма Philips выпустила группу микроконтроллеров, названную RX+. По сути, это дальнейшее развитие группы FX, в которой расширен объем внутреннего ОЗУ (512 байт, 1 Кбайт) и программной памяти (до 64 Кбайт). Группа RX+ обладает также всеми возможностями, предоставляемыми технологией.

С 1997 года фирма Philips держит курс на развитие Flash-технологии в производстве микроконтроллеров. Отчасти это вызвано высокими технологическими возможностями Philips, отчасти – успехами конкурентов, в первую очередь Atmel. Несмотря на то, что Flash-память дороже в производстве, чем EPROM, в конечном итоге для фирмы дешевле будет поддерживать единый технологический процесс.

7.5. Микроконтроллеры фирмы Analog Devices

Фирма Analog Devices выпускает широкую номенклатуру микроэлектронного оборудования, включая однокристальные микроконтроллеры. Отличительной особенностью новейшего семейства микросхем серии ADuC8xx является сам их принцип построения. Эти микросхемы не являются "микроконтроллером со встроенными АЦП-ЦАП". Они представляют собой удачно скомбинированные АЦП и ЦАП со встроенным в них микроконтроллером и флэш-памятью. Поэтому их основным достоинством является высокая точность аналого-цифрового и цифро-аналогового преобразования, удачно сочетаемая с возможностью непосредственной обработки получаемой информации. Иногда такие приборы называют микроконверторами, позиционируя их как отдельный класс микроконтроллеров со сверхвысокой степенью интеграции. Широкую известность получил микроконтроллер ADuC812, специфицируемый фирмой как 8-канальный прецизионный 12-разрядный аналого-цифровой преобразователь со встроенным микропроцессором. Выпускаются также ADuC810 (10-разрядный АЦП, два 12-разрядных ЦАП, микроконтроллерное ядро с Flash РПП), ADuC816 (два 16-разрядных сигма-дельта АЦП, 12-разрядный ЦАП, микроконтроллер с Flash РПП), ADuC824, ADuC831 и ряд других микроконтроллеров (табл. 7.3).

Микроконтроллеры ADuC8xx выполнены в корпусах типа PQF52 или CSP56. Исключение составляет ADuC814, выполненный в корпусе типа TSSOP28. Начиная с модели ADuC831 все микроконтроллеры имеют 2 Кб статического ОЗУ (в дополнение к 256 байтам ядра 8052), а объем Flash РПП увеличен до 62 Кб. Некоторые типы микроконтроллеров (см. табл. 7.3) имеют скоростное ядро 8052 с машинным циклом, равным тактовому. Рассмотрим более детально некоторые модели ADuC8xx.

7.5.1. Микроконтроллер ADuC812

Первой микросхемой семейства ADuC8xx, запущенной в серийное производство в мае 1999 года, является ADuC812. ADuC812 – интегральная 12-разрядная система сбора информации, включающая

Таблица 7.3. Микроконтроллеры фирмы Analog Devices с ядром 8052

Тип	АЦП	ЦАП	РПП Кб	РПД		Дополнительно
				ЕЕ байт	ОЗУ байт	
ADuC812	8/12	2/12	8	640	256	АЦП 5мкс.
ADuC814	6/12	2/12	8	640	256	Миниатюрный; дешевый; микро-мощный
ADuC816	2/16	1/12	8	640	256	Программир. вх. усилитель
ADuC824	1/24 1/16	1/12	8	640	256	Обратная совместимость с ADuC816
ADuC831	8/12	2/12 2ШИМ	62	4K	256 +2K	Модификация ADuC812
ADuC832	8/12	2/12 2ШИМ	62	4K	256 +2K	Модификация ADuC812, ФАПЧ
ADuC834	1/24 1/16	1/12 2ШИМ	62	4K	256 +2K	Модификация ADuC824
ADuC836	2/16	1/12 2ШИМ	62	4K	256 +2K	Модификация ADuC816
ADuC841	8/12	2/12 2ШИМ	62	4K	256 +2K	Модификация ADuC812, СЯ52
ADuC842	8/12	2/12 2ШИМ	62	4K	256 +2K	Модификация ADuC812, ФАПЧ, СЯ52
ADuC844	1/24 1/16	1/12 2ШИМ	62	4K	256 +2K	Модификация ADuC824, СЯ52
ADuC845	10/24 1/24	1/12 2/16 2ШИМ 16 бит	62	4K	256 +2K	Модификация ADuC824, СЯ52
ADuC846	2/16	1/12 2ШИМ	62	4K	256 +2K	Модификация ADuC816, СЯ52
ADuC847	10/24	2/16 2ШИМ 16 бит	62	4K	256 +2K	Модификация ADuC816, СЯ52

Примечание. Для АЦП и ЦАП в числителе указано количество каналов, в знаменателе – разрядность; ЕЕ – EEPROM данных; СЯ52 – скоростное ядро 8052.

в себя прецизионный 8-канальный АЦП с самокалибровкой, два 12-разрядных ЦАП с выходом по напряжению, встроенный источник опорного напряжения, внутренний температурный сенсор и программируемое 8-битовое микропроцессорное ядро, совместимое с 8052 (рис. 7.1). Номинальная тактовая частота составляет 12 МГц, а максимальная – 16 МГц. Три 16-разрядных счетчика/таймера, 9 источников прерываний и 2 уровня приоритета. Микропроцессорное ядро поддерживается встроенной Flash РПП 8 Кбайт, 640 байт EEPROM РПД со страничной организацией, внутренний генератор подкачки заряда, 256 байт статической РПД с произвольной выборкой, 16 Мбайт пространства внешней памяти данных, контроллер прямого доступа к внешней памяти данных, 64 Кбайт пространства внешней памяти программ. Дополнительное встроенное оборудование: сторожевой таймер, монитор питания и канал прямого доступа к памяти для АЦП. Для мультипроцессорного обмена и расширения ввода-вывода имеется 32 программируемые линии ввода-вывода, порт с высоким током (Port 3), I^2C-, SPI- и UART-интерфейсы. Для гибкого управления в приложениях с низким потреблением в микроконтроллере и аналоговой части предусмотрены 3 режима работы: нормальный, холостой и дежурный. Продукт специфицирован для +3/+5 В работы в промышленном диапазоне температур и поставляется в 52-выводном пластмассовом корпусе PQFP.

Микросхема состоит из двух основных частей – аналоговой и цифровой (см. рис. 7.1).

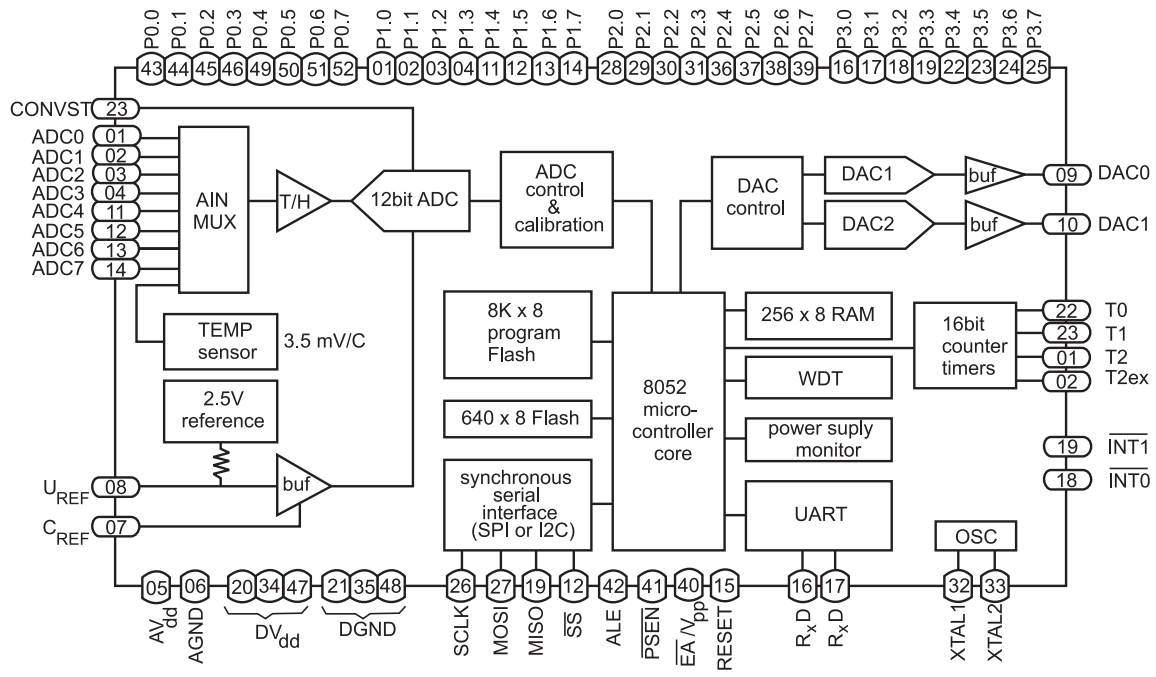


Рис. 7.1. Функциональная блок-схема микроконтроллера ADuC812

Аналоговые входы микросхемы соединены с 8-входовым мультиплексором. На выходе мультиплексора стоит усилитель выборки/хранения, фиксирующий значение аналогового сигнала на выбранном входе на время осуществления преобразования АЦП. Помимо него к аналоговой части микросхемы относятся также два 12-разрядных ЦАП с буферными усилителями на выходе каждого из них. Источник опорного напряжения может использоваться либо внутренний, который имеет напряжение 2.5 В и температурную стабильность 40 ppm/°C, либо внешний, напряжение которого не превышает уровень источника питания. Также к входному мультиплексору подключен внутренний датчик температуры, позволяющий оперативно измерять температуру кристалла (а значит, с определенными поправками, и температуру окружающей среды) или, к примеру, осуществлять компенсацию температуры холодного спая термопар, подсоединенных к одному или нескольким входам микросхемы. На аналоговые входы допустима подача сигналов в диапазоне от 0 до Vref.

Интерфейсом между аналоговой и цифровой частями микросхемы служат регистры управления и калибровки. Цифровая часть состоит из собственно ядра микроконтроллера, полностью совместимого по системе команд с наиболее широко распространенными в мире микроконтроллерами серии 8051, блока памяти и набора дополнительных периферийных устройств. Микросхема ADuC812 может питаться от источника напряжением 3 или 5 В и имеет несколько экономичных режимов работы.

АЦП последовательных приближений может работать как в режиме единичных, так и непрерывных преобразований с максимальной скоростью 200 тысяч преобразований в секунду (одно преобразование каждые 5 мкс). Для запоминания результатов преобразования используется либо режим прерываний (как правило, его удобно использовать при невысокой частоте работы АЦП), либо режим прямого доступа, не влияющий на работу собственно контроллера и позволяющий сохранять результаты преобразования во внешнем ОЗУ с адресуемым пространством 16 Мбайт. АЦП имеет очень хорошую точность (соотношение сигнал/шум 70 дБ, что соответствует реальному разрешению на уровне 11.5 разрядов), и высокую линейность (типовая интегральная нелинейность на уровне $\pm 1/2$ МЗР). Микросхема выпускается с заводской калибровкой под оптимальную производительность. При каждом включении источника питания микросхемы эти коэффициенты записываются в соответствующие регистры.

В большинстве приложений этих коэффициентов достаточно для хорошей работы системы, однако пользователь в процессе работы может перезаписать их для избавления от дополнительных системных ошибок. Все режимы работы АЦП определяются тремя регистрами

управления, находящимися во внутренней памяти микроконтроллера. Результаты преобразования могут быть считаны из двух регистров, один из которых показывает номер канала мультиплексора и старшие 4 бита результата, а второй -— младшие 8 бит результата.

Что касается ЦАП, они управляются одним регистром управления и четырьмя регистрами данных. Обновление информации на выходе ЦАПов может происходить отдельно для каждого из них либо одновременно. Кроме того, каждый из них может быть сконфигурирован для работы либо в 12-разрядном, либо 8-разрядном режиме.

Микроконтроллер представляет собой “стандартное” ядро 8052 с максимальной рабочей частотой 16 МГц (12 МГц — типовая), тремя байтовыми портами ввода-вывода, один из которых, порт 3, обладает повышенной нагрузочной способностью, тремя 16-разрядными таймерами/счетчиками и расширенной периферией, которая будет описана ниже.

Блок памяти состоит из флэш-памяти программ объемом 8 Кбайт, флэш-памяти данных объемом 640 байт и ОЗУ объемом 256 байт. Информация во внутреннюю флэш-память программ может быть записана как с любого внешнего программатора в параллельном режиме через порты микроконтроллера, так и непосредственно в системе в последовательном режиме через асинхронный последовательный порт.

Дополнительные периферийные устройства

К блоку расширенной периферии можно отнести дополнительные аппаратные возможности микросхемы, отсутствующие в оригинальной архитектуре 8051. Это дополнительные последовательные порты, дающие микросхеме возможность работать в ставших стандартными 2-проводных и 3-проводных синхронных протоколах SPI и I^2C. Также микросхема дополнена двумя мониторами: один следит за отсутствием «зависания» микроконтроллера и в случае обнаружения вырабатывает сигнал сброса в начальное состояние, а второй следит за тем, чтобы напряжение источника питания не падало ниже определенного задаваемого пользователем значения (от 2.6 до 4.6 В). Он позволяет в случае, близком к потере питания, сохранить содержимое внутренних регистров, запомнить свое состояние и возобновить работу только после восстановления питания.

Таймер-счетчик T/C2. Микропроцессорное ядро i8052 содержит третий (дополнительный) 16-разрядный таймер-счетчик T/C2, отсутствующий в оригинальной архитектуре 8051. Данный таймер-счетчик предназначен для работы в одном из трех режимов: перезагружаемый аппаратно 16-разрядный счетчик-таймер, 16-разрядный модуль

входного захвата, программируемый генератор импульсов для тактирования контроллера последовательного порта. Для обслуживания T/C2 предназначены пять 8-разрядных регистров (табл. 7.4) и два бита порта P1, наделенных альтернативными функциями:

P1.0 – внешний вход T2 счетчика таймера T/C2;
P1.1 – вход T2EX триггера захвата/перезагрузки.

Таблица 7.4. Регистры таймера-счетчика T/C2 микроконтроллера ADuC812

Параметр	Название регистра				
	T2CON	TH2	TL2	RCAP2H	RCAP2L
Адрес Исходное состояние	C8H 00H	CDH 00H	CCH 00H	CBH 00H	CAH 00H

Управляющий регистр T2CON служит для программного управления счетчиком/таймером T/C2. Регистр T2CON поддерживает адресацию отдельных битов. Спецификация битов регистра T2CON приведена в табл. 7.6. TH2 и TL2 – старший и младший байты регистра данных T/C2. RCAP2H и RCAP2L – старший и младший байты регистра перезагрузки/ захвата. Программирование режима работы T/C2 производится в соответствии с табл. 7.5.

Таблица 7.5. Выбор режима работы таймера-счетчика T/C2

TCLK (RCLK)	CAP2	TR2	Режим работы
0	0	1	16-разрядная аппаратная перезагрузка
0	1	1	16-разрядный модуль входного захвата
1	x	1	Тактовый генератор UART
x	x	0	Таймер-счетчик отключен

В режиме 16-разрядной аппаратной перезагрузки (рис. 7.2) возможны два варианта перезагрузки, определяемые битом EXEN2 регистра T2CON. В случае EXEN2 = 0 при переполнении таймера-счетчика T/C2 устанавливается флаг переполнения TF2 и происходит загрузка в регистры данных T2H и T2L содержимого регистров RCAP2H и RCAP2L соответственно. В случае EXEN2 = 1 при переполнении происходит все то же самое, но, кроме того, перезагрузка может быть вызвана спадом уровня на внешнем входе T2EX.

В режиме 16-разрядного входного захвата (рис. 7.3) также возможны два варианта захвата. В случае EXEN2 = 0 при переполнении таймера-счетчика T/C2 устанавливается флаг переполнения TF2. При

Таблица 7.6. Регистр управления/статуса таймера **T2CON**

Бит	Позиция	Имя и назначение
TF2	T2CON.7	Флаг переполнения T/C2. Устанавливается аппаратурно. Сброс программный. При RCLK = 1 или TCLK = 1 флаг TF2 не устанавливается
EXF2	T2CON.6	Внешний флаг T/C2. Устанавливается аппаратурно при захвате или перезагрузке по спаду уровня на T2EX при EXEN2 = 1. Сброс программный
RCLK	T2CON.5	При установленном бите приемник последовательного порта в режимах 1 и 3 будет тактироваться импульсами переполнения T/C2. Устанавливается и сбрасывается программно
TCLK	T2CON.4	При установленном бите передатчик последовательного порта в режимах 1 и 3 будет тактироваться импульсами переполнения T/C2. Устанавливается и сбрасывается программно
EXEN2	T2CON.3	При установленном бите разрешен захват или перезагрузка по спаду уровня на T2EX, если T/C2 не используется в этот момент для тактирования последовательного порта. Устанавливается и сбрасывается программно
TR2	T2CON.2	Бит управления пуском/остановом T/C2. Устанавливается (пуск) и сбрасывается (останов) программно
CNT2	T2CON.1	Выбор функции таймера или счетчика событий на ножке T2. Устанавливается (счетчик) и сбрасывается (таймер) программно
CAP2	T2CON.0	Выбор функции захвата или перезагрузки. Устанавливается программно для разрешения захвата по отрицательному переходу на T2EX при EXEN2 = 1. Сбрасывается программно для разрешения автоперезагрузки таймера 2 по его переполнению или отрицательному перепаду на T2EX при EXEN2 = 1. Если RCLK = 1 или TCLK = 1, то этот бит игнорируется и таймер 2 принудительно перезагружается при переполнении

EXEN2 = 1, вдобавок к этому возможен захват текущего состояния таймера спадом уровня на внешнем входе T2EX. При захвате содержимое регистров данных T2H и T2L копируется в регистры RCAP2H и RCAP2L соответственно. Кроме того, при захвате происходит уста-

новка флага EXF2, который можно использовать для организации прерывания по событию захвата.

В режиме программируемого генератора импульсов для тактирования последовательного порта (рис. 7.4) при переполнении T/C2 флаг TF2 не устанавливается и, следовательно, прерывания по переполнению не происходят. В этой связи в данном режиме нет необходимости запрещать прерывания от T/C2. В то же время установка флага EXF2 будет вызывать прерывание, поэтому вход T2EX в этом режиме может быть использован как третий вход запросов внешнего прерывания. Скорость приема/передачи рассчитывается по формуле

$$f = f_{\mathrm{T}}/(32 \times (65536 - \mathrm{RCAP2})), \tag{7.1}$$

где f_{T} – тактовая частота микроконтроллера. В табл. 7.7 приведены рассчитанные значения скорости приема/передачи f, ближайшие к стандартным значениям f_{ST}.

Таблица 7.7. Скорости приема/передачи UART при тактировании от T/C2

f_{ST}, бод	f_{T}, МГц	RCAP2H	RCAP2L	f, бод
19200	12	FFH	ECH	19661
9600	12	FFH	D7H	9591
2400	12	FFH	5CH	2398
1200	12	FEH	B8H	1199
19200	11.0592	FFH	EEH	19200
9600	11.0592	FFH	DCH	9600
2400	11.0592	FFH	70H	2400
1200	11.0592	FFH	E0H	1200

Контроллер векторных прерываний. Встроенный контроллер приоритетных векторных прерываний рассчитан на 9 источников прерываний (табл. 7.8) и 2 уровня приоритета. Для обслуживания системы прерываний предназначены три специальных регистра: два регистра прерываний (IE и IE2) и регистр приоритетов (IP). Спецификация битов этих регистров приведена в табл. 7.9 и 7.10. В регистрах IE и IP полностью сохранена структура и специфика битов, используемых в архитектуре i8051. Новые управляющие биты (EADC, ET2, PSI, PADC, PT2) размещены в позициях, которые не были заняты в оригинальной архитектуре i8051. В отношении этих регистров достигнута полная совместимость снизу вверх. Новый регистр IE2 с адресом A9H содержит только две маски прерываний, относящиеся к контроллеру

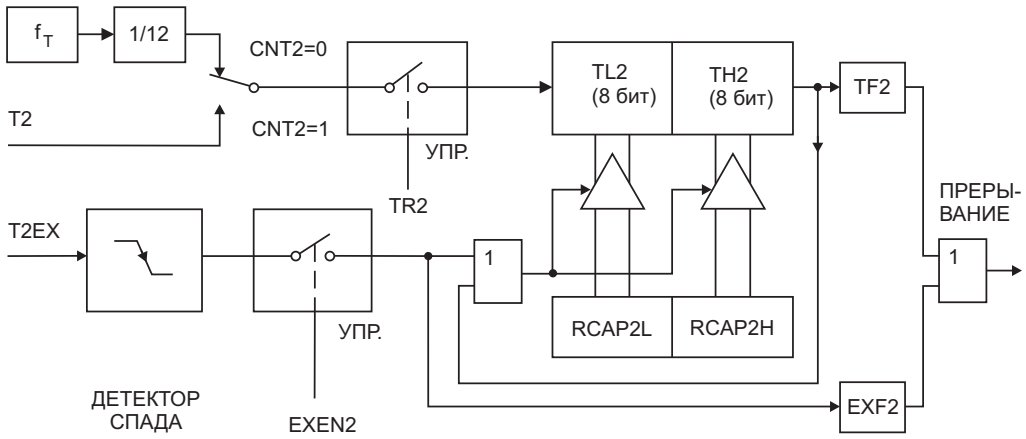


Рис. 7.2. Режим 16-разрядной аппаратной перезагрузки T/C2

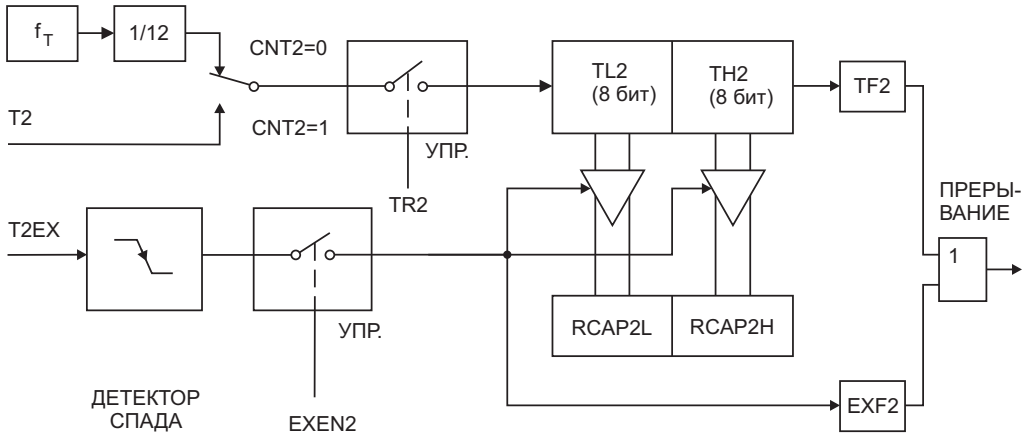


Рис. 7.3. Режим 16-разрядного модуля входного захвата T/C2

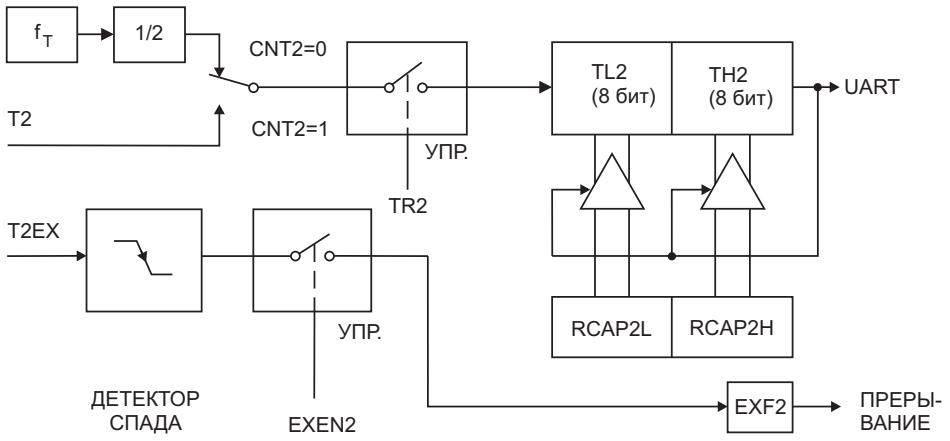


Рис. 7.4. Режим генератора импульсов тактирования UART

I2C/SPI и монитору питания. Остальные биты регистра E2 не используются. Биты всех трех регистров устанавливаются и сбрасываются программно. Регистры IE и IE2 допускают также адресацию отдельных битов. При включении питания все три регистра имеют нулевое

Таблица 7.8. Система прерываний микроконтроллера ADuC812

Источник	Вектор	Приоритет	Назначение
INT0	03H	2	Внешнее прерывание 0
TF0	0BH	4	Переполнение T/C0
INT1	13H	5	Внешнее прерывание 1
TF1	1BH	6	Переполнение T/C1
RI + TI	23H	8	Контроллер UART
TF2 + EXF2	2BH	9	Переполнение/захват T/C2
ADCI	33H	3	Конец преобразования АЦП
I2CI + ISPI	3BH	7	Контроллеры I2C/SPI
PSMI	43H	1	Монитор питания

Таблица 7.9. Регистр масок прерывания **IE** и **IE2**

Символ	Позиция	Назначение
EA	IE.7	Снятие блокировки всех прерываний. Сбрасывается программно для запрета всех прерываний независимо от состояний IE6 – IE0
EADC	IE.6	Бит разрешения прерывания от АЦП
ET2	IE.5	Бит разрешения прерывания от таймера T/C2
ES	IE.4	Бит разрешения прерывания от УАПП
ET1	IE.3	Бит разрешения прерывания от таймера T/C1
EX1	IE.2	Бит разрешения внешнего прерывания INT1
ET0	IE.1	Бит разрешения прерывания от таймера T/C0
EX0	IE.0	Бит разрешения внешнего прерывания INT0
EPSMI	IE2.1	Бит разрешения прерывания от монитора источника питания
ESI	IE2.0	Бит разрешения прерывания от I2C/SPI

Таблица 7.10. Регистр приоритетов прерываний **IP**

Символ	Позиция	Назначение
PSI	IP.7	Бит приоритета контроллера I2C/SPI
PADC	IP.6	Бит приоритета АЦП
PT2	IP.5	Бит приоритета таймера T/C2
PS	IP.4	Бит приоритета УАПП
PT1	IP.3	Бит приоритета таймера T/C1
PX1	IP.2	Бит приоритета внешнего прерывания INT1
PT0	IP.1	Бит приоритета таймера T/C0
PX0	IP.0	Бит приоритета внешнего прерывания INT0

содержимое. В ADuC812 сохранена технология управления приоритетами запросов, используемая в i8051. Для любого источника прерываний x, перечисленного в табл. 7.10, можно назначить высший приоритет (IP.x = 1) или низший приоритет (IP.x = 0). Сначала производят опрос в группе высшего приоритета, потом – в группе низшего приоритета. Порядок опроса в каждой группе соответствует табл. 7.8: от меньших численных значений приоритета – к большим значениям. Монитор источника питания имеет всегда высший приоритет.

Сторожевой таймер. Назначение сторожевого таймера (Watch Dog Timer, WDT) – инициировать аппаратный сброс микроконтроллера при появлении ошибок в его работе. WDT представляет из себя 16-разрядный таймер, для управления работой которого предназначен бит-адресуемый регистр WDCON (C0H). Сторожевой таймер может быть выключен очисткой бита WDE. При WDE = 1 сторожевой таймер включен. Если прикладная программа не установит биты обновления (WDR1, WDR2) в течение предустановленного интервала времени, то будет установлен бит состояния WDS = 1 и инициирован аппаратный сброс. При WDT-сбросе бит состояния WDS не очищается. Предустановленный интервал задается трехбитным числом n, размещенным в позициях ($PRE_2PRE_2PRE_0$). Величина интервала t в миллисекундах вычисляется по формуле

$$t = 2^{(n+4)}. \qquad (7.2)$$

Рассмотрим пример. Пусть предустановленный интервал будет 2.048 с.

7	6	5	4	3	2	1	0
PRE_2	PRE_1	PRE_0	–	WDR_1	WDR_2	WDS	WDE

```
MOV   WDCON,#0E0H    ; Установка t = 2.048 с
SETB  WDE            ; Запуск WDT
```

Не позднее чем через 2.048 с после запуска WDT необходимо установить биты обновления, иначе будет инициирован аппаратный сброс микроконтроллера.

```
SETB  WDR1    ; Установка бита WDR1
SETB  WDR2    ; Установка бита WDR2
```

Биты обновления необходимо устанавливать только в такой последовательности.

Для отладки и апробации компания Analog Devices выпустила средство отладки QuickStart, которое содержит:

- отладочную плату;
- кабель подключения к Com-порту персонального компьютера и блок питания;
- две микросхемы ADuC812;
- компакт-диск (CD-ROM), содержащий следующее программное обеспечение:
  - кросс-ассемблер ASM51 для MCS51;
  - программный симулятор ADSIM812 для ADuC812, работающий под управлением Windows;
  - отладчик DEBUG812 для ADuC812;
  - загрузчик DLOAD812 для ADuC812, работающий по последовательному каналу;
  - полный комплект документации для ADuC812 в pdf-формате.

Основные области применения: интеллектуальные сенсоры (IEEE 1451.2), батарейные системы (портативные PC, инструмент, мониторы), системы слежения, системы сбора информации и средства коммуникации.

7.5.2. Микроконтроллер ADuC824

ADuC824 — микросхема, которая применяется в индустриальных интеллектуальных датчиках. Для краткости остановимся на отличительных чертах и характеристиках данной микросхемы. В аналоговой части вместо 8-канального 12-разрядного АЦП последовательных приближений применены два сигма-дельта АЦП. Один из них (основной канал) имеет реальное разрешение более 19 разрядов при входном сигнале ± 2.56 В и снабжен программируемым усилителем, позволяющим получить реальное 13-разрядное разрешение при входном сигнале ± 20 мВ. Дополнительный канал характеризуется 16-разрядным разрешением. Кроме того, на кристалле имеется два согласованных (с разбросом не хуже 0.1 %) стабильных источника тока величиной 200 мкА, которые могут служить для питания внешних датчиков.

Блок ЦАП состоит из одного прецизионного 12-разрядного ЦАП с выходом по напряжению, который может работать либо в 8-разрядном, либо в 12-разрядном режиме с диапазоном выходных сигналов от 0 до 2.5 В (при использовании внутреннего источника опорного напряжения), либо от 0 до напряжения источника питания на нагрузку до 10 кОм/100 пФ.

Еще одной отличительной особенностью данной микросхемы является ее малое энергопотребление (всего 3 мА при питании от 3-вольтового источника), что дает возможность ее реального использования в

индустриальных приложениях с питанием от токовой петли. Это стало возможным за счет применения специального тактового генератора, позволяющего “завести” микросхему от стандартного резонатора с частотой 32768 кГц.

Остальные узлы ADuC824 и комплект инструментальных средств имеют практически ту же структуру и функции, что и у ADuC812.

7.5.3. Микроконтроллер ADuC842

Микроконвертор ADuC842 – полностью интегрированная 12-битная однокристальная система сбора данных с высокоскоростным ядром 8052 (машинный цикл равен тактовому циклу), являющаяся развитием системы ADuC832. Как и другие приборы этого семейства, ADuC842 имеет высокоточные АЦП, ЦАП и перепрограммируемый микроконтроллер. Прибор выпускается в 52-выводном PQFP-корпусе или 56-выводном CSP-корпусе (8×8 мм^2) и имеет напряжение питания 3 В или 5 В.

Отличительные особенности микроконтроллера ADuC842:

– 8-канальный 12-битный АЦП (400 тыс. преобразований в секунду) с самокалибровкой;

– два 12-битных ЦАП с потенциальными выходами и динамическим диапазоном, равным напряжению питания;

– два выхода ШИМ/ 16-битного сигма-дельта АЦП;

– скоростное ядро промышленного стандарта 8052 с машинным циклом, равным тактовому, и производительностью 16.7 MSPS;

– 62 Кб FLASH памяти программы;

– 4 Кб FLASH памяти данных;

– 2 Кб статического ОЗУ (в дополнение к 256 байтам ядра 8052);

– задающий генератор со схемой фазовой автоподстройки частоты и программируемой частотой, работающий с 32 кГц кварцевым резонатором;

– температурный датчик;

– прецизионный источник опорного напряжения (ИОН) (20 ppm/°C), последовательные интерфейсы (UART, I2C и SPI), сторожевой таймер, таймер измерения длительности, монитор напряжения питания, схема сброса по включению питания (POR) и т.д.;

– встроенная система загрузки, отладки и эмуляции;

– совместимость по выводам с прибором ADuC832.

Основные области применения: управление питанием лазера в оптических системах связи, управление смещением усилителя в базовых станциях, прецизионный инструмент, интеллектуальные датчики, системы съема информации, сбора информации и связи.

8. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ ПРОГРАММ ДЛЯ МИКРОКОНТРОЛЛЕРОВ

Разработка и отладка программного обеспечения для однокристальных микроконтроллеров имеют определенную специфику. Во-первых, резидентные ресурсы микроконтроллера (объема памяти, быстродействие), как правило, сравнительно невелики и их совершенно недостаточно для размещения и функционирования даже простейших сервисных программ (редактор текста, транслятор и отладочный монитор), необходимых для написания и отладки программы. Во-вторых, некоторые архитектурные особенности (раздельные области памяти для хранения программ и данных, устройства защиты памяти программ) микроконтроллеров затрудняют или делают просто невозможным редактирование (написание, отладку) программ непосредственно в микроконтроллере. Это обусловливает необходимость применения специальных инструментальных средств для разработки и отладки программ.

В случае построения средств разработки и отладки на базе универсального компьютера становится возможным существенно облегчить разработку программ: использовать языки высокого уровня (Си, Паскаль), построить дружественный интерфейс, использовать принципы объектного и визуального программирования и т.д. Рассмотрим различные варианты построения инструментальных средств разработки и отладки.

К числу основных инструментальных средств отладки относятся внутрисхемные эмуляторы, программные симуляторы, платы развития, мониторы отладки, эмуляторы ПЗУ и некоторые другие средства. Данный список далеко не исчерпывает всех типов существующих инструментальных средств отладки. Кроме указанных, существуют и комбинированные устройства и наборы, которые позволяют компенсировать недостатки основных средств, взятых порознь.

8.1. Инструментальные средства отладки

8.1.1. Внутрисхемный эмулятор

Внутрисхемный эмулятор – программно аппаратное средство, способное замещать собой эмулируемый (моделируемый) процессор в реальной схеме. Внутрисхемный эмулятор – это наиболее мощное и универсальное отладочное средство.

По сути дела, *хороший* внутрисхемный эмулятор делает процесс функционирования отлаживаемого контроллера прозрачным, т.е. легко контролируемым, произвольно управляемым и модифицируемым по воле разработчика.

Особенности внутрисхемных эмуляторов

Обычно стыковка внутрисхемного эмулятора с отлаживаемой системой производится при помощи эмуляционного кабеля со специальной эмуляционной головкой. Эмуляционная головка вставляется вместо микроконтроллера в отлаживаемую систему. Если микроконтроллер невозможно удалить из отлаживаемой системы, то использование эмулятора возможно, только если этот микроконтроллер имеет отладочный режим, при котором все его выводы находятся в третьем состоянии. В этом случае для подключения эмулятора используют специальный адаптер-клипсу, который подключается непосредственно к выводам эмулируемого микроконтроллера.

Функционально внутрисхемные эмуляторы делятся на стыкуемые с внешним компьютером (обычно это бывает IBM PC) и функционирующие автономно.

Автономные внутрисхемные эмуляторы имеют индивидуальные вычислительные ресурсы, средства ввода-вывода, не требуют для своей нормальной работы стыковки с какими-либо внешними вычислительными средствами, но за это пользователю приходится расплачиваться либо существенно более высокой ценой, либо пониженными функциональными и сервисными возможностями по сравнению с аналогичными моделями, стыкуемыми с IBM PC.

Набор функциональных возможностей, которые предоставляют разработчику внутрисхемные эмуляторы, весьма широк и включает в себя практически все разнообразие функциональных модулей средств разработок.

Наличие в программной оболочке эмулятора встроенного редактора, встроенного менеджера проектов и системы управления может существенно облегчить работу разработчика. В этом случае стирается грань между написанием программы, ее редактированием и отладкой. Переход от редактирования исходного текста к отладке – началу работы собственно эмулятора – и обратно происходит *прозрачно* и синхронно с активизацией соответствующих окон, менеджер проектов автоматически запускает компиляцию по мере необходимости и активизирует соответствующие окна программного интерфейса.

При работе внутрисхемного эмулятора в составе интегрированной среды столь же просто можно осуществить и переход к отладке проекта с помощью имеющегося отладчика-симулятора или приступить к занесению в ПЗУ микроконтроллера отлаженной программы.

Некоторые модели внутрисхемных эмуляторов могут предоставлять пользователям и другие дополнительные возможности. Среди них отметим одну, хотя и достаточно специфическую, но в ряде случаев имеющую принципиальное значение: возможность построения многоэмуляторных комплексов, необходимых для отладки мультипроцессорных систем. Отличительной особенностью такого комплекса является возможность синхронного управления (с одного компьютера) несколькими эмуляторами.

К достоинствам внутрисхемных эмуляторов следует отнести:

- широкий набор функциональных возможностей, что делает внутрисхемные эмуляторы наиболее мощным и универсальным средством отладки;
- работу внутрисхемного эмулятора в реальной схеме электронного блока, в котором предполагается работа микроконтроллера;
- большая гибкость моделирования временных и электрических характеристик микроконтроллера, что связано с преимущественно программным методом их моделирования.

Однако внутрисхемные эмуляторы имеют и недостатки. Основным из них является трудность программного моделирования электрических сигналов на выводах микроконтроллера в реальном масштабе времени. Для адекватного моделирования быстродействие моделирующего процессора или компьютера должно быть существенно выше, чем эмулируемого микроконтроллера, что достижимо далеко не всегда, особенно в случае эмуляции современных высокопроизводительных микроконтроллеров.

Кроме того, даже в случае работы в замедленном масштабе времени различные модели внутрисхемных эмуляторов могут иметь разного рода ограничения по контролю и управлению функционированием отлаживаемых устройств, что связано с трудностью их моделирования. Например, это может быть некорректная обработка прерываний в пошаговом режиме или запрет на использование последовательного порта и т.п.

Возможности *реального* внутрисхемного эмулятора проиллюстрируем на примере модели PICE-51.

Внутрисхемный эмулятор PICE-51

PICE-51 (Phyton Inc.) – это внутрисхемный эмулятор 8-разрядных микроконтроллеров семейства 8051. Он аннотируется как эмулятор нового поколения, созданный с применением новых технологий разработки аппаратуры и программного обеспечения.

Применение программируемых матриц (ПЛИС) большой емкости позволило резко сократить размеры эмулятора без какого-либо ущер-

ба его функциональным возможностям, минимизировать отклонения электрических и частотных характеристик эмулятора от характеристик эмулируемого процессора и тем самым добиться максимальной точности эмуляции на частотах до 30 МГц при напряжениях питания от 3.3 до 5 В.

Перезагружаемая аппаратная структура эмулятора обеспечивает эмуляцию практически всех микроконтроллеров семейства 8051 как отечественного производства, так и фирм Intel, Philips, Siemens, Atmel, Dallas, Temic, OKI, AMD, MHS и др.

Мощный программный интерфейс в среде Windows представляет собой интегрированную среду разработки, поддерживающую все этапы разработки программного обеспечения (от написания исходного текста программы до ее компиляции и отладки). Программа поддержки эмулятора ориентирована на отладку программ на языке высокого уровня по исходному тексту.

Эмулятор состоит из основной платы размером 80×76 мм^2, сменного адаптера под конкретный процессор и сменной эмуляционной головки под конкретный тип корпуса. На основной плате реализованы трассировщик и процессор точек останова. Плата сменного адаптера содержит эмулирующий процессор под конкретный тип микроконтроллера. Эмуляционные головки обеспечивают установку эмулятора в колодки DIP и PLCC на плате пользователя. Питание эмулятора осуществляется от блока питания +5 В, 0.5 А или непосредственно от отлаживаемого устройства. Связь с компьютером – по гальванически развязанному каналу RS-232C на скорости 115 Кбод.

Характеристики аппаратной части эмулятора:

- точная эмуляция – отсутствие каких-либо ограничений на использование программой пользователя ресурсов микроконтроллера;
- до 256 Кбайт эмулируемой памяти программ и данных. Поддержка банкированной модели памяти. Распределение памяти между эмулятором и устройством пользователя с точностью до одного байта;
- до 512 Кбайт аппаратных точек останова по доступу к памяти программ и данных;
- аппаратная поддержка для отладки программ на языках высокого уровня;
- трассировка 8 произвольных внешних сигналов;
- 4 выхода синхронизации аппаратуры пользователя;
- трассировщик реального времени с буфером объемом от 16 до 64 Кб фреймов по 64 бита с доступом *на лету*. Трассировка адреса, данных, сигналов управления, таймера реального времени и 8 внешних сигналов пользователя;
- программируемый фильтр трассировки;

- аппаратный процессор точек останова с возможностью задания сложного условия останова эмуляции по комбинации сигналов адреса, данных, управления, 8 внешних сигналов, таймера реального времени, счетчиков событий и таймера задержки;
- четыре комплексных точки останова, которые могут быть использованы независимо или в комбинациях по условиям AND/OR/IF-THEN;
- 48-разрядный таймер реального времени;
- прозрачная эмуляция – доступ *на лету* к эмулируемой памяти, точкам останова, процессору точек останова, буферу трассировки, таймеру реального времени;
- управляемый генератор тактовой частоты для эмулируемого процессора. Возможность плавного изменения тактовой частоты от 500 кГц до 40 МГц;
- гальванически развязанный от компьютера канал связи RS232C со скоростью обмена 115 Кбод;
- встроенная система самодиагностики аппаратуры эмулятора.

Характеристики программного обеспечения:

- программное обеспечение ориентировано на работу в среде Windows на IBM-совместимых компьютерах с процессорами типа Pentium;
- встроенный многооконный редактор предназначен для написания исходных текстов программ. Редактор поддерживает операции с блоками текста, поиск/замену, цветовое выделение синтаксических конструкций языка ассемблера и Си;
- встроенный менеджер проектов обеспечивает автоматическую компиляцию программ. Все опции задаются в диалоговой форме. Переход от редактирования исходного текста к отладке и обратно происходит *прозрачно*, т.е. менеджер проектов автоматически запускает компиляцию проекта при необходимости;
- PICE-51 обеспечивает символьную отладку и отладку по исходному тексту для программ, созданных с помощью следующих компиляторов:
  - ассемблер ASM51 фирмы Intel;
  - ассемблер MCA-51 фирмы “Фитон/МикроКосм”;
  - компилятор PL/M фирмы Intel;
  - ассемблер и компилятор Си фирмы IAR Systems;
  - ассемблер и компилятор Си фирмы Avocet Systems Inc./HiTech;
  - ассемблер и компилятор Си фирмы Keil Software Inc.
- автоматическое сохранение и загрузка файлов конфигурации аппаратуры, интерфейса и опций отладки. Обеспечивается совместимость файлов конфигурации с симулятором PDS-51. Обеспечена переносимость проектов между эмулятором PICE-51 и симулятором PDS-51;

- возможность настройки цветов, шрифтов и других параметров для всех окон одновременно и для каждого окна в отдельности;
- эмулятор снабжен печатным руководством по эксплуатации и контекстным электронным руководством, в которых детально описаны его принципы работы, команды, меню, “горячие” клавиши.

8.1.2. Программный симулятор

Симулятор – программное средство, способное имитировать работу микроконтроллера и его памяти. Как правило, симулятор содержит в своем составе отладчик, а также модель процессора и памяти. Более продвинутые симуляторы содержат в своем составе модели встроенных периферийных устройств, таких как таймеры, порты, АЦП, системы прерываний.

Симулятор должен уметь загружать файлы программ во всех популярных форматах, максимально полно отображать информацию о состоянии ресурсов симулируемого микроконтроллера, а также предоставлять возможности по симуляции выполнения загруженной программы в различных режимах. В процессе отладки модель *выполняет* программу, и на экране компьютера отображается текущее состояние модели. Загрузив программу в симулятор, пользователь имеет возможность запускать ее в пошаговом или непрерывном режиме, задавать условные и безусловные точки останова, контролировать и свободно модифицировать содержимое ячеек памяти и регистров симулируемого микропроцессора. С помощью симулятора можно быстро проверить логику выполнения программы, правильность выполнения арифметических операций.

В зависимости от класса используемого отладчика, различные симуляторы могут поддерживать высокоуровневую символьную отладку программ. Некоторые модели симуляторов могут содержать ряд дополнительных программных средств, таких, например, как интерфейс внешней среды, встроенную интегрированную среду разработки.

В реальной системе микроконтроллер обычно занимается считыванием информации с подключенных внешних устройств (датчиков), обработкой этой информации и выдачей управляющих воздействий на исполнительные устройства. Для того чтобы в симуляторе, не обладающем интерфейсом внешней среды, смоделировать работу датчика, нужно вручную изменять текущее состояние модели периферийного устройства, к которому в реальной системе подключен датчик. Если, например, при приеме байта через последовательный порт взводится некоторый флажок, а сам байт попадает в определенный регистр, то оба эти действия нужно производить в таком симуляторе вручную. Наличие же интерфейса внешней среды позволяет пользователю создавать и гибко использовать модель внешней среды микроконтроллера,

функционирующую и взаимодействующую с отлаживаемой программой по заданному алгоритму.

Очевидной особенностью программных симуляторов является то обстоятельство, что исполнение программ, загруженных в симулятор, происходит в масштабе времени, отличном от реального. Однако низкая цена, возможность ведения отладки даже в условиях отсутствия макета отлаживаемого устройства делают программные симуляторы весьма эффективным средством отладки. Отдельно необходимо подчеркнуть, что существует целый класс ошибок, которые могут быть обнаружены только при помощи симулятора.

8.1.3. Плата развития

Платы развития (Evaluation Boards) являются своеобразными конструкторами для макетирования прикладных систем. В последнее время при выпуске новой модели кристалла микроконтроллера фирма-производитель обязательно выпускает и соответствующую плату развития.

Обычно это печатная плата с установленным на ней микроконтроллером, а также вся необходимая стандартная обвязка. На этой плате также устанавливают схемы связи с внешним компьютером. Как правило, там же имеется свободное поле для монтажа прикладных схем пользователя. Иногда имеется уже готовая разводка для установки дополнительных устройств, рекомендуемых фирмой, например ПЗУ, ОЗУ, дисплей на жидких кристаллах (ЖКИ-дисплей), клавиатура, АЦП и др. Кроме учебных или макетных целей, такие доработанные пользователем платы стало выгодно (экономия времени) использовать в качестве одноплатных контроллеров, встраиваемых в малосерийную продукцию (5…20 шт.).

Для большего удобства платы развития комплектуются еще и простейшим средством отладки на базе монитора отладки. Однако здесь проявились два разных подхода: один используется для микроконтроллеров, имеющих внешнюю шину, а второй – для микроконтроллеров, не имеющих внешней шины.

В первом случае отладочный монитор поставляется фирмой в виде микросхемы ПЗУ, которая вставляется в специальную розетку на плате развития. Плата также имеет ОЗУ для программ пользователя и канал связи с внешним компьютером или терминалом. Примером здесь может служить плата развития фирмы Intel для микроконтроллера 8051.

Во втором случае плата развития имеет встроенные схемы программирования внутреннего ПЗУ микроконтроллера, которые управляются от внешнего компьютера. В этом случае программа монитора просто заносится в ПЗУ микроконтроллера совместно с прикладны-

ми кодами пользователя. Прикладная программа при этом специально должна быть подготовлена: в нужные ее места вставляют вызовы отладочных подпрограмм монитора. Затем осуществляется пробный прогон. Для того чтобы внести в программу исправления, пользователю надо стереть ПЗУ и произвести повторную запись. Готовую прикладную программу получают из отлаженной путем удаления всех вызовов мониторных функций и самого монитора отладки.

Важно отметить, что платы развития могут комплектоваться не только монитором, но иногда еще и программами отладки, которые запускаются на внешнем компьютере в связке с монитором. Эти программы в последнее время заметно усложнились и зачастую имеют высокопрофессиональный набор отладочных функций, например отладчик-симулятор или различные элементы, присущие в чистом виде интегрированным средам разработки. В состав поставляемых комплектов могут входить и программы прикладного характера, наиболее часто встречающиеся на практике. Возможности по отладке, предоставляемые комплектом *плата развития плюс монитор*, безусловно, не столь универсальны, как возможности внутрисхемного эмулятора, да и некоторая часть ресурсов микропроцессора в процессе отладки отбирается для работы монитора. Тем не менее наличие законченного набора готовых программно-аппаратных средств, позволяющих без потери времени приступить к монтажу и отладке прикладной системы, во многих случаях является решающим фактором. Особенно если учесть, что стоимость такого комплекта несколько меньше, чем стоимость более универсального эмулятора.

8.1.4. Отладочный монитор

Отладочный монитор – специальная программа, загружаемая в память отлаживаемой системы. Она вынуждает процессор пользователя выполнять, кроме прикладной задачи, еще и отладочные функции:

- загрузку прикладных кодов пользователя в свободную от монитора память;
- установку точек останова;
- запуск и останов загруженной программы в реальном времени;
- проход программы пользователя по шагам (часть функций трассировщика);
- просмотр, редактирование содержимого памяти и управляющих регистров.

Программа монитора обязательно должна работать в связке с внешним компьютером или пассивным терминалом, на которых и происходит визуализация и управление процессом отладки. Отметим,

что отладочные мониторы используют тот процессор, который уже стоит на плате пользователя.

Достоинством этого подхода являются очень малые затраты при сохранении возможности вести отладку в реальном времени.

Главным недостатком является отвлечение ресурсов микроконтроллера на отладочные и связные процедуры, например: монитор занимает некоторый объем памяти, прерывания, последовательный канал. Объем отвлекаемых ресурсов зависит от искусства разработчика монитора. В последнее время появились изделия, которые практически не занимают аппаратных ресурсов процессора, о них рассказано в п. 8.1.5.

Стало обычной практикой, что каждая фирма-разработчик семейства микроконтроллеров выпускает и вариант отладочного монитора, который обычно поставляется вместе с платами развития.

8.1.5. Эмулятор ПЗУ

Эмулятор ПЗУ – программно аппаратное средство, позволяющее замещать ПЗУ на отлаживаемой плате и подставляющее вместо него ОЗУ, в которое может быть загружена программа с компьютера через один из стандартных каналов связи. Это устройство позволяет пользователю избежать многократных циклов перепрограммирования ПЗУ. Эмулятор ПЗУ имеет смысл только для микроконтроллеров, которые в состоянии обращаться к внешней памяти программ. Это устройство сравнимо по сложности и по стоимости с платами развития. Оно имеет одно большое достоинство – универсальность. Эмулятор ПЗУ может работать с любыми типами микроконтроллеров.

Ранние эмуляторы ПЗУ позволяли только загружать программу, запускать ее и останавливать, используя общий сброс. Затем появились усложненные модели с аппаратной выработкой сигналов трассировки на осциллограф по достижении определенного адреса. Эмулируемая память в таких изделиях была доступна для просмотра и модификации, но очень важный контроль за внутренними управляющими регистрами микроконтроллера был до недавнего времени невозможен.

Появились модели интеллектуальных эмуляторов ПЗУ, которые позволяют *заглядывать* внутрь микроконтроллера на плате пользователя и по управлению отладкой стали похожими на внутрисхемный эмулятор. Фирма Cactus даже представляет свой фактически интеллектуальный эмулятор ПЗУ как внутрисхемный эмулятор ряда микропроцессоров, настолько они малоразличимы. В действительности, процессор здесь не замещается, а используется тот, что стоит на плате пользователя.

Интеллектуальные эмуляторы ПЗУ представляют собой гибрид из обычного эмулятора ПЗУ, монитора отладки и схем быстрого переключения шины с одного на другой. Этим создается эффект, как если бы монитор отладки был установлен на плате пользователя, и при этом он не занимает у микроконтроллера никаких аппаратных ресурсов, кроме небольшой зоны программных шагов, примерно 4К. Например, такое устройство разработала фирма “Фитон” для всех существующих и будущих микроконтроллеров, которые имеют ядро от 8051, но дополнительно насыщены различными устройствами ввода-вывода. Это устройство поддерживает множество самых разных микроконтроллеров фирм Philips, Siemens, OKI.

8.2. Типичные функциональные модули средств разработки и отладки

Любое из перечисленных инструментальных средств состоит из нескольких взаимодействующих (программных либо аппаратных) функциональных модулей. Каждый из них обеспечивает определенный круг сервисных услуг при разработке и отладке программ. Некоторые из модулей специфичны для того или иного типа инструментальных средств разработки, другие используются практически во всех вариантах систем разработки программ для микроконтроллеров.

Система разработки содержит следующий минимальный набор функциональных блоков: отладчик, узел эмуляции микроконтроллера, эмуляционную память, подсистему точек останова.

Более продвинутые модели могут содержать дополнительно процессор точек останова, трассировщик, профилировщик (анализатор эффективности программного кода), таймер реального времени, программно-аппаратные средства, обеспечивающие возможность чтения и модификации ресурсов эмулируемого процессора *на лету*, т.е. в процессе выполнения программы пользователя в реальном времени, программно - аппаратные средства, обеспечивающие синхронное управление, необходимое для эмуляции в мультипроцессорных системах, интегрированную среду разработки.

Отладчик

Отладчик является своеобразным мостом между разработчиком и отладочным средством. Состав и объем информации, проходящей через средства ввода-вывода, доступность ее для восприятия, контроля и, при необходимости, для коррекции и модификации напрямую зависят от свойств и качества отладчика.

Хороший отладчик позволяет осуществлять загрузку отлаживаемой программы в память системы, вывод на монитор состояния и содержимого всех регистров и памяти, а также, при необходимости, их модификацию, управление процессом эмуляции.

Более мощные отладчики, обычно их называют высокоуровневыми (High-Level Debuggers), помимо этого позволяют вести символьную отладку благодаря тому, что отладчик *знает* адреса всех символьных переменных, массивов и структур (за счет использования специальной информации, поставляемой компилятором). При этом пользователь может оперировать более приемлемыми для человека символьными именами, не утруждая себя запоминанием их адресов, контролировать и анализировать не только дизассемблированный текст, но и исходный текст программы, написанной на языке высокого уровня, и даже с собственными комментариями.

Такой отладчик позволяет пользователю одновременно контролировать ход выполнения программы и видеть соответствие между исходным текстом, образом программы в машинных кодах и состоянием всех ресурсов эмулируемого микроконтроллера.

Следует отметить, что высокоуровневый отладчик обеспечивает выполнение всех своих функций только в том случае, если используется кросс-компилятор, поставляющий полную и правильную отладочную информацию (не все компиляторы, особенно их пиратские версии, поставляют такую информацию), и при этом формат ее представления должен быть известен отладчику.

Узел эмуляции микроконтроллера

Узел эмуляции микроконтроллера – модуль, позволяющий моделировать микроконтроллер.

Данный блок необходим в системах разработки на основе внутрисхемных эмуляторов и симуляторов, в других вариантах средств разработки в системах присутствует реальный микроконтроллер, и поэтому его эмуляция не нужна.

Как правило, при эмуляции микроконтроллера предусматривается возможность запуска программ, их останова и выполнения с различной скоростью, в том числе и в пошаговом режиме. Также обычной является функция просмотра и изменение содержимого внутренних регистров микроконтроллера и состояния его внешних выводов.

Эмуляционная память

Наличие эмуляционной памяти дает возможность использовать ее в процессе отладки вместо ПЗУ в отлаживаемой системе и, более того, отлаживать программу без использования реальной системы или

ее макета. При необходимости внесения изменений в отлаживаемую программу достаточно загрузить новую или модифицированную программу в память эмулятора, вместо того чтобы заниматься перепрограммированием ПЗУ.

Существуют модели эмуляторов, которые позволяют пользователю *подставлять* вместо ПЗУ эмуляционную память не только целиком, но и поблочно (в некоторых моделях минимальный размер блока может достигать одного байта), в порядке, определенном пользователем. Для этого пользователю достаточно задать распределение памяти данных и памяти программ, в соответствии с которым процессор будет получать доступ и к содержимому ПЗУ в отлаживаемой системе, и к содержимому эмуляционной памяти внутрисхемного эмулятора. Такая память обычно называется памятью с возможностью мэппинга.

Подсистема точек останова

Подсистема точек останова – набор средств, управляющий процессом выполнения программы. Он позволяет останавливать выполняемую в реальном (или приближенном к реальному) масштабе времени программу при выполнении команды, размещенной по заданному адресу. Частный случай работы системы точек останова – пошаговое выполнение. Другие, часто используемые случаи – останов при проведении операций ввода-вывода.

В том или ином виде данный модуль присутствует как в системах с эмуляцией или симуляцией микроконтроллера, так и в системах с реальным микроконтроллером. В последнем случае при достижении точки останова микроконтроллер останавливается и/или переводится на выполнение специальной мониторинговой программы, при помощи которой можно зафиксировать или изменить состояние микроконтроллера перед последующим стартом.

Процессор точек останова

Более развитый набор сервисных функций аналогичного назначения имеет процессор точек останова.

Процессор точек останова позволяет останавливать выполнение программы или выполнять иные действия, например запускать или останавливать трассировщик при выполнении заданных пользователем условий. В отличие от механизма обычных точек останова, процессор точек останова позволяет формировать и отслеживать условия практически любой степени сложности, и при этом эмулируемый процесс не выводится из масштаба реального времени.

Трассировщик

В сущности, трассировщик представляет собой логический анализатор, работающий синхронно с процессором и фиксирующий поток выполняемых инструкций и состояния выбранных внешних сигналов. Существуют модели внутрисхемных эмуляторов, которые позволяют трассировать не только внешние сигналы, но и состояния внутренних ресурсов микроконтроллера, например регистров. Такие эмуляторы используют специальные версии микроконтроллеров (эмуляционные кристаллы).

Профилировщик

Профилировщик (иначе анализатор эффективности программного кода) позволяет получить по результатам прогона отлаживаемой программы следующую информацию: количество обращений к различным участкам программы, время, затраченное на выполнение различных участков программы.

Анализ статистической информации, поставляемой профилировщиком, позволяет легко выявлять *мертвые* или перенапряженные участки программ и в результате оптимизировать структуру отлаживаемой программы.

Интегрированная среда разработки

Интегрированная среда разработки – это совокупность программных средств, поддерживающая все этапы разработки программного обеспечения, от написания исходного текста программы до ее компиляции и отладки, и обеспечивающая простое и быстрое взаимодействие с другими инструментальными средствами (программным отладчиком-симулятором, внутрисхемным эмулятором, эмулятором ПЗУ и программатором).

Строго говоря, интегрированные среды разработки не относятся к числу средств отладки, тем не менее обойти вниманием данный класс программных средств, существенно облегчающий и ускоряющий процесс разработки и отладки микропроцессорных систем, было бы неправильно.

При традиционном подходе начальный этап написания программы строится следующим образом:

• исходный текст набирается при помощи какого-либо текстового редактора. По завершении набора работа с текстовым редактором прекращается и запускается кросс-компилятор. Как правило, вновь написанная программа содержит синтаксические ошибки, и компилятор сообщает о них на консоль оператора;

• вновь запускается текстовый редактор, и оператор должен найти и устранить выявленные ошибки; при этом сообщения о характере ошибок, выведенные компилятором, уже не видны, так как экран занят текстовым редактором.

Этот цикл может повторяться не один раз. Если программа имеет большой объем, собирается из различных частей и подвергается длительному редактированию или модернизации, то даже этот начальный этап может потребовать много сил и времени. После этого наступает этап отладки программы, и к редактору с компилятором добавляется эмулятор или симулятор, за работой которого хотелось бы следить прямо по тексту программы в текстовом редакторе.

Избежать большого объема однообразных действий и тем самым существенно повысить эффективность процесса разработки и отладки позволяют интегрированные среды (оболочки) разработки (Integrated Development Environment, IDE).

Работа в интегрированной среде дает программисту следующие возможности:

- использовать встроенный многофайловый текстовый редактор, специально ориентированный на работу с исходными текстами программ;
- проводить диагностику выявленных при компиляции ошибок, которые выводятся в многооконном режиме одновременно с исходным текстом программы, доступном для редактирования;
- организовать и вести параллельную работу над несколькими проектами (менеджер проектов позволяет использовать любой проект в качестве шаблона для вновь создаваемого проекта);
- заново компилировать только редактировавшиеся модули;
- загружать отлаживаемую программу в имеющиеся средства отладки и работать с ними без выхода из оболочки;
- подключать к оболочке практически любые программные средства.

В последнее время функции интегрированных сред разработки становятся стандартной принадлежностью программных интерфейсов эмуляторов и отладчиков-симуляторов.

Подобные функциональные возможности, в сочетании с дружественным интерфейсом, в состоянии существенно увеличить скорость разработки программ для микроконтроллеров и процессоров цифровой обработки сигналов.

8.3. Программные средства для MCS51

В данном разделе перечислены лишь некоторые из существующих программных средств для разработки программного обеспечения для микроконтроллеров семейства MCS51.

8.3.1. Базовые программные средства

Макроассемблер A51

Макроассемблер A51 фирмы Keil Software специально разработан для семейства микроконтроллеров 8051. Ассемблер в основном применяется при написании фрагментов программ, наиболее критичных к скорости, размеру кода и возможностям аппаратного управления. В ассемблер включен макроязык, использование которого ускоряет разработку и экономит общее время проектирования. Макроязык позволяет также осуществлять доступ ко всем ресурсам микроконтроллеров с использованием символьных обозначений регистров.

Макроассемблер A51 совместим с ассемблером ASM-51 Intel для всего семейства микроконтроллеров Intel 8051. Ассемблер транслирует символическую мнемонику в перемещаемый объектный код, имеющий высокое быстродействие и малый размер. A51 транслирует исходный файл ассемблера в перемещаемый объектный модуль. При отладке или при включенной опции *Debug* этот объектный файл будет содержать полную символическую информацию для отладчика/имитатора или внутрисхемного эмулятора.

Ассемблер A51 поддерживает все разновидности соответствующего семейства микроконтроллеров. В ассемблер встроено стандартное описание регистров специальных функций. В то же время специальная директива NOMOD позволяет аннулировать эти определения путем подключения файла заголовка для конкретного микроконтроллера. Ассемблер содержит файлы заголовков для всех основных членов семейства.

Оптимизирующий кросс-компилятор C51

Язык C – универсальный язык программирования, который обеспечивает эффективность кода, содержит элементы структурного программирования и имеет богатый набор операторов. Универсальность, отсутствие ограничений реализации делают язык C удобным и эффективным средством программирования для широкого круга задач. Множество прикладных программ может быть написано легче и эффективнее на языке C, чем на других более специализированных языках.

Компилятор C51 – полная реализация стандарта ANSI (Американского национального института стандартов), насколько это возможно для архитектуры Intel 8051. В C51 входят различные расширения, связанные с реализацией компилятора для микропроцессора с гарвардской архитектурой, содержащей два адресных пространства – для кода и данных, и для более эффективной работы в условиях

ограниченных ресурсов микроконтроллеров. В исполняющей системе (библиотеке) отсутствуют функции, связанные с вызовами операционной системы (операции с файлами и т.д.). C51 генерирует код для всего семейства микроконтроллеров Intel 8051. Транслятор сочетает гибкость программирования на языке С с эффективностью кода и быстродействием ассемблера.

Использование языка высокого уровня С имеет следующие преимущества над программированием на ассемблере:

- не требуется глубокого знания системы команд процессора, элементарное знание архитектуры Intel 8051 желательно, но не необходимо;
- распределение регистров и способы адресации управляются полностью транслятором;
- лучшая читаемость программы, используются ключевые слова и функции, которые более свойственны человеческой мысли;
- время разработки программ и их отладки значительно короче в сравнении с программированием на ассемблере;
- библиотечные файлы содержат много стандартных подпрограмм, которые могут быть включены в прикладную программу;
- существующие программы могут многократно использоваться в новых программах с помощью модульных методов программирования.

Редактор связей L51

Редактор связей, или компоновщик, объединяет один или несколько объектных модулей в одну исполняемую программу. Компоновщик размещает внешние и общие ссылки, назначает абсолютные адреса перемещаемым сегментам программ. Он может обрабатывать объектные модули, созданные транслятором C51, ассемблером A51, транслятором PL/M-51 Intel и ассемблером ASM51 Intel.

Компоновщик автоматически выбирает соответствующие библиотеки поддержки и связывает только требуемые модули из библиотек. Установки по умолчанию для L51 выбраны так, чтобы они подходили для большинства прикладных программ, но можно определить и заказные установки.

Отладчик/симулятор WinSim51

Отладчик исходных текстов используется с транслятором C51, ассемблером A51, транслятором PL/M-51 Intel и ассемблером ASM51 Intel. Отладчик/симулятор позволяет моделировать большинство особенностей Intel 8051 без наличия аппаратных средств. Можно исполь-

зовать его для проверки и отладки прикладной программы прежде, чем будут изготовлены аппаратные средства. При этом моделируется широкое разнообразие периферийных устройств, в том числе последовательный порт, внешний ввод-вывод и таймеры.

8.3.2. Интегрированные среды разработки

Интегрированная среда ProView

ProView фирмы Franklin Software Inc. – интегрированная среда разработки программного обеспечения для однокристальных микроконтроллеров семейства Intel 8051 и его клонов. Она включает в себя стандартный интерфейс Windows, полнофункциональный редактор исходных текстов с выделением синтаксических элементов цветом, организатор проекта, транслятор с языка C51, макроассемблер A51, редактор связей L51, симулятор-отладчик, операционную систему реального времени и встроенную справочную систему. Имеется бесплатная демонстрационная версия с ограничением по размеру кода программы на уровне 2 Кбайт. При вводе серийного регистрационного номера демонстрационная версия превращается в полнофункциональную. Достоинства: удобная оболочка, хорошая справочная система, хороший симулятор, есть версии для win16, win32 и DOS. Недостатки: много мелких ошибок, которые, впрочем, легко выявляются с помощью симулятора.

Интегрированная среда IAR Embedded Workbench

Интегрированная среда 8051 Embedded Workbench фирмы IAR Systems (Швеция) в рамках единой программной среды поддерживает разработку и отладку программ для большого количества микропроцессоров и микроконтроллеров. Содержит макросредства для описания периферийных узлов. В целом набор программных инструментальных средств тот же самый, что у среды фирмы Franclin, за исключением операционной системы реального времени. Программный симулятор эффективно моделирует работу микропроцессорного ядра отлаживаемого микроконтроллера, однако моделирование периферийных узлов сопровождается ошибками. Доступна бесплатная полнофункциональная демонстрационная версия с ограничениями по ресурсам.

Интегрированная среда μVision3

μVision3 – новая отладочная среда фирмы Keil Software для микроконтроллеров семейства MCS51. Она включает в себя стандартный

интерфейс Windows, средства управления проектами, мощный текстовый редактор и многофункциональный отладчик в удобной программной оболочке, транслятор с языка C51, макроассемблер A51, редактор связей L51, симулятор-отладчик, операционную систему реального времени и встроенную справочную систему. В комплект входит подробное руководство, в котором есть справочная информация по всем вопросам и раздел для быстрого освоения программы. Поддерживаются микроконтроллеры фирм: Analog Devices, AMD, Atmel, Dallas Semiconductor, Infineon, Intel, OKI, Philips, Temic, Winbond. По сравнению с другими интегрированными средами μVision3 характеризуется хорошей оптимизацией, не сопровождаемой различными ошибками компилятора, которые часто присущи другим пакетам. Franklin в этом смысле сильно проигрывает.

Интегрированная среда Project-51

Интегрированная среда Project-51 фирмы Phyton Inc. Microsystems & Development Tools включает в себя симулятор-отладчик PDS-51, внутрисхемный эмулятор PICE-51. Дополнительное программное обеспечение, входящее в базовую комплектацию, состоит из оптимизирующего компилятора с языка C51, MCA-51, редактора связей MCLINK, разработанных фирмой Phyton Inc. MicroCosm Ltd., дизассемблера и ряда других утилит. Кроме того, предусмотрена поддержка работы с макроассемблерами Keil Software A51 версий 5.x и IAR Systems версий 4.x и 5.20, не входящими в базовую комплектацию. Имеется полнофункциональная бесплатная демонстрационная версия программного пакета (Light-версия), которая имеет единственное отличие от коммерческой версии – размер программы в Light-версии не может превышать 4K.

Интегрированная среда для ADuC812

Фирма Analog Devices подготовила комплект программного обеспечения для микроконтроллера ADuC812. В комплект программного обеспечения входят ассемблер фирмы MetaLink, демо-версия компилятора C51 фирмы Keil, программное обеспечение для программирования на плате, программный симулятор ADSIM812 для ADuC812, работающий под управлением Windows, полное описание программы для ADSim812 на русском языке, отладчик и библиотека программ с примерами применения.

Эти программы могут работать с фирменной платой развития или отдельно как среда разработки, позволяя проводить полный цикл разработки и отладки программ для микроконтроллера. Для ADuC812 подходят также любые инструментальные средства, разработанные

под архитектуру MCS51. Для отладочной платы приведены принципиальная схема и разводка печатной платы.

Интегрированная среда ТФ-ИНФО-51

Программно-аппаратный комплекс Инфо-51 разработан и поставляется фирмой ИТФ «Технофорт» (г. С.-Петербург), предназначен для комплексной разработки и отладки программных и аппаратных средств, реализованных на однокристальных ЭВМ семейства MCS51 и их аналогах. Система работает на ПЭВМ, совместимых с IBM PC под управлением MS-DOS версии 3.30 и выше. Поддерживается работа с видеоконтроллерами CGA, EGA, MDA (Hercules), VGA. Минимальный объем требуемой оперативной памяти – 400 Кбайт, дисковой памяти – 300 Кбайт. Для работы аппаратной части комплекса ПЭВМ должна иметь последовательный канал ввода-вывода RS-232. При работе в среде Windows система Инфо-51 может вызываться в окне DOS.

Система Инфо-51 представляет собой интегрированную среду, состоящую из текстового редактора, компилятора с языка ассемблера микроЭВМ семейства MCS51, подсистемы управления файлами, программной модели (симулятора) и программной поддержки внутрисхемного эмулятора.

В состав внутрисхемного эмулятора ВСЭ-51 входит однокристальная микроЭВМ из семейства MCS51, кварцевый генератор и микросхемы оперативной памяти общей емкостью 128 Кбайт.

8.4. Язык программирования ASM-51

8.4.1. От исходного текста к машинным кодам

Язык программирования ASM-51 поддерживает модульное написание программ. Графическое изображение процесса написания программы на языке программирования ASM-51 приведено на рис 8.1. Файл, в котором хранится программа, написанная на языке ASM51 (исходный текст программы), называется *исходным модулем*. Для исходного текста программы принято использовать расширения файла: `asm`, `a51`, `srs` или `s51`. Исходный текст программы можно написать, используя любой текстовый редактор.

Получить объектный модуль можно, указав имя исходного модуля программы в качестве параметра программы-транслятора в командной строке или строке командного файла:

```
asm51.exe modul.asm
```

Программа *редактор связей* позволяет объединять несколько объектных файлов (модулей) в один. Для объединения нескольких модулей в исполняемую программу имена всех модулей передаются в

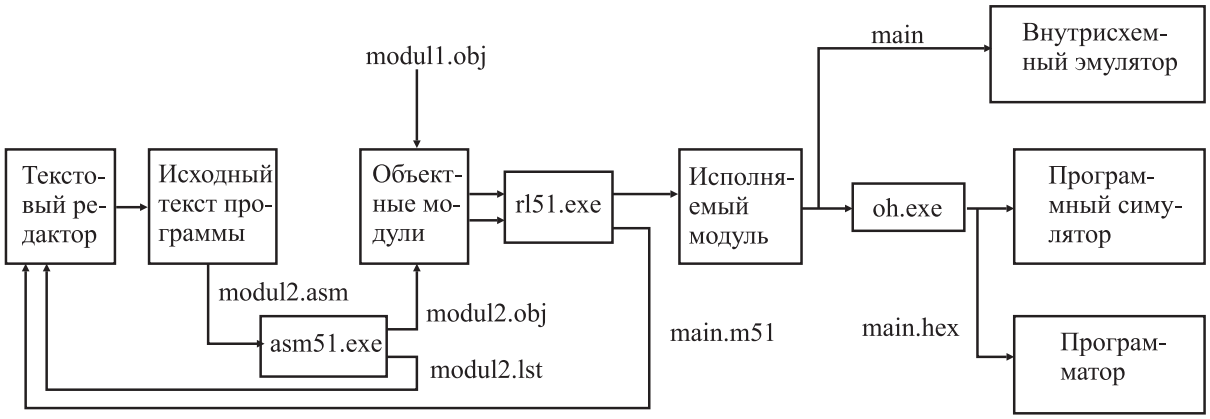


Рис. 8.1. Процесс написания программы на языке программирования ASM-51

редактор связей в качестве параметров при запуске этой программы. Пример вызова редактора связей из командной строки DOS для объединения трёх модулей:

```
rl51.exe main.obj, modul1.obj, modul2.obj
```

Имя исполняемого модуля программы по умолчанию совпадает с именем первого объектного файла в списке параметров строки запуска редактора связей. Исполняемый модуль программы записывается в файл без расширения. Объектный двоичный формат файла представляет собой двоичное представление каждого байта данных без каких-либо адреса, контрольной суммы или описания формата. Этот формат используют, например, для передачи данных внутрисхемному эмулятору.

При компиляции исходного текста программы транслятор составляет таблицу ссылок на константы, переменные и команды. Если при втором просмотре исходного текста программы, во время которого формируется объектный модуль, транслятор не обнаружит имени переменной или метки в своей таблице, то будет сформировано сообщение об ошибке и объектный модуль будет удален с диска компьютера.

Для того чтобы транслятор вместо формирования сообщения об ошибке записал в объектный модуль информацию, необходимую для редактора связей, нужно использовать специальные директивы ссылок на внешние переменные или метки. Обычно эти директивы называются PUBLIC (общие) и EXTRN (внешние). Для ссылки на переменную или метку используется директива EXTRN, в которой перечисляются через запятую метки и переменные. Редактор связей должен сначала получить их точное значение из другого модуля, а затем модифицировать все команды, в которых эти метки или переменные используются. Для того чтобы редактор связей мог осуществить связывание модулей в единую программу, переменные и метки, объявленные, по крайней мере, в одном из модулей как EXTRN, в другом

модуле должны быть объявлены как доступные для всех модулей при помощи директивы PUBLIC.

Большинство программаторов не может работать с объектным форматом исполняемого модуля программы, поэтому для загрузки машинного кода в процессор необходимо преобразовать объектный формат исполняемого модуля в общепринятый для программаторов гексадецимальный (`hex`) формат. При преобразовании форматов вся отладочная информация теряется. Машинный код процессора в гексадецимальном формате называется загрузочным модулем.

Загрузочный модуль программы получают при помощи программы-преобразователя `oh.exe`, передав ей в качестве параметра имя файла исполняемого модуля программы:

```
oh.exe main
```

Полученный файл main.hex загружается в разрабатываемое устройство или программный симулятор для последующей отладки.

8.4.2. Запись исходного текста программы

Исходный текст программы представляет собой последовательность операторов языка, сгруппированных в сегменты и оформленных в виде файла.

Оператор – это базовая конструкция языка программирования, определяющая действия в программе. В одной строке может быть записан только один оператор. Максимальный размер строки – 255 символов. Признаком конца оператора является символ *возврат каретки*.

Оператор состоит из четырех полей:

```
<метка>     <операция> <операнд>    ; <комментарий>
```

Любое из полей, в том числе и все поля, может отсутствовать. Оператор, в котором все поля отсутствуют, называется пустым оператором. Он используется для увеличения наглядности программы.

Метка. В поле метки размещается символическое имя ячейки памяти, в которой хранится отмеченная команда или операнд. Метка представляет собой буквенно - цифровую комбинацию, начинающуюся с буквы. Используются только буквы латинского алфавита. Ассемблер ASM-51 допускает использование в метках символа подчеркивания (_). Длина метки не должна превышать 31 символ. Метка всегда завершается двоеточием (:).

Пример оператора, записанного на языке ASM-51:

```
SumDig:     ADD  A,#56    ; Сложить (A) + 56
            mov  R0,A     ;
            mov  A,@R0    ;
```

Если в операторе присутствует только метка, то она помечает ближайший следующий оператор, в котором присутствует инструкция процессора или директива ассемблера. Признаком конца поля метки является символ *двоеточие* (`:`). Однако язык программирования ASM-51, в виде исключения, допускает использовать символы интервала как признак конца поля метки.

Пример использования оператора, содержащего только метку:

```
Podprog1:                 ; Помечен след. оператор
            mov  R0,A     ;
            mov  A,@R0    ;
```

Операция. В поле операции записывается мнемоническое обозначение команды MCS51 или директивы ассемблера. Используется строго определенный и ограниченный набор мнемонических кодов. Любой другой набор символов, размещенный в поле операции, воспринимается ассемблером как ошибочный.

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

Операнд может быть задан непосредственно или в виде его адреса (прямого или косвенного). Непосредственный операнд представляется числом (`MOV A,#15`) или символическим именем (`ADDC A,#OPER2`) с обязательным указанием префикса непосредственного операнда (`#`). Прямой адрес операнда может быть задан мнемоническим обозначением (`IN A, P1`), числом (`INC 40`), символическим именем

```
MOV A, MEMORY
```

Указанием на косвенную адресацию служит префикс (`@`). В командах передачи управления операндом может являться число (`LCALL 0135H`), метка (`JMP LABEL`), косвенный адрес (`JMPP @A`) или выражение (`JMP $-2`, где `$` – текущее содержимое счетчика команд).

Используемые в качестве операндов символические имена и метки должны быть определены, а числа представлены с указанием системы счисления.

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

Алфавит языка

Символы исходной программы представляют собой подмножество таблиц символов ASCII для DOS и ANSI для WINDOWS. В исходном тексте программы, написанном на языке программирования ASM-51, допустимо использование символов интервала, букв, знаков цифр.

Символы интервала определяют один или несколько пробелов в предложении исходного модуля. К этим символам относятся *пробел* и *табуляция*.

В качестве букв воспринимаются латинские буквы верхнего и нижнего регистра, цифры (0 1 2 3 4 5 6 7 8 9) и знаки (`# $ ' ( ) * + , - . / : ; < > = ? @`)

Знаки, комбинации знаков (<>, >=, <=), а также символы интервала являются разделителями конструкций языка. До и после знака *минус* разделителя в любой конструкции языка могут быть вставлены символы интервала.

ASCII-символы, не входящие в перечень основных символов алфавита языка, считаются дополнительными. Эти символы могут использоваться для пояснений в исходном тексте программы, а также для определения символьных констант.

Из символов формируются идентификаторы и числа.

Идентификаторы

Идентификатор – это символическое обозначение объекта программы. В качестве идентификатора может быть использована любая последовательность букв и цифр. При этом в качестве буквы могут быть использованы любая буква латинского алфавита, а также вопросительный знак (?) и знак *нижнее подчеркивание* (_). Идентификатор может начинаться только с буквы! В идентификаторах язык программирования ASM-51 различает буквы верхнего и нижнего регистров.

Количество символов в идентификаторе ограничено длиной строки (255 символов). Транслятор различает идентификаторы по первым 31 символам.

Примеры идентификаторов:

`ADD5`, `FFFFH`, `?`, `ALFA_1`.

В языке программирования ASM-51 имеется три категории идентификаторов: *ключевые слова*, *встроенные имена* и *определяемые имена*.

Ключевые слова. Ключевое слово является определяющей частью оператора языка ассемблера. Значения ключевых слов языка ассемблера ASM-51 не могут быть изменены или переопределены в программном модуле каким-либо образом. Ключевому слову не может быть назначено имя- синоним. Ключевые слова могут быть написаны буквами как верхнего, так и нижнего регистров.

В языке ASM-51 имеются следующие категории ключевых слов:

– инструкции;

– директивы;

– вспомогательные слова;

– операции.

Инструкции по форме записи совпадают с мнемоническими обозначениями команд микроконтроллеров семейства MCS51 и совместно с операндами составляют команды микроконтроллера.

Директивы совместно со вспомогательными словами определяют действия в программе, которые должны быть выполнены ассемблером в процессе преобразования исходного текста программы в объектный код. В языке программирования ASM-51 используются следующие директивы: BIT, BSEG, CODE, CSEG, DATA, DB, DBIT, DS, DSEG, DW, END, EQU, EXTRN, IDATA, ISEG, NAME, ORG, PUBLIC, RSEG, SEGMENT, SET, USING, XDATA, XSEG. Спецификация директив приведена в п. 8.4.3.

Директивы ассемблера не преобразуются в двоичные коды, а потому не могут иметь меток. Исключение составляют директивы резервирования памяти и определения данных (DS, DB, DW). У директив, осуществляющих определение символических имен, в поле метки записывается определяемое символическое имя, после которого двоеточие не ставится.

В качестве символических имен и меток не могут быть использованы мнемокоды команд, директивы и операторы ассемблера, а также мнемонические обозначения регистров и других внутренних блоков микроконтроллера.

Вспомогательные слова, используемые в ассемблере: AT, BIT, BITADDRESSABLE, CODE, DATA, IDATA, INBLOCK, INPAGE, NUMBER, PAGE, UNIT, XDATA.

Ассемблер ASM-51 допускает использование выражений в поле операндов, значения которых вычисляются в процессе трансляции. Перечень операций, использующихся языком ASM-51:

AND, EQ, GE, GT, HIGH, LE, LOW, LT, MOD, NE, NOT, OR, SHL, SHR, XOR.

Выражение представляет собой совокупность символических имен и чисел, связанных операторами ассемблера. Операторы ассемблера обеспечивают выполнение арифметических ("+" – сложение, " – вычитание, * – умножение, / – целое деление, MOD – деление по модулю) и логических (OR – ИЛИ, AND – И, XOR – исключающее ИЛИ, NOT – отрицание) операций в формате 2-байтных слов. Например, запись ADD A, #((NOT 13) + 1) эквивалентна записи ADD A,#0F3H и обеспечивает сложение содержимого аккумулятора с числом −13, представленным в дополнительном коде.

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

Встроенные имена. Встроенные имена присвоены адресам регистров специальных функций, адресам флагов специальных функций AR0-AR7, рабочим регистрам R0-R7 текущего банка регистров, а также аккумулятору A и флагу переноса C.

`A`, `R0`, `R1`, `R2`, `R3`, `R4`, `R5`, `R6`, `R7`, `DPTR`, `PC`, `C`, `AB`, `SP`

Определяемые имена. Определяемые имена объявляются пользователем.В языке программирования ASM-51 имеются следующие категории определяемых идентификаторов:

- метки;
- внутренние и внешние переменные адресного типа;
- внутренние и внешние переменные числового типа;
- имена сегментов;
- названия программных модулей.

Представление чисел

В языке программирования ASM-51 используются целые беззнаковые числа, представленные в двоичной, восьмеричной, десятичной и шестнадцатеричной формах записи. Для определения основания системы счисления используется суффикс (буква, следующая за числом): `B` (двоичное число), `Q` или `O` (восьмеричное число), [`D`] (десятичное число) и `H` (шестнадцатеричное число). Для десятичного числа суффикс может отсутствовать. Количество символов в числе ограничено размером строки, однако значение числа определяется по модулю 2^{16} (т.е. диапазон значений числа находится в пределах от 0 до 65535).

Примеры записи чисел:

011101b, 1011100B, 735Q, 456o, 256, 0fah, 0CBH

Часто число используется для представления символов. В этом случае для определения числа можно воспользоваться литеральной константой. Литеральная константа заключается в апострофы:

'a', 'W'

```
mov SBUF, #'Б'
```

Для записи фраз в памяти программ можно воспользоваться литеральными строками:

```
Nadp: DB 'Ошибка в блоке 5'
```

В этом случае каждый символ заменяется отдельным байтом и запоминается в ПЗУ памяти программ.

Сегменты памяти

Ассемблер поддерживает определенную логическую структуру памяти микроконтроллера для спецификации размещения отдельных частей программы. Основу логической структуры составляет понятие родового (Generic) сегмента.

Родовой сегмент имеет имя, класс памяти (class) и другие атрибуты. Родовые сегменты с одним и тем же именем, но расположенные в разных объектных модулях считаются частями одного и того же сегмента и называются частными (partial) сегментами. Объединение частных сегментов осуществляет редактор связей. Родовой сегмент создается директивой SEGMENT, специфицирующей имя и класс сегмента.

```
MYPROG      SEGMENT  CODE    ; Имя = MYPROG,
                             ; класс = CODE
```

Параметр class используется редактором связей для группировки сегментов с одним классом памяти. В ассемблере поддерживаются классы памяти BIT, CODE, DATA, IDATA, XDATA.

Необязательные параметры директивы SEGMENT позволяют задать параметры для настройки редактора связей, характеризующие тип перемещаемости и тип выделяемых ресурсов. Ассемблер поддерживает следующие типы перемещаемости (reloctype):

AT – абсолютный сегмент. Вспомогательной директивой AT задается абсолютный адрес начала сегмента;

BITADDRESSABLE – сегмент, размещаемый в области памяти прямоадресуемых битов. Эта область расположена в РПД от 20H до 2FH. Длина сегмента не может превышать 16 байтов. Такой тип сегмента может быть задан лишь для класса памяти DATA;

INBLOCK – сегмент, размещаемый в 2048-байтовом блоке. Может быть специфицирован только для класса памяти CODE;

INPAGE – сегмент, размещаемый на странице памяти в 256 битов;

OVERLAYABLE – оверлейный сегмент. Занимаемая область памяти может быть общей для нескольких оверлейных сегментов. Есть ряд ограничений на выбор имен для оверлейных сегментов.

Ассемблер ASM-51 поддерживает только один тип выделяемых (alloctype) ресурсов: PAGE предписывает редактору связей разместить стартовый адрес сегмента точно на границе 256-байтовой страницы памяти.

Для выбора одного из ранее созданных родовых сегментов используется директива RSEG, которая делает указанный родовой сегмент активным.

Пример назначения сегмента стека:

```
STACK       SEGMENT  IDATA    ; Создать сегмент
            RSEG     STACK    ; Активировать сегмент
            DS       10H      ; Под сегмент 16 байтов
```

В ассемблере ASM-51 предопределены пять родовых сегментов, соответствующих пяти областям адресного пространства в архитектуре MCS51: сегмент программ (CSEG), сегмент данных в РПД, доступных по прямому адресу (DSEG), сегмент данных в РПД, доступных по косвенному адресу (ISEG), сегмент внешних данных (XSEG) и сегмент битов (BSEG). Для каждого сегмента поддерживается свой собственный счетчик адреса. Счетчик становится активным, когда соответствующий сегмент активизируется. Когда сегмент активизируется в первый раз, его счетчик адреса равен нулю. Вспомогательная директива AT позволяет задать требуемое начальное значение счетчика сегмента. В начале работы компилятора активным считается по умолчанию сегмент программных адресов (CSEG). Это позволяет при необходимости работать с ассемблером без явного декларирования родового сегмента и его активизации.

Счетчиком адреса текущего сегмента можно управлять при помощи директив ORG, DS, DBIT. Если после переключения сегмента мы возвратимся к ранее используемому сегменту, то значение его счетчика адреса будет восстановлено таким, каким оно было, когда произошел переход к другому сегменту. Переменная $ содержит текущее значение счетчика адреса активного сегмента.

8.4.3. Описание директив ассемблера

Ассемблер ASM-51 имеет набор директив, которые позволяют установить значение меток, зарезервировать и инициализировать память, управлять размещением программы.

Директивы присваивания

EQU Присваивает числовое значение символическому имени. В процессе ассемблирования всюду, где встретится данное символиче-

ское имя, оно будет заменено числом или выражением. Символическое имя может быть определено директивой EQU только один раз.

```
<имя>       EQU  <выражение>    ;

PET         EQU  13             ; Пример: PET := 13
MAT         EQU  PET + 4        ; Будет MAT := 17
COUNTER     EQU  R0             ;
ASCII_D     EQU  'D'            ;
```

SET Присваивает новое числовое значение символическому имени. Действует аналогично директиве EQU, но символическое имя, изначально определенное директивой SET, может быть далее по тексту переопределено директивой SET неограниченное число раз.

```
<имя>       SET  <выражение>    ;

PET         SET  3              ; Пример: PET := 3
PET         SET  PET + 2        ; Будет PET := 5
```

BIT Присваивает указанному символическому имени адрес бита. Имя будет иметь тип сегмента BSEG и может быть использовано только в битовых операндах и операторе DB. Карта расположения адресуемых битов в РПД и регистрах специальных функций приведена на рис. 6.19. Биты с адресами от 0 до 127D расположены в области РПД от 20H до 2FH, с адресами от 128 до 255D – в регистрах специальных функций. Численные значения битовых адресов, превышающие 255D, воспринимаются как ошибка.

```
<имя>       BIT  <выражение>    ;

Flag        BIT  17H.2          ;
Err         BIT  OV             ;
```

CODE Присваивает символическому имени адрес ячейки, расположенной в памяти программ. Имя будет иметь тип сегмента CSEG. Численное значение адреса не должно превосходить 65535D.

```
<имя>       CODE  <выражение>          ;

RESET       CODE  0                    ;
EXTI0       CODE  RESET + (1024/16)    ;
```

DATA Присваивает символическому имени адрес прямоадресуемой ячейки в РПД. Имя будет иметь тип сегмента DSEG и не может быть использовано в операциях с битами и переходах. Численное значение, превышающее 255D, воспринимается как ошибка.

```
<имя>       DATA  <выражение>    ;

CONIN       DATA  SBUF           ; CONIN := 99H
Table       DATA  70H            ; Table := 70H
```

IDATA Присваивает символическому имени адрес ячейки в РПД, адресуемой косвенно. Имя будет иметь тип сегмента ISEG и не может быть использовано в операциях с битами и переходах. Численное значение, превышающее 255D, воспринимается как ошибка.

```
<имя>       IDATA  <выражение>   ;

TOKEN       IDATA  60            ;
BYTE_CNT    IDATA  TOKEN + 1     ;
ADDR        IDATA  TOKEN + 2     ;
```

XDATA Присваивает указанному символическому имени адрес внешней памяти данных. Имя будет иметь тип сегмента XSEG и может быть использовано только в командах DW и загрузки DPTR. Численное значение, превышающее 65535, воспринимается как ошибка.

```
<имя>       XDATA  <выражение>   ;

Date        XDATA  777H          ;
Time        XDATA  Date + 3      ;
```

Директивы управления сегментами

SEGMENT Создает новый родовой сегмент. Обязательные параметры директивы: уникальное имя родового сегмента (Name) и класс (class) памяти сегмента. Необязательные параметры reloctype и alloctype служат для настройки редактора связей.

```
Name SEGMENT class  [reloctype] [alloctype]    ;
```

RSEG Назначает указанный (Name) родовой сегмент текущим. Действует до следующего применения директивы RSEG. Специфицируемый родовой сегмент должен быть ранее создан директивой SEGMENT.

```
             RSEG  Name     ;
```

BSEG Выбирает сегмент битовых адресов, восстанавливая последнее значение битового счетчика адреса. Счетчик адреса этого сегмента может быть изменен директивами ORG и DBIT. Вспомогательная (необязательная) директива AT позволяет задать значение счетчика при выборе сегмента.

```
[метка:]    BSEG  [AT <выражение>]    ;

            BSEG                      ;
            BSEG  AT 32               ; BSEG := 32D
```

CSEG Выбирает сегмент программных адресов, восстанавливая последнее значение программного счетчика адреса. Этот сегмент устанавливается по умолчанию в начале программы. Счетчик адреса этого сегмента может быть изменен директивами ORG, DS, DB, DW, а также любой ассемблерной командой. Вспомогательная (необязательная) директива AT позволяет программисту задать конкретное значение счетчика при выборе сегмента.

```
[метка:]    CSEG  [AT <выражение>]    ;

            CSEG                      ;
            CSEG  AT 32               ; CSEG := 32D
```

DSEG Выбирает сегмент адресов внутрикристальных данных, восстанавливая последнее значение счетчика адреса внутрикристальных данных. Счетчик адреса этого сегмента может быть изменен директивами ORG и DS. Вспомогательная (необязательная) директива AT позволяет задать значение счетчика при выборе сегмента.

```
[метка:]    DSEG          ;

            DSEG          ;
            DSEG  AT 32   ; счетчик DSEG := 32D
```

ISEG Выбирает сегмент адресов РПД, восстанавливая последнее значение счетчика адреса данных в РПД. Счетчик адреса этого сегмента может быть изменен директивами ORG и DS. Вспомогательная (необязательная) директива AT позволяет задать значение счетчика при выборе сегмента.

```
[метка:]    ISEG          ;

            DSEG          ;
            ISEG  AT 32   ; счетчик ISEG := 32D
```

XSEG Выбирает сегмент адресов внешних данных, восстанавливая последнее значение счетчика адреса внешних данных. Счетчик адреса этого сегмента может быть изменен директивами ORG и DS. Вспомогательная (необязательная) директива AT позволяет задать значение счетчика при выборе сегмента.

```
[метка:]    XSEG          ;

            XSEG          ;
            XSEG  AT 32   ; счетчик XSEG := 32D
```

Директивы резервирования памяти

DS Резервирует память в указанном количестве байтов. Директиву DS можно использовать в сегментах DSEG, ISEG, или XSEG. Счетчик сегмента увеличивает свое значение на указанную величину. Численное значение указанной величины не должно приводить к выходу счетчика за границы сегмента.

```
[метка:]    DS  <выражение>    ;

            DS  10H            ; Резервирует
                               ; 16 битов памяти
STRING:     DS  10H            ; STRING := адрес
                               ; первого байта
```

DBIT Резервирует область в сегменте битов, он может использоваться только в битовом сегменте.

```
[метка:]    DBIT  <выражение>    ;
```

ORG Применяется в любых сегментах. Когда в программе встречается ORG, значение выражения присваивается счетчику адреса текущего сегмента.

```
            ORG  <выражение>    ;

            ORG  800H           ;
```

DB Инициализирует программную память байтовыми величинами и символьными строками. Байтовые величины должны быть разделены запятой.

```
[метка:]    DB  <список_выражений>    ;

            DB  0FFH                  ;
Lab:        DB  1,2,3,5H,7D           ;
Age:        DB  'MARY',27,'JOE',18    ;
```

DW Инициализирует программную память с помощью списка 16-битовых значений. Старший байт помещается по старшему адресу.

```
[метка:]    DW  <список_выражений>    ;

            DW  0EFFH                 ;
Lab:        DB  950,0FFFFH            ;
```

Прочие директивы

USING Назначает текущий банк регистров общего назначения. Директива упрощает выбор текущего банка регистров, но результат ее действия может быть переопределен командами переключения банков регистров. Численное значение номера банка регистров лежит в диапазоне от 0 до 3D.

```
            USING  <выражение>    ;

            USING  0              ; Выбран банк 0
            USING  1+1+1          ; Выбран банк 3
```

NAME Задает имя объектного модуля, генерируемого данной программой. Имя может иметь длину до 40 символов и записывается в объектный файл вместе с соответствующим модулем. В общем случае имя объектного модуля может не совпадать с именем объектного файла. В исходном файле может быть только одна директива Name. Если эта директива отсутствует, то в качестве имени объектного модуля используется имя исходного файла без расширения.

```
            NAME  <имя объектного модуля>    ;
```

PUBLIC Перечисляет символьные имена, определенные в данном модуле, которые редактор связей должен сделать доступными для использования в других модулях. В качестве разделителя имен используется запятая. В список не входят имена регистров и сегментов.

```
PUBLIC  <список символьных имен>       ;

PUBLIC  PUT_CRLF,PUT_STRING,PUT_EOS    ;
PUBLIC  ASCBIN, BINASC                 ;
PUBLIC  GETTOKEN, GETNUMBER            ;
```

EXTRN Перечисляет символьные имена, определенные в других модулях и декларированные там директивой PUBLIC, в целях использования этих символьных имен в данном модуле. В директиве специфицируется класс памяти (class) декларируемых символов: BIT, CODE, DATA, IDATA, XDATA или NUMBER. Параметр NUMBER означает любой класс памяти. Внутри круглых скобок символьные имена разделяются запятыми.

```
EXTRN   class (<список символьных имен>)    ;

EXTRN   CODE (PUT_CRLF), DATA (BUFFER)      ;
EXTERN  CODE (BINASC, ASCBIN)               ;
EXTRN   NUMBER (TABLE_SIZE)                 ;
```

END Завершает ассемблерную программу.

```
        END     ;
```

Пример программы на языке ассемблера ASM-51

```
;-----------------------------------------------
; ПРИМЕР программы на языке ассемблера ASM-51
;-----------------------------------------------

NAME    SAMPLE

EXTRN   CODE    (PUT_CRLF, PUTSTRING)
PUBLIC  TXTBIT

PROG    SEGMENT CODE
CONST   SEGMENT CODE
VAR1    SEGMENT DATA
BITVAR  SEGMENT BIT
STACK   SEGMENT IDATA

    RSEG  STACK
    DS    10H  ; 16 байтов под стек

    CSEG  AT   0
    USING   0  ; Банк регистров 0
; При включении программа стартует с адреса 0.
    JMP   START

    RSEG  PROG
; Установит указатель стека
START:  MOV   SP,#STACK-1

; Инициализация последовательного интерфейса
; Использовать таймер 1 для задания скорости передачи
; Частота резонатора = 11.059 MHz
    MOV   TMOD,#00100000B       ;C/T = 0, Mode = 2
    MOV   TH1,#0FDH
    SETB  TR1
    MOV   SCON,#01010010B

; Очистка TXTBIT для чтения CODE-памяти
    CLR   TXTBIT

; Это головная программа.
; В цикле выводит текст на консоль
REPEAT:
; печать сообщения
```

```
    MOV   DPTR,#TXT
    CALL  PUTSTRING
    CALL  PUT_CRLF
; повторение
    SJMP  REPEAT
;
    RSEG  CONST
TXT:    DB    'TEST PROGRAM',00H

; Исключительно для примера
    RSEG  VAR1
DUMMY:  DS    21H

; TXTBIT = 0 Чтение текста из CODE-памяти
; TXTBIT = 1 Чтение текста из XDATA-памяти
    RSEG  BITVAR
TXTBIT: DBIT  1

    END

;-----------------------------------------------
```

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Яценков В.С. Микроконтроллеры Microchip: Практическое руководство. Схемы применения, программы, описания /В.С. Яценков. М.: Горячая линия-Телеком, 2005. 280 с.

2. Схемотехника электронных систем: Микропроцессоры и микроконтроллеры /В.И. Бойко, А.Н. Гуржий, В.Я. Жуйков, А.А. Зори, В.М. Спивак, Т.А. Терещенко, Ю.С. Петергеря. СПб.: БХВ-Санкт-Петербург, 2004. 464 с.

3. Евстифеев А.В. Микроконтроллеры семейства Classic фирмы Atmel /А.В. Евстифеев. М.: Изд. дом "Додека-XXI", 2004. 288 с.

4. Новиков Ю.В. Основы микропроцессорной техники: курс лекций /Ю.В. Новиков, П.К. Скоробогатов. М.: ИНТУИТ.РУ "Интернет-университет информационных технологий", 2004. 440 с.

5. Предко М. Справочник по PIC-микроконтроллерам /М. Предко. М.: Изд. дом "Додека-XXI", 2004. 512 с.

6. Огородников И.Н. Введение в микропроцессорную технику /И.Н. Огородников. Екатеринбург: УГТУ-УПИ, 2004. 143 с.

7. Голубцов М.С. Микроконтроллеры AVR: от простого к сложному /М.С. Голубцов. М.: Солон Пресс, 2003. 288 с.

8. Каспер Э. Программирование на языке ассемблера для микроконтроллеров семейства i8051 /Э. Каспер. М.: Горячая линия-Телеком, 2003. 192 с.

9. Тавернье К. PIC-контроллеры /К. Тавернье. Практика и применение. М.: ДМК Пресс, 2003. 272 с.

10. Андрэ Ф. Микроконтроллеры семейства SX фирмы Scenix /Ф. Андрэ. М.: Изд. дом "Додека-XXI", 2002. 272 с.

11. Бродин В.Б. Системы на микроконтроллерах и БИС программируемой логики. /В.Б. Бродин, А.В. Калинин. М.: ЭКОМ, 2002. 400 с. (Серия "Современная микропроцессорная техника").

12. Евстифеев А.В. Микроконтроллеры Microchip: практическое руководство /А.В. Евстифеев. М.: Горячая линия-Телеком, 2002. 296 с.

13. Предко М. Руководство по микроконтроллерам: в 2 т. /М. Предко. М.: Постмаркет, 2001. Т.1. 415 с.; Т.2. 487 с. (Серия "Библиотека современной электроники").

14. Пухальский Г.И. Проектирование микропроцессорных систем: учеб. пособие для межвузовского использования /Г.И. Пухальский. СПб.: Политехника, 2001. 544 с.

15. Корнеев В.В. Современные микропроцессоры /В.В. Корнеев, А.В. Киселев. М.: НОЛИДЖ, 2000. 320 с.

16. Угрюмов Е.П. Цифровая схемотехника /Е.П. Угрюмов. СПб.: БХВ-Санкт Петербург, 2000. 528 с.

17. Ровдо А.А. Микропроцессоры от 8086 до Pentium III Xeon и AMD-K6-3. Регистры. Команды. Ассемблер /А.А. Ровдо. М.: ДМК, 2000. 592 с.

18. Бродин В.Б. Микроконтроллеры. Архитектура, программирование, интерфейс: справочник /В.Б. Бродин, М.И. Шагурин. М.: ЭКОМ, 1999. 400 с.

19. Мокрецов В.П. Микропроцессоры и микропроцессорные системы: уч. пособие /В.П. Мокрецов. Екатеринбург: УГТУ-УПИ, 1999. 125 с.

20. Комаров А.В. Введение в микропроцессоры: конспект лекций по курсу "Микропроцессорные устройства" /А.В. Комаров. Обнинск: ИАТЭ, 1998. 74 с.

21. Горбунов Б.Б. Современные микроконтроллеры: Архитектура, средства проектирования, примеры применения, ресурсы сети Интернет /под ред. И.В. Коршуна. М.: Аким, 1998. 272 с.

22. Нерода В.Я. Однокристальные микроЭВМ MCS51. Архитектура /В.Я. Нерода, В.Э. Торбинский, Е.Л. Шлыков. М.: Диджитал Компонентс, 1995. 164 с. (Серия "Однокристальные микроЭВМ").

23. Однокристальные микроЭВМ /А.В. Боборыкин, Г.Н. Липовецкий, Г.В. Литвинский, О.Н. Оксинь, С.В. Прохорчик, Л.В. Проценко, Н.В. Петренко, А.А. Сергеев, П.В. Сивобородов. М.: МИКАП, 1994. 400 с.

24. Однокристальные микроЭВМ: Семейство МК48. Семейство МК51: техническое описание и руководство по применению /Г.Н. Липовецкий, Г.В. Литвинский, О.Н. Оксинь, Л.В. Проценко, Н.В. Петренко, П.В. Сивобородов. М.: Бином, 1992. 334 с.

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

26. Григорьев В.Л. Программирование однокристальных микропроцессоров /В.Л. Григорьев. М.: Энергоатомиздат, 1987. 288 с.

27. Балашов Е.П. Микро- и миниЭВМ: учебн. пособ. для студ. вузов /Е.П. Балашов, В.Л. Григорьев, Г.А. Петров. Л.: Энергоатомиздат, 1984. 376 с.

28. Инструментальные средства для микроконтроллеров [Электронный ресурс]. Режим доступа: http://www.phyton.ru/cp1251/infos/articles.shtml.

Предметно-именной указатель

Б

В

Г

Д

Ж

З

И

К

Л

М

H

O

П

Р

С

Т

У

Ф

Ц

Ш

Э

Я

Приложение 1. СИСТЕМА КОМАНД МП i8080

Команда		Описание	Б	Z	S	P	C	Ac
Команды пересылки данных								
MOV	D,S	(D)←(S)	1	–	–	–	–	–
MVI	D,data	(D)←data	2	–	–	–	–	–
LXI	rp,data	(rp)←data	3	–	–	–	–	–
LDA	addr	(A)←(addr)	3	–	–	–	–	–
STA	addr	(addr)←(A)	3	–	–	–	–	–
LHLD	addr	(L)←(addr); (H)←(addr+1)	3	–	–	–	–	–
SHLD	addr	(addr)←(L); (addr+1)←(H)	3	–	–	–	–	–
LDAX	rp	(A)←((rp))	1	–	–	–	–	–
STAX	rp	((rp))←(A)	1	–	–	–	–	–
XCHG		(H)↔(D); (L)↔(E)	1	–	–	–	–	–
Арифметические команды								
ADD	S	(A)←(A)+(S)	1	+	+	+	+	+
ADI	data	(A)←(A)+data	2	+	+	+	+	+
ADC	S	(A)←(A)+(S)+(C)	1	+	+	+	+	+
ACI	data	(A)←(A)+data+(C)	2	+	+	+	+	+
DAD	rp	(HL)←(HL)+(rp)	1	–	–	–	+	–
SUB	S	(A)←(A)−(S)	2	+	+	+	+	+
SUI	data	(A)←(A)−data	2	+	+	+	+	+
SBB	S	(A)←(A)−(S)−(C)	1	+	+	+	+	+
SBI	data	(A)←(A)−data−(C)	2	+	+	+	+	+
INR	D	(D)←(D)+1	1	+	+	+	+	+
INX	rp	(rp)←(rp)+1	1	–	–	–	–	–
DCR	D	(D)←(D)−1	1	+	+	+	–	+
DCR	rp	(rp)←(rp)−1	1	–	–	–	–	–
DAA		Десятичная коррекция	1	+	+	+	+	+
Логические команды								
ANA	S	(A)←(A)∧(S)	1	+	+	+	0	+
ANI	data	(A)←(A)∧data	2	+	+	+	0	+
XRA	S	(A)←(A)⊕(S)	1	+	+	+	0	0
XRI	data	(A)←(A)⊕data	2	+	+	+	0	0
ORA	S	(A)←(A)∨(S)	1	+	+	+	0	0
ORI	data	(A)←(A)∨data	2	+	+	+	0	0
CMP	S	(A)−(S)	1	+	+	+	+	+
CPI	data	(A)−data	2	+	+	+	+	+
RAR		рис. 1.31, *c*	1	–	–	–	+	–
RAL		рис. 1.31, *d*	1	–	–	–	+	–

RRC		рис. 1.31, *e*	1	–	–	–	+	–
RLC		рис. 1.31, *f*	1	–	–	–	+	–
STC		(C)←1	1	–	–	–	+	–
CMC		(C)←($\overline{\text{C}}$)	1	–	–	–	+	–
CMA		(A)←($\overline{\text{A}}$)	1	–	–	–	–	–
Команды передачи управления								
JMP	addr	(PC)←addr	3	–	–	–	–	–
Jcond	addr	JMP addr, если условие (cond) в поле ссс истинно	3	–	–	–	–	–
CALL	addr	((SP-1))←(PCH); ((SP-2))←(PCL); (PC)←addr	3	–	–	–	–	–
Ccond	addr	CALL addr, если условие (cond) в поле ссс истинно	3	–	–	–	–	–
RET		(PCL)←[SP-2]; (PCH)←[SP-1]	1	–	–	–	–	–
Rcond		RET, если условие (cond) в поле ссс истинно	1	–	–	–	–	–
PCHL		(PC)←(HL)	1	–	–	–	–	–
RST	n	(PC)← 8 × *n*	1	–	–	–	–	–
Команды прочие								
IN	port	(A)←(port)	2	–	–	–	–	–
OUT	port	(port)←(A)	2	–	–	–	–	–
PUSH	rp	[SP-1]←(rpH); [SP-2]←(rpL)	1	–	–	–	–	–
PUSH	PSW	[SP-1]←(A); [SP-2]←(F)	1	–	–	–	–	–
POP	rp	(rpL)←[SP-2]; (rpH)←[SP-1]	1	–	–	–	–	–
POP	PSW	(F)←[SP-2]; (A)←[SP-1]	1	+	+	+	+	+
XTHL		(HL)↔[SP]	1	–	–	–	–	–
SPHL		(SP)←(HL)	1	–	–	–	–	–
EI		INTE←1	1	–	–	–	–	–
DI		INTE←0	1	–	–	–	–	–
NOP		Нет операции	1	–	–	–	–	–
HLT		Останов	1	–	–	–	–	–

Примечание. Условные обозначения: Б – формат команды в байтах; Z, S, P, C, Ac – биты регистра признаков; D, S (Destination, Source) – регистры приемника и источника данных (A, B, C, D, E, H, L, M); addr – два байта адреса; data – один или два байта данных; rp – регистровая пара (BC, DE, HL или SP); rpL и rpH – младший и старший байты регистровой пары; PCL и PCH – младший и старший байты PC; port – один байт адреса порта; (…) – содержимое ячейки памяти или регистра; ((SP)) – содержимое ячейки стековой памяти; *n* – номер команды рестарта (*n* = 1…7); ссс – трехбитный код признака.

Приложение 2. СИСТЕМА КОМАНД МП i8086

Команда		Описание	Б
1		2	3
Команды пересылки данных			
MOV	dest,src	(dest = sr/r/m)←(src = imd/sr/r/m)	2-6
XCHG	dest,src	(dest = r/m)↔(src = r)	2-4
XLAT		(AL)←ES:[BX + (AL)]	1
LEA	reg16,m	reg16←EA	2-4
LDS	reg16,m	reg16←[m]; DS←[m+2]	2-4
LES	reg16,m	reg16←[m]; ES←[m+2]	2-4
LAHF		AH←FL	1
SAHF		FL←AH	1
PUSH	src	[SS:SP]←(src = sr/reg16/mem16)	1-4
PUSHF		[SS:SP]←F	1
POP	dst	(dst = sr/reg16/mem16)←[SS:SP]	1-4
POPF		F←[SS:SP]	1
IN	AL,P8	AL←Port(P8)	2
IN	AL,DX	AL←Port([DX])	1
IN	AX,P8	AX←Port(P8)	2
IN	AX,DX	AX←Port([DX])	1
OUT	P8,AL	Port(P8)←AL	2
OUT	DX,AL	Port([DX])←AL	1
OUT	P8,AX	Port(P8)←AX	2
OUT	DX,AX	Port([DX])←AX	1
Арифметические команды			
ADD	dest,src	(dest = r/m)←dest + (src = imd/r/m)	2-5
ADC	dest,src	(dest = r/m)←dest + (src = imd/r/m) + CF	2-5
INC	r/m	(r/m)←(r/m) + 1	2-4
AAA		Коррекция сложения распак. ДДК	1
DAA		Коррекция сложения упаков. ДДК	1
SUB	dest,src	(dest = r/m)←dest - (src = imd/r/m)	2-6
SBB	dest,src	(dest = r/m)←dest - (src = imd/r/m) - CF	2-6
DEC	r/m	(r/m)←(r/m) - 1	2-4
NEG	r/m	(r/m)← (−r/m)	2-4
CMP	dest,src	F⇐(dest = r/m) - (src = imd/r/m)	2-6
AAS		Коррекция вычитания распак. ДДК	1
DAS		Коррекция вычитания упаков. ДДК	1
MUL	src	AX←AL×(src = reg8, mem8)	2-4
MUL	src	DX_AX←AX×(src = reg16, mem16)	2-4
IMUL	src	AX←AL×(src = reg8, mem8)	2-4

1		2	3
IMUL	src	DX_AX←AX×(src = reg16, mem16)	2-4
AAM		Коррекция умножения распак. ДДК	2
DIV	src	AL (+AH)←AX:(src = reg8, mem8)	2-4
DIV	src	AX (+DX)←DX_AX:(src = reg16, mem16)	2-4
IDIV	src	AL (+AH)←AX:(src = reg8, mem8)	2-4
IDIV	src	AX (+DX)←DX_AX:(src = reg16, mem16)	2-4
AAD		Коррекция деления распак. ДДК	2
CWB		AH←(AL.7)	1
CWD		DX←(AX.15)	1
Логические команды			
NOT	r/m	(r/m)←$\overline{(r/m)}$	2-4
AND	dest,src	(dest = r/m)←dest ∧ (src = imd/r/m)	2-6
OR	dest,src	(dest = r/m)←dest ∨ (src = imd/r/m)	2-6
XOR	dest,src	(dest = r/m)←dest ⊕ (src = imd/r/m)	2-6
TEST	dest,src	F⇐(dest = r/m) ∧ (src = imd/r/m)	2-6
Команды сдвигов (src = reg8/mem8)			
RCL	src,1	рис. 1.31, *f*, сдвиг влево на 1 поз.	2
RCR	src,1	рис. 1.31, *e*, сдвиг вправо на 1 поз.	2
RCL	src,CL	рис. 1.31, *f*, сдвиг влево на CL поз.	2
RCR	src,CL	рис. 1.31, *e*, сдвиг вправо на CL поз.	2
ROL	src,1	рис. 1.31, *d*, сдвиг влево на 1 поз.	2-4
ROR	src,1	рис. 1.31, *c*, сдвиг вправо на 1 поз.	2-4
ROL	src,CL	рис. 1.31, *d*, сдвиг влево на CL поз.	2-4
ROR	src,CL	рис. 1.31, *c*, сдвиг вправо на CL поз.	2-4
SAL	src,1	Арифм. сдвиг влево на 1 поз.	2-4
SAR	src,1	Арифм. сдвиг вправо на 1 поз.	2-4
SAL	src,CL	Арифм. сдвиг влево на CL поз.	2-4
SAR	src,CL	Арифм. сдвиг вправо на CL поз.	2-4
SHL	src,1	Логич. сдвиг влево на 1 поз.	2-4
SHR	src,1	Логич. сдвиг вправо на 1 поз.	2-4
SHL	src,CL	Логич. сдвиг влево на CL поз.	2-4
SHR	src,CL	Логич. сдвиг вправо на CL поз.	2-4
Строковые команды			
MOVSB		ES:[DI]←DS:[SI] (байт)	1
MOVSW		ES:[DI]←DS:[SI] (слово)	1
LODSB		AL←DS:[SI] (байт)	1
LODSW		AX←DS:[SI] (слово)	1
STOSB		DS:[SI]←AL (байт)	1
STOSW		DS:[SI]←AX (слово)	1
CMPSB		F⇐ES:[DI] − DS:[SI] (байт)	1

1	2	3
CMPSB	F⇐ES:[DI] – DS:[SI] (слово)	1
SCADB	F⇐ DS:[SI] – AL (байт)	1
SCADW	F⇐ DS:[SI] – AX (слово)	1
	Префиксы строковых команд	
REP	Повторения команд при CX≠0	1
REPE	Повторение при ZF = 1 & CX≠0	1
REPZ	Повторение при ZF = 1 & CX≠0	1
REPNE	Повторение при ZF = 0 & CX≠0	1
REPNZ	Повторение при ZF = 0 & CX≠0	1
	Команды передачи управления	
JMP adr	IP ← IP + (adr = disp16)	3
JMP NEAR adr	IP ← IP + (adr = disp8)	2
JMP r/m	IP ← IP + r/m	2-4
JMP FAR adr	IP ← offset adr; CS ← seg adr	5
JMP FAR m	IP ← [m]; CS ← [m + 2]	2-4
J(cond) adr	Переход, если (cond) истинно	2
LOOP adr	Цикл: CX←CX - 1 и переход, если CX≠0	2
LOOPE adr	Цикл: CX←CX - 1 и переход, если (CX≠0 и ZF = 1)	2
LOOPZ adr	Цикл: CX←CX - 1 и переход, если (CX≠0 и ZF = 1)	2
LOOPNE adr	Цикл: CX←CX - 1 и переход, если (CX≠0 и ZF = 0)	2
LOOPNZ adr	Цикл: CX←CX - 1 и переход, если (CX≠0 и ZF = 0)	2
CALL NEAR adr	SP ← SP - 2; [SS:SP] ← IP IP ← (adr = r/m/disp16)	3
CALL FAR adr32	IP ← offset(adr32); CS ← seg(adr32)	5
CALL FAR m	SP ← SP - 2; [SS:SP] ← CS SP ← SP - 2; [SS:SP] ← IP IP ← [m]; CS ← [m + 2]	2-4
RET	IP ← [SS:SP]; SP ← SP + 2	1
RET NEAR	RET	1
RET FAR	IP ← [SS:SP]; SP ← SP + 2 CS ← [SS:SP]; SP ← SP + 2	1
INT n	SP ← SP - 2; [SS:SP] ← F; SP ← SP - 2; [SS:SP] ← CS; SP ← SP - 2; [SS:SP] ← IP; IP ← [0000:(4n)]; CS←[0000:(4n + 2)]	2
INT 3	Программное прерывание n = 3	2
INTO	Программное прерывание n = 4	2
IRET	Возврат из прерывания:	1

1	2	3
	IP ← [SS:SP]; SP ← SP + 2 CS ← [SS:SP]; SP ← SP + 2 F ← [SS:SP]; SP ← SP + 2	1
Команды управления состоянием МП		
CLC	CF ← 0	1
CMC	CF ← $\overline{CF}$	1
STC	CF ← 1	1
CLD	DF ← 0	1
STD	DF ← 1	1
CLI	IF ← 0	1
STI	IF ← 1	1
HLT	Останов МП	1
WAIT	Ожидание сигнала на линии TEST	1
ESC msk/m	Передача кода команды msk или кода и операнда m сопроцессору	2 2-4
LOCK	Префикс блокировки шины на время выполнения следующей команды	1
NOP	Нет операции	1

Примечание. Условные обозначения: Б – формат команды в байтах; sr – сегментный регистр; r – 8/16-разрядный регистр общего назначения; reg16 – 16-разрядный регистр; reg8 – 8-разрядный регистр; m – 8/16-разрядный операнд в памяти; mem16 – 16-разрядное слово в памяти; mem8 – байт в памяти; dest – операнд-назначение; src – операнд-источник; imd – непосредственный операнд; Port(P8) – порт с 8-разрядным адресом P8; Port([DX]) – порт с 16-разрядным адресом, заданным в регистре DX; adr – 8/16/32-разрядный адрес; adr32 – 32 разрядный адрес, n – номер вектора прерывания, mks – код следующей команды; ДДК – двоично-десятичный код; disp8/16 – 8/16-разрядное смещение.

Приложение 3. СИСТЕМА КОМАНД MCS-51

Название команды	Мнемокод	Т	Б	Ц	Операция
1	2	3	4	5	6
Группа команд передачи данных					
Пересылка в А из регистра Rn (n=0÷7)	MOV A, Rn	1	1	1	(A)←(Rn)
Пересылка в А прямоадресуемого байта	MOV A, ad	3	2	1	(A)←(ad)
Пересылка в А байта из РПД (i=0,1)	MOV A, @Ri	1	1	1	(A)←[Ri]
Загрузка в А константы #d	MOV A, #d	2	2	1	(A)←#d
Пересылка в регистр Rn из А	MOV Rn, A	1	1	1	(Rn)←(A)
Пересылка в регистр Rn прямоадресуемого байта	MOV Rn, ad	3	2	2	(Rn)←(ad)
Загрузка в регистр Rn константы #d	MOV Rn, #d	2	2	1	(Rn)←#d
Пересылка А по прямому адресу	MOV ad, A	3	2	1	(ad)←(A)
Пересылка Rn по прямому адресу	MOV ad, Rn	3	2	2	(ad)←(Rn)
Пересылка прямоадресуемого байта по прямому адресу	MOV add, ads	9	3	2	(add)←(ads)
Пересылка байта РПД по прямому адресу	MOV ad, @Ri	3	2	2	(ad)←[Ri]
Пересылка константы по прямому адресу	MOV ad, #d	7	3	2	(ad)←#d
Пересылка в РПД из аккумулятора	MOV @Ri, A	1	1	1	[Ri]←(A)
Пересылка в РПД прямоадресуемого байта	MOV @Ri, ad	3	2	2	[Ri]←(ad)
Пересылка в РПД константы	MOV @Ri, #d	2	2	1	[Ri]←#d
Загрузка DPTR	MOV DPTR, #d16	13	3	2	(DPTR)←#d16
Пересылка в аккумулятор байта из ПП	MOVC A, @A+ +DPTR	1	1	2	(A)←[A + DPTR]
Пересылка в аккумулятор байта из ПП	MOVC A, @A+PC	1	1	2	(PC)←(PC) + 1, (A)←[A + PC]
Пересылка в аккумулятор байта из ВПД	MOVX A, @Ri	1	1	2	(A)←[Ri]
Пересылка в А байта из ВПД	MOVX A, @DPTR	1	1	2	(A)←[DPTR]

1	2	3	4	5	6
Пересылка в ВПД из аккумулятора	MOVX @Ri, A	1	1	2	[Ri]←(A)
Пересылка в расширенную ВПД из A	MOVX @DPTR, A	1	1	2	[DPTR]←(A)
Загрузка в стек	PUSH ad	3	2	2	(SP)←(SP) + 1, [SP]←(ad)
Извлечение из стека	POP ad	3	2	2	(ad)←[SP], (SP)←(SP) - 1
Обмен аккумулятора с регистром	XCH A, Rn	1	1	1	(A)↔(Rn)
Обмен аккумулятора с прямоадресуемым байтом	XCH A, ad	5	2	1	(A)↔(ad)
Обмен аккумулятора с байтом из РПД	XCH A, @Ri	1	1	1	(A)↔[Ri]
Обмен младшей тетрады аккумулятора с младшей тетрадой байта РПД	XCHD A, @Ri	1	1	1	$(A_{0-3})\leftrightarrow(Ri)_{0-3}$
Группа команд арифметических операций					
Сложение аккумулятора с регистром (n=0÷7)	ADD A, Rn	1	1	1	(A)←(A)+(Rn)
Сложение аккумулятора с прямоадресуемым байтом	ADD A, ad	3	2	1	(A)←(A)+(ad)
Сложение аккумулятора с байтом из РПД (i=0,1)	ADD A, @Ri	1	1	1	(A)←(A)+[Ri]
Сложение аккумулятора с константой	ADD A, #d	2	2	1	(A)←(A)+#d
Сложение аккумулятора с регистром и переносом	ADDC A, Rn	1	1	1	(A)←(A)+(Rn)+(C)
Сложение аккумулятора с прямоадресуемым байтом и переносом	ADDC A, ad	3	2	1	(A)←(A)+(ad)+(C)
Сложение аккумулятора с байтом из РПД и переносом	ADDC A, @Ri	1	1	1	(A)←(A)+[Ri]+(C)
Сложение аккумулятора с константой и переносом	ADDC A, #d	2	2	1	(A)←(A)+#d+(C)
Десятичная коррекция аккумулятора	DA A	1	1	1	Если $(A_{0-3})>9\vee((AC)=1)$, то $(A_{0-3})\leftarrow(A_{0-3})+6$; затем если $(A_{4-7})>9\vee((C)=1)$, то $(A_{4-7})\leftarrow(A_{4-7})+6$

1	2	3	4	5	6
Вычитание из аккумулятора регистра и заема	SUBB A, Rn	1	1	1	(A)←(A)-(Rn)-(C)
Вычитание из аккумулятора прямоадресуемого байта и заема	SUBB A, ad	3	2	1	(A)←(A)-(ad)-C
Вычитание из аккумулятора байта РПД и заема	SUBB A, @Ri	1	1	1	(A)←(A)-[Ri]-C
Вычитание из аккумулятора константы и заема	SUBB A, #d	2	2	1	(A)←(A)-#d-(C)
Инкремент аккумулятора	INC A	1	1	1	(A)←(A)+1
Инкремент регистра	INC Rn	1	1	1	(Rn)←(Rn)+1
Инкремент прямоадресуемого байта	INC ad	3	2	1	(ad)←(ad)+1
Инкремент байта в РПД	INC @Ri	1	1	1	[Ri]←[Ri]+1
Инкремент указателя данных	INC DPTR	1	1	2	(DPTR)←(DPTR)+1
Декремент аккумулятора	DEC A	1	1	1	(A)←(A)-1
Декремент регистра	DEC Rn	1	1	1	(Rn)←(Rn)-1
Декремент прямоадресуемого байта	DEC ad	3	2	1	(ad)←(ad)-1
Декремент байта в РПД	DEC @Ri	1	1	1	[Ri]←[Ri]-1
Умножение аккумулятора на регистр B	MUL AB	1	1	4	(B)(A)←(A)×(B)
Деление аккумулятора на регистр B	DIV AB	1	1	4	(A).(B)←(A)%(B)
Группа команд логических операций					
Логическое И аккумулятора и регистра	ANL A, Rn	1	1	1	(A)←(A)∧(Rn)
Логическое И аккумулятора и прямоадресуемого байта	ANL A, ad	3	2	1	(A)←(A)∧(ad)
Логическое И аккумулятора и байта из РПД	ANL A, @Ri	1	1	1	(A)←(A)∧[Ri]
Логическое И аккумулятора и константы	ANL A, #d	2	2	1	(A)←(A)∧#d
Логическое И прямоадресуемого байта и аккумулятора	ANL ad, A	3	2	1	(ad)←(ad)∧(A)
Логическое И прямоадресуемого байта и константы	ANL ad, #d	7	3	2	(ad)←(ad)∧#d
Логическое ИЛИ аккумулятора и регистра	ORL A, Rn	1	1	1	(A)←(A)∨(Rn)

1	2	3	4	5	6
Логическое ИЛИ аккумулятора и прямоадресуемого байта	ORL A, ad	3	2	1	(A)←(A)∨(ad)
Логическое ИЛИ аккумулятора и байта из РПД	ORL A, @Ri	1	1	1	(A)←(A)∨[Ri]
Логическое ИЛИ аккумулятора и константы	ORL A, #d	2	2	1	(A)←(A)∨#d
Логическое ИЛИ прямоадресуемого байта и аккумулятора	ORL ad, A	3	2	1	(ad)←(ad)∨(A)
Логическое ИЛИ прямоадресуемого байта и константы	ORL ad, #d	7	3	2	(ad)←(ad)∨#d
Исключающее ИЛИ аккумулятора и регистра	XRL A, Rn	1	1	1	(A)←(A)∀(Rn)
Исключающее ИЛИ аккумулятора и прямоадресуемого байта	XRL A, ad	3	2	1	(A)←(A)∀(ad)
Исключающее ИЛИ аккумулятора и байта из РПД	XRL A, @Ri	1	1	1	(A)←(A)∀[Ri]
Исключающее ИЛИ аккумулятора и константы	XRL A, #d	2	2	1	(A)←(A)∀#d
Исключающее ИЛИ прямоадресуемого байта и аккумулятора	XRL ad, A	3	2	1	(ad)←(ad)∀(A)
Исключающее ИЛИ прямоадресуемого байта и константы	XRL ad, #d	7	3	2	(ad)←(ad)∀#d
Сброс аккумулятора	CLR A	1	1	1	(A)←0
Инверсия аккумулятора	CPL A	1	1	1	(A)←($\overline{A}$)
Сдвиг аккумулятора влево циклический	RL A	1	1	1	(A_{n+1})←(A_n), n=0÷6, (A_0)←(A_7)
Сдвиг аккумулятора влево через перенос	RLC A	1	1	1	(A_{n+1})←(A_n), n=0÷6, (A_0)←(C), (C)←(A_7)
Сдвиг аккумулятора вправо циклический	RR A	1	1	1	(A_n)←(A_{n+1}), n=0÷6, (A_7)←(A_0)
Сдвиг аккумулятора вправо через перенос	RRC A	1	1	1	(A_n)←(A_{n+1}), n=0÷6, (C)←(A_0), (A_7)←(C)
Обмен местами тетрад в аккумуляторе	SWAP A	1	1	1	(A_{0-3} ↔A_{4-7}
Группа команд операций с битами					
Сброс переноса	CLR C	1	1	1	(C)←0
Сброс бита	CLR bit	4	2	1	(b)←0
Установка переноса	SETB C	1	1	1	(C)←1
Установка бита	SETB bit	4	2	1	(b)←1

1	2		3	4	5	6
Инверсия переноса	CPL	C	1	1	1	(C)←($\overline{C}$)
Инверсия бита	CPL	bit	4	2	1	(b)←($\overline{b}$)
Логическое И бита и переноса	ANL	C, bit	4	2	2	(C)←(C)∧(b)
Логическое И инверсии бита и переноса	ANL	C, /bit	4	2	2	(C)←(C)∧($\overline{b}$)
Логическое ИЛИ бита и переноса	ORL	C, bit	4	2	2	(C)←(C)∨(b)
Логическое ИЛИ инверсии бита и переноса	ORL	C, /bit	4	2	2	(C)←(C)∨($\overline{b}$)
Пересылка бита в перенос	MOV	C, bit	4	2	1	(C)←(b)
Пересылка переноса в бит	MOV	bit, C	4	2	2	(b)←(C)
Группа команд передачи управления						
Длинный переход в полном объеме памяти программ	LJMP	ad16	12	3	2	(PC)←ad16
Абсолютный переход внутри страницы в 2 Кбайта	AJMP	ad11	6	2	2	(PC)←(PC)+2 (PC_{0-10})←ad11
Короткий относительный переход внутри страницы в 256 байт	SJMP	rel	5	2	2	(PC)←(PC)+2 (PC)←(PC)+rel
Косвенный относительный переход	JMP	@A+DPTR	1	1	2	(PC)←(A)+(DPTR)
Переход, если аккумулятор равен нулю	JZ	rel	5	2	2	(PC)←(PC)+2; если (A)=0: (PC)←(PC)+rel
Переход, если аккумулятор не равен нулю	JNZ	rel	5	2	2	(PC)←(PC)+2; если (A)≠0: (PC)←(PC)+rel)
Переход, если перенос равен единице	JC	rel	5	2	2	(PC)←(PC)+2; если (C)=1: (PC)←(PC)+rel
Переход, если перенос равен нулю	JNC	rel	5	2	2	(PC)←(PC)+2; если (C)=0: (PC)←(PC)+rel
Переход, если бит равен единице	JB	bit, rel	11	3	2	(PC)←(PC)+3; если (b)=1: (PC)←(PC)+rel
Переход, если бит равен нулю	JNB	bit, rel	11	3	2	(PC)←(PC)+3; если (b)=0: (PC)←(PC)+rel
Переход, если бит установлен, с последующим сбросом бита	JBC	bit, rel	11	3	2	(PC)←(PC)+3; если (b)=0: (b)←0 и (PC)←(PC)+rel

1	2	3	4	5	6
Декремент регистра и переход, если не нуль	DJNZ Rn, rel	5	2	2	(PC)←(PC)+2 и (Rn)←(Rn)-1; если (Rn)≠0: (PC)←(PC)+rel
Декремент прямоадресуемого байта и переход, если не нуль	DJNZ ad, rel	8	3	2	(PC)←(PC)+2 и (ad)←(ad)-1; если (ad)≠0: (PC)←(PC)+rel
Сравнение аккумулятора с прямоадресуемым байтом и переход, если не равно	CJNE A, ad, rel	8	3	2	(PC)←(PC)+3; если (A)≠(ad): (PC)←(PC)+rel, (A)<(ad): (C)←1, иначе: (C)←0
Сравнение аккумулятора с константой и переход, если не равно	CJNE A, #d, rel	10	3	2	(PC)←(PC)+3; если (A)≠(#d): (PC)←(PC)+rel, если (A)<(#d): (C)←1, иначе: (C)←0
Сравнение регистра с константой и переход, если не равно	CJNE Rn, #d, rel	10	3	2	(PC)←(PC)+3; если (Rn)≠(#d): (PC)←(PC)+rel, если (Rn)<(#d): (C)←1, иначе: (C)←0
Сравнение байта в РПД с константой и переход, если не равно	CJNE @Ri, #d, rel	10	3	2	(PC)←(PC)+3; если [Ri]≠(#d): (PC)←(PC)+rel, если [Ri]<(#d): (C)←1, иначе: (C)←0
Длинный вызов подпрограммы	LCALL ad16	12	3	2	(PC)←(PC)+3; (SP)←(SP)+1; [SP]←(PC_{0-7}), (SP)←(SP)+1; [SP]←(PC_{8-15}), (PC)←ad16
Абсолютный вызов подпрограммы в пределах страницы в 2 Кбайта	ACALL ad11	6	2	2	(PC)←(PC)+2; (SP)←(SP)+1; [SP]←(PC_{0-7}), (SP)←(SP)+1; [SP]←(PC_{8-15}), (PC_{0-10})←ad11
Возврат из подпрограммы	RET	1	1	2	(PC_{8-15})←[SP]; (SP)←(SP)-1; (PC_{0-7})←[SP]; (SP)←(SP)-1
Возврат из подпрограммы обработки прерывания	RETI	1	1	2	(PC_{8-15})←[SP]; (SP)←(SP)-1; (PC_{0-7})←[SP]; (SP)←(SP)-1
Холостая команда	NOP	1	1	1	(PC)←(PC)+1

Примечание. Ассемблер допускает использование обобщенного имени команд JMP и CALL, которые в процессе трансляции заменяются оптимальными по формату командами перехода (AJMP, SJMP, LJMP) или вызова (ACALL, LCALL). Буквами обозначены: Т – тип команды в соответствии с рис. 6.18; Б – формат в байтах, Ц – число машинных циклов.

УКАЗАТЕЛЬ СОКРАЩЕНИЙ

АЛУ	Арифметико-логическое устройство
АЦП	Аналого-цифровой преобразователь
БИС	Большая интегральная схема
ВП	Внешняя память
ВПД	Внешняя память данных
ВПП	Внешняя память программ
КМОП	Комплементарный МОП
КОП	Код операции
МК	Микроконтроллер
МОП	Металл-окисел-полупроводник
ОЗУ	Оперативное запоминающее устройство
ПЗУ	Постоянное запоминающее устройство
ПЛИС	Программируемая логическая интегральная схема
ППЗУ	Программируемое ПЗУ
РОН	Регистр общего назначения
РПД	Резидентная память данных
РПП	Резидентная память программ
РСФ	Регистр специальных функций
РУД	Регистр-указатель данных
РУС	Регистр-указатель стека
СБР	Сброс
СК	Счетчик команд
СППЗУ	Стираемое программируемое ПЗУ
ССП	Слово состояния программы
ТТЛ	Транзисторно-транзисторная логика
УВВ	Устройство ввода-вывода
УФ	Ультрафиолет
ЦАП	Цифро-аналоговый преобразователь
ЭВМ	Электронно-вычислительная машина
ШИМ	Широтно-импульсный модулятор
BP	Base Pointer
BROWN OUT	Схема сброса при понижении напряжении питания
CAN	Controller area network
CISC	Complete Instruction Set Computer
CMOS	Complementary MOS
DAC	Digital Analog Converter
DP	Data Pointer
EEPROM	Electrically erasable PROM
EPM	External Program Memory
EPROM	Erasable PROM
FLASH	Флэш-память
IDE	Integrated Development Environment
IDLE	Режим холостого хода
I2C	Inter-Integrated Circuit Bus
LCD	Liquid Crystal Display
LVD	Low Voltage Detection
mI2C	Master Inter-Integrated Circuit Bus
PCCP	Programmable Consumer Controller Processors (Программируемые микроконтроллеры широкого применения
PGA	Programmable Gain Amplifier
PROM	Programmable ROM
PSP	Parallel Slave Port
RAM	Random access memory (ОЗУ)
RISC	Reduced Instruction Set Computer

ROM	Read only memory (ПЗУ)
SMB	System Management Bus
SPI	Serial Peripheral Interface
UART	Universal asynchronous receiver transmitter
USART	Universal synchronous/asynchronous receiver transmitter
USB	Universl serial bus
Watch	Dog Сторожевой таймер

Оглавление

Учебное издание

Огородников Игорь Николаевич

Микропроцессорная техника

Редактор *Л.Ю. Козяйчева*

Компьютерная верстка *И.Н. Огородникова*

Подписано в печать 29.11.2006 Формат $60\times84\frac{1}{16}$

Бумага писчая Офсетная печать Усл.печ.л. 22.1

Уч.-изд.л. 22.4 Тираж Заказ Цена “С”

Редакционно-издательский отдел ГОУ ВПО УГТУ-УПИ
620002, Екатеринбург, ул. Мира, 19

Ризография НИЧ ГОУ ВПО УГТУ-УПИ
620002, Екатеринбург, ул. Мира, 19