
КОМПЛЕКТ БИС K1804

в процессорах
и контроллерах

© КОМПЛЕКТ БИС K1804

КОМПЛЕКТ БИС К1804 в процессорах и контролерах

Под редакцией В. Б. Смолова



МОСКВА
«РАДИО И СВЯЗЬ»
1990



Scan AAW

ББК 32.973.2

К63

УДК 621.3.049.771.14 : 681.325.5

АВТОРЫ: В. М. МЕЩЕРЯКОВ, И. Е. ЛОБОВ, С. С. ГЛЕБОВ,
В. В. НОВОСЕЛОВ, Л. А. ШУМИЛОВ

Рецензенты: д-р техн. наук А. Д. ИВАННИКОВ,
канд. техн. наук Н. Н. ЩЕЛКУНОВ

Редакция литературы по электронике

Комплект БИС К1804 в процессорах и контроллерах /
К63 В. М. Мещеряков, И. Е. Лобов, С. С. Глебов и др.; Под
ред. В. Б. Смолова.—М.: Радио и связь, 1990.—256 с.: ил.

ISBN 5-256-00250-3.

Приведены сведения о микросхемах комплекта БИС К1804 и рассмотрены примеры проектирования высокопроизводительных устройств на их основе. Систематизированы методы и схемные решения, служащие повышению быстродействия, сокращению аппаратных затрат, улучшению диагностируемости. Описаны средства отладки микропрограмм.

Для инженерно-технических работников, разрабатывающих цифровые устройства автоматики и вычислительной техники.

К 2302030700-152 77-90
046(01)-90

ББК 32.973.2

ISBN 5-256-00250-3

© Мещеряков В. М., Лобов И. Е.,
Глебов С. С. и др., 1990

Большой популярностью у разработчиков быстродействующих микропроцессоров и микроконтроллеров пользуется комплект БИС К1804. Высокие технические характеристики устройств на его основе обусловлены как особенностями схемотехнического базиса, так и удачными архитектурными принципами, заложенными в отдельные БИС и комплект в целом.

Подробно первые микросхемы комплекта К1804 описаны в [1]. К настоящему времени комплект пополнился новыми БИС. Возникла необходимость в массовом издании, где эти микросхемы и вопросы их применения были бы освещены с достаточной полнотой. Однако наличие информации об элементной базе еще не гарантирует успех в разработке микроконтроллера. Какая его структура и какой способ обработки микрокоманд обеспечат заданное быстродействие при минимуме аппаратных затрат? Как лучше распределить функции между аппаратной и микропрограммной реализациями? Эти и многие другие вопросы проектирования встают перед разработчиком. Сжатые сроки, отпускаемые на создание новых изделий в условиях промышленного производства, часто не позволяют проанализировать достаточное число альтернативных вариантов проекта. Результат — нерациональное использование возможностей элементной базы.

В гл. 1 подробно описана элементная база комплекта К1804. В гл. 2 систематизированно представлены разнообразные схемные решения, которые нетрудно адаптировать к специфике конкретных применений.

В гл. 3 нашли отражение вопросы организации встроенного контроля, ибо неуклонно расширяется перечень областей применения БИС К1804, где требуются микропроцессоры не только быстродействующие, но и надежные.

Глава 4 посвящена отладочным комплексам, позволяющим облегчить процесс разработки микропрограмм. Практика показывает, что оснащенность инструментальными средствами во многом определяет как объективную эффективность работы проектировщиков, так и субъективное их отношение к предлагаемой элементной базе.

В заключительной гл. 5 с разной степенью детализации рассмотрены примеры проектных решений. Наиболее подробные из них, изложенные в § 5.3, представлены на основе анализа реальных схем и микропрограмм, разработанных совместно В. В. Грушиным, С. И. Запорожаном и В. В. Новоселовым.

Описание архитектуры элементной базы (гл. 1) выполнено С. С. Глебовым, И. Е. Лобовым, В. М. Мещеряковым и Л. А. Шумиловым. Вопросы проектирования микропрограммных устройств (предисловие, введение, гл. 2—5 и заключение) изложены В. В. Новоселовым.

ВВЕДЕНИЕ

Одним из ключевых факторов, определяющих успех в реализации высокопроизводительных устройств, является выбор элементной базы. Подавляющее большинство разработчиков сегодня являются пользователями элементной базы, производимой крупносерийно. К такой элементной базе относятся и комплекты биполярных микропроцессорных БИС, предназначенные для реализации устройств с микропрограммным управлением. Заказные и полузаказные матричные БИС применяются, как правило, при проектировании устройств специального назначения.

Основным критерием, чаще всего определяющим преимущество биполярных микропрограммируемых БИС перед однокристальными микропроцессорами, является обеспечение повышенной производительности вычислительных средств в конкретной области применения. Биполярные микропрограммируемые БИС оказались наиболее подходящими для реализации следующих устройств:

- контроллеров периферийного оборудования и запоминающих устройств большой емкости, которые в состоянии автономно выполнять весьма сложные функции обработки файлов;

- интеллектуальных графических терминалов и специализированных графических процессоров, обладающих высокой производительностью при решении задач анализа, синтеза и преобразования изображений;

- процессоров мини- и микроЭВМ, способных эмулировать одну или несколько систем команд для обеспечения совместимости с программным обеспечением серийных ЭВМ;

- функциональных расширителей, подключаемых к серийным микроЭВМ в качестве периферийных устройств и позволяющих резко повысить производительность при решении задач соответствующих классов;

- бортовых спецвычислителей;

- контроллеров высокопроизводительных станков с программным управлением и промышленных роботов;

- специализированных процессоров обработки сигналов звукового и ультразвукового диапазонов частот;

- встроенных контроллеров в аппаратуру передачи данных;

- интерфейсных контроллеров локальных сетей;

- базовых вычислительных модулей многопроцессорных систем.

Комплект биполярных микропрограммируемых БИС К1804 развивался в традиционных направлениях, связанных со специализацией микросхем по типам функций, реализуемых микроЭВМ и особенно ее процессором. К таким функциям относятся:

микропрограммное управление (К1804ВУ1, К1804ВУ2, К1804ВУ3, К1804ВУ4, К1804ВУ5);

арифметическая и логическая обработка данных с разрядностью, кратной четырем (К1804ВС1, К1804ВС2, К1804ВР1);

арифметическая и логическая обработка данных с фиксированной разрядностью 16, побайтная и побитовая (К1804ВМ1);

выполнение сдвиговых операций в АЛУ (К1804ВР2);

обработка и хранение признаков состояния (К1804ВР2);

адресная обработка с разрядностью, кратной четырем (К1804ВУ5), в процессорах команд;

управление синхронизацией (К1804ГГ1);

приоритетное прерывание (К1804ВН1, К1804ВР3);

прямой доступ к памяти (К1804ВУ6, К1804ВУ7);

вспомогательные функции ввода-вывода (К1804ИР3, К1804ВА1, К1804ВА2, К1804ВА3) и хранения (К1804ИР1, К1804ИР2) информации.

В табл. В.1 перечислены микросхемы комплекта К1804, с которыми можно ознакомиться по [1]. Следует подчеркнуть, что с появлением новых микросхем серии К1804 они не отошли на «второй план». Вопрос целесообразности применения той или иной микросхемы (например, К1804ВС1, К1804ВС2 или К1804ВМ1) решается в каждом конкретном случае исходя из специфики применения. Исключение составляют микросхемы К1804ВУ1, К1804ВУ2 и К1804ВУ3: при адресации памяти емкостью до 4096 микрокоманд преимущества микросхемы К1804ВУ4 над ними неоспоримы.

Для начинающих разработчиков отметим, что ориентация на комплект К1804 не исключает возможности, а часто и целесообразности его

Т а б л и ц а В1. Состав первой очереди комплекта К1804

Обозначение	Наименование	Число выводов
К1804ВС1	Четырехразрядная микропроцессорная секция	40
К1804ВР1	Схема ускоренного переноса	16
К1804ВУ1	Четырехразрядная секция управления адресом микрокоманд	28
К1804ВУ2	Четырехразрядная секция управления адресом микрокоманд	20
К1804ВУ3	Схема управления следующим адресом	16
К1804ИР1	Четырехразрядный регистр	16
К1804ВС2	Четырехразрядная микропроцессорная секция	48
К1804ВР2	Схема управления состоянием и сдвигами	40
К1804ВУ4	Схема управления последовательностью микрокоманд	40

совместного применения с другими сериями БИС (например, K1802) и сопутствующими микросхемами меньшей степени интеграции—K531, K555, K155 или их более совершенными аналогами.

Управляющие и вычислительные устройства, реализуемые на основе комплекта БИС, можно разделить на два класса: микропрограммные контроллеры (микроконтроллеры) и процессоры команд (микропроцессоры). Принципиальное отличие их состоит не в габаритах устройств и сложности реализуемых ими функций (хотя и это имеет место), а в числе уровней управления процессом обработки данных. В первом из классов такой уровень единственный — уровень микропрограммного управления. Во втором уровней два: системы команд (он же уровень программного управления) и микропрограммного управления.

На рис. В.1 упрощенно показана одна из типовых структур микропрограммного контроллера. Назначение и реализация его компонентов еще будут подробно обсуждаться; здесь нам важно отметить, что прикладной алгоритм (алгоритм функционирования конкретного устройства в конкретном приложении) полностью записан в микропрограммной памяти в виде совокупности микрокоманд, задающих на каждом такте функционирование операционной и управляющей частей микроконтроллера, т. е. блока обработки данных (БОД) и блока микропрограммного управления (БМУ).

Процессор команд в предельно упрощенном варианте изображен на рис. В.2. Часть его компонентов — формирователь адресов микрокоманд (ФАМ), микропрограммная память (МПП), регистр микрокоманд (РгМК) — присутствует и в микропрограммном контроллере.

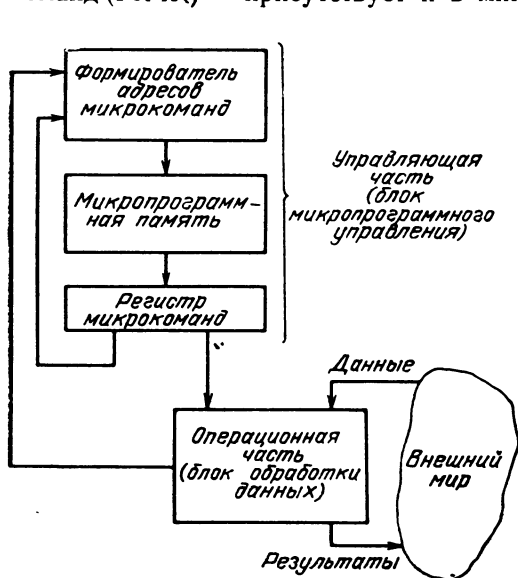


Рис. В.1. Микропрограммный контроллер

Существенным отличием является то, что прикладной алгоритм полностью (или большей частью) записан не в МПП, а в памяти программ, внешней по отношению к процессору. Прикладной алгоритм выполняется в процессоре как последовательность команд программы. В свою очередь, каждая команда интерпретируется микропрограммой, ей соответствующей. Таким образом, в МПП хранятся микропрограммы интерпретации команд программы. Соответствие между кодом команды, поступающим в регистр команд (РгК), и микропрограммой осуществ-

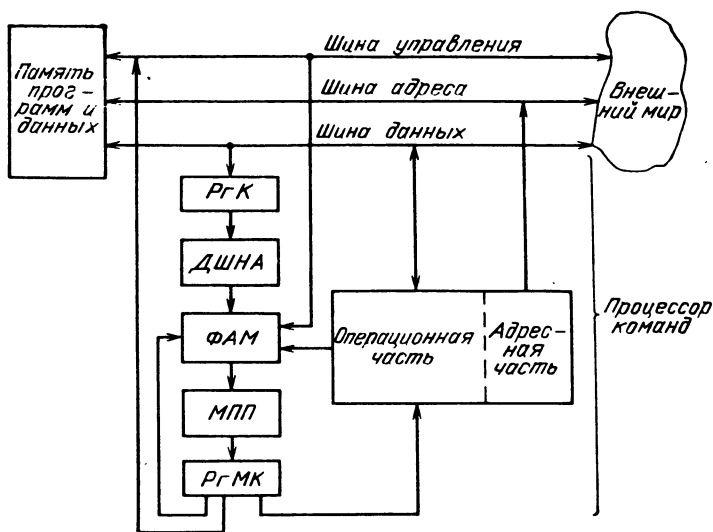


Рис. В.2. Процессор команд

вляется дешифратором начальных адресов (ДШНА) микропрограмм, реализуемым на основе БИС программируемых постоянных запоминающих устройств (ППЗУ) или программируемых логических матриц (ПЛИМ).

В составе процессора команд принято выделять управляющую часть и операционно-адресную часть. В операционно-адресной части выполняются все действия по арифметическо-логической переработке информации, поступающей извне по шине данных, хранению промежуточных результатов, формированию адресов команд, исходных данных и результатов. Для хранения команд программы и данных можно использовать как общую, так и разделенную память.

В процессорах невысокой производительности функции формирования адресов выполняются той же аппаратурой, что и арифметическо-логическая обработка данных (последовательно во времени), а операционно-адресная часть называется просто операционной, или блоком обработки данных. Напротив, в высокопроизводительных процессорах функции адресной обработки возложены на специальную аппаратуру, работающую в режиме предельного временного совмещения с арифметическо-логической обработкой данных, образуя конвейер команд.

Схемотехнические и структурные различия устройств, отражающие специфику конкретных применений, связаны прежде всего с операционной частью; структура же управляющей части весьма консервативна. Особенности прикладных алгоритмов сказываются прежде всего на формате микрокоманды и содержимом МПП. Поэтому информация, по-

лученная из этой книги, может оказаться полезной при разработке микропрограммируемых устройств самого различного назначения.

Итак, комплект биполярных микропрограммируемых БИС К1804 выгодно применять в специализированных вычислительных и управляющих устройствах при повышенных требованиях к их быстродействию (до 8 млн. коротких операций в секунду в контроллерах и до 5 млн. — в процессорах) при отсутствии доминирующих ограничений на энергопотребление устройства (10—30 Вт в зависимости от его сложности). Приведенные цифры годятся только для первоначальной, ориентировочной оценки. Конкретные же значения параметров определяются в ходе реального проектирования: приведенные в книге схемные решения позволяют варьировать их в широком диапазоне.

Глава 1.

ЭЛЕМЕНТНАЯ БАЗА

1.1. ШЕСТНАДЦАТИРАЗЯДНЫЙ ПРОЦЕССОРНЫЙ ЭЛЕМЕНТ

Шестнадцатизрядный биполярный микропрограммируемый процессорный элемент К1804ВМ1 предназначен для построения операционных блоков цифровых вычислительных устройств, ориентированных на высокопроизводительную поразрядную, побайтную и пословную (16 разрядов) обработку данных. Основу его составляет трехходовое 16-разрядное АЛУ со встроенной схемой ускоренного переноса, выполняющее кроме стандартных арифметических и логических операций особые операции: установку требуемого бита в 1 или 0, проверку значения бита, циклический сдвиг и слияние, циклический сдвиг и сравнение, генерацию циклического избыточного кода и т. д. Регистровое запоминающее устройство на кристалле имеет емкость тридцать два 16-разрядных слова. Комбинационный сдвигатель позволяет выполнять за один такт циклический сдвиг (в сторону старших разрядов) как младшего байта, так и всего 16-разрядного слова на любое число разрядов (от 1 до 15). Восьмиразрядный регистр состояния служит для хранения переноса (*C*), переполнения (*OVR*), знака (*N*), нуля (*Z*), бита связи (*L*), а также трех флажков (*FL1*, *FL2*, *FL3*), определяемых пользователем. Мультиплексор условия предназначен для передачи одного из 12 различных условий (признаков) в блок микропрограммного управления. Имеются две шины данных, одна из которых (*I0—I15*), используется как для ввода данных, так и для ввода инструкции, а другая (*Y0—Y15*) — двунаправленная для ввода-вывода данных. Для хранения промежуточных результатов вычислений предусмотрены аккумуляторы и регистр данных.

Микросхема К1804ВМ1 изготавливается по технологии биполярных схем с общим эмиттером, за исключением интерфейса, который выполнен совместимым с ТТЛ-схемами, и выпускается в 52-выводном корпусе. Типовое значение длительности выполнения любой инструкции не превышает 100 нс. Источник питания +5 В, потребляемая мощность около 3,5 Вт. Для синхронизации требуется одна синхропоследовательность. Цоколевка микросхемы показана на рис. 1.1.

Назначение выводов микросхемы

$I0—I15$ — вход инструкции. Определяет выполняемую операцию, источники операндов, приемники результата. При выполнении двухтактных инструкций на втором такте может быть применен для ввода данных.

$Y0—Y15$ — двунаправленная шина данных. Используется как для ввода данных (при $\overline{OEY}=1$), так и для вывода данных (при $\overline{OEY}=0$).

\overline{OEY} — вход разрешения вывода данных. При $\overline{OEY}=0$ разрешается вывод данных через шину Y . При $\overline{OEY}=1$ вывод данных через шину Y запрещен, поэтому через нее можно осуществить ввод данных в процессорный элемент K1804BM1.

$T1—T4$ — вход управления выбором кода условия (при $OET=0$) либо выход признаков состояния (при $OET=1$), считываемых из регистра состояния.

OET — вход разрешения вывода признаков состояния. При $OET=1$ разрешается вывод признаков состояния через шину $T1—T4$. При $OET=0$ вывод признаков запрещен, и шина $T1—T4$ может использоваться как входная.

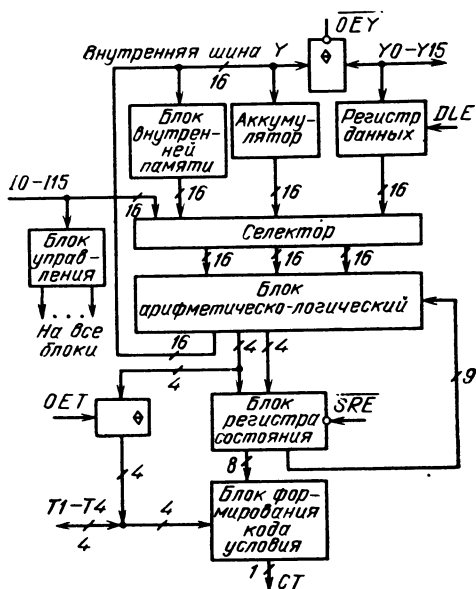
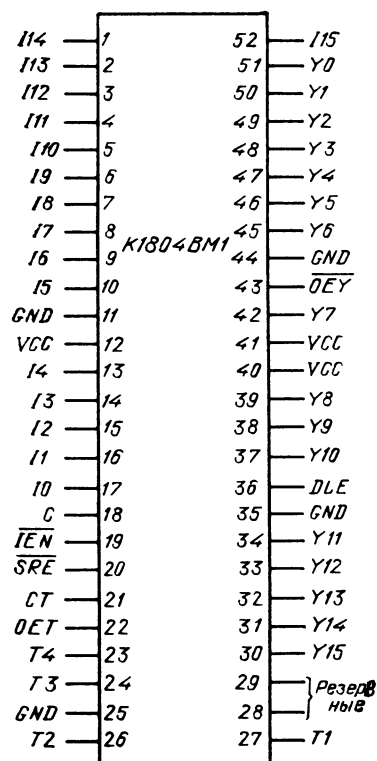


Рис. 1.2. Структура БИС K1804BM1

Рис. 1.1. Цоколевка микросхемы K1804BM1

\overline{SRE} — вход разрешения записи в регистр состояния. При $\overline{SRE}=0$ и $\overline{IEN}=0$ запись разрешена, а при $\overline{SRE}=1$ запрещена.

\overline{IEN} — вход разрешения инструкции. При $\overline{IEN}=1$ запрещается запись в регистровое запоминающее устройство, аккумулятор и регистр состояния независимо от выполняемой инструкции. Это позволяет использовать поле инструкции 10—115 в микрокоманде для других приемников.

DLE — вход разрешения регистра данных. При $DLE=1$ информация со входа регистра передается на его выход, а при $DLE=0$ на выходе регистра сохраняется то состояние, которое было на его входе в момент перехода сигнала DLE из «1» в «0».

CT — выход условия. Используется для вывода значения условия из процессорного элемента.

C — вход тактовый.

VCC — вывод питания.

GND — вывод общий.

Структурная схема БИС представлена на рис. 1.2. В ней можно выделить семь основных блоков: блок внутренней памяти; аккумулятор; регистр данных; блок арифметическо-логический; блок регистра состояния; блок формирования кода условия; блок управления.

Блок внутренней памяти (БВП) (рис. 1.3) состоит из однопортового регистрового запоминающего устройства (РЗУ) емкостью тридцать два 16-разрядных слова и выходного 16-разрядного регистра (РгА). Этот блок может использоваться в качестве источника операндов АЛУ и приемника результата. Регистровое запоминающее устройство содержит дешифратор адреса (ДША), тридцать два 16-разрядных регистра общего назначения (РОН) и схему записи-считывания. Выбор любого из РОН БВП в качестве источника или приемника информации осуществляется передачей соответствующих сигналов инструкции на адресный вход РЗУ и преобразованием их с помощью дешифратора адреса (ДША). При считывании информация из выбранного РОН передается в 16-разрядный РгА на выходе РЗУ, который построен на триггерах типа «защелка», управляемых тактовым сигналом C . При $C=1$ информация со входа РгА передается на выход РгА. При $C=0$ на выходе РгА сохраняется то состояние, которое было на входе РгА в момент перехода сигнала C из «1» в «0». Эта информация сохраняется

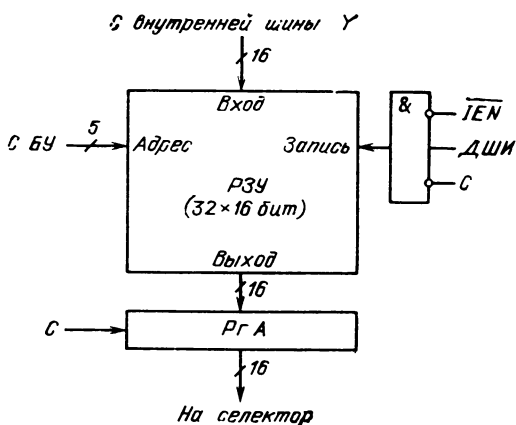


Рис. 1.3. Блок внутренней памяти

на выходе РгА до тех пор, пока на входе C не будет установлена «1». Информация, считанная из БВП, поступает на входы мультиплексоров БАЛ и может быть использована в качестве операндов R, S, U АЛУ. Запись информации в РЗУ производится при выполнении инструкции, определяющей РЗУ как приемник результата, и при наличии «0» на входе разрешения (\overline{IEN}) и на тактовом входе (C). Наличие «1» на входе \overline{IEN} запрещает запись в РЗУ. Моментом начала записи является переход тактового сигнала с уровня «1» на «0». С этого момента фиксируется информация на выходе РгА, что устраняет неопределенность в кольце передачи информации с выхода БВП на его вход. При выполнении операций с целым словом в РЗУ записываются все 16 разрядов результата, при операциях с байтами выполняется запись лишь в младшие 8 разрядов РОН, а старшая половина слова остается неизменной.

Аккумулятор (АК) (рис. 1.4) представляет собой 16-разрядный регистр, построенный на триггерах, запись информации в которые производится по фронту тактового сигнала C . АК используется как один из источников операндов R, S, U АЛУ и приемник результата. Запись информации в АК производится при выполнении инструкции, определяющей аккумулятор как приемник результата при наличии «0» на входе разрешения инструкции (\overline{IEN}) по фронту тактового сигнала C . Наличие «1» на входе \overline{IEN} запрещает запись в АК независимо от выполняемой инструкции. При операциях с целым словом в АК записываются все 16 разрядов результата. При операциях с байтами производится запись лишь в младшие 8 разрядов АК, а старшие 8 разрядов содержимого АК остаются неизменными.

Шестнадцатиразрядный регистр данных (РгД) (рис. 1.5) построен на триггерах типа «защелка», управляемых сигналом разрешения РгД (DLE). Регистр используется для хранения данных, поступающих в БИС через двунаправленную шину Y . При $DLE = 1$ информация со входа РгД передается на его выход. При $DLE = 0$ на выходе РгД сохраняется то состояние, которое было на входе РгД в момент перехода сигнала DLE из «1» в «0». Эта информация сохраняется на выходе РгД

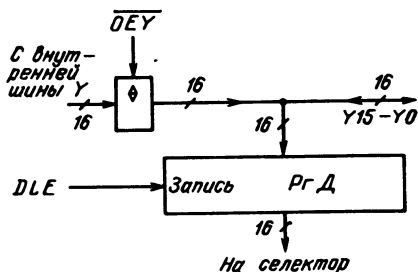
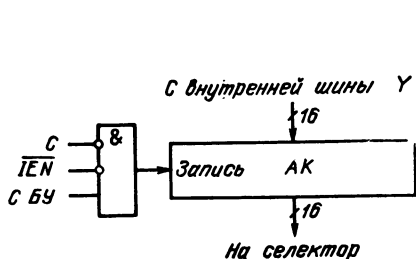


Рис. 1.4. Управление аккумулятором

Рис. 1.5. Управление регистром данных

зависимости от кода инструкции 10—115. Источниками операндов R и U являются РЗУ, АК и РгД, а источниками операнда S — РЗУ, АК и шина 10—115, которая используется в качестве источника операнда при выполнении двухтактных инструкций. На выходе мультиплексора расположен сдвигатель, осуществляющий циклический сдвиг операнда на заданное число разрядов (от 1 до 16) в сторону старших разрядов. При выполнении операций с целым словом производится сдвиг всех 16 разрядов, а при операциях с байтами — лишь 8 младших разрядов. АЛУ выполняет операции с одним, двумя и тремя операндами: передачу операнда, инвертирование, сложение, вычитание, логические операции (И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ и т. д.), циклический сдвиг и слияние, циклический сдвиг и сравнение, генерацию циклического кода и т. д. Все операции в АЛУ могут производиться как с полными словами, так и с байтами (с младшими 8 разрядами слова). Более подробно набор инструкций будет описан ниже. АЛУ формирует три признака состояния: перенос (C), знак (N), переполнение (OV). Четвертый признак состояния — ноль (Z) — вырабатывается формирователем признака нуля (ФПН). Формирование признаков состояния производится как для слов, так и для байтов. Признаки состояния (C , N , OV , Z) могут быть записаны в один из РОН БВП или в РгС, а также выведены через двунаправленную шину $T1$ — $T4$, если требуется более сложная обработка информации о состоянии. При $OET = 1$ разрешается вывод признаков через шину $T1$ — $T4$. При $OET = 0$ шина $T1$ — $T4$ может использоваться как входная. Мультиплексор входного переноса формирует сигнал переноса в АЛУ, выбирая значения 0, 1 или сигнал переноса (C) из регистра состояния. Применение последнего дает возможность выполнять операции над числами произвольной разрядности, большей шестнадцати.

Приоритетный шифратор (ПШ) формирует на выходе двоичный код, который определяет номер самого старшего разряда, имеющего единичное значение. Сигналы, поступающие на вход ПШ, представляют собой результат поразрядной операции И, выполняемой в АЛУ над операндом R и инвертированным значением операнда S (рис. 1.7). Выходные значения сигналов ПШ приведены в табл. 1.1, 1.2.

Выходной мультиплексор передает на выход БАЛ информацию о выходе АЛУ, ПШ или регистра состояния. В зависимости от выполняемой инструкции эта информация может быть записана в РЗУ, аккумулятор или выведена через двунаправленную тристабильную шину Y , которая управляется сигналом разрешения вывода информации (\overline{OEY}). При $\overline{OEY} = 0$ разрешается вывод информации через шину Y , при $\overline{OEY} = 1$ шина Y может быть использована как входная.

Блок регистра состояния (БРС, рис. 1.8) содержит 8-разрядный регистр состояния (РгС) и мультиплексор (MUX). Регистр состояния построен на триггерах, запись информации в которые производится по фронту тактового сигнала C при «0» на входе разрешения записи в РгС (\overline{SRE}) и на входе разрешения микрокоманды (\overline{IEN}). При $\overline{IEN} =$

Таблица 1.1. Функции приоритетного шифратора для операций с полным словом

Номер старшего разряда, в котором находится «1»	Выход ПШ
Нет	00000
15	00001
14	00010
13	00011
12	00100
11	00101
10	00110
9	00111
8	01000
7	01001
6	01010
5	01011
4	01100
3	01101
2	01110
1	01111
0	10000

Примечание. Если во всех разрядах операнда «0», то выход шифратора равен 00000.

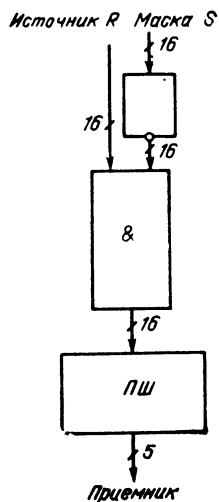


Рис. 17. Устройство приоритетного шифратора

$\overline{SRE} = 1$ или $\overline{SRE} = 1$ запись в PгC запрещена. В младшие четыре разряда PгC могут быть записаны признаки состояния АЛУ: нуль (Z), перенос (C), знак (N), переполнение (OVR) или четыре младших бита данных с шины Y. Источник информации для четырех младших разрядов PгC определяется мультиплексором на входе PгC в зависимости от выполняемой инструкции. Запись в младшие четыре разряда PгC производится после выполнения всех инструкций, за исключением следующих: «нет операции», «хранение содержимого PгC», «проверка состояния» и «установка в 1» или «установка в 0» старших битов PгC. В старших че-

Таблица 1.2. Функции приоритетного шифратора при операциях с байтом

Номер старшего разряда, в котором находится «1»	Выход ПШ	Номер старшего разряда, в котором находится «1»	Выход ПШ
Нет	00000	3	00101
7	00001	2	00110
6	00010	1	00111
5	00011	0	01000
4	00100		

Примечания: 1. Если во всех разрядах операнда «0», то выход шифратора равен 00000. 2. При операциях с байтами разряды 8–15 операнда не используются.

тырех разрядах PgC хранятся бит связи (L) и три флажка ($FL1$, $FL2$, $FL3$), определяемые пользователем. Запись в старшие разряды PgC производится с шины Y после выполнения инструкций «установка в 1» или «установка в 0» старших разрядов PgC , а также «загрузка PgC » при операциях с целым словом. Запись бита связи (L) выполняется также после каждой инструкции сдвига.

Двухнаправленная шина Y используется и для ввода данных в PgC (при $\overline{OEY} = 1$) и для вывода информации из PgC (при $\overline{OEY} = 0$) в зависимости от выполняемой инструкции. При осуществлении операций с целым словом в PgC записываются все восемь признаков состояния (Z , C , N , OVR , L , $FL1$, $FL2$, $FL3$), а при операциях с байтами — лишь четыре признака состояния (Z , C , L , OVR). Содержимое PgC может быть записано в один из РОН или в аккумулятор. При этом если выполняется операция с целым словом, то в РОН или в АК записываются все восемь признаков состояния в качестве младших разрядов слова, а старшие восемь разрядов слова устанавливаются в «0». При операциях с байтами содержимое PgC также записывается в младшие разряды РОН или АК, содержимое же старших разрядов остается неизменным. Это удобно для сохранения и восстановления состояния при реализации прерываний и выполнении подпрограмм.

Блок формирования кода условия (БФКУ), представленный на рис. 1.9, состоит из формирователя кодов условий (ФКУ), мультиплексора условий ($MUXY$) и мультиплексора (MUX), управляющего мультиплексором условий. Формирователь кодов условий вырабатывает 12 возможных условий, а мультиплексор условия обеспечивает

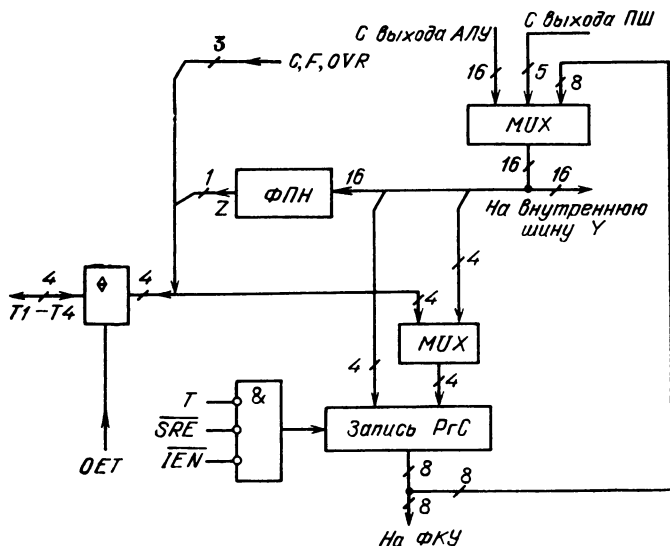


Рис. 1.8. Блок регистра состояния

прохождение требуемого условия на выход СТ под действием управляющих сигналов с выхода мультиплексора управления выбором условия. Управление выбором требуемого кода условия осуществляется двумя способами: с помощью специальной инструкции, что делает невозможным одновременное выполнение операций в АЛУ, и через шину $T1-T4$, что увеличивает разрядность микрокоманды, но позволяет одновременно с выполнением операций в АЛУ осуществлять проверку условия.

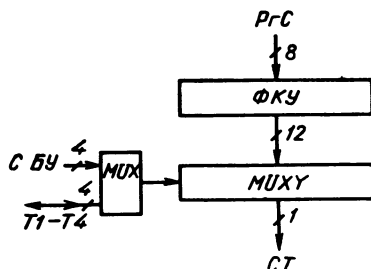


Рис. 1.9. Схема управления выбором условия СТ

Блок управления содержит регистр инструкций (РгИ) и дешифратор инструкций (ДШИ). Шестнадцатиразрядный РгИ построен на триггерах типа «защелка», которые передают сигналы инструкции со входов $10-115$ в ДШИ для преобразования их во внутренние сигналы, управляющие работой блоков БИС. Почти все инструкции выполняются за один такт. Исключение составляют двухтактные инструкции с непосредственным операндом. На первом такте ДШИ определяет, что выполняется двухтактная инструкция, и запирает РгИ, в котором сохраняется инструкция, записанная на первом такте. На втором такте со входов $10-115$ операнд поступает на входы БАЛ, где с использованием его в качестве операнда S АЛУ выполняется операция, определяемая инструкцией, записанной в РгИ на предыдущем такте. В конце второго такта производится отпираание РгИ.

Система инструкций микропроцессорной БИС включает 11 групп инструкций: с одним операндом; с двумя операндами; сдвига на один разряд; с битами; шифрации; циклического сдвига на n разрядов ($1 \leq n \leq 16$); циклического сдвига и слияния; циклического сдвига и сравнения; генерации циклического избыточного кода (CRC); установки и проверки состояния; инструкцию «нет операции». Все мнемоники инструкций могут использоваться при ассемблировании микропрограмм с использованием микроассемблера *AMDASM*, описанного в гл. 4.

Инструкции с одним операндом. Формат однооперандных инструкций включает четыре поля: режим (байт B или слово W), код операции, источник и приемник. Инструкции делятся на два типа: в первом (*SOR*) используется РЗУ как источник и/или приемник, во втором (*SONR*) РЗУ не используется. Форматы инструкций:

	15	14—13	12—9	8—5	4—0
<i>SOR</i>	B/W	10	КОП	ИСТ, ПРМ	Адрес РЗУ
<i>SONR</i>	B/W	11	КОП	ИСТ	ПРМ

Здесь и далее ИСТ означает источник операнда, ПРМ — приемник результата. Пятиразрядный двоичный код в поле адреса РЗУ указывает регистр операнда ($R00—R31$). Если не указано иначе, то инструкция выполняется как с байтом ($B/W = 0$), так и с полным словом ($B/W = 1$).

После выполнения заданной функции результат направляется в указанный приемник и/или помещается на шину Y . Если требуется преобразование 8-разрядного операнда в 16-разрядный, схема K1804BM1 может выполнять расширение знакового разряда или двоичного нуля на 16-разрядное слово. На младшие четыре бита RrC (OVR, C, N, Z) данная функция воздействует, на старшие четыре бита — нет. В аспекте адресации единственным ограничением является невозможность использования РЗУ в качестве источника, если АК и RrC одновременно определены как приемники. В табл. 1.3 — 1.5 приведено описание однооперандных инструкций.

Инструкции с двумя операндами. Двухоперандные инструкции задаются пятью полями: режим, КОП, источник R , источник S , приемник. Далее инструкции делятся на два типа: первый использует РЗУ в качестве источника и или приемника, а второй не использует. Инструкции первого типа имеют два формата ($TOR1$ и $TOR2$), единственное отличие которых заключается в значении квадранта (специального поля, служащего для кодирования форматов). Под управлением входов инструкции над описанными источниками выполняется требуемая функция, после чего результат заносится в описанный приемник и/или выводится через шину Y . Младшие четыре бита RrC (OVR, N, C, Z) модифицируются в результате выполнения арифметических функций. При выполнении логических функций модифицируются биты N и Z , а биты OVR и C RrC устанавливаются в нуль. Инструкции сложения и вычитания с учетом переноса удобны при выполнении операций с многократной точностью. Форматы инструкций:

	15	14—13	12—9	8—5	4—0
$TOR1$	B/W	00	ИСТ, ИСТ, ПРМ	КОП	Адрес РЗУ
$TOR2$	B/W	10	ИСТ, ИСТ, ПРМ	КОП	Адрес РЗУ
$TONR$	B/W	11	ИСТ, ПРМ	КОП	ПРМ

В табл. 1.6—1.9 содержится описание двухоперандных инструкций.

Инструкции сдвига на один разряд. Эти инструкции задаются четырьмя полями: режим (байт/слово), управление сдвигом, источник, приемник. Далее они делятся на два типа: первый использует РЗУ в

Таблица 1.3. Инструкции с одним операндом; тип инструкций *SOR*

КОП			ИСТ, ПРМ			
Код	Мнемоника	Операция	Код	Мнемоника	Источник	Приемник
1100	<i>MOVE</i>	$\overline{\text{ИСТ}} \rightarrow \text{ПРМ}$	0000	<i>SORA</i>	РЗУ	АК
1101	<i>COMP</i>	$\overline{\text{ИСТ}} \rightarrow \text{ПРМ}$	0010	<i>SORY</i>	РЗУ	Шина Y
1110	<i>INC</i>	$\overline{\text{ИСТ}} + 1 \rightarrow \text{ПРМ}$	0011	<i>SORS</i>	РЗУ	РгС
1111	<i>NEG</i>	$\overline{\overline{\text{ИСТ}}} + 1 \rightarrow \text{ПРМ}$	0100	<i>SOAR</i>	АК	РЗУ
			0110	<i>SODR</i>	РгД	РЗУ
			0111	<i>SOIR</i>	I	РЗУ
			1000	<i>SOZR</i>	0	РЗУ
			1001	<i>SOZER</i>	РгД.0	РЗУ
			1010	<i>SOSE</i>	РгД.S	РЗУ
			1011	<i>SORR</i>	РЗУ	РЗУ

Примечание. РгД.0 означает расширение байта до слова нулями, а РгД.S — значением знакового (N) разряда; ИСТ — источник; ПРМ — приемник, черта сверху $\overline{\text{ИСТ}}$ означает инверсию содержимого источника.

Таблица 1.4. Инструкции с одним операндом; тип инструкций *SONR*

КОП			ИСТ, ПРМ			Приемник		
Код	Мнемоника	Операция	Код	Мнемоника	Источник	Код	Мнемоника	
1100	<i>MOVE</i>	$\overline{\text{ИСТ}} \rightarrow \text{ПРМ}$	0100	<i>SOA</i>	АК	00000	<i>NRV</i>	Шина Y
1101	<i>COMP</i>	$\overline{\text{ИСТ}} \rightarrow \text{ПРМ}$	1100	<i>SOD</i>	РгД	00001	<i>NRA</i>	АК
1110	<i>INC</i>	$\overline{\text{ИСТ}} + 1 \rightarrow \text{ПРМ}$	0111	<i>SOI</i>	I	00100	<i>NRS</i>	РгС
1111	<i>NEG</i>	$\overline{\overline{\text{ИСТ}}} + 1 \rightarrow \text{ПРМ}$	1000	<i>SOZ</i>	0	00101	<i>NRAS</i>	АК, РгС
			1001	<i>SOZE</i>	РгД.0			
			1010	<i>SOSE</i>	РгД.S			

Таблица 1.5. Управление шиной Y и регистром состояния в однооперандных инструкциях; типы инструкций *SOR*, *SONR*

КОП	Операция	Шина Y
<i>MOVE</i>	$\overline{\text{ИСТ}} \rightarrow \text{ПРМ}$	$Y \leftarrow \overline{\text{ИСТ}}$
<i>COMP</i>	$\overline{\text{ИСТ}} \rightarrow \text{ПРМ}$	$Y \leftarrow \overline{\text{ИСТ}}$
<i>INC</i>	$\overline{\text{ИСТ}} + 1 \rightarrow \text{ПРМ}$	$Y \leftarrow \overline{\text{ИСТ}} + 1$
<i>NEG</i>	$\overline{\overline{\text{ИСТ}}} + 1 \rightarrow \text{ПРМ}$	$Y \leftarrow \overline{\overline{\text{ИСТ}}} + 1$

Примечание. Разряды Z, C, N, OVR регистра состояния модифицируются, остальные не изменяются.

Т а б л и ц а 1.6. Двухоперандные инструкции; тип инструкций TOR1

ИСТ, ИСТ, ПРМ					КОП		
Код	Мнемоника	R	S	ПРМ	Код	Мнемоника	Операция
00000	TORAA	PЗУ	AK	AK	0000	SUBR	$S - R$
0010	TORIA	PЗУ	I	AK	0001	SUBRC	$S - R + C$
0011	TODRA	PrД	PЗУ	AK	0010	SUBS	$R - S$
1000	TORAY	PЗУ	AK	Шина Y	0011	SUBSC	$R - S + C$
1010	TORIY	PЗУ	I	Шина Y	0100	ADD	$R + S$
1011	TODRY	PrД	PЗУ	Шина Y	0101	ADDC	$R + S + C$
1100	TORAR	PЗУ	AK	PЗУ	0110	AND	$R \& S$
1110	TORIR	PЗУ	I	PЗУ	0111	NAND	$\overline{R \& S}$
1110	TODRR	PrД	PЗУ	PЗУ	1000	EXOR	$R \oplus S$
					1001	NOR	$\overline{R \vee S}$
					1010	OR	$R \vee S$
					1011	EXNOR	$\overline{R \oplus S}$

Т а б л и ц а 1.7. Двухоперандные инструкции; тип инструкций TOR2

ИСТ, ИСТ, ПРМ					КОП		
Код	Мнемоника	R	S	ПРМ	Код	Мнемоника	Операция
0001	TODAR	PrД	AK	PЗУ	0000	SUBR	$S - R$
0010	TOAIR	AK	I	PЗУ	0001	SUBRC	$S - R + C$
0101	TODIR	PrД	I	PЗУ	0010	SUBS	$R - S$
					0011	SUBSC	$R - S + C$
					0100	ADD	$R + S$
					0101	ADDC	$R + S + C$
					0110	AND	$R \& S$
					0111	NAND	$\overline{R \& S}$
					1000	EXOR	$R \oplus S$
					1001	NOR	$\overline{R \vee S}$
					1010	OR	$R \vee S$
					1011	EXNOR	$\overline{R \oplus S}$

Т а б л и ц а 1.8. Двухоперандные инструкции; тип инструкций TONR

ИСТ, ИСТ, ПРМ				КОП			Приемник	
Код	Мнемоника	R	S	Код	Мнемоника	Операция	Код	Мнемоника
0001	TODA	PrД	AK	0000	SUBR	$S - R$	00000	NRVY
0010	TOAI	AK	I	0001	SUBRC	$R - S + C$	00010	NRRA
0101	TODI	PrД	I	0010	SUBS	$R - S$	00100	NRNS
				0011	SUBSC	$R - S + C$	00101	NRAS
				0100	ADD	$R + S$		
				0101	ADDC	$R + S + C$		
				0110	AND	$R \& S$		
				0111	NAND	$\overline{R \& S}$		
				1000	EXOR	$R \oplus S$		
				1001	NOR	$\overline{R \vee S}$		
				1010	OR	$R \vee S$		
				1011	EXNOR	$\overline{R \oplus S}$		

Таблица 1.9. Управление шиной Y и регистром состояния в двухоперандных инструкциях; типы инструкций $TOR1$, $TOR2$, $TONR$

КОП	Операция	Шина Y	PrC			
			Z	C	N	OVR
$SUBR$	$S - R$	$Y \leftarrow S - R$	Модификация			
$SUBRC$	$S - R + C$	$Y \leftarrow S + \overline{R} - 1 + C$				
$SUBS$	$R - S$	$Y \leftarrow R - S$				
$SUBSC$	$R - S + C$	$Y \leftarrow R + \overline{S} - 1 + C$				
ADD	$R + S$	$Y \leftarrow R + S$				
$ADDC$	$R + S + C$	$Y \leftarrow R + S + C$				
AND	$R \& S$	$Y_i \leftarrow R_i \& S_i$	0	U	0	U
$NAND$	$\overline{R \& S}$	$Y_i \leftarrow \overline{R_i \& S_i}$	0	U	0	U
$EXOR$	$R \oplus S$	$Y_i \leftarrow R_i \oplus S_i$	0	U	0	U
NOR	$\overline{R \vee S}$	$Y_i \leftarrow \overline{R_i \vee S_i}$	0	U	0	U
OR	$R \vee S$	$Y_i \leftarrow R_i \vee S_i$	0	U	0	U
$EXNOR$	$\overline{R \oplus S}$	$Y_i \leftarrow \overline{R_i \oplus S_i}$	0	U	0	U

Примечание. U — модификация разряда; разряды L , $FL1$, $FL2$, $FL3$ не изменяются.

качестве источника и/или приемника, второй — нет. Под управлением входов инструкции требуемая функция выполняется над операндом источника и результат направляется в приемник и/или на шину Y . В поле управления сдвигом указывается как направление сдвига (в сторону старших или младших разрядов), так и правило заполнения освобождающегося при сдвиге бита. При сдвиге в сторону старших разрядов в самый младший разряд может загружаться нуль, единица или бит связи (L) из PrC. Самый старший разряд загружается в бит связи (C) или бита связи (L) PrC, а также функцией $N \oplus OVR$ битов PrC. Младший разряд загружается в бит связи (L) PrC (рис. 1.11). Биты N и Z регистра состояния модифицируются, а биты OVR и C устанавливаются в нуль. Функция $N \oplus OVR$ используется при умножении чисел в дополнительном коде. Форматы инструкций:

	15	14—13	12—9	8—5	4—0
$SHFTR$	B/W	10	ИСТ, ПРМ	КОП	Адрес РЗУ
$SHFTNR$	B/W	11	ИСТ	КОП	ПРМ

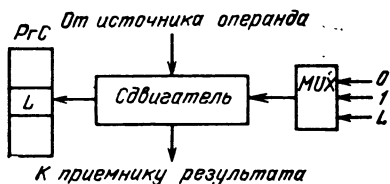


Рис. 1.10. Реализация сдвига влево

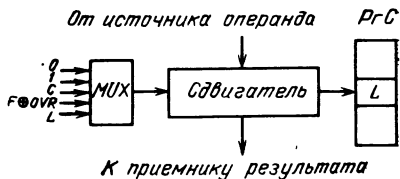


Рис. 1.11. Реализация сдвига вправо

В табл. 1.10—1.12 описаны инструкции сдвига на один разряд, выполняемые схемой K1804BM1.

Инструкции с битами. Инструкции задаются четырьмя полями: режим (байт/слово), операция, источник/приемник и позиция бита, который должен быть обработан (младшим полагается бит 0). Эти инструкции делятся на два типа. В первом используется РЗУ как источник и приемник, причем данный тип представлен двумя форматами —

Таблица 1.10. Инструкции сдвига на один разряд; тип инструкций *SHFTR*

ИСТ, ПРМ				КОП			
Код	Мнемоника	ИСТ	ПРМ	Код	Мнемоника	Направление сдвига	Вдвигаемое значение
0110	<i>SHRR</i>	РЗУ	РЗУ	0000	<i>SHUPZ</i>	Вправо	0
0111	<i>SHDR</i>	РгД	РЗУ	0001	<i>SHUP1</i>	»	1
				0010	<i>SHUPL</i>	»	L
				0100	<i>SHDNZ</i>	Влево	0
				0101	<i>SHDN1</i>	»	1
				0110	<i>SHDNL</i>	»	L
				0111	<i>SHDNC</i>	»	C
				1000	<i>SHDNOV</i>	»	$N \oplus OVR$

Таблица 1.11. Инструкции сдвига на один разряд; тип инструкций *SHFTNR*

ИСТ, ПРМ			КОП				Приемник		
Код	Мнемоника	ИСТ	Код	Мнемоника	Направление сдвига	Вдвигаемое значение	Код	Мнемоника	
0110	<i>SHA</i>	АК	0000	<i>SHUPZ</i>	Вправо	0	00000	<i>NRV</i>	Шина Y
0111	<i>SHD</i>	РгД	0001	<i>SHUP1</i>	»	1			
			0010	<i>SHUPL</i>	»	L	00001	<i>NRA</i>	АК
			0100	<i>SHDNZ</i>	Влево	0			
			0101	<i>SHDN1</i>	»	1			
			0110	<i>SHDNL</i>	»	L			
			0111	<i>SHDNC</i>	»	C			
			1000	<i>SHDNOV</i>	»	$N \oplus OVR$			

Т а б л и ц а 1.12. Управление шиной Y и регистром состояния при сдвиге на один разряд; типы инструкций $SHFTR$ и $SHFTNR$

Мнемоника	Операция	Режим	Шина	PrC	
				N	L
$SHUPZ$ $SHUP1$ $SHUPL$	Влево, 0 Влево, 1 Влево, G	1 — слово (W)	$Y_i \leftarrow \text{ИСТ}_{i-1},$ $i = 1-15;$ $Y_0 \leftarrow \text{вход при сдвиге}$	ИСТ_{14}	ИСТ_{15}
		0 — байт (B)	$Y_i \leftarrow \text{ИСТ}_{i-1},$ $i = 1-7;$ $Y_0 \leftarrow \text{вход при сдвиге};$ $Y_8 \leftarrow \text{ИСТ}_7,$ $Y_i \leftarrow \text{ИСТ}_{i-8},$ $i = 9-15$	ИСТ_6	ИСТ_7
$SHDNZ$ $SHDN1$ $SHDNL$ $SHDNC$	Вправо, 0 Вправо, 1 Вправо, L Вправо, C	1 — слово (W)	$Y_i \leftarrow \text{ИСТ}_{i+1},$ $i = 14-0;$ $Y_{15} \leftarrow \text{вход при сдвиге}$	Вход при сдвиге	ИСТ_0
$SHDNOV$	Вправо, $N \oplus \oplus OVR$	0 — байт (B)	$Y_i \leftarrow \text{ИСТ}_{i+1},$ $i = 1-6;$ $Y_i \leftarrow \text{ИСТ}_{i-7},$ $i = 8-14;$ $Y_7, Y_{15} \leftarrow \text{вход при сдвиге}$	То же	

Примечания: 1. Нижний индекс у ИСТ определяет соответствующий разряд источника.

2. Разряды $FL1, FL2, FL3$ регистра состояния не изменяются, разряды C и OVR устанавливаются в нуль.

BOR1 и BOR2. Во втором типе РЗУ не используется. Под управлением входов инструкции над операндом из заданного источника выполняется требуемая функция, после чего результат заносится в указанный приемник и/или выдается по шине Y . Могут выполняться такие операции, как установка n -го разряда в «1» или сброс в «0» при сохранении значений остальных разрядов; проверка n -го разряда (установка бита Z PrC в состояние, определяемое значением n -го разряда); загрузка 2^n (единицы в n -й разряд и нулей в остальные); загрузка инверсии 2^n (нуля в n -й разряд и единиц в остальные разряды); инкремент на 2^n (сложение константы 2^n с операндом); декремент на 2^n (вычитание константы 2^n из значения операнда). При реализации всех команд загрузки, установки, сброса и проверки модифицируются биты N и Z PrC, а биты OVR и C устанавливаются в нуль. При выполнении всех арифметических операций биты OVR, C, N, Z регистра состояния модифицируются. Форматы инструкций:

	15	14—13	12—9	8—5	4—0
<i>BOR1</i>	<i>B/W</i>	11	<i>n</i>	КОП	Адрес РЗУ
<i>BOR2</i>	<i>B/W</i>	10	<i>n</i>	КОП	Адрес РЗУ
<i>BONR</i>	<i>B/W</i>	11	<i>n</i>	1100	КОП

Подробное описание инструкций с битами приведено в табл. 1.13—1.18.

Инструкции циклического сдвига на n разрядов. Эти инструкции задаются четырьмя полями: режим, источник, приемник и число бит, на которое нужно сдвинуть операнд. Инструкции делятся на два типа: первый использует РЗУ в качестве источника или приемника, а второй не использует. Первый тип описывается двумя форматами (*ROTR1* и *ROTR2*), отличающимися только квадрантом. Второй тип имеет единственный формат. Под управлением входов инструкции операнд источника циклически сдвигается на n позиций ($0 \leq n \leq 15$) в сторону старших разрядов, а результат помещается в приемник и/или на шину Y . В режиме полного слова (W) сдвигаются все 16 бит, в байтовом (B) режиме — только биты 0—7. В режиме полного слова циклический сдвиг в сторону старших разрядов на n позиций эквивалентен циклическому сдвигу в сторону младших разрядов на $16 - n$ позиций. Аналогично в байтовом режиме сдвиг в сторону старших разрядов на n позиций эквивалентен сдвигу в сторону младших разрядов на $8 - n$ позиций. Биты N и Z PrC модифицируются, а биты C и OVR устанавливаются в нуль. Форматы инструкций:

	15	14—13	12—9	8—5	4—0
<i>ROTR1</i>	<i>B/W</i>	00	<i>n</i>	ИСТ, ПРМ	Адрес РЗУ
<i>ROTR2</i>	<i>B/W</i>	01	<i>n</i>	ИСТ, ПРМ	Адрес РЗУ
<i>ROTRN</i>	<i>B/W</i>	11	<i>n</i>	1100	ИСТ, ПРМ

Таблица 1.13. Инструкции с битами;
тип инструкций *BOR1*

Код	Мнемоника	Операция
1101	<i>SETNR</i>	Установка в «1» РЗУ
1110	<i>RSTNR</i>	Сброс в «0» РЗУ
1111	<i>TSTNR</i>	Проверка РЗУ

Таблица 1.14. Инструкции
с битами; тип инструкций *BOR2*

Код	Мнемоника	Операция
1100	<i>LD2NR</i>	$2^n \rightarrow \text{РЗУ}$
1101	<i>LDC2NR</i>	$\overline{2^n} \rightarrow \text{РЗУ}$
1110	<i>A2NR</i>	$\text{РЗУ} + 2^n \rightarrow \text{РЗУ}$
1111	<i>S2NR</i>	$\text{РЗУ} - 2^n \rightarrow \text{РЗУ}$

Таблица 1.15. Инструкции с битами; тип инструкций *BONR*

Код	Мнемоника	Операция	Код	Мнемоника	Операция
00000	<i>TSTNA</i>	Проверка АК	10000	<i>TSTND</i>	Проверка РгД
00001	<i>RSTNA</i>	Сброс «0» АК	10001	<i>RSTND</i>	Сброс в «0» РгД
00010	<i>SETNA</i>	Установка «1» АК	10010	<i>SETND</i>	Установка в «1» РгД
00100	<i>A2NA</i>	$\text{АК} + 2^n \rightarrow \text{АК}$	10100	<i>A2NDY</i>	$\text{РгД} + 2^n \rightarrow Y$
00101	<i>S2NA</i>	$\text{АК} - 2^n \rightarrow \text{АК}$	10101	<i>S2NDY</i>	$\text{РгД} - \overline{2^n} \rightarrow Y$
00110	<i>LD2NA</i>	$2^n \rightarrow \text{АК}$	10110	<i>LS2NY</i>	$2^n \rightarrow Y$
00111	<i>LDC2NA</i>	$\overline{2^n} \rightarrow \text{АК}$	10111	<i>LDC2NY</i>	$\overline{2^n} \rightarrow Y$

Таблица 1.16. Управление шиной *Y* и регистром состояния
при операциях с битами, тип инструкций *BOR1*

Мнемоника	Операция с битом	Шина <i>Y</i>	РгС, разряд <i>Z</i>
<i>SETNR</i>	Установка РЗУ	$Y_i \leftarrow \text{РЗУ}_i$ для $i \neq n$, $Y_n \leftarrow 1$	0
<i>RSTNR</i>	Сброс РЗУ	$Y_i \leftarrow \text{РЗУ}$ для $i \neq n$, $Y_n \leftarrow 0$	<i>U</i>
<i>TSTNR</i>	Проверка РЗУ	$Y_i \leftarrow 0$ для $i \neq n$, $Y_n \leftarrow \text{ИСТ}_n$	<i>U</i>

Примечание. Разряды *L*, *FL1*, *FL2*, *FL3* регистра состояния не изменяются, разряды *C* и *OVR* устанавливаются в нуль.

Т а б л и ц а 1.17. Управление шиной Y и регистром состояния с битами, тип инструкций *BOR2*

Мнемоника	Операция с битом n	Шина Y	PrC, разряды
			Z, C, OVR
<i>LD2NR</i>	$2^n \rightarrow P3Y$	$Y_i \leftarrow 0$ для $i \neq n, Y_n \leftarrow 1$	0
<i>LDC2NR</i>	$\overline{2^n} \rightarrow P3Y$	$Y_i \leftarrow 1$ для $i \neq n, Y_n \leftarrow 0$	0
<i>A2NR</i>	$P3Y + 2^n \rightarrow P3Y$	$Y_i \leftarrow P3Y + 2^n$	U
<i>S2NR</i>	$P3Y - 2^n \rightarrow P3Y$	$Y_i \leftarrow P3Y - 2^n$	U

Примечание. Разряды $L, FL1, FL2, FL3$ регистра состояния не изменяются.

Т а б л и ц а 1.18. Управление шиной Y и регистром состояния при операциях с битами, тип инструкций *BONR*

Мнемоника	Операция с битом n	Шина Y	PrC	
			Z	C, OVR
<i>TSTNA</i>	Проверка АК	$Y_i \leftarrow 0$ для $i \neq n, Y_n \leftarrow AK_n$	U	0
<i>RSTNA</i>	Сброс АК	$Y_i \leftarrow AK_i$ для $i \neq n, Y_n \leftarrow 0$	U	0
<i>SETNA</i>	Установка АК	$Y_i \leftarrow AK_i$ для $i \neq n, Y_n \leftarrow 1$	0	0
<i>A2NA</i>	$AK + 2^n \rightarrow AK$	$Y_i \leftarrow AK + 2^n$	U	U
<i>S2NA</i>	$AK - 2^n \rightarrow AK$	$Y_i \leftarrow AK - 2^n$	U	U
<i>LD2NA</i>	$2^n \rightarrow AK$	$Y_i \leftarrow 0$ для $i \neq n, Y_n \leftarrow 1$	0	0
<i>LDC2NA</i>	$\overline{2^n} \rightarrow AK$	$Y_i \leftarrow 1$ для $i \neq n, Y_n \leftarrow 0$	0	0
<i>TSTND</i>	Проверка PrД	$Y_i \leftarrow 0$ для $i \neq n, Y_n \leftarrow PrД_n$	U	0
<i>RSTND</i>	Сброс PrД	$Y_i \leftarrow PrД_i$ для $i \neq n, Y_n \leftarrow 0$	U	0
<i>SETND</i>	Установка PrД	$Y_i \leftarrow PrД_i$ для $i \neq n, Y_n \leftarrow 1$	0	0
<i>A2NDY</i>	$PrД + 2^n \rightarrow Y$	$Y_i \leftarrow PrД + 2^n$	U	U
<i>S2NDY</i>	$PrД - 2^n \rightarrow Y$	$Y_i \leftarrow PrД - 2^n$	U	U
<i>LD2NY</i>	$2^n \rightarrow Y$	$Y_i \leftarrow 0$ для $i \neq n, Y_n \leftarrow 1$	0	0
<i>LDC2NY</i>	$\overline{2^n} \rightarrow Y$	$Y_i \leftarrow 1$ для $i \neq n, Y_n \leftarrow 0$	0	0

Примечание. Разряды $L, FL1, FL2, FL3$ регистра состояния не изменяются.

Таблица 1.19. Инструкции циклического сдвига на n разрядов; тип инструкций *ROTR1*

Код	Мнемоника	ИСТ	ПРМ
1100	<i>RTRA</i>	РЗУ	АК
1110	<i>RTRY</i>	РЗУ	Шина Y
1111	<i>RTRR</i>	РЗУ	РЗУ

Таблица 1.20. Инструкции циклического сдвига на n разрядов; тип инструкций *ROTR2*

Код	Мнемоника	ИСТ	ПРМ
0000	<i>RTAR</i>	АК	РЗУ
0001	<i>RTDR</i>	РгД	РЗУ

Таблица 1.21. Инструкции циклического сдвига на n разрядов, тип *ROTNR*

Код	Мнемоника	ИСТ	ПРМ
11000	<i>RTDY</i>	РгД	Шина Y
11001	<i>RTDA</i>	РгД	АК
11100	<i>RTAY</i>	АК	Шина Y
11101	<i>RTAA</i>	АК	АК

Таблица 1.22. Управление шиной Y при сдвиге на n разрядов; типы инструкций *ROTR1*, *ROTR2*, *ROTNR*

Режим B/W	Шина Y	РгС, разряд N
0 — байт (B)	$Y_i \leftarrow \text{ИСТ}_{(i-n) \bmod 16}$	ИСТ_{15-n}
1 — слово (W)	$Y_i \leftarrow \text{ИСТ}_{(i-n) \bmod 8}$ для $i = 1-7$	ИСТ_{8-n}

Примечание. Разряды L , $FL1$, $FL2$, $FL3$, регистра состояния не изменяются, разряды C и OVR устанавливаются в нуль.

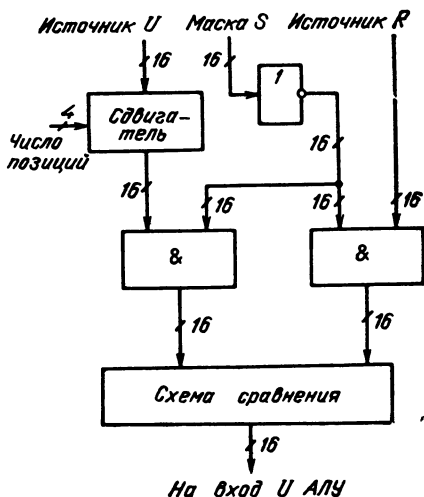
Подробно действие инструкций определено в табл. 1.19—1.22.

Пример 1. Режим W (полное слово), $n = 4$. Источник 0001 0011 0111 1111, приемник 0011 0111 1111 0001.

Пример 2. Режим B (байтовый), $n = 4$. Источник 0001 0011 0111 1111, приемник 0001 0011 1111 0111.

Инструкции циклического сдвига и сравнения определяются пятью полями: режим, источник сдвигаемого операнда, источник несдвигаемого операнда, маска и число позиций, на которое нужно сдвинуть операнд в сторону старших разрядов. Под управлением входов инструкции выполняется функция, которая поясняется рис. 1.12.

Рис. 1.12. Реализация сдвига и сравнения



Операнд U циклически сдвигается на n разрядов на сдвигателе. Маска инвертируется и используется в поразрядной конъюнкции с выходом сдвигателя и входом R . Таким образом, единичное значение разряда маски исключает сравнение соответствующих разрядов операндов, а нулевое — разрешает. Биты N и Z регистра состояния модифицируются, а биты OVR и C сбрасываются в нуль.

Пример. $n = 4$, режим полного слова (W). Операнд $U = 0011\ 0001\ 0101\ 0110$. Сдвинутый операнд $U = 0001\ 0101\ 0110\ 0011$. Операнд $R = 0001\ 0101\ 1111\ 0000$. Маска $S = 0000\ 0000\ 1111\ 1111$. Результат сравнения $Z = 1$. Формат инструкций:

	15	14—13	12—9	8—5	4—0
ROTC	B/W	01	n	Сдвигаемый ИСТ, несдвигаемый ИСТ, ПРМ, маска	Адрес РЗУ

Действие инструкций определено в табл. 1.23 1.24.

Инструкции циклического сдвига и слияния. Эти инструкции определяются пятью полями: режим, источник сдвигаемого операнда, источник несдвигаемого операнда (он же приемник результата), маска и число разрядов n , на которое следует сдвинуть операнд. Под управлением входов инструкции выполняется функция, которая поясняется рис. 1.13. Операнд U циклически сдвигается на n позиций в сторону старших разрядов, затем под действием маски формируется результат: в качестве i -го его разряда берется i -й разряд сдвинутого операнда R , если i -й разряд маски равен нулю, или i -й разряд операнда U , если

Таблица 1.23. Инструкции циклического сдвига и сравнения; тип инструкций ROTC

Код	Мнемоника	Сдвигаемый ИСТ	ИСТ. ПРМ	Маска
0010	<i>CDAI</i>	РгД	АК	I
0011	<i>CDRI</i>	РгД	РЗУ	I
0100	<i>CDRA</i>	РгД	РЗУ	АК
0101	<i>CRAI</i>	РЗУ	АК	I

Таблица 1.24. Управление шиной Y и регистром состояния при циклическом сдвиге и сравнении; тип инструкций ROTC

Режим B/W	Шина Y
1 — слово (W)	$Y_i \leftarrow (\text{сдвигаемый ИСТ})_i \& (\text{маска})_i \oplus (\text{несдвигаемый ИСТ})_{(i-n) \bmod 16} \& (\text{маска})_i$
0 — байт (B)	$Y_i \leftarrow (\text{сдвигаемый ИСТ})_i \& (\text{маска})_i \oplus (\text{несдвигаемый ИСТ})_{(i-n) \bmod 8} \& (\text{маска})_i$

Примечание. Разряды L , $FL1$, $FL3$, $FL3$ регистра состояния не изменяются, разряды C и OVR устанавливаются в нуль.

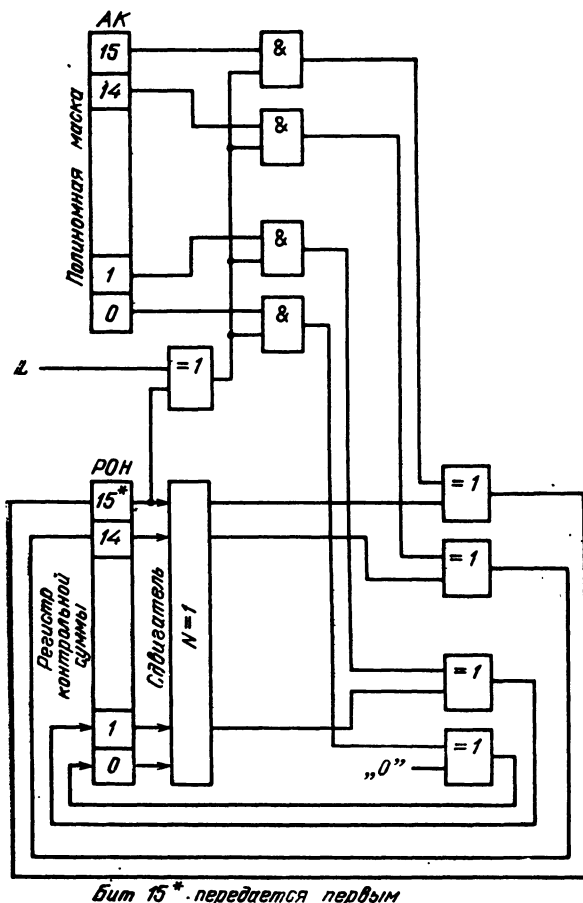


Рис. 1.13. Реализация прямой свертки

i -й разряд маски равен единице. Результат заносится по адресу несдвигаемого операнда. Биты N и Z регистра состояния модифицируются, а биты OVR и C устанавливаются в нуль.

Пример. $n = 4$, режим полного слова (W). Операнд $U = 0011\ 0001\ 0101\ 0110$. Сдвинутый операнд $U = 0001\ 0101\ 0110\ 0011$. Операнд $R = 1010\ 1010\ 1010\ 1010$. Маска $S = 0000\ 1111\ 0000\ 1111$. Результат (R) = $1010\ 0101\ 1010\ 0011$. Формат инструкций:

	15	14—13	12—9	8—5	4—0
<i>ROTM</i>	<i>B/W</i>	01	n	Сдвигаемый ИСТ, несдвигаемый ИСТ, ПРМ, маска	Адрес РЗУ

Таблица 1.25. Инструкции циклического сдвига и слияния; тип инструкций *ROTM*

Код	Мнемоника	Сдвигаемый ИСТ	ИСТ/ПРМ	Маска
0111	<i>MDAI</i>	РгД	АК	<i>I</i>
1000	<i>MDAR</i>	РгД	АК	РЗУ
1001	<i>MDRI</i>	РгД	РЗУ	<i>I</i>
1010	<i>MDRA</i>	РгД	РЗУ	АК
1100	<i>MARI</i>	АК	РЗУ	<i>I</i>
1110	<i>MRAI</i>	РЗУ	АК	<i>I</i>

Таблица 1.26. Управление шиной *Y* и регистром состояния при циклическом сдвиге и слиянии; тип инструкций *ROTM*

Режим	Шина <i>Y</i>
1 — слово (<i>W</i>)	$Y_i \leftarrow (\text{несдвигаемый ИСТ})_i \& (\text{маска})_i \vee (\text{сдвигаемый ИСТ})_{(i-n) \bmod 16} \& (\text{маска})_i$
0 — байт (<i>B</i>)	$Y_i \leftarrow (\text{несдвигаемый ИСТ})_i \& (\text{маска})_i \vee (\text{сдвигаемый ИСТ})_{(i-n) \bmod 8} \& (\text{маска})_i$

Примечание. Разряды *L*, *FL1*, *FL2*, *FL3* регистра состояния не изменяются, размеры *C* и *OVR* устанавливаются в нуль.

Описание инструкции приведено в табл. 1.25 и 1.26. Инструкция может быть эффективно использована для преобразования одного кода в другой.

Инструкции шифрации. Эти инструкции задаются пятью полями: режим, источник операнда (*R*), источник маски (*S*) и приемник. Функция, выполняемая ПШ, уже обсуждалась выше: производится поразрядная конъюнкция операнда *R* и инвертированного значения маски *S*. Нулевое значение разряда маски разрешает участие соответствующего разряда операнда *R* в шифрации приоритета, а единичное — запрещает, т. е. устанавливает разряд операнда *R* в нулевое состояние. Входной 16-разрядный код преобразуется ПШ в выходной 5-разрядный двоичный код, указывающий номер самого старшего разряда, содержащего единицу после преобразования с учетом маски. Биты *N* и *Z* РгС модифицируются, а биты *OVR* и *C* устанавливаются в нулевое состояние. Единственным ограничением при выполнении данной инструкции является необходимость использования различных источников для операнда и маски. Форматы инструкций:

	15	14—13	12—9	8—5	4—0
<i>PRT1</i>	<i>B/W</i>	10	ПРМ	ИСТ (<i>R</i>)	Адрес РЗУ/маска (<i>S</i>)
<i>PRT2</i>	<i>B/W</i>	10	Маска <i>S</i>	ПРМ	Адрес РЗУ/ИСТ

PRT3

<i>B/W</i>	10	Маска <i>S</i>	ИСТ <i>R</i>	Адрес РЗУ /ПРМ
------------	----	----------------	--------------	----------------

PRTNR

<i>B/W</i>	11	Маска <i>S</i>	ИСТ <i>R</i>	ПРМ
------------	----	----------------	--------------	-----

Действие инструкций шифрации описано в табл. 1.27—1.31. Инструкции шифрации удобны при выполнении нормализации. Нормализация состоит в проверке старшего разряда и сдвиге в сторону старших разрядов, если в этом разряде «0», до тех пор, пока в старшем разряде не окажется «1». При многократных операциях на это расходуется значительное время. Инструкция шифрации же за один такт определяет позицию старшей единицы. Во втором такте можно произвести сдвиг слова на нужное число разрядов с помощью той же БИС К1804ВМ1.

Таблица 1.27. Инструкции шифрации; тип инструкций *PRT1*

Приемник		Источник <i>R</i>	
Код	Мнемоника	Код	Мнемоника
1000	<i>PRTA</i> АК	0111	<i>PRTIA</i> АК
1010	<i>PRTY</i> Шина <i>Y</i>	1001	<i>PRTD</i> РгД
1011	<i>PRTI</i> РЗУ		

Таблица 1.28. Инструкции шифрации; тип инструкций *PRT2*

Маска <i>S</i>		Приемник	
Код	Мнемоника	Код	Мнемоника
1000	<i>PRA</i> АК	0000	<i>PR2A</i> АК
1010	<i>PRZ</i> 0	0010	<i>PR2Y</i> Шина <i>Y</i>
1011	<i>PRI</i> 1		

Таблица 1.29. Инструкции шифрации; тип инструкций *PRT3*

Маска <i>S</i>		Источник <i>R</i>	
Код	Мнемоника	Код	Мнемоника
1000	<i>PRA</i> АК	0011	<i>PR3R</i> РЗУ
1010	<i>PRZ</i> 0	0100	<i>PR3A</i> АК
1011	<i>PRI</i> 1	0110	<i>PR3D</i> РгД

Таблица 1.30. Инструкции шифрации; тип инструкций *PRTNR*

Маска <i>S</i>		Источник <i>R</i>		Приемник	
Код	Мнемоника	Код	Мнемоника	Код	Мнемоника
1000	<i>PRA</i> АК	0100	<i>PRTA</i> АК	00000	<i>NRV</i> Шина <i>Y</i>
1010	<i>PRZ</i> 0	0110	<i>PRTD</i> РгД	00001	<i>NRA</i> АК
1011	<i>PRI</i> 1				

Таблица 1.31. Управление шиной Y и регистром состояния в инструкциях шифрации; типы инструкций PRT1, PRT2, PRT3, PRTNR

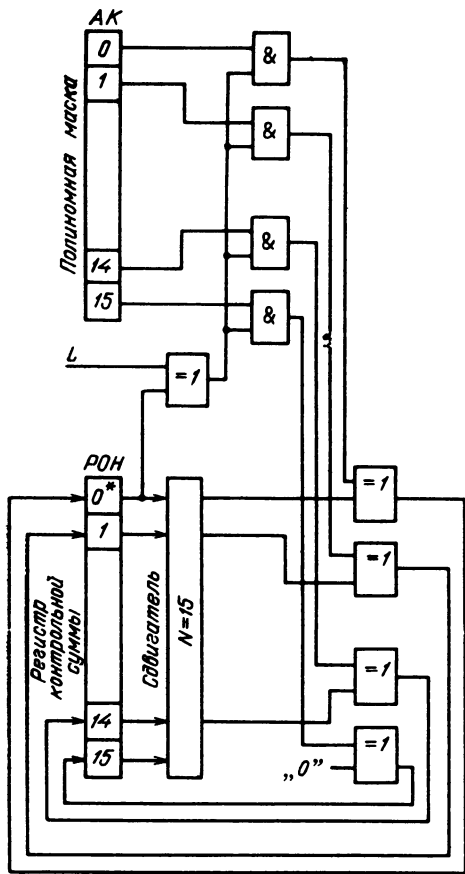
Режим В/В	Шина Y
1 — слово (W)	$Y_i \leftarrow \text{код}(\text{ИСТ}_n \& \overline{\text{маска}_n}); Y_m \leftarrow 0;$ $i = 0-4; n = 0-15; m = 5-15$
0 — байт (B)	$Y_i \leftarrow \text{код}(\text{ИСТ}_n \& \overline{\text{маска}_n}); Y_m \leftarrow 0;$ $i = 0-2, n = 0-7, m = 3-15$

Примечание. Разряды *L*, *FL1*, *FL2*, *FL3* регистра состояния не изменяются, разряды *C* и *OVR* устанавливаются в нуль.

Инструкции генерации циклического избыточного кода (CRC). Эти инструкции определяются одним полем: адресом регистра РЗУ,

который будет использован для хранения контрольной суммы. Инструкции обеспечивают генерацию контрольных разрядов в циклическом избыточном коде. Одна из инструкций называется «прямой», другая — «обратной». Разница между ними заключается в том, с какого разряда данных (старшего или младшего) начинается обработка. Выполнение их поясняется рис. 1.13 и 1.14. Две инструкции необходимы потому, что существующие стандарты применения циклических избыточных кодов не определяют, какой разряд данных (старший или младший) должен передаваться первым.

Бит L используется как вход последовательного кода. В соответствии с полиномом, задаваемым с помощью маски, последовательный вход комбинируется со старшим разрядом регистра контроля-



Бит 0^* передается первым

Рис. 1.14. Реализация обратной свертки

ной суммы. После того как последний входной бит обработан, регистр содержит контрольные разряды. Биты L , N и Z РгС модифицируются, а биты OVR и C устанавливаются в нуль.

Итак, перед началом операции необходимо загрузить в АК полиномную маску. Операция начинается с загрузки первого бита данных в разряд связи L РгС. Во время следующего такта этот бит появляется на выводе L РгС. Результат операции ИСКЛЮЧАЮЩЕЕ ИЛИ этого бита и старшего бита РОИ передается на вентили И, куда поступает и соответствующий бит маски S . Результат поразрядной операции ИСКЛЮЧАЮЩЕЕ ИЛИ сигналов с выходов схем И и сдвинутого на один разряд операнда U записывается в РЗУ. Очередной такт начинается с загрузки следующего разряда данных в разряд связи L РгС. Процесс продолжается до тех пор, пока все разряды данных не пройдут такой цикл. Форматы инструкций:

	15	14—13	12—9	8—5	4—0
<i>CRCF</i>	1	10	0110	0011	Адрес РЗУ
<i>CRCR</i>	1	10	0110	1001	Адрес РЗУ

Описание инструкций приведено в табл. 1.32. Инструкции обеспечивают вычисление циклического избыточного кода по любому полиному с разрядностью не более 16.

Инструкции установки и сохранения состояния. Формат 8-разрядного РгС уже был приведен выше. Теперь рассмотрим инструкции, выполняемые над битами РгС. Инструкции установки в «1» определяют,

Таблица 1.32. Управление шиной Y и регистром состояния при генерации циклических избыточных кодов

Тип	Шина Y	РгС, разряд L
<i>CRCF</i>	$Y_i \leftarrow [(L \oplus PЗУ_{15}) \& AK_i] \oplus PЗУ_{i-1}, i = 15 \dots 1;$ $Y_0 \leftarrow [(L \oplus PЗУ_{15}) \& AK_0] \oplus 0$	$PЗУ_{15}$
<i>CRCR</i>	$Y_i \leftarrow [(L \oplus PЗУ_0) \& AK_i] \oplus PЗУ_{i+1}, i = 14 \dots 0$ $Y_{15} \leftarrow [(L \oplus PЗУ_0) \& AK_{15}] \oplus 0$	$PЗУ_0$

Примечание. Разряды L , $FL1$, $FL2$, $FL3$ регистра состояния не изменяются, разряды C и OVR устанавливаются в нуль.

какому биту или группе битов PгC следует присвоить единичное значение. Формат инструкций:

	15	14—13	12—9	8—5	4—0
SETST	0	11	1011	1010	ОП

Инструкции сброса в «0» имеют формат:

RSTST	0	11	1010	1010	КОП
-------	---	----	------	------	-----

Инструкции хранения содержимого PгC имеют два формата:

SVSTR	B W	10	0111	1010	Адрес РЗУ/ПРМ
-------	-----	----	------	------	------------------

SVSTNR	B W	11	0111	1010	ПРМ
--------	-----	----	------	------	-----

и могут использоваться для сохранения информации о состоянии процессора с последующим возвратом ее в PгC, что удобно при обработке прерываний. Действие инструкций определено в табл. 1.33—1.35.

Таблица 1.33. Инструкции установки и сохранения состояния; тип инструкций SETST

Код	Мнемоника	Операция
00011	SONCZ	Установка в «1» OVR, N, C, Z
00101	SL	Установка в «1» L
00110	SF1	Установка в «1» FL1
01001	SF2	Установка в «1» FL2
01010	SF3	Установка в «1» FL3

Таблица 1.34. Инструкции установки и сохранения состояния; тип инструкций RSTST

Код	Мнемоника	Операция
00011	RONCZ	Сброс в «0» OVR, N, C, Z
00101	RL	Сброс в «0» L
00110	RF1	Сброс в «0» FL1
01001	RF2	Сброс в «0» FL2
01010	RF3	Сброс в «0» FL3

Таблица 1.35. Управление шиной Y при выполнении инструкций установки, сброса и сохранения состояния

Тип	Мнемоника	Операция	Шина Y	PгC							
				Z	C	OVR	N	L	FL1	FL2	FL3
SETST	SONCZ	Установка в «1» OVR, N, C, Z	$Y_i \leftarrow 1, i = 0-15$	1	1	1	1	NU	NU	NU	NU
	SL	Установка в «1» L		NU	NU	NU	NU	1	NU	NU	
	SF1	Установка в «1» FL1		NU	NU	NU	NU	1	NU	NU	
	SF2	Установка в «1» FL2		NU	NU	NU	NU	NU	1	NU	
	SF3	Установка в «1» FL3		NU	NU	NU	NU	NU	NU	1	
RSTST	RONCZ	Сброс в «0» OVR, N, C, Z	$Y_i \leftarrow 0, i = 0-15$	0	0	0	0	NU	NU	NU	
	RL	Сброс в «0» L		NU	NU	NU	NU	0	NU	NU	
	RF1	Сброс в «0» FL1		NU	NU	NU	NU	0	NU	NU	
	RF2	Сброс в «0» FL2		NU	NU	NU	NU	NU	0	NU	
	RF3	Сброс в «0» FL3		NU	NU	NU	NU	NU	NU	0	
SVSTR SVSTNR		Сохранение PгC (см. примечание)	$Y_i \leftarrow PгC_i, i = 0-7$ $Y_i \leftarrow 0, i = 8-15$	Не изменяются							

Примечание. В байтовом режиме с шины Y в PЗУ или АК загружается только младший байт, в режиме полного слова – все 16 разрядов. Обозначение NU означает, что соответствующий разряд остается без изменений.

Т а б л и ц а 1.36. Инструкции проверки состояния; тип инструкций *TSTST*

Код	Мнемоника	Операция	Код	Мнемоника	Операция
00000	<i>TNOZ</i>	Проверка $(N \oplus OVR) \vee Z$	01100	<i>TZC</i>	Проверка $Z \vee \bar{C}$
00010	<i>TNO</i>	Проверка $N \oplus OVR$	01110	<i>TN</i>	Проверка N
00100	<i>TZ</i>	Проверка Z	10000	<i>TL</i>	Проверка L
00110	<i>TOVR</i>	Проверка OVR	10010	<i>TF1</i>	Проверка $FL1$
01000	<i>TLOW</i>	Проверка 0 (константа)	10100	<i>TF2</i>	Проверка $FL2$
01010	<i>TC</i>	Проверка C	10110	<i>TF3</i>	Проверка $FL3$

Т а б л и ц а 1.37. Управление проверкой *RrC*

<i>T4</i> <i>I4</i>	<i>T3</i> <i>I3</i>	<i>T2</i> <i>I2</i>	<i>T1</i> <i>I1</i>	<i>CT</i>	<i>T4</i> <i>I4</i>	<i>T3</i> <i>I3</i>	<i>T1</i> <i>I1</i>	<i>T1</i> <i>I1</i>	<i>CT</i>
0	0	0	0	$(N \oplus OVR) \vee Z$	0	1	1	0	$Z \vee \bar{C}$
0	0	0	1	$N \oplus OVR$	0	1	1	1	N
0	0	1	0	Z	1	0	0	0	L
0	0	1	1	OVR	1	0	0	1	$FL1$
0	1	0	0	0	1	0	1	0	$FL2$
0	1	0	1	C	1	0	1	1	$FL3$

В поле *PRM* инструкций типа *SVSTNR* код 00000 и мнемоника *NRV* соответствуют шине *Y*, а код 00001 и мнемоника *NRA* — аккумулятору.

Содержимое *RrC* помещается всегда в младший байт регистра *R3Y* или аккумулятора. В зависимости от режима (байт/полное слово) старший байт либо остается неизменным, либо загружается нулевыми значениями. Инструкции загрузки *RrC* отдельно не рассматриваются, поскольку они являются подмножеством инструкций с одним операндом.

Инструкции проверки состояния. Эти инструкции задают одно из 12 тестовых условий, которое должно быть передано на выход *CT* (табл. 1.36). В качестве *CT* может быть выбран один из битов *RrC* (Z , C , N , OVR , L , $FL1$, $FL2$, $FL3$) или значение одной из логических функций: $N \oplus OVR$, $(N \oplus OVR) \vee Z$, $Z \vee \bar{C}$ либо «0». Эти функции используются при проверке результатов арифметических операций в дополнительном коде и без знака. При действии инструкций проверки состояния значение шины *Y* не определено, а содержимое *RrC* не меняется. Содержимое *RrC* может быть также проверено через двунаправленную шину *T*. Как показано в табл. 1.37, код для проверки *RrC* через шину *T* идентичен коду, подаваемому на линии *I0—I4* инструкции. Линии *I0—I4* имеют приоритет над шиной *T* по тестированию *RrC* с передачей результата на выход *CT*. Формат инструкций:

	15	14—13	12—9	8—5	4—0
<i>TSTST</i>	0	11	1001	1010	КОП

Инструкция «Нет операции». Данная инструкция имеет формат:

15	14—13	12—9	8—5	4—0
0	11	1000	1010	0000

Инструкция не изменяет содержимое ни одного из внутренних регистров БИС К1804ВМ1. При ее выполнении значение на шине *Y* не определено.

Мнемоники сопоставлены не только типам инструкций и кодам операций, но также каждому сочетанию источников и приемников для всех типов инструкций. Поэтому инструкции можно записывать с помощью мнемоник в соответствии с форматом. Например, для инструкции типа *SOR* в записи кодируется:

<i>W</i>	<i>INC</i>	<i>SORA</i>	<i>R24</i>
(Режим полного слова)	(Код операции инкрементирования)	(Операнд извлекается из РЗУ, а результат помещается в АК)	(Источником является регистр номер 24 РЗУ)

Это особенно удобно при использовании микроассемблера *AMDASM*.

Инструкции с непосредственным операндом (они же двухтактные) в мнемонике способа адресации содержат литеру «*I*». Например, *TORIY* означает выборку первого операнда из РЗУ по указанному адресу, второго операнда — со входов *I0—I15* и выдачу результата на шину *Y*. Все это произойдет на следующем такте после приема инструкции, содержащей в *TORIY*, схемой К1804ВМ1. Таким образом, поле *I0—I15* на двух соседних тактах должно иметь значения:

Микрокоманда на такте <i>i</i>	<i>B/W</i>	00	<i>TORIY</i> (1010)	КОП	Адрес РЗУ
--------------------------------	------------	----	------------------------	-----	-----------

Микрокоманда на такте <i>i+1</i>	Значение непосредственного операнда (<i>I</i>)				
----------------------------------	--	--	--	--	--

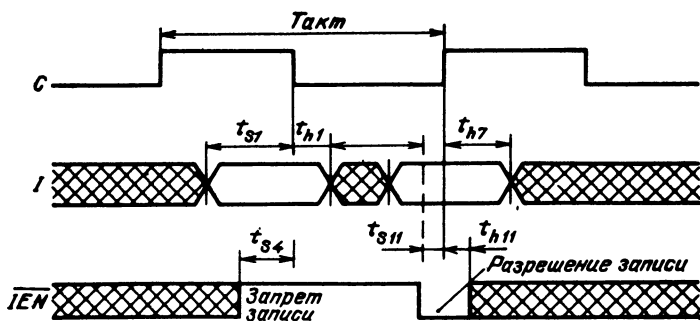


Рис. 1.15. Временные диаграммы для одноадресных операций

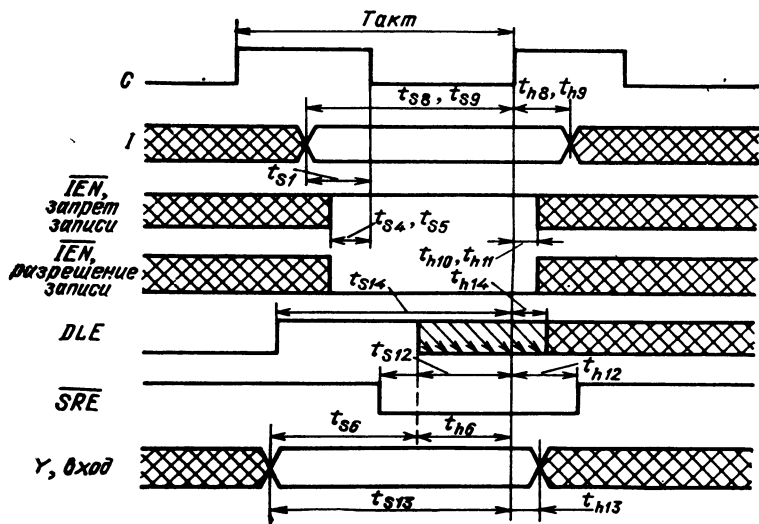


Рис. 1.16. Временные диаграммы для двухадресных операций

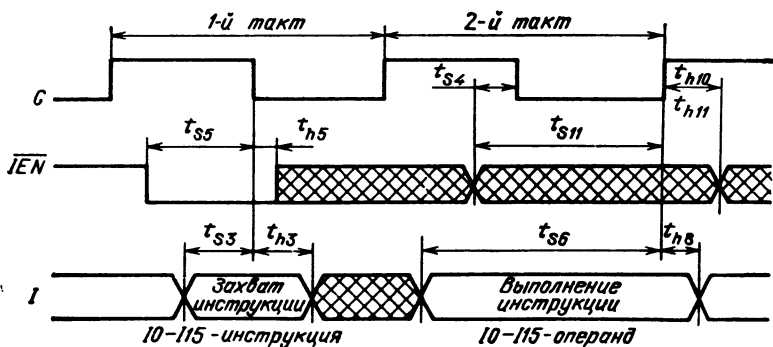


Рис. 1.17. Временные диаграммы для операций с непосредственным операндом

Двухтактные инструкции выгодны тем, что для представления констант (непосредственных операндов) не требуется выделять в микрокоманде дополнительного 16-разрядного поля, что существенно сказывается на суммарной разрядности микрокоманды.

Временные диаграммы сигналов на входах микросхемы К1804ВМ1, которые должны соблюдаться при выполнении одноадресных, двухадресных операций и операций с непосредственным операндом (константой в поле I микрокоманды), показаны соответственно на рис. 1.15—1.17. Определения используемых обозначений для времени предустановки t_s и удержания t_h , а также конкретные значения этих параметров включаются в техническое описание и руководящие технические материалы по применению БИС К1804ВМ1.

1.2. СХЕМА АДРЕСНОЙ ОБРАБОТКИ

Четырехразрядная секция адресной обработки (CAO) К1804ВУ5 может использоваться для формирования адресов как на программном уровне (адреса команд и операндов в оперативной памяти), так и на микропрограммном уровне (адреса микропрограммной памяти). Основной особенностью CAO является наличие 17-уровневого стека и сумматора, обеспечивающего двенадцать различных модификаций относительной адресации и имеющего возможность организации ускоренного переноса при наращивании CAO до произвольной разрядности, кратной четырем. Выполняет CAO тридцать две инструкции формирования адреса, шестнадцать из которых являются условными на состояние внешнего входа условия. Все внутренние регистры построены на триггерах с занесением информации по фронту тактового сигнала. Выходы адреса тристабильные. Микросхема К1804ВУ5 изготавливается по технологии маломощных ТТЛ-схем и диодами Шотки и выпускается в 28-выводном корпусе. Источник питания +5 В. Цоколевка выводов корпуса микросхемы показана на рис. 1.18.

Структурная схема CAO приведена на рис. 1.19. В ней можно выделить пять основных блоков: сумматор, вспомогательный регистр, стек, программный счетчик, дешифратор инструкций.

Полный сумматор формирует сумму операндов, поступающих на его входы A и B с учетом значения сигнала на входе переноса $C0$. Мультиплексор $MUXA$, стоящий на входе A сумматора, позволяет выбирать в качестве операнда A содержимое вспомогательного регистра R , информацию с шины адреса D или «0». Мультиплексор $MUXB$ позволяет выбирать в качестве операнда B содержимое вспомогательного регистра R , содержимое вершины стека S , содержимое программного счетчика PC или «0». При наращивании CAO имеется возможность организации как последовательного, так и ускоренного переноса. Последовательный перенос организуется путем соединения входа переноса в сумматор ($C0$) каждой старшей CAO с выходом переноса из сумматора $C4$ соседней младшей CAO. Для организации ускоренного переноса используются выход распространения переноса из сумматора \bar{P} и вы-

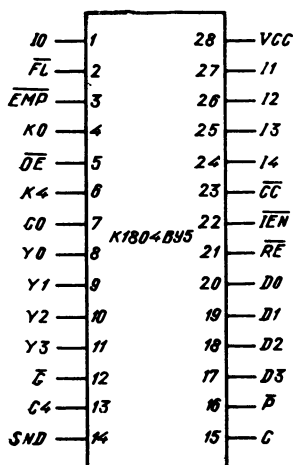


Рис. 1.18. Цоколевка выводов микросхемы K1804BV5

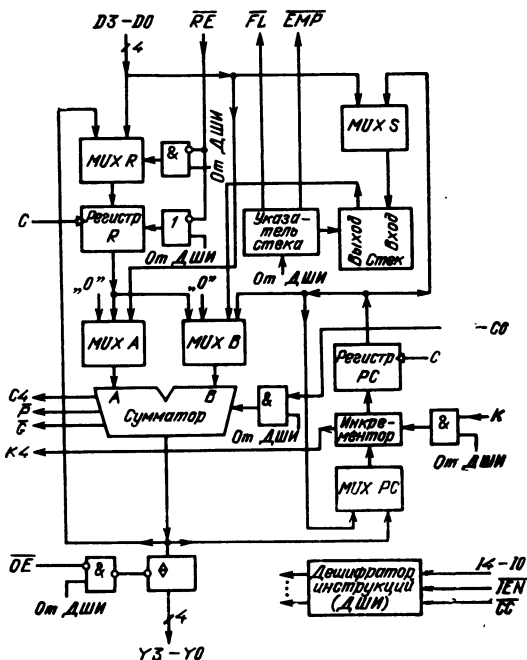


Рис. 1.19. Структурная схема БИС K1804BV5

ход генерации переноса из сумматора \bar{G} . Для организации ускоренного переноса может служить, например, схема ускоренного переноса K1804BP1. При выполнении инструкций формирования адреса, не требующих суммирования, вход переноса в сумматор $C0$ устанавливается в нуль внутренними сигналами. Информация с выхода сумматора поступает на тристабильную выходную шину адреса $Y3-Y0$, управляемую сигналом на входе разрешения выходов адреса \overline{OE} . При $\overline{OE} = 1$ выходная шина $Y3-Y0$ находится в состоянии высокого сопротивления. Кроме того, шина переходит в это состояние при выполнении инструкции «Приостановка», если значение на входе условия $\overline{CC} = 0$.

Программный счетчик состоит из регистра с инкрементором на входе и мультиплексора MUX_{PC} . Регистр счетчика команд — это 4-разрядный регистр, построенный на D -триггерах, срабатывающих по фронту тактового сигнала C (переход C из «0» в «1»). Информация в него загружается с выхода инкременторов в конце выполнения каждой инструкции формирования адреса. При нулевом сигнале на входе переноса в инкрементор ($K0$) информация через инкрементор передается без изменения. При единичном — происходит увеличение на 1. При наращивании САО необходимо соединить выход переноса из инкрементора $K4$ каждой младшей САО со входом переноса в инкрементор $K0$ следующей старшей САО. Вход переноса в инкрементор $K0$ устанавли-

вается в нуль внутренними сигналами при выполнении инструкций «Безусловное хранение», «Условное хранение» и «Приостановка», если сигнал на входе условия $\overline{CC} = 0$. Мультиплексор $MUXPC$ выбирает в качестве источника информации для инкремента выходы сумматора во время выполнения инструкций перехода, перехода к подпрограмме, возврата (инструкции 16—29) при $\overline{CC} = 0$, а также инструкции сброса. Во всех остальных случаях на вход инкремента выбирается содержимое программного счетчика PC .

Стек состоит из указателя стека, памяти стека объемом семнадцать 4-разрядных слов и мультиплексора. Указатель стека всегда адресует последнее слово, записанное в стек, — вершину стека S . Изменение указателя стека происходит по фронту тактового сигнала C (переход C из «0» в «1») в конце выполнения соответствующей инструкции формирования адреса. При занесении в стек ($PUSH$) происходит увеличение на 1 указателя стека, затем запись в стек. После семнадцатого занесения в стек на выходе \overline{FL} появляется нулевой логический уровень, сигнализирующий о том, что стек полон. Запись в полный стек аппаратно блокируется. При выталкивании из стека (POP) происходит уменьшение указателя стека на 1. После выталкивания из стека последнего слова или в конце выполнения инструкции сброса на выходе \overline{EMP} появляется нулевой логический уровень, сигнализирующий о том, что стек пуст. При выталкивании из пустого стека значение указателя стека не меняется. Мультиплексор передает на вход памяти стека информацию с шины адреса $D3-D0$ или содержимое PC .

Четырехразрядный вспомогательный регистр R построен на D -триггерах, запись информации в которые происходит по фронту тактового сигнала C при выполнении инструкций, предписывающих запись во вспомогательный регистр, а также если на вход разрешения \overline{RE} подать нулевой сигнал. Если на входе разрешения инструкции \overline{IEN} находится нулевой сигнал, то при выполнении инструкций с кодами 8 и 9 регистр R загружается с выходов сумматора, а при выполнении инструкции с кодом 10 — с шины адреса. Если же $\overline{IEN} = 1$, то регистр R при $\overline{RE} = 0$ загружается информацией с входной шины адреса независимо от выполняемой инструкции.

Дешифратор инструкций (ДШИ) вырабатывает необходимые внутренние управляющие сигналы под действием входных сигналов инструкции 14—10, сигнала условия (\overline{CC}) и сигнала разрешения инструкции (\overline{IEN}). Для условных инструкций при $\overline{CC} = 1$ выполняется переход по PC , а при $\overline{CC} = 0$ — по условному адресу. Для безусловных инструкций значение сигнала на входе условия \overline{CC} безразлично.

Если $\overline{IEN} = 1$, то PC и указатель стека переводятся в режим хранения, запись в память стека блокируется, а регистр R управляется сигналом на входе разрешения \overline{RE} независимо от инструкции 14—10. При $\overline{IEN} = 0$ все внутренние регистры находятся под управлением

Т а б л и ц а 1.38. Инструкции схемы адресной обработки К1804ВУ5

Мнемоника	Код	14—10	\overline{CC}	\overline{TEN}	Операция	Y'3—Y'0	Следующее состояние			
							PC	$R(\overline{RE} \neq 0)$	$R(\overline{RE} = 1)$	Стек
—	—	XXXXX	X	1	—	Прим. 1	Не меняется	D	Не меняется	Не меняется
PRST	0	0 0 0 0 0	X	0	Сброс	0	0+K0	D	То же	Сброс указателя
FPC	1	0 0 0 0 1	X	0	Выборка PC	PC	PC+K0	D	«	Не меняется
FR	2	0 0 0 1 0	X	0	Выборка R	R	PC+K0	D	«	«
FD	3	0 0 0 1 1	X	0	Выборка D	D	PC+K0	D	«	«
FDD	4	0 0 1 0 0	X	0	Выборка R+D	R+D+C0	PC+K0	D	«	«
FPD	5	0 0 1 0 1	X	0	Выборка PC+D	PC+D+C0	PC+K0	D	«	«
FPR	6	0 0 1 1 0	X	0	Выборка PC+R	PC+R+C0	PC+K0	D	«	«
FSD	7	0 0 1 1 1	X	0	Выборка S+D	—	PC+K0	D	«	«
FPLR	8	0 1 0 0 0	X	0	Выборка и запись PC в R	PC	PC+K0	PC	PC	«
FRDR	9	0 1 0 0 1	X	0	Выборка и запись R+D в R	R+D+C0	PC+K0	R+D+C0	R+C+D0	«
PLDR	10	0 1 0 1 0	X	0	Загрузка R	PC	PC+K0	D	—	«
PSHR	11	0 1 0 1 1	X	0	Заталкивание PC	«	PC+K0	D	Не меняется	Заталкивание
PSHD	12	0 1 1 0 0	X	0	Заталкивание D	«	PC+K0	D	То же	Заталкивание
POPS	13	0 1 1 0 1	X	0	Выталкивание S	S	PC+K0	D	Не меняется	Выталкивание
POPP	14	0 1 1 1 0	X	0	Выталкивание PC	PC+K0	PC+K0	D	«	То же
PHLD	15	0 1 1 1 1	X	0	Хранение	PC	Не меняется	D	«	Не меняется
—	16—31	1 XXXX	1	0	Выполняется инструкция с кодом 1	«	PC+K0	D	«	«
JMPR	16	1 0 0 0 0	0	0	Переход к R	R	R+K0	D	«	«

<i>JMPD</i>	17	1	0	0	1	0	0	Переход к <i>D</i>	<i>D</i>	<i>D + K0</i>	<i>D</i>	«
<i>JMPZ</i>	18	1	0	0	1	0	0	Переход к «0»	0	<i>0 + K0</i>	<i>D</i>	«
<i>JMPRD</i>	19	1	0	0	1	1	0	Переход к <i>R + D</i>	<i>R + D + C0</i>	<i>R + D + C0 + K0</i>	<i>D</i>	«
<i>JMPPD</i>	20	1	0	1	0	0	0	Переход к <i>PC + D</i>	<i>PC + D + C0</i>	<i>PC + D + C0 + K0</i>	<i>D</i>	«
<i>JMPPR</i>	21	1	0	1	0	1	0	Переход к <i>PC + R</i>	<i>PC + R + C0</i>	<i>PC + R + C0 + K0</i>	<i>D</i>	«
<i>JSBR</i>	22	1	0	1	1	0	0	Переход к п/п по <i>R</i>	<i>R</i>	<i>R + K0</i>	<i>D</i>	«
<i>JSBD</i>	23	1	0	1	1	1	0	Переход к п/п по <i>D</i>	<i>D</i>	<i>D + K0</i>	<i>D</i>	«
<i>JSBZ</i>	24	1	1	0	0	0	0	Переход к п/п по 0	0	<i>0 + K0</i>	<i>D</i>	«
<i>JSBRD</i>	25	1	1	0	0	1	0	Переход к п/п по <i>R + D</i>	<i>P + C0</i>	<i>R + D + C0 + K0</i>	<i>D</i>	«
<i>JSBPD</i>	26	1	1	0	1	0	0	Переход к п/п по <i>PC + D</i>	<i>PC + C0</i>	<i>PC + D + C0 + K0</i>	<i>D</i>	«
<i>JSBPR</i>	27	1	1	0	1	1	0	Переход к п/п по <i>PC + R</i>	<i>PC + P + C0</i>	<i>PC + R + C0 + K0</i>	<i>D</i>	«
<i>RTS</i>	28	1	1	1	0	0	0	Возврат по <i>S</i>	<i>S</i>	<i>S + K0</i>	«	«
<i>RTST</i>	29	1	1	1	0	1	0	Возврат по <i>S + D</i>	<i>S + D + C0</i>	<i>S + C0 + K0</i>	«	«
<i>CHLD</i>	30	1	1	1	1	0	0	Условное хранение	<i>PC</i>	Не меняется	«	«
<i>PSUS</i>	31	1	1	1	1	1	0	Приостановка	<i>Z</i>	То же	«	«

Затягивание *PC*
То же

Выталкивание
То же

Не меняется
То же

Примечание. Если $\overline{IEN}=1$, на шину $Y3-Y0$ выводится информация, определяемая сигналами $I4-I0$ и \overline{CS} при $\overline{IEN}=0$; X — значение безразлично.

инструкции 14—10. Необходимо отметить, что значение сигнала \overline{TEN} не влияет на значения сигналов на выходах $Y3—Y0$.

Набор инструкций формирования адреса САО может быть разделен на пять типов инструкций (табл. 1.38): безусловные переходы, условные переходы, условные переходы к подпрограмме (п/п), условные возвраты из подпрограммы, прочие инструкции. В табл. 1.38 и далее символом X отмечено безразличное состояние, Z — состояние высокого сопротивления. САО выполняет девять инструкций безусловного перехода (коды 1—9). Для этих инструкций содержимое PC увеличивается, если $K0 = 1$. Для инструкций с кодами 1—7 регистр R загружается информацией с шины адреса $D3—D0$ при нулевом сигнале на входе разрешения \overline{RE} регистра R . Для инструкций с кодами 8 и 9 вспомогательный регистр загружается содержимым PC и результатом операции $R + D + C0$ соответственно, независимо от значения сигнала на входе \overline{RE} . Значение указателя стека и содержимое стека во время выполнения инструкций безусловного перехода не изменяются.

Имеется шесть инструкций условного перехода (коды 16—21). Во время выполнения этих инструкций регистр R управляется сигналом на входе \overline{RE} , а указатель стека и память стека не изменяются. Вышеуказанные операции выполняются, если сигнал на входе условия $\overline{CC} = 0$. Если $\overline{CC} = 1$, то вместо инструкции условного перехода выполняется инструкция с кодом 1.

Шесть инструкций условного перехода к подпрограмме (п/п) имеют коды 22—27. При $\overline{CC} = 0$ под управлением сигналов инструкции на шине Y формируется значение адреса, а в PC загружается значение $Y + K0$. Указатель стека увеличивается, и в стек загружается содержимое PC . Регистр R управляется сигналом на входе \overline{RE} . Если же $\overline{CC} = 1$, то вместо инструкции условного перехода к подпрограмме выполняется инструкция с кодом 1.

Есть две инструкции условного возврата из подпрограммы (коды 28 и 29). При $\overline{CC} = 0$ под управлением инструкции на выходы Y помещается содержимое верхушки стека S или результат операции $S + D + C0$ для инструкций с кодом 28 и 29 соответственно. Значение $Y + K0$ загружается в PC . Значение указателя стека уменьшается (происходит выталкивание из стека). Регистр R управляется сигналом на входе \overline{RE} . При $\overline{CC} = 1$ вместо инструкций условного возврата из подпрограммы выполняется инструкция с кодом 1.

Инструкция с кодом 0 «Сброс» устанавливает в нуль выходы Y , загружает $K0$ в счетчик команд и сбрасывает указатель стека. Содержимое стека не изменяется. Регистр R управляется сигналом на входе \overline{RE} . Инструкция с кодом 10 «Загрузка R » вызывает запись во вспомогательный регистр R с шины адреса. В программный счетчик загружается значение $PC + K0$. Содержимое стека не изменяется. Инструкция с кодом 11 «Заталкивание PC » аналогична инструкции с кодом 1, за ис-

ключением того, что текущее значение PC заталкивается в стек (при этом содержимое указателя стека увеличивается). Инструкция с кодом 12 «Заталкивание D » состоит в том, что внешние данные с шины адреса $D3—D0$ заталкиваются в стек (при этом содержимое указателя стека увеличивается). На выходную шину адреса выводится значение PC , который инкрементируется (в зависимости от значения $K0$). Инструкция с кодом 13 «Выталкивание S » помещает содержимое верхушки стека S на выходную шину адреса, содержимое указателя стека уменьшается. Программный счетчик наращивается в зависимости от $K0$. Регистр R управляется сигналом на входе \overline{RE} . Инструкция с кодом 14 «Выталкивание PC » аналогична инструкции с кодом 1, за исключением того, что содержимое указателя стека в конце такта уменьшается. Инструкция с кодом 15 «Хранение» помещает содержимое PC на шину Y и блокирует PC и стек. Регистр R управляется сигналом на входе \overline{RE} . Инструкция с кодом 30 «Условное хранение» выполняется аналогично инструкции с кодом 15 («Хранение»), если $\overline{CC} = 0$. Если же $\overline{CC} = 1$, то выполняется инструкция с кодом 1. Инструкция с кодом 31 «Приостановка» при $\overline{CC} = 0$ переводит выходную шину адреса в состояние высокого сопротивления, блокирует PC и стек. При $\overline{CC} = 1$ выполняется инструкция с кодом 1. Регистр R в обоих случаях управляется сигналом на входе \overline{RE} .

Назначение выводов микросхемы K1804BY5

14—10 — входы инструкции. Определяют одну из 32 инструкций CAO.

\overline{TEN} — вход разрешения инструкции. Используется для блокирования внутренних регистров: если $\overline{TEN} = 0$, все внутренние регистры находятся под управлением входов инструкции, если $\overline{TEN} = 1$, все регистры (кроме R) и стек заблокированы, регистром R управляет вход \overline{RE} .

\overline{CC} — вход условия. Применяется при выполнении инструкций с кодами 16—31: если $\overline{CC} = 0$, то условие выполняется, если же $\overline{CC} = 1$, то выполняется инструкция с кодом 1.

\overline{RE} — вход разрешения вспомогательного регистра R . Используется для разрешения ($\overline{RE} = 0$) загрузки регистра R с шины адреса $D3—D0$, если не выполняются инструкции с кодами 8, 9 и $\overline{TEN} = 0$.

$C0$ — вход переноса в сумматор.

$C4$ — выход переноса из сумматора.

$\overline{P}, \overline{G}$ — выходы распространения и генерации переноса из сумматора. Используются для организации ускоренного переноса при наращивании CAO.

$K0$ — вход переноса в программный счетчик.

$K4$ — выход переноса из программного счетчика.

$Y3-Y0$ — выходная шина адреса с тристабильными выходами: при $\overline{OE} = 0$ на них появляется информация с выхода полного сумматора.

\overline{OE} — вход разрешения выходов адреса. При $\overline{OE} = 1$ выходная шина адреса $Y3-Y0$ находится в третьем состоянии — состоянии высокого сопротивления.

$D3-D0$ — входная шина адреса. Это входы адреса, используемые для передачи информации в регистр R , в стек или в полный сумматор в зависимости от выполняемой инструкции.

\overline{EMP} — выход признака пустоты стека. Сигнал на выходе \overline{EMP} равен «0», когда стек пуст.

\overline{FL} — выход признака полноты стека. Сигнал на выходе \overline{FL} равен «0», когда стек полон.

C — вход тактовый. Все внутренние регистры синхронизируются фронтом сигнала T (переход из «0» в «1»).

VCC — вывод питания +5 В.

GND — вывод общий.

Необходимость в применении схем адресной обработки, отдельных от АЛУ, осуществляющего обработку операндов, появляется при построении высокопроизводительных процессоров с конвейерной обработкой команд. Схема адресной обработки должна быть хорошо приспособлена к формированию адресов команд и операндов при всех способах адресации: вычислять адрес за один такт и без дополнительных к БИС аппаратных затрат на элементах меньшей степени интеграции. В какой степени отвечает этим требованиям БИС K1804BY5?

Для примера вспомним методы адресации в популярной системе команд микроЭВМ «Электроника-60». Регистровая прямая адресация — регистр содержит операнд. Регистровая косвенная — регистр содержит адрес операнда. Автоинкрементная прямая — регистр содержит адрес операнда, используемый при выборке операнда из памяти, после чего содержимое регистра увеличивается на 2 (на 1 для байтовых команд). Автоинкрементная косвенная — регистр содержит адрес операнда; после его использования содержимое регистра увеличивается на 2. Автодекрементная прямая — содержимое регистра уменьшается на 2 (на 1 для байтовых команд), после чего используется как адрес операнда. Автодекрементная косвенная — содержимое регистра уменьшается на 2 и используется как адрес адреса операнда. Индексная прямая — содержимое регистра складывается со смещением и используется как адрес операнда (смещение расположено в следующей за словом команды ячейке памяти). Индексная косвенная — содержимое регистра складывается со смещением и используется как адрес адреса операнда. Непосредственная — операнд выбирается из ячейки, следующей за словом команды. Абсолютная — из ячейки, следующей за словом команды, выбирается адрес операнда. Относительная — операнд выбирается из ячейки, адрес которой определяется как сумма содержимого счетчика команд и ячейки, следующей за словом команды. Относительная косвенная — из ячейки, адрес которой определяется как сумма содержимого счетчика команд и ячейки, следующей за словом команды, выбирается адрес операнда. Адресация в командах ветвления — формируется адрес ветвления, являющийся суммой содержимого счетчика команд и удвоенного смещения (смещение в виде числа со знаком содержится в слове команды).

Таким образом, без подачи констант из поля микрокоманды с помощью БИС K1804BY5 можно реализовать регистровую прямую адресацию, регистровую косвенную, автоинкрементную прямую адресацию в байтовых командах индексную прямую (при условии, что содержимое регистра предварительно занесено в

стек или во вспомогательный регистр R), индексную косвенную (при том же условии), непосредственную, абсолютную, относительную, относительную косвенную. При условии подачи на вход $D3 - D0$ константы из регистра микрокоманд возможна реализация автоинкрементной прямой адресации, автоинкрементной косвенной, автоинкрементной прямой, автодекрементной косвенной. Затруднено (без дополнительной аппаратуры) формирование с помощью БИС K1804BY5 адресов ветвления в командах условного перехода. Это вызвано отсутствием средств организации левого сдвига в ней (для удвоения смещения), а также тем, что смещение задается в виде числа со знаком. Кроме того, нужно отметить следующие существенные препятствия при использовании этой БИС для реализации системы команд микроЭВМ «Электроника-60». Во-первых, это неограниченная глубина программного стека, организуемого в оперативной памяти (в K1804BY5 имеется аппаратный стек глубиной всего 17 ячеек). Во-вторых, счетчиком команд и указателем стека являются $POH\ R7$ и $R6$, а они отсутствуют в БИС K1804BY5. Наконец, неэффективно выполнение самой распространенной операции — увеличения содержимого счетчика команд на 2. Поэтому здесь в качестве САО более удобно было бы применить микросхему K1804BC2 или K1804BC1.

Таким образом, применению САО в конкретном процессоре должен предшествовать тщательный анализ соответствия архитектуры БИС заданным методам адресации. В отношении рассмотренной микросхемы K1804BY5 это означает, что целесообразно ее применять в узкоспециализированных микроконтроллерах и процессорах, при разработке которых удастся эффективно использовать ее достоинства: аппаратный стек, режимы относительной адресации, инструкции с тестированием условия.

1.3. КОНТРОЛЛЕР ПРЕРЫВАНИЙ

Хорошим введением в организацию систем прерывания является гл. 6 популярной книги [4]. Там же можно найти исчерпывающее описание функционирования микросхем $Am\ 2914$ и $Am\ 2913$, аналогичных отечественным микросхемам K1804BH1 и K1804BP3. Ниже приведем в основном сведения, отсутствующие в [4]: цоколевку микросхем, обозначение выводов и некоторые схемы каскадирования контроллеров прерываний. Место контроллера прерываний в общей структуре микропроцессора обсуждается подробнее в § 5.1. Остановимся на описании основных функциональных узлов, наиболее часто используемых в системах прерывания и нашедших отражение в архитектуре БИС контроллера прерываний.

Каждому источнику прерывания должен соответствовать логический элемент, осуществляющий фиксацию запроса прерывания. Совокупность этих элементов можно представить в виде регистра с числом разрядов, равным числу фиксируемых запросов на прерывание. Для управления прерываниями служит маска прерываний, представляющая собой двоичное слово с числом разрядов, равным числу маскируемых (игнорируемых) причин прерывания. Если разряд маски $M_i = 1$, то прерывание по уровню i запрещено (маскировано); если $M_i = 0$, то прерывание разрешено (не маскировано). Маска прерывания обычно хранится в специальном регистре, называемом регистром маски. Этот регистр позволяет в динамике реализовать программируемую логику работы системы прерывания.

Возможно одновременное получение нескольких запросов прерывания, но в каждый момент времени может обслуживаться только один из них. Порядок вы-

деления одного запроса на прерывание из множества определяется организацией системы прерывания. Аппаратная часть системы прерывания может выполнять набор операций управления. Сброс — приведение элементов системы прерывания в исходное состояние. Маскирование — блокирование некоторых ее элементов. Каждая из названных операций по характеру выполнения может иметь три модификации: общая (выполняется над всеми элементами), индивидуальная (выполняется над отдельными элементами), избирательная или групповая (выполняется над группой элементов). Общий сброс используется при включении источника питания, при отказах, при подготовке к выполнению операции начальной загрузки. Если требуется приведение в исходное состояние некоторой группы сигналов прерывания, применяется избирательный сброс (через маску). Операция блокирования может инициироваться командой программы или появлением запроса на прерывание, принятым на обслуживание процессором. Общее блокирование запрещает поступление запросов на обслуживание прерывания в следующую ступень системы прерывания или в процессор. Установление порога прерывания (избирательное блокирование) — исключение сигналов прерывания, которые имеют приоритет ниже порога.

Алгоритм функционирования системы прерывания определяет основные действия, которые производятся с момента появления сигнала прерывания до окончания обслуживания прерывания. Этот алгоритм называется обобщенной процедурой прерывания.

Этапами такой процедуры являются [4,5]:

1. Формирование запросов прерывания устройствами, требующими обслуживания.
2. Выделение запроса с максимальным приоритетом из множества поступивших.
3. Прерывание выполнения текущей программы, если текущий приоритет меньше приоритета поступившего запроса.
4. Запрещение реакции на другие запросы прерывания.
5. Сохранение состояния прерванной программы — минимума элементов памяти, претерпевающих изменение при переходе к обслуживанию прерывания.
6. Загрузка нового состояния процессора, начального для программы, обслуживающей прерывание.
7. Разрешение реакции на другие запросы прерывания.
8. Выполнение программы обслуживания прерывания.
9. Сброс запроса, вызвавшего прерывание.
10. Возврат к прерванной программе или подпрограмме.

Первые два этапа процедуры прерывания характеризуют время отклика системы на появление причины прерывания. Это время определяется способом обнаружения запросов на прерывание и способом выявления источника запроса. Применяются три вида обнаружения запросов [4]: опрос (в порядке приоритета) запросов прерывания, организация запросов по схеме МОНТАЖНОГО ИЛИ (сигналы запроса имеют активный нулевой уровень) и схема приоритетного прерывания с числом входов, равным числу приоритетов. Возможны комбинации.

Источник запроса может быть выявлен в результате полного опроса в порядке приоритета, с помощью цепочки сигнала подтверждения или по индивидуальной линии запроса.

Прекращение выполнения текущей программы происходит в тот момент, когда запрос на обслуживание прерывания будет обнаружен и проанализирован процессором. Допустимые моменты времени для прекращения выполнения текущей программы определяются разработчиком системы. Ими могут являться последний такт в рамках выполнения любой команды программы или любой такт работы процессора. Конкретный вариант определяется архитектурой процессора, его микропрограммным обеспечением и требованием определенности состояния текущей программы.

При необходимости произвести переключение процессора на выполнение программы, обслуживающей прерывание, встает задача запоминания текущего со-

стояния вычислительного процесса. Сохранение состояния прерываемой программы может быть реализовано программным, аппаратным и программно-аппаратными способами. Время, затрачиваемое на запоминание состояния прерванной программы, является одним из основных факторов, влияющих на время отклика. Известно несколько основных способов сохранения состояния. Так, при возникновении запроса на обслуживание прерывания производится активизация дублирующей аппаратуры. Дублирующими могут быть процессоры, блоки РОН и т. д. Если в системе продублированы все компоненты, содержимое которых может измениться при выполнении обслуживающей программы, то время запоминания состояния является наименьшим. Наибольшая возможная кратность вложенных подпрограмм обслуживания прерывания (глубина прерывания) ограничивается коэффициентом резервирования.

Альтернативой дублированию является использование стекового запоминающего устройства (ЗУ). Стековое ЗУ представляет собой набор регистров (как, например, в микросхеме К1804ВУ5) или специально выделенные для этих целей ячейки ОЗУ машины. Возможны и смешанные сочетания быстродействующих регистров с отдельным счетчиком для нескольких «верхних» ячеек стекового ЗУ в сочетании с регистром-указателем стека и ячейками основной памяти для его «нижних» ячеек. Такая конструкция стекового ЗУ уменьшает число обращений к ОЗУ и тем самым ускоряет работу системы. Глубина прерывания для аппаратного стека определяется числом регистров стекового ЗУ, а для смешанного стекового ЗУ также объемом памяти, отводимой под стек.

Дисциплина обслуживания с относительными приоритетами заключается в том, что из множества запросов на прерывание выбирается для обработки один, наиболее важный, согласно присвоенным приоритетам. Если же в процессе обслуживания этого запроса приходит запрос с большим приоритетом, то последний ожидает обработки до тех пор, пока не будет выполнена текущая программа реакции — на менее важный запрос прерывания, пришедший ранее. Дисциплина обслуживания с абсолютными приоритетами состоит в том, что в каждый момент времени обрабатывается запрос на прерывание с наибольшим приоритетом независимо от того, сколько запросов с меньшим приоритетом ожидает выполнения или довыполнения. Сброс обслуженного запроса прерывания может производиться по инициативе источника запроса или командами обслуживающей программы.

Для реализации аппаратной части систем прерывания предназначены микросхемы К1804ВН1 и К1804ВРЗ. С их помощью обеспечивается регистрация запросов на прерывание, маскирование запросов, выделение запроса с максимальным приоритетом из немаскированных, сравнение приоритета этого запроса с текущим приоритетом процессора, формирование требований прерывания для устройства управления процессора и выработка двоичного кода, соответствующего выделенному запросу.

Контроллер векторных приоритетных прерываний К1804ВН1, структурная схема которого приведена на рис. 1.20, имеет 8 входов запросов на прерывание. Запросы на прерывание могут быть импульсными или потенциальными и фиксируются в специальном регистре внутри БИС. Имеется встроенный регистр маски, над которым предусмотрены 6 различных операций: загрузка регистра маски с шины M , чтение регистра маски на шину M , сброс регистра маски, установка регистра маски, сброс разряда регистра маски по шине M , установка разряда регистра маски по шине M . Встроенный регистр состояния содержит код низшего приоритета разрешенных прерываний. На выходе микросхемы можно получить «вектор» — двоичный код запроса на прерывание, имеющего наивысший приоритет из немаскированных

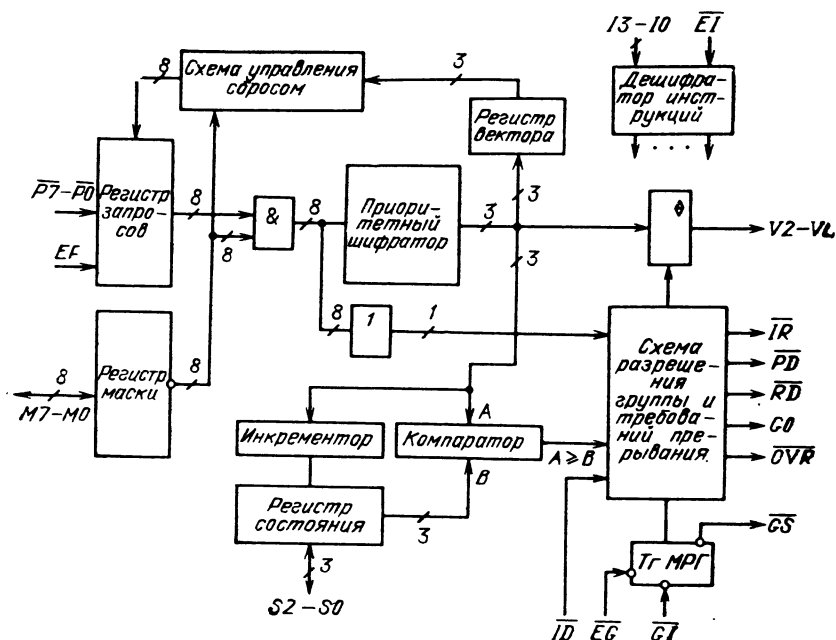
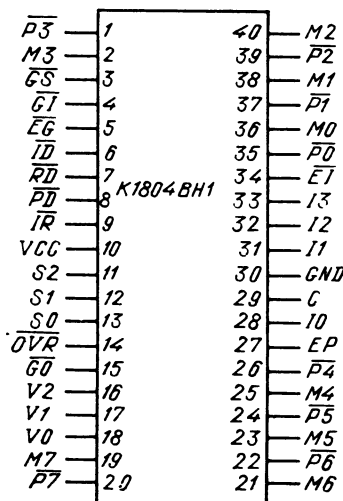


Рис. 1.20 Структурная схема БИС К1804ВН1

запросов. Любое число микросхем К1804ВН1 может быть объединено для построения систем прерывания с числом входных запросов, большим восьми.



Микросхема имеет высокое быстродействие: задержка от поступления запроса на прерывание в регистр прерываний до вывода требуемого вектора прерывания — около 70 нс. Микросхема К1804ВН1 выпускается в 40-выводном корпусе, цоколевка которого приведена на рис. 1.21. Источник питания +5 В.

Назначение выводов микросхемы

$P0-P7$ — входы запросов на прерывание.

$M0-M7$ — разряды маски (двухнаправленная шина).

$I0-I3$ — входы инструкции БИС.

Рис. 1.21. Цоколевка выводов микросхемы К1804ВН1

EP — разрешение ввода запросов на прерывание.

\overline{EI} — разрешение выполнения инструкции БИС.

\overline{GI} — вход межгрупповой связи.

\overline{EG} — вход разрешения группы.

C — вход сигнала синхронизации.

$S0-S2$ — двунаправленная шина состояния.

OVR — выход переполнения регистра состояния.

\overline{IR} — выход требования прерывания.

\overline{ID} — вход запрета прерываний.

\overline{RD} — выход последовательного запрета.

\overline{PD} — выход параллельного запрета.

\overline{GS} — выход сигнала группы.

$V0-V2$ — выход вектора прерывания.

\overline{GO} — выход межгрупповой связи.

VCC — вывод питания.

GND — вывод общий.

Инструкции, выполняемые микросхемой К1804ВН1 при $\overline{EI} = 0$, представлены в табл. 1.39. Дешифратор инструкций формирует требуемые внутренние сигналы управления элементами микросхемы К1804ВН1. При $EP = 0$ вентили фиксатора запросов на прерывание действуют как «ловушки» отрицательных импульсов на входах. При $EP = 1$ фиксаторы запросов прерывания становятся «прозрачными», пропуская запросы на вторую ступень регистра запросов, синхронизируемую тактовым сигналом.

Таблица 1.39. Инструкции микросхемы К1804ВН1

Код				Мнемоника	Операция
I_3	I_2	I_1	I_0		
0	0	0	0	$MCLR$	Общий сброс
0	0	0	1	$CLRIN$	Сброс всех прерываний
0	0	1	0	$CLRMB$	Сброс прерывания по шине M
0	0	1	1	$CLRM R$	Сброс прерывания по регистру маски
0	1	0	0	$CLRV C$	Сброс прерывания по последнему прочитанному вектору
0	1	0	1	$RDVC$	Чтение вектора
0	1	1	0	$RDSTA$	Чтение регистра состояния
0	1	1	1	RDM	Чтение регистра маски
1	0	0	0	$SETM$	Установка регистра маски
1	0	0	1	$LDSTA$	Загрузка регистра состояния
1	0	1	0	$BCLRM$	Сброс разрядов регистра маски
1	0	1	1	$BSETM$	Установка разрядов регистров маски
1	1	0	0	$DISIN$	Запрещение прерываний
1	1	0	1	$CLRM$	Сброс регистра маски
1	1	1	0	LDM	Загрузка регистра маски
1	1	1	1	$ENIN$	Разрешение прерываний

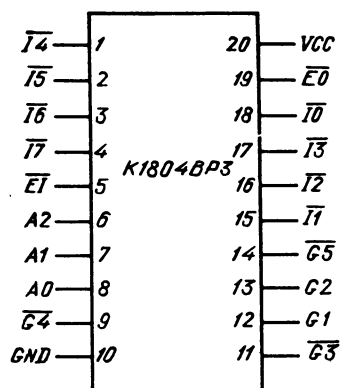


Рис. 1.22. Цоколевка выводов микросхемы К1804ВРЗ

Регистр маски содержит восемь разрядов маски, соответствующих восьми уровням прерывания. Весь регистр или его отдельные разряды могут быть установлены в «1» или сброшены в «0». Кроме того, регистр может быть загружен с шины M или прочитан на нее. Восьмивходовый шифратор определяет наивысший приоритет немаскированного входа запроса на прерывание и формирует двоично-кодированный вектор прерывания $V0—V2$. Схема управления сбросом вырабатывает восемь сигналов сброса для восьми разрядов регистра запросов на прерывание. При выполнении инструкции «Чтение вектора» двоично-кодированный вектор прерывания, сформированный шифратором, загружается в регистр вектора. Запоминаемый вектор может быть использован затем для сброса запроса на прерывание, связанного с последним вектором, который был прочитан. Трехразрядный компаратор сравнивает вектор прерывания с содержанием регистра состояния. Имеющийся на кристалле БИС «триггер младшей разрешенной группы» (ТгМРГ) используется, когда несколько микросхем К1804ВН1 объединяются в каскад. В объединенной системе в любой момент времени только один ТгМРГ имеет на выходе низкий логический уровень. Он индицирует группу, содержащую низший приоритет прерывания, который будет принят, а также используется для формирования старших разрядов состояния. При выполнении инструкции «Общий сброс» ТгМРГ загружается со входа \overline{GI} . Когда выполняется инструкция «Чтение вектора», ТгМРГ устанавливается в «1» при наличии одного из трех условий: не принят сигнал \overline{GI} и не обнаружены прерывания в этой группе; сигнал \overline{GI} послан из этой группы; прерывания из этой группы запрещены.

Регистр состояния может быть загружен с шины S или прочитан на шину S . При выполнении загрузки регистра состояния значение шины S загружается в регистр только в том случае, если $\overline{EG} = 0$. Если же $\overline{EG} = 1$, то производится сброс регистра. При выполнении инструкции «Чтение регистра состояния» выходы регистра разблокированы для шины S в том случае, если ТгМРГ этой группы находится в нулевом состоянии.

При выполнении инструкции «Чтение вектора» инкрементор увеличивает значение вектора на единицу и результат загружается в регистр состояния. Таким образом, регистр состояния всегда указывает наименьший уровень приоритета, на котором может быть принято прерывание.

Схема разрешения группы и требований прерывания вырабатывает сигналы: \overline{RD} (последовательный запрет), \overline{PD} (параллельный запрет), \overline{TR} (требование прерывания), \overline{GO} (выход межгрупповой связи). Выходной сигнал $\overline{TR} = 0$ образуется в том случае, если требование прерывания в этой группе разрешено и обнаружен запрос с достаточным приоритетом. Выходной сигнал $\overline{GO} = 0$ вырабатывается, когда считывается вектор со значением 7 (наивысший приоритет).

Расширитель приоритетного прерывания — микросхема К1804ВРЗ — может использоваться в системах с числом уровней прерывания, большим восьми (номера от 0 до 7), обслуживаемых микросхемами К1804ВН1. Она выполняет функции, сходные с шифрацией приоритета. Цоколевка микросхемы К1804ВРЗ приведена на рис. 1.22.

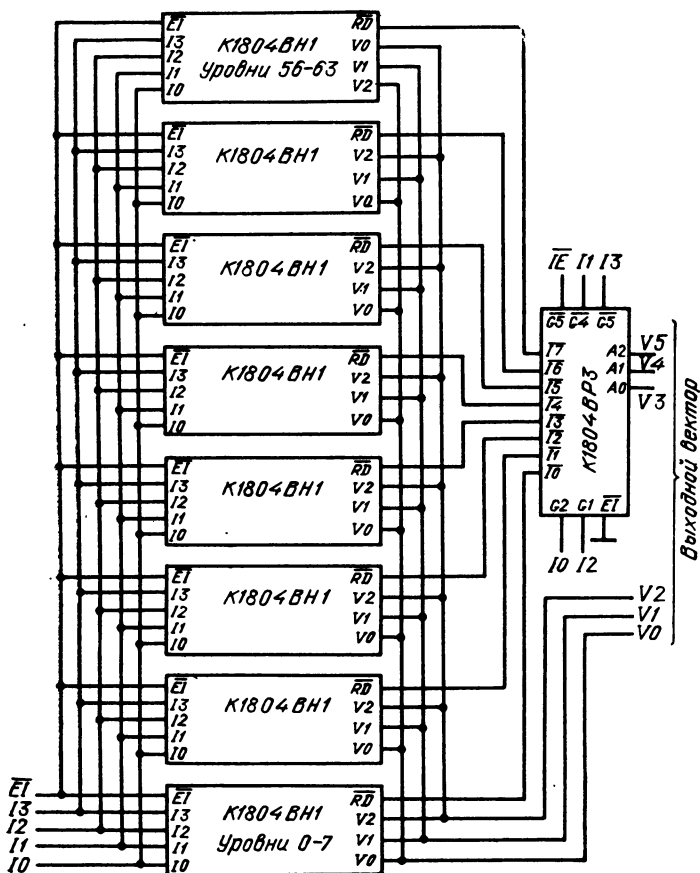


Рис. 1.23. Формирование вектора в 64-разрядной схеме прерывания

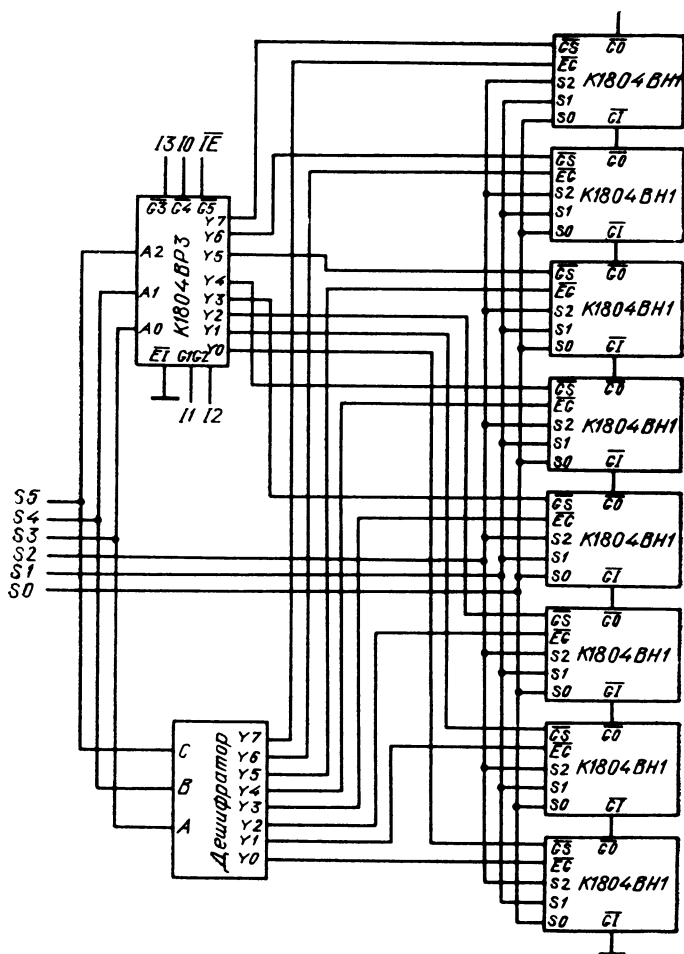


Рис. 1.24. Схема межгрупповой связи

С подробным описанием работы этой микросхемы удобно ознакомиться по [4], где также рассмотрены примеры ее применения в 32-уровневых схемах прерывания. Ниже на рис. 1.23 и 1.24 приведены примеры 64-уровневых схем прерывания.

1.4. СЧЕТЧИКИ АДРЕСОВ ПРЯМОГО ДОСТУПА К ПАМЯТИ

Прямой доступ к памяти (ПДП) позволяет осуществлять непосредственный обмен данными между памятью и периферийными устройствами под управлением контроллера ПДП без участия процессора, что повышает скорость выполнения обмена.

Устройство, осуществляющее управление передачей данных при ПДП, называется контроллером ПДП и выполняет следующие функции: управление шиной адреса, управление передачей данных, формирование адреса, подсчет числа слов, управление режимов передачи. Контроллер ПДП должен передавать соответствующий адрес на шину адреса памяти и вырабатывать сигналы управления передачей данных между памятью и устройством ввода-вывода. Контроллер ПДП должен содержать указатель адреса, который формирует адрес ячейки памяти, из которой считывается или в которую записывается очередное передаваемое слово данных. Содержимое этого указателя должно увеличиваться или уменьшаться после передачи очередного слова данных. Перед началом передачи данных в контроллер ПДП поступает информация из процессора о числе передаваемых слов. Во время передачи данных контроллер ПДП должен выполнять подсчет числа переданных слов и закончить передачу по достижении заданного числа слов. Таким образом, контроллер ПДП должен содержать схему управления режимом передачи, которая определяет направление потока данных, способ определения конца передачи и т. д. и устанавливается в нужное состояние центральным процессором перед началом обмена. Контроллер ПДП может быть размещен в каждом устройстве ввода-вывода (распределенная система ПДП). В другом варианте контроллеры ПДП нескольких периферийных устройств могут быть выделены и в отдельное устройство (централизованная система ПДП).

Восьмиразрядные счетчики адресов ПДП К1804ВУ6 и К1804ВУ7 предназначены для использования в контроллерах ПДП. Основные функции адресного генератора ПДП заключаются в формировании последовательности адресов ячеек памяти при передаче данных в память или из памяти, в подсчете числа переданных слов и формировании соответствующего сигнала по завершении передачи данных. Адресный генератор ПДП К1804ВУ7 кроме перечисленных функций может выполнять еще и функцию программируемого счетчика.

Адресные генераторы ПДП К1804ВУ6 и К1804ВУ7 имеют следующие особенности: возможность наращивания для получения адреса любой разрядности, кратной восьми (три адресных генератора ПДП позволяют адресовать 16 мегаслов); при использовании микросхемы К1804ВУ7 в качестве программируемого счетчика допустимо как внутреннее наращивание за счет соединения двух внутренних 8-разрядных счетчиков в один 16-разрядный, так и внешнее наращивание путем соединения нескольких БИС К1804ВУ7. Схемы микропрограммируемые: адресный генератор ПДП К1804ВУ6 выполняет восемь инструкций, а адресный генератор К1804ВУ7 — восемнадцать (девять инструкций как адресный генератор и девять инструкций как программируемый счетчик). Имеется возможность выполнения повторной передачи данных без участия процессора за счет сохранения внутри адресного генератора ПДП начального адреса (в регистре адреса) и числа передаваемых слов (в регистре счетчика слов).

Микросхемы К1804ВУ6 и К1804ВУ7 изготавливаются по технологии маломощных ТТЛ-схем с диодами Шотки. Корпус К1804ВУ6 имеет 28 выводов, а корпус К1804ВУ7 — 22. Типовое значение потребляемой мощности не превышает 1 Вт. Источник питания +5 В. Цоколевка корпусов микросхем К1804ВУ6 и К1804ВУ7 приведена на рис. 1.25 и 1.26.

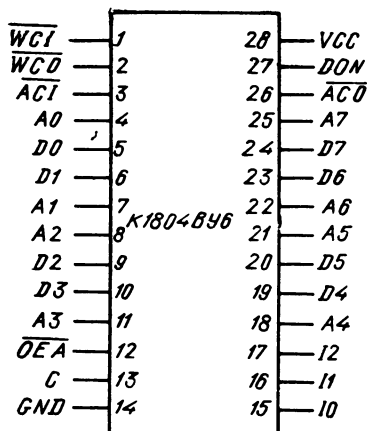


Рис. 1.25. Цоколевка выводов микро-
схемы K1804BY6

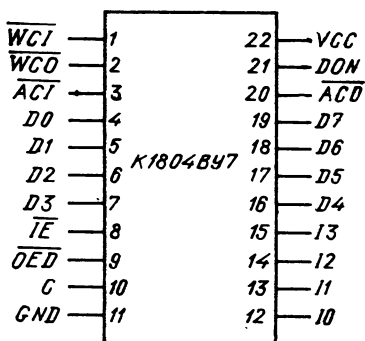


Рис. 1.26. Цоколевка выводов микро-
схемы K1804BY7

Структурная схема адресных генераторов ПДП показана на рис. 1.27. В схеме можно выделить десять основных блоков: регистр управления, регистр адреса, мультиплексор адреса, счетчик адреса, регистр числа слов, мультиплексор счетчика слов, счетчик слов, схема определения конца передачи, мультиплексор данных с триста-бильным выходом, дешифратор инструкций.

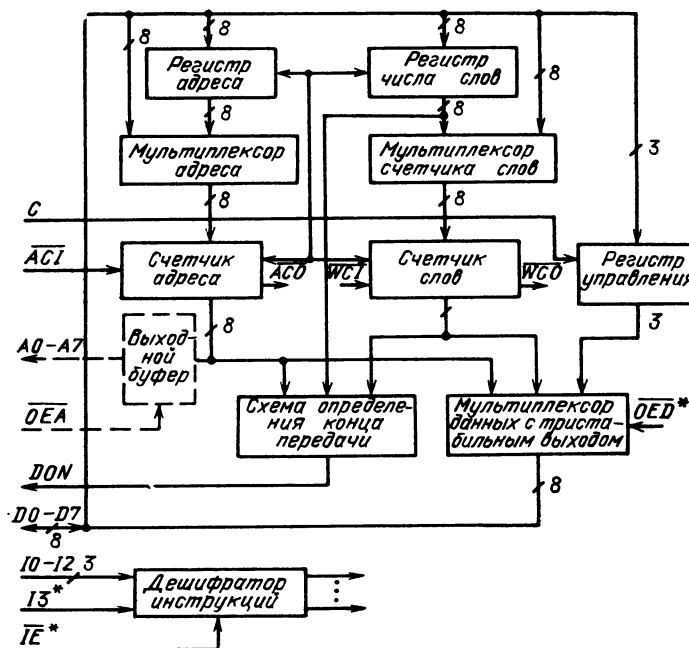


Рис. 1.27. Структурная схема генераторов адреса прямого доступа к памяти

Таблица 1.40. Режимы управления передачей данных

Разряды РгУ			Номер режима управления	Режим управления	Счетчик слов	Значение сигнала <i>DON</i>		Счетчик адреса
2	1	0				при $WC1=0$	при $WC1=1$	
X	0	0	0	Равенство нулю содержимого СчС	Уменьшение	1, если $CчC=1$	1, если $CчC=0$	X
X	0	1	1	Сравнение содержимого РгС и СчС	Увеличение	1, если $CчC+1=$ $=PгC$	1, если $CчC=PгC$	X
X	1	0	2	Сравнение содержимого СчА и СчС	Уменьшение	1, если $CчC=CчA$	1, если $CчC=CчA$	X
X	1	1	3	Определение конца передачи по сигналу выходного переноса СчС	Увеличение	0	0	X
0	X	X	X	X	X	X	X	Увеличение
1	X	X	X	X	X	X	X	Уменьшение

му определения конца передачи, мультиплексор данных, дешифратор инструкций. Компоненты, имеющиеся помимо общей части только в микросхеме К1804ВУ7, помечены на рис. 1.27 символом *, а те, которые имеются только в микросхеме К1804ВУ6, — штриховой линией.

Трехразрядный регистр управления (РгУ) предназначен для хранения информации, определяющей режим управления передачей данных (разряды 0 и 1) и режим работы счетчика адреса (разряд 2), которые приведены в табл. 1.40. Запись информации в регистр управления производится со входов *D0—D2* по фронту тактового сигнала *C*.

Восьмиразрядный регистр адреса предназначен для записи и хранения начального адреса передаваемых данных. После окончания передачи данных этот начальный адрес может быть вновь передан в счетчик адреса для выполнения повторной передачи данных. Запись информации в регистр адреса производится с шины *D* по фронту тактового сигнала *C*.

Двухвходовый 8-разрядный мультиплексор адреса передает на вход счетчика адреса информацию либо с шины данных *D*, либо с выхода регистра адреса в зависимости от выполняемой инструкции.

Счетчик адреса (СчА) представляет собой 8-разрядный двоичный реверсивный счетчик со сквозным переносом, предназначенный для формирования адреса очередного передаваемого слова. Наличие в счетчике входа переноса \overline{ACI} и выхода переноса \overline{ACO} позволяет выполнять наращивание счетчика для получения адреса большей разрядности. В зависимости от выполняемой инструкции осуществляется запрещение, разрешение или загрузка счетчика адреса с шины или с

выхода регистра адреса. Запись информации в счетчик адреса производится по фронту тактового сигнала. При разрешении $C\check{A}$ и подаче «0» на вход \overline{ACI} по фронту тактового сигнала C происходит увеличение или уменьшение содержимого $C\check{A}$. В микросхеме K1804BY6 в отличие от микросхемы K1804BY7 имеется выходной буфер адреса с тристабильными выходами, управляемыми входом разрешения выдачи адреса (\overline{OEA}), который позволяет передавать содержимое $C\check{A}$ на специальные выходы адреса $A0\text{—}A7$. На структурной схеме (рис. 1.27) выходной буфер адреса, вход \overline{OEA} и выходы $A0\text{—}A7$ обозначены штриховой линией. При нулевом сигнале на входе разрешения выдачи адреса \overline{OEA} содержимое $C\check{A}$ передается на тристабильные выходы адреса $A0\text{—}A7$. При $\overline{OEA} = 1$ шина A отключается (находится в состоянии высокого сопротивления).

Восьмиразрядный регистр числа слов ($R\check{G}C\check{C}$) предназначен для записи и хранения информации о количестве передаваемых слов, поступающей с шины D . После завершения передачи данных информация из $R\check{G}C\check{C}$ может быть вновь загружена в счетчик слов ($C\check{C}C$) для выполнения повторной передачи данных. Запись информации в $R\check{G}C\check{C}$ производится по фронту тактового сигнала.

Двухвходовый 8-разрядный мультиплексор передает на вход $C\check{C}C$ информацию либо с шины D , либо с выхода $R\check{G}C\check{C}$. Счетчик слов представляет собой 8-разрядный двоичный реверсивный счетчик со сквозным переносом и предназначен для подсчета числа переданных слов. Наличие входа переноса (\overline{WCI}) и выхода переноса (\overline{WCO}) позволяет осуществлять наращивание счетчика для получения большей разрядности. В зависимости от выполняемой инструкции производится запрещение, разрешение или загрузка $C\check{C}C$ с шины D или с выхода регистра $C\check{C}C$. Запись информации в $C\check{C}C$ выполняется по фронту тактового сигнала C . При разрешении $C\check{C}C$ и подаче «0» на вход \overline{WCI} по фронту тактового сигнала происходит увеличение или уменьшение содержимого $C\check{C}C$. Схема определения конца передачи определяет окончание передачи данных в трех режимах управления (режимы 0, 1, 2, табл. 1.40) и вырабатывает соответствующий сигнал (\overline{DON}).

Трехвходовый 8-разрядный мультиплексор $M\check{P}D$ передает на двунаправленную шину данных D информацию с выхода $C\check{A}$, $C\check{C}C$ или $R\check{G}Y$. Тристабильные выходы мультиплексора данных разрешаются внутренним управляющим сигналом с выхода дешифратора инструкций в микросхеме K1804BY6 или внешним сигналом разрешения выдачи данных $\overline{OED} = 0$ в микросхеме K1804BY7. В противном случае выходы мультиплексора отключаются (находятся в состоянии высокого сопротивления). На структурной схеме рис. 1.27 вход \overline{OED} отмечен звездочкой, так как он относится только к микросхеме K1804BY7.

Дешифратор инструкций вырабатывает необходимые внутренние управляющие сигналы в зависимости от сигналов инструкции (сиг-

налы 10—12 в микросхеме K1804BY6 и сигналы 10—13 в микросхеме K1804BY7) и значений разрядов 0 и 1 регистра управления. В микросхеме K1804BY7 в отличие от K1804BY6 имеется вход разрешения инструкций (\overline{IE}), который блокирует входы инструкции 10—12 при $\overline{IE} = 1$. Вход \overline{IE} , так же как и входы 13 и \overline{OED} , на структурной схеме рис. 1.27 отмечен звездочкой как относящийся только к микросхеме K1804BY7. Микросхемы используются в следующих режимах.

Режим равенства нулю содержимого СчС (режим 0). В этом режиме управления информация о числе передаваемых слов загружается в РгЧС и в СчС. При разрешении СчС и подаче на вход переноса $\overline{WCI} = 0$ по фронту тактового сигнала происходит уменьшение содержимого СчС (обратный счет). Сигнал $DON = 1$, определяющий конец передачи, вырабатывается, когда содержимое СчС равно 1 (при $\overline{WCI} = 0$) или 0 (при $\overline{WCI} = 1$), т. е. при передаче последнего слова данных (см. табл. 1.40).

Режим сравнения содержимого РгЧС и СчС (режим 1). В режиме 1 информация о числе передаваемых слов данных записывается в РгЧС, а СчС устанавливается в нуль. Затем при разрешении СчС и подаче $\overline{WCI} = 0$ по фронту тактового сигнала происходит увеличение содержимого СчС на единицу. При этом содержимое СчС определяет число переданных слов. Схема определения конца передачи сравнивает содержимое СчС с содержимым РгЧС и вырабатывает сигнал $DON = 1$, когда содержимое СчС плюс 1 равно содержимому РгЧС (при $\overline{WCI} = 0$) или когда содержимое СчС равно содержимому РгЧС (при $\overline{WCI} = 1$), т. е. во время передачи последнего слова данных (см. табл. 1.40).

Режим сравнения содержимого СчА и СчС (режим 2). В этом режиме управления передачей должны быть определены начальный и конечный адреса передаваемых слов. Начальный адрес загружается в РгА и СчА, а конечный адрес — в РгЧС и в СчС. При этом СчС запрещается и используется как регистр хранения конечного адреса. Счетчик адреса разрешается; и при подаче $\overline{ACI} = 0$ по фронту тактового сигнала происходит увеличение или уменьшение содержимого СчА в зависимости от значения разряда 2 регистра управления (см. табл. 1.28). Схема определения конца передачи сравнивает содержимое и вырабатывает сигнал $DON = 1$, когда содержимое СчА равно содержимому СчС, т. е. при передаче последнего слова данных (см. табл. 1.40).

Режим определения конца передачи по сигналу выходного переноса СчС (режим 3). В этом режиме РгЧС и СчС загружаются обратным кодом числа передаваемых слов данных. При разрешении СчС и подаче $\overline{WCI} = 0$ по фронту тактового сигнала происходит увеличение содержимого СчС. При передаче последнего слова данных на выходе переноса СчС формируется сигнал $\overline{WCO} = 0$. Сигнал DON в этом режиме не используется, он всегда равен нулю (см. табл. 1.40).

Таблица 1.41. Инструкция

<i>I2—I0</i>	Инструкция	Мнемоника	Режим управления
0 0 0	Запись в регистр управления	<i>WRCR</i>	0, 1, 2, 3
0 0 1	Чтение регистра управления	<i>RDCR</i>	0, 1, 2, 3
0 1 0	Чтение счетчика слов	<i>RDWC</i>	0, 1, 2, 3
0 1 1	Чтение счетчика адреса	<i>RDAC</i>	0, 1, 2, 3
1 0 0	Восстановление счетчиков	<i>REIN</i>	0, 2, 3 1
1 0 1	Запись адреса	<i>LDAD</i>	0, 1, 2, 3
1 1 0	Запись в счетчик слов	<i>LDWC</i>	0, 2, 3 1
1 1 1	Разрешение счетчиков	<i>ENTC</i>	0, 1, 3 2

Инструкции, выполняемые генератором адресов ПДП К1804ВУ6, приведены в табл. 1.41. По инструкции «Запись в регистр управления» производится запись данных со входов *D0—D2* в РГУ; входы *D3—D7* не используются. По инструкции «Чтение регистра управления» его содержимое передается на двунаправленные выходы *D0—D2*, которые функционируют как выходы. На выводах *D3—D7* устанавливаются уровни лог 1. При выполнении инструкции «Чтение счетчика слов» информация с выхода СчС передается на шину данных *D*, которая работает как выход. При выполнении инструкции «Чтение счетчика адреса» на шину данных *D* передается содержимое СчА. Выполнение инструкции «Восстановление счетчиков» зависит от режима управления. В режимах 0, 2, 3 содержимое РгА и РгЧС передается соответственно в СчА и СчС; в режиме управления 1 содержимое РгА передается в СчА, а СчС устанавливается в нуль. При этом шина *D* находится в состоянии высокого сопротивления. Данная инструкция позволяет повторить передачу данных без перезагрузки адреса и числа передаваемых слов с шины данных в соответствующие регистры. По инструкции «Запись адреса» информация с шины данных *D* записывается в РгД и СчА. Выполнение инструкции «Запись в счетчик слов» зависит от режима управления. В режимах управления 0, 2, 3 информация с шины данных *D* записывается и в СчС, и в РгЧС. В режиме управления информация с шины данных *D* записывается только в РгЧС, а СчС устанавливается в нулевое состояние. Выполнение инструкции «Разрешение счетчиков» также зависит от режима управления. В режимах управления 0, 1, 3 производится разрешение как СчА, так и СчС; в режиме управления 2 счетчик адреса разрешается, а СчС сохраняет состояние. При выполнении этой инструкции шина данных *D* находится в состоянии высокого сопротивления. При разрешении счетчиков и подаче лог. 0 на входы переноса \overline{ACI} , \overline{WCI} по фронту тактового сигнала

микросхемы К1804ВУ6

Функция регистров					Шина D0—D7
РгЧС	СчС	РгА	СчА	РгУ	
Хранение	Хранение	Хранение	Хранение	D0—D2→РгУ	Вход
»	»	»	»	Хранение	РгУ→D0—D2 D3—D7 = 111
»	»	»	»	»	СчС→D
»	»	»	»	»	СчА→D
»	РгЧС→СчС	»	РгА→СчА	»	Z
»	0→СчС	»	РгА→СчА	»	Z
»	Хранение	D→РгА	D→РгА	»	Вход
D→РгСч	D→СчС	Хранение	Хранение	»	»
D→РгСч	0→СчС	»	»	»	»
Хранение	Хранение	»	Разрешение	»	Z
»	Разрешение	»	»	»	Z

С происходит изменение состояния счетчиков (режим счета). Таким образом, при выполнении этой инструкции можно с помощью входов переноса управлять работой счетчиков — разрешать или запрещать счет.

Назначение выводов микросхемы К1804ВУ6

D0—D7 — двунаправленная шина данных. Используется для ввода/вывода информации.

A0—A7 — выход адреса. Предназначен для вывода информации из СчА.

$\overline{OE\bar{A}}$ — вход разрешения выдачи адреса. Используется для разрешения тристабильной шины A: при $\overline{OE\bar{A}} = 0$ разрешается вывод содержимого СчА через шину A, при $\overline{OE\bar{A}} = 1$ шина A отключается (находится в состоянии высокого сопротивления).

I0—I2 — вход инструкции. Определяет одну из восьми инструкций адресного генератора ПДП.

\overline{ACI} — вход переноса счетчика адреса. Используется как вход переноса счетчика адреса (при $\overline{ACI} = 0$ содержимое СчА увеличивается на 1, при $\overline{ACI} = 1$ содержимое счетчика не меняется), а также для управления работой счетчика при его разрешении (при $\overline{ACI} = 0$ разрешается режим счета).

\overline{ACO} — выход переноса счетчика адреса; $\overline{ACO} = 0$, если $\overline{ACI} = 0$ и во всех разрядах СчА единицы.

\overline{WCI} — вход переноса счетчика слов. Используется как вход переноса счетчика слов (при $\overline{WCI} = 0$ содержимое СчС увеличивается на 1, при $\overline{WCI} = 1$ содержимое СчС не меняется) и для управления работой СчС при его разрешении (при $\overline{WCI} = 0$ разрешается режим счета).

\overline{WCO} — выход переноса счетчика слов; $\overline{WCO} \Rightarrow 0$, если $\overline{WCI} = 0$ и во всех разрядах СчС единицы.

DON — выход индикации конца передачи. Наличие «1» на этом выходе определяет окончание передачи данных в трех режимах управления. Выход с открытым коллектором.

C — вход тактовый. Используется для синхронизации работы блоков.

VCC — вывод питания.

GND — вывод общий.

По фронту тактового сигнала происходит запись в регистры и изменение содержимого счетчиков на единицу.

В табл. 1.42 приведены инструкции, выполняемые адресным генератором ПДП К1804ВУ7. Из восемнадцати инструкций, входящих в набор, девять соответствуют применению схемы в качестве адресного генератора ПДП и девять — в качестве программируемого счетчика. Режим работы микросхемы К1804ВУ7 определяется сигналом на входе инструкции $I3$: при $I3 = 0$ она работает как адресный генератор ПДП, а при $I3 = 1$ — как программируемый счетчик.

Первые восемь инструкций в табл. 1.42 полностью совпадают с набором инструкций для микросхемы К1804ВУ6. Кроме того, в качестве адресного генератора ПДП микросхемы К1804ВУ7 выполняет еще одну инструкцию — «Запрещение инструкции». При подаче «1» на вход разрешения инструкций \overline{TE} входы инструкций $I0$ — $I2$ блокируются, и если $I3 = 0$, то выполняется инструкция, аналогичная инструкции «Разрешение счетчиков». Таким образом можно управлять работой счетчиков (разрешать или запрещать счет) с помощью входов переноса \overline{ACI} , \overline{WCI} при выполнении инструкций «Разрешение счетчиков» или «Запрещение инструкции».

Инструкции, выполняемые микросхемой К1804ВУ7 в качестве программируемого счетчика, отличаются от описанных выше тем, что позволяют осуществлять независимое управление СчА, СчС и РгУ. По инструкции «Запись в регистр управления» информация с выводов $D0$ — $D2$ записывается в регистр управления, СчА и СчС разрешаются, и на выходы мультиплексора данных передается содержимое РгУ. Выводы $D3$ — $D7$ не используются. При выполнении инструкции «Восстановление счетчика адреса» содержимое РгА передается в СчА, счетчик слов разрешается и на выходах мультиплексора данных появляется содержимое СчА. По инструкции «Чтение счетчика слов» оба счетчика разрешаются и содержимое СчС передается на выход мультиплексора данных. Аналогично при выполнении инструкции «Чтение счетчика адреса» оба счетчика разрешаются и содержимое СчА поступает на выходы мультиплексора данных.

Выполнение инструкции «Восстановление счетчика адреса и счетчика слов» зависит от режима управления. В режимах 0, 2, 3 содержимое РгА и РгС передается соответственно в СчА и СчС; в режиме управления 1 содержимое РгА передается в СчА, а СчС устанавливается

Т а б л и ц а 1.42. Инструкции микросхемы К1804ВУ7

\overline{TE}	I3—I0	Инструкция	Мнемоника	Режим управления	Функция регистров				Передача через мультиплексор данных
					РЧЧС	СЧС	РГА	СЧА	
0	0 0 0 0	Запись в регистр управления	WRCR	0, 1, 2, 3	Хранение	Хранение	Хранение	Хранение	Единицы
0	0 0 0 1	Чтение регистра управления	RDCR	0, 1, 2, 3	»	»	»	»	РГУ
0	0 0 1 0	Чтение счетчика слов	RDWC	0, 1, 2, 3	»	»	»	»	СЧС
0	0 0 1 1	Чтение счетчика адреса	RDAC	0, 1, 2, 3	»	»	»	»	СЧА
0	0 1 0 0	Восстановление счетчиков	REIN	0, 2, 3	»	РЧЧС → СЧС	»	РГА → СЧА	СЧА
0	0 1 0 1	Запись адреса	LDAD	1	»	0 → СЧС	»	РГА → СЧА	СЧА
0	0 1 1 0	Запись в счетчик слов	LDWC	0, 1, 2, 3 0, 2, 3	» D → РЧЧС	Хранение D → СЧС	D → РГА Хранение	D → СЧА Хранение	СЧС Единицы
0	0 1 1 1	Разрешение счетчиков	ENCD	1 0, 1, 3	D → РЧЧС Хранение	0 → СЧС Разрешение	» »	» Разрешение	» СЧА
1	0 X X X	Запрещение инструкции	—	2 0, 1, 3 2	» » »	Хранение Разрешение Хранение	» » »	» То же »	СЧА СЧА СЧА

\overline{TE}	I3—I0	Инструкция	Мнемоника	Режим управления	Функция регистров				Передача через мультиплексор данных
					РЧС	СЧС	РГА	СЧА	РГУ
0	1 0 0 0	Запись в регистр управления	WCRT	0, 1, 2, 3	Хранение	Разрешение	Хранение	Разрешение	D0—D2→РГУ
0	1 0 0 1	Восстановление счетчика адреса	REAC	0, 1, 2, 3	»	»	»	РГА→СЧА	Хранение
0	1 0 1 0	Чтение счетчика слов	RWCT	0, 1, 2, 3	»	»	»	Разрешение	»
0	1 0 1 1	Чтение счетчика адреса	RACT	0, 1, 2, 3	»	»	»	То же	»
0	1 1 0 0	Восстановление счетчика адреса и счетчика слов	RAWC	0, 2, 3	»	РГЧС→СЧС	»	РГА→СЧА	»
0	1 1 0 1	Запись адреса	LDAT	0, 1, 2, 3	»	0→СЧС	«	РГА→СЧА	»
0	1 1 1 0	Запись в счетчик слов	LWCT	0, 2, 3	D→РГЧС	Разрешение D→РГЧС	D→РГА	D→СЧА	»
0	1 1 1 1	Восстановление счетчика слов	REWC	1	0→РГЧС	0→СЧС	»	Разрешение	»
1	1 X X X	Запрещение инструкции	—	0, 1, 3	»	РГЧС→СЧС	»	То же	»
				2	»	0→СЧС	»	»	»
					»	Разрешение	»	»	»
					»	Хранение	»	»	»

в нуль. При выполнении этой инструкции содержимое СчА передается на выходы мультиплексора данных.

По инструкции «Запись адреса» информация с шины данных D записывается в РгА и СчА, счетчик слов разрешается и его содержимое передается на выходы мультиплексора данных. Выполнение инструкции «Запись в счетчик слов» зависит от режима управления. В режимах 0, 2, 3 информация с шины данных записывается и в СчС, и в РгЧС; в режиме управления 1 информация с шины данных записывается только в РгЧС, а СчС устанавливается в нуль. При выполнении этой инструкции СчА разрешается, а на выходах мультиплексора данных устанавливаются единичные значения. При выполнении инструкции «Восстановление счетчика слов» в режимах 0, 2, 3 содержимое РгЧС записывается в СчС, а в режиме управления 1 СчС устанавливается в нуль. При этом СчА разрешается, а содержимое СчС передается на выходы мультиплексора данных. Выполнение инструкции «Запрещение инструкции» также зависит от режима управления. В режимах управления 0, 1, 3 СчА и СчС разрешаются. В режиме управления 2 СчА разрешается, а СчС сохраняет свое состояние. На выходы мультиплексора данных передается содержимое СчС. Напомним, что информация с выходов мультиплексора данных поступает на шину данных D только при подаче «0» на вход разрешения выдачи данных \overline{OED} . При $\overline{OED} = 1$ выходы мультиплексора данных отключаются (находятся в состоянии высокого сопротивления).

Назначение выводов микросхемы K1804BV7.

$D0-D7$ — двунаправленная шина данных. Используется для ввода/вывода информации.

\overline{OED} — вход разрешения выдачи данных. Используется для разрешения тристабильных выходов мультиплексора данных: при $\overline{OED} = 0$ разрешается вывод данных с выхода мультиплексора данных через шину D , при $\overline{OED} = 1$ выходы мультиплексора данных находятся в состоянии высокого сопротивления.

$I0-I3$ — вход инструкции. Определяет одну из 16 инструкций. При $I3 = 0$ схема работает как адресный генератор ПДП, а при $I3 = 1$ — как программируемый счетчик.

\overline{IE} — вход разрешения инструкции. При $\overline{IE} = 0$ разрешается выполнение соответствующей инструкции, а при $\overline{IE} = 1$ входы инструкции $I0-I2$ блокируются.

\overline{ACI} — вход переноса СчА. Используется как вход переноса (при $\overline{ACI} = 0$ содержимое СчА увеличивается на 1, при $\overline{ACI} = 1$ его содержимое не меняется) и для управления работой СчА (при $\overline{ACI} = 0$ разрешается режим счета).

\overline{ACO} — выход переноса счетчика адреса; $\overline{ACO} = 0$, если $\overline{ACI} = 0$ и во всех разрядах СчА единицы.

\overline{WCI} — вход переноса СчС. Используется как вход переноса (при $\overline{WCI} = 0$ содержимое СчС увеличивается на 1, при $\overline{WCI} = 1$ его содер-

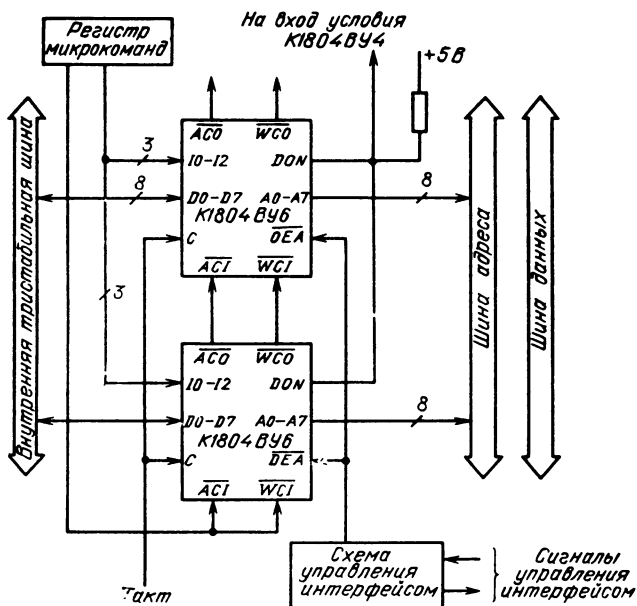


Рис. 1.28. Схема объединения БИС К1804ВУ6

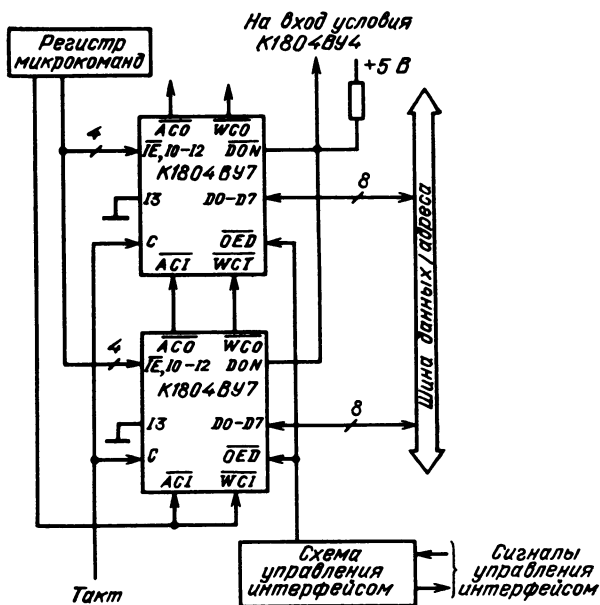
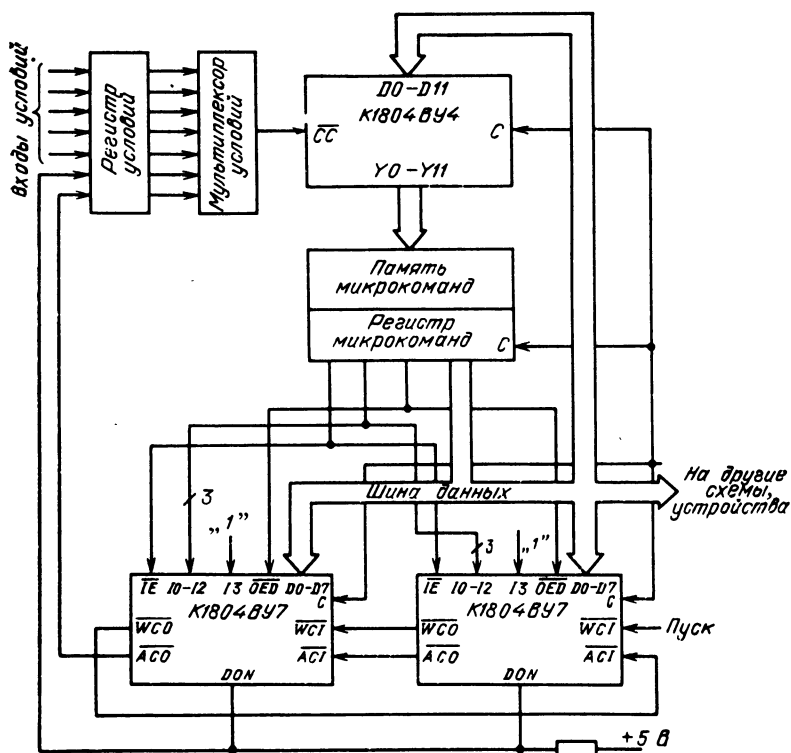


Рис. 1.29. Схема объединения БИС К1804ВУ7



жимое не меняется) и для управления работой СчС (при $\overline{WCI} = 0$ разрешается режим счета).

товые входы. Выходы *DON* с открытым коллектором всех БИС также объединяются.

На рис. 1.28 показано соединение двух микросхем K1804ВУ6 для получения 16-разрядного адресного генератора ПДП. Входы инструкций (10—12) обеих БИС и входы переносов \overline{ACI} , \overline{WCI} младшей БИС соединяются с соответствующими выходами регистра микрокоманд. На рис. 1.29 представлена схема соединения двух микросхем K1804ВУ7 для получения 16-разрядного адресного генератора ПДП. Общая шина данных и адреса соединяется с двунаправленными выводами данных *D*. Соответствующие выходы регистра микрокоманд соединены со входами инструкций (10—12) и входом разрешения инструкций \overline{TE} обеих микросхем, а также со входами переносов \overline{ACI} , \overline{WCI} младшей из них. Входы инструкций 13 заземляются, так как микросхемы работают в режиме адресного генератора ПДП.

Использование адресного генератора ПДП в периферийном контроллере описано в [4]. Там же рассмотрены варианты построения программируемых 8-, 16- и 32-разрядных счетчиков. Один из примеров приведен на рис. 1.30. Здесь показано объединение двух микросхем K1804ВУ7 для получения 32-разрядного программируемого счетчика. Два СчС образуют 16 младших разрядов, а два СчА — 16 старших разрядов 32-разрядного счетчика. При этом выход СчС старшей микросхемы соединяется с входом переноса СчА младшей. Эта схема позволяет загружать и читать одновременно 16 разрядов слова.

1.5. МИКРОПРОГРАММИРУЕМЫЙ ТАКТОВЫЙ ГЕНЕРАТОР

Микросхема тактового генератора (ТГ) K1804ГГ1 предназначена для построения систем синхронизации цифровых устройств, прежде всего микропрограммных процессоров и контроллеров с повышенными требованиями к их производительности. Микросхема ТГ может вырабатывать одну из восьми диаграмм для четырех синхросерий, что представляет разработчику возможность реализации в процессоре (контроллере) переменной длительности такта, повышающей быстродействие на десятки процентов (вопросы правильного выбора длительностей такта рассматриваются в гл. 2). Микросхема ТГ содержит внутренний генератор опорной синхропоследовательности: опорная частота (до 30 МГц) задается внешним кварцевым резонатором. Предусмотрены также сервисные режимы для отладки процессоров (контроллеров) в пошаговом режиме.

Цоколевка микросхемы приведена на рис. 1.31, а структурная схема ТГ — на рис. 1.32. В последней можно выделить основные блоки: регистр микрокоманды, логика управления состоянием, регистр управления, блоки управления режимами «Работа», «Приостановка», «Шаг» и «Ожидание», внутренний генератор опорной синхропоследовательности.

Регистр микрокоманды состоит из трех триггеров типа «защелка», на входы которых можно либо подать постоянную комбинацию сигналов, либо соединить их с выходами соответствующих разрядов микропрограммной памяти, что позволит в каждом такте управлять диаграммой синхропоследовательностей. Информация в регистре микрокоманды фиксируется по фронту сигнала $C1$ и определяет одну из восьми диаграмм синхропоследовательностей ($F3—F10$) для текущего такта. Диаграммы формируются по следующему правилу. Сигнал на тактовом выходе $C1$ имеет нулевой уровень на последнем микротакте опорной синхропоследовательности $F0$ и единичное значение на всех предыдущих микротактах данного такта. Сигнал на тактовом выходе $C2$ имеет единичное значение на всех микротактах текущего такта, кроме двух последних. Сигнал $C4$ имеет нулевой уровень на первом микротакте текущего такта и единичный уровень на всех последующих микротактах. Сигнал на выходе $C3$ имеет единичное значение приблизительно на половине такта: моментом перехода его в нулевое состояние является начало n -го микротакта по отношению к началу такта. В табл. 1.43 приведены значения n , соответствующие кодам на входе

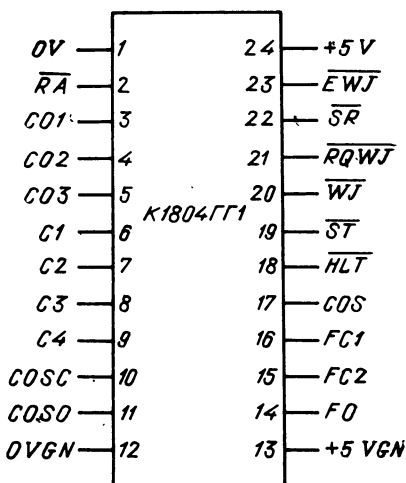


Рис. 1.31. Цоколевка выводов микросхемы К1804ГГ1

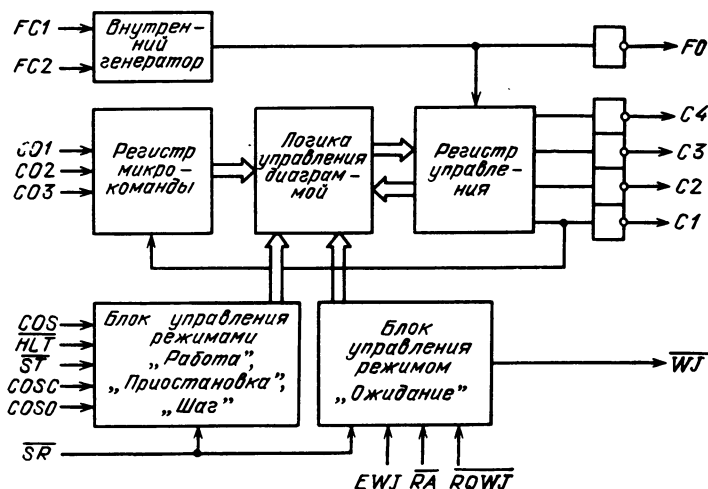


Рис. 1.32. Структурная схема БИС К1804ГГ1

Т а б л и ц а 1.43. Формирование сигнала $C3$

Код микро-команды $COS-COI$	Обозначение диаграммы	Число микро-тактов в такте	n	Код микро-команды $COS-COI$	Обозначение диаграммы	Число микро-тактов в такте	n
000	$F3$	3	2	011	$F7$	7	4
001	$F4$	4	3	010	$F8$	8	5
101	$F5$	5	3	110	$F9$	9	5
111	$F6$	6	4	100	$F10$	10	6

регистра микрокоманды. Рисунком 1.33 поясняется формирование диаграмм на примере самой «короткой» ($F3$) и самой «длинной» ($F10$).

Выходы блока управления режимами «Работа», «Приостановка», «Шаг» удобно присоединить к переключателям на передней панели микроЭВМ. Дополнительные RS-триггеров для подавления дребезга контактов не понадобится: они реализованы на кристалле К1804ГГ1. На входы \overline{ST} и \overline{HLT} , $COSC$ и $COSO$ нужно подавать противофазные сигналы с контактов переключателей. При этом, если подать $\overline{HLT} = 0$ ($\overline{ST} = 1$), то ТГ перейдет на режим «Приостановка» в момент, определяемый состоянием входа COS : при $COS = 1$ приостановка наступает в первом микротакте текущего такта ($C1 = C2 = C3 = 1$, $C4 = 0$), при $COS = 0$ — в последнем микротакте ($C1 = C2 = C3 = 0$, $C4 = 1$). Вход можно подсоединить и к постоянному единичному или нулевому потенциалу. При переходе сигналов на входах \overline{HLT} и \overline{ST} в обратное состояние схема ТГ продолжит формирование диаграммы.

Входы $COSO$ и $COSC$ используются для запуска пошагового режима в то время, как ТГ находится в режиме «Приостановка»: после установки сигналов $COSO = 0$, $COSC = 1$ будет сформирована диаграмма одного такта с последующей остановкой в момент, определяемый состоянием входа COS .

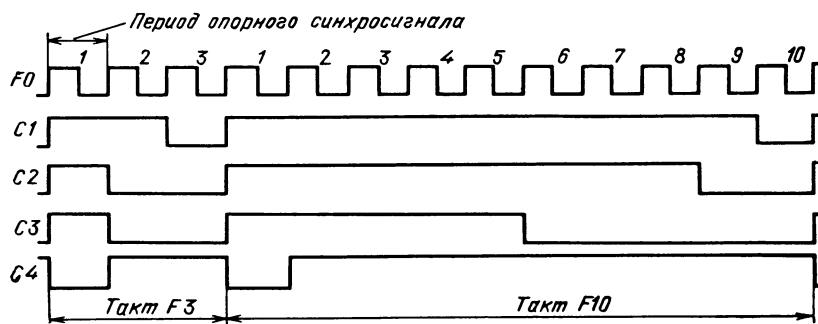


Рис. 1.33. Формирование временных диаграмм $F3$, $F10$

Режим «Ожидание» состоит в динамическом растягивании такта с целью синхронизации центрального процессора с другими (более медленными) частями вычислительной системы, например, для микропрограммного управления вводом-выводом данных через канал от внешних устройств. Отметим, что вход \overline{EWJ} удобно подсоединить к одному из тактовых выходов $C1—C4$.

Назначение выводов микросхемы K1804ГГ1

$C01—C03$ — входы для выбора одной из восьми диаграмм тактовых импульсов на выходах $C1—C4$. Соединяются с разрядами микрокоманды при необходимости управления выбором диаграмм или с постоянными потенциалами, если используется только одна из восьми диаграмм.

$C1—C4$ — выходы тактовых сигналов.

$F0$ — выход внутреннего генератора опорной синхропоследовательности.

$FC1, FC2$ — входы для подключения кварцевого резонатора или иного источника, задающего опорную синхропоследовательность.

\overline{SR} — вход «Запуск», который можно использовать при включении питания или иной инициализации системы: подача на ТГ потенциала $\overline{SR} = 0$ вызывает работу ТГ в этом режиме, независимо от значений поступающих сигналов \overline{HLT} , \overline{ST} , \overline{COSC} , \overline{COSO} , \overline{RQWJ} , \overline{RA} .

\overline{HLT} — вход «Приостановка». При $\overline{HLT} = 0$ ($\overline{ST} = 1$) схема ТГ переходит в режим приостановки в момент, определяемый значением сигнала \overline{COS} .

\overline{ST} — вход «Продолжение». При подаче $\overline{ST} = 0$ ($\overline{HLT} = 1$) схема ТГ возобновляет функционирование в режиме «Работа».

\overline{COS} — вход для указания момента перехода в режим «Приостановка». В совокупности с $\overline{HLT} = 0$, $\overline{ST} = 1$ при $\overline{COS} = 0$ приостановка наступает в момент, когда $C1 = C2 = C3 = 0$, $C4 = 1$ (т. е. на последнем микротакте диаграммы), а при $\overline{COS} = 1$ — в момент, когда $C1 = C2 = C3 = 1$, $C4 = 0$ (т. е. на первом микротакте диаграммы).

\overline{COSC} , \overline{COSO} — входы управления пошаговым режимом. Поступление противофазных сигналов $\overline{COSO} = 0$ ($\overline{COSC} = 1$) во время режима «Приостановка» вызывает однократное формирование диаграммы такта с остановкой в момент, определяемый значением сигнала на входе \overline{COS} .

\overline{RQWJ} — вход «Запрос ожидания». Сигнал $\overline{RQWJ} = 0$ переводит ТГ в режим «Ожидание» на следующем микротакте после того, как на входе \overline{EWJ} будет установлен нулевой (разрешающий) потенциал.

\overline{EWJ} — вход разрешения ($\overline{EWJ} = 0$) перехода схемы ТГ в режим «Ожидание».

\overline{WJ} — выходной сигнал, подтверждающий ($\overline{WJ} = 0$) пребывание ТГ в состоянии «Ожидание».

\overline{RA} — входной сигнал «Готовность» ($\overline{RA} = 0$), предписывающий возобновление режима «Работа» посредством выхода ТГ из режима

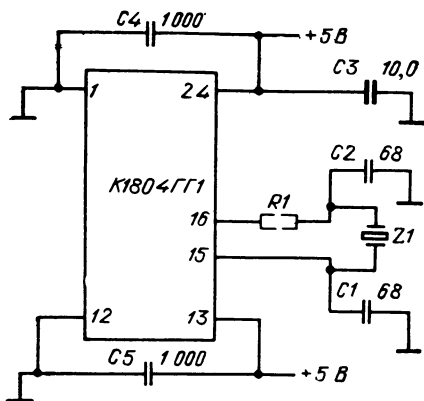


Рис. 1.34. Схема подключения время-задающих элементов

включения он может работать как генератор первой гармоники внешнего кварца, LC -генератор или как передатчик внешнего тактового сигнала. Выход внутреннего генератора $F0$ может быть использован в добавление к тактовым выходам $C1$ — $C4$.

Подключение кварцевого резонатора для получения опорной частоты от 5 до 20 МГц показано на рис. 1.34. По цепям питания рекомендуется ставить развязывающие конденсаторы (показаны на рисунке), а проводные соединения делать как можно короче. На частотах, меньших 5 МГц, подсоединение кварцевого резонатора к выводу 16 микросхемы следует осуществлять через резистор $R1$, что исключит возбуждение генератора на третьей гармонике. Для получения частот, больших 20 МГц, между правой (по схеме) обкладкой конденсатора $C1$ (емкостью более 1000 пФ) и общим проводом следует включить параллельный колебательный контур с резонансной частотой, лежащей между первой и третьей гармониками кварца, чтобы предотвратить возбуждение генератора на первой гармонике. Вместо кварцевого резонатора можно подключить последовательный колебательный контур. Если же входы $FC1$ и $FC2$ соединить через резистор 4,7 кОм, то выработку опорной частоты можно инициировать подачей внешнего сигнала с амплитудой от 1 до 5 В на вход $FC1$ через разделительный конденсатор емкостью 0,01 мкФ.

Из разнообразных применений схемы $K1804GG1$ упомянем только одно, связанное с архитектурой микропроцессорной секции (МПС) $K1804BC2$. Если тактовый выход $C2$ подключить ко входам \overline{IEN} МПС, а тактовый выход $C3$ — ко входам T всех остальных микросхем ($K1804BP2$, $K1804BY4$ и т. д.), то на диаграммах такта $F5$ — $F10$ в МПС $K1804BC2$ можно организовать трехадресные операции. Трехадресность выполнения операции в течение одного такта достигается путем изменения информации на адресном входе ($B3$ — $B0$) МПС после считывания операнда и перед записью результата операции в регистровое запоминающее устройство МПС. При этом используется вход \overline{IEN} МПС. При $\overline{IEN} = 1$ на выходе \overline{W}

«Ожидание». Например, сигнал \overline{RA} может поступать от устройства, внешнего по отношению к процессору, когда это внешнее устройство выполнило заданную функцию и процессор может продолжить работу.

OV — вывод общий ТГ.

$OVGN$ — вывод общий внутреннего генератора.

$+5V$ — вывод питания ТГ.

$+5VGN$ — вывод питания внутреннего генератора.

В заключение остановимся на вопросах использования внутреннего генератора микросхемы $K1804GG1$. При различных

младшей МПС устанавливается «1». Если входы \overline{WE} (разрешение записи) всех МПС соединить с выходом \overline{W} младшей МПС, то подача «1» на вход \overline{IEN} будет запрещать запись во всех МПС. Для тактирования МПС К1804ВС2 удобно использовать выход $C3$ микросхемы К1804ГГ1: этот сигнал остается равным единице приблизительно половину длительности такта. Тогда выход $C2$, который всегда равен нулю на двух последних периодах внутреннего генератора, будет управлять входом \overline{IEN} МПС. Если управлять мультиплексором на входе адреса $B3-B0$ МПС с помощью выхода $C3$, то для длительности такта, большей $F4$, адрес результата поступит на вход адреса МПС прежде, чем начнется запись результата МПС. Тем самым реализуется трехадресный режим работы блока обработки данных, построенного на основе МПС К1804ВС2.

1.6. РЕГИСТРЫ И ПРИЕМОПЕРЕДАТЧИКИ

В дополнение к 4-разрядному регистру с дублированными выходами К1804ИР1 [1] в комплект К1804 включены следующие микросхемы.

Микросхема К1804ИР2 — 8-разрядный параллельный регистр, построенный на триггерах D -типа. Цоколевка выводов корпуса микросхемы представлена на рис. 1.35, а структурная схема регистра — на рис. 1.36. Запись информации в регистр осуществляется по фронту тактового сигнала C при наличии разрешающего нулевого сигнала на входе \overline{EC} (при $\overline{EC} = 1$ запись информации в регистр не производится). При подаче нулевого сигнала на вход сброса \overline{R} производится установка всех разрядов регистра в нуль независимо от значения сигналов на других входах. Тристабильные выходы регистра управляются сигналом на входе разрешения выходных данных \overline{EDY} . При $\overline{EDY} = 1$ выходы данных переходят в состояние высокого сопротивления. Режимы работы регистра перечислены в табл. 1.44. Микросхема К1804ИР2 изготавливается по технологии маломощных ТТЛ схем с диодами Шотки и выпускается в 22-выводном корпусе. Типовое значение потребляемой мощности 0,12 Вт.

Таблица 1.44. Режимы работы микросхемы К1804ИР2

Режим работы	Входы					Выход DY_i
	\overline{EDY}	\overline{R}	\overline{EC}	D_i	C	
Запрещение выходов	1	X	X	X	X	Z
Сброс	1	0	X	X	X	Z
Сброс	0	0	X	X	X	0
Хранение	X	1	1	X	X	Не изменяется
Загрузка	1	1	0	1	Переход из «0» в «1»	Z
Загрузка	0	1	0	0	То же	0
Загрузка	0	1	0	1	»	1

Назначение выводов микросхемы К1804ИР2

\bar{R} — вход сброса.

$DY7-DY0$ — выходы данных, с 7-го по 0-й разряд.

$D7-D0$ — входы данных, с 7-го по 0-й разряд.

C — вход тактовый.

\overline{EDY} — вход разрешения выходных данных.

\overline{EC} — вход разрешения записи в регистр.

OV — общий вывод.

$+5V$ — вывод питания.

Данные о режимах работы микросхемы приведены в табл. 1.44.

Микросхема К1804ИР3 — 8-разрядный параллельный двунаправленный регистр или порт ввода-вывода. Цоколевка выводов микросхемы приведена на рис. 1.37. Микросхема К1804ИР3 изготавливается по технологии маломощных ТТЛ-схем с диодами Шотки и выпускается в

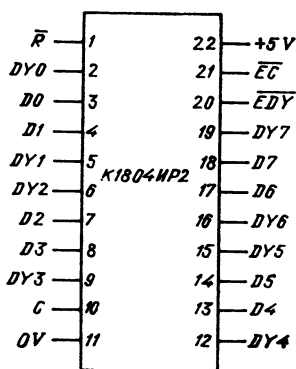


Рис. 1.35. Цоколевка выводов микросхемы К1804ИР2

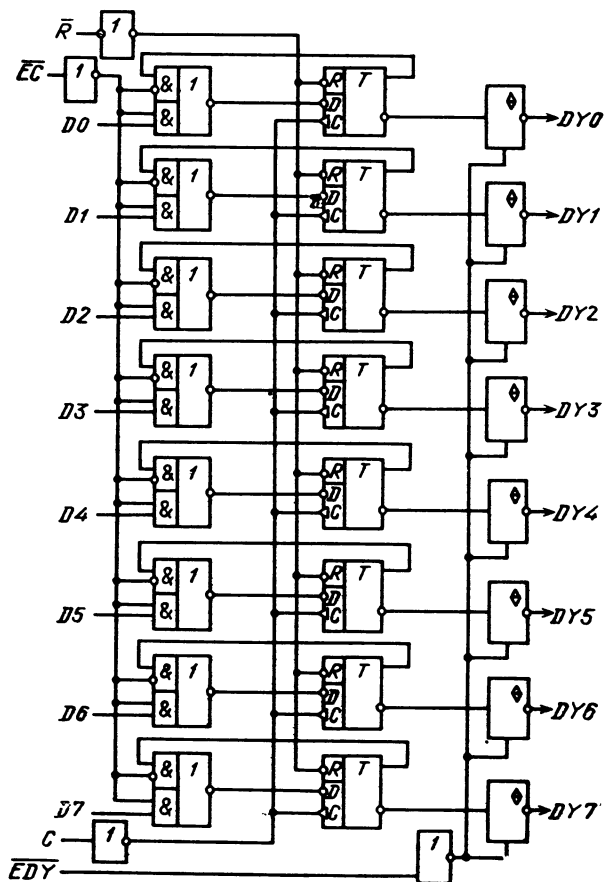


Рис. 1.36. Структурная схема регистра К1804ИР2

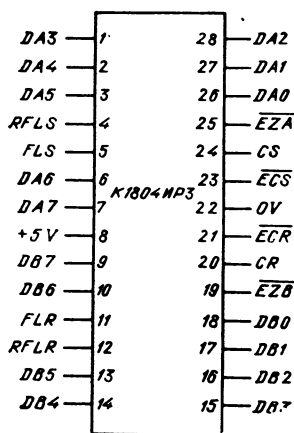


Рис. 1.37. Цоколевка выводов микросхемы К1804ИРЗ

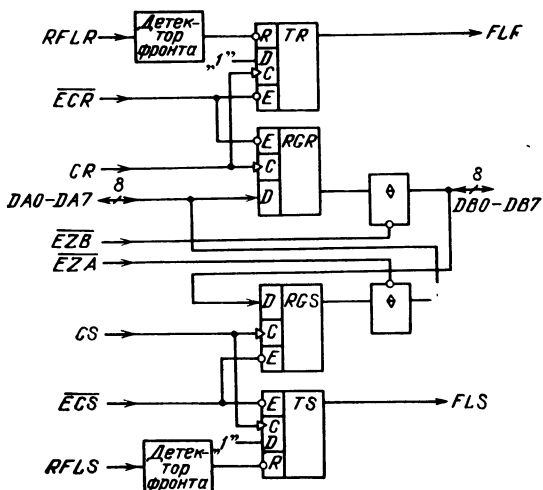


Рис. 1.38. Структурная схема порта К1804ИРЗ

28-выводном корпусе. Типовое значение потребляемой мощности 0,78 Вт.

На структурной схеме (рис. 1.38) основными блоками являются два 8-разрядных регистра (*RGR* и *RGS*), построенные на *D*-триггерах. Запись информации с шины *DA* в *RGR* производится по фронту тактового сигнала *CR* при наличии нулевого сигнала на входе разрешения записи \overline{ECR} (при $\overline{ECR} = 1$ запись в *RGR* запрещена). Информация с выходов *RGR* поступает на двунаправленную шину *DB*, управляемую сигналом на входе \overline{EZB} . При $\overline{EZB} = 0$ шина *DB* является выходной (выход *RGR*), а при $\overline{EZB} = 1$ — входной. В схеме присутствуют два триггера-флажка: *TR* (для *RGR*) и *TS* (для *RGS*). Одновременно с записью информации в *RGR* производится установка в единицу *TR*. Установка в нуль триггера-флажка *TR* осуществляется по фронту сигнала на входе *RFLR*.

Назначение выводов микросхемы К1804ИРЗ

DA6—DA0 — двунаправленные выводы шины *DA*, с 7-го по 0-й разряд.

RFLS — вход установки в нуль триггера *TS*.

FLS — выход триггера *TS*.

+5V — вывод питания.

DB7—DB0 — двунаправленные выводы шины *DB*, с 7-го по 0-й разряд.

FLR — выход триггера *TR*.

RFLR — вход установки в нуль триггера *TR*.

\overline{EZB} — вход разрешения буфера *R*.

\overline{CR} — вход тактовый регистра R .

\overline{ECR} — вход разрешения записи в регистр R .

\overline{OV} — общий вывод.

\overline{ECS} — вход разрешения записи в регистр S .

\overline{CS} — вход тактовый регистра S .

\overline{EZA} — вход разрешения буфера S .

Микросхема К1804ВА1 — 4-разрядный каналный приемопередатчик с открытым коллектором и мультиплексором на входе регистра передатчика. Изготавливается по технологии маломощных ТТЛ-схем с диодами Шоттки и выпускается в 24-выводном корпусе. Типовое значение потребляемой мощности 0,35 Вт. Цоколевка выводов корпуса микросхемы представлена на рис. 1.39, а структурная схема — на рис. 1.40. Режимы работы микросхемы К1804ВА1 приведены в табл. 1.45. Информация на вход регистра-передатчика $RG1$ при нулевом сигнале на входе \overline{SED} поступает с шины \overline{DA} , а при единичном — с шины \overline{DB} . Запись в $RG1$ происходит по фронту сигнала \overline{C} . С выходов регистра-передатчика при нулевом сигнале на входе \overline{EZB} информация передается в инвертированном виде на двунаправленную шину \overline{B} . Если же $\overline{EZB} = 1$, то шина \overline{B} становится входной и информация с нее поступает на вход регистра-приемника $RG2$. При нулевом сигнале на входе \overline{EWRRC} информация с шины \overline{B} проходит на выходы регистра приемника, а при $\overline{EWRRC} = 1$ на выходах регистра-приемника сохраняется информация, которая была на его входах в момент перехода сигнала на входе \overline{EWRRC} из нуля в единицу. С выходов регистра-приемника при нулевом сигнале на входе \overline{EZDR} информация поступает в инвертированном виде на тристабильную шину \overline{DR} . При $\overline{EZDR} = 1$ шина \overline{DR} находится в третьем состоянии (состоянии высокого сопротивления).

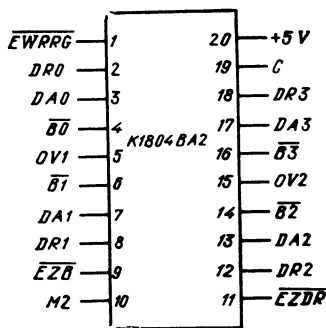


Рис. 1.39. Цоколевка выводов микросхемы К1804ВА1

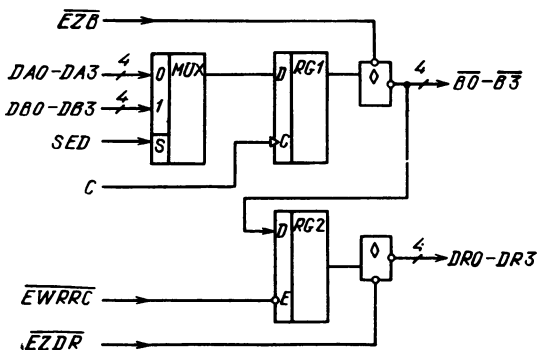


Рис. 1.40. Структурная схема приемопередатчика К1804ВА1

Таблица 1.45. Режимы работы микросхемы К1804ВА1

Входы							Внутренние сигналы		Вход-выход B_i	Выход DR_i	Режим работы
\overline{SED}	\overline{DA}	$\overline{DB_i}$	\overline{C}	\overline{EZB}	\overline{EWRR}	\overline{EZDR}	D_i	Q_i			
X	X	X	X	1	X	X	X	X	1	X	Запрещение выходов передатчика
X	X	X	X	X	X	1	X	X	X	Z	Запрещение выходов приемника
X X	X X	X X	X X	1 1	0 0	0 0	X X	0 1	0 1	1 0	Прием данных из канала
0 0 1 1	0 1 X X	X X 0 1	Переход из «0» в «1»	X X X X	X X X X	X X X X	0 1 0 1	X X X X	X X X X	X X X X	Загрузка регистра передатчика
X X	X X	X X	0 1	X X	X X	X X	Не изменяется	X X	X X	X X	Хранение данных в регистре передатчика
X X	X X	X X	X X	0 0	X X	X X	0 1	X X	1 0	X X	Передача данных в канал
X	X	X	X	X	1	X	X	Не изменяется	X	X	Запрещение приема данных

Назначение выводов микросхемы К1804ВА1

\overline{EWRR} — вход разрешения записи в регистр приемника.

$DR3$ — $DR0$ — выходы приемника, с 3-го по 0-й разряд.

$DB3$ — $DB0$ — входы DB передатчика, с 3-го по 0-й разряд.

$DA3$ — $DA0$ — входы DA передатчика, с 3-го по 0-й разряд.

$\overline{B3}$ — $\overline{B0}$ — двунаправленные выходы приемника-передатчика, с 3-го по 0-й разряд.

$\overline{OV1}$ — общий вывод 1.

\overline{EZB} — вход управления двунаправленным выводом.

\overline{EZDR} — вход разрешения выходов приемника.

SED — вход управления мультиплексором передатчика.

$OV2$ — общий вывод 2.

C — тактовый вход.

$+5V$ — вывод питания.

Микросхема К1804ВА2 — 4-разрядный канальный приемопередатчик с открытым коллектором и логикой контроля четности. Изготавливается по технологии маломощных ТТЛ-схем с диодами Шотки и выпускается в 20-выводном корпусе. Типовое значение потребляемой мощности 0,375 Вт. Цоколевка выводов корпуса микросхемы представлена на рис. 1.41. Структурная схема показана на рис. 1.42. Работа микросхемы К1804ВА2 аналогична работе микросхемы К1804ВА1 (см. табл. 1.31). Отличием является отсутствие мультиплексора на входе регистра передатчика и наличие логики контроля четности. При нулевом сигнале на входе \overline{EZB} на выходе контроля четности вырабатывается значение $M2 = DA0 \oplus DA1 \oplus DA2 \oplus DA3$, а при $\overline{EZB} = 1$ значение $M2 = Q0 \oplus Q1 \oplus Q2 \oplus Q3$.

Назначение выводов микросхемы К1804ВА2

\overline{EWRRG} — вход разрешения записи в регистр приемника.

$DR3$ — $DR0$ — выходы приемника, с 3-го по 0-й разряд.

$DA3$ — $DA0$ — входы передатчика, с 3-го по 0-й разряд.

$\overline{B3}$ — $\overline{B0}$ — двунаправленные выходы приемника-передатчика, с 3-го по 0-й разряд.

$OV1$ — общий вывод 1.

\overline{EZB} — вход управления двунаправленным выводом.

$M2$ — выход контроля четности.

\overline{EZDR} — вход разрешения выхода приемника.

$OV2$ — общий вывод 2.

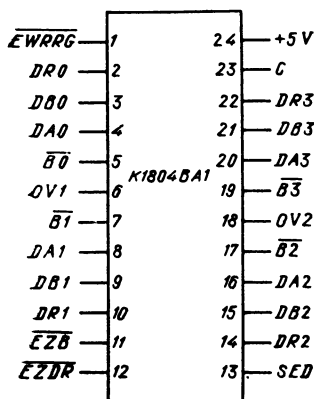


Рис. 1.41. Цоколевка выводов микросхемы К1804ВА2

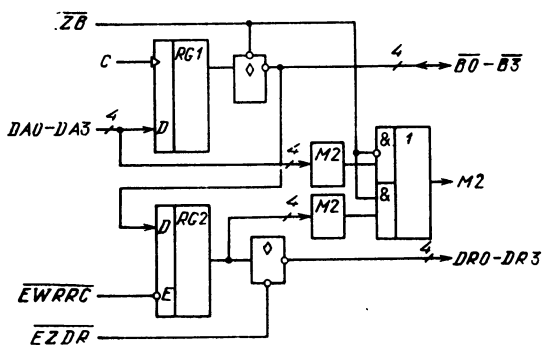


Рис. 1.42. Структурная схема приемопередатчика К1804ВА2

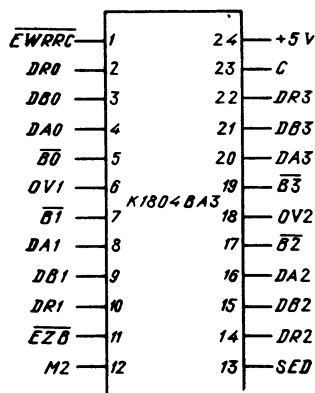


Рис. 1.43. Цоколевка выводов микросхемы K1804BA3

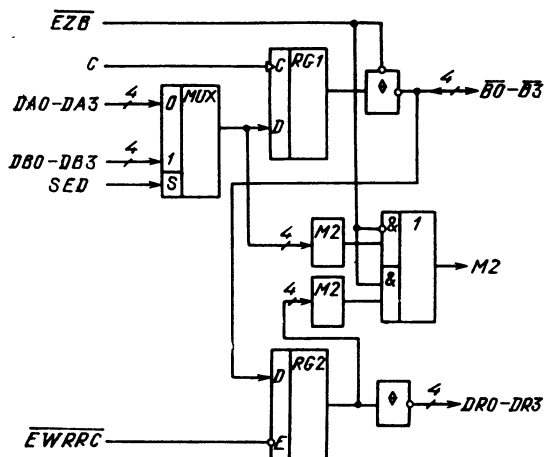


Рис. 1.44. Структурная схема приемопередатчика K1804BA3

C — входной тактовый.

$+5V$ — вывод питания.

Микросхема K1804BA3 — 4-разрядный каналный приемопередатчик с трехстабильными выходами на шину, мультиплексором на входе и логикой контроля четности. Изготавливается по технологии маломощных ТТЛ-схем с диодами Шотки и выпускается в 24-выводном корпусе. Типовое значение потребляемой мощности 0,375 Вт. Цоколевка выводов корпуса микросхемы K1804BA3 приведена на рис. 1.43, структурная схема — на рис. 1.44. Работа микросхемы K1804BA3 аналогична работе микросхемы K1804BA1 (см. табл. 1.45). Отличием является отсутствие буфера с тремя состояниями на выходе регистра приемника и наличие логики контроля четности. При нулевом сигнале на входе \overline{EZB} на выходе контроля четности вырабатывается значение $M2 = MUX0 \oplus MUX1 \oplus MUX2 \oplus MUX3$, а при $\overline{EZB} = 1$ значение $M2 = Q0 \oplus Q1 \oplus Q2 \oplus Q3$.

Назначение выводов микросхемы K1804BA3

\overline{EWRRC} — вход разрешения записи в регистр приемника.

$DR3-DR0$ — выходы DR приемника, с 3-го по 0-й разряд.

$DB3-DB0$ — входы DB передатчика, с 3-го по 0-й разряд.

$DA3-DA0$ — входы DA передатчика, с 3-го по 0-й разряд.

$\overline{B3}-\overline{B0}$ — двунаправленные выходы приемника-передатчика, с 3-го по 0-й разряд.

$OV1$ — общий вывод 1.

\overline{EZB} — вход управления двунаправленными выводами приемника-передатчика.

$M2$ — выход контроля четности.

SED — вход управления мультиплексором передатчика.
OV2 — общий вывод 2.
C — вход тактовый.
+5V — вывод питания.

Глава 2.

МИКРОПРОГРАММНОЕ УПРАВЛЕНИЕ: СХЕМНЫЕ РЕШЕНИЯ

2.1. ТИПОВЫЕ СХЕМЫ МИКРОПРОГРАММНОГО УПРАВЛЕНИЯ

В данном параграфе рассмотрены типовые варианты организации микропрограммного управления в микропроцессорах, элементной базой которых служит комплект БИС К1804. Очень многое из того, что относится к комплекту К1804, справедливо и для других комплектов микропрограммируемых БИС, поэтому при изложении материала, где это возможно, использована некоторая обобщенная символическая форма. Цель введения символики — упрощение изложения за счет абстрагирования от несущественных деталей конкретных реализаций.

Основой всех рассматриваемых схемных решений является принцип микропрограммного управления. Как упоминалось во введении, БМУ включает МПП, ФАМ и регистр микрокоманд (РгМК). Важнейшим (но не единственным) элементом ФАМ в большинстве современных разработок является БИС К1804ВУ4. Она выполнена в 40-выводном корпусе, потребляет ток около 200 мА от источника +5 В и выполняет 16 наиболее типичных операций по управлению последовательностью микрокоманд. Краткая информация о БИС К1804ВУ4, приведенная ниже, вполне достаточна для того, чтобы освежить в памяти особенности данной микросхемы. Познакомиться же с ней впервые удобнее всего по книге [4], проводя аналогию с микросхемой *Am2910*, описание которой сделано с полнотой, достаточной для самого взыскательного читателя. Подойдет также [1], где, кстати, представлена и цоколевка выводов корпуса микросхемы К1804ВУ4.

Структурная схема БИС К1804ВУ4 приведена на рис. 2.1, а набор инструкций дан в табл. 2.1. Перед освоением системы инструкций целесообразно вспомнить назначение основных структурных элементов микросхемы.

Четырехвходовый мультиплексор предназначен для выбора в качестве источника адреса следующей микрокоманды содержимого регистра адреса/счетчика (РгА/Сч) или внешнего входа адреса *D11 — D0*, или счетчика микрокоманд, или верхушки стека в зависимости от значений сигналов инструкции *10—13* и, возможно, сигналов условия (*СС*) и разрешения условия (*ССЕ*). Выбранный мульт-

Таблица 2.1. Инструкции микросхемы К1804ВУ4

Инструкция			Текущее значение PгA/Cч	Значение условия				Операция над PгA/Cч	Активный разрешающий выход
Код	Мнемоника	Операция		FALSE		TRUE			
				Источник адреса	Стек	Источник адреса	Стек		
0000	JZ	Переход к нулевому адресу	X	«0»	Очистка	«0»	Очистка	Хранение	\overline{PE}
0001	CJS	Условный переход к подпрограмме	X	СМК	Хранение	D	Загрузка	»	\overline{PE}
0010	JMAP	Переход к начальному адресу микропрограммы	X	D	»	D	Хранение	»	\overline{ME}
0011	CJP	Условный переход по адресу из PгМК	X	СМК	»	D	»	»	\overline{PE}
0100	PUSH	Засылка в стек и загрузка счетчика по условию	X	СМК	Загрузка	СМК	Загрузка	Загрузка по условию	\overline{PE}
0101	JSRP	Условный переход к одной из двух подпрограмм	X	PчA/Cч	»	D	»	Хранение	\overline{PE}
0110	CJV	Условный переход по вектору прерывания	X	СМК	Хранение	D	Хранение	»	\overline{VE}
0111	JRP	Условный переход по адресу из PгA/Cч или PгМК	X	PгA/Cч	»	D	»	»	\overline{PE}

Инструкция			Текущее значение P-A/Cч	Значение условия				Операции над P-A/Cч	Активный разрешающий выход
Код	Мнемоника	Операция		FALSE		TRUE			
				Источник адреса	Стек	Источник адреса	Стек		
1000	RFCT	Повторение цикла, пока счетчик не равен нулю	≠0 =0	Стек СМК	Хранение Выталкивание	Стек СМК	Хранение Выталкивание	\overline{PE} \overline{PE} Уменьшение Хранение	
1001	RPCT	Переход по адресу из PгМК, пока счетчик не равен нулю	≠0 =0	D СМК	Хранение »	D СМК	Хранение »	\overline{PE} \overline{PE} Уменьшение Хранение	
1010	CRTN	Условный возврат из подпрограммы	X	»	»	Стек	Выталкивание	\overline{PE} »	
1011	CJPP	Условный переход по адресу из PгМК и выталкивание из стека	X	»	»	D	»	\overline{PE} »	
1100	LDCT	Загрузка счетчика и продолжение	X	»	»	СМК	Хранение	\overline{PE} Загрузка	
1101	LOOP	Выход из цикла по условию	X	Стек	»	СМК	Выталкивание	\overline{PE} Хранение	
1110	CONT	Последовательное продолжение	X	СМК	»	СМК	Хранение	\overline{PE} »	
1111	TWVB	Ветвление на три направления	≠0 =0	Стек D	» Выталкивание	СМК СМК	Выталкивание »	\overline{PE} \overline{PE} Уменьшение Хранение	

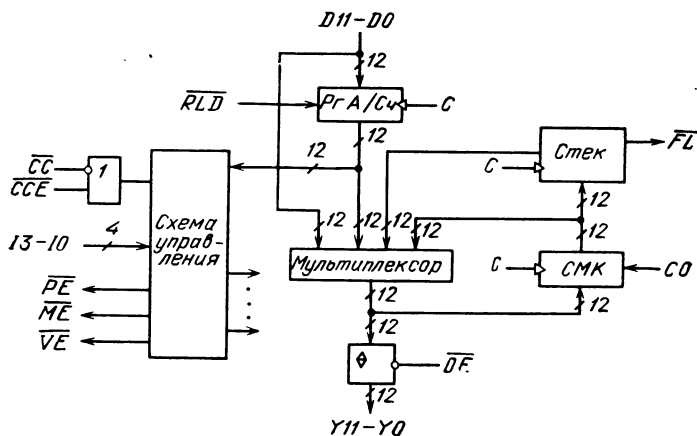


Рис. 2.1. Структурная схема БИС К1804ВУ4

типлексором адрес поступает через буферный элемент на выходную тристабильную шину Y . При наличии нулевого потенциала на входе разрешения выбора адреса (\overline{OE}) разрешается вывод адреса через шину Y . При $\overline{OE} = 1$ шина Y отключена (находится в состоянии высокого сопротивления).

Регистр адреса/счетчик состоит из двенадцати триггеров D -типа, запись информации в которые производится по фронту (переходу из «0» в «1») тактового сигнала C при выполнении соответствующих инструкций или $\overline{RLD} = 0$. В зависимости от выполняемой инструкции РгА/Сч может использоваться либо в качестве буфера для записи и хранения адреса или числа циклов, принимаемых от внешнего источника через шину D , либо в качестве счетчика циклов, содержимое которого на каждом такте уменьшается на 1 до установки счетчика в нуль. Если в РгА/Сч загружено число N , то при выполнении соответствующей инструкции цикл повторится $N + 1$ раз. Настройка РгА/Сч на тот или иной режим работы происходит под действием внутренних управляющих сигналов, вырабатываемых схемой управления.

Счетчик микрокоманд (СМК) содержит регистр и схему приращения — инкрементор. Любой текущий адрес с выхода мультиплексора передается через инкрементор в регистр. Этот регистр состоит из двенадцати триггеров D -типа, запись информации в которые производится по фронту тактового сигнала C . Адрес с выхода регистра поступает на вход мультиплексора и в стек.

Схема приращения имеет вход переноса $C0$. При $C0 = 0$ адрес с выхода мультиплексора передается через схему приращения немодифицированным. При $C0 = 1$ происходит увеличение на единицу адреса, передаваемого с выхода мультиплексора. Схема приращения не вырабатывает сигнал выходного переноса, что является препятствием для наращивания микросхем, подобных К1804ВУ2, однако на практике подавляющее большинство микропрограмм содержит не более 4096 микрокоманд. В крайнем случае можно использовать страничную организацию памяти, разбив микропрограмму на блоки по 4096 микрокоманд.

Основными частями стека являются указатель стека (реверсивный счетчик со специальными цепями установки) и накопитель стека. Накопитель стека представляет собой ОЗУ емкостью пять 12-разрядных слов. Запись и считывание информации осуществляются для той ячейки накопителя, которую адресует указатель стека. Эта ячейка называется вершиной стека. Информация может считываться из вершины стека на вход мультиплексора адреса как без изменения содержимого стека, так и с изменением. Информация записывается в стек при выполнении операции *PUSH*; при этом сначала содержимое указателя стека увели-

чивается на 1, а затем происходит собственно запись. Таким образом, указатель стека всегда указывает на последнее записанное слово. В стек может записываться только содержимое счетчика микрокоманд.

При выполнении над стеком операции *POP* содержимое указателя уменьшается на единицу после чтения вершины стека. Логически это означает заполнение вершины стека другой, ранее записанной при операции *PUSH* информацией. Операция *POP* над пустым стеком не изменяет его состояния. Заполнение всех пяти уровней стека вызывает появление осведомительного сигнала $\overline{FL} = 0$. При выполнении очередной операции *PUSH* указатель стека остается без изменения, а в вершину стека записывается новая информация. Старая информация при этом теряется. Такая потеря является очень нежелательной, и рекомендуется тщательно продумать микропрограммы, чтобы ее не допустить.

Схема управления БИС K1804BY4 преобразует внешние управляющие сигналы ($I3$ — $I0$, \overline{CC} , \overline{CCE}) и внутренний управляющий сигнал — признак нулевого содержимого $PgA/Cч$ — в набор управляющих сигналов для всех элементов микросхемы. Кроме того, схема управления вырабатывает три внешних управляющих сигнала (\overline{PE} , \overline{ME} , \overline{VE}), которые используются для разрешения одного из трех внешних источников информации, подключенных в типовой конфигурации ко входу D : $PgMK$, дешифратора начальных адресов и ПЗУ векторов прерывания. Сигналы \overline{CCE} и \overline{CC} имеют активный нулевой уровень. При $\overline{CCE} = 0$ разрешен анализ значения условия, поступающего на вход \overline{CC} . Если $\overline{CCE} = 1$, то условие тождественно истинно (*TRUE*). Если $\overline{CCE} = 0$ (т. е. анализ условия разрешен), то наличие $\overline{CC} = 0$ эквивалентно логическому «да» (*TRUE*), а наличие $\overline{CC} = 1$ — логическому «нет» (*FALSE*).

Сделаем несколько замечаний о микросхемах K1804BY1, K1804BY2 и K1804BY3, предназначенных для выполнения функций адресации микропрограммной памяти, подобной K1804BY4. Часто можно слышать утверждение о том, что БИС K1804BY4 просто объединяет в себе три микросхемы K1804BY2 и одну K1804BY3. Это не совсем верно. Двумя наиболее существенными отличиями являются режимы функционирования стека и регистра адреса. Последний в микросхеме K1804BY4 может служить в качестве вычитающего счетчика для подсчета числа циклов, тогда как в K1804BY1 и K1804BY2 он является обычным регистром, а для подсчета циклов необходимо использовать дополнительные внешние схемы на дискретной логике. Особенности стеков (кроме различия в числе уровней) состоят в следующем: указатель стека в K1804BY1 и K184BY2 работает как обычный реверсивный счетчик, а в K1804BY4 изменение содержимого счетчика блокируется при попытке занести слово в заполненный стек или вытолкнуть слово из пустого стека. Стоит обратить внимание и на другие отличия, например, в инструкциях с кодом 1111 и в активных уровнях сигналов условий \overline{CC} и *TEST*, анализируемых микросхемами K1804BY4 и K1804BY3 соответственно.

В подавляющем большинстве практических приложений емкость микропрограммной памяти не превышает 4096 слов, поэтому БИС K1804BY4 удовлетворит почти всех. На нее мы и будем ориентироваться в последующем изложении, когда возникает необходимость в конкретных примерах. В целом же рассматриваемые схемные решения обладают значительной общностью и могут быть интерпретированы при построении БМУ на основе различных БИС подобного назначения. Характеристики этих БИС, а также основы микропрограммного управления (способы адресации, управляющие конструкции) изложены в гл. 3 книги [1].

Итак, один из основных элементов ФАМ — схема формирования последовательности адресов микрокоманд (например, K1804BY4). Как правило, ФАМ включает два-три дополнительных элемента

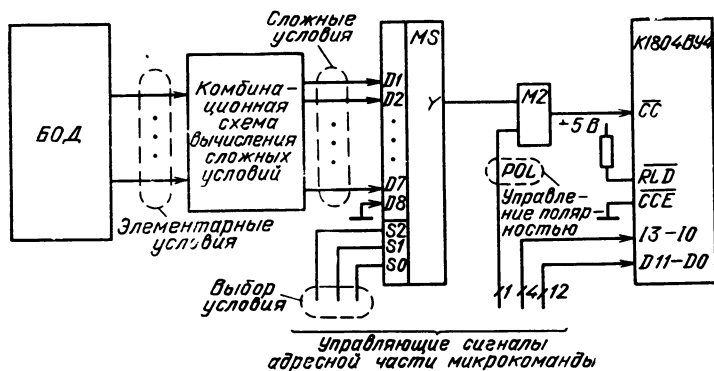


Рис. 2.2. Вспомогательные узлы формирователя адресов микрокоманд

(рис. 2.2). Так, для выполнения условных переходов по значениям нескольких условий (перенос, знак, нуль и т. д.) необходимо организовать подключение соответствующей линии ко входу \overline{CC} микросхемы К1804ВУ4. Очевидно, эту функцию с успехом выполнит мультиплексор. Отметим, что один из свободных информационных входов мультиплексора удобно подключить к нулевому потенциалу для превращения условных переходов в безусловные без выделения разряда микрокоманды на управление входом \overline{CCE} , служащим той же цели. В этом случае \overline{CCE} необходимо соединить с нулевым потенциалом, что повлечет зависимость всех условных инструкций только от значения на входе \overline{CC} .

Для того чтобы условные переходы можно было осуществлять как по прямому, так и по инверсному значениям условия, выход мультиплексора соединяется со входом \overline{CC} микросхемы К1804ВУ4 не непосредственно, а через сумматор по модулю 2 (см. рис. 2.2.) На второй вход этого сумматора поступает из микрокоманды сигнал управления полярностью условия (POL), позволяющий либо инвертировать значение условия, либо передавать его неизменным (табл. 2.2). Наконец, бывает необходимо осуществлять переходы по значениям не только

Таблица 2.2. Управление полярностью условия

Y	POL	\overline{CC}	Комментарии	Y	POL	\overline{CC}	Комментарии
0 ($TRUE$)	0	0 ($TRUE$)	Переход по Y	0 ($TRUE$)	1	1 ($FALSE$)	Переход по \overline{Y}
1 ($FALSE$)	0	1 ($FALSE$)	«	1 ($FALSE$)	1	0 ($TRUE$)	«

Таблица 2.3. Формирование условия

Соотношение	Для чисел без знака	Для чисел со знаком
$A = B$	$Z = 1$	$Z = 1$
$A \neq B$	$Z = 0$	$Z = 0$
$A \geq B$	$C = 1$ (нет заема при вычитании)	$OVR \oplus N = 0$
$A < B$	$C = 0$ (заем при вычитании)	$OVR \oplus N = 1$
$A > B$	$\bar{C} \vee Z = 0$	$(OVR \oplus N) \vee Z = 0$
$A \leq B$	$\bar{C} \vee Z = 1$	$(OVR \oplus N) \vee Z = 1$

элементарных, но и более сложных условий, являющихся булевыми функциями элементарных условий. Например, если элементарными условиями считать флажки нуля (Z), переноса (C), переполнения (OVR) и знака ($F3$), вырабатываемые микропроцессорными секциями K1804BC1 и K1804BC2, то для выполнения одноктактных переходов по сложным условиям ($A \geq B$, $A > B$ и т. д.) придется построить комбинационную схему, реализующую систему булевых функций, перечисленных в табл. 2.3. Предполагается, что тестированию условий предшествует операция вычитания $A - B$, где A и B — числа со знаком в дополнительном коде или числа без знака.

Применение БИС K1804BP2 позволяет избежать введения дополнительных аппаратурных затрат на реализацию перечисленных выше

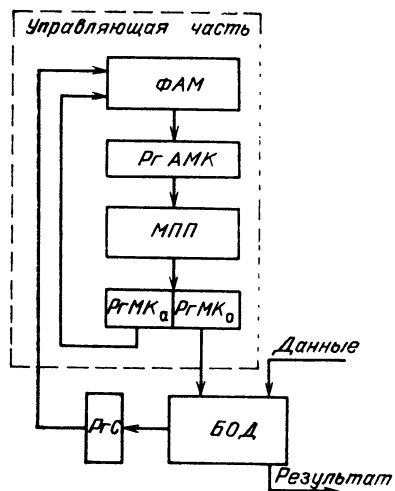


Рис. 2.3. Структура микропрограммного контроллера

схем, если в качестве элементарных условий используются только C , OVR , N ($F3$), Z . Значения элементарных условий могут поступать как непосредственно с БОД, так и с регистра состояния ($РгС$). Кстати, последний также может быть реализован на БИС K1804BP2, если число одновременно загружаемых элементарных условий не превышает четырех.

Длительное (более одного такта) хранение значений условий — не единственное назначение $РгС$. Он используется и для организации конвейерной обработки микрокоманд, о чем пойдет речь ниже.

В структуре микропрограммного контроллера удобно выделить три крупных элемента: БОД, ФАМ и МПП. Такой уровень детализации позволяет придать по-

следующему рассмотрению общность, не зависящую от конкретных схем БОД, ФАМ и МПП, построение которых определяется особенностями конкретных приложений. В трактах передачи информации между ФАМ, МПП и БОД могут находиться регистры (см. рис. 2.3): адреса микрокоманд ($R_{ГМ\bar{K}}$), операционного поля микрокоманды ($R_{ГМ\bar{K}_0}$), адресного поля микрокоманды ($R_{ГМ\bar{K}_a}$) и $R_{ГC}$. В зависимости от наличия или отсутствия перечисленных регистров образуются одиннадцать работоспособных структур (табл. 2.4). Остальные пять из шестнадцати возможных комбинаций наличия и отсутствия четырех регистров неработоспособны из-за возникновения эффекта гонок (соstryзаний) сигналов. Каждая из одиннадцати работоспособных структур обладает своими характеристиками — быстродействием, аппаратными затратами и т. д. Различным структурам соответствуют различные способы построения микропрограмм и различные возможности по управлению длительностью такта с целью повышения быстродействия. Например, «длинный» такт при условных переходах и «короткий» при безусловных реализуется в структурах S_1 , S_2 , S_6 и т. д.

Таблица 2.4. Типовые структуры и их оценка

Структура S_r	Расстановка регистров				Полная длительность такта t_r	Сокращенная длительность такта t_r^*	Число тактов	
	$R_{ГМ\bar{K}}$	$R_{ГМ\bar{K}_a}$	$R_{ГМ\bar{K}_0}$	$R_{ГC}$			при $УП_1$	при $УП_2$
S_1	+	—	—	—	$t_{МПП} + t_{БОД} + t_{ФАМ}$	$t_{МПП} + \max\{t_{БОД}, t_{ФАМ}\}$	1	1
S_2	—	+	+	—	$t_{БОД} + t_{ФАМ} + t_{МПП}$	$\max\{t_{БОД}, t_{ФАМ} + t_{МПП}\}$	1	1
S_3	—	+	—	+	$t_{ФАМ} + t_{МПП} + t_{БОД}$	Нет	2	1
S_4	—	+	+	+	$\max\{t_{БОД}, t_{ФАМ} + t_{МПП}\}$	«	2	1
S_5	+	—	—	+	$t_{МПП} + \max\{t_{БОД}, t_{ФАМ}\}$	«	2	2
S_6	+	+	+	—	$\max\{t_{МПП}, t_{БОД} + t_{ФАМ}\}$	$\max\{t_{БОД}, t_{ФАМ}, t_{МПП}\}$	2	2
S_7	+	+	+	+	$\max\{t_{БОД}, t_{ФАМ}, t_{МПП}\}$	Нет	3	2
S_8	+	—	+	—	$t_{ФАМ} + \max\{t_{БОД}, t_{МПП}\}$	$\max\{t_{МПП} + t_{ФАМ}, t_{БОД}\}$	2	1
S_9	+	—	+	+	$\max\{t_{МПП} + t_{ФАМ}, t_{БОД}\}$	Нет	3	1
S_{10}	+	+	—	—	$t_{МПП} + t_{БОД} + t_{ФАМ}$	$\max\{t_{ФАМ}, t_{МПП} + t_{БОД}\}$	2	2
S_{11}	+	+	—	+	$\max\{t_{ФАМ}, t_{МПП} + t_{БОД}\}$	Нет	2	2

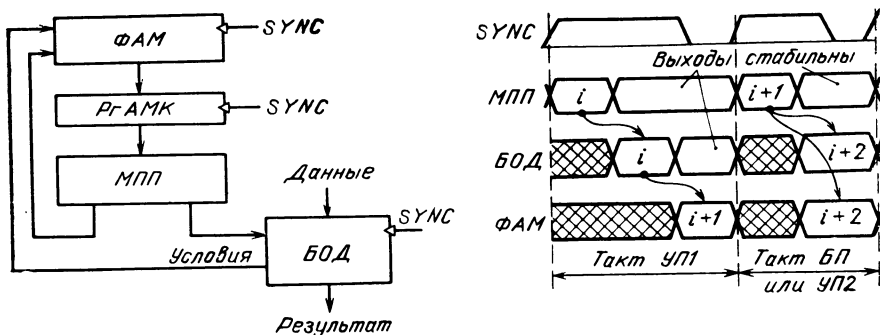


Рис. 2.4. Структура S_1 и ее временные диаграммы

Обработка каждой микрокоманды включает три этапа: формирование адреса микрокоманды, выборку ее и исполнение (соответственно в ФАМ, МПП и БОД). Обработка микрокоманд называется конвейерной, если имеет место временное совмещение нескольких или всех перечисленных этапов. В противном случае обработка микрокоманд называется последовательной. Так же называются и типы структур микроконтроллера — конвейерные или последовательные в зависимости от того, какой способ обработки микрокоманд на них реализован.

Структуры S_1 — S_3 , S_{10} являются последовательными, структуры S_4 — S_9 и S_{11} — конвейерными, причем в структуре S_7 реализован трехуровневый конвейер микрокоманд (максимальное распараллеливание действий ОЧ, ФАМ, МПП), а в остальных — двухуровневый конвейер. Примеры нескольких структур и их временные диаграммы показаны на рис. 2.4 — 2.9. Символами i , $i+1$, $i+2$ обозначены порядковые номера микрокоманд, этапы обработки которых реализуются в соответствующем интервале времени, а заштрихованные интервалы означают неопределенное или безразличное состояние, связанное с переходными процессами. Построение временных диаграмм начинается с фронта синхросигнала $SYNC$, когда информация заносится в РгМК, РгАМК и РгС (если они есть в рассматриваемой структуре). Например, в структуре S_1 (см. рис. 2.4) адрес i -й микрокоманды заносится в РгАМК, затем производится выборка ее из МПП, после чего операционное поле микрокоманды инициирует требуемую операцию в БОД, а адресное — в ФАМ. Если операция в ФАМ является условным переходом по признаку, вырабатываемому на текущем такте в БОД, то адрес следующей $(i+1)$ -й микрокоманды может вычисляться только по готовности этого

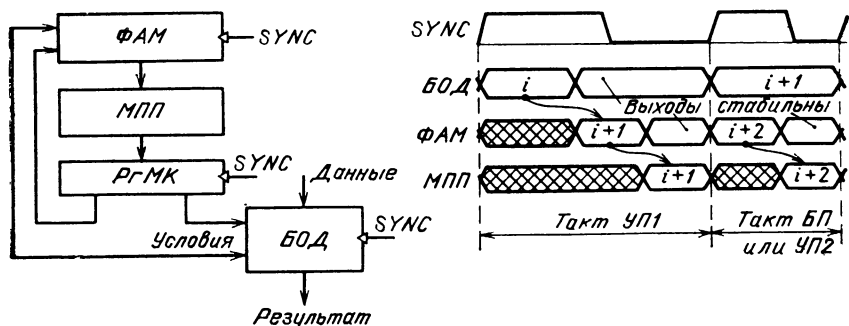


Рис. 2.5. Структура S_2 и ее временные диаграммы

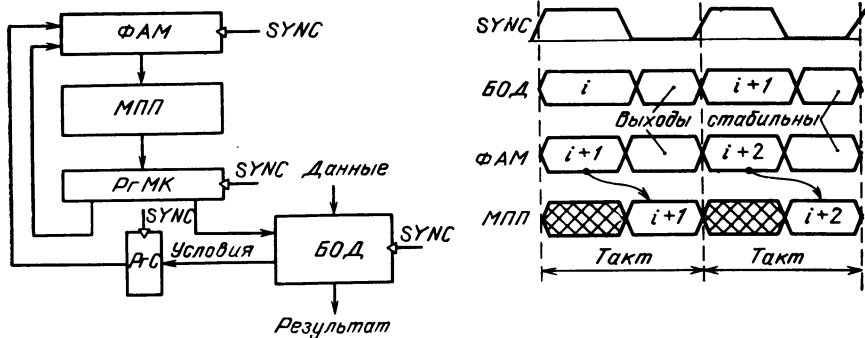


Рис. 2.6. Структура S_4 и ее временные диаграммы

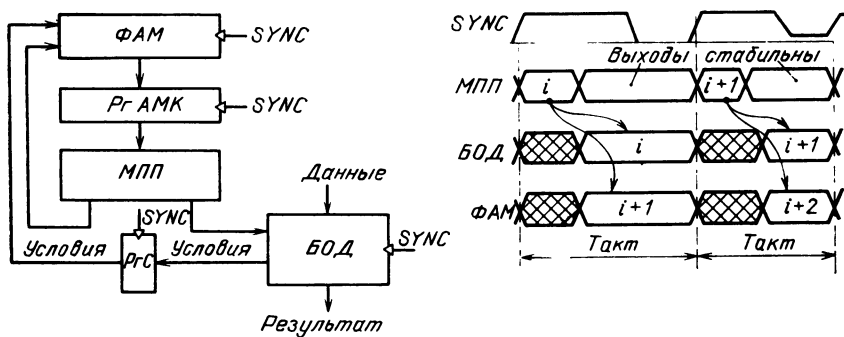


Рис. 2.7. Структура S_5 и ее временные диаграммы

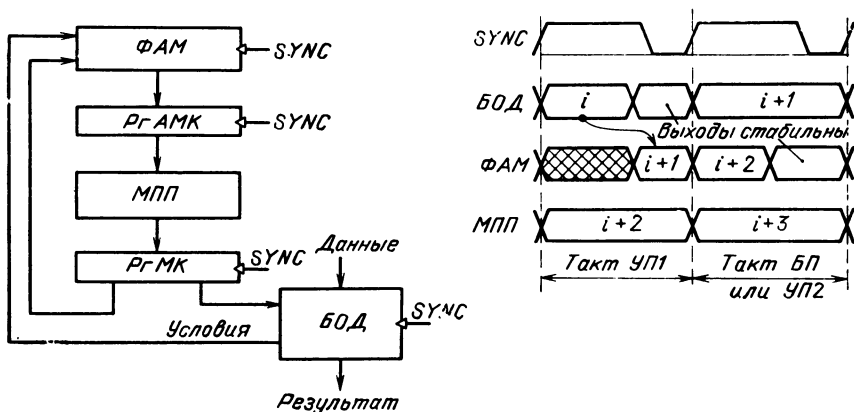


Рис. 2.8. Структура S_6 и ее временные диаграммы

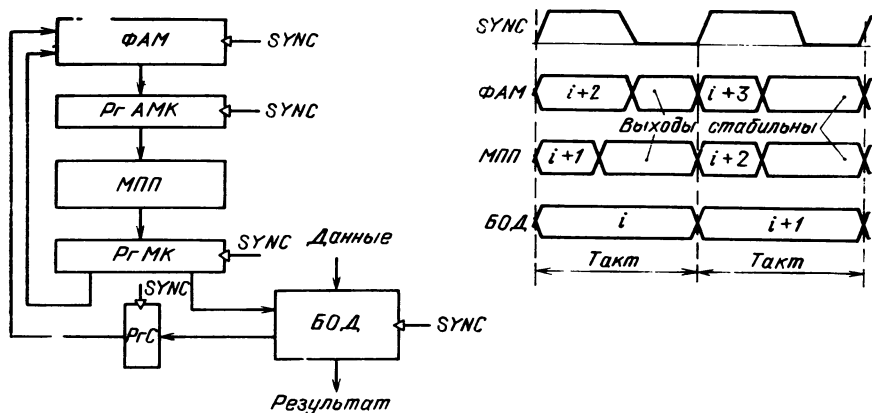


Рис. 2.9. Структура S_7 и ее временные диаграммы

признака. Следовательно, длительность такта можно оценить как $t_{\text{МПП}} + t_{\text{БОД}} + t_{\text{ФАМ}}$, где $t_{\text{БОД}}$, $t_{\text{ФАМ}}$, $t_{\text{МПП}}$ — максимальные задержки распространения сигналов в соответствующих компонентах структуры микроконтроллера. Ситуация, когда тестируется условие, сформированное непосредственно предшествующей операцией в БОД, называется условным переходом первого типа УП1. Существуют и условные переходы второго типа УП2, когда тестируется условие, ранее сформированное в БОД и хранимое в РгС до момента тестирования. Очевидно, при УП2 нет необходимости ожидать готовности условия, т. е. действия в ФАМ можно начать одновременно с действиями в БОД, а значит, и закончить их раньше. Это позволяет сократить длительность такта при выполнении УП2 и безусловных переходов (БП), т. е. увеличить быстродействие микроконтроллера (см. рис. 2.4).

В конвейерной структуре S_4 (см. рис. 2.6) длительность такта одинакова для БП, УП1 и УП2. Она оценивается как $\max \{t_{\text{БОД}}, t_{\text{ФАМ}} + t_{\text{МПП}}\}$. Зато на УП1 требуются два такта вместо одного: на первом такте условие вычисляется в БОД и попадает в РгС, на втором — в ФАМ выполняется условный переход по значению условия, получаемому из РгС.

Структура S_7 (см. рис. 2.9) характеризуется минимальной длительностью такта — $\max \{t_{\text{БОД}}, t_{\text{ФАМ}}, t_{\text{МПП}}\}$, но максимальным числом тактов при выполнении УП1 (три такта). Кстати, структуры S_6 и S_7 более сложны и для микропрограммирования: логика переходов становится менее тривиальной и объем микропрограммы (число микрокоманд) возрастает.

Таким образом, конвейерная обработка микрокоманд (совмещение во времени этапов выполнения двух или трех микрокоманд) позволяет сократить длительность тактов, но влечет увеличение их числа на условных переходах. Поэтому при выборе структуры микроконтроллера следует учитывать соотношение значений $t_{\text{БОД}}$, $t_{\text{ФАМ}}$, $t_{\text{МПП}}$; характеристики алгоритма — число выполнений УП1, УП2 и БП.

Рассмотрим ряд наиболее распространенных частных случаев.

1. Независимо от характеристик алгоритма при постоянной длительности такта наибольшим быстродействием (т. е. наименьшим временем выполнения микропрограммы) обладают структуры:

- S_4 , если $t_{\text{БОД}} \geq t_{\text{ФАМ}} + t_{\text{МПП}}$;
 S_5 , если $t_{\text{ФАМ}} \geq t_{\text{МПП}} + t_{\text{БОД}}$;
 S_6 , если $t_{\text{МПП}} \geq t_{\text{БОД}} + t_{\text{ФАМ}}$;
 S_7 , если $t_{\text{БОД}} = t_{\text{ФАМ}} = t_{\text{МПП}}$.

2. Быстродействие структур при использовании переменной длительности такта (ПТ) превышает или в худшем случае равно быстродействию соответствующих конвейерных структур: т. е. $S_2^{\text{ПТ}}$ лучше (не хуже) S_4 , $S_1^{\text{ПТ}}$ лучше (не хуже) S_5 , а $S_6^{\text{ПТ}}$ лучше (не хуже) S_7 . Остальные структуры (S_3 , S_8 — S_{11}) уступают по быстродействию перечисленным. Кроме того, S_4 лучше (не хуже) S_5 , а $S_2^{\text{ПТ}}$ лучше $S_1^{\text{ПТ}}$.

3. Если выполняется система неравенств

$$\begin{aligned}
 t_{\text{БОД}} &< t_{\text{ФАМ}} + t_{\text{МПП}}, \\
 t_{\text{ФАМ}} &< t_{\text{БОД}} + t_{\text{МПП}}, \\
 t_{\text{МПП}} &< t_{\text{ФАМ}} + t_{\text{БОД}},
 \end{aligned}$$

то наиболее быстродействующей является структура $S_2^{\text{ПТ}}$ или $S_6^{\text{ПТ}}$ при использовании переменной длительности такта. Более точную оценку следует производить по формулам (которые будут приведены ниже) с учетом характеристик алгоритма.

4. Наиболее экономными в смысле аппаратных затрат являются структуры S_1 , $S_1^{\text{ПТ}}$, S_5 .

Рассмотрим правила построения микропрограмм для типовых структур, обладающих лучшими характеристиками. Исходные данные для построения микропрограммы — блок-схема алгоритма, представленная с такой степенью детализации, что каждый ее операторный блок (ОБ) выполним за один такт в БОД, а каждый условный блок (УБ) — в ФАМ за один такт. Предполагается, что и любые безусловные переходы выполняются в ФАМ также за один такт.

Пример исходного фрагмента блок-схемы алгоритма приведен на рис. 2.10. Здесь и далее используется следующая символика: H — начало фрагмента; K — конец; $O_i \Rightarrow \varphi$ означает формирование условия φ при выполнении операции O_i в БОД; $Y_j [\varphi]$ означает выполнение в ФАМ операции условного перехода по значению тестируемого условия φ .

Построение микропрограмм для структур S_1 и S_2 включает следующую последовательность шагов.

Шаг 1. Если данному УБ предшествует более чем один блок или только УБ, то перед данным УБ вставить операторный блок «нет операции» (НОП), выполнение которого не изменяет состояния БОД.

Шаг 2. Объединить блоки в группы, соответствующие микрокомандам. Микрокоманда с условным переходом соответствует УБ с непосредственно предшествующим ОБ, а микрокоманда с безусловным переходом — ОБ, не предшествующий условному блоку.

Шаг 3. Сопоставить адреса группам блоков и отдельным блокам, соответствующим микрокомандам.

Шаг 4. Записать микропрограмму по полученной блок-схеме, кодируя ОБ в операционном поле микрокоманды, а УБ или безусловные переходы — в адресном поле микрокоманды. При этом адреса переходов соответствуют на блок-схеме блокам, непосредственно следующим за данным.

Итоговая блок-схема представлена на рис. 2.11, а микропрограмма — в табл. 2.5.

Построение микропрограмм для структур S_4 , S_5 состоит из той же последовательности шагов с добавлением одного шага, выполняемого до шага 1, — назовем его шагом 0.

Шаг 0. Если УБ тестирует условие, формируемое непосредственно предшествующим ОБ (т. е. это УП1), то вставить между УБ и ОБ блок НОП.

Необходимость этого действия связана с тем, что при конвейерной обработке микрокоманд в структурах S_4 и S_5 невозможно в одном и том же такте формировать условие в БОД и затем тестировать его в ФАМ. Передаче условия препятствует РгС, куда информация заносится лишь на рубеже двух соседних тактов (по фронту синхросигнала),

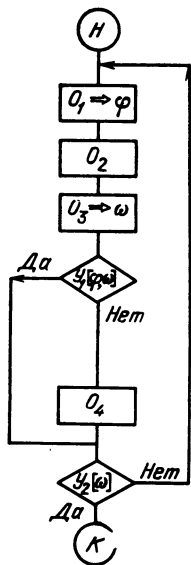


Рис. 2.10. Исходная блок-схема алгоритма

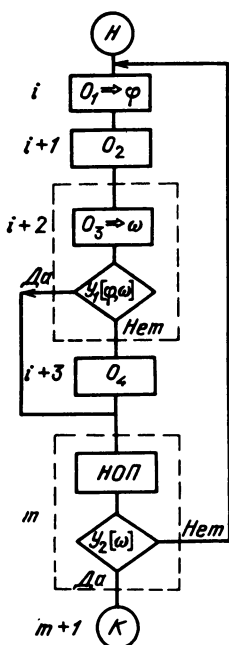


Рис. 2.11. Блок-схема алгоритма для S_1 , S_2

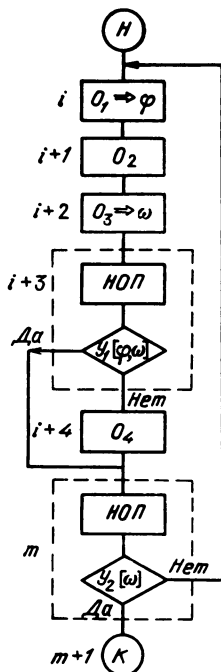


Рис. 2.12. Блок-схема алгоритма для S_4 , S_5

Т а б л и ц а 2.5. Микропрограмма
для S_1, S_2

Адрес	Микрокоманда	
	Операци- онная часть	Адресная часть
i	$O_1 \Rightarrow \varphi$	<i>CONT</i>
$i+1$	O_2	<i>CONT</i>
$i+2$	$O_3 \Rightarrow \omega$	$Y_1[\varphi, \omega] < m,$ $i+3 >$
$i+3$	O_4	<i>JMP m</i>
\vdots	\vdots	\vdots
\vdots	\vdots	\vdots
m	НОП	$Y_2[\omega] < m+1, i >$

Т а б л и ц а 2.6. Микропрограмма
для S_4, S_5

Адрес	Микрокоманда	
	Операци- онная часть	Адресная часть
i	$O_1 \Rightarrow \varphi$	<i>CONT</i>
$i+1$	O_2	<i>CONT</i>
$i+2$	$O_3 \Rightarrow \omega$	<i>CONT</i>
$i+3$	НОП	$Y_1[\varphi, \omega] < m,$ $i+4 >$
$i+4$	O_4	<i>JMP m</i>
\vdots	\vdots	\vdots
\vdots	\vdots	\vdots
m	НОП	$Y_2[\omega] < m+1, i >$

что, собственно, и позволяет сократить длительность такта при конвейерной обработке. Итоговая блок-схема алгоритма приведена на рис. 2.12, а микропрограмма — в табл. 2.6. На рис. 2.13 показаны временные диаграммы выполнения нескольких микрокоманд в структурах S_4 и S_2 . Видно, что если $t_{\text{БОД}} = t_{\text{ФАМ}} + t_{\text{МПП}}$, то конвейерная (на S_4) и последовательная (на S_2) обработки микрокоманд с переменной длительностью такта имеют одинаковую производительность. При ином соотношении $t_{\text{БОД}}$, $t_{\text{ФАМ}}$ и $t_{\text{МПП}}$ переменная длительность такта обеспечивает большее быстродействие.

Построение микропрограммы для структуры S_6 менее тривиально. Предположим, что на текущем такте в БОД выполняется операция $O_3 \Rightarrow \omega$, условие ω поступает на ФАМ, где вычисляется адрес перехода посредством операции $Y_1[\varphi, \omega]$. В это же время из МПП производится выборка микрокоманды, которая будет управлять БОД на следующем такте независимо от реальных значений условий. И только через один такт в РгМК появится микрокоманда, которая соответствует реальным значениям условий. Поэтому необходимо позаботиться о том, чтобы предшествующая ей микрокоманда не «увела» вычислительный процесс в непредсказуемом направлении.

Таким образом, на каждом такте в ФАМ формируется адрес микрокоманды, которая будет выполняться через один такт от текущей. Это отражается на построении микропрограммы, которое в данном случае включает следующую последовательность шагов.

Шаг 1. Если данному УБ предшествует более чем один блок или только УБ, то перед данным УБ вставить НОП.

Шаг 2. На один из выходов УБ (например, по исходу «да») поместить идентичный ему дублирующий УБ в совокупности с блоком НОП. Выходы дублирующего УБ (ДУБ) соединить с теми же блоками, которые прежде следовали за дублируемым УБ.

Шаг 3. Объединить блоки в группы, соответствующие микрокомандам. Микрокоманда с условным переходом соответствует УБ с непосредственно предшествующим ОБ, а микрокоманде с безусловным переходом — ОБ, не предшествующий УБ.

Шаг 4. Сопоставить адреса группам блоков и отдельным блокам, соответствующим микрокомандам.

Шаг 5. Записать микропрограмму по полученной блок-схеме, кодируя ОБ в операционном поле микрокоманды, а УБ или безусловные переходы — в адресном. При этом от ОБ и дублирующих УБ осуществляется переход к адресам блоков, следующих на блок-схеме алгоритма через один вперед от данного. Так же (т. е. через один вперед от данного) осуществляется переход от дублируемого УБ по выходу, на котором помещен дублирующий УБ. По другому выходу УБ переход осуществляется к адресу следующего блока, а не через один. От операторного блока, предшествующего дублируемому УБ, необходимо организовать переход к дублирующему УБ, а не к альтернативной ветви, исходящей из УБ.

На рис. 2.14 приведена блок-схема алгоритма, полученная после выполнения шага 5. Штриховыми линиями обведены дублируемые

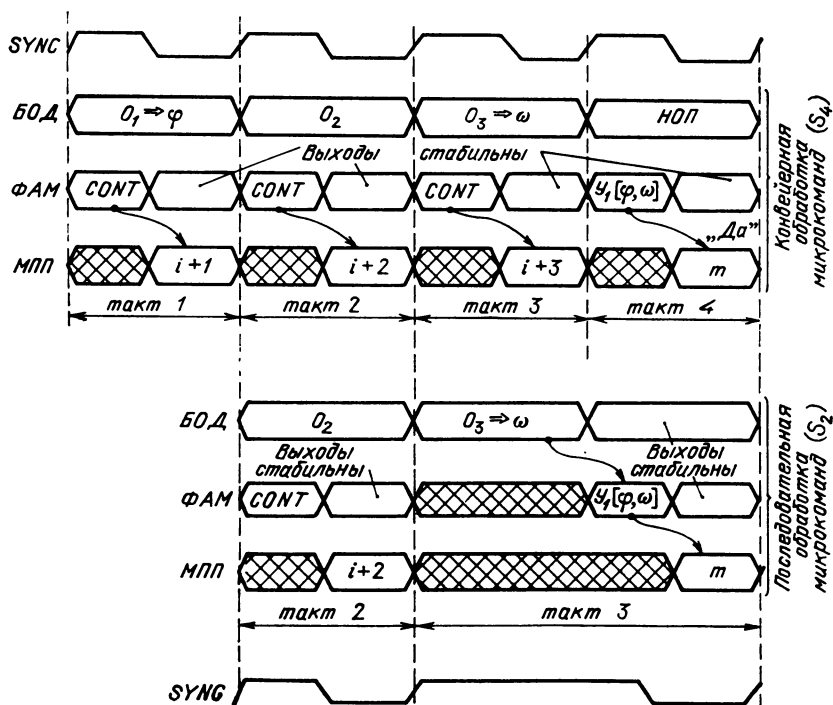
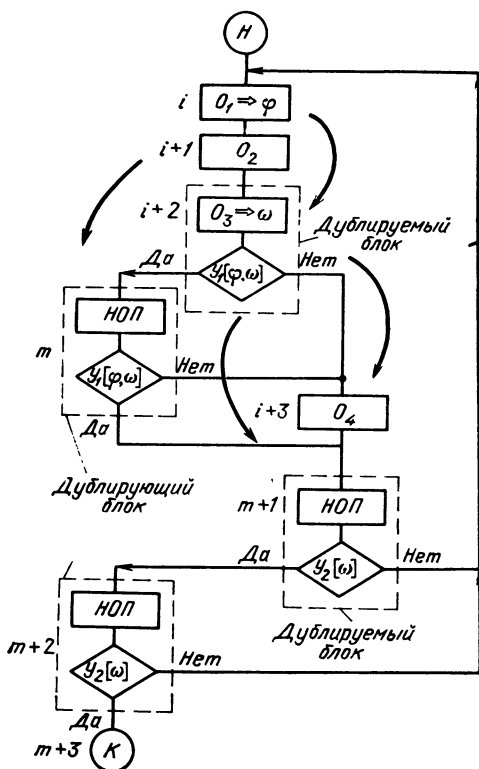


Рис. 2.13. Временные диаграммы конвейерной и последовательной обработок микрокоманд

Таблица 2.7. Микропрограмма для S_7

Адрес	Микрокоманда	
	Операционная часть	Адресная часть
i	$O_1 \Rightarrow \varphi$	$JMP\ i+2$
$i+1$	O_2	$JMP\ m$
$i+2$	$O_3 \Rightarrow \omega$	$Y_1[\varphi, \omega] < m+1,$ $i+3 >$
$i+3$	O_4	$JMP\ m+2$
\vdots	\vdots	\vdots
m	НОП	$Y_1[\varphi, \omega] < m+2,$ $m+1 >$
$m+1$	НОП	$Y_2[\omega] < m+3,$ $i >$
$m+2$	НОП	$Y_2[\omega] < m+4,$ $i+1 >$

Рис. 2.14. Блок-схема алгоритма для S_6



УБ и ДУБ. Утолщенными стрелками показаны некоторые переходы при адресации блоков в микропрограмме (табл. 2.7) по правилу, описанному в шаге 5.

На рис. 2.15 изображены временные диаграммы выполнения части фрагмента микропрограммы для структуры S_6 с переменной длительностью такта при соотношении $t_{\text{БОД}} = t_{\text{ФАМ}} = t_{\text{МПП}}$.

Обратим внимание на то, что при реализации условных переходов в структуре S_6 на двух соседних тактах выполняются два идентичных

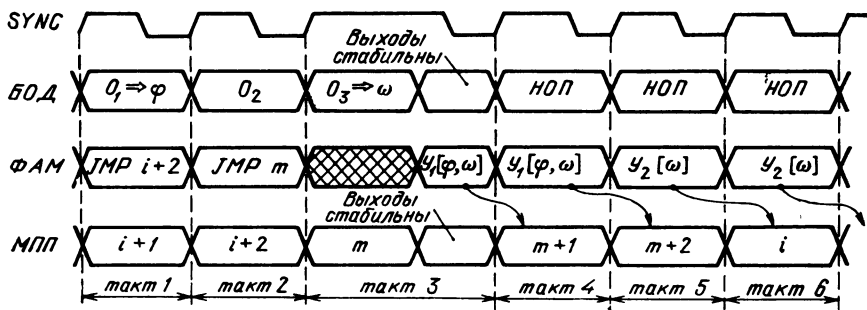


Рис. 2.15. Временные диаграммы выполнения фрагмента микропрограммы на S_6

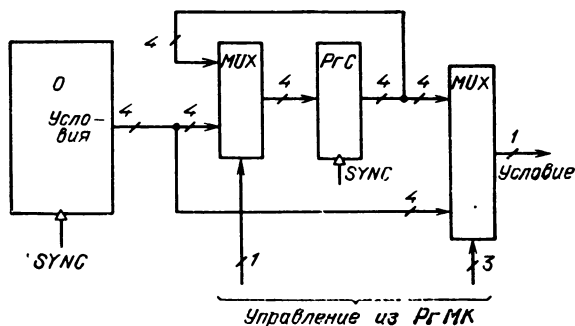


Рис. 2.16. Управление регистром состояния и выбор условий

УБ, тестирующих одни и те же условия. Очевидно, значения этих условий должны быть неизменны. Для хранения их используется PrC, хотя он и не изображен на рисунке типовой структуры S_6 (см. рис. 2.8), потому что сформированное в БОД условие может на том же такте поступать для тестирования в ФАМ. Схема, позволяющая передавать условия, сформированные на текущем такте или выработанные ранее и хранимые в PrC, показана на рис. 2.16. Управление мультиплексорами осуществляется из адресного поля микрокоманды.

Замечания о построении микропрограмм для структуры S_6 .

1. На линейных участках алгоритма инструкции *JMP* с непосредственным указанием адреса перехода можно заменить на *CONT* (продолжение), ибо СМК микросхемы K1804BY4 содержит адрес, выработанный на предыдущем такте.

2. Если по каждой ветви, исходящей из УБ, следует не менее двух блоков с последовательными адресами, то вместо ДУБ можно использовать микрокоманду, содержащую НОП и *CONT*. При этом можно записать ее в памяти один раз для нескольких УБ.

3. Если дублируется условный переход типа УП2, то входящий в него операторный блок можно перенести в ДУБ вместо НОП, а УБ объединить в микрокоманду с предшествующим операторным блоком. Таким образом, условный переход будет выполняться на такт быстрее.

Построение микропрограммы для структуры S_7 состоит из той же последовательности шагов, что и для S_6 с единственным отличием: добавляется шаг 0, идентичный описанному выше для структур S_4 , S_6 .

Исходя из правил построения микропрограмм для конкретных структур (S_r) можно привести простые формулы, позволяющие оценить время выполнения микропрограммы T_r . Например, для структур S_1 , S_2 , S_4 — S_7 с постоянной длительностью такта τ_r ($r = 1, 2, 4$ — 7) оценки имеют вид

$$T_1 \leq \tau_1 (N_0 + N_{y1} + N_{y2}), \quad T_2 \leq \tau_2 (N_0 + N_{y1} + N_{y2}),$$

$$T_4 \leq \tau_4 (N_0 + 2N_{y1} + N_{y2}), \quad T_5 \leq \tau_5 (N_0 + 2N_{y1} + N_{y2}),$$

$$T_6 \leq \tau_6 (N_0 + 2N_{y1} + 2N_{y2}), \quad T_7 \leq \tau_7 (N_0 + 3N_{y1} + 2N_{y2}),$$

где N_O — сумма числа выполнений всех операторных блоков, кроме тех, что входят в микрокоманды с условным переходом; N_{Y_1} — сумма числа выполнений всех УП1; N_{Y_2} — сумма числа выполнений всех УП2, присутствующих в исходной блок-схеме алгоритма до ее преобразования.

Для некоторых структур с переменной длительностью такта («длинный» такт при выполнении УП1, «короткий» при выполнении УП2 и БП) оценки таковы: $T_1^{PT} \leq \tau_1 N_{Y_1} + \tau_1^* (N_O + N_{Y_2})$, $T_2^{PT} \leq \tau_2 N_{Y_1} + \tau_2^* (N_O + N_{Y_2})$, $T_6^{PT} \leq \tau_6 N_{Y_1} + \tau_6^* (N_O + 2N_{Y_2} + N_{Y_1})$. Отметим, что две различные длительности такта — это лишь частный случай организации нескольких различных длительностей такта (см. § 2.2), отсюда и знак « \leq » в формулах вместо строгого равенства.

Примеры использования приведенных оценок. Пусть $t_{\text{БОД}} = 210$ нс, $t_{\text{ФАМ}} = 140$ нс, $t_{\text{МПП}} = 80$ нс, $N_O = 290$, $N_{Y_1} = 100$, $N_{Y_2} = 100$. Тогда $\tau_1 = \tau_2 = 210 + 80 + 140 = 430$ нс, $\tau_1^* = 80 - \max\{210, 140\} = 290$ нс, $\tau_2^* = \max\{(140 + 80), 210\} = 220$ нс, $T_1 = T_2 = 430 \cdot 490 = 210$ мкс, $T_1^{PT} = 430 \cdot 100 + 290 \cdot 390 = 156$ мкс, $T_2^{PT} = 430 \cdot 100 + 220 \cdot 390 = 129$ мкс. Таким образом, для заданного соотношения $t_{\text{БОД}}$, $t_{\text{ФАМ}}$, $t_{\text{МПП}}$ и N_O , N_{Y_1} , N_{Y_2} применение переменной длительности такта дает заметный выигрыш в быстродействии. Наибольшее быстродействие обеспечивает структура S_2^{PT} с переменной длительностью такта, но по аппаратным затратам она превышает S_1^{PT} из-за того, что разрядность РгМК значительно больше разрядности РгАМК. Аналогично можно подсчитать $T_4 = 220 \cdot (290 + 200 + 100) = 130$ мкс и $T_5 = 290 \cdot (290 + 200 + 100) = 171$ мкс и сделать вывод об одинаковом быстродействии S_2^{PT} и S_4 в данном случае. Нетрудно доказать, что структуры S_6 и S_7 при заданных значениях $t_{\text{БОД}}$, $t_{\text{ФАМ}}$, $t_{\text{МПП}}$, N_O , N_{Y_1} , N_{Y_2} будут обеспечивать существенно меньшее быстродействие.

Осталось ответить на вопрос, как вычислить значения N_O , N_{Y_1} и N_{Y_2} , используемые в приведенных формулах для оценки быстродействия. Оценка числа тактов, требуемых для выполнения микропрограммы алгоритма (см. рис. 2.11), включает следующие шаги.

Шаг 1. Фиксировать число обращений к рассматриваемой микропрограмме (20) и вероятности p_i исхода по ветвям «да», «нет» для каждого УБ.

Шаг 2. Построить граф переходов (рис. 2.17) между фрагментами блок-схемы алгоритма. Вершина графа образуется линейной последовательностью блоков и заканчивается либо разветвлением, либо слиянием ветвей. Дуги графа взвешиваются вероятностями переходов между вершинами.

Шаг 3. Составить систему линейных алгебраических уравнений, в i -й строке

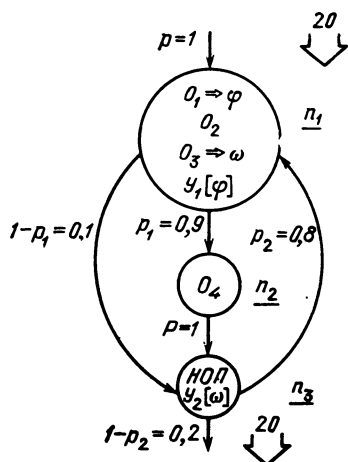


Рис. 2.17. Граф переходов

выражая число вхождений в i -ю вершину графа переходов [6], на-
пример (см. рис. 2.17): $n_1 = 20 \cdot 1 + n_3 \cdot 0,8$; $n_2 = n_1 \cdot 0,9$; $n_3 =$
 $= n_1 \cdot 0,1 + n_2 \cdot 1$.

Шаг 4. Решив систему уравнений, получить значения n_i .

Шаг 5. По значениям n_i определить значения N_O , N_{y1} и N_{y2} .

Затем можно использовать N_O , N_{y1} и N_{y2} для ориентировочной
оценки времени выполнения микропрограмм по приведенным выше
формулам, соответствующим различным типовым структурам. Оче-
видно, достоверность этих оценок определяется достоверностью вероят-
ностей p_i , а также правомочностью округления значений всех задер-
жек ОЧ и ФАМ до двух максимальных — $t_{ОЧ}$ и $t_{ФАМ}$. Последнее чаще
всего дает заниженную оценку быстродействия, и, чтобы в полной мере
реализовать скоростные возможности элементной базы, необходи-
мо перейти к более тщательной оценке длительности такта. Этому по-
священ следующий параграф.

2.2. СИНХРОНИЗАЦИЯ

После написания микропрограмм для фиксированной структуры
микропроцессора становится возможным определение путей распро-
странения сигналов при выполнении каждой из микрокоманд. На осно-
вании паспортных данных о задержках микросхем и функциональных
схем БОД и БМУ для каждой микрокоманды определяется критичес-
кий путь распространения сигнала, имеющий максимальную суммар-
ную задержку t_i .

В качестве изобразительного аппарата для расчета задержек исполь-
зуются временные диаграммы, таблицы и графы, причем для сложных
схем предпочтительнее последние в силу их простоты и наглядности.
Кратко охарактеризуем каждый из способов на примере расчета за-
держек простейшей схемы (рис. 2.18), состоящей из регистра (Рг) с
загрузкой информации по фронту синхронимпульса и трех комбина-

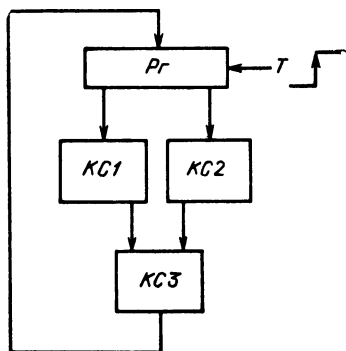


Рис. 2.18. Структурная схе-
ма для расчета задержек

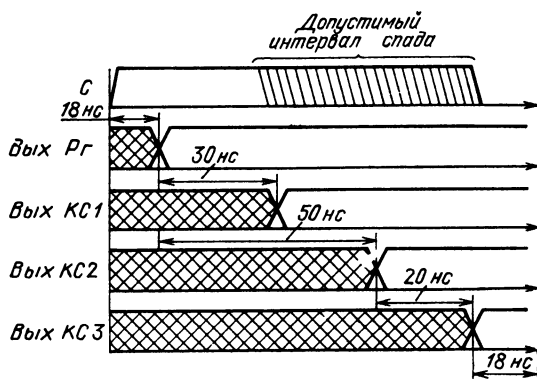


Рис. 2.19. Временные диаграммы распространения
сигналов

онных схем (КС1, КС2 и КС3). Известны временные характеристики всех перечисленных элементов: время установки данных на входе регистра 5 нс, время считывания информации на выход регистра от момента поступления тактового сигнала 18 нс, задержки вход-выход для КС1 30 нс, КС2 50 нс и КС3 20 нс. Если пренебречь задержками распространения сигнала между микросхемами, то длительность такта для указанной схемы определяется задержкой прохождения сигнала с выхода регистра на его вход через КС2 и КС3. Для простой схемы этот факт можно установить «невооруженным глазом», а для расчета сложной приходится использовать какой-либо изображительный аппарат. Пример использования временных диаграмм показан на рис. 2.19. Достоинство временных диаграмм — возможность указания конкретных значений сигналов («1», «0», высокое сопротивление) в точках схемы, что способствует лучшему пониманию ее работы. Недостаток — громоздкость для сложных схем с множеством связей. В [4] расчет задержек иллюстрируется с помощью таблиц (см., например, табл. 2.8).

Максимальная из суммарных задержек всех возможных путей распространения сигнала определяет длительность такта. Недостаток табличной формы представления — сравнительная ненаглядность, трудоемкость безошибочного перехода от схемы к таблице, невозможность (по сравнению с диаграммами) отражения конкретных значений сигналов. Последним из упомянутых недостатков характеризуется и представление задержек графами. Однако графы превосходят таблицы по простоте и наглядности из-за более естественной процедуры их построения по функциональной схеме устройства. Резко сокращаются и расчеты: вместо перечисления всех возможных путей в таблице с последующим суммированием задержек их элементов достаточно применить несложный алгоритм поиска пути максимальной длины на графе [7, 8].

В [1] задержки обозначены вершинами графа, а дуги графа соответствуют физическим соединениям элементов, вносящих задержки, в схеме устройства. Рассматриваемому примеру будет соответствовать граф на рис. 2.20. Дуги, соединяющие вершины критического пути, выделены утолщенными линиями. Существенно, что взвешенными являются здесь вершины. Более естественным переход от схемы к графу задержек получается, если задержками взвешивать дуги, а не вершины графа (рис. 2.21). При этом вершины соответствуют физическим соединениям элементов, вносящих задержки (например, соединениям

Т а б л и ц а 2.8. Расчет задержек

Элемент	Операция	Время распространения сигнала, нс	
		Путь 1	Путь 2
Рг	Чтение	18	18
КС1	Передача	30	—
КС2	«	—	50
КС3	«	20	20
Рг	Установка	5	5
Суммарная задержка		73	93

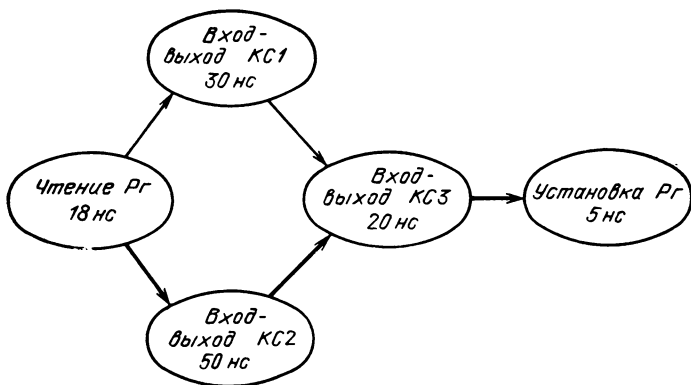


Рис. 2.20. Представление задержек вершинами графа

выводов микросхем). Кроме задержек типа вход-выход для комбинационных схем дугами (с фиктивной конечной вершиной) следует обозначать минимальное время цикла и времена установки для схем с памятью.

Алгоритм поиска критического пути в таком графе состоит в последовательном вычислении весов вершин (веса записываются в вершинах) и включает следующие шаги.

Шаг 1. Всем начальным вершинам (имеющим только выходящие дуги и не имеющим входящих дуг) присвоить нулевой вес.

Шаг 2. Если определены веса W_F всех вершин $\{F\}$, из которых поступают дуги на вершину G , то определить вес W_G вершины G как максимум из суммы весов каждой из вершин, предшествующих G с весом ω_{FG} дуги, идущей из этой предшествующей вершины к вершине G :

$$W_G = \max_{\{F\} \rightarrow G} \{W_F + \omega_{FG}\}.$$

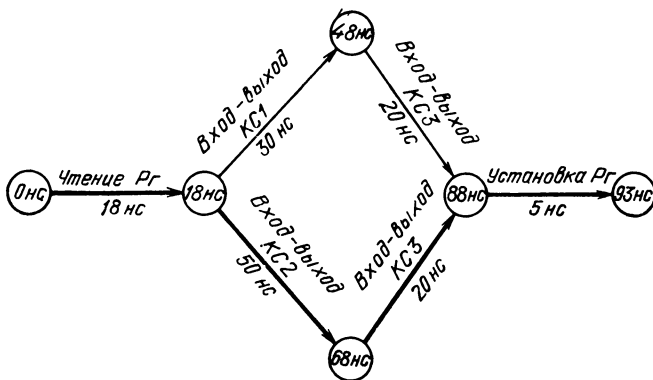


Рис. 2.21. Представление задержек дугами графа

В примере $\max \{48+20, 68+20\} = 88$. Шаг 2 повторять до тех пор, пока веса всех вершин графа не будут определены. Граф корректно построенной схемы устройства отражает ее функционирование в рамках одного такта и поэтому не должен содержать циклов, иначе возможен эффект состязаний сигналов.

Шаг 3. Сравнить веса всех конечных вершин (имеющих только входящие дуги и не имеющих выходящие). Значение максимального из весов является минимальной длительностью такта.

Выше использовалось допущение о том, что задержки, вносимые, схемами КС1, КС2, КС3, не зависят от выполняемых ими операций если последних несколько и возможна «настройка» схем на выполнение той или иной операции внешними управляющими сигналами. Особенностью микропрограммируемых БИС является именно упомянутая настройка на ту или иную операцию на каждом такте функционирования. Поэтому индивидуальный граф задержек необходимо строить для каждой микрокоманды, учитывая задеиствуемые пути распространения сигналов. Полученные значения задержек критических путей являются реальными длительностями выполнения микрокоманд (в предположении равенства паспортных и действительных задержек для каждой из микросхем). Далее, для каждой микрокоманды $МК_i$ должен быть назначен интервал времени (такт) τ_i , такой, что $\tau_i \geq t_i$, где t_i есть реальная длительность выполнения микрокоманды.

Простейшее решение — тактировать все микрокоманды интервалом $T_m = \max t_i$. Недостатком его является необходимость выполнения «коротких» микрокоманд с максимальной длительностью такта, что влечет непроизводительные затраты времени в каждом такте выполнения коротких микрокоманд. Потери времени тем больше, чем чаще встречаются короткие микрокоманды и чем больше их реальная длительность отличается от длительности такта T_m .

Противоположное решение — установить для каждой микрокоманды такт, равный ее реальной длительности: $\tau_i = t_i$. При этом значительно усложняется схема синхронизации, а выигрыш в быстродействии оказывается очень мал (единицы процентов) по сравнению с третьим решением — использованием переменной длительности такта, которое применяется при построении современных микропроцессоров [1, 4] и обсуждается ниже.

Суть переменной длительности такта состоит в следующем: исходное множество длительностей $\{t_i\}$ микрокоманд с учетом вероятностей $\{p_i\}$ их выполнения разбивают на L подмножеств так, чтобы в рамках каждого подмножества все микрокоманды тактировать одинаковым интервалом, а разные подмножества — разными интервалами, кратными Δ -периоду задающего генератора. Информация о длительности такта хранится в микрокоманде (рис. 2.22).

При заданном числе подмножеств L (в итоге оно и определяет сложность схемы синхронизации) существует C_{Q-1}^{L-1} вариантов разбиения исходного множества. Здесь Q есть мощность исходных множеств $\{t_i\}$ и $\{p_i\}$, а C_{Q-1}^{L-1} — число сочетаний из $Q-1$ по $L-1$. Среди вариантов есть более и менее удачные с точки зрения результирующего быстродействия. Есть и оптимальный вариант, при задан-

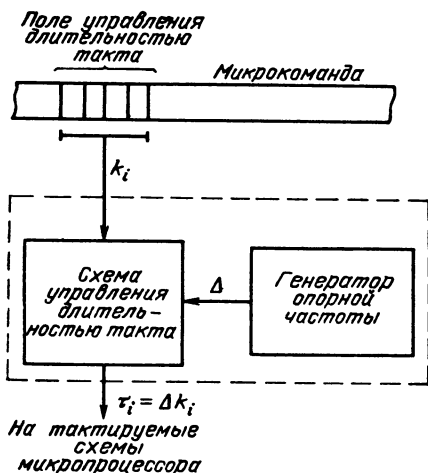


Рис. 2.22. Управление длительностью такта

ных $Q, L, \{t_i\}, \{p_i\}$ обеспечивающий максимальное быстродействие микропроцессора. Рассмотрим задачу оптимального разбиения и сделаем выводы.

При разбиении множества $\{t_i\}$ на подмножества для каждой микрокоманды MK_i может быть назначен такт $\tau_i = \Delta k_i$, где $k_i \geq \lceil t_i / \Delta \rceil$. Здесь $\lceil X \rceil$ — ближайшее целое, большее или равное X ; k_i — число полных периодов задающего генератора, необходимых для покрытия величины t_i . Оценкой быстродействия микропроцессора в таком случае будет величина S , обратная взвешенной длительности такта T , где $T = \Delta \sum_{i=1}^Q p_i k_i$.

Для заданных $\{t_i\}, \{p_i\}$ требуется подобрать L, Δ и разбиение при фиксированных L и Δ так, чтобы обеспечить максимальное быстродействие. Простейший пример: $L \leq 3$, $\Delta = \{42, 60, 70, 80, 90, 100\}$, $t_1 = 162$,

$t_2 = 200$, $t_3 = 260$, $p_1 = 0.4$, $p_2 = 0.5$, $p_3 = 0.1$. Возможные варианты представлены в табл. 2.9. Максимум функции S соответствует последней строке табл. 2.9: для получения максимального быстродействия необходимо установить задающий генератор с периодом 42 нс, выделить в микрокоманде два разряда для управления длительностью такта. Схема управления должна вырабатывать такты с длительностью 168 (4·42), 210 (5·42) и 294 (7·42) нс.

При заданных вероятностях $\{p_i\}$ итоговое быстродействие составит 5,25 млн. микрокоманд/с. Отметим, что тактирование всех микрокоманд интервалом 250 нс привело бы к снижению быстродействия до 3,85 млн. микрокоманд/с, хотя разность $t_3 - t_2$ составляет всего 60 нс. Удачное решение представлено в шестой строке табл. 2.9: повышение быстродействия на 30 % по отношению к первой строке достигается введением двух различных длительностей такта (200 нс для первой и второй микрокоманды, 300 нс для третьей). Управление обеспечивается всего одним разрядом микрокоманды, а опорная синхропоследовательность имеет удобный для технической реализации период (100 нс). Кроме того, счет периодов потребует меньше аппаратных затрат, чем в предыдущем случае, если

Таблица 2.9. Варианты тактирования

L	Δ , нс	τ_1 , нс	τ_2 , нс	τ_3 , нс	T , нс	S , млн. микрокоманд/с
1	260	260	260	260	260	3,85
2	60	180	300	300	241,2	4,15
2	60	240	240	300	240	4,17
2	70	210	210	280	210,5	4,75
2	80	240	240	320	240,8	4,15
2	100	200	200	300	201	4,98
2	90	180	270	270	225,9	4,43
3	60	180	240	300	211,2	4,73
3	42	168	210	294	190,3	5,25

схема управления реализуется на дискретных элементах средней и малой степени интеграции.

Сформулируем общие ограничения на значения Δ и L . Во-первых, $\text{НОД} \{t_i\} \leq \Delta \leq T_m$, где НОД — наибольший общий делитель. При выборе Δ играет роль и удобство технической реализации задающего генератора (например, наличие соответствующих кварцевых резонаторов).

Ограничения на L : $2 \leq L \leq Q - 1$ (иначе постановка задачи оптимального разбиения по L теряет смысл) и $\log_2 L \leq d$, где d — допустимая разрядность поля микрокоманды, управляющего длительностью такта. Косвенно d характеризует и сложность схемы управления длительностью такта. Таким образом, если для исходных $\{t_i\}$, $\{p_i\}$ значение $E = T/T_m$ будет близко к единице, то во введении переменной длительности такта нет необходимости. В противном случае нужно задаться диапазоном значений L и Δ для каждого их сочетания и отыскать оптимальный вариант разбиения множества $\{k_i\}$ на L подмножеств. Далее, сравнением величин S (или T), соответствующих оптимальным вариантам, определить экстремум. Указанные процедуры перебора несложно реализовать программно. Исключение может составлять лишь поиск оптимального варианта разбиения при фиксированных Δ , L . Применение стратегии полного перебора характеризуется здесь экспоненциально растущей трудоемкостью. Альтернативой стратегии полного перебора является применение аппарата линейного программирования с булевыми переменными [1]. Решением задачи являются значения переменных, соответствующие оптимальному разбиению множества $\{t_i\}$ на L подмножеств. При этом оптимальными длительностями тактов являются максимальные в рамках выделенных подмножеств.

При фиксированном значении Δ интерес представляет зависимость S от L , позволяющая ответить на вопрос, каким значением L целесообразно ограничиться. В результате вычислительных экспериментов сделаны следующие наблюдения. При $L = 8$ быстродействие всего на 1—3 % не достигает предельного максимального значения независимо от конкретных значений $\{t_i\}$ и $\{p_i\}$. Для большинства же практических разработок микропроцессоров достаточно взять $L = 4$. При этом отличие от E_{\min} составляет 10—15 %, что вполне достаточно с учетом точности задания исходных данных — значений $\{t_i\}$ и $\{p_i\}$. Однократный поиск оптимального разбиения при $L = 4$ можно провести на ЭВМ методом полного перебора.

Итак, переменная длительность такта является дешевым и эффективным способом повышения быстродействия микропроцессора до предельных значений, обеспечиваемых элементной базой. Дополнительные аппаратные затраты невелики; это два-три корпуса микросхем средней степени интеграции или одна микросхема К1804ГГ1, а также 1—3 разряда микрокоманды. При использовании оптимального набора длительностей такта типовой выигрыш в быстродействии микропроцессора составляет десятки процентов.

2.3. СПОСОБЫ ПОВЫШЕНИЯ БЫСТРОДЕЙСТВИЯ

Быстродействие микропрограммного контроллера на заданном алгоритме оценивается величиной, обратной времени выполнения микропрограммы этого алгоритма. В свою очередь, время выполнения микропрограммы определяется необходимым числом тактов и длительностью этих тактов. Поэтому все способы повышения быстродействия микропрограммных контроллеров (рис. 2.23) ориентированы либо на уменьшение числа тактов, либо на сокращение их длительности, либо на то и другое одновременно.

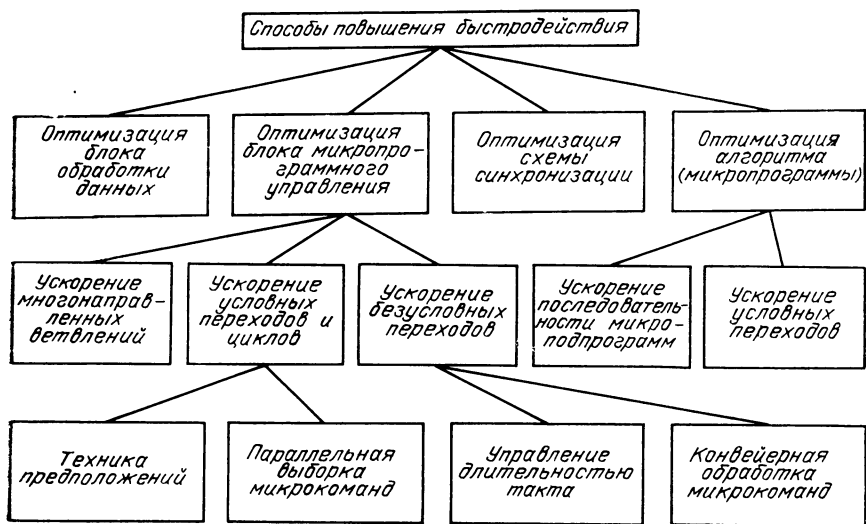


Рис. 2.23. Систематизация способов повышения быстродействия

Кардинальным решением проблемы согласно [9] могла бы быть такая структура микроконтроллера, в которой длительность такта минимальна и равна $\max\{t_{\text{БОД}}, t_{\text{ФАМ}}, t_{\text{МПП}}\}$, а число тактов минимально из-за того, что одновременно с выполнением в БОД текущей микрокоманды производится выборка из МПП всех возможных микрокоманд — последователей текущей, и в ФАМ в это же время формируются возможные адреса всех микрокоманд, которые могут следовать за этими последователями, т. е. через один такт вперед от настоящего. Реализация подобной структуры наталкивается на ряд трудностей. Во-первых, это многократное увеличение аппаратных затрат на модули МПП, мультиплексоры, регистры и т. д., а значит, невозможность реализации компактных одноплатных изделий. Задержки, возникающие из-за введения дополнительных элементов на платы и межплатных связей, вынуждают увеличивать длительность такта. Во-вторых, одновременная генерация нескольких адресов в ФАМ (например, четырех при выполнении последовательности из двух условных переходов) возможна лишь при разбиении ФАМ на несколько (здесь на четыре) модулей, работаю-

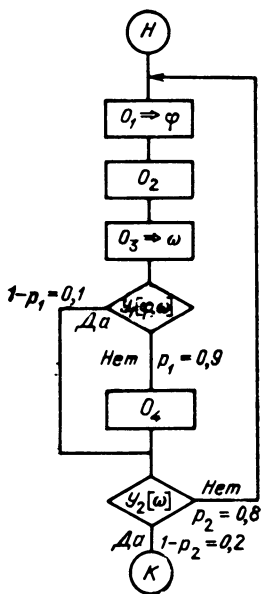


Рис. 2.24. Исходная блок-схема алгоритма

щих одновременно и независимо. Использование для этих целей четырех БИС К1804ВУ4 невозможно из-за необходимости установки элементов памяти каждой БИС в любое требуемое состояние за один такт. Эта ситуация имеет место, когда два из четырех адресов выбраны в качестве исходных для модулей памяти, а затем когда из этих двух устанавливается единственно верный.

Для всех (четырех) возможных адресов перехода можно было бы ввести дополнительные поля в микрокоманду и вообще не использовать ФАМ в интегральном исполнении. Но тогда длина микрокоманды (а стало быть, и разрядность обоих модулей МПП) возрастает более чем на четыре десятка бит, а полученное устройство перестает обладать теми качествами, которые определяют эффективность микропроцессорной техники. Поэтому ниже рассмотрим лишь те способы повышения быстродействия, для которых характерно преобладание БИС в структуре устройства и незначительное увеличение аппаратных затрат для обеспечения повышенного быстродействия.

Под оптимизацией БОД понимается (рис. 2.23) введение специализированных БИС или схемных решений, позволяющих уменьшить время арифметическо-логической обработки данных при ограничении на допустимый объем аппаратных затрат. Например, для умножения со скоростью 1 бит/такт при использовании микропроцессорных секций К1804ВС1 необходимо ввести двухвходовый мультиплексор и инвертор, а для достижения максимального быстродействия на операции умножения — включить в состав микроконтроллера параллельный умножитель из серии К1802. Другим примером может служить использование БИС сдвигателя, позволяющей за один такт выполнять сдвиг на произвольное число разрядов. К оптимизации БОД относятся и вопросы рациональной организации информационных связей (трактов передачи данных) между модулями: чем более разветвленные связи, тем меньше потерь на временное мультиплексирование передач данных, но больше и аппаратные затраты. К оптимизации БОД можно отнести и введение ПЗУ для таблично-алгоритмического вычисления функций.

Аспекты оптимизации системы синхронизации подробно обсуждались в § 2.2, а конвейерной обработки микрокоманд — в § 2.1. Здесь лишь отметим, что управление длительностью такта («короткий» при безусловных переходах или «длинный» при условных) можно рассматривать как частный случай выбора группы длительностей такта для множества длительностей микрокоманд. Что касается конвейерной обработки микрокоманд, то быстродействие микроконтроллера при выполнении безусловных переходов определяется соотношением значений $t_{\text{БОД}}$, $t_{\text{ФАМ}}$, $t_{\text{МПП}}$, стало быть, правильностью выбора типовой структуры. Например, равенству $t_{\text{БОД}} = t_{\text{ФАМ}} + t_{\text{МПП}}$ наиболее соответствуют структуры S_4 (если не используется переменная длительность такта) или S_2 (если используется), а случаю $t_{\text{БОД}} = t_{\text{ФАМ}}$ $t_{\text{МПП}}$ соответственно S_7 и S_6 .

Техника предположений — весьма изящный способ сокращения числа тактов при выполнении условных переходов — подробно рассматривается в § 2.4. О параллельной выборке микрокоманд (включая параллельное вычисление адресов в ФАМ) речь шла в начале данного параграфа.

Преобразование алгоритма (микропрограммы) как способ повышения быстродействия позволяет сократить число тактов при выполнении алгоритма без каких-либо модификаций БОД и ФАМ, когда они уже фиксированы. Преобразование удобно проводить на блок-схеме алгоритма (представленной с соответствующей степенью детализации), а затем уже по итоговой блок-схеме кодировать микропрограмму. Не претендуя на полноту охвата вопросов формального преобразования алгоритмов микропрограмм, обсудим ряд примеров, иллюстрирующих указанный способ повышения быстродействия.

Пусть задана блок-схема алгоритма, представленная с такой степенью детализации, что каждый ее ОБ выполним за один такт в БОД, а каждый УБ — в ФАМ (рис. 2.24). Кроме того, для оценки времени выполнения микропрограммы будем считать известными вероятности исхода из условных блоков (p_i) по возможным направлениям. Согласно правилам, изложенным в § 2.1, и с использованием там же введенных обозначений исходная блок-схема будет преобразована в модифицированную (рис. 2.25) введением блоков НОП и объединением блоков в группы, соответствующие микрокомандам. Микрокоманда включает операционную часть, в которой кодируется O_i , и адресную, в которой кодируется условный переход Y_i или безусловный переход.

Рядом с блоками на рис. 2.25 проставлены подчеркнутые числа, которые означают количество выполнений блока при двадцати обращениях к изображенному фрагменту алгоритма. Время выполнения микропрограммы (в тактах) равно сумме числа обращений к слагающим ее микрокомандам. Для блок-схемы алгоритма на рис. 2.25 это время равно 490 тактам при 20 обращениях к микропрограмме.

Теперь преобразуем исходную блок-схему алгоритма (см. рис. 2.25) к модифицированной (рис. 2.26) путем «расщепления» УБ, которым предшествует более одного блока. В примере расщеплен (т. е. тиражирован) блок Y_2 [ω]. Очевидно, что такое преобразование никак не отражается на логике реализации алгоритма, а вот время выполнения микропрограммы станет меньше: в этом можно убедиться, построив микропрограмму для модифицированной блок-схемы алгоритма (рис. 2.26) по тем же правилам. Такая микропрограмма будет выполняться за 400 тактов при 20 обращениях, а число хранимых в МПП микрокоманд останется равным пяти.

Итак, если условному блоку предшествует более одного блока, то следует тиражировать его по числу предшествующих блоков и объединить соответствующие выходящие дуги из тиражированных УБ. Рассмотренное преобразование всегда гарантирует неувеличение числа тактов, т. е. в худшем случае быстродействие не изменится, а в ос-

тальных — увеличится. Что касается числа хранимых в МПП микрокоманд, то оно может либо остаться тем же (как показано в примере), либо увеличиться, либо уменьшиться в зависимости от того, сколько блоков и какие именно предшествуют тиражируемому УБ. Например, если предшествуют только ОБ, то тиражирование целесообразно, а если только УБ — нецелесообразно: быстродействие не увеличится, а число хранимых микрокоманд возрастет.

Другое преобразование состоит в перенесении ОБ со входа УБ на все его выходы, если данные ОБ и УБ не связаны формированием и тестированием одного и того же условия. Так, микропрограмма для исходной блок-схемы алгоритма (рис. 2.27) при 100 обращениях к ней будет выполнена за 327 тактов при числе хранимых микрокоманд, равном 5. Преобразованная блок-схема алгоритма (рис. 2.28) получается путем перенесения блока O_2 на все выходы блока $Y_1[\varphi]$, что не отражается на логике алгоритма. По модифицированной блок-схеме строится микро-

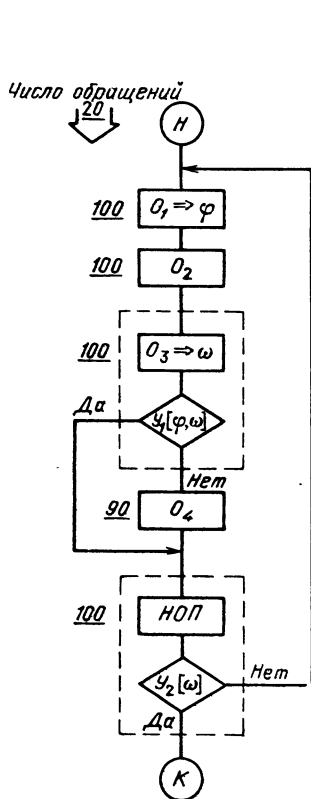


Рис. 2.25. Модифицированная блок-схема алгоритма

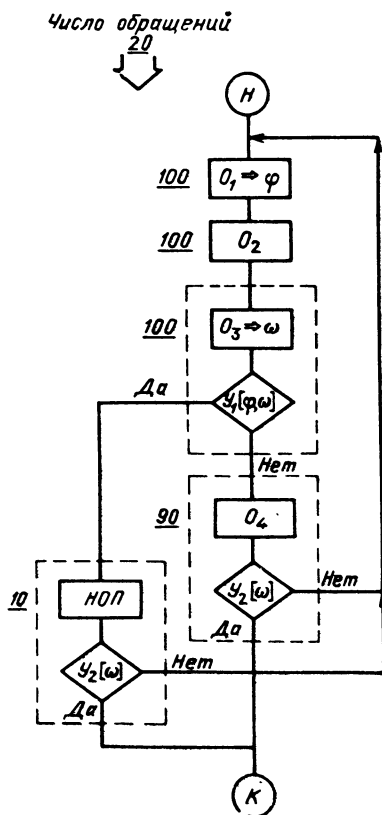


Рис. 2.26. Преобразованная блок-схема алгоритма

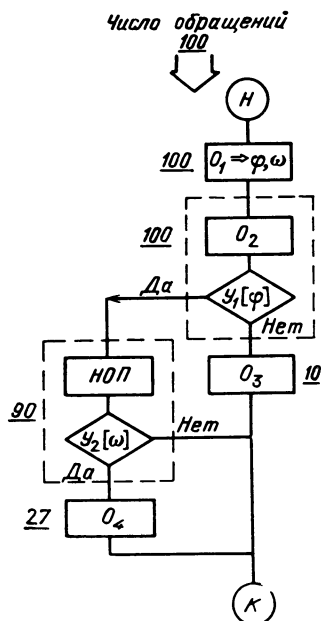


Рис. 2.27. Исходная блок-схема алгоритма

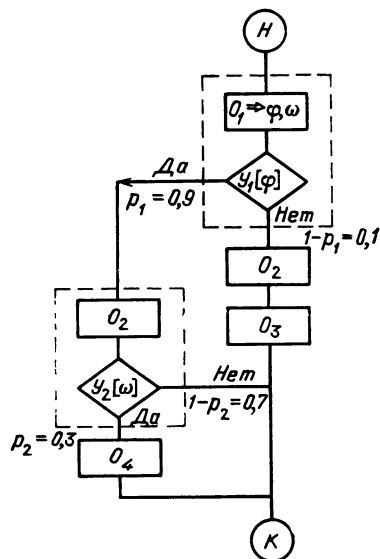


Рис. 2.28. Преобразованная блок-схема алгоритма

программа, которая при тех же 100 вхождениих во фрагмент будет выполнена за 237 тактов (на треть быстрее) при том же (в данном случае) числе хранимых микрокоманд.

Возможность перенесения ОБ со входа УБ на его выходы отнюдь не всегда означает целесообразность такого шага. Если микропрограмма выполняется на типовой структуре с РгС, то объединение в микрокоманду ОБ и УБ, связанных одним условием (φ), невозможно и быстродействие уменьшится вместо того, чтобы увеличиться. Кроме того, целесообразность описанного преобразования определяется числом и типом блоков, следующих за УБ и предшествующих переносимому ОБ, а также соотношением вероятностей исхода из УБ по всем направлениям. Например, если за УБ следуют только ОБ, то число хранимых микрокоманд неизбежно увеличится, а быстродействие в лучшем случае останется неизменным: этот лучший случай заключается в отсутствии УБ на входе переносимого ОБ. И, наоборот, преобразование целесообразно, если за УБ следуют только УБ, а переносимому блоку предшествует ОБ. Все остальные варианты требуют оценки на основе вероятностей переходов.

Теперь коснемся вопросов организации микроподпрограмм. Пусть прикладной алгоритм, представленный блок-схемой с соответствующей детализацией (каждая операция O_i выполняема за один такт в БОД), содержит последовательность из групп повторяющихся операций:

$$\dots O_0, \underbrace{O_1, O_2, O_3, O_4}_{\text{группа 1}}, \underbrace{O_5, O_6, O_7, O_8}_{\text{группа 2}}, \underbrace{O_1, O_2, O_3, O_4}_{\text{группа 3}}, O_9 \dots$$

Многократное повторение этих групп в прикладном алгоритме микропрограммного управления наталкивает на мысль о выделении их в микроподпрограммы. Допустим, так мы и поступили, выделив операции O_1, O_2, O_3, O_4 в микроподпрограмму 1, а операции O_5, O_6, O_7, O_8 — в микроподпрограмму 2. Обращение к микроподпрограмме осуществляется из головной (вызывающей) микропрограммы. Для этого адресная часть микрокоманды должна содержать код инструкции *CJS* для БИС К1804ВУ4, на вход \overline{CC} этой БИС необходимо принудительно подать нулевой потенциал ($\overline{CC} = 0$), соответствующий исходу *TRUE* при выполнении условных инструкций (см. табл. 2.1), а на вход *D* той же БИС должен поступать адрес начала микроподпрограммы (в последующем примере — *M1* или *M2*) из РгМК. Операционное поле микрокоманды при этом может использоваться по назначению (инициализировать в БОД операцию O_i) или содержать код НОП. Переход к адресу *M1* или *M2* будет сопровождаться запоминанием адреса возврата в стеке микросхемы К1804ВУ4. По окончании микроподпрограммы одновременно с последней операцией (O_4 или O_8) осуществляется возврат в головную микропрограмму с помощью инструкции *CRTN* при $\overline{CC} = 0$, которая предписывает извлечь адрес возврата из стека БИС К1804ВУ4. Последовательность обращений к микроподпрограммам будет выглядеть так:

Адрес i	O_0 .	<i>CJS</i>	<i>M1</i> .	$\overline{CC} = 0$.
Адрес $i + 1$	НОП.	<i>CJS</i>	<i>M2</i> .	$\overline{CC} = 0$.
Адрес $i + 2$	НОП.	<i>CJS</i>	<i>M1</i> .	$\overline{CC} = 0$.
Адрес $i + 3$	O_9 .	<i>CONT</i> .		

Сами же микроподпрограммы будут иметь следующий вид:

Адрес <i>M1</i>	O_1 .	<i>CONT</i>	
Адрес <i>M1</i> + 1	O_2 .	<i>CONT</i>	
Адрес <i>M1</i> + 2	O_3 .	<i>CONT</i>	
Адрес <i>M1</i> + 3	O_4 .	<i>CRTN</i>	$\overline{CC} = 0$.
Адрес <i>M2</i>	O_5 .	<i>CONT</i>	
Адрес <i>M2</i> + 1	O_6 .	<i>CONT</i>	
Адрес <i>M2</i> + 2	O_7 .	<i>CONT</i>	
Адрес <i>M2</i> + 3	O_8 .	<i>CRTN</i>	$\overline{CC} = 0$.

Заметим, что в последовательность операций O_0, O_1, \dots, O_9 вклинились НОП, а это означает потерю быстродействия при последовательном обращении к микроподпрограммам. Потерю эту нетрудно устранить, если в микроподпрограммах выделить не максимальные повторяющиеся фрагменты, а включающие на одну операцию меньше. Эта операция будет тогда выполняться в головной микропрограмме вместо НОП. Стоит отметить, что при этом суммарное число ячеек микропрограммной памяти не обязательно возрастет. В нашем примере, выделив группы операций O_2, O_3, O_4 и O_6, O_7, O_8 в микроподпрограммы, получим результат:

Адрес i	O_0 , $CONT$,	
Адрес $i+1$	O_1 , $CJS\ M1$, $\overline{CC}=0$	
Адрес $i+2$	O_3 , $CJS\ M2$, $\overline{CC}=0$.	
Адрес $i+3$	O_1 , $CJS\ M1$, $\overline{CC}=0$.	
Адрес $i+4$	O_0 , $CONT$.	
Адрес $M1$	O_3 , $CONT$.	
Адрес $M1-1$	O_3 , $CONT$.	
Адрес $M1+2$	O_4 , $CRTN$, $\overline{CC}=0$.	
Адрес $M2$	O_4 , $CONT$.	
Адрес $M2+1$	O_7 , $CONT$.	
Адрес $M2+2$	O_8 , $CRTN$, $\overline{CC}=0$.	

В заключение параграфа рассмотрим схемные решения, служащие ускорению многонаправленных ветвлений. Очевидно, любое многонаправленное ветвление может быть представлено как последовательность обычных условных переходов, имеющих два исхода, однако иногда многотактность такого ветвления может вызывать недопустимое уменьшение быстродействия. В таких случаях целесообразно использовать следующий прием: при многонаправленном ветвлении за один такт формировать младшую часть адреса перехода как функцию значений тестируемых условий. Старшая («базовая») часть адреса перехода при этом остается постоянной. Таким образом, в МПП образуется «таблица ветвлений», содержащая в соседних ячейках МПП микрокоманды, соответствующие исходам многонаправленного ветвления.

На рис. 2.29 приведена блок-схема алгоритма, содержащего многонаправленное ветвление. В данном случае для его выполнения потре-

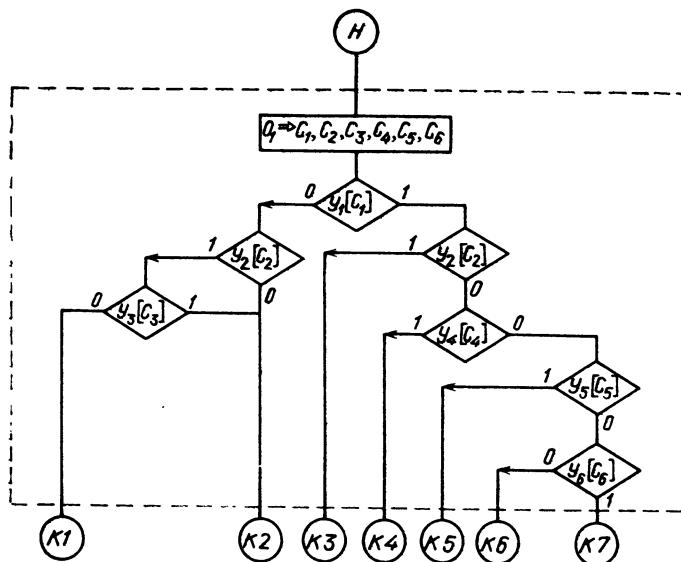


Рис. 2.29. Блок-схема алгоритма с многонаправленным ветвлением

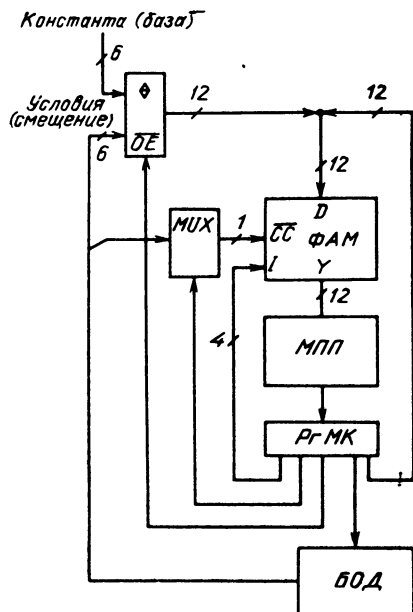


Рис. 2.30. Реализация многонаправленных ветвлений с помощью тристабильного буфера

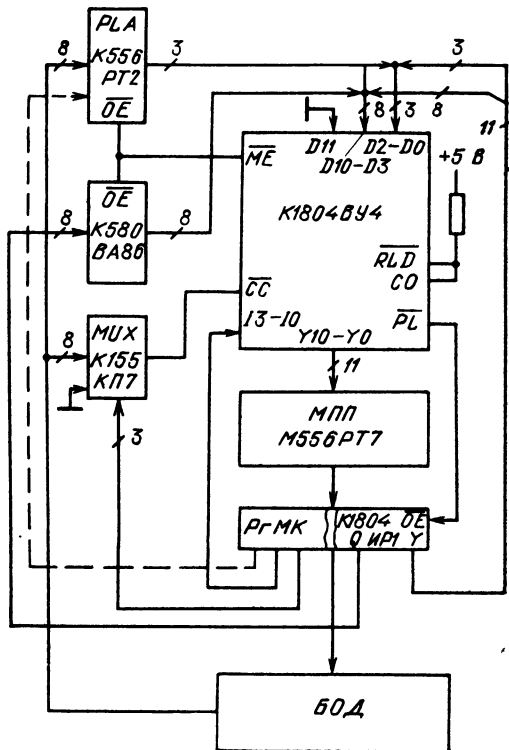


Рис. 2.31. Реализация многонаправленных ветвлений с помощью ПЛМ

бывалось бы до пяти тактов, если не предусмотреть специальных схемных решений. Кстати, отметим, что тестируемые условия могут быть не обязательно результатом операции в БОД: часто это сигналы с двоичных датчиков, поступающие в микроконтроллер и используемые в алгоритме управления объектом.

На рис. 2.30 и 2.31 приведены два схемных решения, позволяющие реализовать многонаправленные ветвления (см. рис. 2.29). за один такт. Первая схема более проста: через тристабильный буферный формирователь на вход D микросхемы $K1804BY4$ может поступать адрес, являющийся конкатенацией (соединением) некоторой константы (базы) и значений тестируемых условий (в примере их шесть: $C1-C6$). Разрешение передачи через буфер осуществляется сигналом из РгМК; для этого в микрокоманде предусмотрен специальный разряд. Во всех тактах, кроме того, на котором выполняется многонаправленное ветвление, информация на вход D микросхемы $K1804BY4$ может поступать из РгМК, а мультиплексор условий на входе \overline{CC} использоваться по обычному назначению.

Пусть значение базы равно 000001. Тогда таблица ветвлений будет располагаться с адреса 000001000000 по адрес 000001111111 и занимать $2^6 = 64$ ячейки микропрограммной памяти. Подавляющее их число будет содержать одинаковые микрокоманды из-за того, что далеко не все комбинации условий используются в качестве исходов многонаправленного ветвления: в примере на рис. 2.29 их всего 8 из 64 возможных. Тем не менее все 64 ячейки должны содержать соответствующие микрокоманды, чтобы любому сочетанию значений условий (с учетом безразличных) соответствовало выполнение требуемой операции в БОД согласно исходу многонаправленного ветвления. Например, одинаковые микрокоманды будут записаны в адресах с 000001000000 по 000001100111, что соответствует исходу ветвления при $C1 \quad C2 = 0$, если условие $C1$ поступает на вход D_5 , а $C2$ — на вход D_4 микросхемы K1804ВУ4.

Рост числа тестируемых условий при многонаправленном ветвлении, а также необходимость реализации нескольких многонаправленных ветвлений вынуждает применять более совершенные схемные решения (рис. 2.31). Все они основаны на применении ПЛМ дополнительно к микросхеме K1804ВУ4. Тестируемые условия (не более 16 при использовании одной ПЛМ) подаются на входы микросхемы, а выходы задействуются в качестве младших разрядов адреса для организации таблицы ветвлений, как было описано выше. Если необходимо реализовать несколько многонаправленных ветвлений, то часть входов ПЛМ (не более двоичного логарифма числа таких ветвлений) потребуется для указания того, какое именно ветвление выполняется в текущем такте. С этой целью необходимо выделить и соответствующее поле в РгМК. Каждому многонаправленному ветвлению должна соответствовать своя база, поэтому в отличие от предыдущей схемы значение базы здесь подается из РгМК. Однако дополнительное поле в РгМК не обязательно, если используются регистры K1804ИР1: с выхода Q может передаваться информация, в то время как выход Y находится в состоянии высокого сопротивления при $\overline{PL} = 1$. В приведенной схеме многонаправленные ветвления выполняются при действии инструкции JMAP K1804ВУ4, вызывающей переход по адресу D с $\overline{ME} = 0$. На рис. 2.31 используются только одиннадцать входов D и выходов Y БИС K1804ВУ4 в связи с тем, что микросхемы МПП имеют емкость 2048 ячеек. Восьмивходовый мультиплексор на входе \overline{CS} служит обычным целям, т. е. используется для выполнения условных инструкций. На схеме не изображен РгС, однако при наличии его логика организации многонаправленных ветвлений не меняется.

Таблица ветвлений для алгоритма рис. 2.29 будет содержать не более восьми ячеек (адреса с 00000001000 по 00000001111). Сопоставив значения трех младших разрядов адреса с каждым из путей в блок-схеме алгоритма ветвлений (табл. 2.10), следует составить систему булевых функций, представленных в дизъюнктивной нормальной форме.

Т а б л и ц а 2.10. Кодирование ПЛМ

Значения условий (входы ПЛМ)						Разряды адреса МПМ (выходы ПЛМ)		
C1	C2	C3	C4	C5	C6	D2	D1	D0
0	1	0	X	X	X	0	0	0
0	0	X	X	X	X	0	0	1
1	0	X	1	X	X	0	1	0
1	0	X	0	1	X	0	1	1
1	0	X	X	0	0	1	0	0
1	0	X	X	0	1	1	0	1
1	1	X	X	X	X	1	1	0
0	1	1	X	X	X	1	1	1

Система булевых функций будет иметь вид:

$$D0 = \overline{C1} \cdot \overline{C2} \vee C1 \cdot \overline{C2} \cdot \overline{C4} \cdot C5 \vee C1 \cdot \overline{C2} \cdot \overline{C5} \cdot C6 \vee \overline{C1} \cdot C2 \cdot C3;$$

$$D1 = C1 \cdot \overline{C2} \cdot C4 \vee C1 \cdot \overline{C2} \cdot \overline{C4} \cdot C5 \vee C1 \cdot C2 \vee \overline{C1} \cdot C2 \cdot C3;$$

$$D2 = C1 \cdot \overline{C2} \cdot \overline{C5} \cdot C6 \vee C1 \cdot \overline{C2} \cdot \overline{C5} \cdot C6 \vee C1 \cdot C2 \vee \overline{C1} \cdot C2 \cdot C3.$$

Затем в соответствии с системой булевых функций производится программирование ПЛМ. Стандартная ПЛМ типа K556РТ2 имеет 16 входов, 8 выходов и 48 термов, поэтому может здесь с успехом использоваться.

Данная система булевых функций составлена в предположении, что необходимо реализовать только одно многонаправленное ветвление. В противном случае в функциях будут фигурировать и переменные (разряды РгМК), указывающие конкретное из нескольких ветвлений, и таблиц ветвлений будет тоже несколько.

Если прикладной алгоритм «сплошь» состоит из многонаправленных ветвлений, то может иметь смысл использование ПЛМ в качестве ФАМ вместо К1804ВУ4. Такое решение имеет как достоинства, так и недостатки. К достоинствам

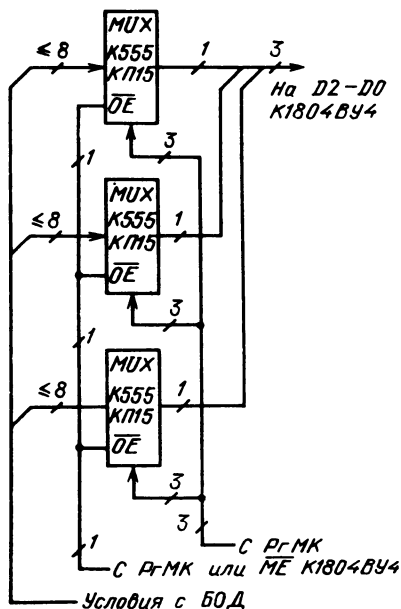


Рис. 2.32. Схема включения мультиплексоров для многонаправленных ветвлений

относятся: сокращение аппаратных затрат за счет отсутствия адресной части в микрокоманде (сокращается РгМК и МПП); не требуются мультиплексор условий и комбинационные схемы вычисления сложных условий из элементарных. Недостатками же являются: необходимость комплексирования нескольких ПЛМ из-за того, что разрядность адреса МПП обычно превышает 8, а также из-за превышения допустимого числа термов, невозможность организации циклов с подсчетом повторений и микроподпрограмм без использования дополнительных аппаратных затрат на счетчик, стек и т. д.

Возможно также применение мультиплексоров с тристабильными выходами для организации многонаправленных ветвлений по значениям нескольких условий одновременно (рис. 2.32). Очевидно, условия должны быть распределены по мультиплексорам так, чтобы одновременно тестируемые были подключены к разным мультиплексорам.

2.4. ТЕХНИКА ПРЕДПОЛОЖЕНИЙ

Техника предположений — это совокупность схемных решений и приемов микропрограммирования, предназначенных для ускорения условных переходов. Ускорение достигается за счет предварительного формирования адреса и выборки наиболее вероятной из двух микрокоманд, следующих за микрокомандой условного перехода до готовности действительного значения условия перехода. По готовности последнего проверяется правильность предварительного перехода, и в случае положительного исхода проверки выполненные действия являются верными. В противном случае их результаты аннулируются и в соответствии с полученным реальным значением условия перехода вычисляется альтернативный адрес, а из МПП считывается действительная следующая микрокоманда.

В первом приближении [10] техника предположений делится на аппаратную и микропрограммную в соответствии с тем, на какие средства возлагается подавляющая часть необходимых действий. Ниже будут рассмотрены оба варианта техники предположений, а также их модификации, учитывающие особенности конкретных алгоритмов микропрограммного управления.

Предлагаемые ниже схемные решения ориентированы на временные соотношения и способы синхронизации, характерные для комплекта БИС К1804. Однако их легко интерпретировать и для других комплектов микропрограммируемых БИС. Поэтому в записях блок-схем алгоритмов и в микропрограммах будут использоваться не конкретные двоичные коды операций, а символические обозначения.

Как было отмечено выше, операции передачи управления делятся на три группы: БП, УП1 и УП2. Первый тип условных переходов соответствует той ситуации, когда тестирование условия производится сразу же за его формированием. Переходы УП2 характеризуются тем, что тестируемое значение условия сформировано заранее и хранится в

РгС микроконтроллера. Необходимость деления условных переходов на два типа станет очевидной из приводимых рассуждений.

Суть аппаратной техники предположений состоит в следующем. При выполнении микрокоманды типа УП1 одновременно с операцией в БОД, вырабатывающей реальное значение тестируемого условия перехода, производится формирование адреса в ФАМ и выборка из МПП следующей микрокоманды, соответствующей наиболее вероятному значению условия. Это наиболее вероятное (т. е. предполагаемое) значение условия определяется при анализе алгоритма и записывается в специальный разряд (*PRED*) микрокоманды на этапе занесения информации в МПП. При выполнении микрокоманды типа УП1 значение *PRED* подается на вход \overline{CC} микросхемы K1804BY4 в первой части такта, когда реальное значение условия (*REAL*) в БОД еще не сформировано. Таким образом, формирование адреса в ФАМ, а затем и выборка микрокоманды из МПП выполняются согласно значению *PRED*, пока формирование *REAL* в БОД не закончено. По готовности значения *REAL* аппаратно производится его сравнение с *PRED*. При совпадении значений предварительно вычисленный адрес следующей микрокоманды оказывается правильным и следующая микрокоманда выбрана верно, поэтому текущий такт завершается. Схема синхронизации вырабатывает нулевой уровень и фронт синхросигнала *SYNC*, поступающего на БОД, ФАМ и РгМК. В результате получается «короткий» такт: при выполнении УП1 удалось избежать увеличения как числа тактов, так и длительности такта. Если же значения *PRED* и *REAL* не совпали, то схема синхронизации растягивает текущий такт (точнее, единичный уровень синхросигнала) на время вычисления в ФАМ правильного адреса согласно *REAL* и выборки правильной следующей микрокоманды. В таком случае такт получается «длинным». Однако выигрыш в быстродействии все равно достигается за счет того, что «длинный» такт короче суммы двух «коротких», которые потребовались бы в типовом варианте организации микропрограммного управления. Итак, при правильных предположениях достигается повышение быстродействия в размере одного такта при каждом выполнении УП1, при неправильных — быстродействие не уменьшается по сравнению с типовыми решениями. При БП и УП2 никаких «предположений» делать не требуется, ибо в первом случае условия не принимаются во внимание, во втором — реальные значения условий получены на предыдущих тактах и хранятся в РгС к моменту их использования.

Из сказанного следует, что на схему аппаратной техники предположений возлагаются следующие функции: коммутация предполагаемого и реального значений условия на вход \overline{CC} микросхемы K1804BY4; сравнение предполагаемого и реального значений условия; управление длительностью такта по результату сравнения.

Вариант реализации аппаратной техники предположений представлен на рис. 2.33. Обсудим его подробнее.

Основными компонентами изображенной схемы являются: БОД; РгС; мультиплексор условий (*DO*), управляемый полем *COND* из РгМК; сумматор по моду-

К 1804ВУ4. При $PRED = REAL$ на выходе элемента $D3$ устанавливается единица, а на выходе $D4$ — нуль. Поскольку $TYPE = 0$ и всем комбинациям счетчика, кроме 1111, соответствует $\overline{RC} = 1$, то на выходе элемента $D5$ устанавливается нуль. По очередному фронту синхросигнала CLK в счетчик произойдет запись начального значения 1001. В результате сигнал $SYNC$ примет единичное значение: по фронту текущий такт будет завершен и начат следующий.

При выполнении микрокоманд типа УП2 $TYPE = 1$, поэтому независимо от значения $Q2$ коммутатор $D6$ передает на вход \overline{CC} реальное значение условия: в данном случае последнее поступает из RrC через мультиплексор $D0$ и управляемый инвертор $D1$. По достижении четвертого периода опорного синхросигнала CLK на выходе элемента $D2$ устанавливается нулевой потенциал независимо от логики функционирования элементов $D3, D4$. В результате сигнал $SYNC$ (выход элемента $D5$) переключается в нулевое состояние, и очередной фронт опорного сигнала CLK вызовет загрузку исходного значения 1001 в счетчик, что знаменует конец текущего такта и начало следующего в момент переключения сигнала $SYNC$ из нуля в единицу.

При выполнении микрокоманд БП $TYPE = 1$, а значения $PRED$ и $REAL$ безразличны. По достижении счетчиком комбинации 1100 ($Q2 = 1$) такт будет завершен.

Итак, выполнение микрокоманд типа УП2, БП и УП1 при условии, что $PRED = REAL$, сопровождается «коротким» тактом синхросигнала $SYNC$, равным четырем периодам опорного, причем три четверти такта сигнал $SYNC$ имеет высокий уровень. Выполнение микрокоманды типа УП1 сопровождается «длинным» тактом только при несовпадении реального значения условия с предполагаемым (наиболее вероятным). Факт несовпадения констатируется при $TYPE = 0$ и $Q2 = 1$ единичным значением на выходе $D4$, обусловленным нулевым выходом элемента $D3$. В результате на комбинациях 1100—1110 счетчика сигнал $SYNC$ остается единичным. По достижении счетчиком комбинации 1111 (а это произойдет на седьмом периоде опорного сигнала CLK) выход \overline{RC} принимает нулевое значение. Выход элемента $D5$ переключается в нулевое состояние, и по очередному фронту сигнала CLK произойдет загрузка счетчика начальным значением 1001. Синхросигнал перейдет из нулевого на единичный уровень, т. е. текущий «длинный» такт закончится и начнется следующий такт, длительность которого определяется типом следующей микрокоманды.

Правила построения микропрограммы по блок-схеме алгоритма аналогичны таковым для типовой неконвейерной структуры микроконтроллера, имеющей $RrMK$. Отличия определяются лишь введением в формат микрокоманды двух дополнительных одноразрядных полей $PRED$ и $TYPE$: первое заполняется наиболее вероятным значением условия перехода при УП1 и безразлично в остальных случаях, второе должно содержать нулевое значение в микрокомандах типа УП1 и единичное в остальных. В табл. 2.11 приведена в символическом виде микропрограмма для фрагмента алгоритма, изображенного на рис. 2.34. На рис. 2.35 показаны временные диаграммы выполнения $(i + 1)$ -й микрокоманды (типа УП1) при совпадении реального значения условия с предполагаемым, а на рис. 2.36 — при несовпадении.

Если конкретный прикладной алгоритм микропрограммного управления не содержит микрокоманд типа УП2, то схемная реализация аппаратной техники предположений может быть упрощена (рис. 2.37).

В микрокоманде выделяется один дополнительный разряд $PRED$, который должен содержать предполагаемое значение условия при условных переходах первого типа и нуль при безусловных переходах. Инверсное значение сигнала управления полярностью POL не имеет принципиального характера: $POL = 0$ бу-

Таблица 2.11. Фрагмент микропрограммы

Адрес	Микрокоманда						
	Операционная часть	Адресная часть					
		$I3-I0$	$D11-D0$	$COND$	POL	$TYPE$	$PRED$
i	O_1	$CONT$	X	X	X	1	X
$i+1$	O_2	CJP	k	Φ	0	0	0
$i+2$	O_4	CJP	k	ω	1	1	X
$i+3$	O_5	$CONT$	X	X	X	1	X
\vdots							
\vdots							
k	O_3	CNT	X	X	X	1	X

дет соответствовать инвертированию условия на элементе $D1$, а $\overline{POL} = 1$ — прохождению условия без инверсии в связи с использованием одинаковых сумматоров по mod 2 с инверсией во всей изображенной на рис. 2.37 схеме. Функционирование схемы при выполнении микрокоманд типа УП1 не отличается от рассмотренного ранее. Выработка схемой синхронизации «короткого» такта при БП обеспечивается подачей кодов $COND$ и \overline{POL} , таких, что на выходе элемента $D1$ устанавливается лог. 0. Вместе с $PRED = 0$ это вызывает переключение выхода элемента $D2$ в лог. 1, а на четвертом периоде опорного сигнала CLK — переключение выхода элемента $D3$ в лог. 0. Поскольку $\overline{RC} = 1$, то на вход \overline{L} счетчика по-

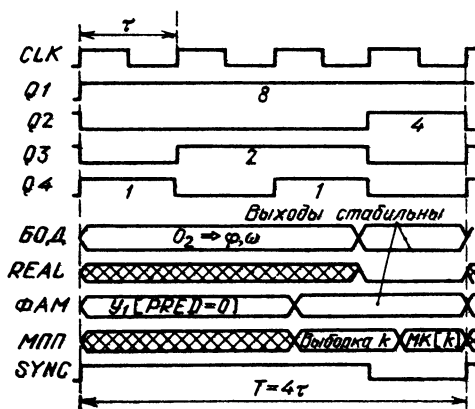
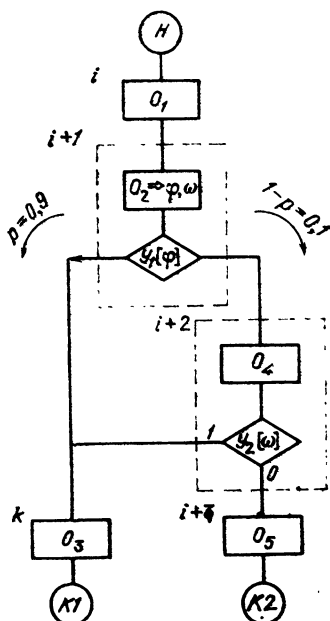


Рис. 2.35. Временные диаграммы при удачном предположении

Рис. 2.34. Фрагмент блок-схемы алгоритма, выполненный с техникой предположений

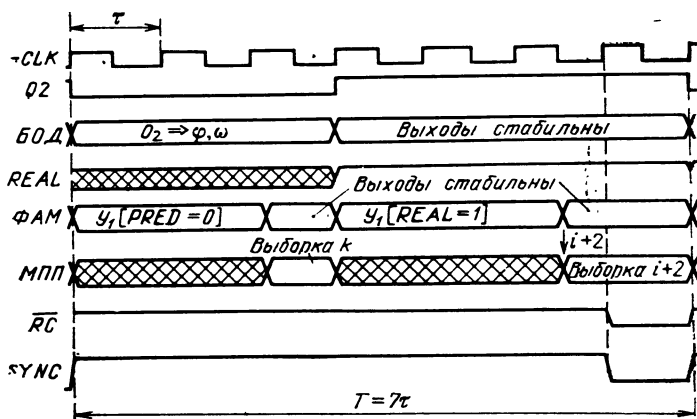


Рис. 2.36. Временные диаграммы при неудачном предположении

ступит лог. 0, разрешающий загрузку счетчика начальным значением 1001. Фрагмент микропрограммы, содержащей БП и УП1, приведен в табл. 2.12. Отметим, что если бы единичное значение условия перехода было более вероятно, то во второй строке таблицы следовало бы указать $PRED = 1$. Пара $COND = 0$, $PRED = 0$ означает формирование короткого такта при БП, причем символ «0» используется для указания того, что нулевое значение имеет не поле $COND$ в микрокоманде, а сигнал на выходе мультиплексора $D0$, управляемого полем $COND$.

Дополнение схемы одним инвертором, а МПП — одним разрядом позволяет задавать две различные длительности «короткого» такта, кратные трем или четырем периодам опорного сигнала CLK (рис. 2.38, табл. 2.13). Введение такой модификации имеет тем больший смысл,

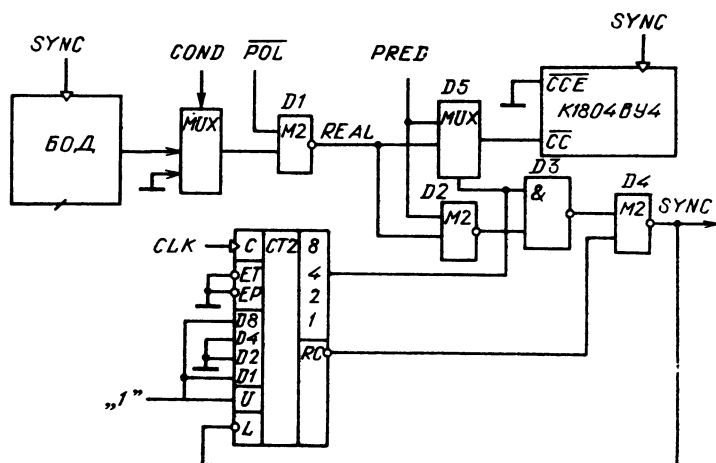


Рис. 2.37. Упрощенная реализация аппаратной техники предположений

Т а б л и ц а 2.12. Фрагмент микропрограммы

Адрес	Микрокоманда					
	Операционная часть	Адресная часть				
		$I3-I0$	$D11-D0$	$COND$	\overline{POL}	$PRED$
i	O_1	$CONT$	X	0	1	0
$i-1$	O_2	CJP	k	4	1	0

чем чаще выполняются микрокоманды с тактом 3Δ и соответственно реже с тактом 4Δ . Например, это могут быть БП и переходы типа УП2. А для того чтобы схемное решение не потребовало дополнительных аппаратных затрат, хорошо, если найдется «лишний» разряд в МПП и «лишний» инвертор на плате. Разумеется, в качестве инвертора могут выступать элементы И—НЕ, ИЛИ—НЕ: сумматор по mod2 с единичным потенциалом на одном из входов и т. д. Если в МПП есть два свободных разряда, то инвертор не понадобится вообще — выходы МПП достаточно будет соединить со входами счетчика (минуя РгМК!). Отметим, что все схемы будут иметь более компактную реализацию, если их элементы (включая мультиплексор условий) выполнить на ПЛМ.

Перейдем теперь к рассмотрению микропрограммной техники предположений. На рис. 2.39 изображен фрагмент блок-схемы алгоритма, содержащий переход типа УП1. Выполнение микропрограммы (табл. 2.14) на типовой структуре микроконтроллера с РгМК и РгС потребовало бы двух тактов в расчете на каждый УП1 из-за введения пустой операции НОП, обусловленной невозможностью формирования и тестирования условия φ в одном и том же такте. Особенность микропрограммной техники предположений состоит в необычном способе построения микропрограммы для фрагментов алгоритма, содержащих

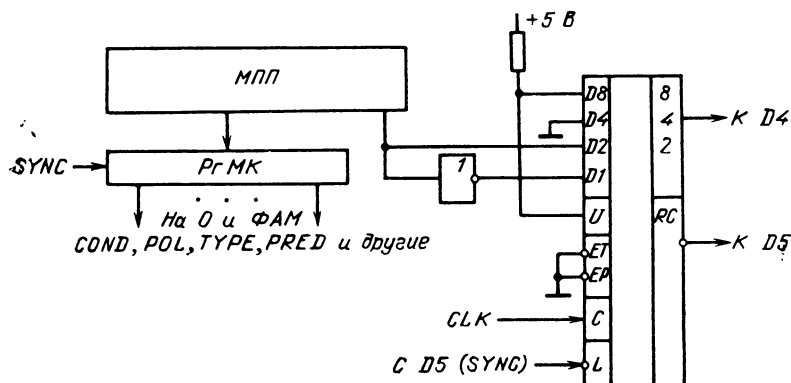
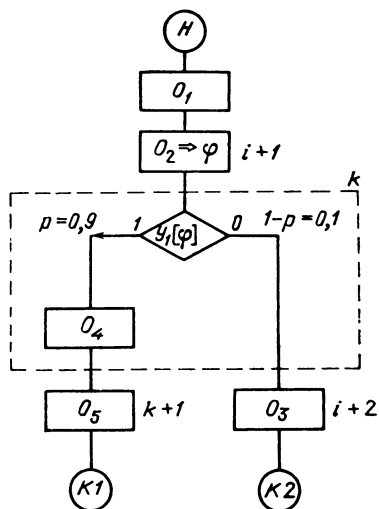


Рис. 2.38. Схема управления длительностью такта

Т а б л и ц а 2.13. Варианты тактирования

Ч а с	T_1 , нс	T_2 , нс	T_3 , нс
50	150	200	350
55	165	220	385
60	180	240	420
65	195	260	455
70	210	280	490
75	225	300	525
80	240	320	560

Рис. 2.39. Фрагмент блок-схемы алгоритма после разметки для микропрограммной техники предположений



переходы типа УП1. В результате среднее число тактов при выполнении УП1 сокращается пропорционально достоверности априорных предположений о направлении перехода.

Пусть вероятность исхода из блока $Y_1[\varphi]$ к блоку O_5 оценивается величиной $p = 0,9$. Тогда процесс перехода от блок-схемы алгоритма к микропрограмме (табл. 2.15) подчиним следующим правилам. Переход типа УП1 (совокупность блоков $O_2 \Rightarrow \varphi$ и $Y_1[\varphi]$) разобьем на две микрокоманды, имеющие адреса $i+1$ и k (рис. 2.39). В операционную часть первой микрокоманды запишем код, соответствующий операции O_2 , а в адресную — код CJS при $COND = 0$, что вызовет БП к адресу k с занесением в стек адреса $i+1$.

Т а б л и ц а 2.14. Фрагмент микропрограммы

Адрес	Микрокоманда				
	Операционная часть	Адресная часть			
		$I3 \dots I0$	$D11 \dots D0$	$COND$	POL
$i+1$	O_1	$CONT$	X	X	X
$i+2$	O_2	$CONT$	X	X	X
$i+3$	НОП	CJP	k	φ	1
\vdots	O_3	$CONT$	X	X	X
\vdots					
k	O_4	$CONT$	X	X	X
$k+1$	O_5	$CONT$	X	X	X

Т а б л и ц а 2.15. Фрагмент микропрограммы

Адрес	Микрокоманда					
	Операционная часть	Адресная часть				
		$I3 \dots I0$	$D11 \dots D0$	COND	POL	ENS
i	O_1	CONT	X	X	X	X
$i + 1$	O_2	CJS	k	0	0	1
$i + 2$	O_3	CONT	X	X	X	1
\vdots						
k	O_4	CRTN	X	φ	0	0
$k + 1$	O_5	LOOP или CJPP	X или m	0	0	1

$+ 2$ одновременно с выполнением операции O_2 . Таким образом осуществляется переход к наиболее вероятной следующей микрокоманде, содержащей операцию O_4 одновременно с вычислением действительного значения условия φ . На границе текущего и следующего тактов действительное значение условия φ попадет в РгС. Перед выполнением операции O_4 на следующем такте производится проверка действительного значения на равенство единице. Если совпадение имеет место, то сигналом синхронизации разрешается загрузка РгМК₀ и выполнение O_4 в БОД. Адресная часть РгМК в этот момент содержит код CRTN и COND = φ . Поэтому при $\varphi = 1$ БИС К1804ВУ4 выполняет переход к следующему по порядку адресу $k + 1$. Ненужный более адрес $i + 2$ при этом остается в стеке. Следовательно, необходимо позаботиться об извлечении его из стека; это действие и осуществляется $(k + 1)$ -й микрокомандой наряду с операцией O_5 (см. табл. 2.15). Инструкция LOOP микросхемы К1804ВУ4 при $\overline{CC} = 0$ вызывает переход к следующему адресу $(k + 2)$ с выталкиванием адреса возврата из стека. Если из адреса $k + 1$ необходимо произвести безусловный переход к адресу $m \neq k + 2$, то вместо LOOP можно использовать инструкцию CJPP с $\overline{CC} = 0$: произойдет переход к адресу m с одновременным выталкиванием адреса $i + 2$ из стека.

В случае несовпадения реального значения φ с единицей при выполнении k -й микрокоманды запрещается тактирование РгМК₀ и БОД и в РгМК₀ остается предыдущий код. Тактирование же ФАМ и РгМК_а не прекращается. Поэтому будет выполнена инструкция CRTN при $\overline{CC} = 0$, вызывающая переход по адресу $i + 2$, выталкиваемому из стека, и в РгМК будет занесена следующая (теперь уже правильная, т. е. соответствующая условию $\varphi = 0$) микрокоманда, расположенная в МПП по адресу $i + 2$ и предписывающая операцию O_3 . Таким образом, вместо одного такта выполнение УП1 потребует двух тактов, что не

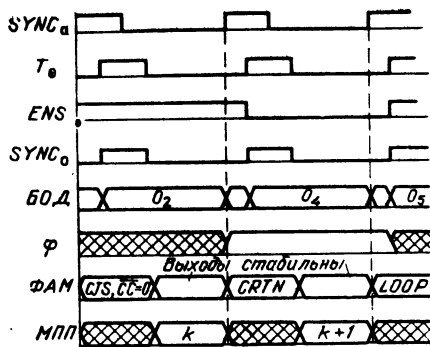
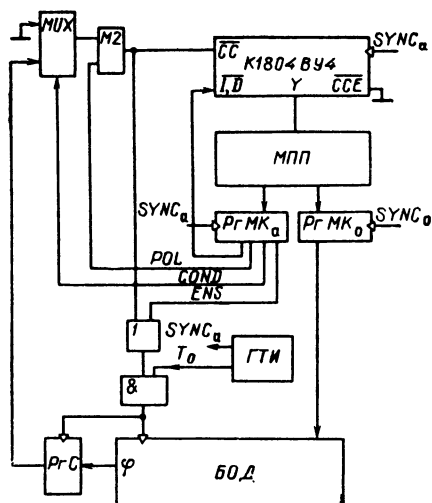


Рис. 2.41. Временные диаграммы при удачном предположении

←
Рис. 2.40. Реализация микропрограммной техники предположений

превышает временные затраты в обычной типовой структуре микроконтроллера.

При выполнении БП и переходов типа УП2 никаких «предположений» делать не требуется и синхронизация БОД сигналом *SYNC*, должна производиться безусловно. Для этого разрешающий сигнал *ENS* в соответствующей микрокоманде (УП2, БП) имеет единичное значение, в то время как в микрокомандах *CRTN* типа УП1 — нулевое.

Вариант схемной реализации представлен на рис. 2.40. Временные диаграммы для случаев совпадения и несовпадения реального значения φ с предположением ($\varphi = 1$) при выполнении УП1 (рис. 2.39) показаны на рис. 2.41 и 2.42 соответственно. Генератор тактовых импульсов (ГТИ), частота которого задается кварцевым резонатором, вырабатывает две синхросерии $SYNC_a$ и T_o с периодом, определяемым максимальными задержками в схеме микроконтроллера. Сигнал $SYNC_a$ используется для синхронизации ФАМ и $PrMK_a$. Сигнал T_o используется для формирования синхросерии $SYNC_o$ в совокупности с полем ENS и значением условия, поступающего на вход \overline{CC} , так, как было описано выше.

Некоторые особенности возникают при использовании микропрограммной техники предположений, если (рис. 2.43) наиболее вероятным является нулевое, а не единичное значение условия φ : $m \neq i + 2$; $n \neq k + 1$. Микропрограмма в данном случае претерпит следующие изменения (табл. 2.16). Для хранения адреса возврата (m) здесь используется не стек, а поле $D11 - D0$ РгМК. Предполагаемый адрес перехода заносится в регистр адреса БИС К1804ВУ4. Загрузка его осуществляется из поля $D11 - D0$ микрокоманды инструкцией $LDCT$. Таким образом, в результате тестирования реального значения условия φ k -й микрокомандой (JRP) будет произведен переход по адресу n с разрешением синхронизации БОД при $\overline{CC} = \overline{\varphi} = 1$ или по адресу возврата m без изменения состояния РгМК₀ и БОД при $\overline{CC} = \overline{\varphi} = 0$.

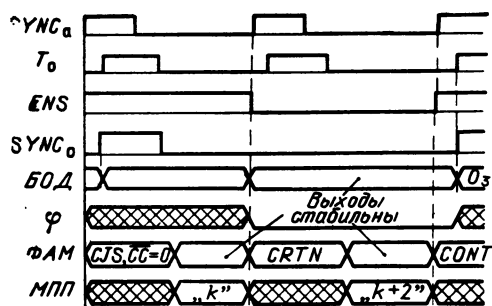


Рис. 2.42. Временные диаграммы при неудачном предположении

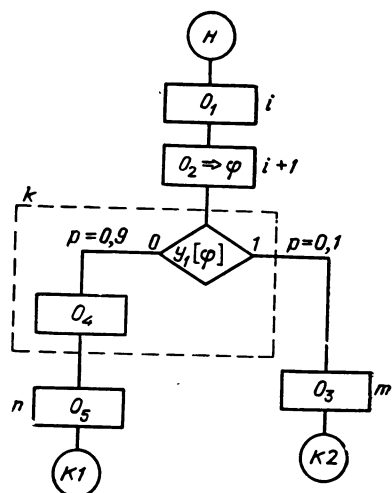


Рис. 2.43. Фрагмент с усложненной расстановкой адресов

Отметим, что хотя реализация микропрограммной техники предположений требует меньшего числа микросхем, чем аппаратная, заметно усложняется процесс микропрограммирования. А это влечет дополнительные трудности как при написании микропрограмм, так и при их отладке.

Все рассмотренные выше схемные решения и приемы микропрограммирования ориентированы только на двухуровневую конвейерную об-

Таблица 2.16. Фрагмент микропрограммы

Адрес	Микрокоманда					
	Операционная часть	Адресная часть				
		$I_3 \cdot I_0$	$D_{11} \cdot D_0$	COND	POL	ENS
i	O_1	LDCT	n	X	X	1
$i+1$	O_2	CJP	k	0	0	1
\vdots						
k	O_4	JRP	m	φ	1	0
\vdots						
m	O_3	CONT	X	X	X	1
\vdots						
n	O_5	CONT	X	X	X	1

работку микрокоманд в предположении того, что $t_{\text{БОД}} \approx t_{\text{ФАМ}} + t_{\text{МПП}}$. При $t_{\text{БОД}} \approx t_{\text{ФАМ}} \approx t_{\text{МПП}}$ большее быстродействие микроконтроллера обеспечивается трехуровневой конвейерной обработкой микрокоманд. Последняя, разумеется, требует специальных схемных решений и приемов микропрограммирования для реализации принципов техники предположений. Следует отметить, что микропрограммная техника предположений в трехуровневом конвейере микрокоманд может применяться для ускорения условных переходов обоих типов.

2.5. СПОСОБЫ СОКРАЩЕНИЯ АППАРАТУРНЫХ ЗАТРАТ

Результатом попыток систематизации способов сокращения аппаратных затрат является рис. 2.44. Как правило, при выборе схемных решений не удастся одновременно повысить быстродействие (производительность) микропрограммного контроллера и сократить аппаратные затраты. В этом смысле сокращение аппаратных затрат является альтернативой повышению быстродействия и может осуществляться оптимизацией (но уже в ином смысле) БОД и БМУ, схем синхронизации и представления алгоритма микропрограммой.

Прежде чем приступить к подробному обсуждению некоторых схемных решений, прокомментируем те направления на рис. 2.44, которые ниже не будут рассматриваться детально.

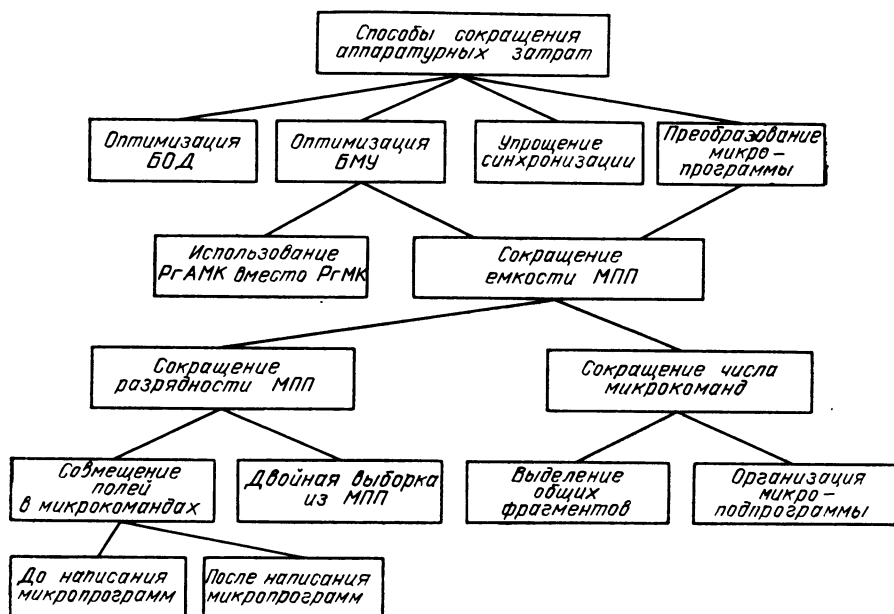


Рис. 2.44. Систематизация способов сокращения аппаратных затрат

Оптимизация схемы операционной части подразумевает сокращение аппаратных затрат за счет уменьшения числа используемых в БОД микросхем и связей между ними. Примерами могут служить мультиплексирование во времени передач информации между модулями БОД по общей внутриспроцессорной шине, микропрограммное выполнение операций умножения и деления без использования схемных модификаций, позволяющих достичь большего быстродействия за счет введения дополнительных микросхем в БОД.

Предельное упрощение схемы синхронизации означает отсутствие каких-либо возможностей по управлению длительностью такта; последняя выбирается исходя из максимального времени распространения сигналов при выполнении микрокоманд. В пределе схема синхронизации сокращается до простейшего генератора, реализуемого на инверторах с соответствующей времязадающей цепью на основе кварцевого резонатора.

Преобразованием микропрограммы сопровождаются все модификации структуры микроконтроллера, ведущие к сокращению аппаратных затрат за счет увеличения числа тактов, необходимых для выполнения заданного алгоритма. В ряде случаев преобразование микропрограммы, позволяющее сократить требуемую емкость МПП, может рассматриваться и как самостоятельный путь уменьшения аппаратных затрат. Одним из наиболее общих приемов является выделение повторяющихся участков вычислений в микроподпрограммы и общие фрагменты. Отличие первых от вторых заключается в способе обращения к ним и выхода из них. Обращение к микроподпрограмме сопровождается запоминанием адреса возврата (следующего в вызывающей микропрограмме), например, в стеке. По окончании микроподпрограммы выполняется возврат в вызвавшую ее микропрограмму по адресу из стека. К последовательности же микрокоманд, выделенной в общий фрагмент, обращение производится без запоминания адреса возврата, и выход из общего фрагмента осуществляется путем специального формирования адреса выхода.

Очевидный выигрыш в аппаратуре дает использование РгАМК, имеющего разрядность не более 12, вместо РгМК, число разрядов которого составляет, как правило, от 32 до 96. Отметим, что для неконвейерной обработки микрокоманд с постоянной длительностью такта ($t_{\text{БОД}} + t_{\text{фам}} + t_{\text{мпп}}$) этот способ не уменьшает быстродействия микроконтроллера из-за того, что число и длительность тактов остаются идентичными, так же как и правила построения микропрограмм.

Весьма широко применяется способ сокращения разрядности МПП (т. е. формата микрокоманды) за счет совмещения полей микрокоманды. Уместно различать два вида совмещения полей в зависимости от того, на каком этапе разработки микропрограммы оно осуществляется.

1. Совмещение полей до написания микропрограмм делает невозможным их одновременное использование по назначению в каждом такте работы микропрограммного контроллера. Так, если в микрокоманде совмещены поля адреса перехода для БИС К1804ВУ4 и константы

для БИС К1804ВС2, то невозможно в одном такте выполнять, например, сложение с константой и условный переход (*CJP*) по адресу из РгМК. Итак, потенциально подобное совмещение увеличивает число тактов при выполнении алгоритма, т. е. уменьшает быстродействие микроконтроллера. Однако совсем не обязательно это потенциальное уменьшение быстродействия становится реальным: часто удается «безболезненно» разнести по разным тактам операции с константами и условные переходы без потери быстродействия. Решение о целесообразности этого шага принимается в каждом конкретном случае исходя из характеристик прикладных алгоритмов микропрограммного управления.

2. Совмещение полей после написания микропрограмм означает по существу исключение из микрокоманды тех разрядов, значения которых совпадают (возможно, после доопределения безразличных состояний или с инверсией) между собой во всех ячейках МПП, например:

Разряд i	Разряд j	Результат (разряд i)
0	X	0
X	X	X
X	0	0
1	1	1
0	0	0

Здесь разряд j из микрокоманды можно исключить, доопределив безразличные значения X в i -м разряде микрокоманды до совпадения с бывшим j -м, и управляемые точки схемы, ранее бывшие подключенными к j -му разряду РгМК, подключить к i -му разряду. До написания микропрограммы выполнить подобную процедуру невозможно, ибо еще неизвестны конкретные сочетания кодов в микрокомандах. На быстродействии микроконтроллера такое совмещение не отражается, и особенно полезным оно оказывается в тех случаях, когда разрядность микрокоманды удается довести до кратной восьми (разрядности БИС ППЗУ), например с 57 до 56.

Наиболее удобными микросхемами для реализации МПП являются быстродействующие ППЗУ с тристабильным выходом типа М556РТ7, имеющие организацию 2048×8 . Если объем микропрограмм не превышает 1024 микрокоманд и при этом желательно сократить аппаратные затраты ценой некоторого уменьшения быстродействия, то выгодно применять двойную выборку микрокоманд из МПП. Суть ее проста: «длинная» микрокоманда (например, 80-разрядная) записывается в две соседние ячейки 40-разрядной МПП «по половинкам». Перед выполнением каждой микрокоманды обе ее половинки должны быть прочитаны из МПП и хотя бы одна из них занесена в буферный РгМК для того, чтобы на БОД и ФАМ подать все необходимые управляющие сигналы. Таким образом, требуемое число корпусов МПП сократится вдвое: с десяти до пяти.

Если 80-разрядную микрокоманду не тривиально разделить пополам, а сгруппировать ее разряды в две 40-разрядные полумикрокоманды

с учетом задержек распространения сигналов, то суммарную длительность такта удастся сократить. Очевидно, для этого разряды, управляющие схемами, обладающими наиболее критическими задержками, должны быть сгруппированы в той половине микрокоманды, которая выбирается первой и запоминается в буферном РгМК во время выборки из МПП второй половины микрокоманды. В результате распространение сигналов по критическим путям начнется заранее, т. е. перед окончанием выборки второй половины микрокоманды.

Еще одно замечание состоит в том, что разрядность буферного РгМК не обязательно должна быть точно равна половине разрядности микрокоманды. Если исходная микрокоманда 56-разрядная, то при использовании 32-разрядной МПП достаточно иметь 24-разрядный буферный РгМК, а вторую (полную 32-разрядную часть микрокоманды) подавать на БОД непосредственно с выхода МПП.

Наличие РгС в структурах с двойной выборкой позволяет осуществлять конвейерную обработку микрокоманд, т. е. формировать адрес следующей микрокоманды в ФАМ одновременно с выполнением текущей в БОД.

Сказанное поясняют примеры структур с двойной выборкой микрокоманд и временные диаграммы на рис. 2.45 – 2.48. Основное отличие первых двух структур заключается в наличии (отсутствии) РгС, а стало быть, в конвейерной (последовательной) обработке микрокоманд. Кроме того, в первой используются все разряды обеих половин микрокоманды, а во второй — только 56 разрядов, что позволило сократить до 24 разрядность буферного РгМК.

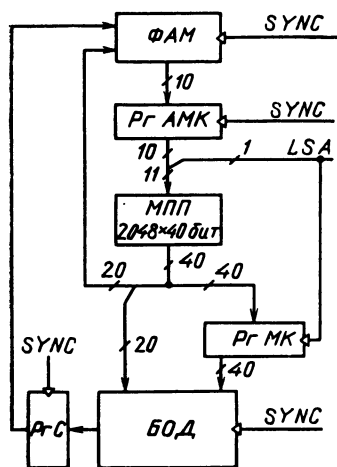


Рис. 2.45. Схема с двойной выборкой и буферизацией операционной части микрокоманды

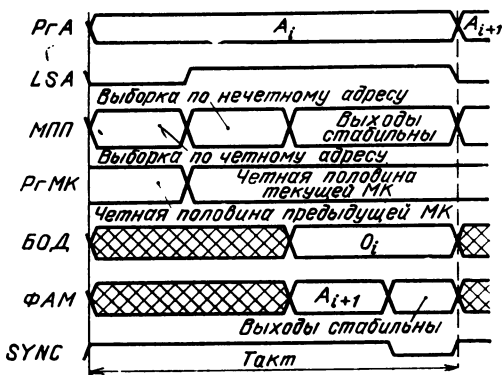


Рис. 2.46. Временные диаграммы двойной выборки

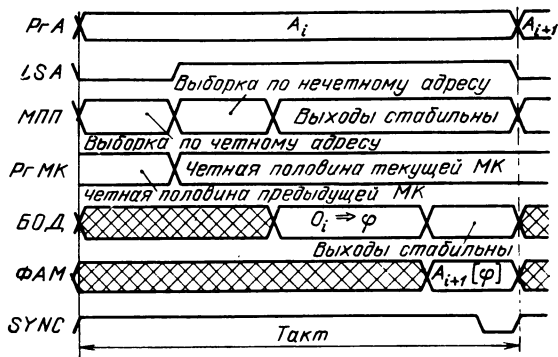
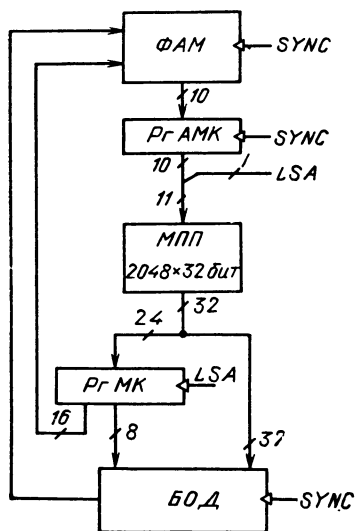


Рис. 2.48. Временные диаграммы двойной выборки

Рис. 2.47. Схема с двойной выборкой и буферизацией адресной части микрокоманды

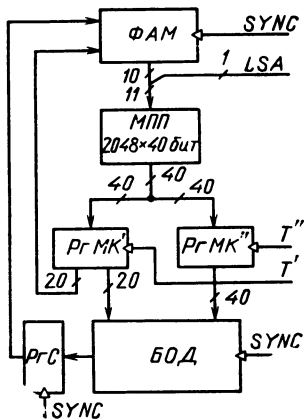


Рис. 2.49. Схема с двойной выборкой и буферизацией всей микрокоманды

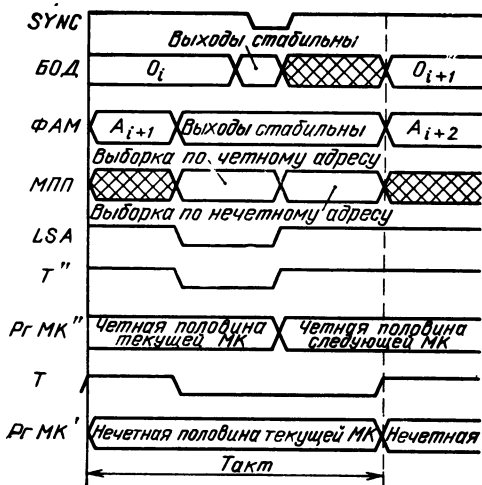


Рис. 2.50. Временные диаграммы двойной выборки

Длительность такта при конвейерной обработке микрокоманд $T = \max \{t_{\text{ФАМ}}, 2t_{\text{МПП}} + t_{\text{БОД}}\}$, а при неконвейерной с постоянной длительностью такта $T = 2t_{\text{МПП}} + t_{\text{БОД}} + t_{\text{ФАМ}}$.

Если быстродействие микроконтроллера с такой длительностью такта оказывается недостаточным и на печатной плате есть место для еще одного буферного РгМК с разрядностью не более половины разрядности микрокоманды, то рекомендуется выбрать третью структуру (рис. 2.49, 2.50). Выигрыш в быстродействии появляется из-за того, что в реальных схемах $t_{\text{МПП}} < t_{\text{БОД}}$, а почти всегда и $2t_{\text{МПП}} \leq t_{\text{БОД}}$. Распараллеливание операции с функционированием ФАМ, и двойной выборкой из МПП позволяет сократить длительность такта приблизительно до $T \approx \max \{t_{\text{БОД}}, t_{\text{ФАМ}} + 2t_{\text{МПП}}\}$ в случае конвейерной обработки микрокоманд. При последовательной обработке микрокоманд с постоянной длительностью такта эту структуру выбирать не стоит, ибо по быстродействию она будет эквивалентна двум предыдущим, а по затратам превышает их из-за того, что разрядность РгМК (или части) больше разрядности РгАМК. Число тактов при выполнении заданного алгоритма для приведенных структур одинаково: в каждом случае оно определяется числом и взаимосвязью ОБ и УБ в конкретной конфигурации блок-схемы алгоритма, а также распределением вероятностей переходов из УБ по всем возможным направлениям.

Глава 3.

ФУНКЦИОНАЛЬНОЕ ДИАГНОСТИРОВАНИЕ БЛОКОВ МИКРОПРОГРАММНОГО УПРАВЛЕНИЯ

3.1. ОБЪЕКТ И ЗАДАЧИ ДИАГНОСТИРОВАНИЯ

Очевидно, что разработка эффективной системы диагностирования, предназначенной для ответа на вопрос «правильно ли работает объект», предполагает наличие у диагноста предельно ясных представлений об особенностях функционирования этого объекта на соответствующем уровне его детализации. Такие же представления должен иметь и разработчик объекта. Если же проектирование системы диагностирования должно осуществляться параллельно с проектированием диагностируемого объекта для удовлетворения некоторых совокупных требований к конечному продукту, то идеальным вариантом исполнителя является разработчик-диагност в едином лице или в едином коллективе. Такая ситуация характерна прежде всего для микропроцессорных средств. Разработка системы диагностирования, встраиваемой в микропрограммируемый процессор или контроллер (что и является предметом дальнейшего обсуждения), не только сопутствует, но и оказывает влияние на выбор структуры микропроцессора, построение микропрограммы, синхронизацию. Обратное влияние еще более сильно: выбор структуры микропроцессора с целью достижения заданных показателей быстродействия (производительности) и аппаратурных затрат налагает ограничения на варианты схем контроля, применение которых допустимо в рамках соответствующих

показателей. Распределение фиксированных ресурсов (аппаратурных, временных, стоимостных и т. д.) между контролирующими и контролируемыми схемами неоднозначно: за счет ухудшения первых можно улучшить вторые и наоборот. Например, самые незамысловатые схемы контроля реализуются меньшим числом микросхем, а значит, на печатной плате микропроцессора останется больше места для реализации его «основного назначения». Это, скорее всего, позволит ввести средства для более эффективного выполнения необходимых пользователю функций. Напротив, более сложные схемы контроля будут и более громоздкими, однако позволят с большей достоверностью судить об исправности процессора. Очевидно, использование слишком большого числа малых микросхем для контроля устройств, основу которых составляют БИС, противостоит естественно. Именно с позиций сравнительной оценки следует подходить к обсуждению описываемых ниже структурно-алгоритмических решений, обзор которых, безусловно, не претендует на исчерпывающую полноту.

Какие именно аспекты диагностирования (рис. 3.1) будут рассмотрены? Согласно традиционной классификации и в зависимости от назначения на систему диагностирования могут возлагаться следующие задачи: обнаружение факта неправильной работы устройства (без выяснения причины); локализация дефекта, т. е. определение причины неправильной работы; нейтрализация дефекта, т. е. устранение по-

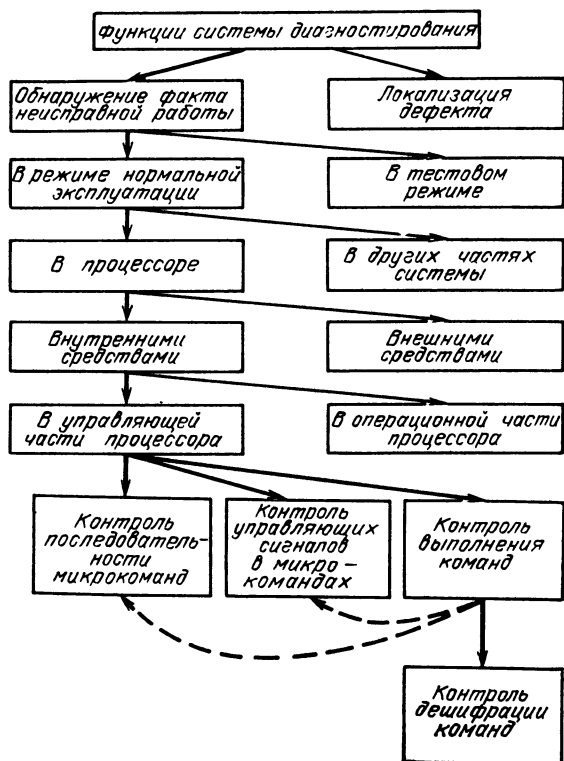


Рис. 3.1. К определению объекта исследования

следствий, вызываемых причиной неправильной работы, позволяющее вновь организовать правильную работу устройства за счет использования некоторых резервных ресурсов (временных или аппаратурных). Остановимся лишь на задаче обнаружения факта неправильной работы: при построении модульных одноплатных микропроцессоров и микроконтроллеров, как правило, нет ни необходимости, ни возможности введения средств локализации и устранения дефекта на печатной плате. Прямой смысл имеет введение структурного резервирования на уровне одноплатных изделий, являющихся готовыми типовыми элементами замены. При этом на схемы контроля, расположенные в пределах платы, возлагается лишь задача обнаружения факта неисправной работы с передачей соответствующего сигнала во внешнюю систему коммутации (если используется динамическое структурное резервирование). Такой контроль осуществляется в режиме нормальной эксплуатации (т. е. при подаче рабочих, а не тестовых воздействий) непрерывно, поэтому в отличие от тестового режима эффективно обнаруживаются кратковременные сбои.

Если при использовании однокристальных микропроцессоров вводимые схемы контроля являются внешними по отношению к микропроцессору, то при построении микропрограммируемых микропроцессоров схемы контроля встраиваются непосредственно в процессор наряду с БИС операционной и управляющей частей.

Какие элементы микропроцессора контролируются и как? Здесь уместно вспомнить о разделении структуры микропроцессора на операционную и управляющую части. Операционная часть, т. е. БОД, осуществляет все необходимые действия над данными (операндами), поступающими в микропроцессор извне, что в итоге приводит к получению данных (результатов), передаваемых в обратном направлении. Процесс обработки данных организуется управляющей частью, т. е. БМУ, основными элементами которой являются МПП и ФАМ. Действенные способы проверки правильности функционирования БОД в режиме нормальной эксплуатации предполагают значительную аппаратную избыточность и не отличаются от распространенных. Наиболее простой из них — дублирование. Микропроцессорной спецификой является невозможность контролировать элементы и тракты передачи информации внутри БИС (скажем, правильность часто встречающихся операций типа регистр — регистр) до тех пор, пока результат не будет выведен наружу. В перспективе средства контроля будут во все возрастающей степени реализованы непосредственно на кристалле БИС, однако разработчик сегодняшних систем (и в частности, на основе серии K1804) не может довольствоваться перспективами. Что касается контроля БМУ, то в 80-х годах здесь произошел «всплеск» активности исследователей и был предложен ряд изящных схемных решений. Ниже обсудим вопросы организации контроля именно управляющей части микропрограммируемых микропроцессоров.

Две задачи контроля (последовательности микрокоманд и корректности управляющих сигналов) совпадают для микроконтроллеров с

одним (микропрограммным) уровнем управления и для процессоров команд (см. рис. 3.1). Этим объясняется значительная общность схемных решений и способов их контроля. Схемные решения и способы контроля правильности функционирования БМУ можно классифицировать по типу контролируемых фрагментов алгоритма микропрограммного управления. Контроль может осуществляться «изолированно» на каждом такте (при выполнении каждой микрокоманды) независимо от последовательности микрокоманд или пофрагментно с проверкой корректности переходов между микрокомандами. В первом случае традиционным решением является введение в каждую микрокоманду контрольного кода, с которым производится сравнение кода, полученного сверткой управляющих сигналов микрокоманды по специальному правилу. Наиболее простой вариант — проверка на четность (нечетность), когда контрольный код имеет минимальную разрядность (1 бит). Но при этом микрокоманда с ошибками в четном числе разрядов будет трактоваться схемой контроля как правильная. Увеличение разрядности контрольного кода позволяет обнаружить большее число ошибочных ситуаций. Контроль правильности передачи информации с помощью специальных кодов не является специфическим применительно к МПП: с этим вопросом можно ознакомиться по обширной литературе, посвященной проблемам помехоустойчивой передачи данных и теории кодирования.

Считается, что БМУ работает правильно, если:

1. Соблюдается правильная (корректная) последовательность микрокоманд.
2. Управляющие сигналы подаются каждой микрокомандой без искажений.

Иными словами, чтобы вся последовательность микрокоманд была корректной, каждая микрокоманда должна формировать корректный адрес следующей микрокоманды, а из МПП должна производиться выборка микрокоманды, соответствующей сформированному корректному адресу. Таким образом, в БОД должны поступать (через РгМК или непосредственно с выхода МПП) именно те управляющие сигналы, которые соответствуют корректной микрокоманде.

Если прикладной алгоритм пользователя полностью или частично реализован на уровне команд (а не полностью на уровне микрокоманд), то в функции БМУ входит интерпретация команд микропрограммами. Поэтому определение правильного функционирования БМУ дополняется следующим: каждая команда должна активизировать последовательность микрокоманд, соответствующую именно этой команде. Очевидно, для этого необходимо: осуществить правильный переход к начальному адресу микропрограммы (дешифрацию), соответствующему данной команде, и соблюсти правильную последовательность микрокоманд этой микропрограммы (см. рис. 3.1). При этом управляющие сигналы каждой микрокомандой должны подаваться без искажений.

Попрагментный контроль микропрограмм может выполняться по принципу предшественник—последователь на двух соседних тактах. При этом производится проверка корректности соответствия микрокоманды-последователя микрокоманде-предшественнику с помощью сравнения присвоенных им «ключей». Корректные микрокоманды-последователи определяются для всех микрокоманд-предшественников согласно фиксированному прикладному алгоритму микропрограммного управления.

Наконец, контроль правильности выполненной последовательности микрокоманд может осуществляться по окончании некоторого фрагмента микропрограммы. Как правило, это линейный, т. е. не содержащий условных переходов, фрагмент микропрограммы. Однако в [11, 12] обсуждаются и возможности контроля фрагментов, включающих условные переходы. В зависимости от допустимых аппаратурных затрат на схемы контроля и требуемой достоверности проверку корректности выполнения фрагмента можно осуществлять с помощью сверки с эталоном контрольных адресов, контрольных сумм или контрольных сигнатур трасс. Последний способ обладает наибольшей обнаруживающей способностью. Отметим, что здесь применяется не «последовательный» сигнатурный анализ, когда свертка последовательного кода выполняется со скоростью 1 бит/такт (подробное описание можно найти в [13, 14]), а «параллельный»: в каждом такте производится свертка N информационных разрядов в N -разрядную сигнатуру. Параллельный сигнатурный анализатор представляет собой сдвигающий регистр с обратными связями. На входе триггера каждого разряда регистра находится сумматор по mod 2, выполняющий одноименную функцию над сигналами из предыдущего разряда и соответствующим битом параллельного кода, поступающего на анализатор. На вход сумматора по mod 2 первого разряда поступает сигнал обратной связи с последующих разрядов (например, в 8-разрядном анализаторе [15], с 7-, 6- и 2-го разрядов), получаемый их суммированием также по mod 2.

Как будет показано ниже, контрольную сигнатуру трассы (фрагмента) микропрограммы можно хранить либо в специальном ППЗУ, либо в той же памяти, что и сами микропрограммы. Порядок накопления сигнатуры и сверки ее с эталоном также может быть различен — схемные решения будут описаны ниже. Однако рассмотрение начнем с более простых схем, осуществляющих контроль последовательности выполняемых микрокоманд или их адресов.

3.2. КОНТРОЛЬ ПОПАРНОГО СЛЕДОВАНИЯ МИКРОКОМАНД

Контроль последовательности микрокоманд осуществляется по принципу предшественник-последователь [11] для каждой пары последовательно выполняемых микрокоманд. Парная проверка правильности следования микрокоманд производится с помощью сравнения специальных контрольных символов — ключей, вводимых в каждую микрокоманду по правилам:

На рис. 3.3 приведен фрагмент блок-схемы алгоритма микропрограммного управления, а на рис. 3.4 представлена временная диаграмма его выполнения с ориентацией на рассматриваемую структуру (см. рис. 3.2). В символике на блок-схеме сохранена преемственность с ранее введенной. Степень детализации операторных и условных блоков на блок-схеме такова, что каждый из них выполним соответственно в БОД и ФАМ за один такт. Рядом с блоками на рис. 3.3 проставлены адреса микрокоманд, инициирующих операции, обозначенные этими блоками. Микрокоманде соответствует одно из сочетаний:

1. Операторный блок с непосредственно следующим УБ (обведен штриховой линией). В этом случае операционное поле микрокоманды инициирует операцию в БОД (например, $O_2 \Rightarrow \varphi$ на рис. 3.3), а адресное — операцию $Y_1[\varphi]$ в ФАМ.

2. Операторный блок (без штриховой линии). В этом случае в БОД будет выполняться операция $O_1 \Rightarrow \varphi$, а в ФАМ — безусловный переход к следующей микрокоманде с адресом $i + 1$.

В данном примере (см. рис. 3.3) значения ключей K_T и K_C выбраны в диапазоне 0—5 и при записи везде употребляется десятичный эквивалент двоичного кода (как минимум, здесь он 3-разрядный). Значения K_T и K_C присвоены по правилам, перечисленным выше.

Какие выводы можно сделать относительно приведенной на рис. 3.2 схемы? Во-первых, она не вносит задержек в выполнение микропрограммы, т. е. не уменьшает быстродействие (производительность) микроконтроллера. Во-вторых, принципиально ошибки могут быть

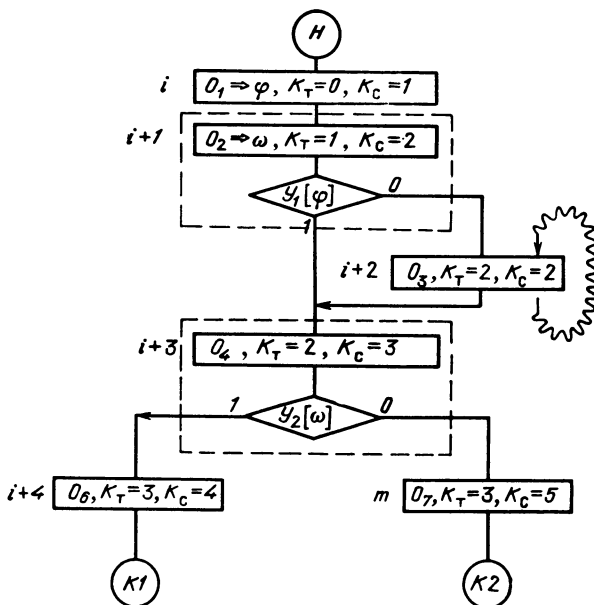


Рис. 3.3. Разметка блок-схемы алгоритма ключами

обнаружены в порядке следования микрокоманд, но не в правильности подачи управляющих сигналов из микрокоманды на схемы БОД. В-третьих, выделим ситуацию, когда неправильный порядок следования микрокоманд не будет обнаружен схемой контроля: это может произойти при ошибочном переходе к микрокоманде, значение ключа K_T которой совпадает со значением ключа K_C микрокоманды, являющейся последователем при правильном переходе. Пример такой ситуации на рис. 3.3 изображен волнистой линией: возврат к микрокоманде с адресом $i + 2$ не будет обнаружен схемой контроля, хотя и является неверным. Итак, чем больше одинаковых ключей в микропрограмме (а это определяется только конфигурацией блок-схемы алгоритма, если не введено ограничение на разрядность ключей), тем меньше достоверность контроля порядка следования микрокоманд. Заметим, что ограничение на разрядность ключей может быть введено из соображений экономии аппаратных затрат (разрядности МПП и регистров). В этом случае одинаковые значения ключей используются в различных фрагментах микропрограммы и ошибочный переход из одного фрагмента в другой с тем же значением K_C схемой контроля не обнаруживается.

Обнаружение бесконечных циклов при выполнении микропрограммы проведем по аналогии с известными способами контроля правильности выполнения программ с помощью сторожевого таймера или счетчика событий. Такой счетчик (например, микросхему К531ИЕ17) можно ввести в состав БМУ для контроля числа выполненных микрокоманд (т. е. тактов) между моментами его установки в исходное состояние. Если исходным состоянием счетчика является код 0000, то через пятнадцать тактов в счетчике будет код 1111, а на выходе переноса появится значение «1», которое и можно использовать в качестве индикатора заикливания при условии, что на любой корректной трассе микропрограммы сброс счетчика выполняется не реже чем один раз за пятнадцать тактов. Для сброса счетчика в исходное состояние необходимо выделить в микрокоманде один разряд, нулевым значением которого разрешается загрузка исходного кода в счетчик (в микросхеме К531ИЕ17П по фронту синхросигнала, что хорошо согласуется с синхронизацией всего БМУ). В качестве исходного кода необязательно

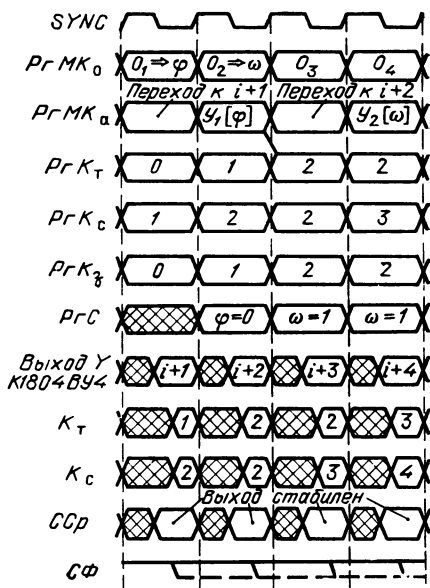


Рис. 3.4. Временные диаграммы следования ключей

принимать нулевой: чем больше его значение, тем чаще необходимо делать установку в ходе выполнения микропрограмм. С одной стороны, благодаря этому циклическое повторение фрагмента может быть обнаружено раньше (например, через 5 тактов, а не через 15). Но, с другой стороны, чем больше микрокоманд, осуществляющих сброс счетчика, тем больше вероятность того, что такая микрокоманда окажется внутри некорректного цикла, который из-за этого не будет выявлен.

Разумеется, в устройстве необходимо предусмотреть и схему фиксации сигнала ошибки, ибо единичное значение выходной перенос из счетчика имеет в течение только одного такта, после чего счетчик продолжит счет с комбинации 0000 при неправильном ходе программы. Отметим, что в счетчике K531IE17П сигнал выходного переноса индицируется активным нулевым уровнем, входы \overline{ET} и \overline{EP} должны быть заземлены, ко входу U/\overline{D} постоянно подключена «1», задающая направление счета (в сторону увеличения), а вход \overline{L} подключается к РгМК или непосредственно к выходу МПП. Тогда загрузка счетчика начальным кодом будет происходить одновременно с загрузкой в РгМК микрокоманды, предписывающей установку начального кода.

Какие модификации можно ввести для уменьшения числа необнаруживаемых неправильных переходов? Очевидно, есть смысл до предела сократить число одинаковых значений ключа. Есть два пути: алгоритмический и схемотехнический. Первый состоит в преобразовании блок-схемы алгоритма. Интересно отметить, что эквивалентное преобразование, уже рассмотренное нами с точки зрения повышения быстродействия, может служить и повышению контролепригодности микропрограммы. На рис. 3.5 показан результат преобразования исходного фрагмента (см. рис. 3.3) в модифицированный. Обратите внимание на то, что ситуация необнаруживаемого заикливания микрокоманды с адресом $i + 2$ теперь исключена, число значений ключей увеличилось с пяти до шести, число хранимых микрокоманд возросло с шести до семи. Однако быстродействие (т. е. время выполнения микропрограммы) не ухудшилось, ибо суммарное число выполнений всех микрокоманд данного фрагмента осталось прежним. Такой исход, очевидно, более предпочтителен, чем преобразование микропрограммы с потерей быстродействия. К последнему, например, приведет введение микрокоманды (блока) с пустой операцией в ветвь, соответствующую $\varphi = 1$ после блока $У_1 [\varphi]$ на исходной блок-схеме (см. рис. 3.3) с соответствующей перенумерацией всех ключей, число которых также возрастет до шести.

Схемотехнический путь повышения контролепригодности проиллюстрирован рис. 3.6 и состоит в следующем. При выполнении условных переходов ключи микрокоманд-последователей устанавливаются не одинаковыми, а различными в одном (например, младшем) разряде (рис. 3.7). В формат микрокоманды вводится дополнительный разряд M с единичным значением для микрокоманд условного перехода и нулевым для безусловного. Этот сигнал управляет мультиплексором MS (см.

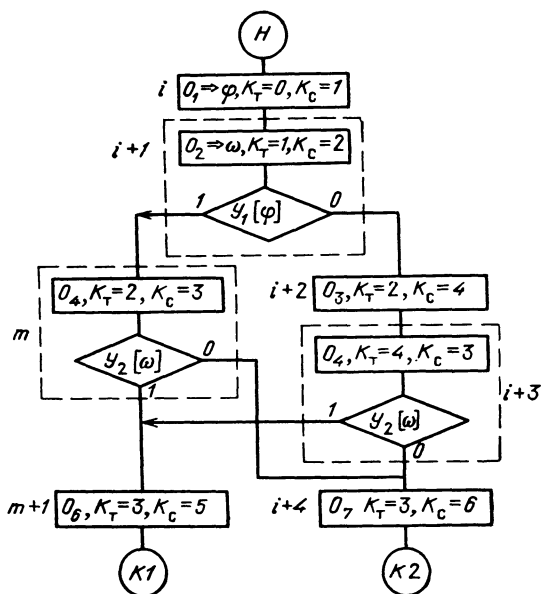


Рис. 3.5. Фрагмент с повышенной контролепригодностью

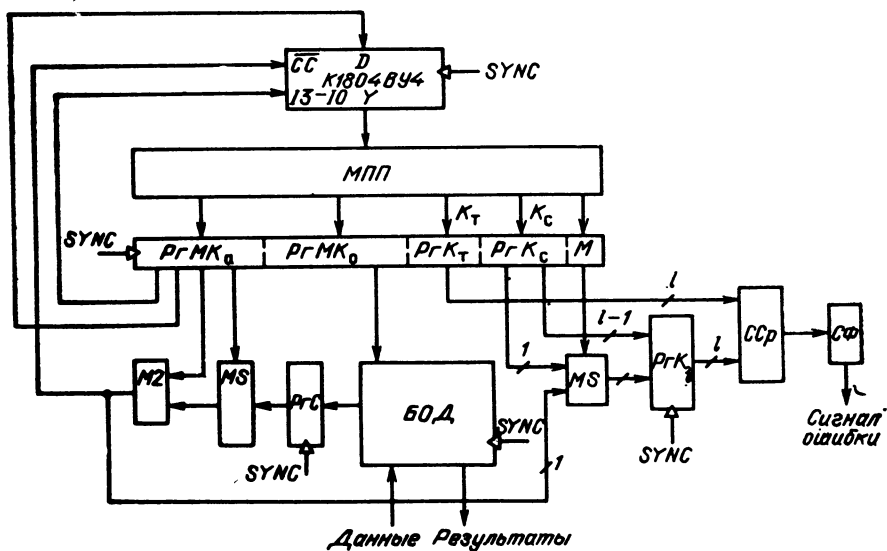


Рис. 3.6. Модификация ключа значением условия

рис. 3.6) младшего разряда на входе PгK_3 . При выполнении условных переходов через MS передается значение условия, при безусловных — младший разряд PгK_6 , который вместе с остальными $l - 1$ разрядами образует в PгK_3 полное слово ключа следующей микрокоманды. Например, при выполнении $(i + 1)$ -й микрокоманды (см. рис. 3.7) из PгK_6 в PгK_3 поступит двоичный код 01, а младший разряд примет значение условия $\varphi = 0$, т. е. вместе они образуют двоичный код $K_c = 010$, или $K_c = 2$ в десятичной форме. При правильной работе устройства следующей микрокомандой будет $(i + 2)$ -я, имеющая $K_T = 2$. При выполнении ее $M = 0$ и в PгK_3 будет занесен код 3 из PгK_6 .

Все перечисленные схемы контроля могут обнаруживать ошибки в порядке следования микрокоманд, но не ошибки в управляющих сигналах, поступающих на БОД. Для расширения возможностей схем контроля в этом направлении предложено [11] ключ K_T не хранить в микрокоманде, а формировать как некоторую функцию ее разрядов с помощью схемы сжатия (рис. 3.8). Один из вариантов сжатия — разбиение микрокоманды на поля, число которых равно числу разрядов ключа, и формирование каждого разряда ключа как бита четности соответствующего поля. Формируя ключи таким способом, можно в общем случае оказаться в ситуации, когда не все микрокоманды-последователи при условном переходе имеют одинаковые ключи, как это должно быть согласно правилу 2. Чтобы сделать эти ключи одинаковыми, применяется следующий прием: к каждому полю микрокоманды добавляется «корректирующий» разряд. Подбором значений этих раз-

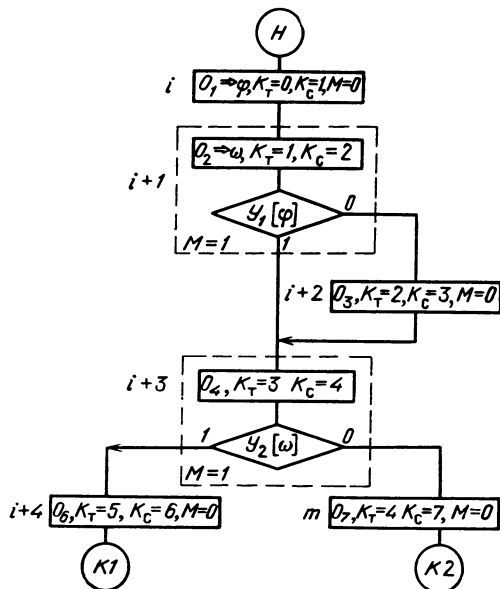


Рис. 3.7. Пример формирования ключей

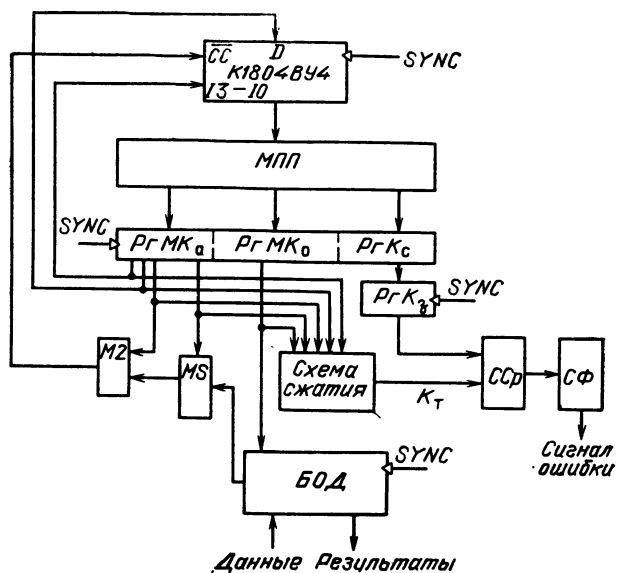


Рис. 3.8. Схема формирования ключа по разрядам микрокоманды

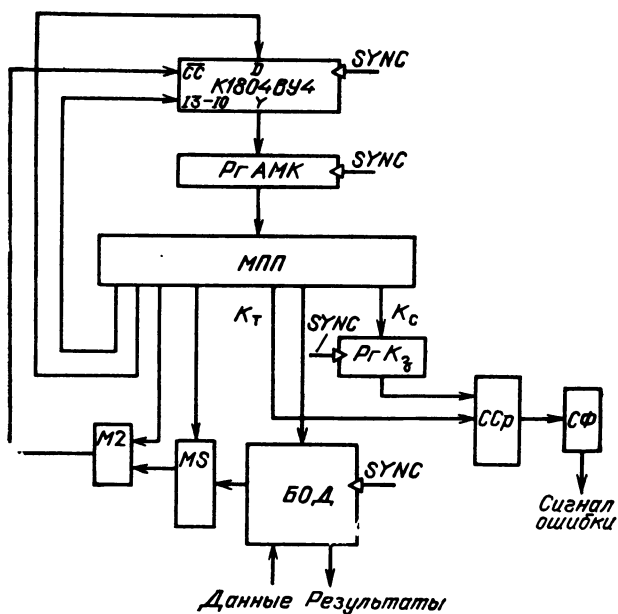


Рис. 3.9. Структура с регистром адреса микрокоманд

рядов добиваются получения нужных битов четности каждого поля. Если же микрокоманда-предшественник имеет только одну микрокоманду-последователя (в случае безусловного перехода), то нет необходимости уравнивать значения ключей, поэтому значения корректирующих разрядов можно установить произвольными. При появлении нечетного числа искажений управляющих сигналов в одном или нескольких полях микрокоманды значение ключа K_T не совпадает с эталонным значением ключа K_C , занесенным в RgK_3 предыдущей микрокомандой. Такая ошибка будет обнаружена. Итак, данная схема контроля служит для обнаружения ошибок обоих типов: в порядке следования микрокоманд и в управляющих сигналах микрокоманд.

Все перечисленные выше схемные решения базировались на структуре БМУ, включающей $RgMK$. Они же могут быть применены и для других типовых структур, приведенных в § 2.1. Так, на рис. 3.9 показана структура, содержащая $RgAMK$. Она характеризуется меньшим объемом аппаратных затрат и меньшим быстродействием за счет увеличения необходимой длительности такта.

При физической реализации рассмотренных схемных решений поля ключей желательно формировать, используя выходы нескольких микросхем, образующих МПП.

3.3. ПОФРАГМЕНТНЫЙ КОНТРОЛЬ МИКРОПРОГРАММ

Следующая группа структурно-алгоритмических решений по организации контроля последовательности микрокоманд основана не на попарной проверке правильности их следования, а на пофрагментной проверке. В качестве фрагментов выделяются линейные участки алгоритма, не содержащие разветвлений. При этом каждому начальному адресу фрагмента будет однозначно соответствовать конечный адрес. По вхождении во фрагмент конечный адрес вырабатывается определенным образом и хранится в регистре конечного адреса ($RgKA$) до окончания фрагмента. По достижении последней микрокоманды фрагмента адрес ее сравнивается с содержимым $RgKA$ и при несовпадении значений фиксируется факт неправильного функционирования устройства управления. Этот принцип и реализует схема на рис. 3.10. Поскольку сравнение текущего адреса с контрольным имеет смысл только на последнем такте фрагмента, в формат микрокоманды введен разряд $TEST$, единичным значением которого разрешается фиксация результата сравнения. Кроме того, в микрокоманду введен разряд \bar{W} , нулевым значением которого разрешается занесение нового контрольного адреса в $RgKA$. Все регистры загружаются, как обычно, по фронту синхросигнала. Аппаратно реализованная таблица переходов (ТП) хранит соответствие начальных адресов фрагментов конечным адресом. Реализовать ТП при разрядности $d \leq 16$ удобно на двух ПЛМ типа К556РТ1 или К556РТ2. Можно использовать и две микросхемы обычных ППЗУ емкостью 512×8 бит при разрядности адреса $d \leq 9$ или емкостью 2048×8 бит при $d \leq 11$. Следует позаботить-

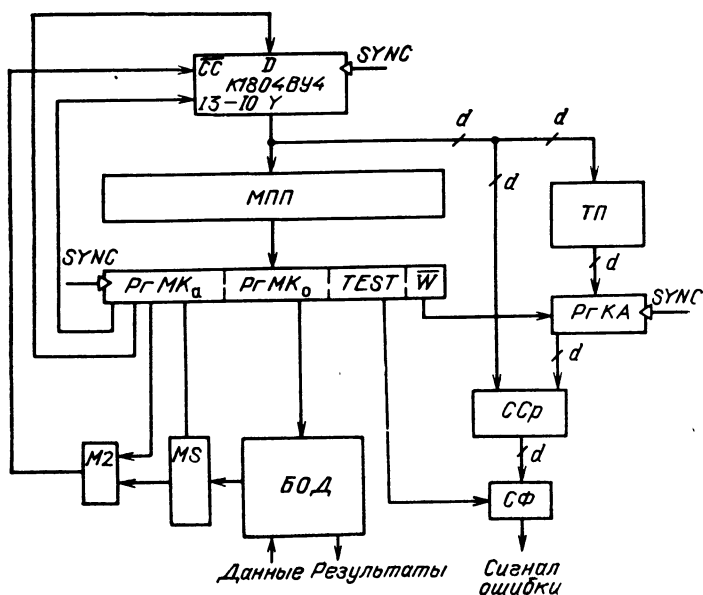


Рис. 3.10. Реализация контроля границ фрагментов

ся о программировании лишь тех ячеек ППЗУ, адреса которых используются в качестве входных для ТП, содержимое остальных ячеек безразлично.

Данная схема, как и все предшествующие, не вносит задержки в выполнение микропрограммы. Выборки микрокоманды из МПП в РгМК и контрольного адреса из ТП в РгКА производятся параллельно, дополнительных тактов не требуется.

Из принятого способа контроля следует, что схемой не обнаруживаются неправильные линейные последовательности микрокоманд, содержащие $TEST = 0$ при условии, что они оканчиваются правильным последним адресом (т. е. последовательность условных переходов соблюдается верная); на рис. 3.11 — это адрес $i + 5$ для фрагмента с начальным адресом $i + 1$. Кроме того, невозможно контролировать последовательность микрокоманд условного перехода — на рис. 3.11 это цикл микрокоманд с адресами $i + 6$ и m . Действительно, при выполнении $(i + 6)$ -й микрокоманды должен быть сформирован контрольный адрес, сверяемый на следующем такте со значением выхода Y схемы К1804ВУ4. На текущем такте неизвестен этот адрес, ибо следующая микрокоманда также выполняет условный переход. При $\omega = 1$ ситуация иная: контрольным адресом является $i + 8$.

Если в микропрограмме с малым числом условных переходов желательно организовать более частый контроль последовательности микрокоманд, то линейные участки алгоритма можно разбить на части и аналогичным образом контролировать каждую из частей.

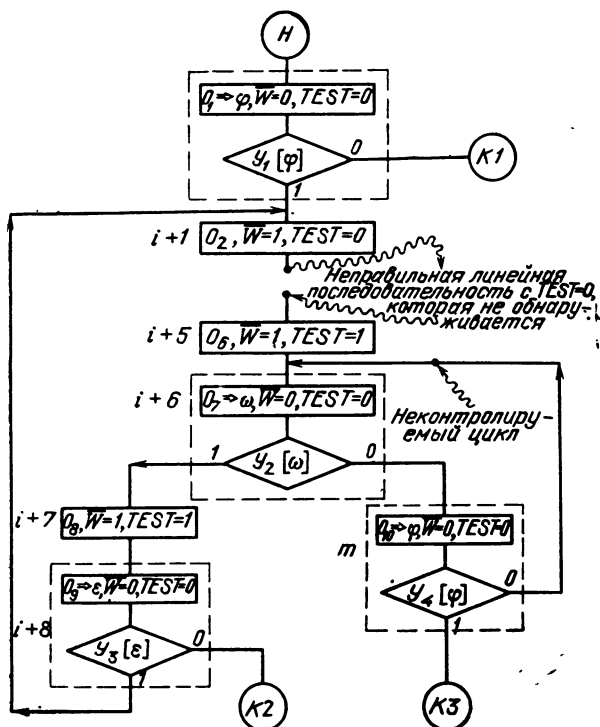


Рис. 3.11. Неконтролируемые фрагменты алгоритма

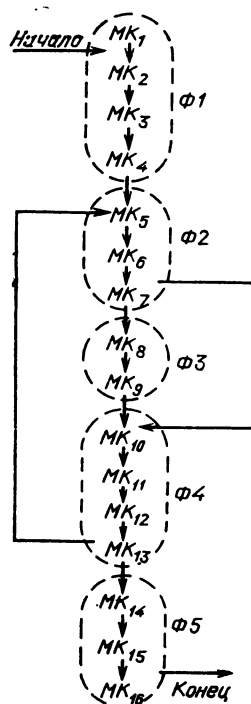


Рис. 3.12. Линейные фрагменты и передачи управления

Последняя группа структурно-алгоритмических решений также основана на пофрагментном контроле микропрограммы, однако способность схем контроля к обнаружению ошибок как в последовательности микрокоманд, так и в управляющих сигналах здесь значительно выше. Это является следствием использования сигнатурного анализа как метода регистрации, причем регистрация производится не в отдельных точках выполнения фрагмента микропрограммы, а непрерывно. Заключение о наличии ошибок делается по окончании каждого фрагмента путем сравнения итоговой сигнатуры с известным эталоном. Рассмотрим этот процесс более подробно.

Принцип контроля основан на понятии «сигнатура трассы» [11] (или «сигнатура фрагмента»). На рис. 3.12 символами $МК_i$ обозначены микрокоманды, а стрелками — возможные (т. е. допустимые, корректные, правильные) направления переходов между микрокомандами. Очевидно, если от микрокоманды (здесь $МК_7, МК_{13}$) идет более чем одна стрелка, то эта микрокоманда предписывает условный переход. В противном случае — безусловный переход. Для использования сиг-

натурного анализа необходимо прежде всего выделить линейные фрагменты в исходной блок-схеме алгоритма — на рис. 3.2 они обведены штриховыми линиями. Сделать это несложно. Назовем точкой входа микрокоманду $МК_i$, которая:

- а) является самой первой после блока «начало»;
- б) имеет более одной микрокоманды-предшественника, от которых возможна передача управления к данной микрокоманде;
- в) является не единственным последователем своей микрокоманды-предшественника.

Так, на рис. 3.12 точками входа являются: по первому признаку $МК_1$, по второму $МК_5$ и $МК_{10}$, по третьему $МК_5$, $МК_8$, $МК_{10}$ и $МК_{14}$. Суммарное множество точек входа образуют микрокоманды $МК_1$, $МК_5$, $МК_8$, $МК_{10}$, $МК_{14}$.

Точкой выхода назовем микрокоманду $МК_j$, такую, что:

- а) она имеет более одной микрокоманды-последователя (это $МК_7$ и $МК_{13}$);
- б) ее последователь имеет более одного предшественника (это $МК_4$ и $МК_9$);
- в) она является последней перед символическим блоком «конец» (здесь это $МК_{16}$).

Общий список точек выхода для данного примера: $МК_4$, $МК_7$, $МК_9$, $МК_{13}$, $МК_{16}$.

Линейным фрагментом называется последовательность микрокоманд, которая начинается точкой входа и заканчивается точкой выхода. Такие фрагменты Ф1—Ф5 выделены на рис. 3.12 штриховыми контурами. В отношении каждого из линейных фрагментов может быть определено понятие его сигнатуры. Сигнатурой фрагмента является сжатие по определенному правилу информации, сопутствующей выполнению фрагмента. Такой информацией может быть последовательность значений адресов микрокоманд, последовательность значений некоторых или всех полей микрокоманды и т. д. Правило сжатия информации определяется выбором точек обратной связи разрядов сдвигающего регистра, на который в каждом такте поступает и регистрируется информация в параллельном коде. Если при вхождении в линейный фрагмент микропрограммы установить сигнатурный анализатор (СА) в определенное исходное состояние, то при выполнении фрагмента в регистре СА будет накапливаться сигнатура, являющаяся предметом анализа по окончании фрагмента. При несовпадении ее с заранее известным для каждого фрагмента эталоном фиксируется ошибка. В противном случае микропроцессор продолжит работу, начав выполнение очередного фрагмента.

Где хранить эталонные сигнатуры фрагментов? Для этого можно ввести специальную постоянную память или воспользоваться МПП. Первое решение влечет увеличение аппаратных затрат, второе — потерю быстродействия в размере одного такта на каждый линейный фрагмент, хотя не обходится и без некоторой доли дополнительных аппаратных затрат. Чем более длинными явля-

ются линейные фрагменты и чем меньше их общее число, тем более предпочтителен второй путь. При ближайшем рассмотрении этот путь (с хранением сигнатур в МПП) включает ряд вариантов реализации.

Прежде всего вместо эталонной сигнатуры фрагмента, с которой на схеме сравнения необходимо сопоставлять накопленное реальное значение сигнатуры, оказывается целесообразным хранить «корректирующую» сигнатуру фрагмента. Эта корректирующая сигнатура выбирается такой, чтобы итоговая накопленная сигнатура по выполнению фрагмента принимала одинаковое для всех фрагментов значение, например 00 ... 0.

Корректирующую сигнатуру можно хранить в ячейке МПП, предшествующей контролируемому фрагменту, т. е. по первому адресу фрагмента, если саму сигнатуру считать частью фрагмента. Можно хранить ее и в последней ячейке фрагмента, можно и в некоторой промежуточной ячейке — каждый вариант имеет свою специфику. Начнем с обсуждения первого варианта.

Микропрограммная память в этом случае содержит информацию в следующем порядке:

Адрес i Корректирующая сигнатура фрагмента A .

Адрес $i + 1$ Первая микрокоманда фрагмента A .

...

Адрес j Последняя микрокоманда фрагмента A .

Адрес k Корректирующая сигнатура фрагмента B .

Адрес $k + 1$ Первая микрокоманда фрагмента B .

...

Адрес l Последняя микрокоманда фрагмента B .

Адрес m Корректирующая сигнатура фрагмента C .

Адрес $m + 1$ Первая микрокоманда фрагмента C .

Здесь многоточие означает тело фрагмента, в пределах которого адресация может быть произвольной. Важно лишь, чтобы адреса корректирующих сигнатур (i , k , m) непосредственно предшествовали адресам начальных микрокоманд ($i + 1$, $k + 1$, $m + 1$) фрагментов. Существенные связи структурных элементов микропроцессора показаны на рис. 3.13. Отметим, что для обнаружения ошибок в функционировании самих схем контроля (в частности, схем сравнения) их желательно делать самопроверяемыми [16]. Приведенные же здесь схемы из соображений минимизации аппаратных затрат свойством самопроверяемости не обладают.

Итак, рассмотрим работу микропроцессора со схемой контроля, обратившись к рис. 3.13, а также к временным диаграммам на рис. 3.14. По внешнему сигналу сброса $RESET = 0$ на выходе триггера сигнала ошибки в СФ устанавливается единичный потенциал, соответствующий правильной работе микропроцессора. Одновременно разрешается прохождение импульсов синхронизации $SYNC_a$ на управляющую часть микропроцессора: схему К1804ВУ4 и РгМК_а. Сигналом $RESET$ устанавливается код $I3 - I0 = 0000$, который через мультиплексор MS (принудительно $SIG = 1$) передается на микросхему К1804ВУ4, определяя циклическое обращение к нулевой микрокоманде до установки сигнала $RESET = 1$. После этого и начнется выполнение фрагментов микропрограммы. При считывании из МПП корректирующей сигнатуры, а не микрокоманды, необходимо предусмотреть, во-первых, сохранение элементов памяти БОД и, во-вторых, пере-

ход к следующему адресу в K1804BY4. Первое осуществляется блокировкой синхронимпульса на высоком уровне сигналом $\overline{SIG} = 0$, второе — настройкой мультиплексора MS на передачу комбинации 1110 (мнемоника $CONT$) вместо кода из $RгМК_a$ с помощью этого же сигнала $\overline{SIG} = 0$. По окончании текущего такта корректирующая сигнатура будет занесена в CA , где на текущем такте находится контрольная сигнатура предыдущего фрагмента, равная 00 ...0, если БМУ исправен. Сигналом $TEST = 1$ разрешается проверка контрольной сигнатуры. Наличие хотя бы одной единицы в контрольной сигнатуре приведет к установке триггера SF в противоположное состояние, выдаче активного уровня сигнала ошибки, блокировке синхронимпульсов $SYNC_a$ и $SYNC_0$. В исходное состояние схема может перейти только при подаче $\overline{RESET} = 0$.

При хранении корректирующей сигнатуры в первой ячейке фрагмента нет ограничений на расположение фрагментов в микропрограммной памяти. Контролируются значения всех управляющих сигналов микрокоманды (кроме единственного дополнительного разряда \overline{SIG}) и «автоматически» последовательность микрокоманд. Единственной трудностью является вычисление корректирующей сигнатуры, такой, что выполнение фрагмента завершается получением нулевого кода в регистре CA . Если вместо сигнатуры используется контрольная сумма по $\text{mod } 2$ фрагмента [11], такой подбор тривиален, но при нечетном числе ошибок в каждом разряде микрокоманды (контрольной суммы) работа микропроцессора будет трактоваться как правильная.

Для упрощения подбора корректирующей сигнатуры ее можно хранить в последней ячейке фрагмента. Однако здесь возникают другие трудности: ограничения на способы адресации и расположение фрагментов в памяти, что может отрицательно сказаться и на быстродействии. Информация располагается в МПП в следующем порядке:

Адрес i	Первая микрокоманда фрагмента A .
Адрес q	Последняя микрокоманда фрагмента A .
Адрес j	Корректирующая сигнатура фрагмента A .
Адрес $j+1$	Первая микрокоманда фрагмента B .
...	
Адрес k	Последняя микрокоманда фрагмента B .
Адрес l	Корректирующая сигнатура фрагмента B .
Адрес $l+1$	Первая микрокоманда фрагмента C .

Подбор корректирующей сигнатуры становится тривиальным: значение i -го ее разряда (кроме самого младшего) должно быть равно значению предыдущего ($i - 1$)-го разряда итоговой сигнатуры фрагмента. Это обусловит появление нулей на выходах сумматоров по $\text{mod } 2$ и занесение их в триггеры регистра на очередном фронте синхросигнала $SYNC_a$. Значение же самого младшего разряда корректирующей сигнатуры надо установить равным значению сигнала обратной связи, поступающего на второй вход сумматора по $\text{mod } 2$ при нахождении в регистре итоговой сигнатуры фрагмента. Тогда и в младший разряд по фронту синхросигнала будет занесено нулевое значение.

Необходимость размещения корректирующей сигнатуры в микропрограммной памяти непосредственно вслед за последней микрокомандой фрагмента влечет неудобства по следующей причине. Последняя микрокоманда фрагмента (точка выхода) должна передавать управление следующему фрагменту при чтении сигнатуры из МПП с передачей через MS соответствующего кода на входы $I3 - I0$ микросхемы K1804BY4 и выборкой следующего адреса из внутреннего источника этой микросхемы. Например, при подаче кода 1110 ($CONT$) источником является счетчик адреса микрокоманд и переход производится к следующей ячейке. Это означает, что новый фрагмент должен располагаться сразу за

корректирующей сигнатурой предшествующего фрагмента. Но ведь у этого нового фрагмента может быть не один, а несколько предшествующих (см. фрагменты Φ_2 и Φ_4 на рис. 3.12). Кроме того, если последняя микрокоманда (точка выхода) фрагмента предписывает условный переход, то оба альтернативных адреса должны формироваться только внутренними источниками микросхемы K1804BY4 при «аппаратном» управлении его полем I_3 — I_0 через MS . Поле D использовать, увы, нельзя, так как на последнем такте в $RgMK_A$ занесена не часть микрокоманды, а часть корректирующей сигнатуры (см. рис. 3.14, 3.15). Для расширения возможностей «аппаратной» адресации по внутренним источникам адреса микросхемы K1804BY4 число входов мультиплексора MS может быть увеличено до четырех. Для обеспечения условных переходов по внутренним источникам выбор условия будет производиться не в текущей, а в предыдущей микрокоманде; вместо RgC при этом достаточно поставить один триггер. В противном случае при нахождении в $RgMK_A$ корректирующей сигнатуры мультиплексором условий и сумматором по $mod\ 2$ пришлось бы управлять аппаратно, что повлекло бы чрезмерное и ненужное усложнение схемы.

Вспомним инструкции микросхемы K1804BY4, вызывающие формирование адреса по внутреннему источнику:

CONT — для безусловного перехода к следующему по порядку адресу. Адрес берется из счетчика микрокоманд (CMK);

RFCT — для организации циклов по количеству. Адрес берется либо из стека (если содержимое вычитающего счетчика не равно нулю), либо из CMK с изменением значения указателя стека;

LOOP — для организации циклов по внешнему условию, поступающему на вход \overline{CC} микросхемы. Адрес также берется из стека или CMK ;

JRP — для переходов к произвольному адресу, занесенному ранее в регистр адреса (он же вычитающий счетчик), либо к произвольному адресу, поступающему извне на вход D , в зависимости от значения \overline{CC} . Обратим внимание, что нельзя брать адрес из поля D микрокоманды, поэтому необходимо принудительно устанавливать RgC в состояние «0». Это единственный способ перейти к произвольному адресу, когда точка входа не следует за корректирующей сигнатурой предшествующего фрагмента;

CRTN — для возвратов из микроподпрограмм. Но поскольку обращение к микроподпрограмме выполняет инструкция *CJS* с использованием поля D и означает собой конец фрагмента, то от использования микроподпрограмм придется отказаться.

Очевидно, богатые возможности микросхемы K1804BY4 здесь практически не используются, микропрограммы становятся более громоздкими, «неуклюжими» и требуют больше тактов на выполнение.

Своеобразным компромиссом, позволяющим избежать перечисленных недостатков, является хранение корректирующей сигнатуры не в первой или последней, а в промежуточной, предпоследней, ячейке фрагмента.

Порядок расположения микрокоманд и сигнатур в памяти здесь таков:

Адрес i	Первая микрокоманда фрагмента A .
. . .	
Адрес j	Корректирующая сигнатура фрагмента A .
Адрес $j+1$	Последняя микрокоманда фрагмента A .
Адрес k	Первая микрокоманда фрагмента B .
. . .	
Адрес l	Корректирующая сигнатура фрагмента B .
Адрес $l+1$	Последняя микрокоманда фрагмента B .

Здесь переход от предыдущего фрагмента к последующему можно выполнять произвольным образом, ибо микросхема K1804BY4 управляется через *MS* полем микрокоманды, а не аппаратно. Кроме того, для формирования адреса можно использовать поле *D*, а корректирующую сигнатуру всегда можно без ущерба разместить перед последней микрокомандой, так как фрагменты по определению являются линейными и не содержат точек ветвления или слияния. В микрокоманде требуется выделить один дополнительный разряд (\overline{SIG}) для указания типа слова, считываемого из микропрограммной памяти: корректирующая сигнатура ($\overline{SIG} = 0$) или микрокоманда ($\overline{SIG} = 1$). Последовательное соединение двух триггеров с занесением по фронту синхросигнала $SYNC_a$ обеспечивает задержку разрешения схемы фиксации ($TEST = 1$) на два такта от чтения корректирующей сигнатуры.

Возможно еще более существенно сократить аппаратные затраты. Для этого сигнатурному анализу надо подвергать не сами микрокоманды, а их адреса: ведь разрядность микрокоманды в несколько раз превышает разрядность адреса. Кроме того, корректирующую сигнатуру адреса можно совместить с полем D микрокоманды: остальные поля будут содержать микрокоманду (например, предпоследнюю микрокоманду).

Рис. 3.15. Структура с анализом операционной части микрокоманд

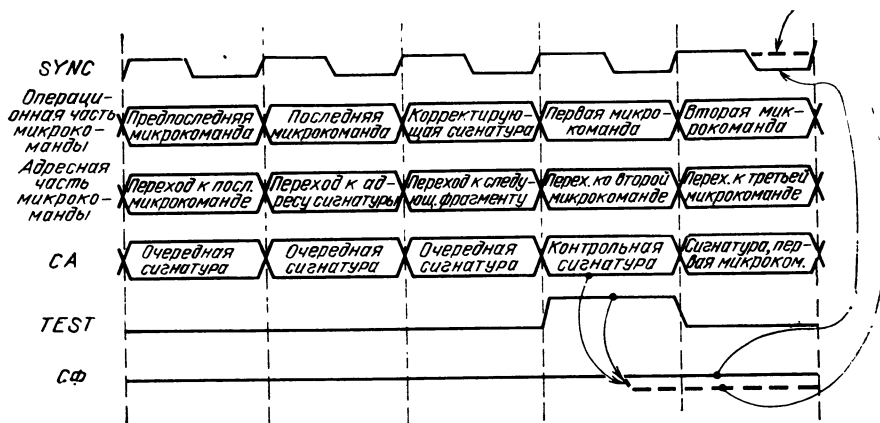


Рис. 3.16. Временные диаграммы при анализе операционной части микрокоманд

манду фрагмента). В результате дополнительного такта на выборку сигнатуры из памяти не потребуется. При выполнении фрагмента микропрограммы последовательность адресов микрокоманд сжимается в адресную сигнатуру фрагмента, которая анализируется по окончании фрагмента с фиксацией результата на СФ. Предпоследняя микрокоманда в поле *D* адресной части содержит корректирующую сигнатуру, которая через мультиплексор *MS* при *SIG* = 1 подается на *CA* вместо адреса этой микрокоманды. При этом поле *I3—I0* предписывает переход по внутреннему источнику адреса микросхемы К1804ВУ4, например *CONT*. В следующем такте (*SIG* = 0) на *CA* через *MS* будет поступать адрес последней микрокоманды фрагмента, а через один такт *CA* будет содержать контрольный код 00...0, если БМУ функционирует исправно. В этом такте на *CA* необходимо подавать сигнал разрешения фиксации состояния *TEST* = 1. Два последовательно соединенных триггера могут служить для задержки сигнала *TEST* относительно сигнала *SIG* на два такта. Одновременно с тестированием на СФ контрольной сигнатуры предшествующего фрагмента из МПП производится выборка, а затем и выполнение первой микрокоманды очередного фрагмента. Адрес ее, хранимый на данном такте с РгАМК, поступит по фронту синхросигнала в *CA*, а в РгАМК будет занесен адрес второй микрокоманды. и т. д.

Существенный недостаток возникает в результате экономии аппаратных затрат: сужение класса обнаруживаемых дефектов и ощутимая деградация быстродействия при выполнении диагностируемых микропрограмм. Недостаток возникает из-за того, что диагностируется лишь последовательность адресов микрокоманд, а не самих микрокоманд (двоичных кодов и последовательности их появления на выходах микропрограммного автомата).

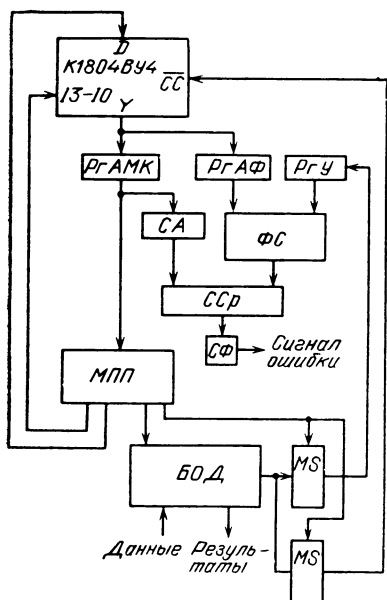


Рис. 3.17. Сигнатурный контроль условных переходов

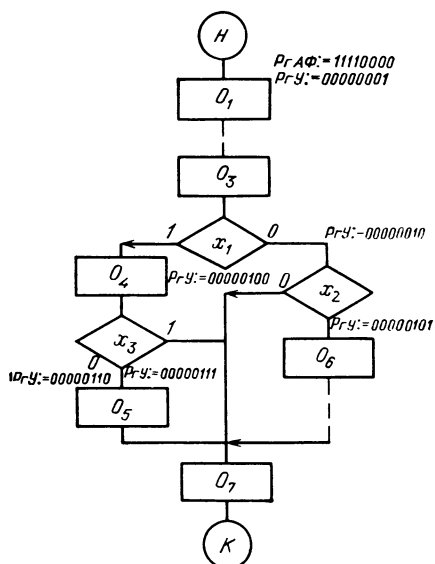


Рис. 3.18. Пример формирования контрольных кодов трасс

Деградация быстродействия при выполнении разветвленных микропрограмм связана с тем, что каждому сочетанию значений условий в микропрограмме будет соответствовать свой ход микропрограммы (трасса), а значит, и своя сигнатура. Большое число разветвлений микропрограммы вызовет резкое возрастание числа сопутствующих эталонных или корректирующих сигнатур. Хранение сигнатур потребует, во-первых, дополнительных затрат МПП и, во-вторых, дополнительных временных ресурсов — тактов на их выборку и обработку. На рис. 3.17 приведено одно из схемных решений, не приводящих к потере быстродействия [12]. Дополнительными элементами структуры являются: PгАФ — регистр адреса фрагмента микропрограммы; PгУ — регистр условий; ФС — формирователь сигнатур.

Основную часть средств контроля удобно реализовать с применением двух ПЛМ типа К556РТ2 в блоке ФС. Кроме того, в схеме используются 8-разрядные регистры типа К1804ИР2 (PгУ, PгАМК, PгАФ, СА), тактируемые общим синхросигналом.

Суть предлагаемого в [12] способа состоит в следующем. В блок-схеме алгоритма микропрограммного управления необходимо выделить фрагменты по правилу: концом фрагмента является блок, которому предшествует слияние ветвей блок-схемы. Начало следующего фрагмента совпадает с концом предыдущего. Пример выделенного таким обра-

зом фрагмента приведен на рис. 3.18. Здесь символами O_i обозначены операции в БОД, а x_j — проверяемые условия ветвления.

Пусть первая микрокоманда фрагмента, инициирующая операцию O_1 и безусловный переход к следующей, располагается в ячейке МПП с адресом $OF0$. При вхождении в данный фрагмент адрес $OF0$ или его часть, не совпадающая с адресами других фрагментов (в примере это 8 из 11 разрядов), загружается в РГАФ, где и хранится до окончания выполнения фрагмента. Кроме того, необходимо установить СА в исходное состояние.

При выполнении микрокоманд фрагмента в СА выполняется сигнатурная свертка их адресов. Поскольку различным значениям x_j соответствуют различные трассы, то каждому из возможных их сочетаний должна соответствовать своя эталонная сигнатура фрагмента. Эталонная сигнатура формируется на ФС как функция адреса фрагмента и конкретных значений условий $\{x_i\}$, имеющих место при данном выполнении фрагмента. Значения $\{x_i\}$ накапливаются в сдвиговом регистре условий РГУ. Максимальное число условий $\{x_i\}$ во фрагменте определяет разрядность РГУ. В примере оно равно семи (т. е. 128 возможных трасс). При вхождении во фрагмент в РГУ необходимо установить комбинацию 00000001 путем сброса сигналом из микрокоманды и единичного сдвига влево с принудительной установкой $\overline{CS} = 1$ на выходе мультиплексора условий MS.

Таким образом, трассе с $x_1 = 0$, $x_2 = 1$ соответствует «адрес» сигнатуры 00001111 00000101 на входе ФС, а трассе с $x_1 = 1$, $x_2 = 1$ — адрес 00001111 00000111. В соответствии со сформированным адресом осуществляется выборка эталонной сигнатуры фиксированной трассы из ФС и сравнение ее с полученной сигнатурой. При совпадении сигнатур выполнение микропрограммы продолжается, при несовпадении вырабатывается и фиксируется сигнал ошибки.

Если число анализируемых условий велико, то вместо РГУ целесообразно использовать СА условий. Несмотря на широкий формат адреса сигнатуры, далеко не все возможные комбинации адреса задействуются, что определяется построением микропрограмм. С этой точки зрения в качестве ФС предпочтительнее использовать ПЛМ по сравнению с ПЗУ. Данная схема обеспечивает функциональное диагностирование произвольных фрагментов микропрограмм. Оперативность диагностирования при этом определяется размером фрагмента и не зависит от числа условных переходов. Однако затраты аппаратуры весьма велики и приближаются к дублированию. Для контроля переходов между фрагментами можно применять способ, состоящий в формировании корректирующих сигнатур, таких, что они обращают итоговую сигнатуру фрагмента не в нуль, а в ключевой код, проверяемый в начале фрагмента-последователя.

Рассмотрим вариант организации схем контроля микропроцессоров, имеющих систему команд, с применением сигнатурного анализа [17]. Понятие правильного и неправильного функционирования про-

цессора команд уже приводилось при первом упоминании о контроле процессоров команд. Структурная схема микропроцессора с подобным контролем приведена на рис. 3.19. На ней изображены, в частности, два сигнатурных анализатора (СА1 и СА2), две схемы сравнения (ССр1 и ССр2) и память эталонных сигнатур (ПЭС). Когда код операции загружается в РгК, регистры анализаторов СА1 и СА2 устанавливаются в исходное состояние. В процессе выполнения микропрограммы каждой команды на СА1 и СА2 образуются сигнатуры соответственно адресов и управляющих сигналов микрокоманд. По окончании микропрограммы полученные значения сигнатур сравниваются с эталонными, которые поступают из ПЭС, адресуемой кодом операции из РгК. Каждой команде соответствует строго определенная микропрограмма, а значит, и сигнатура. Результат сравнения с эталонными сигнатурами фиксируется на СФ. Если команда, содержащаяся в РгК, предписывает условный переход, то возможны две трассы микропрограммы. Тестируемое условие можно подать в качестве младшего разряда адреса ПЭС, тогда для сравнения будут выбраны соответствующие сигнатуры при любом исходе.

На рис. 3.19 структура микропроцессора является производной от типовой структуры S_4 , содержащей РгМК и РгС. Очевидно, рассмот-

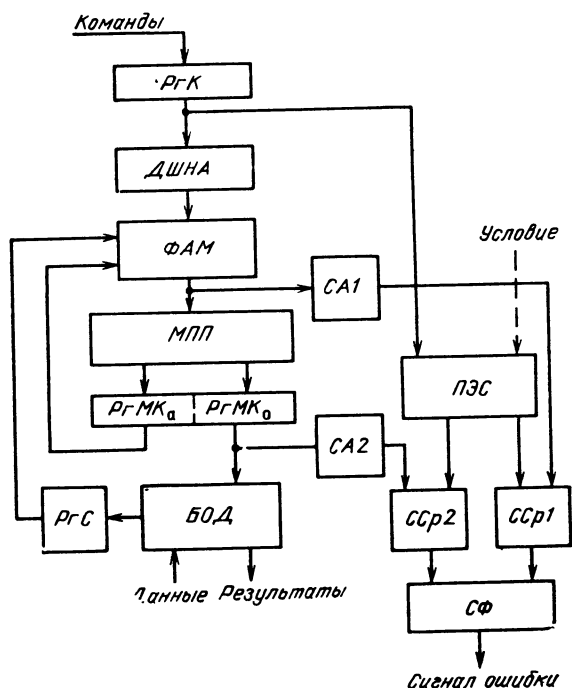


Рис. 3.19. Контроль в процессоре команд

ренные принципы имеют смысл и в отношении других типовых структур.

Систематизируем рассмотренные схемные решения в смысле выбора точек расстановки СА на укрупненной типовой структуре микропроцессора (рис. 3.20). Штриховой линией выделены структурные элементы, сочетание которых может варьироваться в зависимости от выбираемой типовой структуры. Кружками обведены номера шин, информация с которых может быть использована для сигнатурного анализа при функциональном контроле с большим или меньшим успехом.

Если рассматривать микропроцессор как «черный ящик», на который подаются воздействия (команды, данные, управляющие сигналы) извне и с которого снимается реакция (результаты, управляющие сигналы), то он считается исправным, когда обеспечивает адекватную реакцию на воздействия. Правильность результатов обработки данных определяется исправностью как БОД, так и БМУ; правильность управляющих сигналов может не зависеть от дефектов БОД и определяться лишь исправностью БМУ. В предположении, что БОД не имеет дефектов, преобразование данных будет выполнено без ошибок, если с БМУ на БОД будут подаваться правильные воздействия. Контролируя эти воздействия, можно сделать наиболее достоверное заключение о правильности работы БМУ. В этом смысле самыми информативными являются точки 6 (при наличии $RrMK_0$) или 5 (при отсутствии $RrMK_0$) на рис. 3.20. Установка СА в этих точках характеризуется наибольшими аппаратными затратами, ибо разрядность операционного поля микрокоманды, как правило, в 1,5—4 раза превышает разрядность адресного поля и в 3—8 раз разрядность адреса МПП. Следующие по степени информативности точки — это 2 (при наличии $RrAMK$) и 1 (при отсутствии $RrAMK$), поскольку при исправной МПП существует однозначное соответствие между содержанием микрокоманды, в частности ее операционного поля, и адресом этой микрокоманды. Если при отсутствии $RrMK_0$ постулировать отсутствие дефектов в МПП, то точка 2 не уступает по информативности точке 5, а аппаратные затраты на схему контроля будут ориентировочно в 3—8 раз меньше. (Конкретная экономия определяется отношением разрядности операционного поля к разрядности адреса для конкретной разработки.)

Аппаратурные затраты на сигнатурный анализ в точках 3 и 4 превышают таковые в точках 1 и 2. В то же время однозначное соответствие между адресным полем текущей микрокоманды и операционным полем следующей возможно только при безусловных переходах и в предположении отсутствия дефектов как в ФАМ, так и в МПП. По этим соображениям функциональный контроль с помощью сигнатур в точках 3 и 4 малоинформативен и дорог. Информация в точках 7—9 в определенной степени отражает работу БОД.

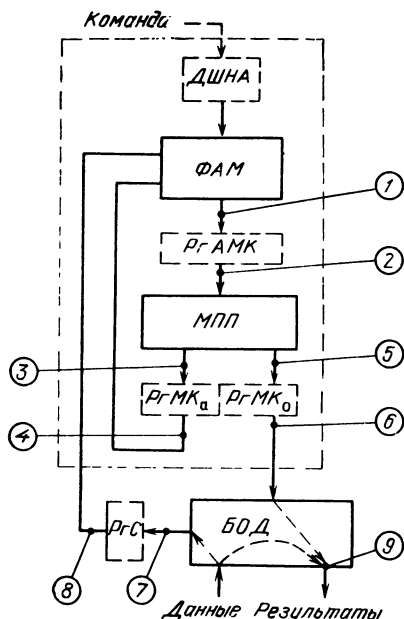


Рис. 3.20. Трассы для сигнатурного анализа

Завершая обсуждение вопросов функционального диагностирования БМУ, перечислим основные факторы, об учете которых должен позаботиться разработчик встроенной схемы контроля.

1. Степень деградации быстродействия микропроцессора (увеличение времени выполнения микропрограмм) по сравнению с базовым вариантом (без схем контроля) за счет увеличения числа тактов и/или длительности такта при заданных характеристиках алгоритма — длине фрагмента, частоте повторения и т. д.

2. Дополнительный объем МПП, используемый для управления схемами контроля (число ячеек, разрядность полей) или специальной памяти при заданных характеристиках алгоритма.

3. Достаточность ресурсов печатной платы для реализации встроенной схемы контроля с требуемыми характеристиками.

4. Класс обнаруживаемых дефектов.

5. Время обнаружения дефекта — число тактов от появления дефекта до выдачи сигнала о неисправности микропроцессора.

6. Соотношение надежности контролирующей и контролируемой аппаратуры. При реализации первой на микросхемах малой и средней степени интеграции, а второй — на БИС достоверность контроля находится под вопросом, особенно если контролирующая схема сложна. Поэтому естественной является тенденция к реализации схем контроля непосредственно на кристалле биполярной БИС [18] для функционального и тестового диагностирования микросхем и устройств на их основе. В ряде случаев целесообразно использование специализированной БИС для контроля (например, сигнатурного анализатора) в составе платы микропроцессора. Что касается тестового контроля микропрограммируемых БИС на этапе их производства, то относительная простота их структуры, системы инструкций и синхронизации позволяет контролировать их более эффективно, чем однокристальные микропроцессоры и микроЭВМ.

Глава 4.

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ МИКРОПРОГРАММ

4.1. ОРГАНИЗАЦИЯ ОТЛАДОЧНЫХ КОМПЛЕКСОВ

Симпатии многих разработчиков проблемно-ориентированных цифровых систем повышенной производительности принадлежат микропрограммируемым БИС и, в частности, комплекту К1804. Основными достоинствами этой элементной базы являются высокое быстродействие, интерпретация произвольных систем команд при любых формах данных, а также возможность реализации прикладных алгоритмов

на микропрограммном уровне для повышения производительности микропроцессорных систем. Недостаток заключается в значительной трудоемкости разработки и отладки оригинальных устройств по сравнению с типовыми конфигурациями микроЭВМ на основе однокристалльных микропроцессоров. Действительной причиной этой трудоемкости является не столько сложность микропрограммного управления, сколько отсутствие в массовом масштабе удобных инструментальных средств, назначение которых — облегчить труд микропрограммиста при резком повышении его производительности.

Понятие удобных инструментальных средств охватывает ряд аспектов:

- возможность интенсивной монопольной эксплуатации инструментального комплекса в течение достаточно длительного периода отладки микропрограмм;

- компактность, простота и надежность отладочных средств;

- легкость освоения входного языка и естественность представления информации пользователю в диалоговом режиме;

- невысокая стоимость и реальная возможность приобретения отладочного комплекса в кратчайшие сроки;

- отсутствие выраженной ориентации его на узкую номенклатуру БИС одной серии, т. е. универсальность;

- возможность отладки устройств в их реальном конструктивном исполнении и на рабочих частотах;

- автоматизированная документация результатов отладки.

Инструментальные средства в первом приближении делятся на два класса: кросс-средства и аппаратно-программные отладочные комплексы. Кросс-средства служат для написания микропрограмм и проверки их смысловой правильности на программной модели отлаживаемого устройства. Так, первой функции служит микроассемблер ГНОМ, а второй — система логической отладки СЛОТ [19], реализованные на мини-ЭВМ общего назначения типа СМ-4. Программная модель отлаживаемого устройства формируется на основе библиотечных программных моделей логики функционирования всех используемых в устройстве микросхем, а также информации о межсоединениях выводов этих микросхем, которая задается пользователем системы. Затем событийным методом моделируется функционирование устройства на заданном временном интервале (числе тактов). Результаты предоставляются пользователю, как правило, в виде распечатки состояний элементов устройства на каждом такте. Тщательный анализ распечатки пользователь выполняет после сеанса работы на ЭВМ и при обнаружении ошибок в микропрограмме или схеме устройства подготавливает исходные данные для следующего программного эксперимента.

Программное моделирование отлаживаемых устройств имеет следующие достоинства:

- возможность «наблюдения» внутренних элементов микропроцессорных БИС на каждом такте и установки их элементов в произвольное исходное состояние для проведения экспериментов;

низкая стоимость программных отладочных средств и легкость их тиражирования;

универсальность (в смысле неограниченных возможностей расширения библиотеки моделей микросхем различных серий);

возможность использования ресурсов мини-ЭВМ в многопользовательском режиме и т.д.

Вместе с тем программное моделирование имеет и ряд недостатков. Так, невозможно гарантировать абсолютное отсутствие ошибок в программных моделях, а стало быть, точное соответствие поведения модели и реальной микросхемы. В логических моделях трудно учесть все тонкости электрического и временного характера. Наконец, главное то, что программное моделирование может служить лишь промежуточной ступенью перехода к отладке натуральных макетов микропроцессорных устройств как в статическом режиме, так и на рабочих частотах, в их реальном конструктивном исполнении. При отсутствии соответствующих инструментальных средств проблемы отладки остаются. Поэтому более удобен для практического разработчика другой класс отладочных средств — аппаратно-программные инструментальные комплексы, позволяющие как писать микропрограммы, так и проверять их правильность при выполнении на реальном макете устройства.

По существу реализуемых функций все инструментальные комплексы такого класса [20] схожи: их основу составляет сервисная микро-ЭВМ, оснащенная дополнительными аппаратными средствами (эмулятор микропрограммной памяти, память трасс для логического анализа или анализа временных диаграмм, средства сопряжения с макетом устройства, дополнительный источник питания, программатор ППЗУ) и специальным программным обеспечением (микроассемблер, отладчик и др.). В совокупности эти средства образуют персональное рабочее место микропрограммиста.

В типовом варианте отлаживаемый макет устройства подключается к инструментальному комплексу (рис. 4.1) с помощью краевых разъемов или кабеля. При этом используются четыре шины: адреса микрокоманды, микроманды, синхронизации и трассы. В состав специализированных средств инструментального комплекса входит отладочное ОЗУ микропрограммы. Сначала это ОЗУ заполняется микропрограммой, являющейся продуктом работы микроассемблера и хранимой в виде файла на гибком диске, а затем (в режиме выполнения микропрограммы) фактически замещает МПП отлаживаемого макета. После отладки микропрограмма заносится на кристаллы ППЗУ, которые занимают свое место в сокетах (соединительных розетках с 24 выводами) на плате устройства, к которым ранее с помощью контактных приспособлений были подключены шины микрокоманд и адреса микрокоманд.

Быстродействие ОЗУ микрокоманд и ОЗУ трасс (в котором запоминаются значения сигналов с любых точек макета, включая адрес микрокоманды) достаточно для работы макета на реальной рабочей частоте (как правило, до 6 МГц).

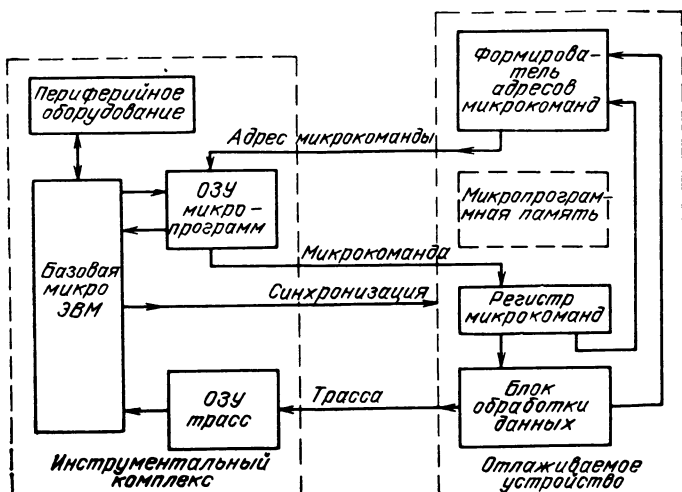


Рис. 4.1. Подключение макета к инструментальному комплексу

Инструментальные комплексы при дальнейшей детализации делятся на универсальные и специализированные. Универсальные аппаратно-программные отладочные комплексы позволяют писать микропрограммы и отлаживать устройства на основе любой серии микропрограммируемых БИС, специализированные же ориентированы на конкретные микросхемы конкретных серий. Классическим примером универсального комплекса является SYSTEM29, структура которого поясняется рис. 4.2. В операционной среде AMDOS можно использо-

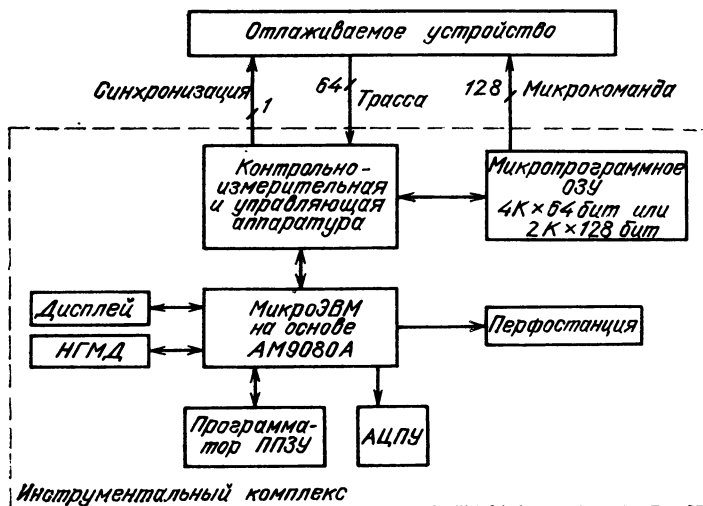


Рис. 4.2. Организация универсального инструментального комплекса

вать программы редактора текстов, AMDASM, AMSCRM, AMPROM и другие для написания и трансляции микропрограммы. Затем микропрограмма загружается в быстродействующее ОЗУ (с временем доступа не более 50 нс), служащее в качестве микропрограммной памяти вместо ППЗУ при отладке макета на рабочих частотах в несколько мегагерц. Микрокоманда может иметь разрядность до 128 бит. При выполнении микропрограммы осуществляется трассировка (запоминание в быстродействующей буферной памяти) 64 контрольных точек на протяжении 256 тактов. Результат затем можно вывести на дисплей и алфавитно-цифровое печатающее устройство (АЦПУ). Для хранения программных средств и файлов данных служат накопители на гибких магнитных дисках (НГМД).

Пример специализированного комплекса описан в [21]. Этот комплекс ориентирован на применение БИС *Am 2910* в качестве схемы формирования адресов микрокоманд. Поэтому на макет пользователя поступают только разряды операционной части микрокоманды (рис.4.3). в количестве не более 104. Микросхема *Am 2910* является частью инструментальной системы, что позволяет по специальной директиве монитора наблюдать на дисплее и модифицировать состояние элементов ее внутренней памяти. На рис. 4.4 детализирована структура отладочного блока микропрограммного управления, являющегося частью инструментального комплекса. Проверяемая микропрограмма исходно загружается в микропрограммное ОЗУ емкостью $4\text{К} \times 124$ бит по последовательному каналу связи с микроЭВМ через сдвиговый регистр. Микропрограмму не обязательно загружать полностью: можно модифицировать содержимое любой ячейки ОЗУ (микрокоманды) индиви-

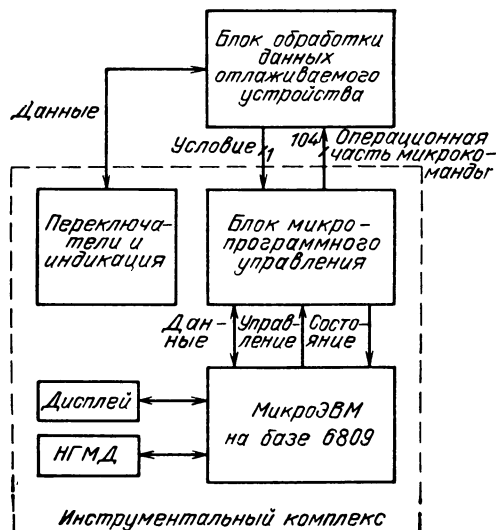


Рис. 4.3. Организация специализированного инструментального комплекса

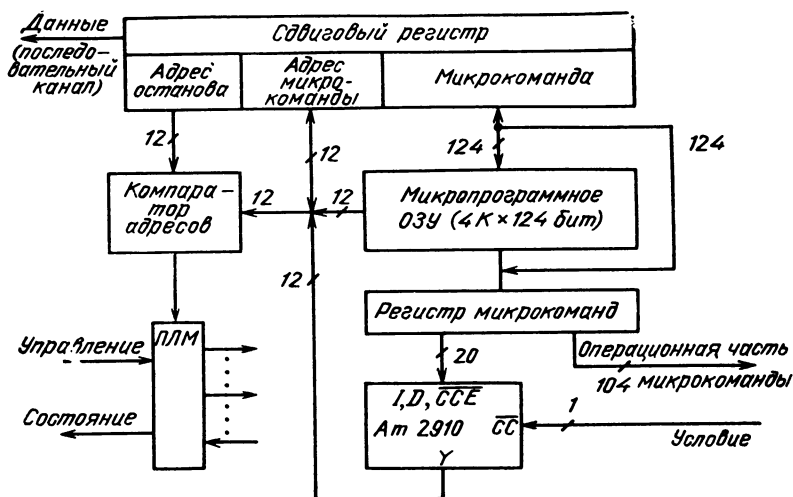


Рис. 4.4. Аппаратная часть специализированного комплекса

дуально. Для указания ее адреса служит соответствующее поле в сдвиговом регистре. Можно подавать микрокоманду и на регистр микрокоманд, минуя ОЗУ, т. е. не изменяя его содержимого.

В режиме выполнения микропрограммы адрес на вход ОЗУ подается не со сдвигового регистра, а с выхода Y БИС Ам 2910 (рис. 4.4). Этот же адрес подается на компаратор для сравнения его с заданным адресом останова. По совпадении адресов выполнение микропрограммы прекращается, о чем подается сообщение в микроЭВМ по шине состояния. Шина управления служит для задания режима функционирования всех элементов данной отладочной системы. Конкретные управляющие сигналы вырабатываются программируемой логической матрицей, которая служит и для шифрации кода состояния.

К отечественным серийно выпускаемым универсальным отладочным комплексам относятся МЕТАМИКРО [22] и АРМ2-05/КПМ-04. В первом базовой микроЭВМ является «Электроника-60». Второй построен на основе микроЭВМ СМ-1800 в комплексе АРМ2-05. В состав КПМ-04 (комплекта прикладных модулей для микропроцессорной серии К1804) входит несколько печатных плат конструктива Е2; часть из них расположена в отдельном каркасе. На платах размещены: микропрограммное ОЗУ емкостью $4К \times 64$ бит на микросхемах КМ132РУ5А, память трассы адресов микрокоманд емкостью $4К \times 16$ бит на таких же микросхемах, РгМК, формирователь адресов микрокоманд (на основе микросхем К1804ВУ2, К1804ВУ3, мультиплексора условий и счетчика), ОЗУ дешифратора кодов операций в начальные адреса микропрограмм емкостью 256×12 бит, а также схемы синхронизации и управления. Кроме того, в данный каркас вставляются макетные платы пользователя. На две платы пользователя разведено по двадцать линий с опе-

рационной части РгМК. Каркас стоит на столе и кабелем соединяется с частью КПМ-04, расположенной в задней части стола (модулями связи с базовой микроЭВМ). Там же находится блок питания (5В, 30А), входящий в состав АРМ2-05. Для отладки микропрограммы КПМ-04 обеспечивает их пошаговое или непрерывное выполнение, остановку по контрольному адресу, получение трассы адресов микрокоманд (до 4096 тактов) с временем цикла 250 нс. При желании пользователь может отказаться от конструктива *Е2* для реализации собственного макета (специализированные устройства, как правило, имеют свой конструктив), введя кабельную связь (что несколько ухудшит помехоустойчивость и снизит предельное быстродействие). При использовании микросхемы К1804ВУ4 в микропрограммных контроллерах формирование адресов микрокоманд осуществлять целесообразно в устройстве пользователя, а не на системной плате. В этом случае специфика ориентации КПМ-04 на серию К1804 исчезает и нет ограничений принципиального характера на отладку устройств, реализованных с использованием любых совместимых ТТЛ-комплектов микропрограммируемых БИС. Программные средства КПМ-04 работают в рамках операционной системы СПО-1800.

Практика показывает, что большая часть ошибок выявляется на этапе проверки логики функционирования устройства в пошаговом режиме (или в непрерывном режиме на малых частотах) с наблюдением десятков контрольных точек на каждом такте (или по условию останова). Для того, чтобы сделать заключение о работоспособности устройства, далеко не достаточно иметь 12-разрядную трассу адресов микрокоманд. Микропрограммы имеет смысл строить и отлаживать по модульному принципу и достаточно короткими. Поэтому возникает естественная потребность в широкой и не очень длинной трассе, ибо анализ трассы длиной в несколько тысяч тактов человеку просто не под силу. Поэтому трасса 64 разряда \times 256 тактов явно более информативна, чем трасса 16 разрядов \times 4096 тактов.

Выполнение инструментальным комплексом функций быстродействующего эмулятора микропрограммной памяти и логического анализатора обуславливает сложность самостоятельного его изготовления в кратчайшие сроки (а для многих вопрос может стоять именно так). В этой ситуации для проверки отлаживаемого макета на рабочей частоте можно пользоваться серийным логическим анализатором, а отладку смысловой правильности микропрограмм выполнять в облегченном (псевдостатическом) режиме с помощью несложных средств, описываемых ниже.

Разработанные сервисные аппаратные и программные средства дополняют базовую микроЭВМ СМ-1800 или комплекс АРМ2-05 на ее основе (рис. 4.5). При несложных модификациях возможно сопряжение их и с персональными компьютерами, совместимыми с СМ-1800 на уровне системного интерфейса и программного обеспечения.

Вся аппаратная часть комплекса для статической отладки микропрограмм состоит из простейшей платы сопряжения (в конструктиве *Е2*) базовой микроЭВМ СМ-1800 с отлаживаемым макетом микропроцессорного устройства. На плате

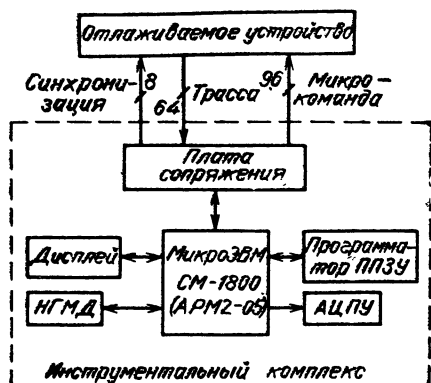


Рис. 4.5. Инструментальный комплекс для статической отладки микропрограмм

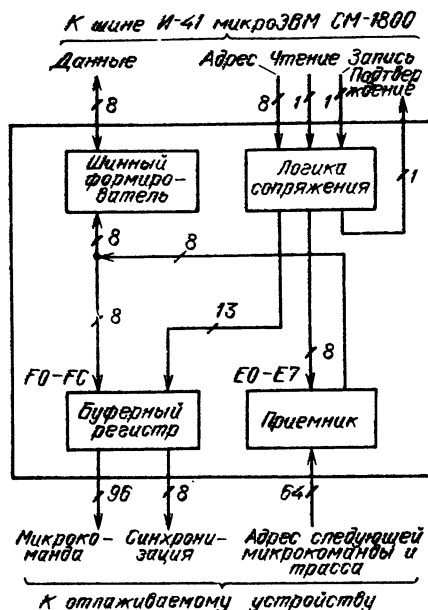


Рис. 4.6. Компоненты платы сопряжения

рис. 4.6. находятся: буферный регистр микрокоманды (не более 96 разрядов), высокоомные приемники (не более 64 разрядов) и логика сопряжения с системным интерфейсом И-41. Необходимость в буферном регистре связана с побайтной передачей информации из базовой микроЭВМ на плату сопряжения по шине И-41.

Каждому байту буферного регистра и приемника соответствует адрес в пространстве адресов устройств ввода — вывода микроЭВМ СМ-1800. Микрокоманда побайтно выводится в порты $F0 - FB$, сигналы синхронизации — в порт FC , а трасса принимается через порты $E0 - E7$, причем порт $E0$ и четыре старших разряда порта $E1$ предназначены для ввода адреса следующей микрокоманды.

С буферного регистра микрокоманда целиком поступает через разъем и многожильный кабель длиной около полуметра на отлаживаемое устройство пользователя, непосредственно (с помощью контактных приспособлений) на соединительные розетки РС-24-8, свободные при этом от микросхем ППЗУ микропрограммной памяти. В обратном направлении, т. е. от отлаживаемого устройства к плате сопряжения, аналогично передается адрес следующей микрокоманды и трасса сигналов с контролируемых точек макета. Управление синхронизацией макета пользователя также осуществляется через плату сопряжения. Никаких ограничений на конструктив макета нет, так как связь с платой сопряжения обеспечивается по кабелю. Поскольку система не предназначена для работы на высоких частотах, то кабельная связь, обеспечивающая произвольность конструктива пользователя, в данном случае предпочтительнее специализированных отладочных плат, выполненных в конструктиве базовой системы. Плата сопряжения располагается в каркасе микроЭВМ СМ-1800 (с лицевой стороны), а отлаживаемый микропроцессорный макет — на столе рядом с дисплеем. На начальном этапе выявления ошибок монтажа любая из точек макета может быть проконтролирована с помощью обычных измерительных приборов (осциллографа, тестера). В дальнейшем при отладке логики выполнения микропрограмм необходимость в этих приборах отпадает: сигналы с трассируемых точек макета отображаются на экране дисплея в цифровом виде и по желанию пользователя могут сопровождаться мнемониками.

Программное обеспечение отладочного комплекса включает как готовые, так и специально разработанные компоненты. Среди них:

популярный микропрограммный ассемблер AMDASM, обладающий свойствами настраиваемости на произвольный формат микрокоманд;

программа AMSCRM перестановки разрядов и полей в микрокомандах для приведения логического формата к физическому размещению разрядов в конкретных ППЗУ;

программа AMPROM доопределения безразличных значений в полях микрокоманды и подготовки файлов для программирования ППЗУ;

программа формирования файла выполняемых микрокоманд;

экранный редактор текстов для первоначальной записи и последующих модификаций текстов микропрограмм как на уровне мнемоник, так и на уровне кодов микрокоманд;

отладочный монитор, организующий в диалоговом режиме (на русском языке) взаимодействие с пользователем при выполнении микропрограмм (ранее оттранслированных с применением перечисленных выше средств) на натурном макете микропроцессорного устройства.

Все программные средства вместе с операционной системой ОС-1800 или *CP/M*, а также подробное руководство по отладочному комплексу размещены на одной дискете. На второй дискете (дискетов В) удобно хранить все микропрограммы как на входном языке микроассемблера, так и в кодах. Диалоговый отладочный монитор написан на языке Паскаль и в скомпилированном виде занимает 35 К байт.

Назначение и правила использования основных компонентов типового программного обеспечения отладочного комплекса более подробно рассматриваются в следующих параграфах настоящей главы.

4.2. МИКРОАССЕМБЛЕР. ФАЗА ОПРЕДЕЛЕНИЯ

Ассемблер — это программа, которая «читает» другие программы, написанные на уровне команд ЭВМ в символической форме, и обеспечивает на выходе двоичные коды, соответствующие символическому входному тексту. Микропрограммный ассемблер является особым видом ассемблера, называемым «мета-ассемблером». Мета-ассемблер отличается от обычного ассемблера конкретной ЭВМ тем, что большая часть символов определяется пользователем прежде, чем начнет выполняться собственно процесс трансляции программы. В обычном ассемблере пользователь может определять метки для команд, однако сами команды, включая соответствующую длину их слов и формат, заранее фиксированы.

Микропрограммный ассемблер должен быть более «гибким», чем традиционный ассемблер, поскольку он должен использоваться для различных конфигураций аппаратных средств ЭВМ. Каждая конфигурация аппаратных средств может требовать различных форматов микрокоманд, разрядность которых различна и достигает 100 бит. Из этих требований следует, что микропрограммный ассемблер должен включать две отдельные операции. Первая операция устанавливает длину слова микрокоманды, разбивку ее на поля и осуществляет определение форматов и констант, вторая является традиционным процессом ассемблирования, выполняемым над микропрограммой, записанной на языке введенных ранее форматов и констант.

Популярный ассемблер AMDASM, разработанный для изделий фирмы Advanced Micro Devices [23], является мощным мета-ассемблером, применимым при разработке любых микропрограммных ЭВМ. Ассемблер является двухфазным: он включает фазу определения (фаза 1) и фазу ассемблирования (фаза 2). Фаза ассемблирования весьма схожа с обычным ассемблером: производится чтение символической программы, присвоение меток, установка адресного счетчика, трансляция в двоичный код, формирование различных листингов и таблиц соответствия.

Фаза определения выполняется прежде всего для получения таблиц, которые связывают имена форматов пользователя и имена констант с соответствующими значениями полей бит. Эта фаза позволяет пользователю самостоятельно определять символы для форматов (имена форматов), символы для констант (имена констант) и разрядность слова микрокоманды. Длина микрокоманды может устанавливаться до 128 бит, что достаточно для всех практических применений.

В результате действия фазы ассемблирования получается файл, который содержит оттранслированную микропрограмму пользователя. Этот файл затем используется как исходный для программирования ППЗУ после соответствующей постобработки (см § 4.4).

В исходных операторах языка AMDASM используются следующие символы:

буквы алфавита от A до Z. Допускаются как строчные, так и прописные буквы. Вообще AMDASM обрабатывают все буквы, как будто они прописные, однако печатаются символы так, как они были представлены на входе исходных файлов;

цифры от 0 до 9;

специальные символы: знак плюс (+), знак минус (—), звездочка (*), штрих ('), запятая, круглые скобки, амперсанд (&), двоеточие, знак доллара (\$) или ₤, процент (%), точка с запятой, пробел, точка, возврат каретки, горизонтальная табуляция.

Файл определений, обрабатываемый на фазе 1, должен иметь следующую структуру:

TITLE имя файла

WORD разрядность микрокоманды

Директивы печати

Определения

Комментарии

END

Следующее через пробел за служебным словом TITLE (заголовок) имя файла не должно превышать 60 символов. Разрядность микрокоманды допускается не более 128 и задается целым положительным десятичным числом через пробел после служебного слова WORD. Директива END с возвратом каретки завершает файл определений.

Рассмотрим **директивы печати**. Директива LIST указывает, что последующие операторы должны быть напечатаны. Директива должна располагаться с новой строки и сопровождаться возвратом каретки, за

которым последуют печатаемые операторы файла определений. Директива *NOLIST* выключает печать, и операторы файла определений не будут печататься, пока не появится директива *LIST*. Однако любой исходный оператор, содержащий ошибки, будет по-прежнему распечатываться. Директива *NOLIST* должна располагаться с новой строки, сопровождаться возвратом каретки и предшествовать операторам файла определений, которые не должны распечатываться.

Директива *SPACE* *n* указывает, что ассемблер должен пропустить *n* строк до печати следующего оператора. Директива *SPACE* должна начинаться с новой строки, сопровождаться пробелом и десятичной цифрой, указывающей число следующих строк, которые должны оставаться пустыми. С помощью директивы *EJECT* ассемблер обеспечивает пустые строки на печатающем устройстве так, что число любых предыдущих строк плюс число пустых строк равно размеру страницы (по умолчанию он составляет 66 строк). Затем начинается новая страница. Директива *EJECT* должна располагаться с новой строки и сопровождаться возвратом каретки.

Определители применяются для введения констант, полных форматов микрокоманды или частичных (субформатов). Применяются три типа определителей в зависимости от определяемых объектов:

EQU — для того, чтобы установить соответствие идентификатора заданной константе; *DEF* — для определения формата микрокоманды; *SUB* — для определения формата части микрокоманды. Подробное описание правил использования определителей будет дано после разделов, посвященных полям, спецификаторам и константам.

Поле представляет собой группу соседних разрядов в микрокоманде. Каждое поле может принадлежать к одному из трех типов, т. е. быть:

- полем констант, содержимое которого представляет собой фиксированное значение бит;

- полем переменной, содержимое которого будет включать в различных ситуациях различные значения разрядов;

- полем «безразличных» значений, содержимое которого не используется в данном формате.

Тип данных в поле указывается с помощью спецификаторов. Используются шесть спецификаторов (*B #*, *Q #*, *D #*, *H #*, *X*, *V*):

B # — константа или поле, содержимое которого будет представляться с использованием двоичных чисел (0 и 1). Каждая цифра имеет неявную разрядность один бит, например *B # 101* (три бита 101).

Q # — константа или поле, содержимое которого будет представляться с использованием восьмеричных чисел (от 0 до 7). Каждая цифра имеет неявную разрядность три бита, например *Q # 32* (шесть бит 011010).

D # — константа или поле, содержимое которого будет представляться с использованием десятичных цифр (от 0 до 9). Неявная разрядность для десятичных цифр равна числу бит, необходимых для представления числа в двоичной форме. Таким образом, *D # 3* имеет не-

явную разрядность, равную двум, а $D \# 4$ имеет неявную разрядность, равную трем. Для полей в формате *DEF* или *SUB* символу $D \#$ должно предшествовать десятичное число, задающее явную разрядность поля, например: $3D \# 4$ означает 100, а $4D \# 8$ соответствует 1000.

$H \#$ — константа или поле, содержимое которого будет представляться с использованием шестнадцатеричных цифр (от 0 до 9 и букв от A до F). Каждая цифра имеет неявную разрядность четыре бита, например $H \# 8A$ означает восемь бит 10001010.

X — спецификатору поля безразличных значений X должно предшествовать десятичное число, задающее явную разрядность этого поля, например $4X$ — четыре бита поля безразличных значений.

V — спецификатору поля переменной V должно предшествовать десятичное число, задающее явную разрядность этого поля. Пример $6V$ — шесть бит поля переменной. Когда спецификатор $B \#$, $Q \#$, $D \#$ или $H \#$ задается после V , он становится постоянным атрибутом этого поля. Ассемблер допускает, чтобы любые значения, подставляемые в это поле, задавались цифрами, соответствующими спецификатору, без указания спецификатора с подставляемым значением.

Содержимое поля переменной может быть задано во время фазы определения для последующего использования по умолчанию на фазе ассемблирования. За спецификатором V могут следовать спецификаторы $B \#$, $Q \#$, $D \#$ или $H \#$, а последние могут сопровождаться соответствующими значениями, назначаемыми по умолчанию для этого поля. Таким образом, $6VQ \#$ указывает шестиразрядное поле переменной, содержимое которого будет задаваться в восьмеричной системе счисления, а $6VQ \# 35$ указывает, что если на фазе ассемблирования не указывается явное значение, то это поле должно иметь значение 011101 по умолчанию.

Каждое поле, следующее за определителем, должно содержать не более 16 бит, если это не поле безразличных значений; заканчиваться запятой, если это не последнее или не единственное поле, следующее за определителем; задавать константу, используя спецификаторы $B \#$, $Q \#$, $D \#$ или $H \#$ и соответствующие цифры, или переменную, которая задается длиной в битах и спецификатором V . Если спецификатор не следует за V , то тип переменной полагается двоичным. Поле может также задавать безразличные значения с явным указанием длины перед спецификатором X , либо быть именем константы или именем субформата, которые были предварительно определены.

Идентификаторами могут быть устанавливаемые пользователем имена констант, имена форматов или имена субформатов. Идентификатор должен быть первым элементом в операторе; начинаться с алфавитного символа (A — Z) или точки (.); завершаться двоеточием (:); состоять не более чем из восьми символов; не содержать внутренних пробелов; предшествовать определителю *EQU*, *DEF* или *SUB*. Идентификаторы могут содержать и более восьми символов, но при трансляции они будут отсекаются до первых восьми символов. Идентифика-

Т а б л и ц а 4.1. Определение констант

Форма	Допустимые цифры	Описание	Форма	Допустимые цифры	Описание
n	От 0 до 9	Десятичное значение (по умолчанию можно не описывать)	$\{i\} Q \# n$	От 0 до 7	Восьмеричное значение
			$\{i\} D \# n$	От 0 до 9	Десятичное значение
$\{i\} B \# n$	0 или 1	Двоичное значение	$\{i\} H \# n$	От 0 до 9 и от A до F	Шестнадцатеричное значение

торы могут также следовать за пробелами и сопровождаться пробелами после двоеточия (:) и перед *EQU*, *DEF* или *SUB*.

Примерами правильных идентификаторов являются

NUMBER:

SHIFT:

REG.4:

Примерами неправильных идентификаторов будут

* *ADD* (использован специальный символ *);

SHIFT LEFT (внутренний пробел, более 8 символов);

4MUX (первый символ не буква и не точка).

Константы применяются, чтобы связать идентификатор со значением, а также для определения фиксированного поля бит. Например, $Q \# 62$ определяет поле бит 110010. Этот тип константы имеет неявную разрядность (каждая восьмеричная цифра представляется тремя битами). Если спецификатору предшествует десятичная цифра (как например, $4H \# 5$), то она указывает явную разрядность поля (здесь равную четырем).

Константы представляются не более чем шестнадцатью разрядами. Допустимые формы для констант приведены в табл. 4.1, где $\{i\}$ означает число, задающее разрядность в явном виде, которое может предшествовать спецификатору.

В любом поле может содержаться выражение. В выражении могут использоваться определители, цифры, метки, а также операторы. В выражениях допускаются следующие операторы:

+ прибавить значение левого операнда к значению правого операнда;

— вычесть значение правого операнда из значения левого операнда;

* умножить значение левого операнда на значение правого операнда;

/ разделить значение левого операнда (делимое) на значение правого операнда (делитель).

Все выражения анализируются транслятором слева направо, при этом иерархия операторов отсутствует и не допускаются скобки для

группирования. Операторы должны давать в результате значение, которое является целой положительной константой (остатки отбрасываются).

Рассмотрим правила использования определителей. Определитель *EQU* применяется, чтобы приравнять имя постоянному значению или выражению. Общая форма его такова:
имя: *EQU* константа или выражение.

Определитель приравнивает символы, заданные в позиции имени (т. е. идентификаторы), значению константы или выражения. После *EQU* допускается только одно выражение или константа. Например, имя *R12* устанавливается равным значению 1100 следующим образом: *R12 : EQU H # C*. Дальнейшие ссылки на значение 1100 могут быть сделаны с использованием имени *R12*. Если за *EQU* не следует спецификатор, то по умолчанию предполагается *D #*. Так, *R10: EQU 10* означает распределение разрядов 1010, неявная длина которого составляет четыре разряда.

Каждый определитель *EQU* должен располагаться с новой строки; начинаться с идентификатора, за которым следует *EQU* (пробелы между двоеточием и *EQU* являются допустимыми); содержать константу, выражение или имя константы, которые представляются полем бит; определять значение, которое может быть представлено не более чем шестнадцатью разрядами. Каждый определитель *EQU* может сопровождаться точкой с запятой и комментарием после константы или выражения; продолжаться на дополнительных строках посредством использования «/» в качестве первого символа на этих строках; применяться как в ассемблируемом файле, так и в файле определений.

Определитель *DEF* применяется для определения полного формата микрокоманды. Он устанавливает содержимое неизменяемых частей микрокоманды, положение и разрядность полей переменных и безразличных состояний. Кроме того, могут быть заданы безусловные значения для переменных частей микрокоманды. Общая форма такова:
идентификатор: *DEF* поле 1, поле 2, ..., поле *n*.

Каждый определитель *DEF* должен начинаться с новой строки совместно с определяемым именем (идентификатором); сопровождаться одним или несколькими пробелами, за которыми следуют поля, выделяемые запятыми; иметь сумму разрядностей всех полей точно равную разрядности микрокоманды, заданной с помощью *WORD*; определять все разряды в микрокоманде как константы, безразличные состояния или переменные.

Определитель *DEF* может содержать пробелы между двоеточием идентификатора и символом *DEF*; продолжаться на дополнительных строках путем использования символа «/» в качестве первого непустого символа (т. е. не пробела) на этих строках; сопровождаться точкой с запятой и комментариями после того, как определено полное слово; содержать имена субформатов и констант, которые были определены предварительно; содержать переменные и безразличные состояния, константы и выражения; содержать переменное поле, которое сопровождает

ся безусловным значением для него (безусловное значение может быть константой или безразличным состоянием); быть наложенным на поля безразличных состояний с другим форматом, чтобы получить полную микрокоманду на фазе ассемблирования. Наложение на поля безразличных состояний повлечет ошибку, так что этим свойством нужно пользоваться осторожно.

Определитель *SUB* применяется для введения субформата, который является форматом части микрокоманды. Субформат подобен формату, за исключением того, что он содержит меньше бит, чем полная микрокоманда. Поля могут быть константами, переменными или безразличными состояниями. Форма определителя *SUB* такова:

идентификатор: *SUB* поле 1, поле 2, ..., поле *n*.

Каждый определитель *SUB* должен начинаться с новой строки, следовать за идентификатором, сопровождаться одним или несколькими пробелами, следующие за которыми поля выделяются запятыми; предшествовать *DEF*, в котором есть ссылка на данный *SUB*.

Каждый определитель *SUB* может быть меньше микрокоманды по разрядности; продолжаться на дополнительных строках благодаря использованию символа «/» в качестве первого непустого символа в этих строках; заканчиваться точкой с запятой и комментариями; содержать имена, которые были определены заранее, а также константы, выражения, переменные или безразличные состояния.

Удобно использовать *SUB* в тех случаях, когда несколько форматов содержат идентичные поля. В этом случае имя субформата может употребляться в каждом *DEF* всякий раз, когда встречаются эти поля. Рассмотрим примеры записи определителей *EQU*, *SUB* и *DEF*.

R2: EQU B # 010

Здесь идентификатор *R2* сопоставляется с 3-разрядной константой 010. Поле бит 010 будет подставляться при трансляции всякий раз, когда встречается идентификатор *R2*.

SHFT: SUB 3V, B # 10110, 5X

Здесь имя *SHFT* определяется как субформат с 3-разрядным полем переменной (3V), 5-разрядной константой (*B # 10110*) и 5-разрядным полем безразличных значений (5X), что в сумме дает 13 бит.

ADD: DEF 3V, B # 10110, 5X, B # 0011, 4X, B # 010

Здесь *ADD* определяется как формат с 3-разрядным переменным полем (3V), 5-разрядным полем константы (*B # 10110*), пятиразрядным полем безразличных значений (5X), 4-разрядным полем константы (*B # 0011*), 4-разрядным полем безразличных состояний (4X) и 3-разрядным полем константы (*B # 010*). Полная длина микрокоманды составляет 24 разряда. Это же имя формата можно записать, используя имя субформата *SHFT* и имя константы *R2*, а именно

ADD: DEF SHFT, B # 0011, 4X, R2

Таблица 4.2. Атрибуты неявной разрядности констант

Константа	Неявная разрядность	Двоичное значение	Определение неявной разрядности каждой цифры
<i>AB: EQU B # 1000</i>	4	1000	Каждая двоичная цифра кодируется одним разрядом
<i>BB: EQU Q # 10</i>	6	001000	Каждая восьмеричная цифра кодируется тремя разрядами
<i>CB: EQU H # 10</i>	8	00010000	Каждая шестнадцатеричная цифра кодируется четырьмя разрядами
<i>DB: EQU 12</i>	4	1100	Предполагается, что 12 есть десятичное число, неявная разрядность отсчитывается до старшего правого разряда, равного единице
<i>EB: EQU 4</i>	3	110	Неявная разрядность равна трем и определяется аналогично по приведенному выше правилу

Разрядность каждого поля может задаваться явно и неявно. Явная разрядность указывается для поля десятичными цифрами до спецификатора. Максимальная длина составляет 16 бит, за исключением полей безразличных значений, максимальная длина которых равна разрядности микрокоманды. Таким образом, *3B # 101* указывает поле с явной разрядностью, равной трем. Спецификаторы десятичных переменных или безразличных полей требуют указания явной разрядности перед *D #*, *V* или *X*. Безразличные или переменные поля нуждаются в указании разрядности потому, что они в начальной стадии не содержат определенного значения бит, а десятичные поля в формате или субформате — из-за отсутствия непосредственной связи между числом десятичных цифр, которые заданы, и числом двоичных бит, желательных для этого поля. Если явная разрядность не задана, то константа имеет неявную разрядность, определяемую используемым спецификатором (табл. 4.2).

При наборе текста любой оператор может быть продолжен на дополнительных строках, если не помещается в одной строке. Для переноса используется символ «/», употребляемый в качестве первого непустого символа на очередной строке. Заметим, что недопустимо его употребление между спецификаторами *B*, *D*, *Q* или *H* и знаком *#*. Пример правильного использования:

ADD: DEF 3V, B # 10110,5X,

/B # 0011, 4X, B # 010

Для внесения пояснений в микропрограммы рекомендуется применять комментарии. Текст комментариев располагается после точки с запятой. Комментарии могут занимать полную строку или часть ее. Все символы от точки с запятой до конца строки не принимаются во внимание ассемблером. Комментарии должны начинаться от точки с запятой и располагаться после полного поля, если они применяются внутри *DEF* или *SUB* (последующие поля этих *DEF* или *SUB* должны начинаться на новой строке с символа «/», указывающего, что поля являются продолжением *DEF* или *SUB*), например:

SHFT : SUB 3V; — это субформат (задающий сдвиг),
/B # 10110,5X; который продолжается на второй строке.
; *ADD*, заданный ниже, есть полный формат микрокоманды
ADD : DEF SHFT, B # 0011, 4X, R2
; полное число бит для *SHFT* равно 13
; поле бит *SHFT* будет подставляться
; в формат *ADD*, приведенный выше.

Здесь строки 3, 5, 6 и 7 представляют собой полные строки комментариев. Строки 1 и 2 являются операторами, подлежащими обработке, поэтому все символы после точки с запятой будут трактоваться как комментарии. Определитель *SUB*, начатый в первой строке, продолжается во второй строке с символа «/».

Модификаторы располагаются после константы или после спецификатора *V*. При размещении после константы они изменяют заданное значение. При размещении после спецификатора *V* модификаторы называются **атрибутами** данного поля и постоянно связаны с этим полем. Атрибуты будут модифицировать как значение, заданное по умолчанию для поля переменной в файле определений, так и любое значение, подставляемое в это поле переменной, когда идентификатор формата упоминается в ассемблируемом файле.

Модификаторы предписывают следующие действия:

- * инверсия (получение обратного кода);
- отрицание (получение дополнительного кода);
- : отсечение разрядов слева для приведения заданного значения в соответствие с числом явных разрядов указанного поля;
- % сдвиг значения вправо в рамках поля с заполнением остающихся слева разрядов нулями;

□ поле трактуется как адресное при «страничной» организации памяти. Этот атрибут допускает подстановку и служит для контроля переходов. Используется только с переменными полями.

Ниже приводятся примеры правильного использования модификаторов с константами:

*D # 5 ** дает поле бит 010 (инвертируется 101, т. е. 5),
B # 0101 — дает поле бит 1011, что является дополнительным кодом.
6Q # 357: дает поле бит 101111 (левые три разряда 011 отсекаются).

12H # A5% дает распределение бит 000010100101 (*A5* представляет собой сдвинутые вправо биты в двенадцатиразрядном поле).

Примерами неправильных полей, обусловленных пропуском модификаторов, являются следующие:

4B # 101 — явная разрядность равна 4 битам, но только 3 бита следуют после B # , а знак %, указывающий выравнивание, не задан.

5Q # 34 — явная разрядность равна 5 битам, но число 34 требует шести бит, а знак отсечения (:) отсутствует.

Модификаторы употребляются после значения константы или после спецификатора V, но перед установленным по умолчанию значением для переменного поля (например, 12V%Q # 46), если они должны быть постоянными атрибутами поля. Для модификации безусловного значения модификаторы должны соответствовать только ему (например, 12VQ # 46%). Модификаторы не должны употребляться с различными состояниями (так, 3X% является ошибкой). Для одного и того же поля нельзя применять оба модификатора: * и —. Модификаторы или атрибуты могут употребляться в любом порядке, однако обрабатываться транслятором они всегда будут в следующем приоритетном порядке: *, —, %, :, □.

В переменных полях (т.е. полях для подстановки переменных значений) в качестве атрибутов могут использоваться спецификаторы B #, Q #, D # и H #, которые постоянно связаны с таким переменным полем. Если переменное поле не имеет специфицированного основания системы счисления, то оно будет по умолчанию двоичным. Если пользователю всегда требуются входные переменные в восьмеричной форме, каждое переменное поле в файле определений следует записывать как nVQ #. Тогда, если в ассемблируемом файле значение для этого поля будет задано как 27, оно будет трактоваться как восьмеричное. Если в ассемблируемом файле нежелательны восьмеричные числа, то поле в нем может быть записано как B # 010111, чтобы восьмеричный атрибут, заданный по умолчанию, не принимался во внимание.

Если переменное поле определяется по умолчанию (например, 4VH# C), то спецификатор (H #) становится атрибутом этого поля. Атрибут H #, если он задается с переменным полем в файле определений, может при необходимости повторяться в ассемблируемом файле, ибо транслятор не может отличить шестнадцатеричные значения, которые начинаются с любой буквы от A до F, от идентификаторов, которые могут также начинаться с букв от A до F.

Атрибут □ может быть использован только с переменными полями для указания страничной адресации. Когда атрибут задан с переменным полем, атрибуты %, : автоматически устанавливаются для этого поля. Атрибут □ указывает, что старшие (левые) разряды значения должны отсекаются и сравниваться с соответствующими разрядами текущего значения адресного счетчика. Если отсеченные разряды не совпадают с соответствующими разрядами адресного счетчика, то констатируется ошибка. Желаемый размер страницы определяется числом бит, заданным как разрядность этого переменного поля. Таким образом, если страница должна иметь 256 слов, то переменное поле должно определяться как 8V□. Любое значение, подставленное в это поле,

будет отсекается слева и оставшиеся восемь правых бит будут подставляться в поле. Если отсеченные слева разряды не совпадают с соответствующими разрядами текущего значения адресного счетчика, то подстановка означает переход к другой странице; в этом случае транслятором выдается сообщение об ошибке. Возможность устанавливать размер страницы через атрибут \square позволяет обнаруживать ошибку условного или безусловного перехода (к адресу в другой странице микрокода).

Безразличные значения применяются для указания бит (поля), состояния которых (значения разрядов) несущественны в рассматриваемой микрокоманде. Безразличные поля специфицируются как nX , где n является числом безразличных разрядов в десятичной форме. Безразличное значение обозначается символом X на выходе фазы ассемблирования и может получать значение 0 или 1 на этапе постобработки (см. § 4.3). Безразличное поле является единственным, длина которого может превышать 16 разрядов и которое может совмещаться с другим форматом, содержащим константы в том же самом поле.

Переменные значения применяются для определения полей микрокоманды, содержимое которых не требует фиксации до момента ассемблирования. Переменное поле (или поле переменной) может иметь значение по умолчанию в файле определений. Возможны следующие формы употребления переменных со спецификатором $V : nV$; nV , атрибуты; nV , атрибуты, значение по умолчанию; nV , атрибуты, значение по умолчанию, модификаторы; nV , значение по умолчанию, модификаторы.

Поле переменной должно сопровождаться указанием разрядности n в явном виде перед спецификатором V , причем в десятичной форме ($n \leq 16$); заканчиваться запятой, если за ним следует другое поле; содержать модификатор $\%$ после спецификатора V , если в ассемблируемом файле для этого поля в качестве подстановки должно использоваться выражение или значение адресного счетчика.

Поле переменной может содержать атрибуты (сразу после спецификатора V), такие как инверсия (*), который всегда будет инвертировать значение, задаваемое для этого поля; содержать спецификатор, задаваемый со значением по умолчанию или без него; содержать значение по умолчанию, заданное в двоичной форме и указываемое спецификатором $B \#$, или восьмеричное $O \#$, или шестнадцатеричное $H \#$, или десятичное $D \#$, за которыми следуют цифры; содержать модификаторы после значения по умолчанию, которые модифицируют только его и не связаны постоянно с этим переменным полем; содержать безусловное значение по умолчанию, заданное как X (безразличное состояние), если пользователь желает повторно использовать это поле на фазе ассемблирования; содержать по умолчанию либо безразличное значение, либо явное значение, но не оба одновременно.

Примерами правильного применения переменных полей с безразличными состояниями по умолчанию являются: $3VX$, $3V*X$, $3V\square X$, $3V*\square X$. Другие примеры переменных полей:

3V — трехразрядное поле. Содержимое представляет собой переменную и будет заменяться фактическим значением во время трансляции ассемблируемого файла. Тип поля по умолчанию двоичный;

3VQ* — трехразрядное поле. Содержимое представляет собой переменную и будет заменяться фактическим значением во время трансляции ассемблируемого файла, причем в этом файле подстановка может задаваться восьмеричным числом без применения спецификатора Q. Если содержимое будет задаваться в двоичной, десятичной и другой форме, то спецификатор B # или D # должен располагаться перед подставляемым числом, задаваемым в ассемблируемом файле;

3V*% — трехразрядное поле, содержимое которого представляет собой переменную. Любое значение, заданное для этого поля в ассемблируемом файле, будет автоматически инвертироваться и выравниваться вправо. Поскольку спецификатор не задан, то поле является по умолчанию двоичным. Если содержимое должно быть задано в восьмеричной или иной форме в ассемблируемом файле, то соответствующий спецификатор (Q #, H #, D #) должен предшествовать подставляемым цифрам;

3VQ # 5 — трехразрядное поле, содержимое которого представляет собой переменную. Если значение на задано явно для этого поля в ассемблируемом файле, то оно будет по умолчанию полем бит 101, описанным как Q # 5 в файле определений;

3VQ # 5* — подобно описанному, но константа 5 инвертируется, чтобы получить поле бит 010. Значения, подставляемые в это поле во время трансляции ассемблируемого файла, автоматически не инвертируются, ибо инверсия здесь не является атрибутом поля;

3V*Q # 5* — такое же поле бит, как и 3VQ # 5*, но любое значение, подставляемое во время трансляции ассемблируемого файла в это поле, будет автоматически инвертироваться, поскольку знак инверсии следует после V, а не после 5 и является здесь атрибутом поля переменной;

3V*Q # 5* — трехразрядное переменное поле со значением по умолчанию, равным 5, инвертированным, а затем в соответствии с модификатором *, следующим за спецификатором V, вновь инвертированным. Результирующим значением бит будет 101. Любое значение, подставленное в это поле в ассемблируемом файле, будет инвертироваться.

Таким образом, атрибуты, расположенные сразу после V, постоянно связаны с этим полем и будут действовать на любое значение по умолчанию, заданное с полем, а также на любое значение, подставляемое в это поле в ассемблируемом файле. Модификаторы, расположенные после значения по умолчанию, относятся только к нему. Примерами неправильного применения переменного поля являются

3VH # 7

Здесь H # 7 требует четырех разрядов, а двоеточие для указания того, что левые биты следует отсечь для соответствия трехразрядному полю, отсутствует.

3:VH # 7

Двоеточие расположено в неправильной позиции. Допустимы варианты 3V:H # 7 или 3VH # 7: в зависимости от того, является отсечение постоянным атрибутом поля или модификатором значения по умолчанию H # 7.

4.3. МИКРОАССЕМБЛЕР. ФАЗА АССЕМБЛИРОВАНИЯ

На фазе ассемблирования выполняется чтение текстового файла операторов исходной микропрограммы, присваивание значений меткам и константам, затем трансляция микрокоманд из символической формы в объектный код (0, 1, X). Для этой трансляции применяются выходные данные фазы определения — таблицы идентификаторов констант, форматов и связанных с ними распределений бит. Общая структура ассемблируемого файла включает:

TITLE имя файла

Директивы управления печатью

Директивы управления адресным счетчиком

Директивы определения констант

Выполняемые операторы

Комментарии

END

Все директивы, приведенные в списке, могут располагаться в любом месте файла между *TITLE* и *END*. Однако выполняемые операторы должны следовать в порядке, который соответствует желаемой последовательности микрокоманд в объектном коде микропрограммы. Каждый выполняемый оператор, т. е. микрокоманда, может сопровождаться (по желанию) меткой. Некоторые типы операторов должны сопровождаться параметром, который может быть константой, идентификатором константы или выражением. Любой оператор может продолжаться на нескольких строках посредством использования символа «/» в качестве первого непустого символа (т. е. не пробела) на этих строках.

Метки или идентификаторы представляют собой группы букв и (или) символов, которые имеют соответствующее значение. Метки допускаются для выполняемых операторов, а идентификатор необходим при определителе *EQU*. Значение идентификатора или метки определяется типом следующего за ними выражения. Так, строка идентификатор: *EQU n* служит для присвоения символьному идентификатору значения *n*, а строкой

метка: идентификатор формата *VFS1*, *VFS2*, ..., *VFSN* метке сопоставляется текущее значение адресного счетчика (так что далее в микропрограмме ссылки к данному адресу могут задаваться в символической форме), а в поля переменных подставляются фактические значения *VFS1*, *VFS2*, ..., *VFSN*.

Метка или идентификатор должны начинаться с алфавитного символа (от A до Z) или с точки; заканчиваться двоеточием; содержать

не более 8 символов, исключая двоеточие (избыточные символы отсекаются справа); не содержать пробелов; определяться однократно. Метка или идентификатор могут предшествовать *EQU*, *RES*, *ORG*, *FF* или выполняемой микрокоманде; использоваться в качестве подстановки переменного поля (*VFS*); использоваться в качестве поля в операторе *FF*, содержать только буквы А — Z, число 0—9 или точку в позициях со второй по восьмую.

Когда идентификатор (имя) определяется с помощью *EQU*, определение должно предшествовать применению имени в качестве поля или константы. Так, строка *AM2910: DEF JSR, 28X* должна размещаться в исходном тексте после строки *JSR : EQU H # 5* (но не обязательно сразу после нее!). Целесообразно размещать все *EQU* в начале ассемблируемого файла микропрограммы.

Когда метка следует за удвоенным двоеточием (::), она называется точкой входа. Точки входа применяются с ориентацией на программирование ППЗУ, чтобы легче было получать значение адресного счетчика, связанное с некоторыми строками микрокода. Точки входа указываются в ассемблируемом исходном файле как

метка :: имя формата *VFS1*, *VFS2*, ..., *VFSN*

За исключением удвоенного двоеточия, точки входа подчиняются всем правилам, применимым к меткам. Список точек входа (символов и значений) может быть получен при трансляции посредством директивы *MAP*.

В ассемблируемом файле используются шесть типов предложений: директивы управления печатью (*LIST*, *NOLIST*, *SPACE*, *EJECT*, *TITLE*), директивы управления адресным счетчиком (*RES*, *ORG*, *ALIGN*), операторы определения констант (*EQU*), выполняемые микрокоманды (с фазы определения или *FF*), комментарии, директива *END*.

Заголовок, указанный в строке с *TITLE*, будет печататься вверху каждой страницы листинга. В заголовок можно ввести не более 60 символов. Когда появляется новый *TITLE*, печатающее устройство выдает пустые строки до завершения текущей страницы, а следующие страницы будут содержать новый заголовок. «Страница» здесь не обязательно является физической страницей, поскольку пользователь может по желанию задавать число ее строк. Остальные директивы печати (*LIST*, *NOLIST*, *SPACE*, *EJECT*) выполняют действия, идентичные описанным в § 4.2.

Перейдем к рассмотрению директив управления адресным счетчиком. Директива *ORG n* используется для установки адресного счетчика на новое значение адреса микропрограммной памяти, в котором будет размещаться следующая микрокоманда. За словом *ORG* должны следовать по крайней мере один пробел и адрес *n*, который трактуется как десятичное число, если ни один из спецификаторов (*B#*, *Q#* или *H#*) не предшествует ему. Директива может использоваться для изменения значения адресного счетчика только в сторону больших значений, а также содержать выражение вместо числа *n*. Если *ORG* отсутст-

вует, то ассемблер использует в качестве начального состояния адресного счетчика нуль.

Директива *RES n* применяется для резервирования слов памяти. Это обеспечивается приращением значения адресного счетчика на *n*. Директива *RES n* должна содержать по крайней мере один пробел и число *n*, трактуемое как десятичное, если не указаны спецификаторы *B#*, *Q#* или *H#*. Директива может содержать выражение вместо *n*.

Директива *ALIGN n* применяется для установки адресного счетчика в следующее значение, которое является кратным числу *n*. Например, следующая микрокоманда будет размещена по адресу, который является целочисленным значением, кратным 2, 4, 8 или 16. Директива *ALIGN n* должна содержать по крайней мере один пробел и *n*, трактуемое как десятичное число, если не указан иной спецификатор (*B #*, *Q #* или *H #*). Директива *ALIGN n* может содержать выражение вместо *n*.

Выполняемые операторы образуют тело ассемблируемой микропрограммы. При трансляции (с соответствующей подстановкой параметров) они образуют выходные (объектные) коды фазы ассемблирования. Операторы должны поступать на вход фазы в порядке, который соответствует желаемому порядку их в объектном коде.

Большинство выполняемых операторов будут ссылаться на имена форматов, введенные на фазе определения. При использовании переменных полей необходимо указать подстановки (*VFS*) для них. Общая форма такова:

{метка} : имя формата *VFS1*, *VFS2*, ..., *VFSN*.

Выполняемые операторы должны начинаться с новой строки; содержать имя формата с фазы определения; подставлять идентификатор константы, метку, константу или выражение в каждое переменное поле. Подстановки должны разделяться запятыми. Если может быть использовано значение по умолчанию, заданное на фазе определения, то фактическое значение *VFS* в соответствующей позиции может быть опущено. Выполняемые операторы могут содержать единственное имя формата или неограниченное число имен форматов, подлежащих совмещению; содержать текущее значение адресного счетчика в качестве подстановки для поля, причем \square может быть и частью выражения (например, $\square + n$), сопровождаться меткой с двоеточием или меткой с двойным двоеточием.

Выполняемые операторы, форматы которых не были описаны на фазе определения, могут быть определены на фазе ассемблирования с помощью оператора свободного формата (*FF*). Общая форма его такова

{метка}: *FF* поле 1, поле 2, ..., поле *N*

Ассемблируемый файл может содержать неограниченное число операторов *FF*. Каждый оператор *FF* должен начинаться с новой строки; содержать символ «/» в качестве первого непустого символа, если оператор продолжается на другой строке; иметь поля, отделенные запяты-

ми; иметь явную разрядность n , заданную для безразличных полей (nX) или для полей, определенных с использованием десятичных чисел (nD); не содержать переменных полей; не содержать идентификатора константы для поля, если эта константа предварительно не была определена в ассемблируемом файле или файле определений; не совмещаться с другим именем формата.

Каждый оператор *FF* может следовать за меткой и двоеточием или двойным двоеточием; содержать выражение для любого поля, но выражение должно заключаться в скобки и следовать за разрядностью поля n . Например, *FF 5X, 10 (Σ — 5), B #101*.

Оператор *FF* может содержать значение для выражения, которое должно автоматически выравниваться в поле вправо. Если же число бит, которые представляют значение, больше, чем длина поля, то выдается сообщение об ошибке при условии, что за закрывающей скобкой не следует знак отсечения. Оператор *FF* может включать поле, содержанием которого является текущее значение адресного счетчика с использованием символа или выражения, содержащего атрибут Σ .

Пример: пусть константы *AZ* и *RB* были введены в файле определенных следующим образом:

WORD 48

AZ: EQU B # 01

RB: EQU B # 001000

Тогда ассемблируемый файл мог бы содержать операторы

C: EQU H # C

*XTRA: FF 12H # 3%, AZ, 18X, C, B # 10111,
/1X, RB*

Микрокоманда (двоичный код) для данного оператора *FF* получится такой:

00000000001101XXXXXXXXXXXXXXXXX110010111X001000

При **совмещении (перекрытии) форматов** для образования микрокоманды используется следующая конструкция:

{метка:} имя формата *VFS1*, ..., *VFSN* & имя формата *VFSM*,

где *VFS* есть подстановка переменного поля, а & — символ совмещения форматов. Формат может быть совмещен с другим форматом при условии, что разрядность каждого из них равна разрядности микрокоманды, а каждому разряду одного формата, содержащему единицу или нуль, соответствует тот же разряд в другом формате, специфицированный как безразличный. Последующие совмещения должны происходить на безразличных полях, остающихся после всех предшествующих совмещений форматов. Микрокоманды, которые были введены с использованием оператора свободного формата *FF*, не могут быть совмещены.

Так, если файл определений содержит строки

ADD: DEF 5X, 8H # A2, 3X

REG1: DEF B # 00001, 11X

CARRY: DEF 15X, B # 1

то строке *ADRG: ADD & REG1 & CARRY* соответствует код 0000110100010XX1.

Операторы комментариев являются невыполняемыми операторами, которые применяются для облегчения восприятия информации о переменных или о структуре программы. Комментарий может быть отдельной строкой или следовать, например, за оператором определения константы. Все символы от точки с запятой до конца строки (возврата каретки) транслятором не обрабатываются и служат главным образом для создания качественной документации.

Правила определения разрядности констант были подробно рассмотрены выше. Однако разрядность, связанная с атрибутом \square , представляет собой особый случай. При обнаружении символа \square в поле констант или в выражении текущее значение адресного счетчика подставляется транслятором вместо \square . Так, если текущее значение адресного счетчика равно 59 для микрокоманды, предшествующей строке *NEXTLOC: EQU \square + 5*, то идентификатору *NEXTLOC* сопоставляется адрес 64. Если в поле подставляется \square , то разрядность адреса вычисляется как число бит от правого до крайнего левого единичного бита. Возможно, что она не будет соответствовать явно определенной разрядности поля. Поэтому при описании в файле определений или в операторе *FF* полей, в которые будет подставляться \square , необходимо употреблять атрибуты % и (или):

В ассемблируемом файле можно использовать **выражения**. Все выражения оцениваются по правилам целочисленной арифметики (остаток отбрасывается) и должны давать в результате положительное значение, которое может быть представлено не более чем шестнадцатью разрядами. Применяются только операторы сложения, вычитания, умножения и деления. Оцениваются в строгой последовательности слева направо, т. е. отсутствует иерархия для операторов и не допускаются скобки для вложения. Выражения могут содержать символ \square для указания того, что в поле должно быть подставлено текущее значение адресного счетчика. Выражения завершаются запятой или концом строки, за исключением случая, когда в качестве поля они используются в *FF*, где заключаются в скобки. Выражения могут продолжаться на нескольких строках, содержать константы, имена констант или метки. Например, если *SBB* является именем формата, а первое переменное поле должно содержать значение 3, то оно может быть записано как *SBB 1 + 2*, что является аналогичным *SBB 3*. Выражение *JMP \square - 5* дает текущее значение адресного счетчика минус пять в качестве подстановки для первого переменного поля в имени формата *JMP*. Выражение *EIGHT 2*2*2* означает *EIGHT = 8*.

(табл. 4.4). В файле определений переменные поля могут содержать и атрибут, как, например,

*ADE: DEF 3V: H # 0, 3V*B # 000, 3V%B # 000*

Теперь генерируются операторы ассемблируемого файла, приведенные в табл. 4.5.

Т а б л и ц а 4.3. Подстановки

Строка	Результат	Описание
<i>TEST6: ADE., 010 или TEST7: ADE., Q # 2</i>	000000010	Поля 1 и 2 имеют значения по умолчанию, а поле 3 представляет собой 010
<i>TEST8: ADE Q # 4 ..., B # 101</i>	100000101	Поле 2 имеет значение по умолчанию, поле 1 представляет собой 100, а поле 3 — 101
<i>TEST9: ADE 011</i>	011000000	Поля 2 и 3 имеют значения по умолчанию, а поле 1 представляет собой 011

Т а б л и ц а 4.4. Подстановки с модификаторами

Строка	Результат	Описание
<i>TEST10: ADE., 101*</i>	000000010	Поля 1 и 2 имеют значения по умолчанию, а поле 3 представляет собой инвертированное значение 101
<i>TEST11: ADE H # 4</i>	100000000	Поле 1 представляет собой шестнадцатеричное значение 4 (двоичное 0100), отсекаемое до 100. Поля 2 и 3 имеют значения по умолчанию

Т а б л и ц а 4.5. Подстановки с атрибутами

Строка	Результат	Описание
<i>TEST12: ADE., 01*</i>	000111010	Поле 1 имеет значение 000 по умолчанию. Поле 2 имеет по умолчанию 111, инвертируемое в 000. Поле 3 инвертируется на 10, затем выравнивается вправо, чтобы стать равным 010
<i>TEST13: ADE 9, /Q # 3*,1</i>	001011001	Поле 1 представляет собой шестнадцатеричное число 9, отсеченное до 001. Поле 2 — восьмеричное число 3, инвертированное на 100, затем инвертированное атрибутом * на 011. Поле 3 представляет собой двоичную единицу, выравненную вправо до 001

Любое значение, заданное как подстановка переменного поля (*VFS*), должно содержать число бит, специфицированное в файле определений для полной длины переменного поля, если не заданы модификаторы % (выравнивание вправо); : (отсечение) или □ (страничная адресация). Модификатор □ применяется в ассемблируемом файле в двух случаях. Во-первых, для указания того, что текущее значение адресного счетчика должно быть подставлено в переменное поле (это называется относительной адресацией), и, во-вторых, в качестве атрибута для указания того, что значение, подставленное в это поле, должно относиться к той же самой странице памяти, что и микрокоманда, в которую оно подставляется (это называется страничной адресацией).

При относительной адресации символ □ как отдельное значение или часть выражения применяется в качестве *VFS*. При страничной адресации □ может быть задан как атрибут переменного поля в файле определений или непосредственно следовать за *VFS* в операторе ассемблируемого файла. Например, если файл определений содержит

```
JSR: DEF 8X, 8V□, H # 27, 12VH #
```

```
JSB: DEF 8V%D #, 8X, 8Q # 013 :, 12X
```

то в ассемблируемом файле может быть записано

```
JSR BEGIN, OBC
```

```
JSB MULT□ + 5
```

```
JSR MULT, BEGIN □
```

```
JSB H # 37
```

```
JSB□ + 5
```

```
⋮
```

```
BEGIN:
```

```
⋮
```

```
MULT:
```

Здесь строки 1—3 являются примерами использования модификатора для страничной адресации. В строке 1 значение адресного счетчика (т. е. адреса, соответствующего метке *BEGIN*) подставляется в первое переменное поле формата *JSR*. Это значение отсекается слева, если необходимо выдерживать соответствие восьмиразрядному полю, а значения разрядов, отсеченных слева, должны быть идентичны соответствующим значениям разрядов адресного счетчика, связанным со строкой 1. Тот же тип подстановки и отсечения имеет место и для строк 2, 3.

Для использования страничной адресации после *MULT* в строке 2 необходим атрибут □, поскольку он не был задан с этим переменным полем в файле определений. Для выражений, таких как в строке 5, константа (пять) прибавляется к значению метки (*MULT*) до того, как

осуществляется проверка условия, что подставленное значение находится по-прежнему на той же странице. Оператор *JSR* в строке 1 не требует указания атрибута \square при *BEGIN*, поскольку данное переменное поле содержало атрибут \square в файле определений. Напротив, оператор *JSR* в строке 3 требует указания \square после *BEGIN*, ибо второе переменное поле не содержит атрибута \square в файле определений. В строке 2 метка с модификатором \square может быть частью выражения. Строка 5 является примером относительной адресации. Текущее значение адресного счетчика, увеличенное на пять, будет подставляться в переменное поле. Отметим, что отсутствует связь между символом \square , применяемым для страничной адресации в качестве атрибута для переменного поля, и символом \square , применяемым как подстановка переменного поля.

Спецификатор *H #*, задаваемый в файле определений с переменным полем, является постоянным атрибутом, но при необходимости должен быть повторен в ассемблируемом файле. Эта необходимость возникает из-за того, что ассемблер не может различать шестнадцатеричное значение, которое начинается с любой буквы от *A* до *F* от метки или имени формата. Так, если файл определений содержит строку

AM2901: DEF 8V% H #, Q # 0, 21X

а оператор ассемблируемого файла записан как *AM2901 3A*, то ясно, что цифры *3A* должны быть подставлены в переменное поле (ибо метка или имя не могут начинаться с цифры). Напротив, оператор *AM2901 AB* однозначно не указывает, означает ли *AB* имя константы или значение *AB*, заданное шестнадцатеричными цифрами. Если программист имеет в виду шестнадцатеричное значение *AB*, то он должен записать *AM2901 H # AB*.

Таблица символов ассемблера содержит список всех символов (имен констант), определенных с помощью *EQU*, а также всех меток в ассемблируемом файле. Таблица символов включает все имена констант и связанные с ними значения, введенные с помощью *EQU* в файле определений. Для каждого символа в таблице приводятся метки и значение адресного счетчика, при котором метка была определена. Если символ представляет собой имя константы, то за именем следует ее значение. Таблица символов является полезной для выявления синтаксических ошибок или пропусков двоеточия после метки. Печать таблицы символов не является обязательной процедурой и задается с помощью директив *SYMBOL* или *NOSYMBOL* при трансляции (см. § 4.4).

Таблица точек входа, генерируемая ассемблером, содержит список всех символов точек входа (меток, следующих перед символом ::) и связанных с ними значений адресного счетчика. Эти значения являются полезными для задания начальных адресов микропрограмм в ППЗУ. Печать таблицы точек входа не является обязательной и управляется с помощью директив *MAP* и *NOMAP* при трансляции (см. § 4.4).

Следующие слова являются служебными (используемыми на фазе ассемблирования) и не могут обозначать метки в операторах ассембли-

руемого файла, идентификаторы формата, имена констант и определений:

*ALIGN, EJECT, END, EQU, FF, LIST,
NOLIST, ORG, RES, SPACE, TITLE.*

Информация, содержащаяся в § 4.2, 4.3, позволяет «расшифровать» многочисленные микропрограммы, приведенные в [4] и представляющие большой интерес для пользователей комплекта K1804.

4.4. ТРАНСЛЯЦИЯ И ПОСТОБРАБОТКА МИКРОКОДА

Прежде всего пользователю необходимо с помощью текстового редактора сформировать файл определений и ассемблируемый файл микропрограммы, содержание которого определяется конкретным применением и должно быть тщательно продумано. Подойдет любой текстовый редактор, работающий в операционной системе ОС-1800 или CP/M, например экранный редактор *WORDMASTER (WM)*. Подготовленные исходные текстовые файлы именуются и записываются на дискету.

Вообще обозначение файла имеет формат *IIIIIII.EEE*, где *IIIIIII* — имя файла (от одного до восьми символов), а *EEE* — тип файла (расширение имени), от одного до трех символов. Ниже будем употреблять *F* для обозначения символов имен файлов определений, а *M* — символов имен исходных файлов микропрограмм. Если на фазе 1 (фаза определения — *PHASE1* или *P1*) и фазе 2 (фаза ассемблирования *PHASE2* или *P2*) используются одинаковые имена, то *FFFFFFFF* будет идентично *MMMMMMMM*. Примеры типов файлов (рис. 4.7), используемых микроассемблером *AMDASM*:

<i>FFFFFFFF.DEF</i>	— исходный файл определений
<i>FFFFFFFF.TBL</i>	— выходной файл фазы 1
<i>MMMMMMMM.TBL</i>	— входной файл фазы 2
<i>MMMMMMMM.SRC</i>	— исходный файл микропрограммы
<i>FFFFFFFF.P1L</i>	— выходной листинг фазы 1
<i>MMMMMMMM.P2L</i>	— выходной листинг фазы 2
<i>MMMMMMMM.OBJ</i>	— выходной файл фазы 2 (объектный код микропрограммы)
<i>MMMMMMMM.MAP</i>	— выходной файл фазы 2 (точки входа и их значения).

При создании файлов с помощью текстового редактора типы *DEF* и *SRC* обязательно должны быть указаны.

После того, как операционная система выдала на экран дисплея символы «A > », необходимо ввести команду *AMDASM PHASEn*

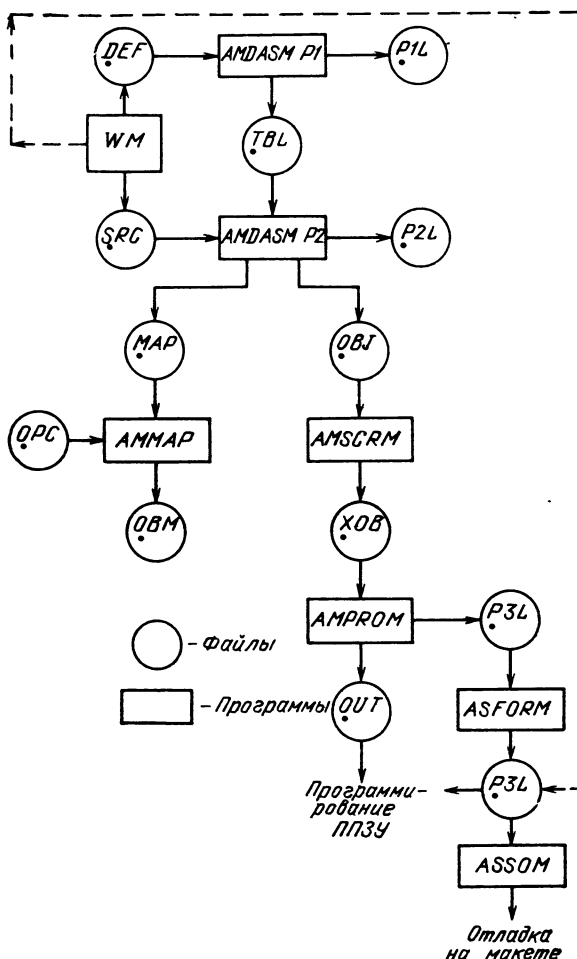


Рис. 4.7. Этапы формирования микрокода

имя файла {директивы} с возвратом каретки. Здесь n означает номер фазы. Вместо символа «=» может употребляться пробел, а вместо слова *PHASE* — буква *P*. Например,

$A > \text{AMDASM PHASE1 } B : \text{EXR}$

задает выполнение фазы определения с использованием файла *EXR*. *DEF*, находящегося на диске (дисковом) *B*, а команда

$A > \text{AMDASM PHASE1} = B : \text{EXR PHASE2} = B : \text{EXR}$

задает выполнение обеих фаз (определения и ассемблирования с использованием ранее созданных файлов *EXR. DEF* (как исходного для определения) и *EXR. SRC* (как исходного для ассемблирования), находящихся на диске (дисковом) *B*. Операционная система и *AMDASM* находятся при этом на дискете (дисковом) *A*.

Еще примеры (где *ASM. SRC* — исходный ассемблируемый файл, а *DEFIN. DEF* — исходный файл определений):

```
A > AMDASM P1 = DEFIN·DEF P2 = ASM·SRC
```

```
A > AMDASM P2 = B : ASM D = DEFIN·TBL
```

```
A > AMDASM P2 = B : ASM D = DEFIN
```

```
A > AMDASM P2 = B : ASM·SRC D = DEFIN·TBL
```

В табл. 4.6 приведены директивы управления, которые могут быть записаны в команде в любом порядке. Полное их название может быть задано, например, как *OBJECT*, сокращение — как *O*. Если директивы в команде не указаны, действуют их значения по умолчанию. В таблице предполагается, что файлы расположены на дисковом *A* (ведущем). В противном случае все выходные файлы, получаемые действием директив по умолчанию, будут помещены на ту же дискету, где находится исходный файл для соответствующей фазы. Директивы в команде следует разделять по крайней мере, одним пробелом, например: *NO SO NM NS*. Рассмотрим примеры использования директив.

```
A > AMDASM P1 = 2900 P2 = 2900
```

Здесь команда задает выполнение обеих фаз с использованием файла *2900. DEF* как входного для фазы 1 и файла *2900. SRC* для фазы 2. Все директивы имеют значения по умолчанию

```
A > AMDASM P2 = SYSTEM1 D = 2900
```

```
IL LINES = 67 WIDTH = 80
```

Эта строка вызывает выполнение фазы 2 с входным исходным файлом микропрограммы *SYSTEM1. SRC* и входным файлом *2900. TBL* определений. Далее следуют директивы управления печатью. В результате фазой 2 будут формироваться файлы: *SYSTEM1. OBJ* (объектный код микропрограммы); *SYSTEM1. P2L* (листинг фазы 2); *SYSTEM1. MAP* (символы и значения точек входа). Таким образом, имя файла можно указать лишь однажды; все следующие образуются автоматически по умолчанию.

Вывести файл на дисплей и АЦПУ можно и с помощью встроенной команды операционной системы *TYPE* (независимо от *AMDASM*):

```
TYPE SYSTEM1·P2L.
```

Каждая фаза трансляции может заканчиваться сообщениями об ошибках, если таковые допущены. Каждый оператор исходного файла обрабатывается до тех пор, пока не обнаружится ошибка, ибо одна про-

Т а б л и ц а 4.6. Директивы

Директива	Сокращенный вариант	Значение по умолчанию	Описание
<i>DEFTBL</i> имя файла или <i>DEFTBL</i> = имя файла	<i>D</i>	<i>FFFFFFFF.TBL</i> или <i>MMMMMMMM.TBL</i>	Указывает имя файла, куда записывается результат действия фазы определения. Если выполняется только фаза 2, то обозначает входной файл обработанных определений
<i>LIST1</i> имя файла или <i>LIST1</i> = имя файла	<i>L1</i>	<i>FFFFFFFF.PIL</i>	Указывает имя файла, куда нужно поместить результат определения. Если указано имя <i>LST</i> , то автоматически выполняется печать на АЦПУ
<i>LIST2</i> имя файла или <i>LIST2</i> = имя файла	<i>L2</i>	<i>MMMMMMMM.P2L</i>	Аналогично описанному, но указывает имя файла, куда следует поместить результат ассемблирования. По умолчанию используется <i>MMMMMMMM</i> — имя исходного ассемблируемого файла
<i>NOLIST</i>	<i>NL</i>	<i>FFFFFFFF.PIL</i> или <i>MMMMMMMM.P2L</i>	Запрещает листинг выхода фаз 1 и 2. Выход запоминается в файлах <i>FFFFFFFF.PIL</i> и <i>MMMMMMMM.P2L</i>
<i>OBJECT</i> имя файла или <i>OBJECT</i> = имя файла	<i>O</i>	<i>MMMMMMMM.OBJ</i>	Указывает имя выходного файла, содержащего объектный код микропрограммы
<i>NOOBJECT</i>	<i>NO</i>	<i>MMMMMMMM.OBJ</i>	Запрещает формирование микрокода, а также печать его в блочном формате
<i>INTER</i>	<i>IL</i>	<i>BLOCK</i>	Указывает расслоенный формат листинга (чередование строк исходного и объектного кодов)

Окончание табл. 4.6

Директива	Сокращенный вариант	Значение по умолчанию	Описание
BLOCK	<i>BL</i>	<i>BLOCK</i>	Указывает блочный формат листинга (начала весь исходный код, потом весь объектный)
<i>WIDTH = n</i> или <i>WIDTH n</i>	<i>W</i>	<i>n = 72</i>	Задаёт число <i>n</i> (десятичное) символов печатающего устройства в строке
<i>LINES = m</i> <i>LINES m</i>	<i>LN</i>	<i>m = 66</i>	Задаёт число <i>m</i> (десятичное) строк на странице. Значение по умолчанию равно 66 строкам (11 дюймам)
<i>MAP</i> имя файла или <i>MAP = имя файла</i>	<i>M</i>	<i>MAP</i>	Вызывает печать символов точек входа, т. е. меток, выделенных посредством двойного двоеточия
<i>NOMAP</i>	<i>NM</i>	<i>MMMMMMMM.MAP</i>	Блокирует печать символов точек входа. Результат запоминается в файле <i>MMMMMMMM.MAP</i>
<i>HEX</i>	<i>H</i>	<i>HEX</i>	Вызывает печать счетчика адресов в шестнадцатеричном формате
<i>OCTAL</i>	<i>O</i>	<i>HEX</i>	Вызывает печать счетчика в восьмеричном формате
<i>SYMBOL</i>	<i>S</i>	<i>SYMBOL</i>	Вызывает печать идентификаторов констант и меток, а также их значений
<i>NOSYMBOL</i>	<i>NS</i>	<i>SYMBOL</i>	Блокирует печать таблицы символов
<i>SRONLY</i>	<i>SO</i>	<i>BLOCK</i>	Печатается только текст исходного файла
<i>OBJONLY</i>	<i>OB</i>	<i>BLOCK</i>	Печатается только текст файла объектного микрокода

пущенная запятая между полями, например, повлечет в результате неправильную обработку оставшихся операторов. Ассемблер регистрирует ошибку и оператор, в котором она обнаружена, и переходит к обработке последующих входных операторов. Сообщение об ошибке *AMDASM* выдает в виде

*****ERROR *n*{*y*}.**

где *n* — число ошибок, а {*y*} если присутствует, то содержит некорректную строку или символ. Сообщения об ошибках появляются на дисплее и в выходном листинге ассемблера (файлах типа *P1L* и *P2L*). При *n* ≥ 100 ассемблер прекращает трансляцию. Могут быть выданы следующие сообщения об ошибках.

ОШИБКА 1: НЕДОПУСТИМЫЙ СИМВОЛ. Печатается символ, который не может быть интерпретирован, а также строка, в которой он появляется. Это сообщение может быть вызвано, например, ошибочным нажатием клавиши на клавиатуре, пропуском запятой или точки с запятой, использованием неправильной формы представления чисел; например, *B* # 3 или *Q* # 8 не могут быть интерпретированы.

ОШИБКА 2: НЕОПРЕДЕЛЕННЫЙ СИМВОЛ: Это сообщение будет наиболее часто появляться, если имеются синтаксические ошибки, пропущен знак # после *B*, *Q*, *D* или *H*, не поставлен пробел после *DEF*, *EQU*, *SUB*, *WORD*, *TITLE*, *RES*, *ORG*, *ALIGN*, *FF*, *SPACE*; на символ ссылаются прежде, чем он определен посредством *EQU* или *SUB*; подстановка *VFS* для шестнадцатеричного поля начинается с букв *A* — *F*, а спецификатор *H* # не предшествует букве.

ОШИБКА 3: НЕОПРЕДЕЛЕННЫЙ ФОРМАТ. Идентификатор формата содержит синтаксическую ошибку или не был определен на фазе определения, или после идентификатора формата нет пробела.

ОШИБКА 4: ДУБЛИРОВАННЫЙ ФОРМАТ. Идентификатор формата уже был использован. Если идентификаторы содержат более восьми символов, то первые восемь не должны повторяться.

ОШИБКА 5: ДУБЛИРОВАННАЯ МЕТКА. Метка применяется более одного раза в качестве идентификатора константы или метки. Если метка имеет более восьми символов, то первые восемь не должны повторяться.

ОШИБКА 6: ДУБЛИРОВАННОЕ СУБОПРЕДЕЛЕНИЕ: Идентификатор, предшествующий субформату *SUB*, уже применялся в качестве идентификатора. Если идентификаторы содержат более восьми символов, то первые восемь не должны повторяться.

ОШИБКА 7: ПЕРЕПОЛНЕНИЕ ФОРМАТА. Превышено максимальное число полей (128) в формате микрокоманды.

ОШИБКА 8: ПЕРЕПОЛНЕНИЕ ПОЛЯ СУБОПРЕДЕЛЕНИЯ: Превышено максимальное число полей (128) в формате субопределения.

ОШИБКА 9: НЕОПРЕДЕЛЕННАЯ ДИРЕКТИВА: Заданная директива не совпадает с *TITLE*, *WORD*, *LIST*, *NOLIST*, *END*, *ORG*, *RES*, *SPACE*, *ALIGN*, *EJECT*.

ОШИБКА 10: ЗАПРЕЩЕННАЯ РАЗРЯДНОСТЬ МИКРОКОМАНДЫ: Каждый раз, когда встречается *DEF* или *FF*, ассемблер следит за тем, чтобы суммарная разрядность всех полей идентификатора формата была точно равна разрядности микрокоманды. Если число бит в этом формате не будет точно равняться числу бит, заданному в *WORD*, то интерпретация неправильных *DEF* или *FF* не принимается во внимание и ассемблер осуществляет интерпретацию следующего исходного оператора.

ОШИБКА 11: ЗАПРЕЩЕННАЯ РАЗРЯДНОСТЬ ПОЛЯ. Рассчитанное значение для данного поля не умещается в 16 разрядов. Между тем никакое поле, за исключением «безразличного», не может превышать 16 разрядов.

ОШИБКА 12: ПРЕВЫШЕНА РАЗРЯДНОСТЬ ПОЛЯ БЕЗРАЗЛИЧНЫХ ЗНАЧЕНИЙ. Заданная разрядность безразличного поля превышает разрядность микрокоманды, специфицированную в *WORD*.

ОШИБКА 13: АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ НА ФИКСИРОВАННОМ ПОЛЕ. Когда поле определяется как переменное в файле определений, то выражение не может применяться в качестве подстановки в ассемблируемом файле, если поле не содержит атрибут % в своем определении.

ОШИБКА 14: ОШИБКА В АТРИБУТЕ. Как знак отрицания (—), так и знак инверсии (*) присвоены одной и той же переменной или константе, что недопустимо.

ОШИБКА 16: ОТСУТСТВИЕ ОПЕРАТОРА *END*.

ОШИБКА 17: ЗАПРЕЩЕННЫЙ СИМВОЛ. В идентификаторе использован символ, отличный от алфавитного *A—Z*, цифр от 0 до 9 или точки, либо между полями пропущена запятая.

ОШИБКА 18: ОШИБКА СОВМЕЩЕНИЯ. Это сообщение выдается, когда два формата перекрываются и оба содержат фиксированные значения в одинаковых по порядку разрядах. Например, если операторы в файле определений будут таковы:

A : DEF 4X, B # 1011

B : DEF B # 01111, 3X

а оператор ассемблируемого файла будет записан как *A&B*, то появится сообщение об ошибке совмещения.

ОШИБКА 19: НЕТ ЗНАЧЕНИЯ ПО УМОЛЧАНИЮ. Если значение по умолчанию не было задано в файле определений, то для данного поля обязательно должна обеспечиваться подстановка в явном виде, когда идентификатор формата используется в ассемблируемом файле.

ОШИБКА 20: ПРОТИВОРЕЧИЕ В РАЗРЯДНОСТИ ПОЛЯ. Рассчитанная или неявная разрядность поля для константы или выражения, заданная после спецификатора, не совпадает с разрядностью, заданной в явном виде. Это сообщение может быть выдано, когда пропущены запятые. Например, в *8H # A39Q # # 274* пропущена запятая между 3 и 9 и ассемблер предполагает, что число *A39* должно подставляться в восьмиразрядное шестнадцатеричное поле.

ОШИБКА 21: ЗНАЧЕНИЕ АДРЕСНОГО СЧЕТЧИКА \square СПЕЦИФИЦИРОВАНО ДЛЯ БЕЗАДРЕСНОГО ПОЛЯ. Для того чтобы в качестве подстановки использовать значение адресного счетчика, указанное как \square , данное поле должно содержать атрибут %.

ОШИБКА 23: ОТСУТСТВУЮЩИЙ СПЕЦИФИКАТОР. Встречается поле, которое содержит десятичные числа без указания их разрядности, что недопустимо для поля в *DEF*, *SUB* или *FF*. Десятичные числа должны вводиться как числа *nD #*, где *n* является разрядностью в явном виде.

ОШИБКА 24: ОШИБКА В ДИРЕКТИВЕ *SPACE*. Значение, следующее за *SPACE*, интерпретируется как меньшее нуля или большее числа строк, задаваемых на страницу.

ОШИБКА 25: ЗНАЧЕНИЕ *ORG* МЕНЬШЕ ТЕКУЩЕГО. При появлении *ORG* задаваемое значение сравнивается с текущим значением адресного счетчика и оказывается меньше.

ОШИБКА 26: ОТСУТСТВУЕТ ИДЕНТИФИКАТОР ФОРМАТА ПОСЛЕ *&*. При окончании строки знаком *&* и одновременном отсутствии символа переноса «*^*», который задается в начале следующей строки, диагностируется эта ошибка.

ОШИБКА 28: АДРЕС НЕ ПРИНАДЛЕЖИТ ТЕКУЩЕЙ СТРАНИЦЕ. Когда пользователь задает метку или метку с символом \square как подстановку, или определяет переменное поле посредством атрибута \square , это сообщение будет выдано, если подлежащие отсечению левые биты не согласуются с соответствующими битами текущего значения адресного счетчика.

ОШИБКА 29: ОТСУТСТВУЕТ РАЗРЯДНОСТЬ ДЛЯ МОДИФИКАТОРА \square . Страничная адресация (применение символа \square в качестве модификатора)

требует указания разрядности поля перед символом в операторах *FF*. Так, запись *6SYMBOL* \square является правильной, а *SYMBOL* \square — неправильной.

ОШИБКА 30: НЕДОПУСТИМАЯ РАЗРЯДНОСТЬ ПОЛЯ В ОПЕРАТОРЕ *FF*. Поле превышает 16 бит в операторе *FF*. Только «безразличные» поля могут быть больше 16 разрядов.

ОШИБКА 32: ДО СКОБКИ ОТСУТСТВУЕТ ЯВНАЯ РАЗРЯДНОСТЬ. Выражение в операторе *FF* должно быть заключено в скобки, а разрядность поля в явном виде должна предшествовать открывающей скобке.

Сообщения об ошибках с номером $n \geq 100$ вызывают останов. Они протоколируются так:

ОШИБКА 100: СИНТАКСИЧЕСКАЯ ОШИБКА в КОМАНДЕ. Директива содержит ошибку (проконтролируйте правильность записи имен файлов в директиве).

Ошибка 101: ПЕРЕПОЛНЕНИЕ ТАБЛИЦЫ *DEF*,

Ошибка 102: ПЕРЕПОЛНЕНИЕ ТАБЛИЦЫ *SUB*.

Ошибка 103: ПЕРЕПОЛНЕНИЕ ТАБЛИЦЫ *EQU*.

Ошибка 106: ПЕРЕПОЛНЕНИЕ ТАБЛИЦЫ ПОЛЕЙ.

Ошибки 101—103 и 106 встречаются, когда превышен имеющийся в распоряжении объем памяти.

ОШИБКА 104: НЕПРАВИЛЬНА ИЛИ ОТСУТСТВУЕТ РАЗРЯДНОСТЬ МИКРОКОМАНДЫ: Директива *WORDn* не задана первой (или первой после *TITLE*) либо значение n меньше единицы или больше 128.

ОШИБКА 105: НЕПРЕДВИДЕННОЕ ОКОНЧАНИЕ ФАЙЛА. Пользователь задал неправильный идентификатор файла или исходный файл содержит ошибку.

Числа 15, 22, 27, 31, 33 — 39 транслятором не используются для нумерации типов ошибок. Если при трансляции ошибок не обнаружено, выдается сообщение

TOTAL PHASE1 ERRORS = 0

или *TOTAL PHASE2 ERRORS = 0*

в зависимости от того, какая фаза выполнена. При обнаружении ошибок следует с помощью текстового редактора откорректировать соответствующий исходный файл и вновь повторить трансляцию. После устранения всех ошибок на дискете будут храниться: объектный файл (микропрограмма в кодах) типа *OBJ*; файл этого же микрокода, предназначенный для вывода на дисплей и АЦПУ (имеющий тип *P2L*); файл исходного текста микропрограммы в символах ранее введенных определений (типа *SRC*). Кроме того, для будущих нужд можно хранить файлы определений *DEF* и *PIL* (см. рис. 4.7).

Для дальнейшей обработки (постобработки) объектной микропрограммы (файла *OBJ* с выхода фазы 2 *AMDASM*) используются сервисные программы (рис. 4.7):

AMSCRM — перестановка разрядов и полей микрокоманды с учетом их физического распределения в модулях микропрограммной памяти.

AMPROM — разбиение общего массива микропрограммной памяти на области, соответствующие конкретным микросхемам памяти с заданной разрядностью и емкостью; замена «безразличных» значений разрядов на заданные.

АММАР — обработка таблицы точек входа и подготовка к программированию ППЗУ начальных адресов микропрограмм.

Кратко рассмотрим перечисленные фазы постобработки. Программа *AMSCRM* позволяет осуществить перестановку отдельных разрядов и полей в формате микрокоманды, т. е. в файле объектного микрокода, для того, чтобы привести их в соответствие с распределением между конкретными разрядами конкретных кристаллов ППЗУ микропрограммной памяти. В результате получается файл реорганизованного объектного микрокода (*ХОВ*), используемый далее программой *AMPROM*, которая будет подробно рассматриваться ниже. Выполнение *AMSCRM* задается командой *A > AMSCRM OLD = имя файла 1 NEW = имя файла 2*, где вместо символа равенства может употребляться пробел. Здесь имя файла 1 — это имя, присвоенное файлу микрокода, сгенерированного программой *AMDASM*; оно получается как *MMMMMMMM.OBJ*, если выполнение *AMDASM* осуществлялось без задания в явном виде имени объектного файла. Имя файла 2 — это определяемое пользователем имя файла, в который необходимо поместить реорганизованный микрокод. Имя этого файла не должно совпадать с первым. Удобно вводить различия в имена первого и второго файлов с помощью указателя типа (*OBJ* и *ХОВ*) при постоянной левой части имени, предшествующей точке. По умолчанию (если тип не указан в явном виде) программа *AMSCRM* трактует тип первого файла как *OBJ*, а второй сопровождает типом *ХОВ*.

По получении символа «возврат каретки» *AMSCRM* выдает сообщение *ENTER TRANSFORMATION PARAMETERS*. В ответ на это пользователь должен ввести параметры преобразования:

S0, D0, W0

S1, D1, W1

SN, DN, WN

Здесь S_i — номер позиции начального (крайнего левого) разряда i -го перемещаемого поля; D_i — номер позиции начального (крайнего левого) разряда, начиная с которого необходимо разместить i -е перемещаемое поле; W_i — разрядность i -го перемещаемого поля, $i = 1 \dots N$. Каждая группа из трех параметров завершается возвратом каретки, а последняя — еще и точкой. Крайний левый разряд в микрокоманде располагается в нулевой позиции, крайний правый в $(N - 1)$ -й позиции, если разрядность микрокоманды равна N бит.

Следить за корректностью сдвига полей — обязанность пользователя. Так, если он вводит параметры 13, 28, 4 с целью перемещения четырехразрядного поля, начинающегося с 13-го разряда микрокоманды, в поле, начинающееся с 28-го разряда (28—31), то необходимо ввести и 28, X , 4, где X — номер позиции, начальной для поля, ранее располагавшегося в разрядах 28—31, например $X = 13$.

Строки Колонки →		C1	C2	C3	C4	C5	C6	C7
Начальный адрес 'шестнадцатеричный') 0000	Разряды ↓ Слова	0-3	4-11	12-15	16-31	32-35	36-39	40-47
R1	0	ПЗУ #1	ПЗУ #2	ПЗУ #3	ПЗУ #4	ПЗУ #5	ПЗУ #6	ПЗУ #7
	255							
R2	256	ПЗУ #8	ПЗУ #9	ПЗУ #10	ПЗУ #11	ПЗУ #12	ПЗУ #13	ПЗУ #14
	767							
R3	768	ПЗУ #15	ПЗУ #16	ПЗУ #17	ПЗУ #18	ПЗУ #19	ПЗУ #20	ПЗУ #21
	895							
R4	896	ПЗУ #22	ПЗУ #23	ПЗУ #24	ПЗУ #25	ПЗУ #26	ПЗУ #27	ПЗУ #28
	1023							

Рис. 4.8. Карта распределения микрокода по кристаллам ПЗУ

После выполнения программ *AMDASM* и *AMSCRM* используется программа *AMPROM* для получения двоичного кода в форме, соответствующей организации ППЗУ микропрограммы.

Предположим, что микропрограммная память имеет организацию 1024 ячейки × 48 разрядов. И пусть эту матрицу 1024 × 48 бит необходимо «разрезать» на части, соответствующие конкретным кристаллам ППЗУ с различной организацией (рис. 4.8). Установив нумерацию строк и столбцов ППЗУ, а также разрядов в слове микрокоманды (слева направо), мы можем обращаться к ППЗУ # 19 либо по номеру 19, либо по номеру строки 3 и номеру столбца 5 одновременно. В данном примере все ППЗУ в строке 1 имеют емкость 256 слов. Разрядность ППЗУ # 1, # 3, # 5, # 6 равна четырем, ППЗУ # 2 и # 7 — восьми, а ППЗУ # 4 — шестнадцати (допустим, что существует и такой кристалл БИС). В строке 2 все ППЗУ имеют емкость 512 слов. Разрядность ППЗУ # 15, # 22, # 17, # 24, # 19, # 20 и # 27 равна четырем, ППЗУ # 16, # 23, # 21, # 28 — восьми, а ППЗУ # 18 и # 25 — шестнадцати.

Для печати содержимого (или программирования) ППЗУ # 1 будет использоваться массив с организацией 4 × 256. Для печати (или программирования) строки 3 будет использоваться массив (т. е. содержимое ППЗУ с # 15 по # 21) с организацией 4 × 128, 8 × 128, 15 × 128, 4 × 128, 4 × 128, 8 × 129. Для печати столбца 4 будет использоваться массив (т. е. содержимое ППЗУ # 4, # 11, # 13, # 25) в виде 16 × × 256, 16 × 512, 16 × 128, 16 × 128.

Формат командной строки для *AMPROM* имеет вид:

$a > AMPROM\ 0 = \text{MMMMMMMM} \cdot \text{EEE} \{ \text{директивы} \}$

Если тип файла не указан в явном виде, то по умолчанию принимается *OBJ*. Список директив приведен в табл. 4.7. Так, если выполнение *AMDASM* было задано в виде

A > AMDASM P1 2900 P2 2900

то полученный файл объектного кода называется 2900. *OBJ*. Для запуска *AMPROM* в этом случае достаточно ввести строку

A > AMPROM 0 2900

и поскольку директивы *LIST* и *PUNCH* не заданы, то именами результирующих файлов по умолчанию станут 2900. *OUT* и 2900. *P3L*.

Выполнение *AMPROM* начинается с диалога: пользователь должен ответить на вопросы системы, касающиеся организации ППЗУ

Т а б л и ц а 4.7. Директивы

Директива	Сокращенный вариант	Значение по умолчанию	Функция
<i>OBJECT</i> имя файла или <i>OBJECT</i> = имя файла	<i>O</i>	Нет	Определяет имя файла, в котором содержится объектный код с выхода <i>AMDASM</i> . Если тип файла не обозначен, то принимается <i>OBJ</i>
<i>MAP</i>	<i>M</i>	<i>MAP</i>	Печать содержимого памяти точек входа
<i>NOMAP</i>	<i>NM</i>	<i>MAP</i>	Запрет печати содержимого памяти точек входа
<i>HEX</i>	<i>H</i>	<i>HEX</i>	Перфорация в шестнадцатеричном формате
<i>BNPF</i>	<i>B</i>	<i>HEX</i>	Перфорация в формате <i>BNPF</i>
<i>INVERT</i>	<i>I</i>	Без инверсии	Все единицы заменяются на нули и наоборот, за исключением безразличных значений <i>X</i> полей микрокоманды
<i>PUNCH</i> имя файла 2 или <i>PUNCH</i> = имя файла 2	<i>P</i>	Имя файла 1. <i>OUT</i>	Определяет имя файла или устройства для выдачи данных на перфорацию
<i>NOPUNCH</i>	<i>NP</i>	Имя файла 1. <i>OUT</i>	Запрещает перфорацию содержимого ППЗУ
<i>LIST</i> имя файла 3 или <i>LIST</i> = имя файла 3	<i>L</i>	Имя файла 1. <i>P3L</i>	Задаст имя файла для устройства печати, на которое следует выдать выходной листинг
	<i>NL</i>	Имя файла 1. <i>P3L</i>	Запрещает выдачу на печать. Результат запоминается в файле типа <i>P3L</i>

Т а б л и ц а 4.8. Правила подстановки параметров

Вопрос	Подстановка	Значение
<i>DON'T CARES?</i>	0 или 1	Безразличные значения X в микрокомандах заменяются на нули или единицы во всей микропрограмме
<i>ENTER PROM WIDTH</i>	n	n — десятичное целое число, равное разрядности ППЗУ. Если разрядность микрокоманды равна 60, а $n=8$, то восемь ППЗУ будут сгенерированы: первые семь полностью, последняя — с безразличными значениями в четырех правых разрядах
	$l * b$	l — десятичное целое, равное числу ППЗУ; b — десятичное целое, равное разрядности каждого из этих ППЗУ. Так, $3 * 4$ означает три ППЗУ разрядностью по 4 бит
	Комбинации $n, l * b$	Для рис. 4.8 можно записать 4, 8, 4, 16, $2 * 4$, 8. Допустимы любые комбинации с разделением запятыми,
<i>ENTER PROM DEPTH</i>	r	r — десятичное целое число, равное емкости каждого ППЗУ. Если они не полностью заполняются микрокодом, то оставшиеся ячейки принимаются безразличными
	$t * d$	t — десятичное целое, равное числу ППЗУ; d — десятичное целое, равное емкости каждого из ППЗУ. Так, $2 * 512$ означает два ППЗУ емкостью по 512 слов
	Комбинации $r, t * d$	Для рис. 4.8. можно записать 256, $512, 2 \times 128$. Разделителем цифр является запятая
<i>WHICH PROMS DO YOU WISH TO PRINT?</i>	Y	Y — десятичное целое число, номер выдаваемого на печать (перфоленту) ППЗУ. Так, 5 означает ППЗУ #5
	$YI - YN$	Десятичные целые числа — номера ряда ППЗУ. Так, 2—5 означает выдачу на печать (перфоленту) четырех ППЗУ: #2, #3, #4, #5
	Комбинации Y и $YI - YN$	Пример: 3,5—7,9 означает печать ППЗУ #3, #5, #6, #7, #9. Номера и группы номеров разделяются запятыми
	CS	Десятичное целое число — номер столбца в ряду ППЗУ (см. рис. 4.8) Так, CS означает печать всех ППЗУ столбца 4
	$CS1 - SN$	Печать нескольких столбцов ППЗУ: например, $CS1-6$ означает печать столбцов с первого по шестой
	Комбинации $CS, SI - SN$	Пример: $CS, 7-9, 11$ (символ S употребляется однократно, а номера разделяются запятыми)

Вопрос	Подстановка	Значение
	<i>RS</i>	Десятичное целое — номер строки. Так, <i>R1</i> означает печать всех ППЗУ строки 1
	<i>RS1—SN</i>	Печать ППЗУ всех строк с <i>S1</i> по <i>SN</i> , например <i>R1—4</i>
	Комбинация <i>RS, RS1—SN</i>	Аналогично комбинациям столбцов, например: <i>R1, 4—6, 11—13</i> .
	<i>N</i>	Буква <i>N</i> подставляется, если пользователь не обозначает содержимое ППЗУ
	<i>A</i>	Буква <i>A</i> подставляется, если необходимо распечатать все ППЗУ

(разрядности и емкости), а также задать определенные значения (единичные или нулевые) для замены безразличных значений в полях микрокоманды (табл. 4.8). Например

DON'T CARES? 1

*ENTER PROM WIDTH 4*8, 4*

ENTER PROM DEPTH 128

Если предписана печать содержимого ППЗУ, то необходимо указать, каких именно секций:

WHICH PROMS DO YOU WISH TO PRINT? 5—7

Программа *AMMAP* выполняет функции, аналогичные *AMPROM* для ППЗУ начальных адресов (точек входа) микрокоманд. Таблица точек входа генерируется транслятором *AMDASM* и содержится в файле *MAP* по умолчанию. Коды операций (команд) задаются файлом типа *OPC* (см. рис. 4.7), а результатом является объектный код в файле типа *OBM*. При обнаружении ошибок в задании директив пользователем программы *AMMROM* и *AMMAP* выдают соответствующие диагностические сообщения. Полученные файлы используются на следующих этапах разработки изделия для отладки микропрограмм, программирования микросхем ППЗУ и документирования результатов (исходных и объектных файлов микропрограмм, таблиц программирования ППЗУ). Примеры таких файлов можно найти в [4].

4.5. ОТЛАДКА МИКРОПРОГРАММ

Наиболее трудоемкий этап в процессе разработки микропрограммного процессора — это отладка микропрограмм. Использование микроассемблеров, таких как *AMDASM*, позволяет избавиться лишь от тех ошибок в микропрограмме, которые проявляются на уровне син-

таксиса языка микропрограммирования. Все остальные (смысловые) ошибки связаны с неправильным представлением разработчика о процессах, происходящих в аппаратуре, с ошибками в структурных и схемных решениях, а также в вопросах синхронизации. Кроме того, трудно избежать случайных ошибок при изготовлении сложных макетных образцов, предназначенных для отладки. Выявить все перечисленные ошибки за короткий срок — непростая задача. Для более эффективного ее решения удобно пользоваться отладочными средствами.

В отладке микропрограмм можно выделить два основных этапа: детальную статическую проверку логики функционирования разработанного устройства при выполнении микропрограммы и контроль сбоев на рабочей частоте во взаимодействии с прочими компонентами микропроцессорной системы. Если для первого этапа в какой-то мере годятся кросс-средства (программы моделирования процессора на другой ЭВМ), то второй этап неизбежно выполняется на аппаратуре. Переход от кросс-средств к аппаратному макетированию может породить массу неожиданностей из-за возможной неадекватности программных моделей реальным схемам. Поэтому, как отмечалось выше, наиболее предпочтительны программно-аппаратные инструментальные комплексы, позволяющие с помощью единых средств писать микропрограммы и выполнять их отладку на натурных образцах.

Этап отладки поддерживается совокупностью аппаратных и программных средств. Аппаратура служит для подачи воздействий на отлаживаемое устройство и приема реакции с него на реальной частоте функционирования (в развитых инструментальных комплексах) или в статическом режиме (в более простых комплексах). Программное обеспечение служит для организации разнообразных отладочных режимов. Программа-отладчик (или отладочный монитор) управляет аппаратурой в соответствии с директивами, задаваемыми пользователем. Набор директив является фактически типовым во всех инструментальных комплексах. Он включает директивы загрузки с диска и выгрузки на диск содержимого отладочного ОЗУ микропрограмм, запуск выполнения микропрограммы с заданного адреса на заданное число тактов или до заданного адреса микрокоманды, выполнение одиночной микрокоманды и так далее.

В качестве конкретного примера рассмотрим возможности отладочного монитора *ASSOM*, предназначенного для статической отладки микропрограмм в инструментальном комплексе на базе микроЭВМ *СМ-1800*, аппаратной частью которого является простая в изготовлении плата сопряжения (см. § 4.1) [24].

Отладочный монитор работает с пользователем в простом диалоговом режиме на русском языке, что способствует легкому его освоению. В данной версии отладочного монитора заложены следующие ограничения на параметры отлаживаемых устройств:

- разрядность микрокоманды — не более 96;
- разрядность трассы контролируемых точек — не более 64, из них адрес следующей микрокоманды — не более 12 бит;
- число синхросерий, подаваемых на отлаживаемый макет, — не более 8;
- число разрядов трассы, значения которых могут использоваться как условие останова при выполнении микропрограммы, — не более 64 в любой совокупности;

число повторений условия останова до остановки выполнения микропрограммы — от 1 до 2^{15} (реально имеет смысл использовать от единиц до десятков);
длина (число тактов) запоминаемой трассы — несколько сот (зависит от рядности трассы и свободного пространства на дискете), реально имеет смысл анализировать несколько десятков шагов;

время, необходимое на все действия, связанные с выполнением одной микрокоманды, — от долей секунды до нескольких секунд (в зависимости от длины файла микропрограммы). Рекомендуется отлаживать микропрограммы модулями размером не более 100—150 микрокоманд.

Файл отлаживаемой микропрограммы необходимо предварительно подготовить в соответствующем формате на рабочей дискете. Данный отладочный монитор использует представление микропрограммы в виде текстового файла. Поэтому файл можно создавать и модифицировать с помощью обычного экранного редактора (в простейших случаях) или в результате применения описанных выше сервисных программ *AMDASM*, *AMSCRM*, *AMPROM*. Для преобразования выходного текстового файла программы *AMPROM (P3L)* к требуемому виду служит вспомогательная программа *ASFORM*, которая выполняет и некоторые факультативные функции по размещению микропрограмм в памяти.

Текстовый файл выполняемой микропрограммы должен удовлетворять следующим требованиям:

микропрограмма представляется в виде последовательности микрокоманд в порядке возрастания их адресов;

каждая микрокоманда начинается с новой строки и может занимать несколько строк. Первая из строк начинается адресом микрокоманды — полем из четырех шестнадцатеричных цифр, за которыми через пробел следуют разряды микрокоманды. Для улучшения визуального восприятия многоразрядных микрокоманд между группами разрядов рекомендуется вводить пробелы;

значениями разрядов микрокоманды могут быть 0, 1, X (при выполнении заменяется на 0).

Пример организации файла микропрограммы:

```
0000 00001111 01010101 XXXX1111 111X0001
      1110110X 10101111
      .
00AC 000000000 1011110X 111XXX10 10011110
      XXXXXXXX00 X1110000
```

Файл может размещаться на дискете *A* или *B* и именуется стандартным образом, например *MICRO.TST* или *B:FLOAT.P3L*.

Взаимодействие пользователя с отладочным монитором осуществляется посредством директив (управляющих команд). В процессе работы монитор может находиться в двух режимах: ожидания директивы (при этом на экране дисплея высвечивается символ *, приглашая к вводу директивы) или выполнения ранее заданной директивы. Для задания одной из двенадцати директив достаточно ввести при включенном русском регистре первые буквы ключевых слов (ниже они подчерк-

нуты), остальные буквы допечатываются автоматически. Директивы таковы:

1. Список управляющих команд
2. Открытие файла микропрограммы
3. Форматирование микрокоманды
4. Форматирование контрольных точек
5. Форматирование синхропоследовательности
6. Форматирование условия останова
7. Шаг с адреса AAAA
8. Запуск с адреса AAAA по BBBB
9. Вывод микропрограммы с адреса AAAA
10. Вывод контрольных точек
11. Такт
12. Конец

Определим действие директив.

Директива 1 служит для вывода на экран дисплея перечня управляющих команд монитора.

Директива 2 выполняет функции указания имени файла, содержащего отлаживаемую микропрограмму. — ДИСКОВОД: ИМЯ. РАСШИРЕНИЕ (например, *B: INIT. P3L*). Если микропрограмма находится на дисковом *A*, то можно его не указывать (будет принят по умолчанию). Ввод имени файла осуществляется по запросу монитора ВВЕДИТЕ ИМЯ ФАЙЛА, высвечиваемому на экране в ответ на задание пользователем директивы 1.

Директива 3 служит для задания разрядности микрокоманды и разбивки микрокоманды на поля с сопровождением их мнемониками (не более восьми символов). Тогда при распечатке микропрограммы директивой 9 поля микрокоманды (двоичные коды) будут сопровождаться заданными мнемониками. В ответ на указание директивы 3 монитор в диалоговом режиме предлагает пользователю задать разрядность микрокоманды, а также разрядности и мнемоники ее полей.

Директива 4 выполняет в диалоговом режиме аналогичные функции разделения 64-разрядной трассы на произвольное число полей (от 1 до 64) с сопровождением их мнемоническими обозначениями (не более восьми символов). При распечатке трассы директивой 10 соответствующие поля двоичных кодов будут сопровождаться мнемониками, что значительно облегчает их восприятие и последующий анализ. Нумерация разрядов трассы осуществляется слева направо от 1 до 64, при этом разряды 1—12 отведены под адрес следующей микрокоманды (слева старшие разряды адреса). Очевидно, их удобно сопоставить с отдельной мнемоникой. Остальные разряды трассы могут иметь произвольную смысловую нагрузку в зависимости от того, к каким точкам отлаживаемого макета они подключены через плату сопряжения.

Директива 5 служит для определения диаграммы синхропоследовательностей, которые должны подаваться на отлаживаемый макет в цик-

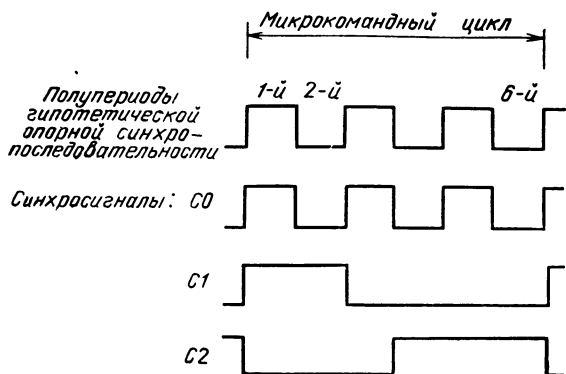


Рис. 4.9. Временные диаграммы синхропоследовательностей

ле выполнения каждой микрокоманды. Для этого цикл гипотетически разбивается на полупериоды некоторой опорной синхропоследовательности, и для каждого полупериода задаются значения сигналов (0 или 1) всех используемых синхропоследовательностей (не более восьми). Например, если на макет необходимо подавать три синхропоследовательности с повторяющимися временными диаграммами, представленными на рис. 4.9, то форматирование синхропоследовательностей выполняется так:

НОМЕР ПОЛУПЕРИОДА 1	ЗНАЧЕНИЯ СИГНАЛОВ 110
НОМЕР ПОЛУПЕРИОДА 2	ЗНАЧЕНИЯ СИГНАЛОВ 010
НОМЕР ПОЛУПЕРИОДА 3	ЗНАЧЕНИЯ СИГНАЛОВ 100
НОМЕР ПОЛУПЕРИОДА 4	ЗНАЧЕНИЯ СИГНАЛОВ 001
НОМЕР ПОЛУПЕРИОДА 5	ЗНАЧЕНИЯ СИГНАЛОВ 101
НОМЕР ПОЛУПЕРИОДА 6	ЗНАЧЕНИЯ СИГНАЛОВ 001

При выполнении микропрограммы заданная диаграмма трех последовательностей будет повторяться на выходах *С0*, *С1*, *С2* платы сопряжения (см. § 4.1). В рамках этой же директивы устанавливается, в какие моменты (т. е. полупериоды) диаграммы цикла необходимо снимать информацию с макета для формирования трассы, а также какие полупериоды соответствуют моментам смены информации на входе регистра микрокоманд устройства пользователя. Последняя возможность предусмотрена для схемных решений (см. § 2.5), где в цикле выполнения одной микрокоманды необходимо дважды читать микропрограммную память (выборка микрокоманды по частям).

Директива 6 позволяет до выполнения микропрограммы задать любую кодовую комбинацию значений сигналов в 64-разрядной трассе контрольных точек, которая (комбинация) после заданного числа повторений будет служить условием остановки микропрограммы. Например:

НОМЕР ТОЧКИ 13	ЗНАЧЕНИЕ 0
НОМЕР ТОЧКИ 14	ЗНАЧЕНИЕ 1
НОМЕР ТОЧКИ 28	ЗНАЧЕНИЕ 0
НОМЕР ТОЧКИ 30	ЗНАЧЕНИЕ 1
ЧИСЛО ПОВТОРЕНИЙ ДО ОСТАНОВА 8	

Директива 7 предназначена для пошагового выполнения микропрограммы, начиная с адреса *AAAA*, задаваемого четырьмя шестнадцатеричными цифрами. По каждому нажатию клавиши возврата каретки выдается диаграмма одного цикла синхросигналов и микрокоманда, а также принимаются значения контрольных точек.

Директива 8 обеспечивает непрерывное выполнение микропрограммы, начиная с адреса *AAAA* и заканчивая адресом *BBBB* (адреса задаются четырьмя шестнадцатеричными цифрами), либо совпадение с условием останова, если оно предварительно было задано директивой 6. Для выхода из ситуаций закливания (когда микропрограмма не доходит до конечного адреса *BBBB* и не выполняется условие останова) по дополнительному запросу задается максимальное число шагов до останова.

Директивы 9 и 10 служат для вывода микропрограммы с мнемониками и трассы с мнемониками (полностью или по частям) на экран дисплея и, если необходимо, на печатающее устройство.

Директива 11 может использоваться в тех случаях, когда перед выполнением микропрограммы необходимо подать на макет устройства несколько синхроимпульсов для установки его в требуемое «исходное» состояние. Каждому нажатию клавиши возврата каретки будет соответствовать выдача в порт синхронизации двух полупериодов синхропоследовательности. Идентифицировать момент установки макета в требуемое состояние можно по значениям контрольных точек, если предварительно задать соответствующее условие останова (которое действует и при выполнении директивы 11). Затем условие останова можно изменить или отменить вообще, если оно использоваться не будет.

Директива 12 служит для выхода из отладочного монитора в режим операционной системы и уничтожения на диске вспомогательных файлов, формируемых монитором во время его работы.

Перечисленные директивы задаются пользователем в следующей типовой последовательности. В начале каждого сеанса работы следует выполнить: открытие файла микропрограммы; форматирование микрокоманды; форматирование синхропоследовательностей; форматирование контрольных точек. Директивы форматирования можно задавать в любой последовательности. Затем в произвольном порядке (как потребуется в ходе отладки) можно пользоваться директивами: форматирование условия останова; такт; шаг с адреса *AAAA*; запуск с адреса *AAAA* по *BBBB*; вывод контрольных точек; вывод микропрограммы с

адреса АААА. Сеанс работы необходимо завершить директивой конец. Полученные распечатки можно затем проанализировать «за столом», намечая стратегию отладки микропрограмм (а возможно, и аппаратуры макета) на следующий сеанс работы с инструментальным комплексом.

Отлаженные микропрограммы заносятся в ППЗУ и при необходимости проверяются с помощью логического анализатора непосредственно в составе разрабатываемого устройства, в штатном режиме его функционирования.

Г л а в а 5.

МИКРОПРОГРАММНЫЕ КОНТРОЛЛЕРЫ И ПРОЦЕССОРЫ

5.1. МИКРОКОНТРОЛЛЕРЫ

Как упоминалось во введении, основным критерием используемой классификации микропрограммных устройств на контроллеры и процессоры является число уровней управления функционированием устройства. Микропрограммные контроллеры имеют один уровень управления (прикладной алгоритм реализован в микрокоде), процессоры — два: программный и микропрограммный, причем прикладной алгоритм кодируется на первом из них. В класс процессоров здесь включаются как ведущие, центральные процессоры (см. § 5.2), так и ведомые (см. § 5.3) с двухуровневым управлением.

Микропрограммные контроллеры (микроконтроллеры) на основе комплекта БИС K1804 — это быстродействующие (до 8 млн. коротких операций в секунду), простые (как правило, одноплатные) специализированные устройства с умеренным энергопотреблением (8—20 Вт), предназначенные для обработки информации в реальном масштабе времени по относительно простому алгоритму. Условие простоты алгоритма продиктовано ограниченной емкостью МПП для хранения реализующей его микропрограммы. В типовом варианте эта емкость не превышает 4096 микрокоманд, каждая из которых предписывает «короткие» операции (арифметические, логические, сдвиговые, пересылочные) и выполняется за один такт длительностью 125—250 нс. На микропрограммном уровне эффективно реализуются разнообразные алгоритмические конструкции: условные и безусловные переходы, циклы с выходом по условию или с заданным числом повторений, микроподпрограммы [1]. Поэтому для многих задач управления объектами емкость МПП даже в 2048 микрокоманд оказывается достаточной.

На рис. 5.1 показан один из вариантов взаимосвязи микроконтроллера с объектами управления, а точнее, с устройствами сопряжения

(УС1, УС2, УС3), встроенными в эти объекты и осуществляющими преобразование входной аналоговой информации в цифровую форму, а цифровых кодов — в управляющие воздействия. Доступ к регистрам этих устройств осуществляется с использованием шин адреса (А) и данных (Д), сигналов чтения (ЧТ) и записи (ЗП). Для обращения к УС (по инициативе микроконтроллера) в РгА загружается требуемый код из поля микрокоманды, которое может быть совмещено с другим полем с целью сокращения разрядности микрокоманды. Затем сигналами ЧТ и ЗП синхронизируется передача данных в том или ином направлении. Очевидно, в это время БОД должен выполнить инструкцию *I* приема или выдачи данных, в данном случае — по двунаправленной шине (У). Синхронный интерфейс предполагает фиксированное время чтения/записи, которое обеспечивается микропрограммно. Для реализации асинхронного интерфейса с УС на мультиплексор условий *MUX* необходимо подать сигнал подтверждения ПВ (лог. 0 на выходе с открытым коллектором), тестируемый микропрограммой циклически. Ожидание прихода сигнала подтверждения можно реализовать с применением микросхемы К1804ГГ1 (см. § 1.5).

Через мультиплексор условий на ФАМ, основу которого составляет микросхема К1804ВУ4 (см. § 2.1), подаются также признаки результата из БОД (знак, ноль и другие) с выхода СТ, если БОД реализован на микросхеме К1804ВМ1. Заметим, что при использовании БИС К1804ВМ1 для достижения минимальной длительности такта между ФАМ и ММП необходимо разместить РгАМК (см. § 2.1).

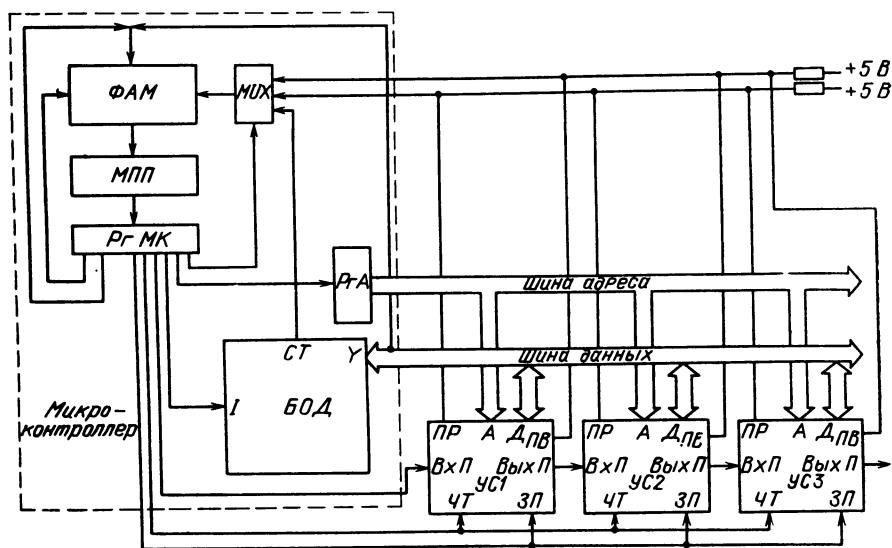


Рис. 5.1. Схема сопряжения микроконтроллера с объектом: асинхронный ввод-вывод, общая линия запроса прерывания

Обмен данными между УС и микроконтроллером может происходить не только по инициативе последнего, но и по запросам от УС, возникающим асинхронно. Для обеспечения быстрой реакции на такие запросы служит механизм прерываний. На рис. 5.1. показана одна из наиболее простых его реализаций. Каждое из УС имеет выход запроса прерываний (ПР) с открытым коллектором. Активный (нулевой) сигнал на этом выходе выставляется, если УС требует обслуживания (по готовности данных, в случае аварийных ситуаций и т. д.). При выполнении микропрограммы в ФАМ периодически тестируется состояние общей линии запроса прерывания (частота тестирования задается при написании микропрограммы). Если сигнал на этой линии нулевой, то выполняется переход к короткой микропрограмме, выявляющей источник запроса. Из РгМК выдается сигнал подтверждения прерывания, который проходит через цепочку УС (в порядке их приоритета). Устройство, выставившее запрос, блокирует дальнейшее прохождение сигналов подтверждения по цепочке (ВхП — ВыхП) и передает свой идентификатор на шину данных, откуда он поступает в микроконтроллер. Далее в зависимости от конкретной реализации этот идентификатор поступает либо в БОД, где затем сравнивается с образцами, и по результатам последовательного сравнения выполняется переход на соответствующую микропрограмму обработки прерывания, либо непосредственно в ФАМ (см. рис. 5.1), где на текущем такте используется для формирования адреса микропрограммы обработки прерывания (что, разумеется, быстрее, хотя и требует небольших дополнительных аппаратных затрат в ФАМ).

В целом эффективность схемы с общей линией запроса прерывания и цепочкой сигнала подтверждения (см. рис. 5.1) ограничивается двумя основными факторами:

- 1) частотой тестирования общей линии запроса (при редких проверках увеличивается время реакции на запросы прерывания, при частых — замедляется выполнение основного алгоритма);
- 2) невозможностью динамического изменения приоритетов устройств, так что при большой интенсивности запросов прерывания последние УС в цепочке могут длительное время оставаться необслуженными.

В режиме реального времени микроконтроллер обслуживает одно или несколько УС в зависимости от сложности алгоритма обработки данных и требуемой частоты выдачи управляющих воздействий (т.е. результатов вычислений) на УС.

В общем случае микроконтроллер соединяется с группой УС через шины данных/результатов (они могут быть отдельные или двунаправленные), шину адреса (иногда она совмещается с шиной данных путем временного мультиплексирования) и шину управления (она включает сигналы чтения и записи, подтверждения ввода-вывода, запросов и подтверждения прерываний и пр.). Конкретный набор и разрядность шин во многом зависят от специфики применений. Однако есть и известная общность в схемных решениях: она определяется

единым принципом микропрограммной организации этих устройств. В схемных реализациях рекомендуется использовать БИС К1804ВМ1, К1804ВУ4, К1804ВН1, К1804ГГ1, а также вспомогательные микросхемы комплекта К1804 и микросхемы сопутствующих серий: К1802 (умножители), К556 (микропрограммная память), К555, К155, К531 (вспомогательные логические элементы).

На рис. 5.2 показана схема взаимосвязи микроконтроллера с объектами управления, особенностью которой являются индивидуальные линии запроса прерываний и, как следствие, отсутствие цепочки сигнала подтверждения. Используемая здесь схема ФАМ детализирована на рис. 5.3. Она включает два дополнительных блока: контроллер прерываний К1804ВН1, на входы которого могут поступать до восьми линий запроса от УС (каскадирование микросхем К1804ВН1 с целью увеличения числа линий запроса рассмотрено в § 1.3), а также память векторов прерывания (ПВП).

С помощью кода маски M можно запрещать реакцию на запросы соответствующего уровня. Запрос с максимальным приоритетом из немаскированных приводит к появлению активного сигнала на выходе требования прерываний ($\overline{IR} = 0$), если приоритет этого запроса больше текущего приоритета, фиксированного в БИС К1804ВН1 (см. § 1.3). При выполнении микропрограммы периодически тестируется состояние выхода \overline{IR} передачей его на вход \overline{CC} микросхемы К1804ВУ4 при выполнении ею инструкции CJV . Если $\overline{CC} = \overline{IR} = 0$, то адрес следующей

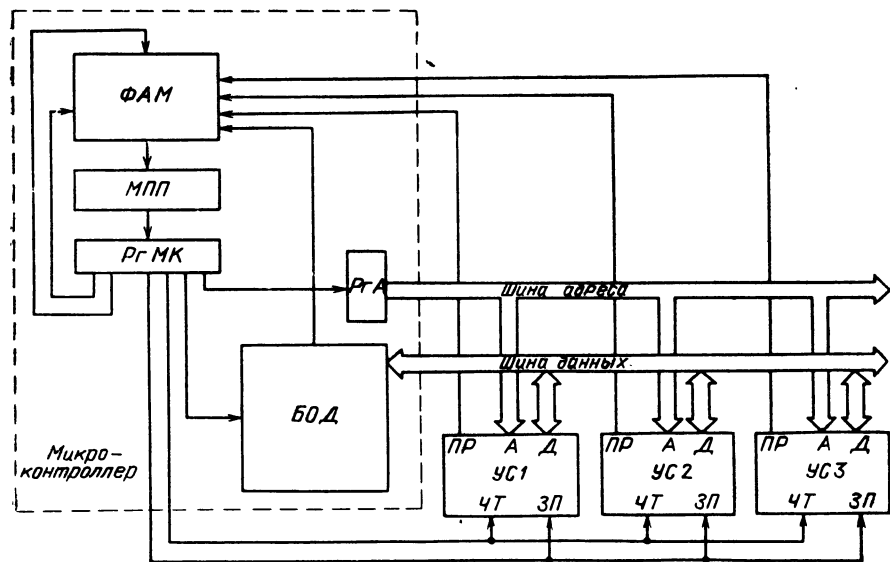


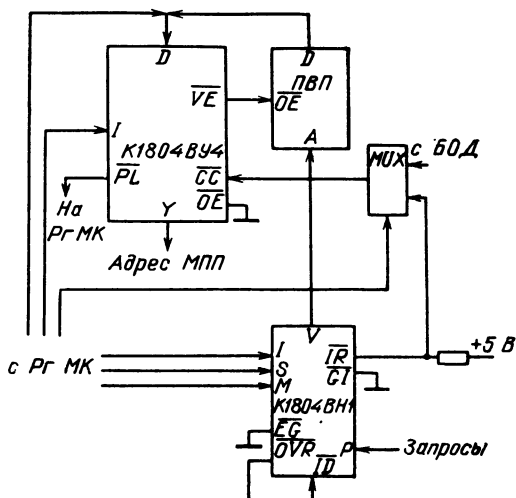
Рис. 5.2. Схема сопряжения микроконтроллера с объектом: синхронный ввод-вывод, отдельные линии запроса прерывания

Рис. 5.3. Компоненты блока формирования адресов микрокоманд.

микрокоманды на вход D микросхемы К1804ВУ4 поступит из ПВП при $\overline{VE}=0$ в соответствии с кодом запроса прерывания, передаваемым с выхода V микросхемы К1804ВН1 на адресный вход ПВП. Так будет выполнен переход к соответствующей микропрограмме обслуживания прерывания.

Для восьми уровней прерывания необходимая емкость ПВП составляет восемь ячеек, а разрядность равна разрядности адреса (9—12 бит). Следовательно, придется использовать две микросхемы ППЗУ с организацией 32 слова \times 8 бит, большая часть емкости которых окажется неиспользуемой. Если число линий запроса невелико, предпочтительнее другое схемное решение: с выхода V микросхемы К1804ВН1 код прерывания передается непосредственно в качестве младших разрядов адреса микрокоманды (без ПВП), тогда как старшие разряды адреса D имеют единичное значение благодаря резисторам, соединяющим линии «отключенной» тристабильной шины адреса D с шиной питания $+5$ В. Микросхема К1804ВН1 должна при этом выполнять инструкцию «Чтение вектора», а К1804ВУ4 — безусловный переход по адресу D (например, $JMAP$ или CJV при $\overline{CCE}=1$). Если же число уровней прерывания достаточно велико или старшая область микропрограммной памяти по каким-либо причинам не может быть отведена под начальные микрокоманды процедур обслуживания прерывания (т. е. под таблицу ветвлений), то наличие ПВП целесообразно, ибо допускает проверку значения \overline{IR} и переход к соответствующей микропрограмме за один такт (CJV).

Однако первый из двух недостатков, отмеченных для структуры на рис. 5.1, присущ и структурам на рис. 5.2, 5.3. Это необходимость периодического микропрограммного тестирования сигнала \overline{IR} , когда частое тестирование идет в ущерб основному алгоритму, а редкое сопряжено с замедлением реакции на запросы прерывания. Для преодоления этого недостатка удобно использовать схему (рис. 5.4), идея построения которой высказана в [4]. Схема обеспечивает реакцию на запрос не более чем за три такта (около 600 нс) без периодического микропрограммного тестирования запросов, что положительно отражается и на производительности микроконтроллера.



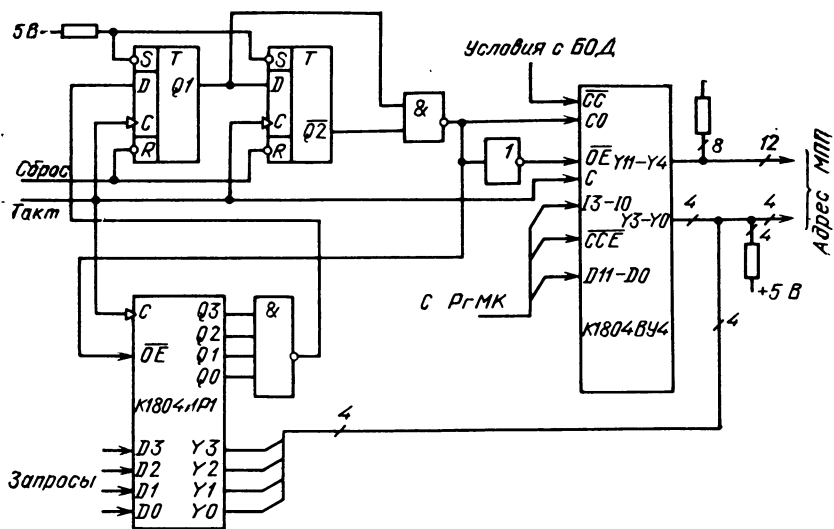


Рис. 5.4. Схема прерывания

Рассмотрим работу схемы на рис. 5.4 на протяжении нескольких тактов, полагая, что в первом такте установился активный (нулевой) уровень хотя бы на одной из входных линий запроса прерывания. Исходно оба триггера сброшены в нулевое состояние. Реакция на запрос начинается со второго такта, в начале которого (по положительному фронту синхросигнала) запросы будут фиксированы в регистре К1804ИР1, а собственно обработка прерывания — с четвертого по счету такта и заканчивается на N -м такте.

Такт 1. Поступление запроса (запросов) прерывания от объекта (объектов) управления: $Q1 = 0$, $Q2 = 1$, $C0 = 1$, $\overline{OE} = 0$. Микросхема К1804ВУ4 выполняет инструкцию 1 (например, *CONT*) с выработкой адреса $A + 1$ на выходе Y . Из микропрограммной памяти выбирается микрокоманда по адресу $A + 1$.

Такт 2. Фиксация запросов в регистре, появление единичного сигнала на входе первого триггера: $Q1 = 0$, $Q2 = 1$, $C0 = 1$, $\overline{OE} = 0$. Микросхема К1804ВУ4 выполняет инструкцию 1, выборка микрокоманды производится по адресу $A + 2$.

Такт 3. Переход к таблице ветвлений: $Q1 = Q2 = 1$, $C0 = 1$, $\overline{OE} = 1$, выборка микрокоманды по адресу 11111111 $Y3Y2Y1Y0$, младшие разряды которого являются функцией поступивших запросов. Адрес $A + 3$, вырабатываемый микросхемой К1804ВУ4, на выход ее не передается (из-за $\overline{OE} = 1$) и запоминается по очередному фронту синхросигнала в регистре СМК (см. § 2.1) без изменения (из-за $C0 = 0$).

Такт 4. В БОД выполняется первая операция, связанная с обработкой прерывания, а микросхеме К1804ВУ4 инструкцией 1 (*CJS*) с $\overline{CCE} = 1$ предписывается переход собственно к микроподпрограмме обработки прерывания по адресу D . По фронту очередного синхросигнала адрес возврата ($A + 3$) из регистра СМК будет переписан в стек, а новым содержимым этого регистра станет адрес $D + 1$. Значения сигналов: $Q1 = 1$, $Q2 = 0$, $C0 = 1$, $\overline{OE} = 0$. Осуществляется выборка микрокоманды по адресу D и подтверждение прерывания. Следует отметить, что если микропрограмма обслуживания прерывания состоит всего из одной микрокоманды, то на микросхему К1804ВУ4 вместо инструкции *CJS* должна подаваться

инструкция *CONT*, которая приведет к выборке следующей микрокоманды по адресу возврата $A + 3$.

Такт 5 ... такт N — 1. Выполнение микроподпрограммы обслуживания прерывания. Микросхема К1804ВУ4 функционирует обычным образом. Адрес $A + 3$ находится в стеке. Допустима вложенность микроподпрограмм, циклов и прерываний.

Такт N. Последняя микрокоманда обработки прерывания. Возврат в прерванную микропрограмму выполняется инструкцией *CRTN* при $\overline{CCE}=1$: адрес $A + 3$ будет извлечен из стека и использован в качестве адреса следующей микрокоманды.

Таким образом, 2^N ячеек микропрограммной памяти со старшими адресами необходимо отвести под таблицу ветвлений, если N — число линий запроса прерывания. При больших значениях N рассмотренная схема (см. рис. 5.4) в неизменном виде неэффективна: слишком большая часть микропрограммной памяти понадобится для таблицы ветвлений. Требуется приоритетная шифрация запросов при использовании их в качестве разрядов адреса.

В предыдущих примерах предполагалось, что микроконтроллер успевает обрабатывать входные данные в темпе их поступления с объекта управления. В других случаях входная информация с объекта в микроконтроллер поступает в виде массивов (размером от десятков до тысяч слов) с интервалом периодичности, достаточным для обработки полученного массива и формирования управляющих воздействий, в частном случае — также в виде массива. Поэтому состав микроконтроллера дополняется такими элементами, как быстродействующее статическое ОЗУ и быстродействующая схема управления прямым доступом к памяти. В зависимости от конкретных требований ОЗУ можно использовать как память с произвольным доступом либо как стековую память, снабдив его реверсивным счетчиком — указателем стека.

В качестве примера рассмотрим структуру (рис. 5.5) контроллера среднескоростной локальной сети с кольцевой топологией и децентрализованным управлением. Таким одноплатным контроллером снабжается каждая из абонентских ЭВМ, включаемых в локальную сеть с децентрализованным управлением. Интерфейсный блок (ИБ) служит для электрического сопряжения контроллера с передающей средой сети (коаксиальным кабелем), а также для преобразования передаваемых в канал данных из параллельного кода в последовательный, а принимаемых данных из канала — в параллельный код из последовательного.

Данные в канале циркулируют в виде пакетов (сообщений) постоянной или переменной длины в зависимости от конкретной организации сети. Оперативное запоминающее устройство предназначено для буферизации принимаемых пакетов с целью их последующего анализа и «распаковки», если адресатом является абонентская ЭВМ, подключенная к контроллеру. Информационный массив, сформированный из одного или нескольких сообщений, передается затем в ЭВМ в режиме прямого доступа к памяти. Режим прямого доступа обеспечивается блоком ПДП и используется также при передаче информации между ин-

лер прерываний (КП) K1804ВН1 и эти сигналы трактовать как запросы на прерывание. Обработка прерываний осуществляется на микропрограммном уровне. При необходимости для ее реализации в схему вводится ПЗУ векторов прерывания. Основу блока ФАМ составляет микросхема K1804ВУ4 (см. § 2.1), на выходе которой целесообразно поставить регистр адреса микрокоманд (РГАМК) для организации трехуровневой конвейерной обработки микрокоманд. Конвейерная обработка микрокоманд обеспечит предельное быстродействие контроллера на линейных участках микропрограмм: в противном случае потенциальное быстродействие микросхемы K1804ВМ1 будет использоваться лишь на две трети, а то и наполовину.

Для реализации блока ПДП используется микросхема K1804ВУ6 или K1804ВУ7 (см. § 1.4): двух микросхем достаточно для адресации ОЗУ емкостью до 64К слов (такую же разрядность адреса обеспечивает и микросхема K1804ВМ1 в режиме чтения/записи отдельных ячеек ОЗУ при формировании пакетов). Все регистры и интерфейсные компоненты также могут быть реализованы на микросхемах серии K1804 (см. § 1.6), а ПЗУ микропрограммной памяти — на элементах серии K556 с временем чтения не более 80 нс. Емкость ОЗУ (и выбор соответствующих микросхем с временем доступа 40 — 60 нс) определяется максимальным размером сообщений и перечнем функций, возлагаемых на контроллер.

На рис. 5.5 не изображены схемы синхронизации, однако следует иметь в виду, что для реализации их удобно использовать микросхему управляемого тактового генератора K1804ГГ1 (см. § 1.5). Таким образом, комплект микросхем K1804 может составлять основу аппаратной части быстродействующих одноплатных контроллеров самого различного назначения.

5.2. КОНВЕЙЕРНЫЕ ПРОЦЕССОРЫ КОМАНД

Комплект микропрограммируемых БИС — наиболее подходящая элементная база для построения процессоров с «нестандартными» системами команд, если под стандартными понимать системы команд микропроцессоров K580, K1801, K1810. Впрочем, при необходимости резко повысить производительность процессора с сохранением большого объема программного обеспечения, написанного на стандартной системе команд, для построения эмулятора также может быть использован комплект микропрограммируемых БИС. Например, одноплатный конвейерный эмулятор микропроцессора K580, реализованный на микросхемах серии K1804, обеспечит выполнение последовательности команд типа регистр — регистр в десять раз быстрее: по 200 нс на команду (в потоке) вместо 2 мкс, как в оригинале. Разумеется, до этого значения необходимо увеличить и быстродействие прочих компонентов микроЭВМ, чтобы получить соответствующий эффект от системы в целом. Реализация эмулятора системы команд K1801 на основе микросхем K1804 позволяет довести производительность примерно до 4 млн.

команд регистр — регистр в секунду, что также заметно выше производительности оригинала. Но растут и аппаратные затраты: микропроцессор из однокристалльного становится одноплатным.

Что касается процессоров с нестандартными системами команд, реализованных десятилетие назад на микросхемах малой и средней степени интеграции и «обросших» огромным программным обеспечением, то построение их эмуляторов на БИС серии K1804 выгодно как для уменьшения габаритно-массовых характеристик процессоров, так и для повышения их производительности. Последнее можно обеспечить за счет конвейерного режима обработки команд и микрокоманд с учетом накопленного в этой области опыта. Разработка же новых процессоров с нетиповыми системами команд—шаг, требующий веских оснований, ибо за ним последуют проблемы создания системного и прикладного программного обеспечения. В ряде случаев этот шаг может оказаться оправданным, например при построении аппаратных интерпретаторов языков высокого уровня; процессоров, ориентированных на быстрое выполнение задач трансляции языков высокого уровня; специализированных процессоров с нетрадиционной архитектурой.

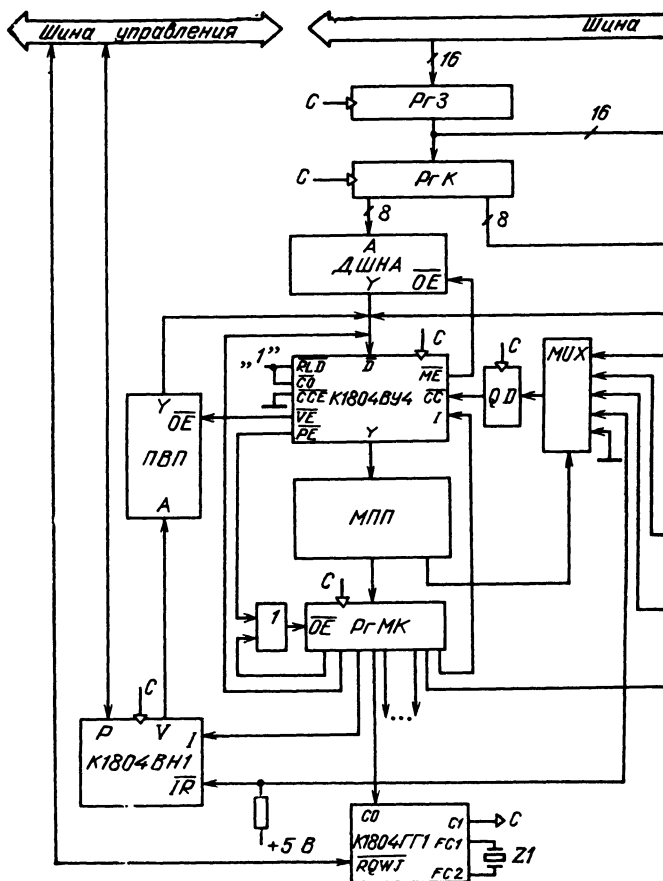
Ниже иллюстрируются хорошо зарекомендовавшие себя схемные решения, используемые при разработке процессоров команд. С организацией процессоров на основе БИС K1804BC1, K1804BC2 лучше всего ознакомиться по книге [4]. Особенности применения БИС K1804BM1 в процессоре команд составляют основное содержание настоящего раздела наряду с практическими советами общего характера.

Несмотря на то, что схема K1804BM1 ориентирована прежде всего на применение в контроллерах, она может эффективно использоваться и в процессорах. Причиной тому является наличие в ней разнообразных и мощных операций, высокое быстродействие, малая площадь, занимаемая на печатной плате. Описываемый ниже центральный процессор (ЦП), построенный с использованием схемы K1804BM1, является программно (на уровне системы команд) совместимым с процессором SUPER-16, описанным в [4]. Архитектура данного ЦП характеризуется наличием конвейерной обработки как команд, так и микрокоманд. Интерфейс не рассматривается, как и вопросы дополнения микроЭВМ различными периферийными устройствами ввода-вывода и хранения информации, арбитража требований на шину и т. д. Существенным будет лишь то, что для использования данных из ОЗУ на $(n + 1)$ -м такте необходимо сформировать адрес данных с помощью БИС K1804BM1 на $(n - 1)$ -м такте, произвести считывание данных из ОЗУ на n -м такте. Длительность этого такта может быть увеличена для согласования с временными параметрами ОЗУ с помощью БИС K1804ГГ1. Поскольку микросхемой K1804BM1 вырабатываются и адрес, и данные, то цикл записи данных в ОЗУ требует двух тактов. В зависимости от типа считываемой из ОЗУ информации (команда или данные) производится загрузка либо регистра данных (РгД), либо конвейерного регистра-защелки (РгЗ).

Хранимые в ОЗУ команды состоят из одного или двух 16-разрядных слов. Старший байт первого слова представляет собой код операции (КОП). Младший байт делится на два 4-разрядных поля, задающих регистры (РОН) — источник операндов и приемник результата. Регистры с 0-го по 15-й (из тридцати двух, имеющихся на кристалле БИС К1804ВМ1) доступны пользователю в качестве рабочих, остальные 16 регистров используются операционной системой для обслуживания стеков в ОЗУ, в качестве программного счетчика и т. д. Второе слово команды (если она двухсловная) является либо полем смещения, либо непосредственным операндом. Восьмиразрядный КОП позволяет закодировать 256 команд, чего с избытком хватает для универсальной микроЭВМ. Система команд включает операции над следующими типами данных: бит, четырехразрядное поле (полубайт), байт, слово. Информация о способе адресации содержится в КОП. Кроме того, система команд включает и команды обслуживания одного или нескольких стеков, команды ввода-вывода, команды десятичной арифметики и арифметики над двончными целыми числами. В зависимости от способа адресации регистр, обозначенный в формате команды, может служить как аккумулятор при выполнении арифметических и логических операций либо как индексный регистр для формирования адреса ОЗУ. Если приемником результата является РОН, в поле $R1$ указывается адрес этого РОН, а $R2$ (или $R2 + d$) является адресом операнда-источника в РОН (или в ОЗУ). В командах, где результат помещается в память, $R1$ есть адрес источника в РОН, а $R2$ (или $R2 + d$) — адрес ячейки-приемника в ОЗУ. Наконец, в командах типа память-память регистр $R1$ указывает приемник, а регистр $R2$ — источник операнда в ОЗУ.

Форматы команд имеют вид:

	15—8	7—4	3—0	
RR	КОП	$R1$	$R2$	
R	КОП	$R1$	$X2$	
SS	КОП	$X1$	$X2$	
	15—8	7—4	3—0	15—0
RX	КОП	$R1$	$X2$	Смещение d
RSI	КОП	$R1$	$X2$	Операнд I



Здесь $R1$ и $R2$ — адреса регистров приемника и источника операнда, а $X1$ и $X2$ — адреса индексных регистров. Команды условного перехода имеют формат, в котором поле M определяет выбор условия

15—8 7—4 3—0 15—0

JM

КОП	M	$X2$	Смещение d
-----	-----	------	--------------

для тестирования, а адрес перехода (при истинном значении условия) вычисляется суммированием содержимого регистра $X2$ со смещением d . При ложном значении условия переход осуществляется к следующей по порядку команде.

В схеме ЦП (рис. 5.6) все внутренние передачи информации осуществляются через 16-разрядную шину. Передачи между системной

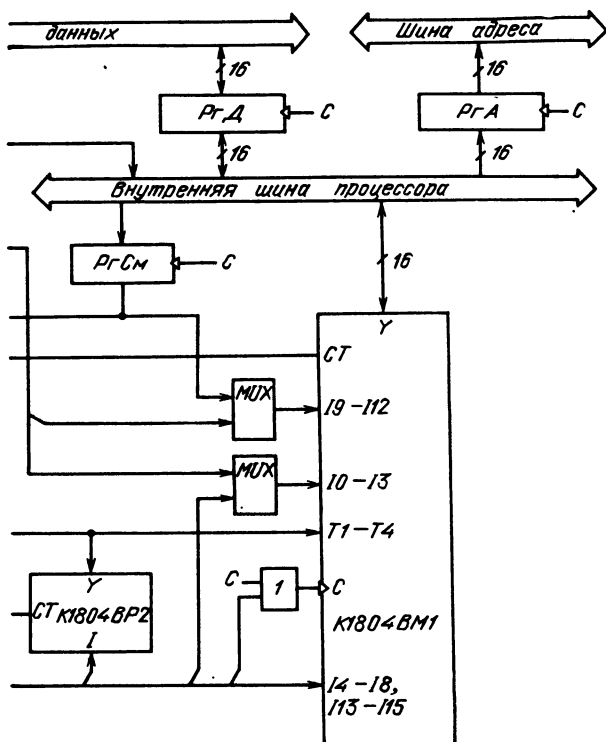


Рис. 5.6. Процессор команд

(внешней) шиной адреса/данных и внутрипроцессорной шиной выполняются через РгД, РгА и РгЗ. Микропрограммное управление организовано в конвейерную структуру: регистр на выходе микропрограммной памяти работает как конвейерный, обеспечивая одновременное функционирование операционной и управляющей частей микропроцессора, т. е. совмещение выборки и выполнения микрокоманд.

Регистр команд и РгЗ позволяют организовать конвейерную обработку команд. В то время как декодируется команда, содержащаяся в РгК, РгЗ может загружаться следующей командой (в режиме регистр — регистр), полем смещения (при индексной адресации) или непосредственным операндом. Команда из ОЗУ поступает в РгЗ или в РгК через РгЗ в «прозрачном» состоянии последнего: куда именно — зависит от способа адресации, указанного в декодируемой команде. Во время выполнения конвейера первая команда загружается в РгК. Все однословные команды попадают в РгЗ, после чего переме-

щаются в РгК по выполнении текущей команды. Поскольку двухсловная команда использует информацию как из РгК, так и из РгСм (смещение), то следующая команда из ОЗУ поступает сразу в РгК. Если содержимое РгЗ при декодировании двухсловной команды интерпретируется как смещение, то оно используется схемой К1804ВМ1 в цикле формирования адреса. Если же оно интерпретируется как непосредственный операнд, то используется в цикле выполнения команды.

Можно загрузить и выдать данные в /из К1804ВМ1 в течение одного такта. Загрузка осуществляется через шину Y с РгЗ или РгД на первой половине такта, когда значение синхросигнала $C = 1$. На второй половине такта (при $C = 0$) буферы схемы К1804ВМ1 устанавливаются в состояние, обеспечивающее прохождение результата с АЛУ на шину микропроцессора. Если и на второй половине такта шина Y должна работать как входная, то на дизъюнктор из РгМК следует подать «1».

Регистр РгСм служит для выполнения таких функций, как ветвление по N направлениям и нормализация. Шестнадцатиразрядное слово может быть зашифровано в схеме К1804ВМ1 под управлением маски. Пятиразрядный вектор загружается в РгСм и используется на следующем такте либо для ветвления на соответствующую микроподпрограмму нормализации, либо как число N в инструкции циклического сдвига.

Для выработки адреса следующей микрокоманды схема управления последовательностью микрокоманд К1804ВУ4 задействует при ветвлении следующие источники адреса: регистр микрокоманд, ППЗУ векторов прерывания (ПВП), ППЗУ дешифратора начальных адресов микропрограмм (ДШНА), регистр смещения РгСм (пять младших разрядов адреса).

Дешифратор начальных адресов получает разряды 15 — 8 кода операции РгК и путем их преобразования обеспечивает переход к микропрограмме каждой команды при выполнении инструкции *JMAP* схемой К1804ВУ4.

Для адресации ПВП используется вектор, вырабатываемый схемой К1804ВН1 и соответствующий устройству, требующему прерывание (имеющему максимальный приоритет из немаскированных). С выхода ПВП поступает стартовый адрес микропрограммы обслуживания прерывания при выполнении инструкции *CJV* схемой К1804ВУ4 с передачей на вход \overline{CS} через мультиплексор сигнала \overline{IR} для анализа.

Регистр РгСм обеспечивает возможность ветвления по N направлениям. Инструкция шифрации К1804ВМ1 формирует 5-разрядный вектор, соответствующий старшему единичному значению разряда слова. Этот вектор с доопределенными остальными разрядами (например, до единичных значений) может быть использован как адрес ветвления. Доопределение до единичных значений разрядов $D11 — D5$ на входе схемы К1804ВУ4 осуществляется соединением их через резисторы с шиной +5 В. Если выходы всех источников информации на входе

схемы К1804ВУ4 приведены в состояние высокого сопротивления, то резисторы обеспечат единичный потенциал. Сигнал разрешения (\overline{PE}) адресного поля РгМК на вход \overline{OE} передается через дизъюнктор для того, чтобы при выполнении ветвления по РгСм можно было применить инструкцию схемы К1804ВУ4, вырабатывающую $\overline{PE} = 0$, например CJP при $\overline{CC} = 0$. Поступающий на дизъюнктор разряд РгМК должен иметь единичное состояние для ветвления по РгСм или нулевое для перехода по полю РгМК.

Решение о ветвлении в схеме К1804ВУ4 принимается на основании значения входа \overline{CC} . Разряды состояния микросхемы К1804ВМ1 через шину $T1 - T4$ могут быть использованы для ветвления на следующем такте путем выбора одного из них через мультиплексор условия. Для выработки более сложных условий (\geq , \leq , $<$ и т. д.) применяется микросхема К1804ВР2, как и для перехода по значению ранее сформированных условий.

Установка динамического D -триггера на вход \overline{CC} микросхемы К1804ВУ4 позволяет разбить критический путь распространения сигнала при условных переходах на два более коротких, т. е. организовать конвейерную обработку микрокоманд совместно с РгМК. Поскольку этот D -триггер задерживает сигнал условия на один такт, то линии управления мультиплексором сигнала \overline{CC} поступают прямо с выхода МПП: это согласует выбор кода условия с выполнением микрокоманды.

Мультиплексор на входах $I0 - I3$ схемы К1804ВМ1 обеспечивает возможность адресации РОН как из полей $R1$ и $R2$ РгК, так и из микрокоманды. Аналогично число N (параметр сдвига) можно брать как из РгК, так и из РгСм: для этого используется мультиплексор на входах $I9 - I12$ схемы К1804ВМ1. Поскольку схема К1804ВМ1 имеет однопортовое РЗУ, требуется два такта для выполнения операции регистр — регистр. В первом такте содержимое $R2$ заносится в аккумулятор, во втором — операция производится с аккумулятором и $R1$ и результат заносится в $R1$.

Четыре дополнительных разряда микрокоманды управляют входами $T1 - T4$ схемы К1804ВМ1. Это позволяет одновременно выполнять операцию в АЛУ К1804ВМ1 и тестировать состояние. Четыре бита состояния (C , N , OVR , Z) можно загрузить или прочитать по этой шине. Использование БИС К1804ВР2 позволяет уменьшить число тактов, необходимое для манипуляций с регистром состояния. Чтобы можно было осуществлять ветвление в следующем такте по этим значениям C , N , OVR , Z , шина $T1 - T4$ тоже подключается ко входу мультиплексора сигнала \overline{CC} . Микросхема К1804ВР2 имеет два регистра состояния. Содержимое регистра «микросостояния» модифицируется на каждом такте, если шина $T1 - T4$ К1804ВМ1 является выходной. Регистр «макросостояния» модифицируется по окончании каждой команды. Тактовый генератор К1804ГГ1 обеспечивает четыре различных диа-

граммы такта и возможность динамического изменения длительности такта в соответствии с асинхронной работой интерфейса.

Цикл выполнения команды включает четыре этапа: формирование адреса команды, выборку команды, дешифрацию команды, выполнение команды. В тех командах, где операнды находятся не в РЗУ, а в ОЗУ, требуются два дополнительных этапа после фазы дешифрации: формирование адреса операнда и выборка операнда. Повышение производительности достигается за счет предварительной выборки следующей команды одновременно с выполнением текущей. В конвейере здесь реализовано только частичное перекрытие этапов различных команд, так как схема К1804ВМ1 используется и как операционный блок, и как блок программного управления. Отдельная реализация последнего позволила бы довести производительность конвейера до предельной. Рассмотрим каждый этап.

Этап 1. Формирование адреса (один такт). Программный счетчик (РС) увеличивается на 2, и результат загружается в регистр адреса памяти (РГА), а также обратно в программный счетчик. Для этого в К1804ВМ1 в качестве приемника устанавливается РЗУ и $\overline{OEY}=0$.

Этап 2. Выборка команды (один такт). Генерируются интерфейсные сигналы чтения ОЗУ. Команда, считанная из ОЗУ, помещается на шину данных и загружается в РгК на фронте синхроимпульса в начале следующего такта.

Этап 3. Дешифрация команды (один такт). Команда попадает через РгЗ в РгК. Схема дешифрации начального адреса (ДШНА) генерирует стартовый адрес микропрограммы, соответствующей данному коду операции.

Этап 4. Формирование адреса операнда (один такт). После выборки команды производится еще один цикл выборки. Информация загружается в РгЗ. В зависимости от способа адресации, который распознается во время дешифрации, определяется, находится ли в РгЗ операнд, смещение или следующая команда. Смещение используется с указанным индексным регистром для формирования адреса операнда и загружается в РГА.

Этап 5. Выборка операнда (один такт). Операнд считывается из памяти и перемещается в РгЗ.

Этап 6. Выполнение операции. Для выполнения команд типа регистр — регистр (RR) в К1804ВМ1 требуется два такта. В двухадресных архитектурах К1804ВС1, К1804ВС2 на это расходуется один такт.

Цикл выполнения команды регистр — регистр приведен в табл. 5.1. Очередная выборка производится после выборки команды для заполнения конвейера, и информация заносится в РгЗ. После исполнения текущей команды следующая команда из РгЗ перемещается в РгК. Одновременно адрес следующей команды помещается на шину адреса. На следующем такте из ОЗУ информация поступает в РгЗ. Таким образом, в установившемся режиме конвейера микросхема К1804ВМ1 используется в каждом такте либо для выполнения операции, либо для вычисления следующего адреса (табл. 5.1). Цикл выполнения команд

типа регистр — индексируемая память (*RX*) показан в табл. 5.2. При заполнении конвейера выборка смещения производится сразу за командой. В установившемся режиме формирование следующего адреса команды и выборка операнда могут осуществляться одновременно. Схема K1804BM1 используется здесь в четырех тактах из пяти.

Назначение разрядов микрокоманды

- 0—15 — инструкция K1804BM1;
- 16 — разрешение загрузки K1804BM1 и запрещение выхода *Y* при $C = 0$;
- 17 — разрешение инструкции;
- 18 — разрешение *PgC*;
- 19 — разрешение выхода *T1* — *T4*;
- 20, 21 — выбор источника *I0*—*I3*;
- 22, 23 — выбор источника *I9*—*I12* и разрешение выхода *PgC*;
- 24 — разрешение чтения *Pg3* на шину;
- 25 — 28 — управление *PgД*;
- 29 — разрешение загрузки *PgA*;
- 30 — разрешение загрузки *PgK*;
- 31 — разрешение загрузки *Pg3*;
- 32 — разрешение загрузки *PgСм*;
- 33 — 36 — инструкция K1804BN1;
- 37 — запрещение прерывания;
- 38, 39 — выбор длительности такта;
- 40 — разрешение бита *Z* K1804BP2;
- 41 — разрешение бита *C* K1804BP2;
- 42 — разрешение бита *N* K1804BP2;
- 43 — разрешение бита *OVR* K1804BP2;
- 44 — разрешение регистра микросостояния K1804BP2;
- 45 — разрешение регистра макросостояния K1804BP2;
- 46—51 — управление тестированием *PgC* K1804BM1 и K1804BP2;
- 52 — выбор источника управления тестированием;
- 53—56 — инструкция K1804BY4;
- 57—68 — адрес перехода;
- 69, 70 — управление интерфейсной логикой.

Рассмотрим варианты структурной организации процессора, ориентированного на выполнение системы команд, включающей двухадресные команды типа регистр — регистр и сложные сочетания кодов операций с методами адресации. Особенности такого процессора являются совмещенный конвейер команд/микрокоманд, позволяющий компенсировать неудобство одноадресной архитектуры БИС K1804BM1 для операций типа регистр — регистр, и двухфазная дешифрация команд (при сложных способах адресации) с отдельным представлением микропрограмм доступа к операндам и микропрограмм обработки операндов.

Будем считать, что время доступа к командам и операндам не является «узким местом» архитектуры микроЭВМ, т. е. фактором, сдерживающим увеличение производительности, и далее вопросы органи-

Таблица 5.1. Конвейер команд типа RR

Этап	Такт														Комментарии
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Формирование адреса команды	A	B				C			D			E			PC + 2 → PC PC + 2 → PгA
Выборка команды		PгK	Pг3				Pг3			Pг3			Pг3		PC + 2 → PгA, нагрузка Pг3 или PгK
Дешифрация		A	B			PгK	H		PгK	C		PгK	D		
Выполнение	+	+	-	-	+	+		H	H	C	C		D	D	Использование K1804BM1

Примечание. A, B, C, D — команды типа RR.

Таблица 5.2. Конвейер команд типа RX

Этап	Такт																Комментарии
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
Формирование адреса команды	A	A _d															
Выборка команды		RgK A	Rg3 A _d			B		B _d			C		C _d				PC+2→PгA, PC
Дешифрация									Rg3 B _d		RгK C		Rг3 C _d				Загрузка RгK, Rг3 и PC+2→PгA, PC при заполнении конвейера
Формирование адреса операнда						A		B				C					PC+2→PгA, PC; декодирование, загрузка RгМК
Выборка операнда									B					C			Rг3+X2→PгA
Выполнение										B							PC+2→PC, PгA загрузка операнда n
Примечание. A _d , B _d , C _d — смещения; A, B, C — команды	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	Использование K1804BM1

тип регистра — индексруемая память.

зации интерфейса с ОЗУ затрагивать не будем, сосредоточившись лишь на вопросах построения процессора. И еще одно предварительное замечание: БИС К1804ВМ1 не приспособлена для быстрого выполнения умножения, с этой операцией микросхема «справляется» хуже, чем БИС К1804ВС1 или К1804ВС2. Поэтому в применениях, связанных с частым выполнением команд умножения, наличие БИС умножителя 16×16 оправдывает связанные с ним дополнительные затраты, повышая производительность на порядок.

Проанализируем несколько типовых решений по структурной организации микропрограммного процессора на БИС К1804ВМ1. Анализ производительности будем проводить для последовательности команд регистр — регистр. Поскольку целью является определение структуры с максимальным быстродействием, то всеми цепями передачи данных будем управлять непосредственно с выхода РгМК, а не через промежуточные схемы. Так, для разрешения тристабильного выхода ДШНА будет использоваться специальный разряд микрокоманды, а не выход \overline{ME} схемы К1804ВУ4. Кроме того, для обеспечения максимальной производительности конвейера команд регистр — регистр в процессоре предполагается наличие отдельной от АЛУ схемы адресной обработки. Параллельно с операцией над содержимым регистров в микросхеме К1804ВМ1 схема адресной обработки формирует следующий адрес команды как «предыдущий + 2».

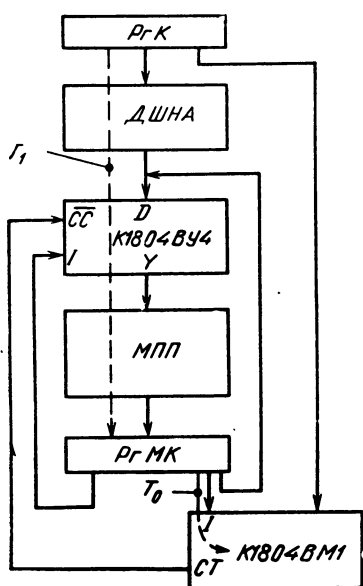


Рис. 5.7. Структурная организация процессора (1-й вариант)

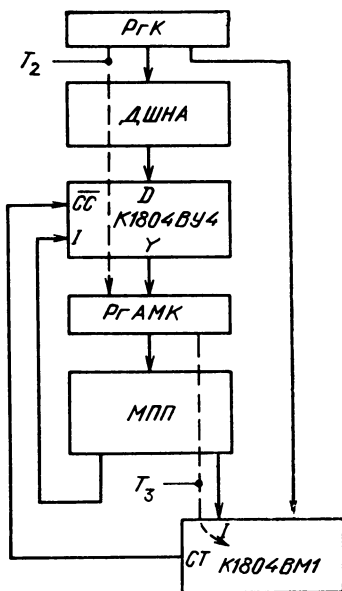


Рис. 5.8. Структурная организация процессора (2-й вариант)

Первая структура показана на рис. 5.7. Максимальная задержка T_0 операционной части не превышает 125 нс, а управляющей — $T_1 = t_{РГК} + t_{ДШНА} + t_{ВУ4} + t_{МПП} + t_{уст. РГМК} = 13 + 80 + 25 + 80 + 5 = 208$ нс при использовании схем К556РТ2 и К556РТ7 в качестве ДШНА и МПП соответственно. Длительность такта $T = \max \{T_0, T_1\} = 208$ нс. Если бы за один такт схема К1804ВМ1 могла производить операцию над двумя РОН, данное проектное решение обеспечивало бы производительность в конвейерном режиме почти 5 млн. команд регистр — регистр в секунду. Увы, микросхема К1804ВМ1 этим качеством (при всех ее прочих достоинствах) не обладает: здесь операция над двумя регистрами потребует двух тактов общей длительностью около 400 нс при постоянной длительности такта. Применение переменной длительности такта в данном случае не дает существенных улучшений. Итак, данное схемное решение обеспечивает производительность всего 2,5 млн. команд регистр — регистр в секунду, потенциальное быстродействие БИС К1804ВМ1 почти наполовину не используется.

Впрочем, такая же производительность может быть обеспечена и меньшими аппаратными затратами: вместо РГМК достаточно поставить РГАМК (рис. 5.8). Тогда задержка $T_2 = t_{РГАМК} + t_{МПП} +$

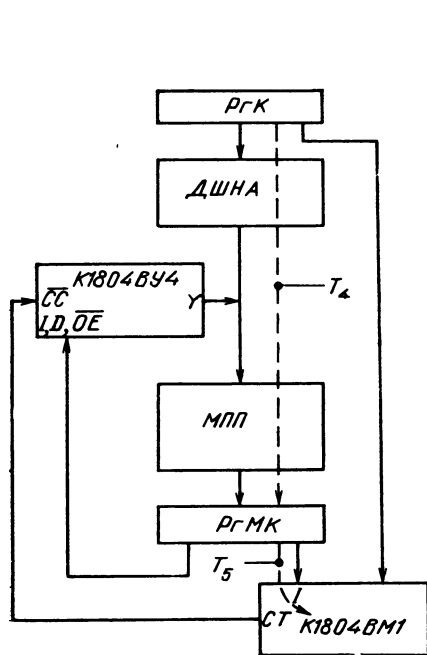


Рис. 5.9. Структурная организация процессора (3-й вариант)

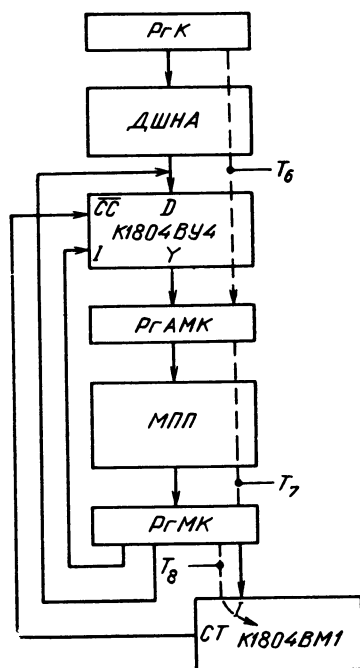


Рис. 5.10. Структурная организация процессора (4-й вариант)

$+ t_{BC1} = 18 + 80 + 100 = 198$ нс, а $T_3 = t_{PrK} + t_{ДШНА} + t_{ВУ4} + t_{уст. PrMK} = 18 + 80 + 25 + 5 = 128$ нс. Длительность такта $T = \max \{T_2, T_3\} = 198$ нс. Для выполнения команды регистр — регистр требуется два такта.

В третьей схеме (рис. 5.9) для сокращения времени дешифрации адрес с выхода ДШНА подается непосредственно на вход МПП. Длительность такта $T = \max \{T_4, T_5\} = 183$ нс. Операции регистр — регистр по-прежнему требуют двух тактов. Результирующая производительность 2,77 млн. команд/с.

Четвертая схема (рис. 5.10) содержит как РгАМК, так и РгМК. Трехуровневый конвейер микрокоманд позволяет сократить длительность такта до $T = \max \{T_6, T_7, T_8\} = 128$ нс, где $T_6 = 18 + 80 + 25 + 5 = 128$ нс, $T_7 = 18 + 80 + 5 = 103$ нс, $T_8 < 125$ нс. Таким образом обеспечивается производительность около 3,8 млн. команд регистр — регистр в секунду.

Наконец, последнее структурное решение обеспечивает производительность 4 млн. команд/с. Ознакомится с этим решением более подробно. Рассмотрим, что происходит в схеме рис. 5.11 на протяжении трех тактов (рис. 5.12), вырезанных из временной диаграммы конвейера команд регистр — регистр. Особенность РгАМК в этой схеме состоит в записи информации по уровню, а не по фронту синхросигнала, как остальные регистры.

Такт 1. В схеме К1804ВМ1 выполняется первая из двух микрокоманд (имеющая четный адрес), образуящую микропрограмму команды регистр — регистр: содержимое регистра R_a пересылается в аккумулятор (АК). Одновременно с этим из МПП производится выборка второй микрокоманды (со следующим, нечетным адресом), а из ОЗУ — очередной $(i + 2)$ -й команды, формируется адрес следующей $(i + 3)$ -й команды и начинается дешифрация $(i + 1)$ -й команды. Дешифрация команды включает: дешифляцию начального адреса, передачу его со входа D на выход Y схемы К1804ВУ4 и выборку начальной микрокоманды из МПП. Первые два действия не «укладываются» в 125 нс, поэтому в качестве РгАМК поставлен регистр, «прозрачный» на каждом четном такте и «закрытый» на нечетном.

Такт 2. По фронту сигнала в РгМК заносится, а затем в К1804ВМ1 выполняется вторая (завершающая) микрокоманда микропрограммы, интерпретирующей команду типа регистр — регистр: над аккумулятором и регистром R_b производится требуемая операция (\cdot), результат возвращается в R_b . По готовности правильного значения адреса A_{i+1} на выходе схемы К1804ВУ4 он через прозрачный буфер поступает на вход МПП, в конкатенации с «0» в младшем разряде образуя адрес микрокоманды, выборка которой начинается немедленно.

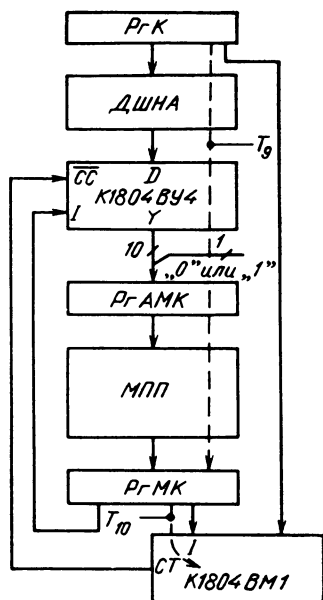


Рис. 5.11. Структурная организация процессора (5-й вариант)

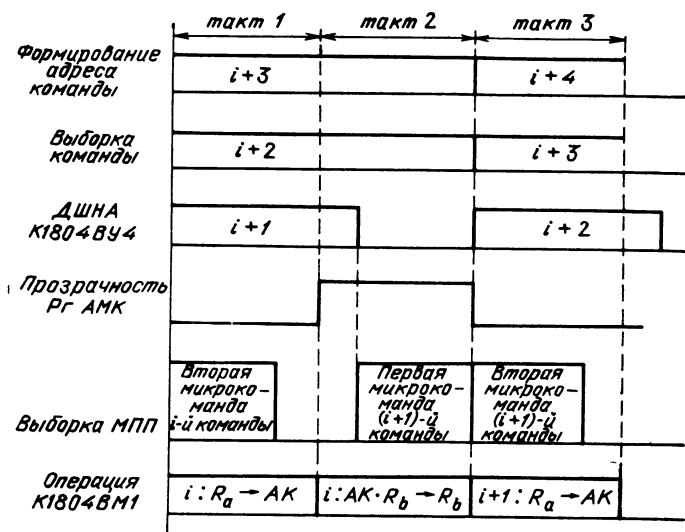


Рис. 5.12. Временные диаграммы конвейера команд и микрокоманд

Такт 3. В РгМК заносится и начинается выполняться первая микрокоманда следующей команды. Тем временем с закрытого буфера поступает прежнее значение A_{i+1} , которое (уже в конкатенации с «1») образует адрес микрокоманды, второй и последней в микропрограмме команды регистр — регистр. Начинается выборка из ОЗУ $(i+3)$ -й команды и формирование адреса $(i+4)$ -й. Итак, несмотря на то, что время распространения сигнала от РгК до РгАМК превышает 125 нс, на длительность такта это не влияет: «запаздывание» через прозрачный буфер «переносится» в начало следующего такта, где выборка МПП требует всего 80 нс, предустановка РгМК 5 нс. Суммарная длительность двух тактов 250 нс, итоговая производительность 4 млн. команд типа регистр — регистр в секунду.

И еще одна существенная деталь. Поскольку содержимое адресного поля микрокоманды (в отличие от операционного поля) изменяется не на каждом такте, а через один, то загружать эту часть РгМК можно в два раза реже, а на нечетных тактах адресное поле (около 20 разрядов) можно использовать по другому назначению, тем самым сокращая разрядность микрокоманды.

Типовая организация ДШНА на основе ПЗУ для преобразования кода операции команды в адрес микрокоманды, с которой начинается микропрограмма выполнения этой команды [4], отличается простотой, высоким быстродействием и небольшими аппаратными затратами. Возможна она благодаря тому, что формат команды разделен на два поля разрядностью 1 байт, причем содержимое только старшего байта отображается в адрес микрокоманды. Очевидно, при таком формате система команд может включать не более 256 сочетаний кодов операций с методами адресации. Как утверждают специалисты фирмы AMD[4] для мини- и микроЭВМ этого вполне достаточно. Иного мнения придерживались специалисты фирмы DEC, создавая систему команд ЭВМ PDP-11, которая нашла отражение и во многих отечественных маши-

нах. Разнообразие форматов команд предоставляет широкие возможности программисту, но разработчику высокопроизводительного эмулятора такой системы команд это разнообразие создает дополнительные проблемы. Рассмотрим их вкратце и обоснуем целесообразные проектные решения на примере системы команд *PDP-11*.

Если для каждого сочетания кода операции с методами адресации обоих операндов в рамках формата двухоперандных команд написать отдельную микропрограмму, то в качестве дешифратора начальных адресов пришлось бы использовать ПЗУ емкостью 1К слов. При этом и каждое слово имело бы разрядность около 12—13 бит, ибо ПЗУ микропрограмм содержит не только начальные микрокоманды («точки входа»), но и собственно микропрограммы. Для сокращения объема ПЗУ микропрограмм процедуры доступа к операндам при разных способах адресации можно оформить в виде микроподпрограмм. Тогда выполнение двухоперандной команды будет состоять из последовательных обращений к микроподпрограмме доступа к первому операнду, микроподпрограмме доступа ко второму операнду, микропрограмме выполнения собственно операции, микроподпрограмме засылки результата по адресу второго операнда. Если при этом не вводить отдельные микроподпрограммы для работы с целыми словами и байтами, для прямых и косвенных методов адресации, а также при условии, что обращение к программному счетчику ничем не отличается от обращения к другим РОН, то суммарный объем микропрограммной памяти можно свести к минимуму. Увы, производительность процессора тоже устремится к минимальной: организация конвейера команд станет неэффективной, многочисленные условные переходы в микропрограммах и передачи управления между микроподпрограммами замедлят обработку команд.

Следующее проектное решение лишено перечисленных недостатков (поясним на примере системы команд *DEC*). Все команды с учетом методов адресации разделим на три группы.

1. Однотактные. Это одно-и двухоперандные команды с регистравой адресацией, не содержащие обращения к *PC*, а также безоперандные однотактные команды. Примеры: *ADD RO, R1*; *CLR R2*; *SEC*.

2. Многотактные без доступа к операндам. Примеры: *BNE, RTN*; *MUL R1, R2*. К этой группе отнесем и «короткие» с обращением к *PC*.

3. С доступом к операндам. Это однооперандные с адресацией через память (пример: *CLR (R1) +*); двухоперандные с приемником в памяти (пример: *MOV # 10, (R4) +*); двухоперандные с источником в памяти и приемником в РОН (пример: *MOV # 10, R1*).

Всем однотактным командам и всем многотактным командам, выполняемым без доступа к операндам в памяти, поставим отдельные микропрограммы со своими точками входа (их множество обозначим *BX1*). Многотактные команды с доступом к операндам реализуются в виде двух сопрограмм: первая микропрограмма осуществляет доступ к операндам, вторая — «основную» операцию. Множества точек входа тех и других микропрограмм обозначим *BX2* и *BX3*. Причем для двух пер-

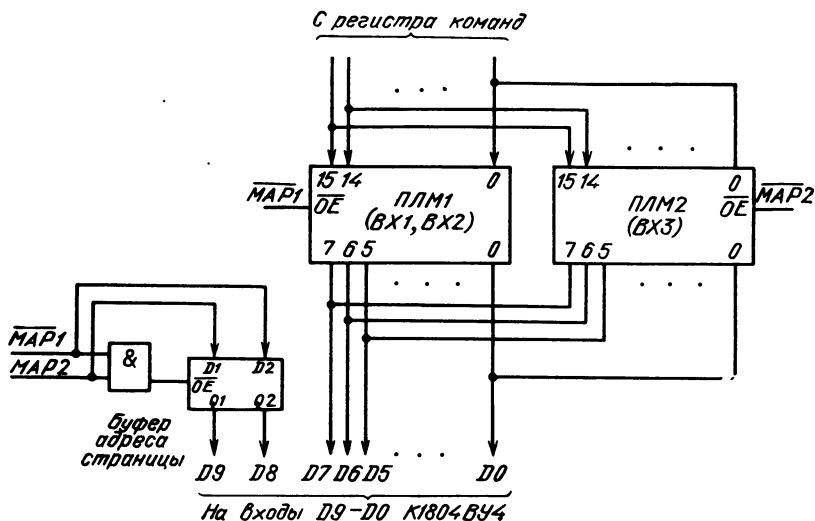


Рис. 5.13. Схема дешифрации команд на ПЛМ

вых подгрупп многотактных команд с доступом к операндам запись результата в память и восстановление конвейера команд производится в «основной» микропрограмме (с точкой входа из множества ВХ3). Для третьей подгруппы запись результата в память не нужна, а поддержание конвейерного режима почти целиком берет на себя микропрограмма доступа. «Основные» микропрограммы для слов/байт разделены, а микропрограммы доступа обобщены.

К выходу регистра команд подключаются (рис. 5.13) параллельно две программируемые логические матрицы (ПЛМ) с 16 входами и 8 трехстабильными выходами, которые поразрядно объединены и направлены на входы D7 — D0 схемы К1804ВУ4. Ко входам D9, D8 схемы К1804ВУ4 подключен тристабильный буферный формирователь, который в открытом состоянии передает код 01 или 10, а в закрытом обеспечивает возможность подачи на вход D схемы К1804ВУ4 адреса из другого источника (поля микрокоманды, вектора прерывания). В рамках цикла выполнения одной команды чтение может производиться как из одной, так и последовательно из обеих ПЛМ. Первая ПЛМ содержит точки входа, т. е. начальные адреса микропрограмм ВХ1 (однотактных и многотактных команд без доступа к операндам) и ВХ2 (доступа к операндам). Вторая ПЛМ — адреса ВХ3 (основных микропрограмм). Дешифрация любой команды начинается с чтения первой ПЛМ (сигнал разрешения $\overline{MAP1} = 0$ активен, $\overline{MAP2} = 1$ пассивен) и передачи значения адреса в конкатенции с $\overline{MAP1}$, $\overline{MAP2}$ на вход D микросхемы К1804ВУ4 и далее с ее выхода Y на РгАМК в «прозрачном» его состоянии (см. выше). Если дешифрируемая команда является однотактной, то на следующем такте, когда начальная мик-

рокоманда попадает в регистр микрокоманд, будет подан сигнал разрешения $\overline{MAP1}=0$. Если команда окажется многотактной, то сигнал разрешения $\overline{MAP1}=0$ будет подан на последнем такте ее микропрограммы. Наконец, если дешифрируемый код с регистра команд будет в первой ПЛМ опознан как «многотактная команда с доступом к операндам», то на выходе этой ПЛМ появится начальный адрес микропрограммы доступа. Последней микрокомандой этой микропрограммы будет разрешено чтение из второй ПЛМ (активный сигнал $\overline{MAP2}=0$), откуда на вход D микросхемы K1804ВУ4 поступит адрес «основной» микропрограммы, производящей требуемую операцию (в соответствии с кодом в регистре команд) над выбранными ранее операндами.

Каждая «основная» микропрограмма может быть представлена в четырех вариантах: с записью результата в ОЗУ и без записи (если второй операнд — РОН), с участием РС и без него. Переход на тот или иной вариант зависит от микропрограммы доступа к операндам: если второй операнд в ОЗУ, то и результат надо занести в ОЗУ по тому же адресу. Целесообразное общее число точек входа микропрограммы ограничено сверху характеристиками серийных ПЛМ. Во всех случаях переход к микропрограммам осуществляется через таблицы ветвлений: по одной таблице на каждую ПЛМ. Начальные адреса таблиц задаются разрядами $D9, D8$ (см. рис. 5.13).

Переход к обработке прерывания производится по окончании выполнения текущей команды, т. е. перед дешифрацией следующей команды. Чтобы не расходовать время на условный переход к тестированию признака, индицирующего наличие требований прерывания, удобно принять следующее схемное решение. В такте дешифрации очередной команды из поля микрокоманды подается активный уровень сигнала «разрешение прерывания». Если требований прерывания нет, то дешифрация происходит нормально, как было рассмотрено выше. Если есть, то поступление адреса с выхода ДШНА блокируется путем принудительной установки сигналов $\overline{MAP1} = \overline{MAP2} = 1$. Одновременно с этим открывается тристабильный буфер вектора прерывания, передавая на вход D микросхемы K1804ВУ4 адрес, являющийся конкатенацией трехразрядного кода-вектора с постоянным базовым адресом (например, 1111111). В восьми ячейках (с 1111111000 по 1111111111) располагается таблица ветвлений, т. е. начальные микрокоманды обработки прерываний каждого типа с безусловным переходом на продолжения этих микропрограмм в произвольную область микропрограммной памяти. При необходимости число точек входа в микропрограммы прерывания (т. е. число уровней прерывания) может быть увеличено или уменьшено. Таким образом на такте дешифрации команды вместо перехода к микропрограмме интерпретации команды будет выполнен переход к микропрограмме обработки прерывания.

И еще несколько общих замечаний о практических схемах дешифрации команд и обработки прерываний в процессорах команд.

Блок ДШНА не обязательно строить только на ПЗУ либо только на ПЛМ: иногда целесообразно их совместное использование. Например, сильнокодированное поле команды экономнее дешифровать на ПЗУ, а слабокодированное — на ПЛМ. При выборе элементов для реализации ДШНА следует иметь в виду и их быстродействие: ПЗУ и ПЛМ меньшей емкости являются, как правило, более быстродействующими, например: время выборки у микросхем КР556РТ11 (256×4 , 45 нс) меньше, чем у КР556РТ17 (512×8 , 50 нс), КР556РТ13 ($1\text{К} \times 4$, 60 нс), КР556РТ15 ($2\text{К} \times 4$, 60 нс), КР556РТ18 ($2\text{К} \times 8$, 60 нс). Разрядность адреса микрокоманды составляет, как правило, от 9 до 12 бит, поэтому для реализации ДШНА требуются две-три микросхемы.

Обработка прерываний в процессорах команд может осуществляться на микропрограммном или программном уровне. В первом случае процедура обработки прерываний реализована в микрокоде и переход к ней может выполняться не обязательно по окончании команды, но и «в середине» ее. Это сокращает время реакции на запрос, но лишь в случаях, когда объем сохраняемой и восстанавливаемой информации о состоянии процессора мал.

Прерывание на программном уровне предполагает реализацию процедур на уровне системы команд процессора и хранение соответствующих подпрограмм во внешнем по отношению к процессору ПЗУ. Для адресации последнего в программный счетчик процессора должен быть загружен адрес подпрограммы, соответствующей запросу с максимальным приоритетом. Такая загрузка осуществляется служебной микропрограммой обработки прерываний (общей для всех запросов), переход к которой выполняется на этапе дешифрации очередной команды во время действия инструкции *JMAP*. Выход \overline{TR} микросхемы К1804ВН1 и выход \overline{ME} микросхемы К1804ВУ4 (через инвертор) подаются на элемент И — НЕ, установленный на входе \overline{OE} ДШНА. Таким образом, при $\overline{ME} = 1$ выход ДШНА находится в состоянии высокого сопротивления независимо от значения сигнала \overline{TR} . При $\overline{ME} = 0$, $\overline{TR} = 1$ выполняется переход к начальным адресам микропрограмм с ДШНА, а при $\overline{ME} = 0$, $\overline{TR} = 0$ — переход к адресу 1111111111. По данному адресу начинается служебная микропрограмма, выявляющая источник запроса прерывания, сохраняющая состояние процессора, загружающая программный счетчик адресом соответствующей подпрограммы обработки прерывания и передающая ей управление.

5.3. ФУНКЦИОНАЛЬНЫЕ РАСШИРИТЕЛИ

В настоящее время для расширения возможностей микроЭВМ в их состав включаются процессоры операций с плавающей точкой (ППТ). Универсальные ППТ (ППТ общего назначения) имеют широкий набор функций и умеренную производительность. Специализированные ППТ обеспечивают высокую производительность при выполнении только ключевых арифметических операций, таких как сложение и умно-

Таблица 5.3. Время выполнения команд, мкс

Формат. функция	Тип процессора			
	Intel 8087	Am 9511	Intel 8231A	ФР
Фиксированная точка 32 разряда				
<i>ADD</i>	2	10	5	1
<i>SUB</i>	2	19	10	1
<i>MUL</i>	25	105	53	15
<i>DIV</i>	45	104	52	18
Плавающая точка, 32 разряда				
<i>ADD</i>	20	184	92	27
<i>SUB</i>	20	185	93	27
<i>MUL</i>	29	84	42	18
<i>X·LOG₂Y</i>	220	—	—	—
<i>DIV</i>	39	92	46	18
<i>COMP</i>	9	—	—	—
<i>SQRT</i>	36	400	200	60
<i>SIN</i>	—	2232	1116	80
<i>COS</i>	—	2059	1030	80
<i>TAN</i>	108	2877	1439	180
<i>ASIN</i>	—	3834	1917	211
<i>ACOS</i>	—	3867	1934	240
<i>ATAN</i>	160	3003	1502	84
<i>LG</i>	—	3666	1783	93
<i>LN</i>	—	3478	1739	74
<i>EXP</i>	130	2438	1219	85
<i>ELT→FIX</i>	18	173	87	15
<i>FIX→FLT</i>	12	189	95	23

жение. К ППТ общего назначения относятся БИС Intel 8087, Am 9511, Intel 8231A [25, часть характеристик, 26] которых приведена в табл. 5.3. Подобные микросхемы ориентированы на совместную работу с определенными ведущими процессорами (например, Intel 8087 соединяется только с микропроцессором Intel 8086, исполняющим роль ведущего), поэтому не могут быть применены в вычислительных машинах, центральные процессоры которых не соответствуют их архитектурной ориентации.

Ниже рассматриваются схемно-микропрограммные решения, принятые при проектировании функционального расширителя (ФР) с богатым набором команд и высоким быстродействием, предназначенного для форсирования производительности микроЭВМ СМ-1800 и совместимых с ней персональных компьютеров путем аппаратного вычисления арифметических и трансцендентных функций, программная реализация которых оказывается слишком медленной для конкретных приложений. Если желательно подключение ФР к другой ЭВМ с интерфейсом, отличным от И-41, то следует в интерфейс расширителя (аппаратуру и микропрограмму) внести изменения и написать программу-драйвер в заданной системе команд. Если же необходимо еще более ориенти-

ровать систему команд ФР на определенные задачи, квалифицированный пользователь имеет возможность изменить ее (частично или полностью), для чего понадобится разработать собственное микропрограммное обеспечение. Эти и другие качества ФР являются хорошей предпосылкой для практического применения подобного устройства в качестве основы для решения широкого класса задач на микроЭВМ невысокой производительности.

Функциональный расширитель реализован на одной стандартной плате в конструктиве Е2 и подключается в качестве периферийного устройства к базовой микроЭВМ через системный интерфейс И-41. Прежде чем перейти к подробному рассмотрению архитектуры ФР, вкратце опишем его работу в составе микроЭВМ.

Для выполнения конкретной функции 32-разрядные операнды побайтно должны быть предварительно загружены из микроЭВМ в стек расширителя: на это будет затрачено время t_o , определяемое пересылками байтов из памяти машины в порт данных. После загрузки операндов можно подавать код команды, который поступает в соответствующий порт расширителя. На это потребуется время t_k . Выполнение команды займет время t_v , после чего из портов ФР можно получить байт состояния с признаками результата (время t_c) и, если необходимо, прочитать сам результат побайтно за время t_p в память машины. Таким образом, в общем случае суммарное время выполнения отдельно взятой команды включает следующие составляющие: $T = t_o + t_k + t_v + t_c + t_p$. В данном случае сумма $t_o + t_k + t_c + t_p$ составляет десятки микросекунд, поэтому есть разумный предел в стремлении уменьшить значение составляющей t_v за счет усложнения аппаратуры функционального расширителя.

Рассматриваемый вариант ФР, реализованный на основе биполярных микропроцессорных БИС серии К1804, обеспечивает выполнение арифметических и трансцендентных функций над 32-разрядными числами за время t_v , не превышающее 0,41—264 мкс, в зависимости от типа функции.

Взаимодействие с базовой микроЭВМ. Чтобы использовать ФР, необходимо предусмотреть способ и средства для организации его взаимодействия с микроЭВМ. Пересылка данных в расширитель, подключенный к микроЭВМ СМ-1800 через стандартный интерфейс И-41 в качестве внешнего устройства, и чтение результата из него осуществляются посредством программируемого ввода-вывода с помощью команд $OUT <\text{адрес порта}>$ и $IN <\text{адрес порта}>$. На плате ФР имеются следующие три порта.

1. Порт исходных операндов и результата с адресом FD . Расширитель характеризуется стек-ориентированной организацией, поэтому перед исполнением команды операнд(ы) должен заноситься в верхушку восьмиуровневого стека, условное изображение которого показано на рис. 5.14. При записи 32-разрядного операнда в верхушку стека необходимо четырежды обратиться к этому порту для загрузки четырех байтов операнда. При чтении 32-разрядного результата из верхушки

стека необходимо четыре раза обратиться к данному порту для выгрузки четырех байтов результата. Запись сопровождается проталкиванием предыдущего содержимого стека на один уровень (стековая операция *PUSH*), а чтение — выталкиванием содержимого стека (стековая операция *POP*).

2. Порт команд с адресом *FE*. Этот порт доступен только для записи: на ФР подается байт команды, вызывающей определенные действия над операндами, находящимися в стеке расширителя. Выполнение команды в ФР начинается приемом байта команды и завершается установкой слова состояния, отражающего готовность результата и признаки (флажки) результата.

3. Порт слова состояния. Этот порт имеет адрес *FF* и доступен только для чтения. Информация в слове состояния фиксируется после выполнения очередной команды в ФР. Ее необходимо анализировать в базовой ЭВМ перед обращениями к портам *FD* и *FE*, чтобы констатировать факт готовности расширителя (т. е. окончания выполнения предыдущей команды) и диагностировать особые вычислительные ситуации. Чтение слова состояния из порта *FF* производится асинхронно по отношению к процессам в ФР и не влияет на выполнение команды последним.

Формат слова состояния (номера разрядов и мнемонические обозначения):

7	6	5	4	3	2	1	0
<i>RDY</i>	\overline{ERR}	<i>DIAGN</i>	\overline{OVR}	\overline{UND}	\overline{N}	\overline{Z}	

Прочитав слово состояния, процессор микроЭВМ проверяет 7-й разряд слова состояния (*RDY*). При равенстве данного разряда нулю необходимо снова прочитать информацию о состоянии расширителя, так как прежде ФР был занят выполнением команды. Если при очередной проверке значение в этом разряде окажется равным «1», то вычисления в ФР завершены. В таком случае расширитель способен реагировать на обращения к портам *FE* и *FD* для получения команды или записи/чтения верхушки стека. Однако при выполнении большинства команд могут возникать особые (аварийные) ситуации, о которых рас-

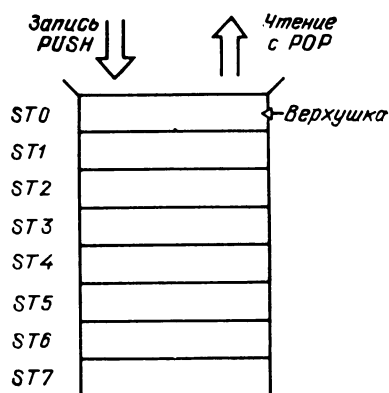


Рис. 5.14. Условное изображение стека функционального расширителя

ширитель должен сообщить центральному процессору, ибо последний управляет ходом вычислений и принимает те или иные решения в зависимости от получаемых результатов. Поэтому при «нормальном» завершении команды в 6-м разряде слова состояния (\overline{ERR}) будет «1». Если же, зафиксировав готовность, центральный процессор обнаружит в этом разряде «0», он должен определить тип особой ситуации и выполнить действия в соответствии с программой, как это предусмотрено пользователем.

Тип особой ситуации однозначно определяется разрядами 2—5 слова состояния следующим образом.

1. Разряды 5 и 4 ($DIAGN$) характеризуют результат проверки в ФР значений исходных данных на допустимость выполнения соответствующей команды:

$DIAGN = 00$ — аргумент слишком велик,

$DIAGN = 01$ — делитель равен нулю,

$DIAGN = 10$ — аргумент отрицательный.

Если аргумент отрицательный или слишком велик, то вместо результата в верхушку стека записывается сам аргумент и вычисления функции не происходит. Если делитель равен нулю, в верхушку стека вместо результата заносится делимое. Аргументы были в порядке при наличии в указанном поле комбинации «11».

2. Разряд 3 (\overline{OVR}) устанавливается в «0» при переполнении в операциях с фиксированной точкой и переполнении порядка в операциях с плавающей точкой. Во втором случае в верхушку стека будет занесен результат с правильным знаком, правильной мантиссой и порядком, уменьшенным на 128 по отношению к реальному.

3. Разряд 2 (\overline{UND}) равен 0, если в ходе вычислений произошло антипереполнение порядка числа с плавающей точкой. В этой ситуации в верхушку стека заносится число с правильным знаком, правильной мантиссой и порядком, увеличенным на 128 по отношению к реальному.

Младшие два разряда (\overline{N} и \overline{Z}) слова состояния характеризуют знак (число отрицательное при $\overline{N} = 0$) и нулевое значение (число равно нулю при $\overline{Z} = 0$) числа, находящегося в верхушке стека. Таким образом, «нормальное» завершение выполнения команды отражает единичное значение 6-го и 7-го разрядов слова состояния, а значения остальных разрядов дают более полное представление о типе особой ситуации или полученном результате.

Функциональные возможности и форматы данных. Функциональный расширитель выполняет операции хранения, пересылки и преобразования 32-разрядных операндов, обрабатывая при этом два типа данных: с фиксированной точкой (целые числа со знаком в дополнительном коде) и с плавающей точкой. В формате с фиксированной точкой могут быть представлены целые числа в диапазоне от $+ (2^{31}-1)$ до -2^{31} . В этом случае точка фиксирована справа от младшего (нуле-

вого) разряда. При представлении чисел с плавающей точкой старший (левый) разряд отведен под знак числа (т. е. знак мантиссы), разряды 30—24 используются для представления порядка в дополнительном коде, а разряды 23 — 0 образуют нормализованную мантиссу в прямом коде. Таким образом, диапазон значений нормализованной мантиссы простирается от 0,5 до $1-2^{-24}$, а значений порядка от -64 до $+63$.

Формат команды — однобайтный. Шесть младших разрядов представляют код функции, которую должен выполнить ФР, а разряды 6 и 7 в байте команды должны иметь нулевое значение (в противном случае произойдет останов ФР по загрузке несуществующей команды). По смыслу команды ФР могут быть сгруппированы следующим образом:

1. Функциональные, заключающиеся в обработке данных путем определенных вычислений. К таковым относятся: арифметические операции (сложение, вычитание, умножение, деление); прямые тригонометрические функции (синус, косинус, тангенс, котангенс); обратные тригонометрические функции (арксинус, арккосинус, арктангенс); гиперболические функции; корень квадратный, возведение в степень; логарифмы натуральный и десятичный; экспонента; обратная величина; «бабочка» — основная операция быстрого преобразования Фурье и другие.

2. Команды преобразования формы представления чисел (число с плавающей точкой в число с фиксированной точкой и наоборот).

3. Команды манипуляции операндами: циклический сдвиг содержимого стека на заданное число уровней и копирование одного из ранее занесенных в стек операндов в его верхушку с одновременным проталкиванием (операция *PUSH*) остальных операндов.

4. Команды диагностики.

Функциональные команды могут быть однооперандными и двухоперандными. В первом случае исходный операнд находится в верхушке стека *ST0*, куда после выполнения команды записывается результат. Во втором случае исходные операнды занимают уровни *ST0* и *ST1* (для операции деления с удвоенной точностью также занят уровень *ST2*), а результат помещается в верхушку *ST0* с одновременным уменьшением числа занятых уровней стека (это не относится к операции умножения с удвоенной точностью, для которой результат занимает те же два уровня), т. е. имеет место стековая операция *POP* — выталкивание. Операнд(ы) должен быть представлен в формате, требуемом для выполнения данной команды. Обязанность соблюдения указанного соответствия ложится на программиста, так как аппаратными средствами ФР нельзя обнаружить несоответствие между форматом данных и типом команды.

Выполнение команд манипуляции операндами поясняет рис. 5.15. Применение команд этой группы оказывается полезным в тех случаях, когда в вычисляемом выражении есть повторяющиеся операнды. Например, в выражении $Y = \ln(A \cdot B - C) + A/C$ операнды *A* и *C* используются дважды. Следовательно, возможно копирование ранее

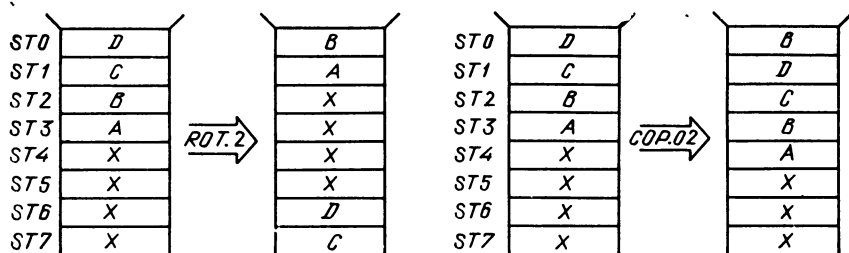


Рис. 5.15. Примеры манипуляции стек-операндами

загруженного операнда в верхушку стека, которое выполняется более чем на порядок быстрее повторной программной загрузки этого же операнда побайтно из микроЭВМ, а это означает сокращение общего времени вычисления таких формульных зависимостей. Особенно ощутим выигрыш во времени при вычислении циклических конструкций, к которым относится, например, вычисление суммы членов ряда. Поэтому при программировании прикладных задач рекомендуется в максимальной степени использовать ресурсы стека ФР, применяя команды манипуляции операндами.

Схемная реализация. Изложение вопросов аппаратной реализации ФР начнем с рассмотрения структурной схемы, которая изображена на рис. 5.16. Структурная схема ФР включает следующие основные

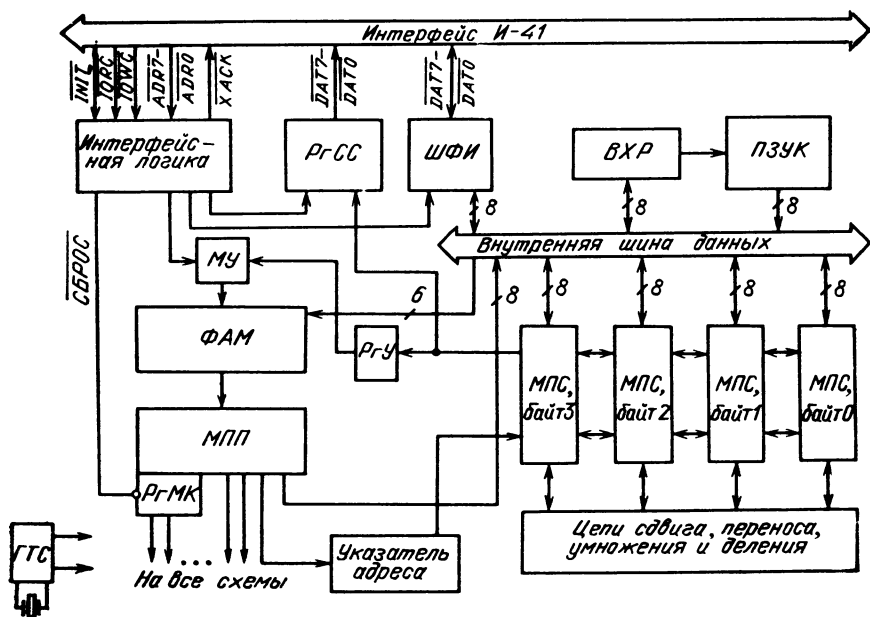


Рис. 5.16. Структурная схема функционального расширителя

элементы: ФАМ, МПП, РгМК, мультиплексор условий (МУ); регистр условий (РгУ); восемь микропроцессорных секций (МПС); схемы организации сдвигов, ускоренного переноса, вспомогательные цепи для умножения и деления; указатель адреса по каналам *A* и *B* микропроцессорных секций; регистр временного хранения (ВХР), он же — регистр адреса ПЗУ коэффициентов; ПЗУ коэффициентов интерполяции (ПЗУК); шинные формирователи с инверсией (ШФИ); регистр слова состояния (РгСС); интерфейсную логику для дешифрации адресов, анализа и формирования управляющих сигналов; внутреннюю 8-разрядную общую шину данных; генератор тактовых синхропоследовательностей (ГТС).

В структурной схеме ФР можно выделить три основных блока, включающих перечисленные элементы: БМУ, БОД и интерфейсный блок ФР. Рассмотрим подробнее состав и функционирование каждого из перечисленных блоков.

Функциональный расширитель является микропрограммируемым устройством, выполняющим все операции в соответствии с хранимыми микропрограммами. Блок микропрограммного управления реализован по принципу горизонтального микропрограммирования с шириной микрокомандного слова 55 разрядов. Разряды логически сгруппированы в поля, управляющие работой соответствующих схем. С одной стороны, предварительная оценка в ходе проектирования ФР показала, что для реализации системы команд ФР достаточно 1024 55-разрядных микрокоманд. С другой стороны, на плате заданного формата *E2* оказалось недостаточно места для построения схемы расширителя, содержащей семь корпусов микросхем микропрограммной памяти и десять корпусов доступных 6-разрядных регистров для построения РгМК, что обеспечило бы максимальное быстродействие ФР. Поскольку из системных соображений некоторое снижение потенциального быстродействия ФР не является критическим, было принято решение по организации микропрограммного управления, иллюстрируемое рис. 5.17.

Каждая 55-разрядная микрокоманда хранится в двух соседних ячейках микропрограммной памяти (МПП): 23 разряда в первой (четной) ячейке и 32 разряда во второй (нечетной). Реализована МПП на микросхемах *M556PT7* с организацией $2K \times 8$ бит и тремя состояниями на выходе. На старшие десять разрядов адреса микропрограммной памяти подается адрес, который образуется в формирователе адресов микрокоманд и фиксируется в регистре адреса микрокоманды (РгАМК) по фронту синхримпulses *CLKS*, поступающего с генератора тактовых синхропоследовательностей (рис. 5.18). Последний выполнен на основе счетчика *K531IE17П* и вырабатывает синхропоследовательности *CLKS* и *CLKM* с требуемой скважностью и сдвигом фаз. Синхросигнал *CLKM* поступает на младший разряд адреса МПП, обеспечивая выборку содержимого двух соседних ячеек МПП, а РгМК хранит в течение каждого цикла часть микрокоманды, извлекаемую из четной ячейки. Вторая половина микрокоманды подается на управляемые

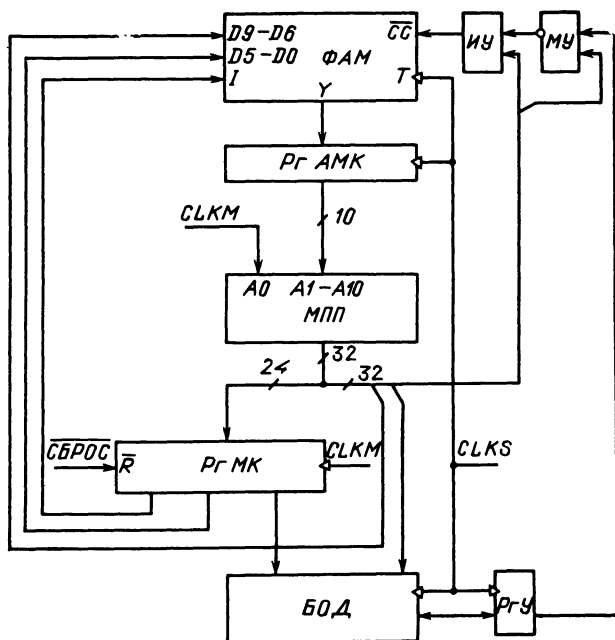


Рис. 5.17. Структурная схема блока микропрограммного управления

схемы непосредственно с выхода МПП. Таким образом, на все схемы расширителя поступают управляющие сигналы с РгМК и МПП.

Формирование адреса следующей микрокоманды осуществляется в ФАМ, реализованном на основе БИС К1804ВУ4. Очередной адрес Y определяется информацией на входах непосредственного адреса D кода инструкции I и сигнала условия $\overline{CС}$. Для организации условных переходов на вход $\overline{CС}$ подается требуемое условие с мультиплексора условий (МУ) через управляемый инвертор условия (ИУ), который при-

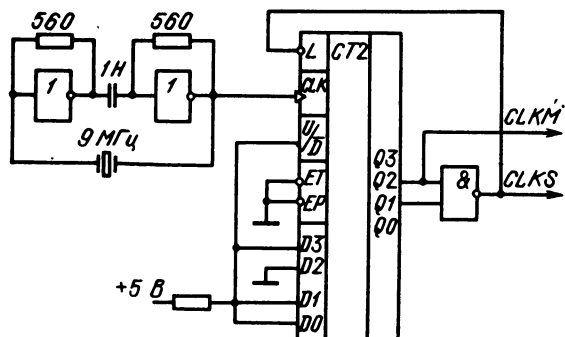


Рис. 5.18. Схема генератора тактовых синхронных последовательностей

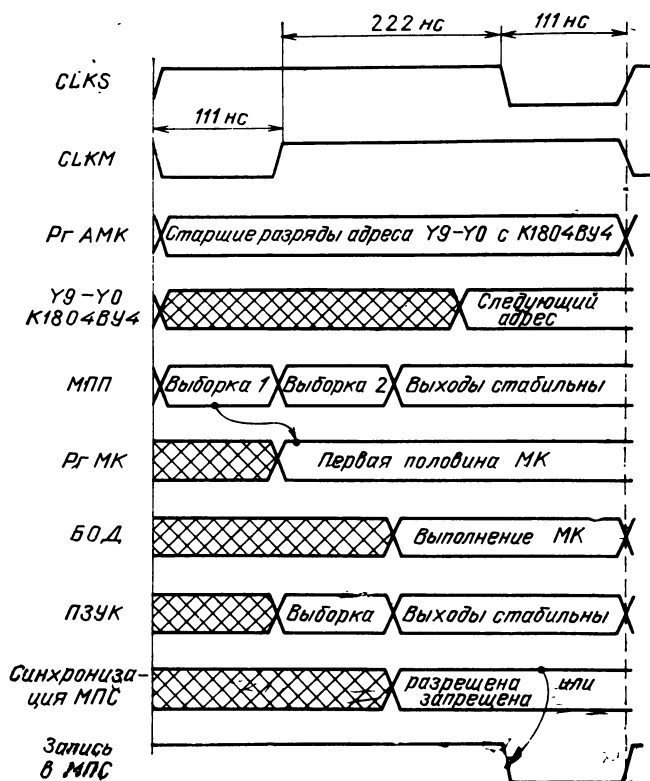


Рис. 5.19. Временные диаграммы цикла выполнения микрокоманды

дает гибкость схеме формирования следующего адреса с точки зрения размещения микрокоманд и сокращения их числа. В качестве сигналов условия используются признаки результата операции, выполненной в блоке обработки данных, а также управляющие сигналы, вырабатываемые интерфейсной логикой.

На рис. 5.19 изображены временные диаграммы, поясняющие работу БМУ и БОД. Микрокомандный цикл начинается с фронта тактового сигнала *CLKS*, по которому в *RgAMK* заносятся десять старших разрядов адреса микрокоманды с выхода *Y* БИС *K1804BY4*. Так как сигнал *CLKM* принимает значение 0, из *МПП* извлекается содержимое четной ячейки, которое загружается в 23-разрядный *RgMK* по фронту сигнала *CLKM*. Затем производится выборка второй части микрокоманды, когда сигнал *CLKM* принимает значение «1». Одновременно (если это предписано первой половиной микрокоманды) из *ПЗУ* коэффициентов интерполяции считывается байт коэффициента или константы для использования в операционном блоке. После установки на выходе *МПП* второй половины микрокоманды в *БОД* выпол-

няется «целая» микрокоманда, а в ФАМ производится вычисление адреса следующей. Если в данной микрокоманде определена запись результата в регистровое запоминающее устройство МПС, то она происходит по сигналу разрешения синхронизации одного или нескольких байтов МПС при низком уровне тактирующего импульса $CLKS$. Приход фронта сигнала $CLKS$ означает начало нового цикла, при этом в регистре условий фиксируются признаки, полученные во время выполнения предыдущей микрокоманды. Следовательно, условные переходы в микропрограммах реализуются за два цикла, а при безусловных переходах обеспечивается конвейерная обработка микрокоманд, т. е. вычисление адреса следующей микрокоманды совмещается во времени с выполнением текущей микрокоманды в БОД. Длительность микрокомандного цикла $T \approx 2t_{МПП} + \max \{t_{БОД}, t_{ФАМ}\}$.

Итак, микрокоманда считывается из МПП в два этапа, причем выборка второй части микрокоманды производится в то время, когда первая часть уже занесена в РгМК. Поэтому для минимизации длительности цикла управляющие сигналы с наиболее критичными задержками распространения должны подаваться из РгМК, а с менее критичными — непосредственно с выхода МПП. Такое распределение разрядов микрокоманды наряду с ее двойной выборкой из памяти существенно усложняет процесс микропрограммирования, если оперировать с физическими номерами разрядов микрокоманды. Поэтому микропрограммы удобнее составлять с использованием «концептуального» формата микрокоманды, не вникая в особенности физического распределения разрядов и двойной выборки из МПП, а затем учесть эти особенности. Ниже описан концептуальный формат микрокоманды: номера разрядов, назначение и активный уровень сигнала для одноразрядных полей (в скобках):

- 1 — разрешение синхронизации младшего байта МПС (1);
- 2 — разрешение синхронизации второго байта МПС (1);
- 3 — разрешение синхронизации третьего байта МПС (1);
- 4 — разрешение синхронизации старшего байта МПС (1);
- 5 — разрешение записи в РгСС (0);
- 6 — разрешение записи в ВХР (1);
- 7 — разрешение передачи данных через ШФИ с/на системную магистраль И-41 (1);
- 8 — разрешение изменения значения указателя стека \overline{ENSP} (0);
- 9 — направление изменения значения указателя стека $UP/DOWN$ (1/0);
- 10—12 — линии $A0 - A2$ адреса регистров МПС по каналу A на чтение;
- 13 — источник адреса по каналу $A \overline{MK}/SP$ (0/1);
- 14—16 — линии $B0 - B2$ адреса регистров МПС по каналу B на чтение и запись;
- 17 — источник адреса по каналу $B \overline{MK}/SP$ (0/1);
- 18,19 — управление мультиплексорами сдвига;

- 20, 21 — управление мультиплексором входного переноса и линии $I3$ МПС;
- 22 — управление мультиплексором линии $I1$ МПС $MK/\overline{PQ0}$;
- 23 — значение $PR31/PR0$, $PQ0$ при сдвигах вправо/влево (1/0);
- 24—32 — линии $I0$ — $I8$ управления МПС К1804BC1;
- 33 — 35 — выбор источника данных для внутренней общей шины;
- 36 — линия $A0$ адреса ПЗУ коэффициентов и значение разряда 2 слова состояния (CC);
- 37 — линии $D0$ К1804BY4, $A1$ ПЗУК, разряд 3 CC;
- 38 — линии $D1$ К1804BY4, $A6$ ПЗУК, разряд 4 CC;
- 39 — линии $D2$ К1804BY4, $A7$ ПЗУК, разряд 5 CC, разряд $\overline{C0}$ константы;
- 40 — линии $D3$ К1804BY4, $A8$ ПЗУК, разряд 6 CC, разряд $\overline{C1}$ константы;
- 41 — линии $D4$ К1804BY4, $A9$ ПЗУК, разряд 7 CC, разряд $\overline{C2}$ константы;
- 42 — линии $D5$ К1804BY4, $A10$ ПЗУК, разряд $\overline{C3}$ константы;
- 43—46 — линии $D6$ — $D9$ К1804BY4, разряды $\overline{C4}$ — $\overline{C7}$ константы;
- 47 — 50 — управление мультиплексором условий;
- 51 — управление полярностью условия на входе \overline{CS} БИС К1804BY4; прямая/инверсная (0/1);
- 52—55 — линии $I0$ — $I3$ кода инструкции К1804BY4.

Обратите внимание на совмещение нескольких полей в разрядах 36—46 формата, которое позволило сократить ширину микрокода. Очевидно, при выполнении каждой микрокоманды может использоваться по назначению не более чем одно из полей, совмещенных в разрядах 36—48 формата. Микропрограмма, написанная с использованием концептуального формата, преобразуется затем в «физическое» представление с помощью программ отладочного комплекса.

Блок обработки данных. Структурная схема 32-разрядного блока обработки данных, включающего восемь БИС К1804BC1, представлена на рис. 5.20. Здесь изображены также вспомогательные узлы для организации ускоренного переноса и различных типов сдвигов, режимов умножения и деления; регистр временного хранения; ПЗУ коэффициентов интерполяции; восьмиразрядная внутренняя общая шина данных. На структурной схеме не показаны некоторые входные и выходные сигналы микропроцессорных секций, дешифратор источника данных на внутреннюю шину и блок указателя стека (коммутатора адреса регистрового запоминающего устройства МПС по каналам A и B). Последний будет рассмотрен ниже.

Данные обрабатываются в БОД согласно поступающим из устройства микропрограммного управления микрокомандам. Подлежащие обработке операнды могут находиться в ПЗУ микропроцессорных секций или подаваться на их входы D извне. Операции, выполняемые в МПС, однозначно определяются сигналами, поступающими на входы

инструкции 10—18. В каждом цикле может иметь место обмен данными между различными элементами ФР, осуществляемый побайтно через внутреннюю общую шину. Информацию на шину может передавать один из следующих восьми источников: младший (нулевой) байт МПС, первый байт МПС, второй байт МПС, старший (третий) байт МПС, системный интерфейс И-41, константа из соответствующего поля микрокоманды, регистр ВХР и ПЗУ коэффициентов интерполяции. Выбор требуемого источника данных производится дешифрацией разрядов 33—35 концептуального формата микрокоманды, определяющих конкретный источник. С этой целью используется дешифратор источника на микросхеме К155ИД4. Подключение того или иного источника к внутренней шине реализуется подачей «0» с дешифратора на управляющие входы выбора кристалла (\overline{CS}) либо с разрешения выхода (\overline{OE}) соответствующих микросхем источников. Если источник не выбран, его выходы находятся в состоянии высокого сопротивления. Принимать информацию с внутренней шины могут одновременно несколько элементов. Получение информации приемниками разрешается разрядами 1—6 микрокоманды и выходом сигнала выборки векторного адреса \overline{VE} БИС К1804ВУ4, разрешающим запись адреса/данных во внутренний регистр адреса/счетчик данной микросхемы.

Передача информации между байтами МПС выполняется за два цикла с промежуточной записью в регистр ВХР, так как на все микропро-

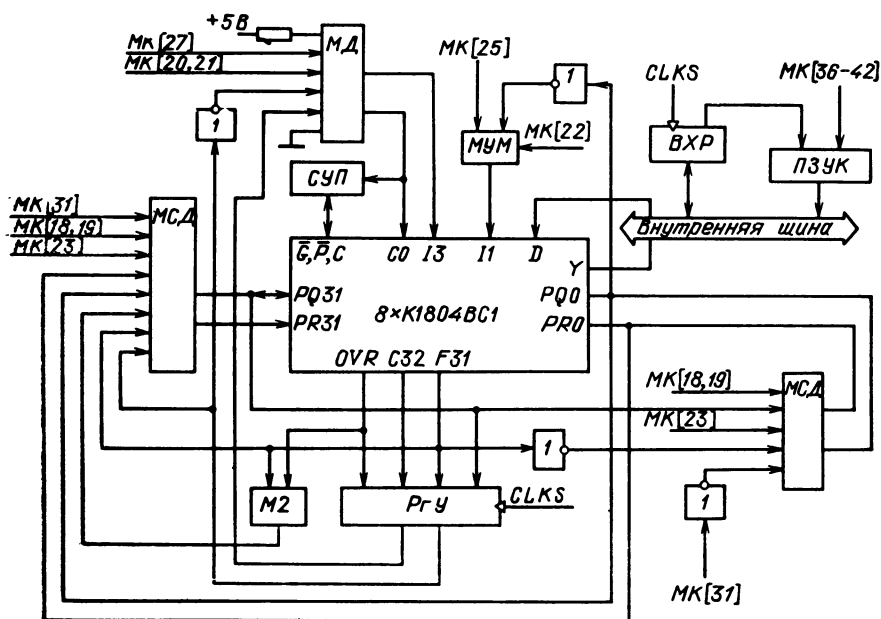


Рис. 5.20. Структурная схема блока обработки данных

цессорные секции поступает один и тот же управляющий код: на первом цикле он предписывает чтение соответствующего регистра МПС на шину с выходов Y и запись (по фронту синхросигнала $CLKS$) в регистр ВХР, а на втором — чтение регистра ВХР и запись в РЗУ других секций. Регистр ВХР, построенный на двух микросхемах К1804ИР2, также используется для выдачи байта результата через ШФИ на системную шину И-41. Кроме того, он служит регистром адреса ПЗУ коэффициентов, реализованного на одной БИС М556РТ7 с тремя состояниями на выходе. Полный 11-разрядный адрес ПЗУ коэффициентов образуется конкатенацией выходов регистра ВХР и разрядов 36—42 микрокоманды. В указанном ПЗУ хранятся коэффициенты интерполяции, используемые при вычислении трансцендентных функций.

Для уменьшения задержек распространения сигналов при арифметических операциях в БОД включена микросхема ускоренного переноса К589ИКОЗ. Идентичную функцию можно реализовать и на трех микросхемах К1804ВР1.

Для организации в БОД разнообразных сдвигов вправо/влево используются мультиплексоры сдвигов (МСД) на основе микросхем К555КП12 с тремя состояниями на выходе. Сдвиговые цепи и реализуемые в ФР варианты сдвигов иллюстрируются рис. 5.20 и 5.21. Предусмотрены восемь типов сдвигов: четыре варианта сдвигов вправо (типы 0—3) и четыре — влево (типы 4—7). Направление сдвига определяется значением управляющего сигнала на входе 17 (разряд МК [31]) МПС. При сдвигах вправо $17 = 0$, выходы МСД соединяются со входами $PQ31$ и $PR31$ МПС. Второй МСД в это время отключен, ибо на его входы разрешения поступает инвертированный сигнал $\overline{17} = 1$. При сдвигах влево мультиплексоры сдвигов меняются ролями, так как $17 = 1$, $\overline{17} = 0$. Разряды 18 и 19 микрокоманды определяют комбинацию сигналов, передаваемых на входы $PQ31$, $PR31$ или $PQ0$, $PR0$.

Для эффективной (в смысле требуемого числа тактов) реализации в ФР операций умножения и деления целых чисел со знаком в дополнительном коде, а также мантисс чисел с плавающей запятой использованы мультиплексоры умножения (МУМ) и деления (МД), построенные на микросхемах К531КП2.

При выполнении операции умножения 32-разрядных целых чисел с фиксированной запятой множитель загружается в регистр Q МПС, а множимое — в один из регистров РЗУ, например в $R1$. Перед началом

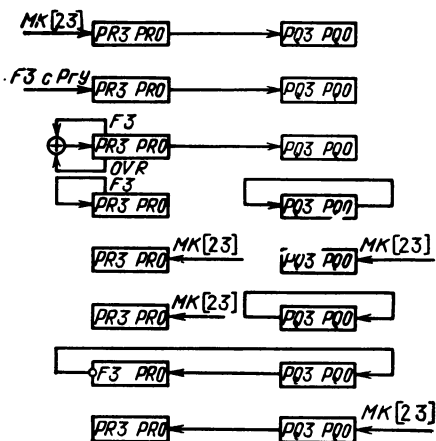


Рис. 5.21. Типы сдвигов, реализуемых в блоке обработки данных

умножения исходная сумма частичных произведений приравнивается нулю. Накопление результата происходит в регистре $R0$. Использован алгоритм умножения со сдвигом множителя и суммы частичных произведений вправо при неподвижном множимом. В цикле умножения одна и те же микрокоманда выполняется 31 раз, причем она предписывает выполнение сдвига вправо (тип 2, рис. 5.21) как множителя в регистре Q , так и суммы частичных произведений в $R0$. Помимо этого на вход $I1$ МПС через МУМ подается инвертированный сигнал с выхода $PQ0$ (такой выбор обеспечивается разрядом 22 микрокоманды). Поэтому в каждом цикле умножения на входе $I1$ присутствует выдвинутый младший разряд множителя в инверсном виде. Если выдвинутый разряд множителя равен нулю, то в качестве источников операндов АЛУ по входам $I0 - I2$ выбирается пара «0, B » и в АЛУ МПС выполняется операция сложения $0 + R0$. Если же выдвинутый разряд множителя равен единице, то будет выбрана пара источников « A , B » и на выходе АЛУ получится сумма $R1 + R0$. Эта новая сумма частичных произведений заносится в регистр $R0$ со сдвигом вправо. Одновременно сдвигается и регистр Q , вследствие чего в его старший разряд попадает младший разряд полученной суммы. На последнем, 32-м такте производится коррекция произведения, так как исходные числа были представлены в дополнительном коде. Если выдвинутый знаковый разряд множителя равен «0», то выполняется операция вычитания $R0 - 0$ и умножение завершается последним сдвигом вправо. В противном случае (когда множитель отрицательный) имеет место операция $R0 - R1$ и осуществляется запись сдвинутого вправо результата. В итоге в регистре Q окажется младшая часть произведения, а в $R0$ — старшая.

При выполнении деления без восстановления остатка под управлением 20-го и 21-го разрядов микрокоманды на входы $C0$ и $I3$ в текущем цикле подается в инверсном виде знаковый разряд $F31$ остатка, зафиксированный в регистре условий на предыдущем такте. В зависимости от его значения в АЛУ производится операция сложения или вычитания. При этом в цикле деления используется сдвиг вправо типа 6, а при восстановлении остатка — сдвиг вправо типа 1 (см. рис. 5.21).

Алгоритм деления, ориентированный на использование в устройствах, реализующих на БИС K1804BC1, включает следующие шаги:

1. Запомнить знак делителя и образовать модуль, если делитель отрицательный.
2. Запомнить знак делимого и образовать модуль, если делимое отрицательно.
3. Младшую часть делимого загрузить в регистр Q .
4. Старшую часть делимого загрузить в один из регистров, например в $R0$.
5. Сдвиг делимого влево. При этом выход $PQ3$ МПС подключается по входу $PR0$, а на вход $PQ0$ подается лог. 0.
6. Если делимое отрицательно, то останов по переполнению, так как значение частного превышает допустимый диапазон; иначе — перейти к следующему шагу.
7. Пробное вычитание делителя из старшей части делимого.
8. Если остаток отрицателен, перейти к следующему шагу, иначе — останов по переполнению.

9. Сдвиг влево остатка. При этом выход $PQ3$ подключается ко входу $PR0$, а на вход $PQ0$ подается инвертированный знак остатка.

10. Прибавить делитель к остатку, если остаток отрицательный. В противном случае вычесть делитель из остатка. Сдвиг остатка влево аналогично шагу 9.

11. Если получены все цифры частного, перейти к шагу 12, иначе—повторить шаг 10.

12. Сдвиг остатка вправо (частное не сдвигать). Если остаток отрицательный, восстановить его сложением с делителем.

13. Остатку присвоить знак делимого. Преобразовать в дополнительный код (ДК) при отрицательном знаке.

14. Сложить по мод 2 знаки делителя и делимого. Полученное значение присвоить знаку частного и преобразовать в ДК, если знак отрицательный.

В результате вычислений по приведенному алгоритму в регистре Q окажется частное, а в $R0$ — остаток. В ФР по данному алгоритму реализованы операции деления с одинарной и двойной точностью. Причем действия, производимые на шаге 10, обеспечиваются аппаратной модификацией линии $I3$ управления МПС и линии входного переноса $C0$. Об этом подробно говорилось при изложении вопросов схемной реализации БОД.

Итак, применение вспомогательных цепей для организации сдвигов, операций умножения и деления позволяет в течение одного микрокомандного цикла анализировать соответствующие разряды и в зависимости от их значений выполнять в МПС ту или иную функцию с записью в регистры МПС полученного результата. Благодаря такой гибкости управления ходом вычислений сокращается время выполнения операций умножения и деления по сравнению с тривиальной микропрограммной реализацией. Эти же цепи используются и в различных операциях с плавающей точкой. В то время, как операции умножения и деления не выполняются, на входы $I1$ и $I3$ поступают 25-й и 27-й разряды микрокоманды, а на вход $C0$ через мультиплексор можно подавать сигналы лог. 0, лог. 1 или сигнал переноса $C32$ с выхода регистра условий.

Помимо указанных на рис. 5.20 признаков в регистр условий заносятся: признак Z равенства нулю результата на выходе АЛУ, значение разряда на выходе $PQ23$ (второй байт МПС) и разряд $F23$, т. е. знак мантиссы для операций с плавающей точкой.

В процессе обработки данных в операционном блоке для хранения исходных операндов, промежуточных и окончательных результатов используется регистровое запоминающее устройство МПС. Ранее отмечалось, что ФР имеет стек-ориентированную организацию. В связи с этим массив ячеек РЗУ разделен на две части: восемь ячеек РЗУ с младшими адресами образуют восьмьюуровневый стек ФР, а оставшиеся ячейки отведены под рабочие регистры $R0$ — $R7$ с произвольным доступом. Для адресации РЗУ по каналам A и B использован коммутатор адреса на двух микросхемах К531КП11П (рис. 5.22). Выбор рабочего регистра осуществляется разрядами 10—12 и 14—16 микрокоманды, адресующими РЗУ по каналам A и B соответственно. При этом разряды 13 и 17 должны иметь нулевое значение. Если же указанные разряды имеют единичное значение, то в РЗУ выбирается определенная ячейка стека ФР. Адресация элементов стека производится указателем стека, построенным на счетчике К531ИЕ17П. Перед началом вычис-

лений в БОД состояние указателя стека соответствует адресу верхушки стека. В ходе вычислений значение указателя стека может изменяться, но после их завершения он обязательно указывает на верхушку стека, так как в нее помещается окончательный результат. Операции *PUSH* и *POP*, производимые в стеке ФР, можно рассматривать как процесс обмена информацией между его регистрами. Физически проталкивание при записи в верхушку стека и выталкивание ее при чтении выполняются сменой верхушки, для чего значение указателя стека соответственно увеличивается или уменьшается на единицу. Выполнение операций *PUSH* и *POP* поясняется временными диаграммами на рис. 5.23 и 5.24. Загрузка данных с проталкиванием осуществляется за два цикла. На первом из них разряд 8 микрокоманды (см. рис. 5.22) принимает значение «0», разрешая тем самым тактирование указателя стека. Единичное значение разряда 9 микрокоманды предписывает инкрементирование указателя стека, производимое по фронту тактового сигнала *CLKS*. На втором такте разряд 17 микрокоманды устанавливается в состояние лог. 1 для передачи адреса новой верхушки на входы *B3* — *B0* МПС. Одновременно дается разрешение синхронизации секции, поэтому происходит занесение информации в новую верхушку стека по низкому уровню сигнала *CLKS*. При этом значение указателя стека не изменяется, ибо управляющий сигнал *ENSP* пассивен, т. е. на следующем такте возможна запись в прежнюю верхушку без проталкивания. Чтение содержимого верхушки стека по каналам *A* и *B* МПС с выталкиванием сопровождается декрементиро-

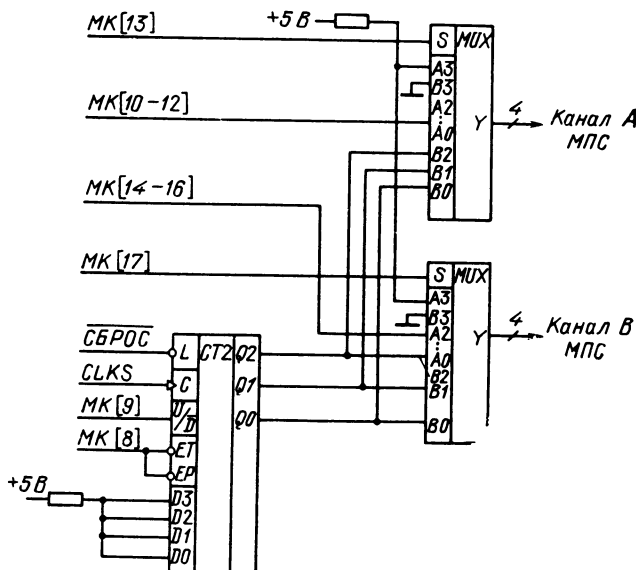


Рис. 5.22. Функциональная схема указателя стека

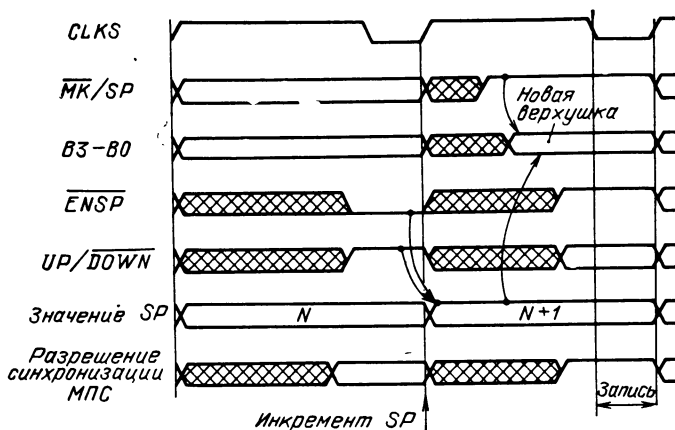


Рис. 5.23. Временные диаграммы записи операнда в стек с операцией PUSH

ванием указателя стека. С этой целью управляющие сигналы \overline{ENSP} и $UP/DOWN$ в цикле чтения имеют низкий уровень. Если необходимо считывать содержимое прежней верхушки без операции POP, сигнал \overline{ENSP} должен иметь единичное значение.

В операциях манипуляции стек-операндами используется «вращение» указателя стека. Последнее обстоятельство позволило в среднем вдвое уменьшить время выполнения команд этой группы. Алгоритмы реализации команд манипуляции приведены в § 5.5.

Интерфейс функционального расширителя. В § 5.3 указывалось, что ФР ориентирован на системный интерфейс И-41, с организацией которого можно ознакомиться в [27]. Интерфейс ФР обеспечивает взаимодействие с базовой микроЭВМ и представляет собой совокупность аппаратной и микропрограммной частей. Аппаратная часть включает 8-разрядный шинный формирователь с инверсией (ШФИ), регистр слова состояния и интерфейсную логику. Реализован ШФИ на двух

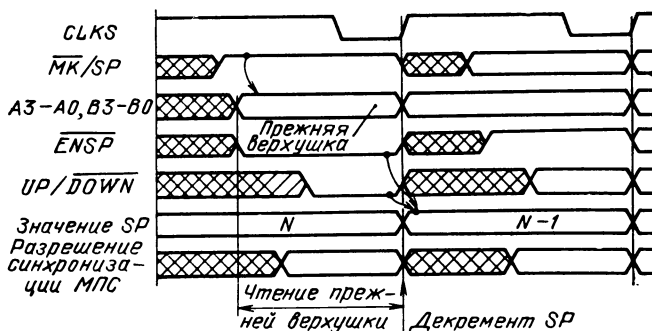
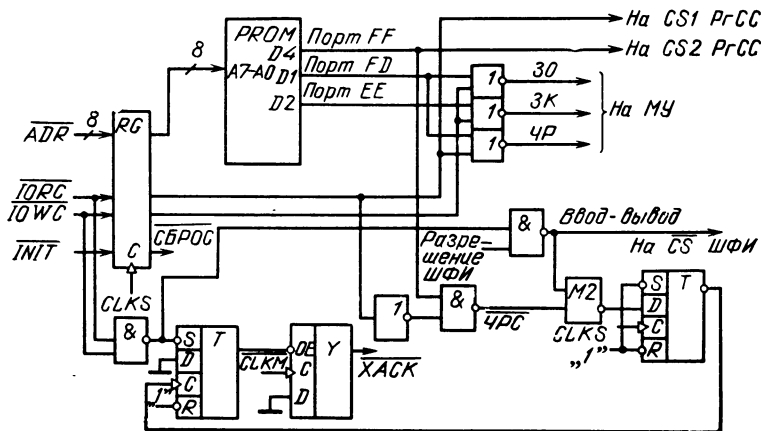


Рис. 5.24. Временные диаграммы чтения операнда из стека с операцией POP

микросхемах К589АП26 с тристабильными выходами и служит для двунаправленной передачи данных с шины И-41 на внутреннюю общую шину ФР и в обратном направлении. Интерфейсная логика, функциональная схема которой представлена на рис. 5.25, осуществляет дешифрацию адреса, поступающего по системной шине адреса $\overline{ADR7} - \overline{ADR0}$, а также анализирует сигналы на шине управления И-41 и вырабатывает ответные и внутренние сигналы, требуемые для функционирования ФР. Адрес с линией $\overline{ADR7} - \overline{ADR0}$ и управляющие сигналы \overline{IORC} , \overline{IOWC} , \overline{INIT} интерфейса И-41 фиксируются в регистре RG , построенном на двух микросхемах К555ТМ9, по фронту синхроимпульса $CLKS$. При включении питания вычислительного комплекса, после поступления сигнала системного сброса \overline{INIT} , на выходе регистра RG появится сигнал $\overline{СБРОС}$, устанавливающий регистр $PrCC$, указатель стека и $PrMK$ в исходное состояние. Благодаря этому БМУ начинает работу с микрокоманды, размещенной в МПП по нулевому адресу.



радна соответственно на шину адреса и шину данных системного интерфейса И-41. После фиксации в регистре RG инверсного адреса порта FD на выходе $D1$ дешифратора адреса появится лог. 0. Далее процессор предписывает операцию записи в порт данных ФР, устанавливая сигнал записи, заносимый по фронту импульса $CLKS$ в регистр RG . Это приводит к появлению на выходе соответствующего элемента ИЛИ — НЕ активного сигнала ZO , который поступает на мультиплексор условий БМУ и сигнализирует о необходимости записи байта операнда в ФР. Обнаружив данный запрос, БМУ выполняет переход к микропрограмме записи операнда в верхушку стека. В ответ на сигнал ZO из 7-го разряда концептуального формата микрокоманды поступает сигнал лог. 1, разрешающий работу ШФИ. На выходе соответствующего элемента И — НЕ установится сигнал $\overline{CS} = 0$ выбора ШФИ. При этом в качестве источника информации выбирается шина данных интерфейса И-41 и в течение одного микрокомандного цикла первый байт операнда загружается в четверть верхушки стека, т. е. в младший байт МПС. В начале следующего цикла на тристабильном выходе регистра $K1804IP1$ устанавливается ответный сигнал низкого уровня \overline{XACK} , уведомляющий об окончании записи байта операнда в порт данных ФР. Получив этот ответный сигнал, процессор снимает сигнал записи $IOWS$, что приводит к снятию сигналов ZO , \overline{CS} ШФИ и \overline{XACK} . Эта процедура повторяется четырехкратно для записи 32-разрядного операнда, после чего на порт команд ФР можно подавать код требуемой команды.

При записи кода команды в ФР процессор производит те же действия, что и при записи байта операнда в верхушку стека. Отличие только в значении адреса FE порта. В интерфейсной логике вырабатывается сигнал высокого уровня ZK , идентифицирующий цикл записи команды. Этот запрос обнаруживается в БМУ микрокомандой условного перехода (CJP), и осуществляется переход к микрокоманде условного перехода по векторному адресу (CJV). В этом микрокомандном цикле дается разрешение ШФИ и 6-разрядный код команды передается на внутреннюю общую шину ФР (рис. 5.26), а затем поступает на входы непосредственного адреса $D5 — D0$ БИС $K1804BY4$ через тристабильный повторитель общей шины (ТСП). Реализован ТСП на микросхеме $K155ЛП10$ и управляется выходом \overline{VE} микросхемы $K1804BY4$. Поскольку выполняется инструкция CJV , то сигнал $\overline{VE} = 0$ активен. Выходы $PgMK$, откуда на входы $D5 — D0$ подается 6-разрядная часть адреса микрокоманды, отключены из-за отсутствия сигнала \overline{PE} разрешения выборки адреса из $PgMK$. На входы $D9 — D6$ из $PgMK$ подаются единичные значения старших разрядов адреса МПП. Следовательно, происходит передача управления на микрокоманды, находящиеся в последних 128 ячейках МПП (вспомним, что младшим разрядом адреса памяти микропрограмм является сигнал $CLKM$), т. е. в область 64 стартовых микрокоманд. Итак, на входах $D9 — D0$ присутствует адрес $1111 <\text{код команды}>$. Можно было бы выполнить безу-

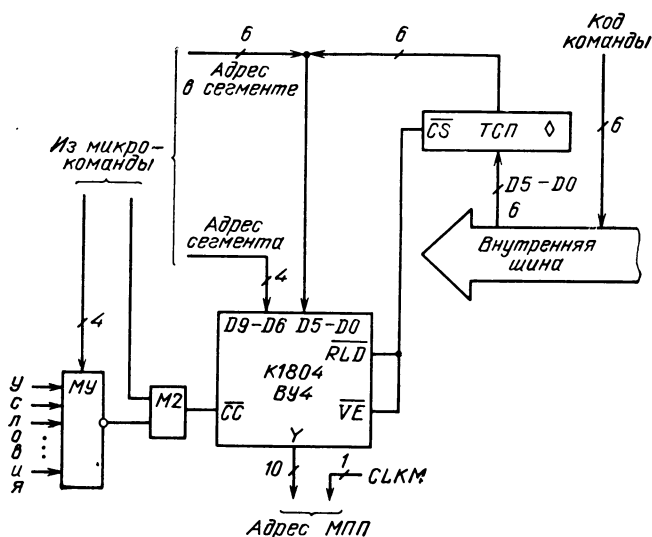


Рис. 5.26. Схемы, участвующие в дешифрации команды

словный переход по этому адресу инструкцией *CJV*. Однако в этом случае все 64 стартовые микрокоманды должны осуществлять завершение цикла асинхронного обмена с процессором, а затем передавать управление соответствующим микропрограммам, что неэффективно. В связи с этим принято решение запоминать в регистре адреса РгА/Сч БИС К1804ВУ4 информацию со входов $D9 - D0$ для завершения обмена с процессором и последующего перехода по этому адресу. Инструкцией *CJV* такая загрузка не обеспечивается, поэтому сигнал разрешения необходимо подавать на вход \overline{RLD} для принудительной загрузки адреса в регистр. С этой целью обычно используется дополнительный разряд микрокоманды (см., например, [4]). В разработанном ФР применено оригинальное схемное решение с подключением выхода \overline{VE} ко входу \overline{RLD} (см. рис. 5.26). Поэтому инструкция *CJV* используется подобно инструкции *LDCT* с тем отличием, что в регистр адреса информация заносится с выходов ТСП, а не из Рг МК. Ниже будет показано, что такое решение очень удобно и для выполнения некоторых операций с плавающей точкой.

Таким образом, по инструкции *CJV* происходит запись адреса перехода в РгА/Сч микросхемы К1804ВУ4. Затем циклически выполняется условная микрокоманда *JRP* для завершения протокола обмена с процессором. Как только сигнал ЗК будет снят, произойдет переход по содержимому РгА/Сч, т. е. переход по коду команды в область стартовых микрокоманд, с которых начинаются микропрограммы выполнения всех команд ФР.

Исполнение команды ФР завершается записью признака полученного результата в $PgCC$. При этом загружаемая информация о состоянии ФР поступает с разрядов 36—41 микрокоманды и с соответствующих выходов МПС. Чтобы получить информацию о состоянии ФР, процессор должен считывать содержимое $PgCC$. Процессор обращается к порту FF и активизирует сигнал чтения \overline{TORC} , что приводит к выработке в интерфейсной логике сигналов $CS1$ и $CS2$, включающих $PgCC$ в режим передачи информации, а также сигнала низкого уровня ЧРС, по которому выдается ответный сигнал $XACK$. Прочитав слово состояния, процессор снимает сигнал \overline{TORC} , а в интерфейсной логике снимаются сигналы ЧРС и $XACK$. Заметим, что чтение $PgCC$ возможно в любой момент времени и не влияет на процессы, происходящие в ФР.

Обнаружив готовность ФР, процессор может прочитать результат из верхушки стека 4-кратным обращением к порту данных. Считывание байта результата происходит следующим образом. В ответ на обращение к порту FD и установление сигнала чтения \overline{TORC} в интерфейсной логике вырабатывается сигнал чтения результата. Блок микропрограммного управления получает его и осуществляет переход к микропрограмме чтения верхушки стека. На первом такте байт результата переписывается из верхушки в регистр ВХР. На втором такте дается разрешение ШФИ, причем ШФИ включается на передачу информации на шину данных $\overline{DAT7} - \overline{DAT0}$. Таким образом, байт результата поступает в процессор. Затем ФР выставляет сигнал $XACK$, что вызывает последовательное снятие сигналов \overline{TORC} , ЧР, \overline{CS} ШФИ и $XACK$.

Микропрограммы арифметики с плавающей точкой работают с 32-разрядным форматом данных, в старшем байте которого содержится разряд знака S числа и семь разрядов порядка E в дополнительном коде (диапазон представления — $64 \leq E \leq 63$), в остальных трех байтах размещается двоично-нормализованная дробная мантисса M . Результат любой операции с ПТ выдается в нормализованном виде. В случае нулевой мантиссы все разряды числа устанавливаются в нуль. При переполнении порядка результат будет содержать верный знак, верную мантиссу и порядок, смещенный на — 128. При антипереполнении порядка знак указанного смещения будет положительным. Попытка деления на нуль приведет к выдаче делимого в качестве результата.

Для внутреннего представления порядка в операционном блоке ФР используется модифицированный код. Основное преимущество такого решения заключается в удобстве работы с порядками, выравненными по границам байта. Это касается считывания признаков результата и маскирования.

Общими характерными этапами арифметической обработки чисел с ПТ являются:

1. Предварительное «расщепление» каждого операнда с засылкой знаков, порядков и мантисс в отдельные рабочие регистры; при этом

не нарушается привязка полей числа с ПТ к разрядности полного формата.

2. Раздельная (по месту и времени) обработка знаков, порядков и мантисс в соответствии с заданным алгоритмом.

3. Окончательная «сборка» результата.

Значительное удобство при обработке чисел с ПТ представляют три рабочие константы, которые постоянно хранятся в регистрах $R5—R7$ МПС. Старший байт первой константы $E8000000_{16}$ можно арифметически интерпретировать как число -24 в дополнительном коде. Учитывая, что порядок в модифицированном коде также занимает весь старший байт, а 24 — разрядность мантиссы, можно использовать число -24 как константу сравнения. Например, при сложении с этой константой легко определить, произойдет ли установка в «0» денормализуемой мантиссы. Если это так, то сумма будет заведомо равна большему из операндов и продолжать вычисления не имеет смысла.

Старший байт второй константы $FF000000_{16}$ можно арифметически интерпретировать как число -1 в дополнительном коде. Суммирование или вычитание этой константы из порядка вызывает соответственно его декремент или инкремент. Константу можно рассматривать и как логическую величину. С ее помощью легко маскировать поле мантиссы либо поле модифицированного порядка. С помощью третьей константы 80000000_{16} , которая содержит только одну единицу в старшем разряде, можно маскировать поле знака, а также инвертировать знак при выполнении операции ИСКЛЮЧАЮЩЕЕ ИЛИ.

Функциональный расширитель реализует широкий набор трансцендентных функций от аргумента, представленного в форме с ПТ. Шесть функций, выбранных в качестве базовых (\sqrt{x} , e^x , $\ln x$, $\sin x$, $\cos x$, $\operatorname{arctg} x$), вычисляются по таблично-многочленному методу менее, чем за 100 мкс. Остальные функции вычисляются микропрограммно с использованием базовых. Собственно вычислению базовой трансцендентной функции в общем случае предшествует предварительная обработка аргумента, которая заключается в его преобразовании по формуле приведения к интервалу $[0,1)$. После вычисления функции от приведенного аргумента определяется искомое значение функции от исходного аргумента в форме с ПТ, что связано с повторным использованием формулы приведения. Таким образом, реализация формул приведения является практически единственной характерной особенностью, которая отличает алгоритм вычисления различных базовых функций.

Применение таблично-многочленного метода вычисления в рассматриваемом ФР предполагает равномерное разбиение интервала $[0,1)$ изменения приведенного аргумента x на подынтервалы и определение значения функции f на соответствующем подынтервале с помощью многочленного приближения Фурье — Чебышева [28]: $f = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4$. Вид данного многочлена четвертой

степени, вычисляемого по стандартной схеме Горнера, одинаков для всех базовых функций. Коэффициенты многочлена для каждого подынтервала каждой функции вычислены заранее и хранятся в ПЗУК емкостью $2K \times 8$ бит.

В процессе вычисления адрес ПЗУК формируется из двух частей с последующей их конкатенацией. Первая часть (переменная) задается полями микрокоманды и содержит 5-разрядный идентификатор считываемого коэффициента (совокупно определяет тип реализуемой функции и номер коэффициента, совпадающий с показателем степени приведенного аргумента x), а также 2-разрядный идентификатор номера байта. Указанная часть адреса поступает из РгМК непосредственно на адресные входы ПЗУК и обновляется при каждом считывании. Вторая часть (постоянная) представляет собой четыре старших разряда приведенного аргумента и задает, таким образом, один из шестнадцати подынтервалов разбиения. Эта вычисляемая часть адреса пересылается из второго байта МПС в ВХР и хранится там на протяжении выборки всех байтов всех коэффициентов многочленного приближения данной функции.

С целью повышения быстродействия в пределах этапа вычисления многочлена приведенный аргумент и коэффициенты обрабатываются по типу 32-разрядных данных с фиксированной точкой. Под хранение коэффициентов каждой из базовых функций отводится 80 слов (5 коэффициентов $\times 16$ подынтервалов). Заметим, что для функций $\sin x$ и $\cos x$ имеется единый набор из 80 коэффициентов, поскольку эти функции легко сводимы одна к другой, что и реализуется на этапе их предварительной обработки. Исключение составляет функция e^x . Для последней кроме основного набора коэффициентов в ПЗУК хранятся еще 64 слова в форме с ПТ, содержащих предварительно вычисленные значения функции e^x от целой части аргумента (в допустимом для данной функции диапазоне x от 0 до 63). На завершающем этапе обработки результат полиномиального приближения умножается на соответствующее целочисленное значение. Таким образом, общий объем памяти коэффициентов для шести базовых функций составляет 464 32-разрядных слова. Оставшиеся 48 слов ПЗУК разервируются под различные рабочие константы.

В данном ФР очень просто реализовать попарный контроль переходов в БМУ по методу, описанному в § 3.2. Для кодирования ключей микрокоманд можно использовать восемь незадействованных разрядов четной части микрокоманды, а схема контроля потребует всего две дополнительные микросхемы: К531ТМ9П и К531СП1П. Таким образом, на печатной плате ФР в конструктиве *E2* всего размещено 52 микросхемы. Потребляемый ток — около 4 А от штатного источника +5 В микроЭВМ СМ-1800.

Некоторые схемные решения, принятые в ФР, продиктованы аппаратными ограничениями, обусловленными форматом платы *E2*. В случае незначительного «смягчения» ограничений (например, при использовании платы микроЭВМ «Электроника-60») в ФР целесооб-

разно реализовать БМУ с одинарной выборкой микрокоманд и увеличить разрядность внутренней общей шины данных до шестнадцати. Это позволит вдвое сократить время выполнения команд. Очевидно, потребуются внести соответствующие модификации в микропрограммное обеспечение ФР.

ЗАКЛЮЧЕНИЕ

В условиях конкуренции с развитыми однокристалльными микропроцессорами рынок сбыта биполярных микропрограммируемых БИС поддерживается благодаря их высокому быстродействию (до 10 млн. операций в секунду) и возможностям более глубокой специализации архитектуры устройств для конкретных применений. Так, однокристалльный микропроцессор K1810BM86 имеет производительность 1,2 млн.команд регистр — регистр в режиме предельного совмещения. Микропрограммируемая БИС K1804BM1 (см. § 1.1) в таком же режиме обеспечивает производительность, всемерно большую. Разумеется, неправомерно сопоставлять упомянутые БИС только по приведенной оценке и вряд ли можно утверждать, что одна из них в целом лучше другой. Каждая будет иметь своего потребителя не только в настоящем, но и в обозримом будущем.

Дальнейшее развитие отечественных и зарубежных комплектов биполярных БИС осуществляется по двум направлениям. Первое заключается в улучшении временных и энергетических параметров серийно выпускаемых микросхем без изменения их архитектуры за счет применения более совершенных технологических процессов. Второе направление состоит в разработке комплектов микропрограммируемых БИС, ориентированных на эффективную реализацию набора функций, характерных для специальных областей применения. Среди них: цифровая обработка сигналов; вычислительные операции над числами в стандартных форматах с плавающей точкой; логические операции в контроллерах быстродействующих периферийных устройств. И пока продолжается совершенствование элементной базы, используемой в рамках принципа микропрограммного управления, не стоит снимать с повестки дня те общие вопросы проектирования [29], которые нашли отражение в настоящей книге.

СПИСОК ЛИТЕРАТУРЫ

1. **Проектирование** цифровых систем на комплектах микропрограммируемых БИС/С. С. Булгаков, В.М. Мещеряков, В.В. Новоселов, Л. А. Шумилов: Под ред. В. Г. Колесникова. — М.: Радио и связь, 1984. — 240 с.
2. **Березенко А.И., Корягин Л. Н., Назарьян А.Р.** Микропроцессорные комплекты повышенного быстродействия. — М.: Радио и связь, 1981. — 168 с.
3. **Микропроцессорные** комплекты БИС на основе интегральной инжекционной логики/В. С. Борисов, Ф. С. Власов, Э. П. Калошкин и др.: Под ред. Э. П. Калошкина. — М.: Радио и связь, 1984. — 248 с.
4. **Мик Дж., Брик Дж.** Проектирование микропроцессорных устройств с разрядно-модульной организацией. В 2-х книгах: Пер. с англ. — М.: Мир, 1984. — 478 с.
5. **Проектирование** микропроцессорных систем / Зарубежная электронная техника. — 1980. — Вып. 11. — С. 99.
6. **Майоров С. А., Новиков Г. И.** Структура электронных вычислительных машин. — Л.: Машиностроение, 1979, с. 183—185.
7. **Кристофидес Н.** Теория графов: Алгоритмический подход Пер. с англ. — М.: Мир, 1978. — 432 с.
8. **Майника Э.** Алгоритмы оптимизации на сетях и графах: Пер. с англ. — М.: Мир, 1981. — 323 с.
9. **Scheher H., Anlauff H.** Avoiding Gaps in a Three-level Pipelined Micro-control Unit // Microprocess. and Microprogramm. — 1982. — Vol. 9, № 3. — P. 155—160.
10. **Baron M.** Control Your Next Pipeline Design with a Microprogram Sequencer. — EDN. — 1980. — Vol. 25, № 6. — P. 157—162.
11. **Namjoo M.** Design of Concurrently Testable Microprogrammed Control Units // SIGMICRO Newslett. — 1982. — Vol. 13, № 4. — P. 173—180.
12. **Календарев А. С., Новоселов В. В.** Функциональное диагностирование микропрограммируемых микропроцессоров // Автоматизация проектирования микропроцессорных устройств: Сб. статей. — ИТК АН БССР. — 1986. — С. 68—73.
13. **Гордон Г., Надиг Х.** Локализация неисправностей в микропроцессорных системах при помощи шестнадцатеричных ключевых кодов / Электроника, — 1977. — № 5. — С. 23—33.
14. **Каган Б. М., Мкртумян И. Б.** Основы эксплуатации ЭВМ Под ред. Б.М. Кагана. — М.: Энергоатомиздат, 1983. — 376 с.
15. **Daniels S.F.** A Concurrent Test Technique for Standard Microprocessors //Proc. IEEE COMPCON/83. — 1983. — P. 389—394.
16. **Согомонян Е.С. Слабаков Е.В.** Самопроверяемые устройства и отказоустойчивые системы. — М.: Радио и связь, 1989. — 208 с.
17. **Duran J., Tulin E.M.** A Desing Approach for a Microprogrammed Control Unit with Built — in Self Test // Proc. IEEE MICRO.16. — 1983. — P. 55—60.
18. **Flaherty T. J.** Building Blocks Stack up to High Performance Computer Design. — 1985. — N 2. — P. 161—167.
19. **Разработка и отладка микропрограммного обеспечения цифровых систем на основе секционированных микропроцессоров/А.Г. Алексенко, А. В. Гапо-**

- ненко, А. Д. Иванников, И. Д. Курилов /Микропроцессорные средства и системы. — 1986. — № 1. — С. 37—43.
20. **Микропроцессоры:** системы программирования и отладки/В.А. Мясников М. Б. Игнатьев, А. А. Кочкин и др.; Под ред. В. А. Мясникова, М. Б. Игнатьева. — М.: Энергоатомиздат, 1985. — 272 с.
 21. **Prosser F., Winkel D.** The Logic Engine Development System Support for Microprogrammed Bit-slice Development // SIGMICRO Newslett. — 1983. — Vol. 14, № 4. — P. 84—91.
 22. **Настраиваемый** инструментальный комплекс для разработки систем на секционированных микропроцессорах / О. И. Семенов, Л. А. Гриншпан, Е. М. Злотник и др. // Управляющие системы и машины. — 1984. — № 2. — С. 36—39.
 23. **AMDASM** Reference Manual Advanced Micro Devices, Inc., Sunnyvale, 1978. — 36 p.
 24. **Сервисные** средства для отладки микропрограммируемых микропроцессоров/В.Б. Головлев, В. П. Емелин, В. В. Новоселов, В. А. Фукс // Роль микропроцессоров и микроЭВМ в автоматизации производственных процессов при реализации программы «Интенсификация-90»: Сб. статей: — ЛДНТП. — 1986. — С. 41—44.
 25. **iAPX86, 88, 186 and 188** User's Manual: Programmer's Reference. — INTEL Corp. Literature Department, Santa Clara, 1985.
 26. **Furt B., Lee P.** An Efficient Software Driver for Am 9511 Arithmetic Processor Implementation // IEEE Micro. — June, 1984. — Vol.4. — P. 7—19.
 27. **МикроЭВМ СМ-1800:** Архитектура, программирование, применение/А.В. Гиглавы, Н. Д. Кабанов, Н. Л. Прохоров, А.Н. Шкамарда. — М.: Финансы и статистика, 1984.—136 с.
 28. **Балашов Е. П., Пузанков Д. В.** Проектирование информационно-управляющих систем. — М.: Радио и связь, 1987.—256 с.
 29. **Новоселов В. В.** Биполярные микропрограммируемые микропроцессоры. архитектура, проектирование, применение / Изв. вузов СССР. Приборостроение. — 1987.—№ 10.—С.40—45

ОГЛАВЛЕНИЕ

Предисловие	3
Введение	4

Глава 1. ЭЛЕМЕНТНАЯ БАЗА

1.1. Шестнадцатиразрядный процессорный элемент	9
1.2. Схема адресной обработки	39
1.3. Контроллер прерываний	47
1.4. Счетчики адресов прямого доступа к памяти	54
1.5. Микропрограммируемый тактовый генератор	68
1.6. Регистры и приемопередатчики	73

Глава 2. МИКРОПРОГРАММНОЕ УПРАВЛЕНИЕ: СХЕМНЫЕ РЕШЕНИЯ

2.1. Типовые схемы микропрограммного управления	80
2.2. Синхронизация	98
2.3. Способы повышения быстродействия	103
2.4. Техника предположений	114
2.5. Способы сокращения аппаратных затрат	125

Глава 3. ФУНКЦИОНАЛЬНОЕ ДИАГНОСТИРОВАНИЕ БЛОКОВ МИКРОПРОГРАММНОГО УПРАВЛЕНИЯ

3.1. Объект и задачи диагностирования	130
3.2. Контроль попарного следования микрокоманд	134
3.3. Пофрагментный контроль микропрограмм	142

Глава 4. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ МИКРОПРОГРАММ

4.1. Организация отладочных комплексов	156
4.2. Микроассемблер. Фаза определения	164
4.3. Микроассемблер. Фаза ассемблирования	176
4.4. Трансляция и постобработка микрокода	185
4.5. Отладка микропрограмм	197

Глава 5. МИКРОПРОГРАММНЫЕ КОНТРОЛЛЕРЫ И ПРОЦЕССОРЫ

5.1. Микроконтроллеры	203
5.2. Конвейерные процессоры команд	211
5.3. Функциональные расширители	228
Заключение	252
Список литературы	253

Производственное издание

**Мещеряков Виталий Михайлович, Лобов Иван Егорович,
Глебов Сергей Савельевич, Новоселов Виктор Владимирович,
Шумилов Лев Алексеевич**

КОМПЛЕКТ БИС К1804 В ПРОЦЕССОРАХ И КОНТРОЛЛЕРАХ

Заведующий редакцией *Ю. Н. Рысев*
Редактор *М. М. Лисина*
Художественный редактор *Н. С. Шеин*
Технический редактор *З. Н. Ратникова*
Корректор *Т. В. Дземидович*

ИБ № 1795

Сдано в набор 14.11.88. Подписано в печать 10.08.89. Т-23791
Формат 60×88¹/₁₆. Бумага офсетная № 2. Гарнитура литературная. Печать офсетная.
Усл. печ. л. 15,68. Усл. кр.-отт. 15,68. Уч.-изд. л. 18,08. Тираж 16 000 экз.
Изд. № 22341 Зак. № 1762 Цена 1 р. 20 к.

Издательство «Радио и связь». 101000 Москва, Почтамт, а/я 693

Московская типография № 4 Госкомпечати СССР
Москва, И-41, Б. Переяславская, 46

